Collaborative Robot Transport System Based on Edge Computing

Zhun Fan, Wenzhao Chen, Guijie Zhu, Yugen You, Furong Deng, Youzhao Hou,
Weixiang Liang, Runzhan Fu, Jiaming Xin, Jingming Chen and Hongmin Wang*

***Abstract*— Collaboration between heterogeneous robots promises increased robustness and efficiency of tasks with great potential in applications, such as manufacturing, search and rescue. In this paper, we present a collaboration system based on an edge computing framework, a mobile robot named "StellaX" and a manipulator. The robot StellaX is assembled by a three-wheel omnidirectional platform, a LIDAR scanner for indoor autonomous navigation, and a stereo camera which can capture objects' images. The information of objects, which can be used for grasping, is stored and shared with the manipulator through the edge computing framework with Docker and ROS. Finally, we test the collaboration system by conducting an object recognition experiment and a navigation experiment. The experimental results demonstrate the system effectiveness and prove that the system yields better real-time performance and reduces total task execution time than a cloud computing-based scenario.**

I. INTRODUCTION

Robotics collaboration such as cooperative transport [1], search and rescue [2], collaborative SLAM [3][4][5] can integrate the capabilities of different robots to accomplish complex tasks or achieve higher efficiency. The basis of robotics collaboration is to ensure that different robots can communicate with each other so that they can share information and assign tasks.

In 2009, the RobotEarth project was announced [6]. It envisioned "a World Wide Web for robots". On this basis, the RoboEarth research team developed a series of system architectures such as Davinci [7], KnowRob [8] and C2TAM [9]. In 2010, James Kuffner [10] proposed the term "cloud robotics" and described a number of potential benefits of "Cloud Robotics". The RoboEarth project includes a cloud computing platform called Rapyuta [11], which is a platform as a service (PaaS) framework for offloading computing tasks of robots from the local into the cloud. Wen et al. [12] proposed a micROS cloud platform, which allowed general ROS applications migration to the cloud environment. ROS applications can be changed from serving a single robot to serving multi-robot result from the container-based isolation mechanism.

Since the cloud robotics solves the problem that the standard robot has the insufficient computing power and low intelligence. Cloud robotics has its disadvantages, such as high network latency and limited bandwidth. Furthermore, heterogeneous robots have different systems, platforms, interfaces or communication protocols, which means they form a heterogeneous environment that has challenges in sharing information and interacting with different types of physical robots.

In this context, edge computing [13] can be used as a supplement to cloud robotics to extend its limitations. The "edge" in edge computing refers to any computing or network resource that is closer to the data source on network topology. The benefit of an edge computing system is that it can offload a portion of computational tasks to the edge node rather than the distant cloud center. Data processing at the edge of the network reduces network latency and makes it more efficient. Moreover, the edge node can be used as an intermediate layer to integrate the devices' interfaces of the heterogeneous robots and then communicate with the cloud center. Recently, edge computing has been well studied by the researchers in many different usage scenarios like video analytics [14][15][16] and firefighting [17]. Youdong Chen et al. [18] proposed a cloud-edge hybrid system framework and applied the system on the robotic welding of the membrane wall cell. However, few efforts have been made to study collaborative robot systems based on edge computing.

The contributions of this paper are listed as follows:

1) Design and implement a collaboration system which consists of two heterogeneous robots, the mobile robot StellaX and a six-degree-of-freedom(DoF) manipulator, as shown in figure 1 and figure 7.
2) An edge computing-based system named Docker Edge Robotics Framework (DERF), as shown in figure 6 is proposed for communication and task assignment between heterogeneous robots.
3) The experimental results proved that in the tasks of object detection, indoor autonomous navigation, the edge computing-based framework has lower latency and faster response than the cloud computing-based framework.

The remainder of this paper is organized as follows. The design and implementation of the mobile robot StellaX and the manipulator are presented in Section II. Section III introduces the Docker Edge Robotics Framework, in Section IV the experimental scenario is described and our experimental results are presented and analyzed. We summarize our works in Section V.

*Corresponding Author: Hongmin Wang is a Postdoctoral with the college of science, Shantou University, Shantou 515063, China, and also a Teacher with the Department of Intelligent Manufacturing, Wuyi University, Jiangmen 529020, China. Email: whm@stu.edu.cn

All authors are with the Guangdong Provincial Key Laboratory of Digital Signal and Image Processing, College of Engineering, Shantou University, Shantou 515063, China.

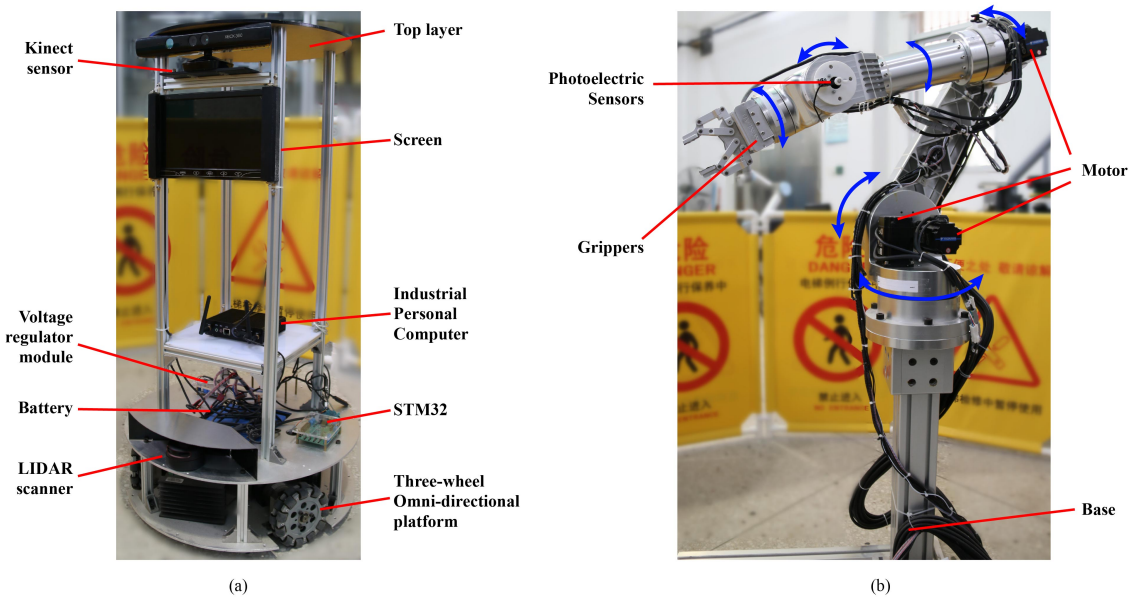

Fig. 1. The physical diagram of two heterogeneous robots. **(a)** The mobile robot StellaX **(b)** The 6-DoF manipulator.

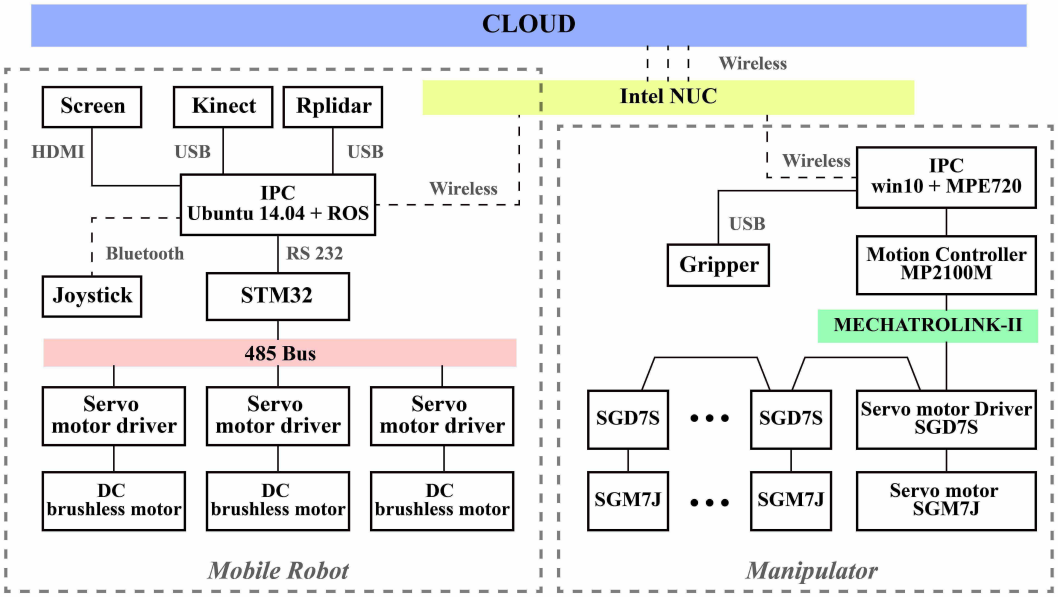

Fig. 2. The overall framework of the collaboration system.

II. ROBOT SYSTEM DESIGN

Figure 1 and figure 2 shows the physical diagram and system framework of the collaboration system respectively. The detail of each robot design and its basic function is introduced in the following subsection.

A. Mobile Robot

The mobile robot StellaX is composed of a three-layer architecture with a three-wheel omnidirectional mobile robot(TOMR) [19][20] at the bottom layer. TOMR is considered here due to its versatility and flexibility. In contrast to differential drive wheeled mobile robot, TOMR can move in an arbitrary direction and with arbitrary orientation, since they move using three independent motor actuators. The key performance parameters of StellaX are listed as table I.

B. Manipulator

The manipulator is a six-degree-of-freedom (DoF) robot. Each joint of the robot is equipped with a photoelectric sensor for performing a zero return operation after the robot restarts. The Yaskawa motor SGM7J and the driver SGD7S are used for motion, the control card MP2100M and a personal computer (PC) installed with software MPE720 form a control platform. Based on that, the posture of the manipulator can be programmed by the PC and transmitted to the driver, which generates motion signals to control motor motion. The key performance

TABLE I

KEY PERFORMANCE PARAMETERS OF STELLAX.

Total weight	$28kg$
Robot height	$585mm$
Wheel diameter	$127mm$
Battery life	$4h$
Max. diameter of the robot	$528mm$
Max. payload of the robot	$60kg$
Max. velocity of the robot	$1.2m/s$
Communication Interface	$RS232$
SLAM method	$LiDAR-based$

TABLE II

KEY PERFORMANCE PARAMETERS OF THE MANIPULATOR.

Total length of the manipulator	$850mm$
Max. active angle of each joint	$270°$
Max. working speed	$180°/s$
Repeatability	$\pm 0.1mm$
Max. payload of the manipulator	$5kg$

parameters of the manipulator are listed as table II.

C. Indoor Navigation

In order to accomplish the transport task, the mobile robot StellaX needs to navigate autonomously indoors. In this work, we built a ROS-based indoor SLAM system, the system interface is shown in Figure 3. We use the wheel odometer and LIDAR data as input to map the environment and locate the robot's position. Rao-Blackwellized particle filter algorithm and *amcl* package are used here for localization, while the *gmapping* package is used to create the two-dimensional occupancy grid map. Based on the grip map, the trajectory between the origin and the destination is solved by the Dijkstra algorithm and *movebase* package. When the robot encounters an obstacle during the motion, the Dynamic Window Approach (DWA) algorithm will replan a new path.

D. Object Detection and Grasping

Object detection plays an important role for robot grasping since automated grasping requires both knowing what the object is and where the object is. In this context, we use the method [21] that we have proposed before to accomplish the detection and grasping task. The difference between these two works lies mainly in the method of object detection. In this paper, we use deep learning method instead of a feature-based approach. More precisely, our method based on Faster R-CNN [22] and SSD model [23]. A batch of labeled data was provided to the SSD model for training as a training set. After the training is completed, RGB images provided by Kinect were input into the model and the bounding box of the target object can be obtained, which means the two-dimensional coordinate of the target object on the image can be calculated. Using this coordinate to perform depth extraction on the depth image, the three-dimensional coordinate in the camera coordinate system can be obtained. After coordinate transformation and inverse kinematics solution, the manipulator can perform the grasping task. The overview of the manipulator grasping system is shown in figure 4.

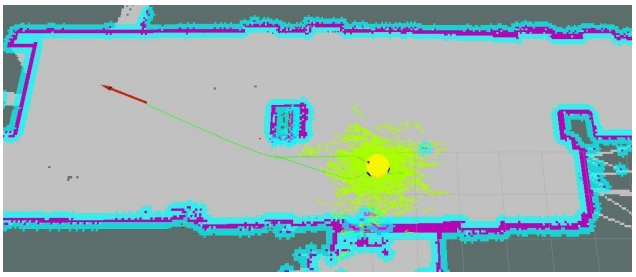

Fig. 3. The diagram of the StellaX navigate autonomously indoors.

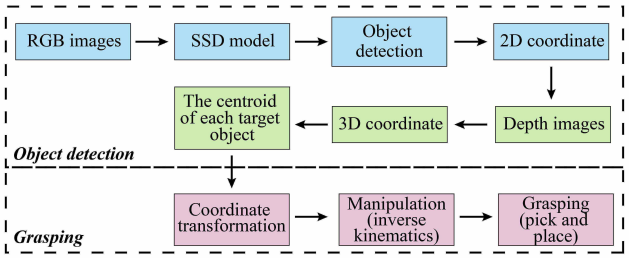

Fig. 4. The overview of the manipulator grasping system.

III. DOCKER EDGE ROBOTICS FRAMEWORK

A. Edge Computing

One of the cores of cloud robotics is cloud computing. Since cloud computing was introduced in 2005, it has gradually changed the way we live, work and study. The services provided by Alipay, Twitter, Wechat and other software that have been widely used in our daily life are typical representatives of software as a service (SaaS), which is a kind of cloud computing. But cloud computing is facing some challenging problems in cloud robotics:

1) The cloud center has the powerful computing power and can handle massive data. But how to deal with the transmission of massive data is a knotty problem which means the performance of cloud computing models is limited by network bandwidth. The time of data transmission and data processing is not to meet the real-time requirements of robots.
2) The role of terminal devices has changed. In traditional cloud computing, phones and PC play data consumers most of the time, such as making requests for services like payment or video. But in cloud robotics, robots generate data in addition to consume data. This means that the number of data generation nodes will increase, and this data cannot be processed in time in a traditional cloud computing model.

In order to solve the above problem, we try to apply the idea of edge computing in the cloud robotics. Edge computing refers to the analysis of data processing at the edge nodes of the network. In this context, an edge node refers to any node with computing power or network

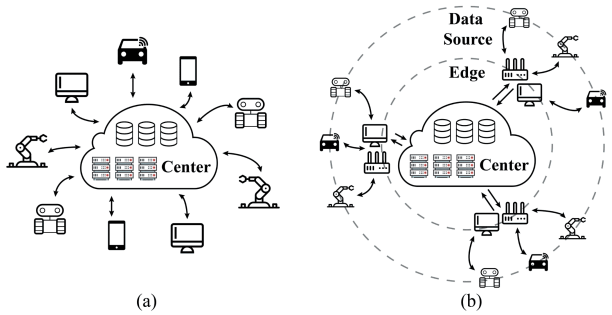

Fig. 5. The difference between cloud computing and edge computing. **(a)** Traditional cloud computing model. **(b)** Edge computing model.

resources between the data source and the cloud center. For example, our phones, PC, and routers can be regarded as edge nodes. The difference between cloud computing and edge computing is shown in Figure 5. The advantage of edge computing is that massive data does not need to be transferred over long distances, which can reduce network traffic and significantly improve the response speed.

B. Docker

Heterogeneous robots usually have different operating systems, interfaces and communication protocols, which mean their applications do not have a unified operating environment. Setting a fixed operating environment in the cloud center would be detrimental to application deployment and computing resource scheduling. In this work, we use the idea of Docker to deploy robots' application in the cloud center. Docker is a container engine technology based on Linux container (LXC), which has the advantage of rapid and efficient deployment, high resource utilization and simple management [24]. With these properties of Docker, developers can ignore the operating environment of the cloud center and be focused on the interfaces and protocols of the corresponding robot. Furthermore, the containers are non-interacting, and the resources of each container can be allocated as needed before instantiation. That means running Docker in the cloud center or edge node not only ensures independence and security between different applications but also enables flexible allocation of computing resources.

C. Docker Edge Robotics Framework

Based on the two techniques mentioned above, Edge computing and Docker, we designed a Docker Edge Robotic Framework (DERF) to connect heterogeneous robots, so they can collaborate on specific tasks. The diagram of DERF is shown in Figure 6.

The DERF consists of five parts. In detail, they are service provider, service requester, cloud center, edge node, and physical layer. The service provider is mainly for developers. When the developer develops a corresponding application according to the interface and protocol of the robot, the application will be packaged into a Docker container image and registered with the cloud center via the registration module as an optional service. The user

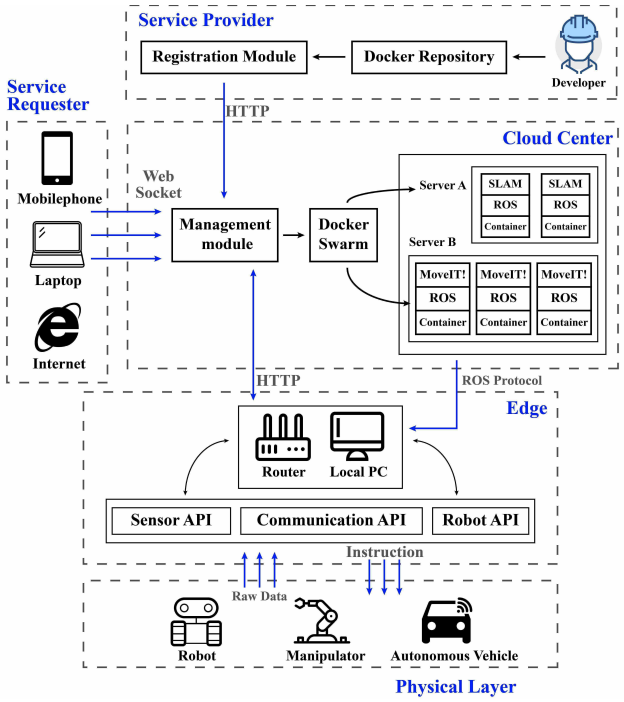

Fig. 6. The diagram of DERF(Docker Edge Robotics Framework).

can initiate a robot service request through the network as the requester. After receiving the service request, the management module in the cloud center first performs a service query in the image repertory to confirm that the service exists and the robot is in idle, and then the image is instantiated to form a functional application which will send a task instruction to the physical layer via the edge node. During the process of task execution, the environment information will be collected and the robot will generate massive data as a data source. These data will be pre-processed on the edge device, such as compression, filtering and then sent to the cloud center for calculation. For some simple processing, the edge device will return the result directly. The DERF realizes the design concept of robot as a service (RaaS) and has the advantages of heterogeneous robot information sharing, low latency and low local computing resource requirements.

IV. EXPERIMENT

In order to verify the collaboration system, as shown in Figure 7. We conduct object detection experiments and indoor navigation experiments under local computing, cloud computing, and edge computing respectively. Figure 8 shows the diagram of the experimental network topology, here we use the method mentioned by Sami Salama et al.[25] to connect the different ROS applications in different network environments.

In this experiment, Aliyun cloud center configuration is 2 vCPU(Intel Xeon E5-2682v4), 4G memory and 10M bandwidth. The mobile robot StellaX is equipped with a Raspberry Pi 3B+. The edge device here is a spare Intel NUC5I7RYH with dual-core CPU i7-5557U, 8G memory, and 250G SSD.

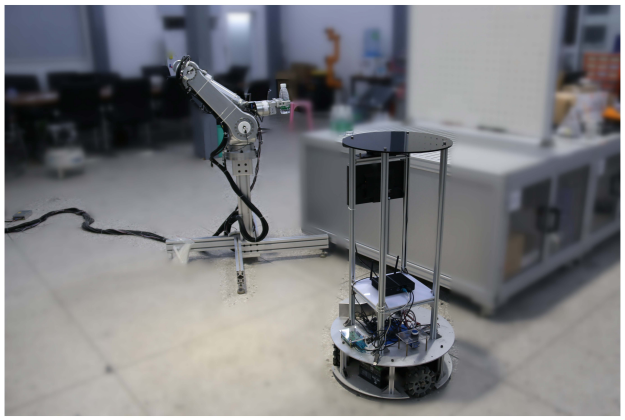

Fig. 7. Experimental environment: StellaX observes the position of the object and shares the information with the manipulator through the Intel NUC. After the manipulator performs the grasping task and places the object on the top of StellaX, autonomous navigation application will be execution.

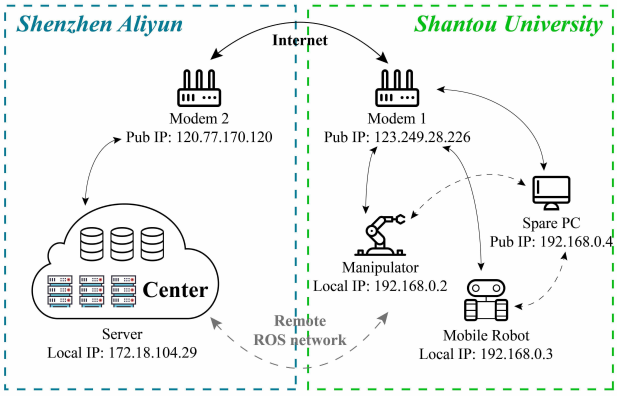

Fig. 8. The diagram of the experimental network topology.

A. Object Detection Experiment

In the grasping experiment, the manipulator needs the position information of objects before grasping. In order to ensure the effectiveness of the task, the manipulator movement program is run locally in all experiments. The only difference is that the object detection program runs on different devices. As shown in Figure 9 (a) and (b), the object detection program is run on the Aliyun server in the cloud computing experiment. In contrast, the object recognition program is run on Intel NUC in the edge computing experiment.

We placed the objects to be detected at a different position on the base to form five sets of experiments, each of which will be repeated 20 times. Figure 10 shows the comparison of total execution time spent on object detection experiments in different network environments. The average total execution time in local computing, cloud computing, and edge computing is 7.43s, 2.61s, and 1.43s. Compared to local computing, both cloud computing and edge computing reduce the total execution time effectively, with percentage reductions of 64.86% and 81.91% respectively. In addition, the time in edge computing is reduced by 48.53% compared to cloud computing.

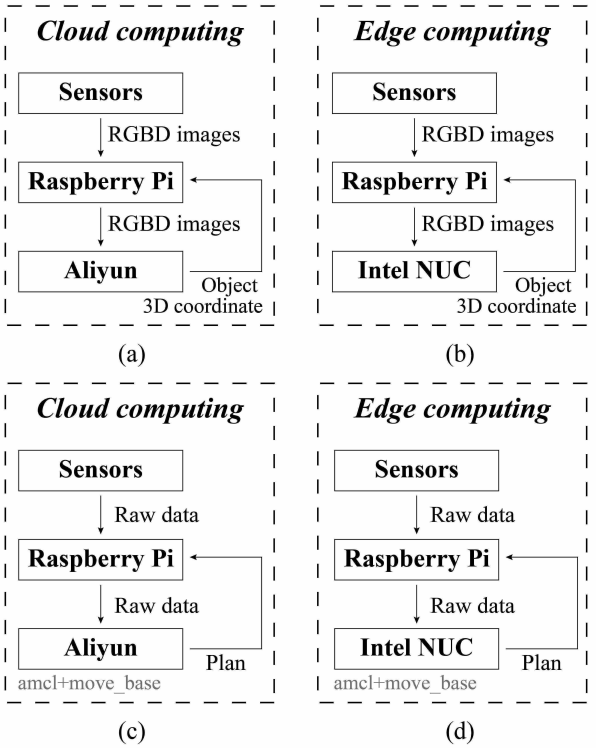

Fig. 9. The flow chart of the object recognition experiment and the navigation experimental. **(a)** Object detection experiment in cloud computing. **(b)** Object detection experiment in edge computing. **(c)** Indoor navigation experiment in cloud computing. **(d)** Indoor navigation experiment in edge computing.

Figure 11 shows the time spent on data uploading in cloud computing and edge computing experiment. The edge computing takes less time to upload data compared to cloud computing in all experiments. The average percentage reduction is 57.56%. This is because the edge device NUC is closer to the data source Raspberry Pi 3B+ than the Aliyun server in Shenzhen, which reduces the number of jumps required for data uploading, resulting in a faster response and lower latency for the entire object detection task.

In the object detection experiment, the average execution time of cloud computing is 2.6117s, and the edge calculation is 1.3441s, which is 1.2676s less than cloud computing. In the process of uploading the original data, the average time of cloud computing is 2.0868s, and the edge calculation is 1.1528s, which is 0.934s less than cloud computing. 73.68% of the total time saved for task execution is reflected in the time saved for data upload. On the task that the data uploading takes more time than the data processing, edge computing obviously has better performance than cloud computing, which is reflected in its low latency and Quick response.

Figure 12 shows the average computing resource occupancy of the Raspberry Pi in different experiments. In cloud computing and edge computing, the neural network for object detection runs on Aliyun servers or edge devices, resulting in CPU and Memory usage can be reduced effec-

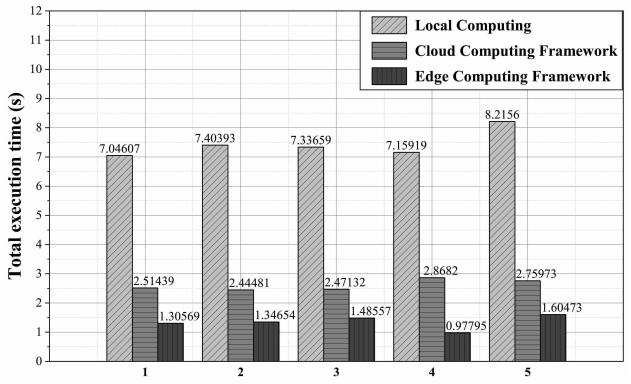

Fig. 10. Comparison of task execution time in object detection experiments.

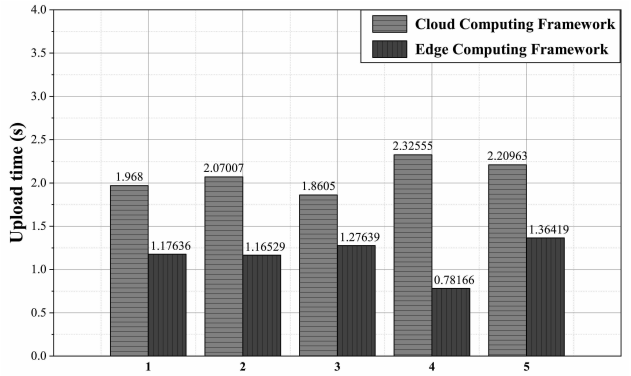

Fig. 11. Comparison of data upload time in object detection experiments.

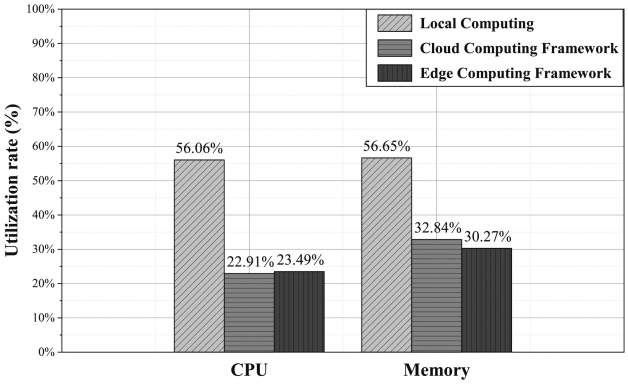

Fig. 12. Comparison of computing resource occupancy in object detection experiments.

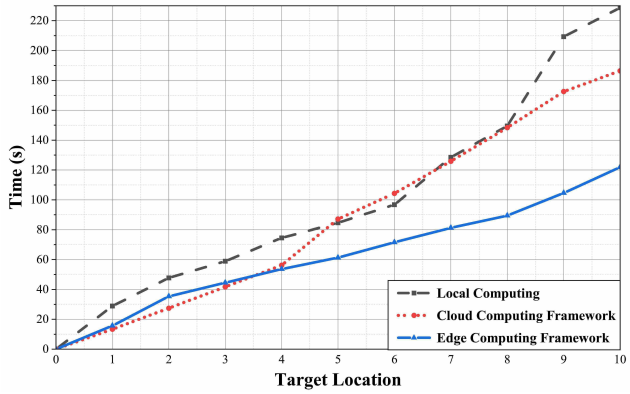

Fig. 13. Comparison of task execution time in indoor navigation experiments.

tively compared to local computing. In cloud computing, the average CPU usage of the Raspberry Pi is 22.91%, and the average memory usage is 32.84%, which is 59.12% and 42.02% lower compared to local computing. In edge computing, the average CPU usage of the Raspberry Pi is 23.49%, and the average memory usage is 30.27%, which is 58.09% and 46.56% lower compared to local computing. The results prove that both cloud computing and edge computing can offload tasks from local and reduce the consumption of onboard computing resources effectively.

B. Indoor Navigation Experiment

In the navigation experiment, we set eleven target points indoors, and the coordinates and access order of the target points are fixed. The mobile robot StellaX will start from the starting point and navigate to these target in proper order. This experiment will be repeated five times in different network environments and the time using between every two points will be recorded. Figure 9 (c) and (d) shows the different process in cloud computing and edge computing. The *amcl* and *movebase* package are run on different device like Aliyun server or Intel NUC. Figure 13 shows the total time spent on navigation in a different experiment. The results show that the shortest using time is edge computing, followed by cloud computing, and the longest using time is local computing. The corresponding average time used is 121.86s, 186.41s, and 228.71s.

Local computing runs the *amcl* and the *movebase* in ROS with limited computing resources. When the robot drifts due to the odometer error during navigation, it needs to be rotated to relocation itself. This process actually relies on environmental information input by LIDAR or camera to recalculate the probability of robot appearing in different locations in the environment. This is a computationally intensive task which takes a long time to calculate when computing resources are limited, resulting in the longest time spent on the entire navigation task.

In cloud computing, data processing has the hysteresis quality due to the high latency between the Raspberry Pi and the Aliyun server. The robot is prone to an emergency stop or a circle during the navigation. This is because the robot keeps the current speed until new commands are received. When the new command is delayed, the robot will continue to move in the current direction, causing it to pass the target or deviate from the original path. The robot needs to re-correct its current location constantly, resulting in a longer navigation process.

In edge computing, the Raspberry Pi and the edge device NUC are in the same LAN with lower latency. In addition, NUC has more computing resources than the Raspberry Pi, which means that the robot can calculate its location in real time and received correct commands to complete the navigation task in the shortest time. The results show that in tasks with real-time requirements, such as SLAM,

navigation, etc., running on the edge device can control the delay in a lower range and improve the response speed of tasks.

V. CONCLUSION

In this paper, we design and implement two heterogeneous robots which are a mobile robot "StellaX" and a six-degree-of-freedom (DoF) manipulator. We also proposed a system based on Docker and edge computing called Docker Edge Robotic Framework (DERF). The system DERF includes Docker container isolation ideas and the idea of offloading computational tasks to edge nodes in edge computing. The collaboration system with DERF and the two heterogeneous robots has the advantages of information sharing, low latency, easy deployment, and low local computing resource requirements. Finally, we demonstrated the effectiveness of the collaboration system through an experiment. StellaX transported the object to a specific location after the manipulator grasped the bottle and placed it on the top of StellaX. We also designed two experiments to verify the feasibility of edge computing in the DERF. The robot performs object detection and autonomous navigation tasks in local computing, cloud computing, and edge computing. The results show that edge computing in the DERF can offload local computing tasks to the edge devices, reduce the consumption of local computing resources. Compared to cloud computing, edge computing in DERF reduce network latency, improve response speed and reduce total task execution time.

In our future work, we will combine the advantages of cloud computing and edge computing with an edge-cloud hybrid framework to test the performance of heterogeneous robots. Simply put, we will put computationally intensive tasks in the cloud for calculations. For tasks with high real-time requirements, we will perform calculations at the edges. For computationally intensive tasks that require high real-time performance, such as 3D reconstruction, outdoor autonomous navigation, behavioral analysis, and more. They will be pre-processed in the edge first and then transferred to the cloud for calculations. The method of preprocessing will depend on the tasks to be performed by the heterogeneous robots system.

REFERENCES

[1] J. Chen, M. Gauci, W. Li, A. Kolling, and R. Groß, "Occlusion-based cooperative transport with a swarm of miniature mobile robots," *IEEE Transactions on Robotics*, vol. 31, no. 2, pp. 307–321, 2015.

[2] Y. Liu and G. Nejat, "Multirobot cooperative learning for semiautonomous control in urban search and rescue applications," *Journal of Field Robotics*, vol. 33, no. 4, pp. 512–536, 2016.

[3] G. Mohanarajah, V. Usenko, M. Singh, R. D. Andrea, and M. Waibel, "Cloud-based collaborative 3d mapping in real-time with low-cost robots," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 2, pp. 423–431, 2015.

[4] P. Fankhauser, M. Bloesch, P. Krüsi, R. Diethelm, M. Wermelinger, T. Schneider, M. Dymczyk, M. Hutter, and R. Siegwart, "Collaborative navigation for flying and walking robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2016, Conference Proceedings.

[5] P. Schmuck and M. Chli, "Ccm-slam: Robust and efficient centralized collaborative monocular simultaneous localization and mapping for robotic teams," *Journal of Field Robotics*, pp. 1–19, 2018.

[6] M. Waibel, M. Beetz, J. Civera, R. d'Andrea, J. Elfring, D. Galvez-Lopez, K. Häussermann, R. Janssen, J. Montiel, A. Perzylo *et al.*, "Roboearth," *IEEE Robotics & Automation Magazine*, vol. 18, no. 2, pp. 69–82, 2011.

[7] R. Arumugam, V. R. Enti, L. Bingbing, W. Xiaojun, K. Baskaran, F. F. Kong, A. S. Kumar, K. D. Meng, and G. W. Kit, "Davinci: A cloud computing framework for service robots," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 3084–3089.

[8] M. Tenorth and M. Beetz, "Knowrob: A knowledge processing infrastructure for cognition-enabled robots," *The International Journal of Robotics Research*, vol. 32, no. 5, pp. 566–590, 2013.

[9] L. Riazuelo, J. Civera, and J. M. Montiel, "C2tam: A cloud framework for cooperative tracking and mapping," *Robotics and Autonomous Systems*, vol. 62, no. 4, pp. 401–413, 2014.

[10] J. Kuffner, "Cloud-enabled humanoid robots," in *Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on, Nashville TN, United States, Dec.*, 2010.

[11] G. Mohanarajah, D. Hunziker, R. D'Andrea, and M. Waibel, "Rapyuta: A cloud robotics platform," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 2, pp. 481–493, 2015.

[12] S. Wen, B. Ding, H. Wang, B. Hu, H. Liu, and P. Shi, "Towards migrating resource-consuming robotic software packages to cloud," in *Real-time Computing and Robotics (RCAR), IEEE International Conference on*. IEEE, 2016, pp. 283–288.

[13] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[14] B. Qi, L. Kang, and S. Banerjee, "A vehicle-based edge computing platform for transit and human mobility analytics," in *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*. ACM, 2017, p. 1.

[15] S. Yi, Z. Hao, Q. Zhang, Q. Zhang, W. Shi, and Q. Li, "Lavea: Latency-aware video analytics on edge computing platform," in *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*. ACM, 2017, p. 15.

[16] G. Grassi, K. Jamieson, P. Bahl, and G. Pau, "Parkmaster: An in-vehicle, edge-based video analytics service for detecting open parking spaces in urban environments," in *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*. ACM, 2017, p. 16.

[17] X. Wu, R. Dunne, Q. Zhang, and W. Shi, "Edge computing enabled smart firefighting: opportunities and challenges," in *Proceedings of the Fifth ACM/IEEE Workshop on Hot Topics in Web Systems and Technologies*. ACM, 2017, p. 11.

[18] Y. Chen, Q. Feng, and W. Shi, "An industrial robot system based on edge computing: An early experience," in {*USENIX*} *Workshop on Hot Topics in Edge Computing (HotEdge 18)*, 2018.

[19] F. G. Pin and S. M. Killough, "A new family of omnidirectional and holonomic wheeled platforms for mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 10, no. 4, pp. 480–489, 1994.

[20] G. Indiveri, "Swedish wheeled omnidirectional mobile robots: Kinematics analysis and control," *IEEE Transactions on Robotics*, vol. 25, no. 1, pp. 164–171, 2009.

[21] Z. Fan, Z. Li, W. Li, Y. You, W. Chen, and C. Li, "A combined texture-shape global 3d feature descriptor for object recognition and grasping," in *2017 International Conference on Industrial Informatics-Computing Technology, Intelligent Technology, Industrial Information Integration (ICIICII)*. IEEE, 2017, pp. 47–54.

[22] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[23] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[24] D. Bernstein, "Containers and cloud: From lxc to docker to kubernetes," *IEEE Cloud Computing*, vol. 1, no. 3, pp. 81–84, 2014.

[25] S. S. H. Hajjaj and K. S. M. Sahari, "Establishing remote networks for ros applications via port forwarding: A detailed tutorial," *International Journal of Advanced Robotic Systems*, vol. 14, no. 3, p. 1729881417703355, 2017.