RoboEC2: A Novel Cloud Robotic System With Dynamic Network Offloading Assisted by Amazon EC2

Boyi Liu, *Member, IEEE*, Lujia Wang, *Member, IEEE*, and Ming Liu, *Senior Member, IEEE*

***Abstract*— Deep neural networks (DNNs) are increasingly utilized in robotic tasks. However, resource-constrained mobile robots often do not have sufficient onboard computing resources or power reserves to run the most accurate and state-of-the-art DNNs. Cloud robotics has the benefit of enabling robots to offload DNNs to cloud servers, which is considered a promising technology to address the issue. However, comprehensive issues exist, including flexibility, convenience, offloading policy, and especially network robustness in its implementations and deployments. Although it is essential to promote cloud robotics to be practical, a cloud robotic system that addresses these issues comprehensively has never been proposed. Accordingly, in this work, we present RoboEC2, a novel cloud robotic system with dynamic network offloading implemented assisted by Amazon EC2. To realize the goal, we present a cloud-edge cooperation framework based on ROS and Amazon Web Services (AWS) and a network offloading approach with a dynamic splitting way. RoboEC2 is capable of executing its network offloading program in any conditions, including disconnected. We model the DNN offloading problem in RoboEC2 to a specific multi-objective optimization problem and address it by proposing the Spotlight Criteria Algorithm (SCA). RoboEC2 is flexible, convenient, and robust. It is the first cloud robotic system with no constraints on time, location, or computing power. Finally, We demonstrate RoboEC2 with analyses and experiments that it performs better in comprehensive metrics compared with the state-of-the-art approach. We open-source the system at *https://github.com/RoboEC2/RoboEC2*.**

***Note to Practitioners*—RoboEC2 is a work that combines cloud computing and robotics. As the deep learning models are becoming larger, robots are becoming more and more difficult to run the state-of-the-art models locally. It has become one of the major problems in robotics. RoboEC2 was proposed to address this problem. It enables more robotics researchers to equip their robots with the power of cloud computing. To be honest, it is very difficult for us to complete this work that is a robotic system with cloud computing. We need to address a lot of difficulties such as network, the cloud platform, algorithms, robot platforms, and conduct various robotic tasks. We have spent more than one year on this system and overcome countless difficulties to complete it. All of what we do is to make robotics developer easier strengthen their robots with cloud. Whether you are an autonomous driving engineer, robotic arm developer, SLAM researcher, mobile robotics researcher, or any other developer working on robotics applications based on ROS and deep learning models, you can use RoboEC2 to make them perform better. You don't need to worry about networking, because RoboEC2 has solved it perfectly. You don't need to worry about the serious algorithms in the system, because we provide easily used interact files for you to configure. You just need to tell RoboEC2 which metrics your robotics application needs to focus on. With RoboEC2, all the robotic researchers/developers are capable of enhancing their robotic applications with cloud computing in just a few simple steps and executing them in any network conditions. So, why not?**

***Index Terms*— Cloud robotics, cloud ROS, amazon EC2, network offloading.**

Manuscript received 26 December 2022; revised 25 April 2023; accepted 4 June 2023. This article was recommended for publication by Associate Editor A. Pietrabissa and Editor G. Fortino upon evaluation of the reviewers' comments. This work was supported in part by the Guangdong Basic and Applied Basic Research Foundation under Project 2021B1515120032, in part by the Hainan Provincial Natural Science Foundation of China under Grant 723QN238 and Grant 722RC678, in part by the Jiangsu Province Science and Technology Project BZ2021056, and in part by the Project of Hetao Shenzhen-Hong Kong Science and Technology Innovation Cooperation Zone under Grant HZQB-KCZYB-2020083. *(Corresponding author: Lujia Wang.)*

Boyi Liu and Lujia Wang are with the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong, SAR (e-mail: by.liu@ieee.org; eewanglj@ust.hk).

Ming Liu is with the Robotics and Autonomous Systems Thrust, The Hong Kong University of Science and Technology (Guangzhou), Nansha, Guangzhou, Guangdong 511400, China, also with the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong, SAR, China, and also with the HKUST Shenzhen-Hong Kong Collaborative Innovation Research Institute, Futian, Shenzhen 518000, China (e-mail: eelium@ust.hk).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TASE.2023.3305522.

Digital Object Identifier 10.1109/TASE.2023.3305522

I. Introduction

DEEP learning models have been demonstrated to achieve superior performance in various robotics tasks, particularly in the areas of perception and decision-making. However, the use of these models can significantly increase the computational requirements and energy consumption of robots, leading to challenges in terms of endurance and resource constraints for autonomous mobile robots such as self-driving cars, delivery drones, and autonomous logistics vehicles. This issue, known as the compute-and-power-limited problem in mobile robotics, presents a significant challenge for the widespread adoption of robots.

Cloud robotics [12] is an effective way to address these issues, and it is an emerging technology that migrates complex computations to the cloud. Offloading deep natural networks (DNNs) to the cloud or the edge for computation effectively reduces local robot's computational stress and power consumption, which is also researched in cloud or edge

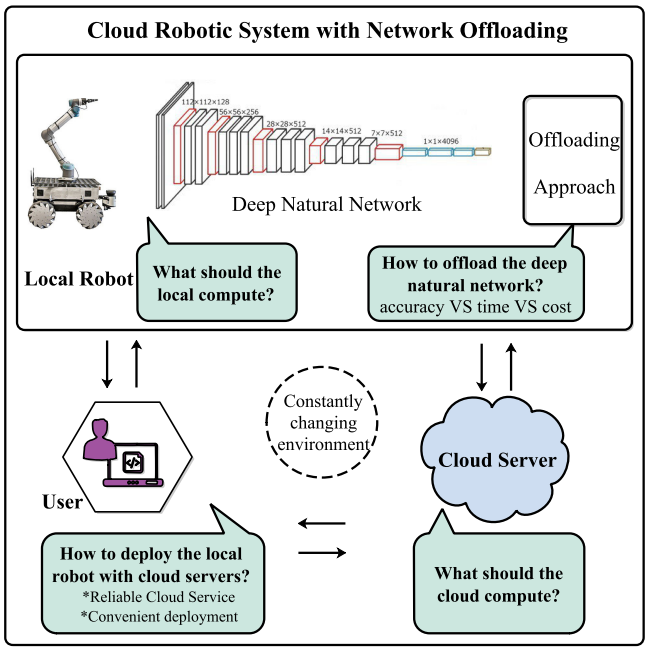


Fig. 1. For the cloud robotic system with DNNs offloading, there exist comprehensive requirements such as convenience for the user, flexible offloading policy, appropriate computation distribution between edge and clouds, and especially robustness on various conditions. A practical system has never been proposed. In this paper, we realize this by presenting RoboEC2, a novel cloud robotic system with dynamic network offloading assisted by Amazon EC2.

computing. The above approaches, referred to as the cloud robotic system with network offloading, impose a set of trade-offs that have hitherto only been marginally addressed in the literature. Specifically, while offloading DNNs to the cloud reduces the onboard computing requirements, it may result in latency that could severely degrade performance if a network is highly congested. We need to make better trade-offs in terms of accuracy, time, computational cost, etc. In addition to this, the deployment of cloud robotics is seldom mentioned in the literature. Indeed, it is impossible to build a separate cloud server for a particular robot and spend amounts of time and efforts to deploy the scheme such as the network offloading every time. Therefore, a convenient cloud robotic system with dynamic network offloading is the possible answer to the compute-and-power-limited problem in robotics. Some generalized framework as FogROS and offloading algorithms for specific robotic tasks are proposed. However, There is no dynamic offloading framework that has the ability to perform robustly for various robotic tasks in all conditions, rather than completely offloading or only optimize specific tasks. ROS based, convenient, robust and adaptive are elements that are essential for developers. The cloud robotics community looks forward to a system that have all these elements.

To address the gap, we present RoboEC2 that is a novel cloud robotic system with dynamic network offloading implemented with Amazon Elastic Compute Cloud (Amazon EC2). RoboEC2 is implemented based on ROS, the most widely used operating system in robotics. RoboEC2 realizes cloud-edge communication by building a Virtual Private Cloud (VPC) in Amazon EC2. Amazon Web Services (AWS) has ensured that the configuration of the computing power of the cloud in RoboEC2 is flexible, which totally depends on needs. Moreover, it can be connected at any time because AWS is constantly running. RoboEC2 takes a more advanced approach proposed in this paper to network offloading. The system simultaneously balances consumption, accuracy, and time, dynamically splits network layers based on current conditions. The above ensures that RoboEC2 is capable of operating in any network conditions, including disconnection, congestion, etc. It is always able to make the right decisions based on a combination of all the considerations. RoboEC2 performs better than the SOTA approach, one of the previous best system paper finalists of RSS. Last but not least, it also provides a robust and practical framework.

In our work, we have developed a novel service that combines DNN segmentation with the ROS system in order to address the specific needs of robot tasks. Our service is designed to be used by users in a variety of conditions, and we have tested it extensively in order to ensure that it is reliable, convenient, and practical. One of the key contributions of our work is the integration of DNN segmentation into the ROS system, which enables the use of this technique for a wide range of robot tasks. This integration is important because it allows for the seamless integration of DNN segmentation with other robot capabilities and enables the use of this technique in real-world applications. Another important contribution of our work is the focus on practicality and usability. Unlike many other DNN segmentation approaches, which are focused on theoretical contributions or the optimization of specific performance metrics, our service is designed to be used reliably and conveniently in various conditions. This focus on practicality and usability sets our work apart from other DNN segmentation approaches and makes it a valuable contribution to the field.

It is important to note that the goal of a “service” is different from the goal of an “algorithm”. While there have been numerous deep neural network (DNN) segmentation algorithms proposed in the literature, to the best of our knowledge, there has not yet been a DNN segmentation embedded robotic system that is able to operate at the level of a “service”. In other words, a service must be able to function robustly and conveniently under a variety of conditions, while an algorithm is typically optimized for a specific set of conditions as specified in the corresponding research paper. This is one of the key distinctions of our work, as we have developed a DNN segmentation embedded robotic system that is able to operate as a service, rather than just an algorithm.

Our system’s algorithm also needs to meet the requirements of a service, including adaptability and robustness to various conditions. This paper makes a contribution in this regard by proposing an algorithm that, while not necessarily the most advanced in terms of certain metrics, is designed with the goal of being applicable to robotic systems in a service context, something that has not been previously achieved. This is an important step towards the development of reliable and flexible cloud robotics services, as it allows us to consider not just the performance of individual algorithms, but also how they can be integrated into a larger system and used in various real-world scenarios. By focusing on the service aspect, we are able to address the challenges of adapting to different conditions and

ensure that our system is capable of handling a wide range of tasks and environments.

Overall, we make the following contributions:

1) Our work aims to address the need for a "service" rather than just an "algorithm" in the field of cloud robotics. Our system, RoboEC2, is designed to be plug-and-play, robust, and flexible enough to be used for a variety of robotics tasks. Its cloud-edge-cooperation-based framework and implementation using Amazon Elastic Compute Cloud (EC2) allow it to be conducted at any time and place with an internet connection, and its computing power can be easily scaled up or down as needed. These features make RoboEC2 the first cloud robotics system that is truly unconstrained by time, location, or computing power, and is able to meet the high demands of a service.
2) To address the issue of adapting the algorithm for general robotic tasks for all conditions that a service requires, rather than the current proposed for specific conditions. We present a novel network offloading policy that is adopted in RoboEC2. Different from the state-of-the-art cloud robotic network offloading policy that only takes choices between the local robot and the cloud, the proposed policy dynamically splits and offloads layers of DNN. It has a better performance in related metrics.
3) We formulated the dynamic splitting and offloading process to a Multi-objective optimization (MOO) problem. We addressed it by proposing a Spotlight Criteria Algorithm (SCA), which is capable of handling diverse network conditions and flexibly trade-off robot and cloud computation. It successfully balances accuracy, time, and computational cost and addresses the gap where the proposed approaches only seek to optimize or make the best use of resources without considering the characteristics of the actual robot task.
4) We demonstrate RoboEC2 with experiments including homography estimation, object detection, and path planning. These are common functions in mobile robots, robotic arms, self-driving cars etc., with different constraints on time, accuracy, cost, etc. RoboEC2 has a better performance compared with the baseline and the SOTA method.

Organization: In Section II, we survey existing work on cloud robotics and network offloading algorithms in it. Significant challenges are also summarized in this section. To address these, we then present the framework and specific technologies of RoboEC2 in Section III and the dynamic network offloading approach in Section IV. Finally, we demonstrate RoboEC2 with experiments analyses in Section V.

II. Related Work

A. Cloud Robotics

The concept of cloud robotics can be traced back about two decades to the advent of "Networked Robotics" [15]. Reference [19] described the advantages of using remote computing for robot control in 1997. In 2001, the IEEE Robotics and Automation Society established the Technical Committee on Networked Robotics [27]. In 2010, James Kuffner first proposed "Cloud Robotics" to describe the increasing number of robotics or automation systems that rely on remote data or code for effective operation [12]. Since then, various researches of cloud robotic systems have been developed [21]. Cloud robotic system is the technology that utilizes remote cloud servers to assist local robots. RoboEarth is a famous cloud robotic system developed by European scientists, RoboEarth project envisioned the construction of "a giant network and database repository where robots can share information and learn from each other about their behaviour and environment" [37]. With the increasing developments of deep learning, a few cloud robotic learning systems are proposed, capable of sharing knowledge and improving or assisting learning of local robots. Reference [22] present a paradigm to build a cloud brain for robots, the author utilized lifelong learning to learn navigation in a federated framework. In this work, the robot learn navigation with reinforcement learning firstly, and then upload its model parameters to the cloud. The cloud server fuse these models and send to edge robots. Also for the cloud brain conduction, the [23] unitized imitation learning to train local models and the [24] generate novel scenarios to the cloud server and improve the local training. There are also some cloud robotics approaches proposed with offloading algorithms to address typical robotic issues such as SLAM [11], [31], object detection [35], grasp planning [34], motion planning [17], human-robot interaction [25], etc. We then introduce them in the following subsection.

B. Computation Offloading of Cloud Robotics

For robot tasks or computation offloading. Some other data or computation offloading algorithms for robotic computation are present. For the SLAM task, [31] present a present a system architecture for offloading computationally expensive localization and mapping tasks to smart Edge gateways which use Fog services. The comparative advantage of using Edge computing for robots and nodes are discussed keeping the focus on energy efficiency and operational latency. Instead of directly transferring a large amount of data from robots to the cloud, the work proposed an Edge-Fog-Cloud based system for performance advantages. Run from the mains supply, the powerful Edge layer can pre-process and analyze data easily, implement advanced features and ensure data security by applying complex encryption algorithms. The Fog layer can intelligently manage the smart gateways of Edge layer. Reference [11] proposes a vision based SLAM system (vSLAM) includes RGB-D cameras with cloud computing back-end. That is ideal for multiple unmanned ground vehicles (UGV) and aerial vehicles (UAV), and cooperative work flow among them. The authors proposed an algorithm in order to better identify key features in the world that could be used in each agents' own map, or one that is shared across multiple agents. The [1] present Edge-SLAM, a system that uses edge computing resources to offload parts of Visual-SLAM. They use ORB-SLAM2 as a prototypical Visual-SLAM system and modify it to a split architecture

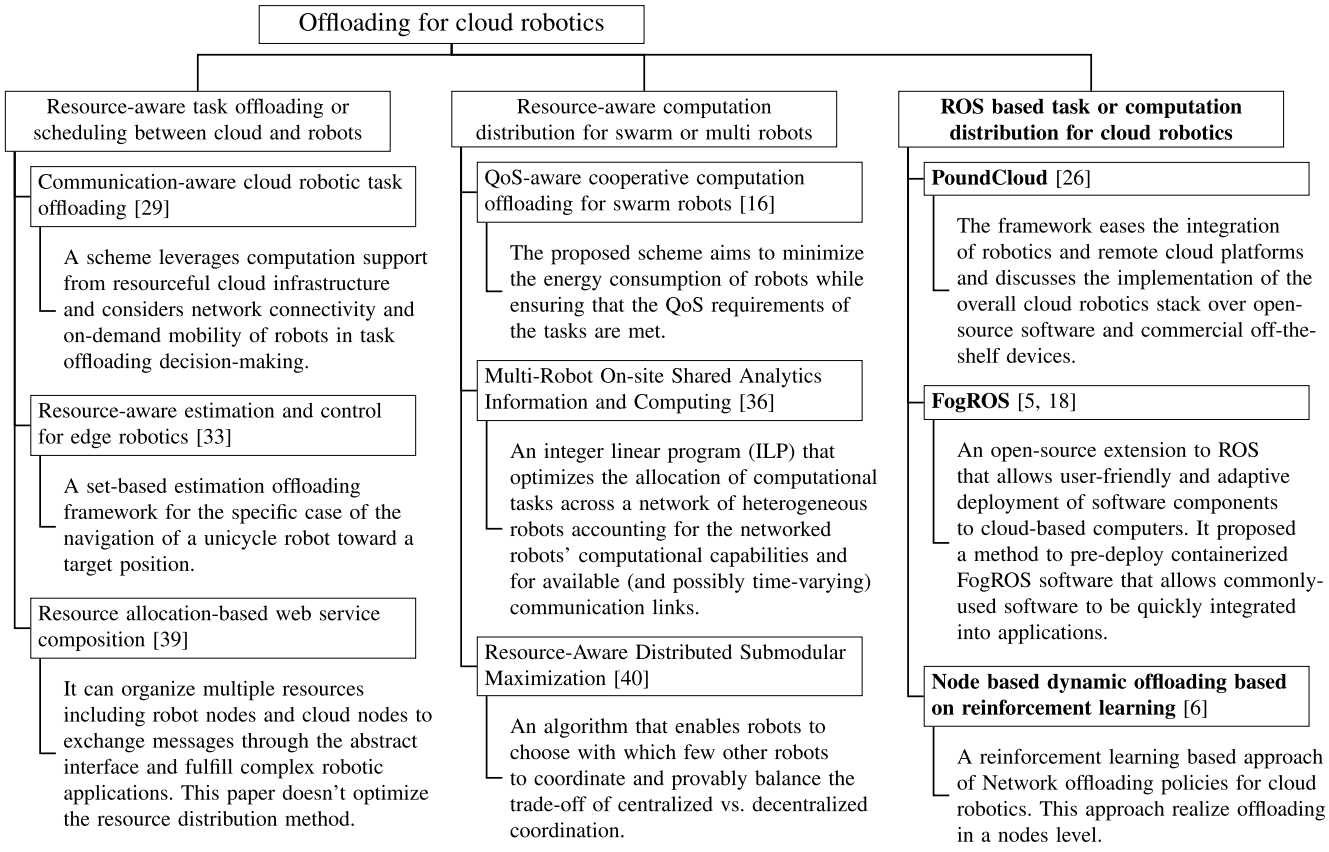


Fig. 2. General approaches of offloading for cloud robotics.

between the edge and the mobile device. For the grasping task, a cloud robotic framework for grasping is proposed [20]. The authors train the grasp model in Google Cloud TPU and then deploy the model in edge robotic arms. For motion planning tasks, the [17] introduce a method for splitting the computation of a robot's motion planning between the robot's low-power embedded computer, and a high-performance cloud-based compute service. The proposed algorithms rapidly computes an initial roadmap and then sends a mixed sparse or dense subgraph to the robot. The proposed method gains significant improvement in the responsiveness and quality of motion plans in interactive scenarios with typical latency and bandwidth limitations. Reference [2], the author present a novel data offloading decision-making framework, where users have the option to partially offload their data to a complex Multi-access Edge Computing (MEC) environment, consisting of both ground and UAV-mounted MEC servers.

However, these works are more focused on purely resource optimization problems that are application-specific and not generalized. Such specific algorithms are weakly adaptable to various robotic tasks and aren't able to react effectively to some extreme conditions, such as disconnection from the Internet. For most robot developers, such algorithms are not in great demand. On the other hand, ROS is the developed standard for creating robot automation applications and components, it serves as the bridge between robots and functional packages. ROS is the basis for most intelligent robots, it has a unique node-based communication mechanism, thus only offloading data or computation without ROS is insufficient. Although these papers have proposed the algorithms of offloading in robotics, most of them are without ROS. The community needs an easy-to-deploy, robust system framework that is suitable for a variety of robotic tasks rather than a specific solution.

Additionally, we conducted research on general offloading approaches for cloud robotics. As shown in the Fig. 2, we have summarized the latest offloading methods for cloud robotics. General offloading methods for cloud robotics can be divided into three directions: task allocation between cloud and robots, computation allocation among robots in swarm robotics, and computation allocation based on ROS Node. From the figure, it is clear that this paper focuses on the direction of computation allocation based on ROS Node. Therefore, the comparative methods we have chosen include PoundCloud [26] FogROS [5], [18] and reinforcement learning based node offloading [6].

C. FogROS

FogROS is the first generalized framework that utilizes ROS to conduct the computation offloading. It is regarded as an extension of the Robot Operating System (ROS), the defacto standard for creating robot automation applications and components. It allows researchers to deploy components of their software to the cloud with minimal effort, and correspondingly gain access to additional computing cores, GPUs, FPGAs, and TPUs, as well as predeployed software made available by other researchers. FogROS allows a

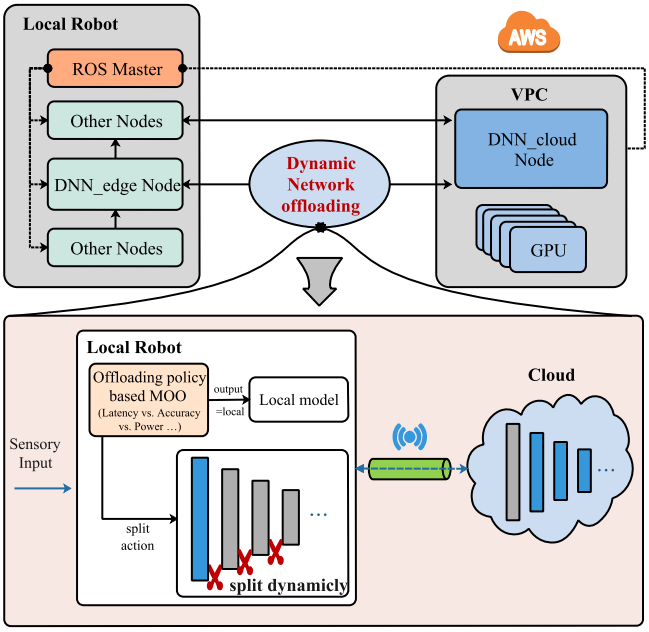


Fig. 3. General framework of RoboEC2.

researcher to specify which software components will be deployed to the cloud and to what type of computing hardware. The authors also evaluate FogROS on robotics tasks as SLAM, grasp planning, and motion planning. FogROS has improved the performance greatly compared with computing on board. However, there is not any adaptive algorithm to improve the offloading performance in FogROS. Although offloading nodes is configurable, we cannot always offload a node to run in the cloud all the time because we cannot guarantee the stability of the robot's operating conditions as the network latency. Once the cloud server is down, the whole system cannot continue to run. The lack of adaptability of FogROS to the conditions hinders it from becoming popular.

Overall, there is still a gap between application-specific offload algorithms and FogROS in terms of widespread use.

III. Design and Implementation of the RoboEC2 Framework

In this section, we firstly introduce the general framework of RoboEC2. Then we list the design principles of RoboEC2 and follow specific implements of the RoboEC2 framework.

A. General Framework of RoboEC2

We present the framework of RoboEC2 as shown in Fig.3. There are robots, cloud servers in the cloud robotic system. For the proposed cloud robotic network offloading system, functional ROS nodes are deployed in the local robot, where contains DNN nodes. It can be seen that the previous DNN node is split into two nodes, one in edge (local robot) and one in the cloud. They compute different layers of the DNN. DNN nodes in edge receive information from sensor nodes or other nodes, and then send the processing results to downstream nodes. The instance generated in AWS is deployed with a DNN node in the cloud, which will receive the output from the local DNN node, and then send its own output to downstream nodes in the local robot. The cloud is with a high computing performance configuration. For the offloading from the local robot to Amazon EC2, there is a split-offloading module in the local robot. The offloading module dynamically splits layers of the DNN into the edge node and the cloud node. In every step, there will be different computations distributed in edge and cloud, or totally local/cloud computing.

B. Design Principles

The design principles of RoboEC2 is based on the challenges that summarised in the Section II. We believe that the system that meets the following design principles is capable of resolving the above challenges in cloud robotics.

1) *Flexible*. As mentioned in the above, it is impossible to build a separate server with network offloading for a local robot every time. Therefore, we need to find a new way to build clouds flexibly for local robots. Specially, the computing power of the cloud is flexible, the connection time is arbitrary, and the location is unconstrained. Everything depends on the user and local robots.
2) *Convenience*. Exiting cloud robotic systems are always with complex functionality and deployment process, which hinders the development of cloud robotics. Accordingly, we need to acquire a new approach to make convenient for users to deploy their network offloading policies in the system. This is reflected in fewer interactions with machines and fewer code modifications.
3) *Robust networkability*. Connectivity is the main issue in cloud robotics. Network instability severely degrades the performance of cloud robotic systems, which has been ignored in much of the literature or simply take the robust network condition as an assumption. Neural Networks offloading runs in real time, so we need a robust solution to cope with connect conditions in running process. It means that the neural network offloading in RoboEC2 is expected to run at any conditions, including disconnection.
4) *Excellent network-offloading*. Deep neural network offloading is regarded as one of the main tasks for the system. As deep neural network models are more widely used in robotics. This task is becoming more urgent to acquire advanced algorithms to cope with it. It means that the network-offloading approach adopted in RoboEC2 should have better performance compared with the state-of-the art (SOTA) method.

C. Specific Implements

1) To Satisfy Design Principle 1): RoboEC2 builds clouds for local robots based on Amazon Web Services (AWS) as seen the cloud elements in Fig.4. AWS is the world's most comprehensive and broadly adopted cloud platform [32], so RoboEC2's adoption of AWS enables most users to get started quickly without acquiring new knowledge. We utilize the elastic computing cloud (EC2) service of AWS to build RoboEC2. Elastic computing is the ability to quickly expand or

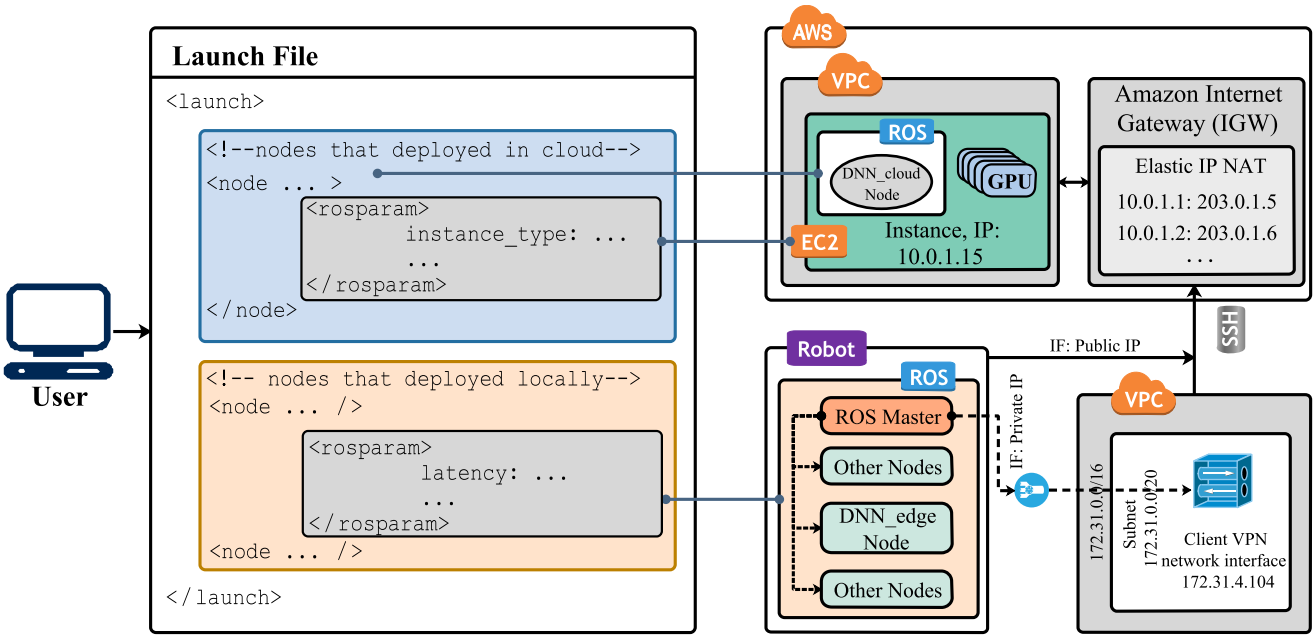


Fig. 4. Deployment of ROS and Amazon EC2. RoboEC2 implements the deployment of ROS and EC2 with a lanunch file.

decrease computer processing, memory, and storage resources to meet changing demands without worrying about capacity planning and engineering for peak usage. Typically controlled by system monitoring tools, elastic computing matches the amount of resources allocated to the amount of resources actually needed without disrupting operations. With cloud elasticity, the user avoids paying for unused capacity or idle resources and doesn't have to worry about investing in the purchase or maintenance of additional resources and equipment. These characters make RoboEC2 flexible in cloud configuration. Elastic computing is more efficient than typical IT infrastructure, is typically automated so it doesn't have to rely on human administrators around the clock, and offers continuous availability of services by avoiding unnecessary slowdowns or service interruptions [7], which makes the connection time is arbitrary and the location is unconstrained in RoboEC2. Using Amazon EC2 eliminates our need to build a separate server for the local robot every time. In particular, operations to the cloud are carried out automatically in RoboEC2. We invoke the AWS API to implement this part of the process in the proposed above. In this process, RoboEC2 will split a relatively separate computation space in AWS and allocate a flexible public IP according to the user's requirements. The users can use Amazon EC2 to launch as many or as few virtual servers as they need, configure security and networking. Based on this, We can therefore infer that RoboEC2, which is based on Amazon EC2 for building clouds, is able to satisfy our principle 1) for system design.

2) To Satisfy Design Principle 2): RoboEC2 utilizes the file-launch mechanism of ROS to execute function module deployment in edge or cloud, which is as illustrated in Fig.4. The Robot Operating System (ROS) is a set of software libraries and tools that help users build robot applications. From drivers to state-of-the-art algorithms, and with powerful developer tools. ROS provides a mechanism of starting the master and many nodes all at once, using a file called a launch file. The use of launch files is widespread through many ROS packages. Any system that deploys more than one or two nodes is likely to take advantage of launch files to specify and configure the nodes to be used. Naturally, to improve the convenience of RoboEC2, we also expect to implement network offloading deployment of edge and cloud nodes through only one roslaunch file, which is also used in the similar way in [5]. More advanced than [5], we need to be specific to the neural network structure inside the node and offload some layers rather than directly uploading the node, the former is more challenging but we have achieved this. An example of the contents of the launch file has included in the RoboEC2 pipeline.[1]

In the example launch file of RoboEC2, the user only need to configure the parameters of EC2 instance type, the latency constraint and the computing power of the local robot. After the configuration, The user only need to follow the pipeline that only has 3 steps. RoboEC2 will automatically run. The user does not need to do anything else. This operation makes RoboEC2 very convenient and thus satisfies principle 2.

3) Interaction Between ROS and Amazon EC2: RoboEC2 utilizes the nodes communication protocol of ROS for cloud-edge(local robot) interaction. As we known, the main mechanism used by ROS nodes to communicate is by sending and receiving messages. The messages are organized into specific categories called topics. Nodes may publish messages on a particular topic or subscribe to a topic to receive information [13]. The Cloud and the local robot share one master in ROS but different hostnames. The role of the master is to enable individual ROS nodes to locate one another. Once these nodes (both in local and cloud) have located each other, they communicate with each other peer-to-peer. Until then, you may encounter problems at the network level. As shown

[1]RoboEC2 pipeline: https://github.com/RoboEC2/RoboEC2

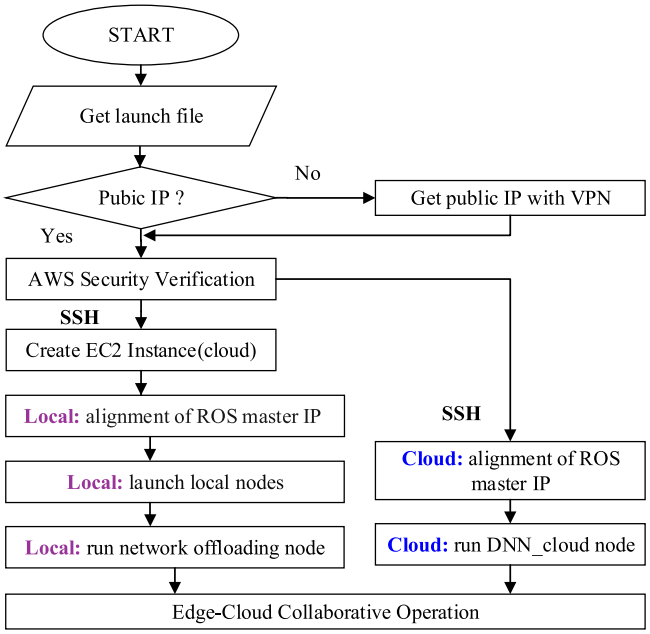


Fig. 5. Key procedures in RoboEC2.

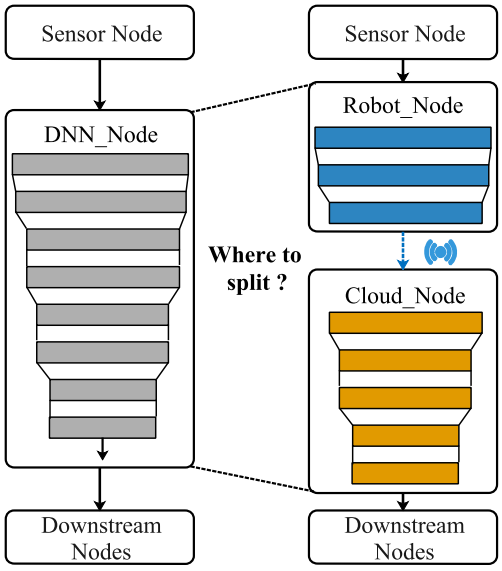


Fig. 6. RoboEC2 offloads the DNN in a split-offloading way.

in Fig.4 and Fig.5, if your local robot is on a LAN and does not have a public IP, then you will need to forward network traffic through a virtual private network (VPN). That is because robots only with private IP in the LAN cannot communicate with EC2 Instance in the WAN directly.

4) The Process in RoboEC2: In RoboEC2, the local robot utilizes SSH to create EC2 Instance, launches nodes in cloud, and builds environments that include deep neural network (DNN), offloading policy, and basic ros, etc. Secure Shell (SSH) is a cryptographic network protocol for operating network services securely over an unsecured network. Typical applications include remote command-line, login, and remote command execution [38]. Firstly, we use SSH to establish a connection to AWS, creating an AWS EC2 Instance, which is an isolated computing region with a public IP. Then, As illustrated in Fig.5, after the user execute the launch file, the nodes in local robots and instances will run. Local nodes are easy to launch, while launching nodes in the cloud will utilize SSH. After all nodes are running, the nodes in local and cloud are able to communicate with each other via WAN or LAN-VPN-WAN. Simultaneously, the offloading policy layers node performs dynamic offloading algorithms to offload some of the local DNN to the cloud. The cloud will return the DNN output or intermediate results to the edge. The key process procedures are shown in Fig. 5. The input is a launch file, and the result is the cloud-edge cooperative running.

Above is the main components and processing of RoboEC2. Note that we omit the distribution of complex authentication process of AWS, the tunneling of SSH, the pushing of the cloud code, etc. All of these need to be implemented in the RoboEC2. We open source all the steps of RoboEC2 at its pipeline.

Implementations in subsection-B satisfy the design principle **1)** and **2)**. We then present a novel network offloading approach that is the key to address principle **3)** and **4)**.

IV. The Dynamic Network Offloading Approach

As we said in principle 3), the robust networkability means RoboEC2 is capable of offloading deep neural networks at anytime and anywhere. As we all known, the network's connectivity is uncontrollable but the algorithm is controllable. Therefore, the way we implement the goal is not to guarantee robustness of the network, but robustness of the network-offloading approach for various network conditions. Combined with principle 4), it can be deduced that the network-offloading approach is the core of RoboEC2. Therefore, we present a novel network offloading approach named Spotlight Criteria Algorithm (SCA) **to satisfy principle 3) and 4).** In the following, we will introduce the proposed approach.

A. Problem Statement of the Network Offloading Approach

In this section, we focus on a common scenario in cloud robotics in which the local robot experiences a stream of sensory input that it must process with DNN. In every time step, it must choose to compute the DNN onboard or offload the DNN to the cloud over a network. Furthermore, if it chooses offloading, the robot also should decide how to offload the DNN. Namely, it must locate the split point that divides the DNN into a local-computing part and a cloud-computing part, which is more flexible than the SOAT cloud robotic offloading approach that only decides into local or cloud. These converge into a policy conducted in the local robot. Factors that decide the output of this policy is time, local cost of computation, accuracy etc. that are all we considered in this process. As presented in Fig.6, The robot flexibly decides to compute with local model or to offload a part of the DNN to the cloud. The robot makes decisions depending on circumstances in every time point. So this is a dynamic offloading. This way ensures that the robot can cope with any network condition including disconnected. Moreover, an optimal action will be chose in the current limited resource condition. Next, we will model the problem mathematically.

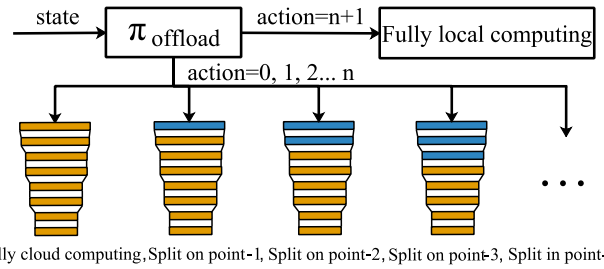


Fig. 7. Key process points in RoboEC2.

The dynamic network offloading problem in cloud robotics can actually be regarded as a discrete process where the robot behaves according to it-self conditions. As such, it can be defined as an environment-action-reward framework. However, it is not a Markov decision process or reinforcement learning, but a Multi-Objective Optimisation (MOO) problem. This is because each state is independent and has no relation to the previous one.

$$\mathcal{R}_{\text{offload}} : \mathcal{S}_{\text{offload}} \rightarrow \mathcal{A}_{\text{offload}}, s.t.\ \mathcal{C}_{\text{offload}}. \tag{1}$$

where, $\mathcal{S}_{\text{offload}}$ is the state space, $\mathcal{A}_{\text{offload}}$ is the action space, $\mathcal{R}_{\text{offload}}$ is rewards of actions, $\mathcal{C}_{\text{offload}}$ is constraint. In the following, we define each of these elements in terms of the abstractions of the dynamic network offloading problem discussed above. Fig.7 shows the actions that the offloading policy will take based on the current condition. In a word, the local robot takes an offloading action based the environment to acquire a higher reward of metrics with constraints.

1) Action Space: The action space for the DNN offloading policy in RoboEC2 includes totally local computing, totally cloud computing, and partially local partially cloud computing. For the split-offloading action set, we set n split points in DNN, which correspond to n actions respectively. Specifically, we have $n+2$ discrete actions:

$$a^t_{\text{offload}} = \begin{cases} 0, \text{ totally cloud computing, } \hat{y}^t = f_{\text{cloud}}\left(x^t\right) \\ 1, \text{ split in point 1, } \hat{y}^t = f_{\text{cloud}}\left(x^{\tau_{\text{cloud}}}\right) \\ 2, \text{ split in point 2, } \hat{y}^t = f_{\text{cloud}}\left(x^t\right) \\ 3, \text{ split in point 3, } \hat{y}^t = f_{\text{cloud}}\left(x^t\right) \\ \vdots \ , \quad \vdots \quad \vdots \\ n, \text{ split in point n, } \hat{y}^t = f_{\text{cloud}}\left(x^t\right) \\ n+1, \text{ totally local computing, } \hat{y}^t = f_{\text{local}}\left(x^t\right) \end{cases} \tag{2}$$

From the action space and Fig.7, we can see that the robot can flexibly choose the way of offloading. When the network is disconnected, it will take the action $n+1$. When the network is particularly excellent, it will take action 0. And when the network is in a mediocre state, it needs to make choices based on rewards, constraints, and other factors in the state space.

2) State Space: Factors in state space should reflect the conditions what the robot is with, including network conditions, computation power, amount of data to be transmitted, and expectation time. `Latency`: Latency is a reflection of the condition of the network, which impacts the policy a lot. In actuality, the "latency" in the paper means "time latency" rather than "network latency". Time latency is a comprehensive result of both bandwidth and network latency. Therefore, we consider both bandwidth and network latency comprehensively. A program for detecting network latency is deployed in RoboEC2. This program will obtain the network latency in real time, which will be considered as one of the factors in state space. `Computing power`: The computing power of an EC2 instance is configured by the user, and the robot's is an inherent property of it. RoboEC2 takes FLOPS [9] as a metric that is widely used to quantify computing power of the local robot and the cloud. Its calculation method is formula (3) [10]:

$$\text{FLOPS} = \text{cores} \times \frac{\text{cycles}}{\text{second}} \times \frac{\text{FLOPs}}{\text{cycle}}. \tag{3}$$

where the first multiplier is the number of cores of the processing unit, the second multiplier can be regarded as frequency and the last multiplier is the number of single cycle operations. Practically, GPU mostly provides its FLOPS, the [3] published by UC Berkeley lists CPU performances with FLOPS. `Amount of data transmitted (ADT)`: Output of the nearest layer before the split-point will be transmitted to the cloud if the local robot decide to do network offloading, so its dimension is ADT. If the local robot decide to compute locally, the ADT is 0. If the split-point is before the first layer that means conducting computation in cloud totally, the ADT is the same as the dimension of this frame of sensor data. To sum up, the state space can be expressed as this:

$$s^t_{\text{offload}} = \{\underbrace{[Data^0, Data^1, \ldots, Data^n, Data^{n+1} = x]}_{\text{amount of data}}, \underbrace{latency}_{\text{network condition}}, \underbrace{C_{local}, C_{cloud}}_{\text{computing power condition}}\} \tag{4}$$

3) Metric Space: Factors in metric space reflect all what we considered, including accuracy, time, and local computation cost. We model the raw sensory input into the local robot as a sequence $\{x^t\}$, where x_t represents the data, such as a video frame that arrives at time t. The computation that we consider offloading to the cloud is the process of estimating some output y^t given some input x^t. The estimating model DNN is based on deep learning, x^t is input and y^t is output of the model. For instance, in the scenario of processing a video stream, y^t could be a object detection result of the input frame x^t, useful for downstream decision making in the local robot. To deal with extreme conditions such as disconnection, RoboEC2 deploys two models, a local lightweight model and a DNN model to be split-offload. Both the local and the DNN models map a query x^t to predictions of y^t and additionally, a score of their accuracy $Accu^t$:

$$\begin{aligned} \hat{y}^t_{\text{local}}, \text{Accu}^t_{\text{local}} &= f_{\text{robot}}\left(x^t\right) \\ \hat{y}^t_{\text{cloud}}, \text{Accu}^t_{\text{cloud}} &= f_{\text{cloud}}\left(x^t\right) \end{aligned} \tag{5}$$

Typically, f_{local} is a computationally efficient model suitable for disconnection conditions or network-resource-constrained mobile robots. In contrast, f_{cloud} represents a more accurate and deep DNN model. These models are generally computationally complex, so once the robot has selected these models, RoboEC2 will execute the split-offloading for DNN.

`Accuracy`: the accuracy of local and cloud models can be measured through a loss function $\mathcal{L}(y^t, \hat{y}^t)$ that penalizes differences between the predictions and the true results. c and $\text{Accu}^t_{\text{cloud}}$ are accuracy measurements of local or cloud models. For developed DNN model, such as VGG, GoogLeNet, YOLO etc., we believe that they have presented their accuracy in papers. RoboEC2 takes these accuracy directly as $\text{Accu}^t_{\text{local}}$ and $\text{Accu}^t_{\text{cloud}}$. of the state space:

$$\text{Accu}^t_{\text{cloud}} = \begin{cases} \text{Accuracy of cloud model,} \\ (a^t_{\text{offload}} = 0, 1, \cdots, n). \\ \text{Accuracy of local model,} \\ (a^t_{\text{offload}} = n+1). \end{cases} \tag{6}$$

`Time`: The computing time includes the computing time in the robot and the cloud, with communication time between them. Practically, the time can be estimated with latency, data volume, computing power. It is impossible to acquire the exactly time but estimation is suffice: $\text{Time}^t_{offload} = f(Data_n, latency, C, Expec_{time})$. `Computational cost`: Computational cost in the local robot. It refers to the local consumption of the DNN. We quantify this metric in terms of the number of nodes computing locally:

$$c^t_{\text{offload}} = \begin{cases} \text{number of nodes of } DNN_{cloud},\ a^t_{\text{offload}} = 0 \\ \text{number of nodes after the split 1,}\ a^t_{\text{offload}} = 1 \\ \text{number of nodes after the split 2,}\ a^t_{\text{offload}} = 2 \\ \cdots \\ \text{number of nodes after the split n,}\ a^t_{\text{offload}} = n \\ \text{number of nodes of } NN_{local},\ a^t_{\text{offload}} = n+1 \end{cases} \tag{7}$$

4) Modeling as a MOO Problem: The network offloading policy always aims to achieve a higher score in all of the metrics. Therefore, it is clearly a Multi-Objective Optimization problem (MOO). We model the problem as this:

$$\begin{aligned} \max_{a^t_{\text{offload}}} \ & F(x) = [R_{time}(a^t_{\text{offload}}), R_{accu}(a), R_{cost}(a^t_{\text{offload}})] \\ \text{s.t. } & Time(s^t_{\text{offload}}, a^t_{\text{offload}}) \le \alpha \\ & Accu(s^t_{\text{offload}}, a^t_{\text{offload}}) \ge \theta \\ & Cost(s^t_{\text{offload}}, a^t_{\text{offload}}) \le \gamma \end{aligned} \tag{8}$$

where s^t_{offload} is the statement in time step t, a^t_{offload} means actions, R is rewards of metrics, α is the expectation time, θ is the expectation accuracy, γ is the expectation computing consumption. After observation, we find that the model has its particularities: 1. The accuracy is binary, either cloud or local; 2. Cost and Time are negatively correlated; 3. Action as a variable is discrete and finite. So this is a particular MOO. It is solvable by general methods, but they are inefficient. So we present a new algorithm for this specific issue based on its particularities and cloud robotic characters. Time refers to the time required to process each frame, including local computation time, communication time, and cloud computation time. Accuracy refers to the degree of correctness in the results. Cost refers to the computational resources consumed.

We take into account the computing power of edge robots, and the algorithm complexity is a factor to be considered. We do our best to reduce the time of running its algorithm and the resource consumption of the edge robot. It must be related to the number of decisions that can be chosen. The complexity of the algorithm, as seen in Algorithm1, depends on L that is the number of layers of DNNs. The complexity of the algorithm is O(L) as it needs to be estimated for each layer of the DNN when running the offloading decision algorithm. The restriction for some indicator boundaries has no effect on the complexity of the algorithm, it just removes some restricted decisions.

Algorithm 1 Spotlight Criteria Algorithm

input : DNN_{cloud}, Accu_{cloud}: deep neural network and its accuracy in cloud; NN_{robot}, $Accu_{robot}$: model in local robot and its accuracy; C_{robot}, C_{cloud}: computing power of robot, cloud; α: the expectation time; θ: the expectation accuracy; γ: the expectation computing consumption; x: sensor data; spotlight: the metrics of greatest concern configured by the user.

output: OffloadAction

```
while x input from the time step t do
    L=NumLay(DNN_cloud);
    latency=NetCondition();
    Data^L=0;
    Action[0]=0;
    for n ← 1 to (L-1) do
        Data^n=OutputDimen(DNN_cloud.layer[n]);
        s^t_offload=[Data^n, Data^{n+1}, C_cloud, C_robot, latency];
        Action=Action+[n];
    Action[L]=L;
    Time^t_offload=Estimate(s^t_offload);
    Cost^t_offload=Calculate(DNN_cloud, NN_robot);
    for i←0 to L do
        if Time_{s^t_offload, a^t_offload=i) > α then
            Action delete Action[i]
    if Accu_local < θ or Cost_{s^t_offload, a^t_offload=n+1 > γ then
        Action delete [L];
    else
        if Spotlight='Accuracy' then
            Action=Action delete [L]
        else
            OffloadAction=LastElement(Action)
    return (OffloadAction)
```

B. Multi-Objective Optimization Problem Solving for Dynamic Network Offloading Problems in Cloud Robotics

We propose the Spotlight Criteria Algorithm (SCA) to solve the specific MOO modeled in above. **Why spotlight?** Robots with different functions pay different attention to different metrics. For example, for high-speed mobile robots, it pays more attention to real-time performance; for the fire detection function of drones, it pursues lower energy consumption; for

the face recognition function of logistics vehicle robots to unlock express boxes, it pays more attention to higher accuracy rather than time or consumption. In one sentence, there is a "spotlight" for each function of the robot and it is inappropriate to make trade-offs among different objective functions as in the regular MOO. Therefore, we proposed the Spotlight Criteria Algorithm to address this specific MOO, as shown in Algorithm 1. In RoboEC2, the users need to provide the spotlight parameter to get the network offloading action. SCA is designed to enable the overall system to achieve service-level performance rather than optimizing for a specific metric, such as DNN offloading. In our system, different functions of the robot may prioritize different metrics, such as real-time performance, energy consumption, or accuracy. SCA allows users to specify the "spotlight" parameter, which determines the priority of each metric for a particular function. This enables the system to make trade-offs among the different objective functions in a way that is appropriate for the specific needs of each function. Algorithm 1 shows the steps for implementing SCA in our system.

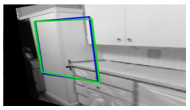
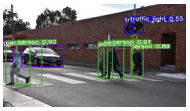
In Algorighm 1, line 1-12 is the function to obtain the state space and actions, line 13-17 is to obtain Pareto set, line 18-23 is to select the best action based on the spotlight. Accuracy is dichotomous, so a division is made directly in line 17-18. Since time and cost are inversely related, we can get Pareto set by the constraint set. SCA chooses the best action among Pareto set, which can be completely determined by spotlight finally.

In Algorithm 1, we focus on the matrices required by robot application scenarios, which is different from traditional cloud computing technology. As demonstrated in the algorithm, we need to eliminate the hard metrics needed for robot applications first, and then optimize. The robot has to choose between the three indicators mentioned above, so there is always one that it pays the most attention to, which is why we call it "Spotlight". In the code implementation, we use the method of Online Learning to predict the transmission time, and use the calculation index to predict the local consumption. This is a dynamic process, that is, adapting to conditions changes through dynamic decision making.

V. Experiments and Comparisons

To verify the effectiveness of RoboEC2, we demonstrate offloadings of three common functions in robotics with RoboEC2, as illustrated in Table. I. Homography estimation plays an important role in SLAM initialization process and grasping tasks. Object detection is essential in environment perceptions of mobile robots as robotic dogs. Path planning is one of the basic functions in self-driving cars. DNNs have achieved state-of-the-art results in the above tasks. In the experiments of this work, we try to offload these DNNs to the cloud with RoboEC2 to achieve the goal we set at the beginning of this paper and compare it with another three approaches the local computation, the PoundCloud [26] and the SOTA method [6], one of the previous best system paper finalists of RSS. And The local is with Intel(R) Core(TM) i7-7700HQ CPU and the cloud instance is Amazon EC2 P3. According to the application scenarios of the three tasks, they pay more attention to accuracy, consumption and time respectively. This gives the experiment a more comprehensive measure of performances on different tasks in robotics.

TABLE I
Robotic Tasks and Corresponding Scenarios

Task	Scenario		
homography estimation			
object detection			
path planning			
	grasp	SLAM	
	robotic dog	mobile robot	
	self driving	logistics vehicle	

The seleted SOTA method is the formulation of a novel Robot Offloading Problem and the proposed solution using deep reinforcement learning. The problem addresses the issue of how and when robots should offload sensing tasks to improve accuracy while minimizing the cost of cloud communication, particularly in resource-constrained environments. The selected SOTA method has been demonstrated to significantly improve vision task performance by up to 2.6 times that of benchmark offloading strategies through simulations and hardware experiments using state-of-the-art vision DNNs. This allows for the potential for robots to significantly transcend their on-board sensing accuracy while minimizing the cost of cloud communication. The selected SOTA method won the best system award at the RSS conference and It is by far the best algorithm for task offloading of robots.

A. Cost Spotlight With A Time Constrain: The Experiment of Object Detection

Object detection is a technology related to computer vision that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos, which is a common function in robotics especially mobile robots as robotic dog. At present, the best object detection algorithms are all based on deep learning. In this experiment, we respectively build a MobileNet [30] in edge and a VGG19 [4] in cloud for the feature extraction. The SOTA and RoboEC2 execute in an edge-cloud-collaboration way and the baseline approach execute in an edge-only way. Object detection is of great significance for mobile robots in crowded scenes. Therefore, in this experiment, we assumed that it would be applied to a crowded scene. In crowded scenes, the speed of robots is not fast, so the constraint on time exists but is limited. At the same time, the endurance of mobile robots is important as robots that are capable of running in

Fig. 8. Representative experimental results in object detection comparison between the baseline approach, the SOTA approach and RoboEC2.

TABLE II
PERFORMANCE COMPARISON ON THE OBJECT DETECTION TASK

Matric	Local	SOTA	**RoboEC2**	PoundCloud
Accuracy	0.61±0.12	0.77±0.08	0.92±0.04	0.92±0.04
Cost	758624	537294	**325784**	**0**
Time	225ms	155ms	137ms	**Timeout**

crowds are always low-load. So we set the spotlight parameter to *cost* and set a time constraint.

As illustrated the Table II and Fig. 8, RoboEC2 has better performances compared with the baseline and SOTA approaches in cost and accuracy metrics, while meeting time constraints. For the PoundCloud, it offloads all computation to the cloud but occurs to timeout in the process. As shown in Fig. 8, with time constraints, completely wrong scenarios that marked with red boxed appear in the previous SOTA method. This is because the result of the previous frame is used in the current frame in its decision action, which will not appear in RoboEC2. So RoboEC2 wins this experiment.

B. Accuracy Spotlight: The Experiment of Homography Estimation

A homography is a mapping between two images of a planar surface from different perspectives. They play an essential role in some robotic tasks. In this subsection, we acquire the homography based on a learning method present in [8]. We build a shallow lightweight neural network that has 8 layers and a computationally intensive deep neural network that as 19 layers, which will be applied in the experiment of SOTA method and RoboEC2. The baseline approach is totally local computing, and it will only use the shallow lightweight network. Homography estimation generally occurs during initialization, so there are limited requirements on time but more on accuracy. So the Spotlight parameter of the Spotlight Criteria Algorithm in RoboEC2 is accuracy. The experimental results are as follows:

From the Table. III and Fig. 9, it can be shown that the compared SOTA method, RoboEC2 and the PoundCloud all achieve better accuracy when time is not limited. For the metric of local cost, it is 0 in both SOTA method and RoboEC2 because the two upload all natural network computations to clouds. Therefore, in this experiment, the PoundCloud, the SOTA and RoboEC2 perform equally well.

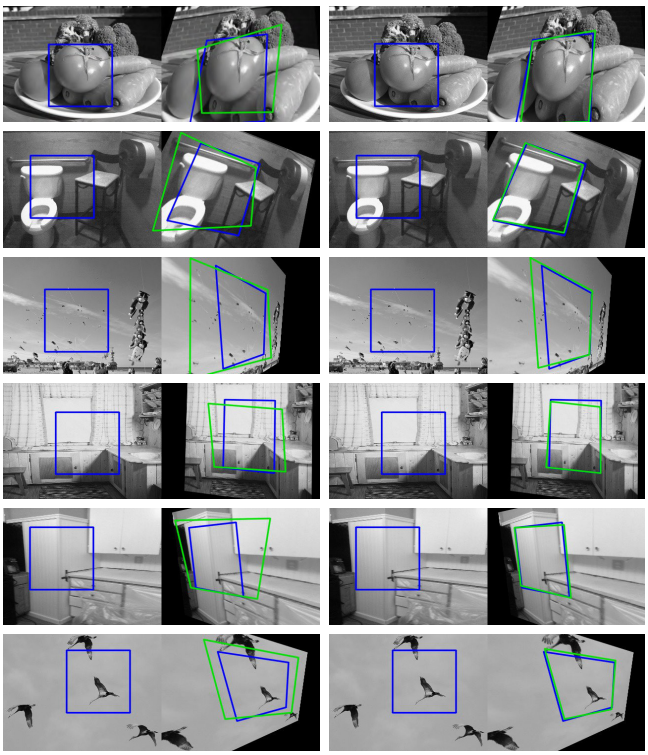


Fig. 9. Representative experimental results in homography estimation comparison between the baseline approach, the SOTA approach and RoboEC2. Note: The SOTA and RoboEC2 have same outputs in the last column.

TABLE III
PERFORMANCE COMPARISON ON THE HOMOGRAPHY ESTIMATION TASK

Matric	Local	SOTA	**RoboEC2**	PoundCloud
Error	24.1±2.43	9.1±0.92	**9.1±0.92**	9.1±0.89
Cost	759652	0	0	0
Time	270ms	425ms	411ms	415ms

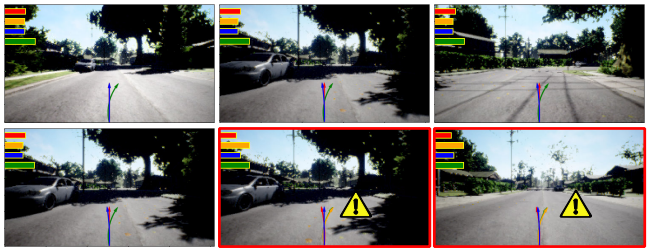


Fig. 10. Representative experimental results in path planning comparison between the baseline approach, the SOTA approach and RoboEC2. In the figure, the red arrow represents the label direction, the orange arrow represents direction that the SOTA approach output, the blue arrow represents the direction that RoboEC2 output, the green arrow represents the direction that the baseline approach output. The red box represents the errors that happened in the SOTA approach.

TABLE IV

PERFORMANCE COMPARISON ON THE PATH PLANNING TASK

Matric	Local	SOTA	**RoboEC2**	PoundCloud
Error	0.07±0.004	0.031±0.002	0.015±0.001	0.015±0.001
Cost	757596	624894	473874	0
Time	152ms	111ms	**110ms**	**Timeout**

C. Time Spotlight: The Experiment of Path Planning

Path planning is one the most import functions in self-driving cars. Therefore, we assume that the offloading of this experiment in this subsection is conducted in a self-driving scenario. Similar to the above experiment, we build two natural networks with different layers, a shallow that has 8 layers and a deep has 20 layers. The two are trained with [23]. The SOTA and RoboEC2 execute in an edge-cloud-collaboration way and the baseline approach execute in an edge only way. As we known, execute time is the most important in self-driving scenarios, because the speed of the autonomous car is relatively higher than other types of robots. There is no doubt that the spotlight parameter in RoboEC2 is time in this experiment. The results are illustrated in Table IV and Fig. 10.

As shown in Fig. 10 and Table. IV, similar to the experimental results in the following experiment, RoboEC2 performs better in the metrics with constraints in this experiment. Meanwhile, with the more enhanced time constraint compared with the object detection experiment in subsection V-A, the SOTA approach appears to have more delayed decisions marked by the red box in Fig. 10, which decreases the accuracy of the SOTA approach. For the PoundCloud approach, it occured timeout because the totally offloading approach and failed in this experiment. So RoboEC2 wins in this experiment.

D. The Comparative Experiment With FogROS

SLAM is one of the common tasks for robots. In this experiment, we perform simultaneous localization experiments in SLAM leveraging pySLAM as present in Fig.11. We fuse end-to-end Homograph estimation into pySLAM and optimize the module leveraging RoboEC2. Fig. 12 shows the experimental scenario. Meanwhile, we provide Fig. 12, an end-to-end SLAM work, which is a potential application scenario. Because end-to-end SLAM approaches have more accurate performance than traditional SLAM approaches, but high

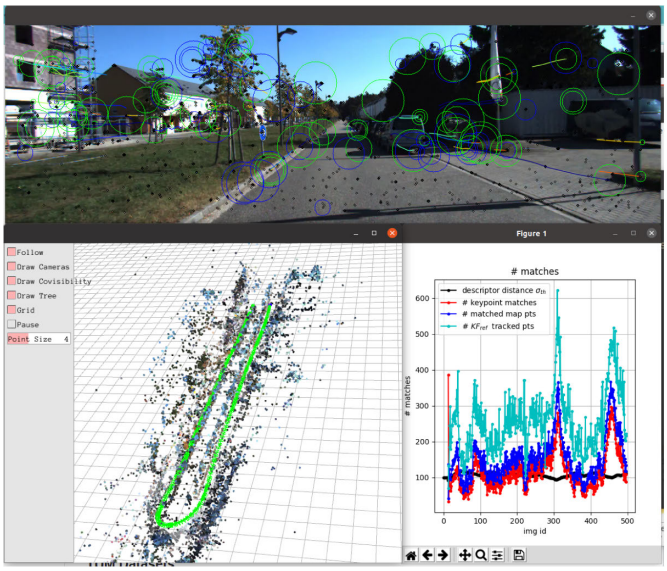


Fig. 11. The experimental scenario of SLAM. Simultaneous localization modules are computing via RoboEC2.

Fig. 12. Figure from a published End-to-end SLAM approach named Droid-slam. RoboEC2's potential application scenarios for high computing power required SLAM tasks.

computational resource requirements hinder their applications. Therefore, we also provide the figure to demonstrate the practical value of RoboEC2.

Fig. 13 depicts the performance comparison between RoboEC2 and FogROS in the SLAM task, and also the performance comparison between the algorithm-level system and the node-level system. The first row of the figure shows the computation with a network speed of 10M/s. In this case of sufficient resources for the network speed, RoboEC2 and FogROS take the same computing strategy, i.e., full cloud computing. A larger performance improvement is achieved compared to the fully local computation in the last row. This also reflects the feasibility of cloud robotics. The second row shows the computing in the case of network speed 5M/s, where the network resources are limited, RoboEC2 also takes a full cloud computing. The third and fourth rows present a comparison for the case of network speed at 2M/s. RoboEC2 takes different computing strategies compared to FogROS, where RoboEC2 chooses actions that transfer less data. It has allocated the computation elastically according to the resources, resulting to a better performance. The fifth and sixth rows show a comparison with a network speed of 1M/s, a scenario with highly constrained network speed, where RoboEC2 embodies a more superior performance improvement, more than twice the performance of the node-level FogROS.

Table V shows a comparison of experimental data results for pure robot computing, FogROS, and RoboEC2. Also,

TABLE V
COMPARISON OF EXPERIMENTAL RESULTS FOR PURE ROBOT-COMPUTING, FOGROS AND ROBOEC2 IN THE SLAM TASK

2[4]*[c]Metrics/ Speed	Latency(s)			CPU (relative usage)			Power consumption (mw/robot frame)		
	FogROS	**RoboEC2**	Robot	FogROS	**RoboEC2**	Robot	FogROS	**RoboEC2**	Robot
512K-up limitation	10.18	↓ **1.99** ↓	2.61	1+17%	↓ **62%** ↓	96%	9614	↓ **3607** ↓	4617
1M-up limitation	5.20	↓ **1.92** ↓	2.61	61%	↓ **58%** ↓	96%	4911	↓ **3481** ↓	4610
2M-up limitation	2.72	↓ **1.81** ↓	2.62	32%	↑ **55%** ↓	95%	2559	↑ **3269** ↓	4621
5M-up limitation	0.97	= **0.97** ↓	2.62	11%	= **11%** ↓	96%	912	≈ **911** ↓	4601
20M-up limitation	0.27	= **0.27** ↓	2.61	3.1%	= **3.1%** ↓	97%	255	≈ **271** ↓	4611
1M-down limitation	0.30	= **0.30** ↓	2.61	3.3%	= **3.3%** ↓	96%	282	≈ **297** ↓	4611

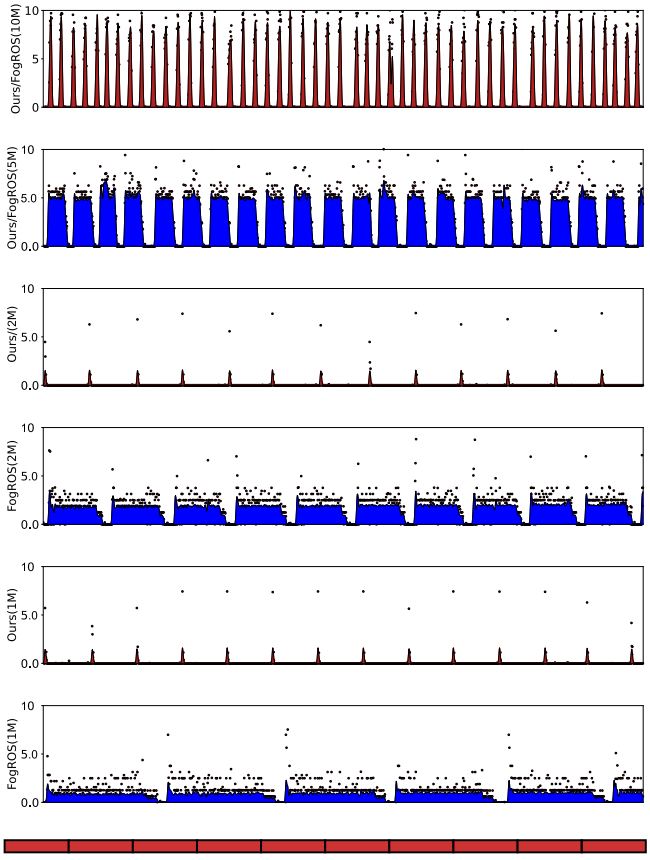


Fig. 13. Comparison of the results of FogROS and RoboEC2. The horizontal axis is the time axis and the vertical axis is the network speed. The color overlay part is fitted to the scatter, and the scale of the vertical axis is after simultaneous adjustment according to the fit. The color overlay also reflects the data transmission volume. The data transmission frequency reflects the system performance. The bottom red line is a fully robot computing rate.

we compare CPU usage and power consumption, noting that these two metrics are relative calculations that take computation frequency into account. The bolded data in the table shows the performance of RoboEC2, and the arrows on its left side indicate the comparison with FogROS and the arrows on the right side indicate the comparison with pure robot computing. Red indicates an increase in the result of the metric where it is located and a decrease in performance; green indicates a decrease in the result of the metric where it is located and an increase in performance; blue indicates a constant result of the metric where it is located and no change in performance. From the table, we can get that RoboEC2 completely improves the latency performance. For CPU usage and power consumption performance, RoboEC2 also improves performance in the vast majority of cases.

TABLE VI
COMPARISON IN DIFFERENT MEASUREMENTS

Matircs	Local	Cloud	SOTA	RoboEC2
Convenience of deployment	-	☹	☹	☺
Workload of revising codes	-	☹	☹	☺
Robustness for location	☺	☹	☺	☺
Robustness for network	☺	☹	☺	☺
Types of tasks	☹	☺	😐	☺
Accuracy	☹	☺	😐	☺
Local cost	☹	☺	😐	😐

[1] Red faces represent bad, yellow faces represent mediocre and green faces represent good.

We can conclude from the SLAM experiments that RoboEC2 is able to perform collaborative computing elastically under different resource conditions, thus achieving computing optimization. It achieves better performance compared to the node-level FogROS.

E. Comparison From System Perspectives

Finally, we analyze four different cloud robotics computing systems in a system perspective based on experimental procedures and inference analysis, including totally local computing, totally offloading to the cloud to compute (PoundCoud and FogROS), the SOTA cloud robotics offloading approach and the proposed RoboEC2. The metrics for the system, as we mentioned in Section III-B, are shown in the first column of the table below. With the experimental procedure, the experimental results and the inference analysis, we are able to obtain the results in the Table. VI. It presents a comprehensive comparison in a system perspective between the totally local computing system, the totally cloud computing

system(PoundCoud and FogROS), the SOTA offloading system and RoboEC2. The first two methods are rather extreme and can achieve optimal performance in terms of computation cost or communication delay, but may lead to operational failure in other metrics. SOTA and RoboEC2 approaches are capable of achieving a balanced execution based on the algorithm's set of criteria. RoboEC2 demonstrates the best overall evaluation when compared to existing approaches.

VI. Conclusion

In this work, we have presented a novel cloud robotic system, RoboEC2, which is designed to provide service-level performance to resource-constrained mobile robots. The key focus of our system is to enable robots to offload computationally expensive tasks, such as perception and decision-making, to cloud servers in order to improve their overall performance and efficiency. To achieve this goal, we have implemented a number of key technologies and algorithms, including a dynamic network offloading approach and the Spotlight Criteria Algorithm, which is responsible for making optimal offloading decisions based on a combination of factors including accuracy, time, and cost. Our system is flexible, convenient, and robust, and is capable of operating under a wide range of conditions, including in disconnected environments. Through our analyses and experiments, we have demonstrated that RoboEC2 outperforms the state-of-the-art approach in a number of metrics. In future work, we plan to further analyze the impact of data loss in communications on the performance of our system.

DNN models typically have two types of structures: topology type and chain type. For chain type DNN models like RNNs, the amount of intermediate data is typically large, so collaborative edge computing is less effective. Most existing work focuses on directly offloading the entire model to the cloud for inference. For example, [28] notes that "given the complexity of RNN-based inference, IoT devices typically offload this task to a cloud server." [41] proposes a fuzzy logic-based strategy to decide whether to offload an RNN-LSTM model to edge, fog or cloud. [14] observes that "Compared to CNNs, RNNs have larger memory footprints, and memory access of the fully connected weight matrices dominates power consumption. RNNs are usually computed on the cloud, which introduces large and variable latency." In summary, the literature generally treats chain type models separately and focuses on offloading the entire model to the cloud. Our work focuses on topology type DNN models in cloud robotics; handling diverse model structures spanning both topology and chain types remains an open challenge. To address this challenge, we are currently working on a new approach that allows us to offload DNN models in the form of a parameter set, which enables us to break free from the structural constraints of DNN models. This approach allows us to handle a wide range of DNN models and to deploy them in a variety of cloud robotics applications. As soon as we complete the development of this approach, we plan to publish a new paper to share our results and to contribute to the field. In addition, we will update our cloud robotics system to version 2.0, which will incorporate this new capability.

References

[1] A. J. B. Ali, M. Kouroshli, S. Semenova, Z. S. Hashemifar, S. Y. Ko, and K. Dantu, "Edge-SLAM: Edge-assisted visual simultaneous localization and mapping," *ACM Trans. Embedded Comput. Syst.*, vol. 22, no. 1, pp. 1–31, Jan. 2023.

[2] P. A. Apostolopoulos, G. Fragkos, E. E. Tsiropoulou, and S. Papavassiliou, "Data offloading in UAV-assisted multi-access edge computing systems under resource uncertainty," *IEEE Trans. Mobile Comput.*, vol. 22, no. 1, pp. 175–190, Jan. 2023.

[3] UC Berkeley. *[EB/OL]*. Accessed: 2023. [Online]. Available: https://setiathome.berkeley.edu/cpu_list.php

[4] T. Carvalho, E. R. S. de Rezende, M. T. P. Alves, F. K. C. Balieiro, and R. B. Sovat, "Exposing computer generated images by eye's region classification via transfer learning of VGG19 CNN," in *Proc. 16th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2017, pp. 866–870.

[5] K. E. Chen et al., "FogROS: An adaptive framework for automating fog robotics deployment," in *Proc. IEEE 17th Int. Conf. Autom. Sci. Eng. (CASE)*, Aug. 2021, pp. 2035–2042.

[6] S. Chinchali et al., "Network offloading policies for cloud robotics: A learning-based approach," *Auton. Robots*, vol. 45, no. 7, pp. 997–1012, Oct. 2021.

[7] Elastic Computing. *[EB/OL]*. Accessed: 2023. [Online]. Available: https://azure.microsoft.com/en-us/overview/what-is-elastic-computing/

[8] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Deep image homography estimation," 2016, *arXiv:1606.03798*.

[9] Analog Devices. (2010). *Fixed-Point vs Floating-Point Digital Signal Processing*. [Online]. Available: https://www.analog.com/en/education/education-library/articles/fixed-point-vs-floating-point-dsp.html

[10] R. Dolbeau, "Theoretical peak FLOPS per instruction set: A tutorial," *J. Supercomput.*, vol. 74, no. 3, pp. 1341–1377, Mar. 2018.

[11] B. A. Erol, S. Vaishnav, J. D. Labrado, P. Benavidez, and M. Jamshidi, "Cloud-based control and vSLAM through cooperative mapping and localization," in *Proc. World Autom. Congr. (WAC)*, Jul. 2016, pp. 1–6.

[12] J. J. Kuffner et al., "Cloud-enabled robots," in *Proc. Int. Conf. Humanoid Robot.*, 2010, pp. 1–2.

[13] C. Fairchild and T. L. Harman, *ROS Robotics by Example*. Birmingham, U.K.: Packt, 2016.

[14] C. Gao, "Energy-efficient recurrent neural network accelerators for real-time inference," Ph.D. thesis, Inst. Neuroinform., Univ. of Zurich, Zürich, Switzerland, 2022.

[15] K. Goldberg and R. Siegwart, *Beyond Webcams: An Introduction to Online Robots*. Cambridge, MA, USA: MIT Press, 2002.

[16] Z. Hong, H. Huang, S. Guo, W. Chen, and Z. Zheng, "QoS-aware cooperative computation offloading for robot swarms in cloud robotics," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 4027–4041, Apr. 2019.

[17] J. Ichnowski, J. Prins, and R. Alterovitz, "Cloud-based motion plan computation for power-constrained robots," in *Algorithmic Foundations of Robotics XII*, vol. 13. USA: Springer, 2020, p. 96.

[18] J. Ichnowski et al., "FogROS 2: An adaptive and extensible platform for cloud and fog robotics using ROS 2," 2022, *arXiv:2205.09778*.

[19] M. Inaba, "Remote-brainde robotics: Interfacing AI with real world behaviors," in *Robotics Research: The Sixth International Symposium*. USA: International Foundation for Robotics Research, 1993, pp. 335–344.

[20] B. Kehoe, A. Matsukawa, S. Candido, J. Kuffner, and K. Goldberg, "Cloud-based robot grasping with the Google object recognition engine," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2013, pp. 4263–4270.

[21] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg, "A survey of research on cloud robotics and automation," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 2, pp. 398–409, Apr. 2015.

[22] B. Liu, L. Wang, and M. Liu, "Lifelong federated reinforcement learning: A learning architecture for navigation in cloud robotic systems," *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, pp. 4555–4562, Oct. 2019.

[23] B. Liu, L. Wang, M. Liu, and C.-Z. Xu, "Federated imitation learning: A novel framework for cloud robotic systems with heterogeneous sensor data," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 3509–3516, Apr. 2020.

[24] B. Liu, L. Wang, X. Chen, L. Huang, D. Han, and C.-Z. Xu, "Peer-assisted robotic learning: A data-driven collaborative learning approach for cloud robotic systems," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 4062–4070.

[25] A. Manzi et al., "Design of a cloud robotic system to support senior citizens: The KuBo experience," *Auton. Robots*, vol. 41, no. 3, pp. 699–709, Mar. 2017.

[26] R. C. Mello et al., "The PoundCloud framework for ROS-based cloud robotics: Case studies on autonomous navigation and human–robot interaction," *Robot. Auton. Syst.*, vol. 150, Apr. 2022, Art. no. 103981.

[27] IEEE Technical Committee on Networked Robotics. *[EB/OL]*. Accessed: 2023. [Online]. Available: http://www-users.cs.umn.edu/~isler/tc/

[28] D. J. Pagliari, R. Chiaro, Y. Chen, E. Macii, and M. Poncino, "Optimal input-dependent edge-cloud partitioning for RNN inference," in *Proc. 26th IEEE Int. Conf. Electron., Circuits Syst. (ICECS)*, Nov. 2019, pp. 442–445.

[29] A. Rahman, J. Jin, A. L. Cricenti, A. Rahman, and A. Kulkarni, "Communication-aware cloud robotic task offloading with on-demand mobility for smart factory maintenance," *IEEE Trans. Ind. Informat.*, vol. 15, no. 5, pp. 2500–2511, May 2019.

[30] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.

[31] V. K. Sarker, J. P. Queralta, T. N. Gia, H. Tenhunen, and T. Westerlund, "Offloading SLAM for indoor mobile robots with edge-fog-cloud computing," in *Proc. 1st Int. Conf. Adv. Sci., Eng. Robot. Technol. (ICASERT)*, May 2019, pp. 1–6.

[32] Amazon Web Service. *[EB/OL]*. Accessed: 2023. [Online]. Available: https://aws.amazon.com/

[33] D. Spatharakis, M. Avgeris, N. Athanasopoulos, D. Dechouniotis, and S. Papavassiliou, "Resource-aware estimation and control for edge robotics: A set-based approach," *IEEE Internet Things J.*, vol. 10, no. 3, pp. 2003–2020, Feb. 2023.

[34] N. Tian et al., "A fog robotic system for dynamic visual servoing," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 1982–1988.

[35] S. Tuli, N. Basumatary, and R. Buyya, "EdgeLens: Deep learning based object detection in integrated IoT, fog and cloud computing environments," in *Proc. 4th Int. Conf. Inf. Syst. Comput. Netw. (ISCON)*, Nov. 2019, pp. 496–502.

[36] J. V. Hook et al., "Multirobot onsite shared analytics information and computing," *IEEE Trans. Control Netw. Syst.*, vol. 10, no. 1, pp. 169–181, Mar. 2023.

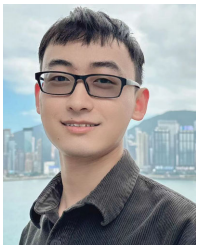
[37] M. Waibel et al., "A world wide web for robots," *IEEE Robot. Automat. Mag.*, vol. 18, no. 2, pp. 69–82, Jan. 2011.

[38] Wiki. *[EB/OL]*. Accessed: 2023. [Online]. Available: https://en.wikipedia.org/wiki/Secure_Shell

[39] Y. Xie, Y. Guo, Z. Mi, Y. Yang, and M. S. Obaidat, "Loosely coupled cloud robotic framework for QoS-driven resource allocation-based web service composition," *IEEE Syst. J.*, vol. 14, no. 1, pp. 1245–1256, Mar. 2020.

[40] Z. Xu and V. Tzoumas, "Resource-aware distributed submodular maximization: A paradigm for multi-robot decision-making," in *Proc. IEEE 61st Conf. Decis. Control (CDC)*, Dec. 2022, pp. 5959–5966.

[41] N. Zenasni, C. Habib, and J. Nassar, "A fuzzy logic based offloading system for distributed deep learning in wireless sensor networks," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2022, pp. 1–8.



Boyi Liu (Member, IEEE) received the bachelor's degree (Hons.) from Hainan University and the M.Phil. degree from the University of Chinese Academy of Science. He is currently pursuing the Ph.D. degree with the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong. His research interests include robotics and federated learning. He received the Best Graduation Honor of "Student of the Year" Award for the bachelor's degree. He was the Winner of the Outstanding Paper Award at the National Conference of Theoretical Computer Science in 2016.

Lujia Wang (Member, IEEE) received the Ph.D. degree from the Department of Electronic Engineering, The Chinese University of Hong Kong, Hong Kong, in 2015. From 2015 to 2016, she was a Research Fellow with the School of Electrical Electronic Engineering, Nanyang Technological University, Singapore. From 2016 to 2021, she was an Associate Professor with the Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen, Guangdong, China.

She is currently a Research Assistant Professor with the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong. Her current research interests include cloud robotics, lifelong federated robotic learning, resource/task allocation for robotic systems, and applications on autonomous driving.

Ming Liu (Senior Member, IEEE) received the B.A. degree in automation from Tongji University, Shanghai, China, in 2005, and the Ph.D. degree from the Department of Mechanical and Process Engineering, ETH Zürich, Zürich, Switzerland, in 2013. During the master's degree, he was with Tongji University, he stayed one year with Erlangen-Nüremberg University, Erlangen, Germany, and the Fraunhofer Institute IISB, Erlangen, as a Master Visiting Scholar. He is currently with the Department of Electronic and Computer Engineering, the Department of Computer Science and Engineering, and the Robotics Institute, The Hong Kong University of Science and Technology, Hong Kong. His research interests include dynamic environment modeling, deep learning for robotics, 3-D mapping, machine learning, and visual control.

He was a recipient of the Best Student Paper Award at the IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems in 2012, the Best Paper in Information Award at the IEEE International Conference on Information and Automation in 2013, the Best RoboCup Paper at the IEEE/RSJ International Conference on Intelligent Robots and Systems in 2013, and twice the Winning Prize of the Chunhui-Cup Innovation Contest.