An Industrial Robot System Based on Edge Computing: An Early Experience

Youdong Chen and Qiangguo Feng
School of Mechanical Engineering and Automation, Beihang University, Beijing 100191, China
Weisong Shi
Department of Computer Science, Wayne State University, USA, weisong@wayne.edu

Abstract

As more sensors and actuators are deployed in industrial manufacturing, the industry requires a new production system architecture that offers better real-time and network transmission performance. Yet cloud computing (based on a centralized datacenter) is limited in its possibilities, because it suffers from heavy bandwidth costs and lengthy time delays. As a solution, we propose an industrial robot system based on edge computing. Here, we present its three-layer architecture in detail: the cloud, edge, and physical resource layers. Initially, we deploy an edge node near the data sources, to integrate various devices' interfaces and acts as a raw data filter. Then, we apply the proposed system on the robotic welding of the membrane wall cell. Finally, we test the system by conducting an experiment. The results demonstrate the system's feasibility and prove that the system yields better real-time and network transmission performance than a cloud-based scenario.

Keywords: edge computing; cloud computing; membrane wall welding; industrial robot system

1. Motivation

As cloud computing sweeps along, it has quickly rolled across the virtual world, permeating into the new multidisciplinary research area of *cloud robotics* [1,2,3]. Cloud robotics, a concept extended from remote-brained robot [4] and networked robot systems [5], has overcome the limitations of standalone robots by integrating robots with Internet and cloud technologies. Cloud robotics transfers the computing process's complexity—along with its massive storage requirements'—to the cloud. Also, because cloud robotics utilizes a high-speed network, robots can interact fluently with the cloud. These significant benefits are a few of the reasons that researchers have explored cloud robotics in recent years (see, for example, RoboEarth [6,7], DavinCi [8], Rapyuta [9], UNR-PF [10], and Robot Cloud [11]).

Current cloud robotic systems aim to improve robot performance with the help of enhanced computing power and data- or knowledge-management capabilities from the cloud. However, cloud computing has its limitations, and real-time demand largely affects the overall performance [12]. For example, to operate an industrial robot properly, it requires synchronization below 1μs with cycle times between 2 us down to 50 μs. Furthermore, manufacturing facilities use different standards, platforms, and communication protocols or interfaces [13,14]. Thus, it forms a heterogeneous environment that experiences challenges in interacting with and integrating diverse types of physical devices.

With the Internet of Things' (IoTs') rapid development, cloud computing and big data have changed and impacted the need for edge computing profoundly. In *edge computing*, data processing occurs partly at the network's edge [15]. The "edge" refers to any computing and network resources along the path between data sources and the cloud [12,15]. In some papers, nodes near the network's edge are also called *cloudlets* [16], *micro datacenters*, or *fog nodes* [17,18,19]. In terms of IoT, some physical resources have computational and storage power [20] and the "edge" includes end devices that convey real-time data from smart products [21]. The advantage of an edge computing system is that it can offload some computationally extensive tasks to the edge node rather than a distant cloud. Furthermore, the edge node can integrate homogeneous devices' interfaces. So, with the benefits above mentioned, edge computing overcomes the drawbacks of cloud robotics.

With this in mind, here we propose a robot system that merges edge computing technology and robotic applications in manufacturing. Section 2 presents the system architecture. Sections 3 and 4 introduce a robotic welding use case scenario and related experiments. Finally, Section 5 concludes the paper.

2. Architecture

The architecture of our robot system, based on edge computing (see Fig. 1), is comprised of cloud, edge, and physical resource layers, from top to bottom. The cloud layer is the system's core, and it is in charge of performing the computing or storage-intensive tasks. The physical resource layer interacts with the real world and is the data sources. Sensors and diverse smart meters are distributed in the robot cell, sensing the environment and emitting observed values to upper layers via gateways for further processing. Robot tasks assigned in the cloud

are taken via robot-as-a-service (RaaS) units. The robot controller interprets the robot task documents from the cloud into process-working steps, and eventually it directly drives the robot. The edge layer filters and processes the raw data dynamically generated by sensors and then uploads the results to the cloud; this drastically reduces response time and network traffic.

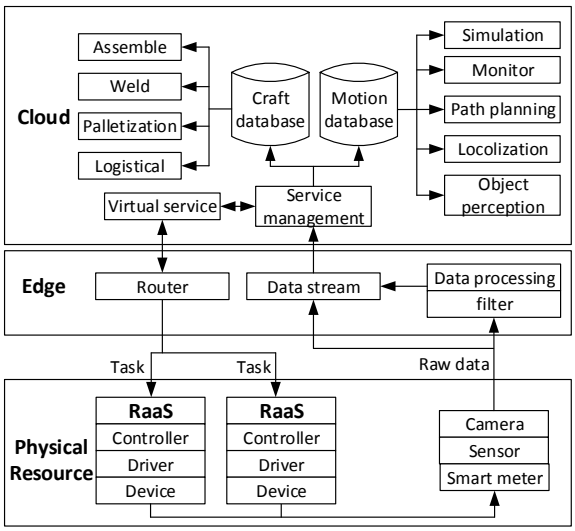


Figure 1: System architecture overview, with three main layers (cloud, edge, and physical resource). RaaS stands for robot as a service.

2.1. Cloud layer

The cloud layer works as the robot system's service coordinator and supervisor. Users are able to access the cloud over the network via an API. The cloud layer's core component—the service management—coherently manages service and executes the service packages accordingly. It keeps track of the state of available resources (information provided by the monitoring service) to identify the best candidates for hosting an application module and allocating the device's resources. The management based on the quality of services requirements is posed by the robot application, so that the robot application receives the quality it needs. It can also adjust the service-management policy's complexity to migrate components and allocate device resources dynamically to components based on diverse criteria (such as perceived latency)—or make the policy as simple as provisioning components statically on an edge device. The current version offers the latter policy (static application placement).

With today's wide-ranging Internet, various robot resources and abilities are sensed and connected easily. The robot resources and abilities are encapsulated into different robot cloud services (RCSs) that can be accessed, invoked, and deployed. The RCSs are classified and aggregated according to specific rules and algorithms. Users can search and invoke the qualified RCSs from a related robot cloud. The cloud primarily provides two categories of service: the motion cloud service (MCS), and the process cloud service (PCS). The MCS is the result of the robot motion ability's service encapsulation, which includes the control algorithm as path planning; forward and inverse kinematics; collision detection, and simulation. The PCS provides various processes and skills that the tasks need, such as welding, assembling, stamping, and painting.

2.2. Edge layer

The edge layer provides a platform for filtering and analyzing data generated by sensors, making the data processing architecture distributed, and thereby scalable. This layer's devices are any elements capable of hosting application modules, such as servers, local PC, routers, and gateways. Between the cloud and physical resource layer, the edge layer is optional for three main reasons.

First, to accomplish complex tasks, a huge amount of field data (such as power, current, and force) is generated dynamically by sensors. It is inefficient to stream all the raw data directly to the cloud, because most content is unessential and generates heavy network traffics. Thus, the edge can filter and preprocess data. Then the filtered result is uploaded to the cloud. This helps to balance the bandwidth load and manage cloud data.

Second, the challenge is to achieve the real-time capability of the robot control's functions. Because some functions are dependent on actual measured values during processing, they cannot tolerate a large latency. In cloud robotics, running the control system on the cloud may be unavailable because of communication failures. Thus, an edge device is needed to keep the real-time capability upright through short communication paths by placing the control system near to the robots.

Third, in some cases, most of the sensors, cameras and so forth are not cloud-ready. So an edge device can act as an interface and interact with the sensors (over IoT, in most cases) and communicate with the cloud.

2.3. Physical resource layer

At the physical resource layer, robot controllers interpret the task files received from the cloud and generate movement instructions. These instructions, including the interpolation method and target point, are sent to drive the servos. The robot controller can function in real time, and the embedded system implements any critical real-time functions for robot control.

The production process is measured and monitored by sensors, such as cameras and smart meters. The measur-

ing results processed by edge devices are fed back to the cloud for simulation, monitoring, performance evaluation, motion planning, and path compensation. Thus, it forms a closed-loop robot system.

3. Use case: a robot cell for membrane wall welding

To validate and evaluate the proposed robot system, the system was implemented on a robot cell for membrane wall welding. Membrane walls are one of the most important components in the boiler industry. They are composed of many long, thin steel tubes, welded together longitudinally using numerous pins (see Fig. 2a). Membrane walls are usually very big in size (over 10m long and 7~9m wide). The existing welding method for a membrane wall is manual spot-welding. The welding points are marked on the surface of tubes with chalk. Workers are assigned to different areas of the same wall to weld artificially (Fig. 2b). Without an online deformation measurement, manual welding has low efficiency and accuracy. However, industry robots can be used to weld the membrane wall.



Figure 2: (a) A diagram of the membrane wall; (b) the existing method for membrane wall welding.

We designed an industry robot system to weld the membrane wall. This case consists of several subtasks. In terms of the computer-aided design (CAD) model, the motion program is generated by an offline programming system. Using sensor feedback, the system can compute the welding points' poses on the tube and compensate the deformation. Then the motion path is downloaded and sent to the robot controller. A gantry or external axis carries the robot to weld pins in the big workpiece. Limited to the single robot power and storage, we used the edge computing-based system to this application.

3.1. System architecture

Fig. 3 presents the architecture of the robotic welding use case. Most of the subtasks are computation-intensive, requiring high computational performance from the system. Cloud computing offers robust power, but uploading all the raw data to the cloud takes significant time and bandwidth, which severely diminishes the real-time performance. To contend with this, in the system we developed an edge node and deployed devices' API and pose measurement on it. The result of the subtask pose measurement is urgently needed for compensating the preplanned paths on the cloud, thus it is assigned to the edge node. This offers an elegant solution, because on one hand, now the edge node directly communicates with the robot cell through API. On the other hand, it filters raw data and sends vital information to the cloud. Other tasks are predefined and assigned to the cloud.

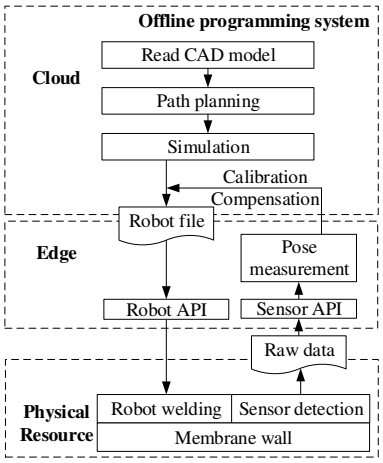


Figure 3: The system architecture. API stands for application programming interface; CAD stands for computer-aided design. The subtasks on the cloud and the edge are predefined.

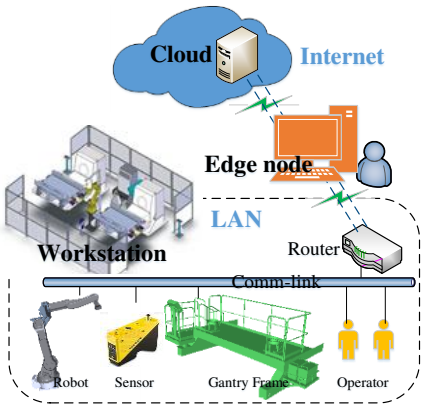


Figure 4: The system network configuration.

3.2. System network configuration

Fig. 4 shows the system network configuration. The cloud is deployed on the Internet. The edge node monitors and manages the physical resources at the workstation level. Inside the workstation, a local area network (LAN) is established through a router, which ensures a high speed of data transmission. Each device in the field is assigned a logical address. Therefore, an edge node serves one robotic task. For the sake of security, the LAN can block the external network. In the future, we plan to establish a private cloud and use access authentication to further augment security.

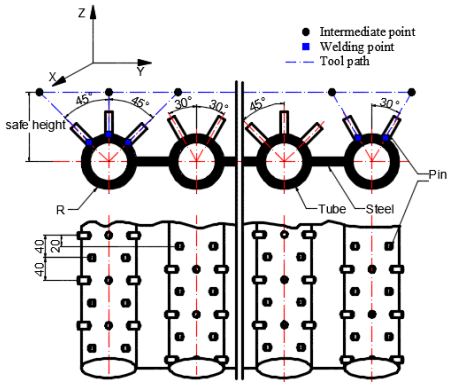


Figure 5: The diagram of welding technical requirements.

3.3. Path planning

Robotic paths are planned based on the CAD. Pins are welded vertically on the surface of the tube in a radial direction, as Fig. 5 shows. The welding points (x,y,z) can be calculated as

$$\begin{cases} x = x_0 + count \times stepdistance \\ y = y_0 + R \times \cos\theta \\ z = z_0 + R \times \sin\theta \end{cases} \quad (1)$$

where R: the tube radius; θ: the angle between the pin axial and Z axis; (x_0,y_0,z_0): the center coordinate of a specific section; *count*: the index of current section; and *stepdistance*: the distance between every two adjacent sections. To avoid collision between the tool and welded pins, the intermediate points are needed. The intermediate point's coordinate (x,y,z) is

$$\begin{cases} x = x_0 + count \times stepdistance \\ y = y_0 + safeheight \times \tan\theta \\ z = z_0 + safeheight \end{cases} \quad (2)$$

3.4. Pose measurement

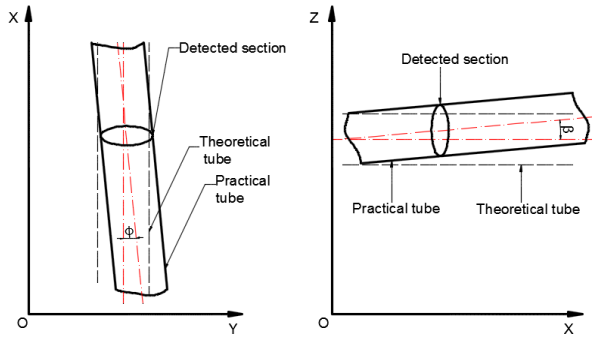


Figure 6: The pose measurement.

Deformations and warps may occur while the tube is being welded (see Fig. 6). Besides, the deviations between the CAD model and the membrane wall are inevitable. The preplanned paths should be calibrated and compensated by the measurement data. We use a light section sensor (a non-compact laser sensor) to measure the profile of cylindrical objects in the cross-section.

The raw data is a set of discrete coordinates. The section center coordinate, radius, and orientation of the current tube are obtained by recognizing and fitting raw data into an ellipse. The elliptic parameters include a center coordinate (x_c,y_c,z_c), major semi-axis a, minor semi-axis b, and the major axis deflection angle β. Related methods are elaborately introduced in [22].

In the XY plane, the angle φ equals the anti-cosine of b and a. That is, $\varphi = \text{rcos}(b/a)$. In the XZ plane, we can determine the angle β using the fitting result. We can calculate the matrix of the final processing point ${}^{T}_{U}T_{p}$ as

$${}^{T}_{U}T_{p} = R(Y,\beta) \times R(Z,\phi) \times \begin{bmatrix} 1 & 0 & 0 & x_c \\ 0 & \cos\theta & -\sin\theta & y_c \\ 0 & \sin\theta & \cos\theta & z_c \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

where

$$R(Z,\phi) = \begin{bmatrix} \cos\phi & -\sin\phi & 0 & 0 \\ \sin\phi & \cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4)$$

$$R(Y,\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5)$$

It is worth mentioning that the above formulas look simple, but over a big size membrane wall, for every 40mm interval in the longitudinal direction of one tube, 5~6 track points are needed to be calculated, which brings a big load for the system, especially for the iterative process in pose measurement.

3.5. System component failure

To enhance the system reliability, diverse component failure (device faults, network failure and the data loss) are taken into consideration. We establish a state table and design a polling program to periodically capture the devices' states. At present, we will stop the running system once the cloud or the edge breaks down. At the physical resource level, the collision detection algorithm is equipped to ensure the field security. To prevent the data loss, such as the sensor detected data, the back-up of the primitive data will be uploaded to the cloud when it is free. More work of this part is now on developing and will be presented in the future.

4. Experiment

We used a Motoman robot MH50II, a light section sensor, a torch model, and a membrane wall to verify the system (see Fig. 7). The light section sensor is a LPS36HI unit of Leuze Electronic. To compare performance, we tested two approaches: the cloud-based approach and the cloud-edge hybrid approach. The cloud-

based approach uploads all raw data to the cloud and executes all tasks on the cloud. The cloud-edge hybrid approach executes pose measurements on the edge node and executes the remaining tasks on the cloud. Table 1 lists the two approaches' tasks. We deployed the system on the Alibaba Cloud. The edge is a PC with two cores and 4-gigabyte memories, with a router connecting to physical devices. We conducted four experiments. The number of times that sensors detected is 1, 5, 20, and 50, respectively. Each experiment was executed 10 times, and we calculated the average as the result (see Fig. 8).

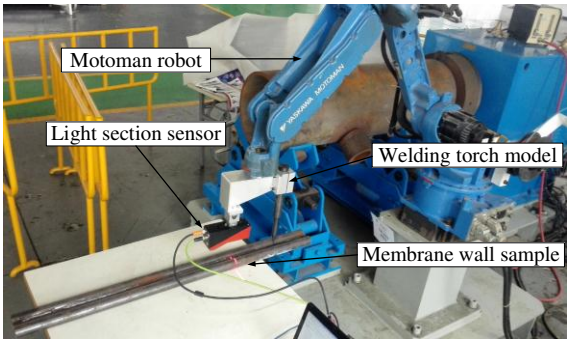


Figure 7: The experiment layout.

Table 1: The two approaches' tasks.

	Cloud-based	Cloud-edge hybrid
Path planning and simulation	√	√
Upload raw data	√	-
Pose measurement	√	√
Upload filtered data	-	√
Transfer robot file back	√	√

The experimental results show that the time for pose measurement in the cloud-based approach is much shorter than the cloud-edge hybrid approach in all four groups. That is because in cloud-based approach, this task is executed on the cloud, which is more powerful than the edge node. There are almost no differences in execution time for path planning, simulation and transferring robot files between two approaches, because they are all conducted in the cloud.

It is obvious that the total execution time in the cloud-based approach is far greater than the cloud-edge hybrid approach. In the first approach, uploading raw data to the cloud takes most of the total execution time. Although the cloud is computationally more robust, it offers no advantages in terms of real-time performance when the number of times detected increases. On the other hand, compared to the cloud-based approach, the cloud-edge hybrid approach saves network bandwidth by nearly three orders of magnitude (see Table 2).

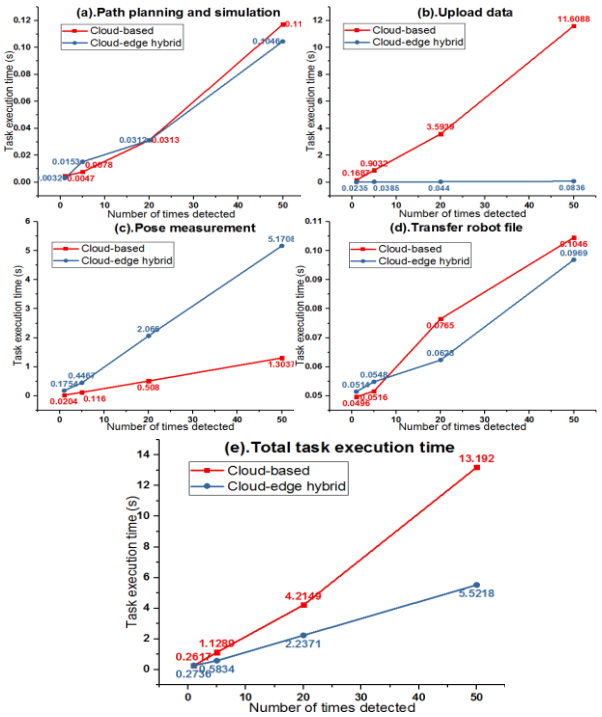


Figure 8: Experimental results: red lines indicate the results of the cloud-based approach; blue lines indicate the results of the cloud-edge hybrid approach.

Table 2: The two approaches' occupied bandwidth.

Number of times detected	Bandwidth of cloud-based approach (Kb/s)	Bandwidth of cloud-edge hybrid approach (Kb/s)	Saved bandwidth (Kb/s)
1	885.25	0.92	884.33
5	1,011.49	2.17	1,009.33
20	1,108.83	2.23	1,106.61
50	885.63	2.25	883.38

5. Conclusions

We proposed a three-layer (cloud, edge, and physical resource) industrial robot system. To respond to the lack of cloud computing solutions for aspects of time delay and network transmission, we deployed functions on an edge node to integrate various devices' interfaces and preprocess raw data from the field. Then, we applied the proposed system to a scenario involving robotic welding on a membrane wall. Finally, we conducted an experiment. The results show that our system offers better real-time and network transmission performance than a cloud-based approach. In the future, we will deploy our system on a private cloud and we intend to develop the dynamic deployment and on-demand allocation between the cloud and the edge. Furthermore, we will develop the data compression module for the sensor data to further augment system real-time performance.

References

[1] Jiafu wan, Shenglong Tang, Hehua Yan, Di Li, Shiyong Wang, Athanasios V. Vasilakos. Cloud robotics: current status and open issues. IEEE ACCESS, vol 4, 2016:2797-2807

[2] B. Kehoe, S. Patil, P. Abbeel, K. Goldberg, A survey of research on cloud robotics and automation, IEEE Transaction Automation Science Enginering. Vol 12 (2),2015:398-409

[3]. J. Kuffner, "Cloud-enabled robots," in Proc. IEEE-RAS Int. Conf. Humanoid Robot., Nashville, TN, USA, 2010

[4] M. Inaba. Remote-brained robotics: Interfacing AI with real world behaviors. Robotics Research, v (6), 1994:335-344

[5] A. Sanfeliu, N. Hagita, and A. Saffiottie. Networked robot systems. Robotics and autonomous systems. v(56),2008:793-797

[6] M. Markus et al, RobotEarth. IEEE Robotics and Automation Magazine, June, 2011:69-82

[7] Riazuelo L, Tenorth M, Marco D D, et al. RoboEarth Semantic Mapping: A Cloud Enabled Knowledge-Based Approach[J]. IEEE Transactions on Automation Science & Engineering, 2015, 12(2):432-443.

[8] R. Arumugam et al., DAvinCi: A cloud computing framework for service robots, in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), May 2010, pp. 3084-3089

[9] G. Mohanarajah, D. Hunziker, R. D'Andrea, and M. Waibel, Rapyuta: A cloud robotics platform, IEEE Trans. Autom. Sci. Eng., vol. 12, no. 2, pp. 481_493, Apr. 2015.

[10] Furrer J, Kamei K, Sharma C, et al. UNR-PF: An open-source platform for cloud networked robotic services[C]// Ieee/sice International Symposium on System Integration. IEEE, 2012:945-950.

[11] Zhihui Du, Ligang He, Yinong Chen, Yu Xiao, Peng Gao, Tongzhou Wang. Robot Cloud: Bridging the power of robotics and cloud computing. Future Generation Compter systems. 74 (2017): 337-348.

[12] Shi W, Cao J, Zhang Q, et al. Edge Computing: Vision and Challenges[J]. IEEE Internet of Things Journal, 2016, 3(5):637-646.

[13] Wang X V, Wang L, Mohammed A, et al. Ubiquitous manufacturing system based on Cloud: A robotics application[J]. Robotics and Computer-Integrated Manufacturing, 2017, 45(C):116-125.

[14] Wang X V, Wang L, Givehchi M. ICMS: A Cloud-Based System for Production Management[M]// Advances in Production Management Systems: Innovative Production Management Towards Sustainable Growth. Springer International Publishing, 2015:444-451.

[15] Shi W, Dustdar S. The Promise of Edge Computing[J]. Computer, 2016, 49(5):78-81.

[16] Satyanarayanan M, Bahl P, Davies N. The Case for VM-Based Cloudlets in Mobile Computing[J]. IEEE Pervasive Computing, 2009, 8(4):14-23.

[17] Satyanarayanan M. The Emergence of Edge Computing[J]. Computer, 2017, 50(1):30-39.

[18] Luan T H, Gao L, Li Z, et al. Fog Computing: Focusing on Mobile Users at the Edge[J]. Computer Science, 2015.

[19] Escamilla-Ambrosio P J, Rodríguez-Mota A, Aguirre-Anaya E, et al. Distributing Computing in the Internet of Things: Cloud, Fog and Edge Computing Overview[J]. 2018.

[20] Lopez P G, Montresor A, Epema D, et al. Edge-centric Computing: Vision and Challenges[J]. Acm Sigcomm Computer Communication Review, 2015, 45(5):37-42.

[21] Escamilla-Ambrosio P J, Rodríguez-Mota A, Aguirre-Anaya E, et al. Distributing Computing in the Internet of Things: Cloud, Fog and Edge Computing Over-view[J]. 2018.

[22] Youdong Chen, Chongxu Liu. Radius and Orientation Measurement for Cylindrical Object by Light Section Sensor. Sensors. 2016, 16, 1981:1-15