# Cloud Robotics Architectures: Challenges and Applications

Aindrila Mukherjee
*Department of Electronics and Communication Engineering*
*Techno India University, West Bengal*
*EM-4, Salt Lake City, Sector V, Kolkata, West Bengal 700091*

***Abstract* - Cloud robotics is an evolving field that allows robots to offload computation and storage intensive jobs into the cloud. Robots are limited in terms of computational memory and storage. Cloud provides unlimited computation memory, storage and especially collaboration opportunity. Cloud-enabled robots are divided into two categories , standalone and networked robots. This paper surveys cloud robotic platforms, standalone and networked robotic works such as simultaneous localization and mapping (SLAM).[5]**

***Index Terms - cloud-enabled robots, cloud robotics, cloud technologies, standalone and networked robots, Software as a Service, Platform as a Service, Infrastructure as a Service..***

## I. INTRODUCTION

Cloud robotics is a field of robotics that invokes cloud technologies such as cloud computing, cloud storage, and other Internet technologies on the benefits of converged infrastructure and shared services for robotics. When connected to cloud, robots can benefit from the powerful computation, and communication resources of modern data centre in the cloud, which can process and share in sequence from various robots or agent (other machines, smart objects, etc.). Humans can also hand over tasks to robots remotely through networks. Cloud computing technologies enable robot systems to be endowed with powerful capability whilst reducing costs through cloud technologies. Thus, it is possible to build lightweight, low cost, smarter robots have intelligent "brain" in the cloud. The "brain" consists of data centre, knowledge base, task planners, deep learning, information processing, environment models, communication support, etc.[1][2][3][4]

## CLOUD COMPUTING TECHNOLOGIES

Cloud computing consist of three models as Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) as shown in fig 1.SaaS applications are served over the internet, thus eliminating the need to install and run the application on the users system. Theyare managed from a central location and accessed remotely by a web browser or a mobile client. Google Apps is the most widely used SaaS application suit. PaaS refers to computing platform served by cloud infrastructure. PaaS offers developers to get hold of all the systems and environments required for the life cycle of software, be it testing, deploying and hosting of web applications. IaaS provides,the required infrastructure as a service. The client need not buy the required servers, data centre or the network resources. The spirit of IaaS model is a pay-as-you-go monetary model. Amazon and Microsoft are also IaaS providers.

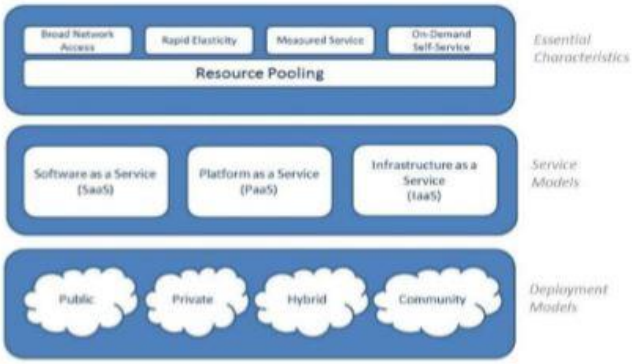



FIG.1: CLOUD COMPUTING INFRASTRUCTURE

Robots make significant socio-economic impacts to human lives [10]. For example, robots can do repeatitive or dangerous tasks, such as painting, packaging, and welding. However, robots are limited in terms of computational capacity and storage. Also they have physical constraints such as size, shape, motion mode and working environment [11]. Robots are usually used for industrial purposes; they are not usually used in daily life due to their cost. Cloud computing can be used to enhance robot capabilities. Cloud computing technologies provide many advantages that can be valuable for working and running robot services. For example complex computations of computation intensive applications, can be offloaded in the cloud like Apple's voice recognition service "Siri". Connecting the robots to semantic knowledge databases hosted in cloud will allow a large number of varied robots to share common sense knowledge [12]. The concept of "robot-as-a-service"(RaaS) refers to robots that can be vigorously combined to give support to the execution of specific applications. RaaS has three aspects: structure, interface, and behaviour. There can be many kinds of robot intelligent devices. For example, robot cops [13], restaurant robot waiters [14], robot pets [15], and patient care robots [16]. These robots are spread in different locations and can be accessed through CR platforms.

## ROS (ROBOTICS OPERATING SYSTEM)

Writing software for robots is not easy, particularly as the scope of robotics continues to grow. Different types of robots can have varying hardware, making code reuse non trivial. On top of this, the size of the required code can be intimidating, as it must contain a stack starting from driver-level software and continuing up through abstract reasoning, and beyond. Since the breadth of expertise is well beyond the capabilities of single researcher.Robotics software architectures must also support large-scale software combination efforts. To meet these challenges many robotics researchers, have previously created a variety of frameworks to manage complexity and ease rapid prototyping of software for experiments, resulting in the many robotic software systems currently used in academia and industry [17]. Each of these frameworks was intended for a particular purpose, perhaps in response to perceived weakness of other available frameworks, or to place stress on aspects which were seen as most important in the design process. ROS, the framework is also the product of trade-offs and prioritizations made during its design cycle. We believe its importance on large-scale robotics research will be useful in a wide variety of situation, as robotic systems grow ever more complex. In this paper we discuss,the design goals of ROS, how our implementations work towards them, and demonstrate how ROS handles several common use cases of robotics software development. We do not claim that ROS is the best framework for all robotics software. In fact, we do not believe that such a framework exists.The field of robotics is too broad for a single solution. ROS was designed to meet a precise set of challenges encountered when developing large-scale

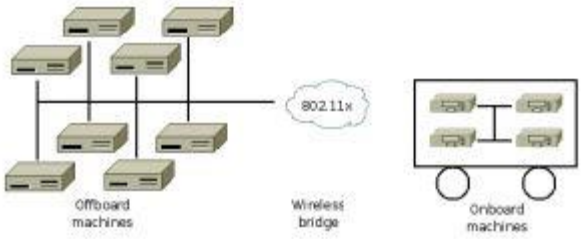



FIG. 2. A TYPICAL ROS NETWORK CONFIGURATION

service robots as part of the STAIR project [18] at Stanford University1 and the Personal Robots Program [19] at Willow Garage2 but the resulting architecture is more general than the service robot and mobile manipulation domains. The philosophical goals of ROS can be summarized as:

• Peer-to-peer
• Tools-based
• Multi-lingual
• Thin
• Free and Open-Source To our knowledge,

A. Peer-to-Peer A system built using ROS consists a number of processes, potentially on different hosts, connected at runtime in a peer-to-peer topology. Although frameworks based on a central server can also realize the profit of the multi-process and multi-host design.Central data server is problematic if computers are connected in a heterogeneous network.

B. Multi-lingual When writing code many individuals have preferences for some programming languages above others. These preferences are the result of personal trade-offs between programming time, ease of debugging, syntax, runtime efficiency, and a host of other reasons, both technical and cultural. For these reasons, we have designed ROS to be language-neutral. ROS currently supports four very different languages: C++, Python, Octave, and LISP, with other language ports in various states of completion. The ROS specification is at the messaging layer, not any deeper. Peer-to-peer connection negotiation and configuration occurs in XML-RPC, for which reasonable implementations exist in most major languages. Rather than provide a C-based implementation with stub interfaces generated for all major languages, we prefer instead to implement ROS natively in each target language, to better follow the conventions of each language. However, in some cases it is expedient to add support for a new language by wrapping an existing library: the Octave client is implemented by wrapping the ROS C++ library. To support cross-language development, ROS uses a simple, language-neutral interface definition language (IDL) to describe the messages sent between modules. The IDL uses (very) short text files to describe fields of each message, and allows composition of messages, as illustrated by the complete IDL file for a point cloud message:

Header header
Point32[] pts
ChannelFloat32[] chan

C. Tools-based

In an effort to manage the complexity of ROS, we have opted for a microkernel design, where a large number of small tools are used to build and run the various ROS components, rather than constructing a monolithic development and runtime environment. These tools perform various tasks, e.g., navigate the source code tree, get and set configuration parameters, visualize the peer-to-peer connection topology, measure bandwidth utilization, graphically plot message data, auto-generate documentation, and so on. Although we could have implemented core services such as a global clock and a logger inside the master module, we have attempted to push everything into separate modules. We believe the loss in efficiency is more than offset by the gains in stability and complexity management.

D. Thin

As eloquently described in , most existing robotics software projects contain drivers or algorithms which could be reusable outside of the project. Unfortunately, due to a variety of reasons, much of this code has become so entangled with the middleware that it is difficult to “extract” its functionality and re-use it outside of its original context. To combat this tendency, we encourage all driver and algorithm development to occur in standalone libraries that have no dependencies on ROS. The ROS build system performs modular builds inside

the source code tree, and its use of CMake makes it comparatively easy to follow this "thin" ideology. Placing virtually all complexity in libraries, and only creating small executable which expose library functionality to ROS, allows for easier code extraction and reuse beyond its original intent. As an added benefit, unit testing is often far easier when code is factored into libraries, as standalone test programs can be written to exercise various features of the library. ROS re-uses code from numerous other open-source projects, such as the drivers, navigation system, and simulators from the Player project , vision algorithms from OpenCV , and planning algorithms from Open RAVE, among many others. In each case, ROS is used only to expose various configuration options and to route data into and out of the respective software, with as little wrapping or patching as possible

- Nodes: Nodes are processes that perform computation. ROS is designed to be modular at a fine-grained scale; a robot control system usually comprises many nodes. For example, one node controls a laser range-finder, one node controls the wheel motors, one node performs localization, one node performs path planning, one Node provides a graphical view of the system, and so on. A ROS node is written with the use of a ROS client library, such as roscpp or rospy.
- Master: The ROS Master provides name registration and lookup to the rest of the Computation Graph. Without the Master, nodes would not be able to find each other, exchange messages, or invoke services.
- Parameter Server: The Parameter Server allows data to be stored by key in a central location. It is currently part of the Master.
- Messages: Nodes communicate with each other by passing messages. A message is simply a data structure, comprising typed fields. Standard primitive types (integer, floating point, boolean, etc.) are supported, as are arrays of primitive types. Messages can include arbitrarily nested structures and arrays (much like C structs).
- Topics: Messages are routed via a transport system with publish / subscribe semantics. A node sends out a message by publishing it to a given topic. The topic is a name that is used to identify the content of the message. A node that is interested in a certain kind of data will subscribe to the appropriate topic. There may be multiple concurrent publishers and subscribers for a single topic, and a single node may publish and/or subscribe to multiple topics. In general, publishers and subscribers are not aware of each others' existence. The idea is to decouple the production of information from its consumption. Logically, one can think of a topic as a strongly typed message bus. Each bus has a name, and anyone can connect to the bus to send or receive messages as long as they are the right type.
- Services: The publish / subscribe model is a very flexible communication paradigm, but its many-to-many, one-way transport is not appropriate for request / reply interactions, which are often required in a distributed system. Request / reply is done via services, which are defined by a pair of message structures: one for the request and one for the reply. A providing node offers a service under a name and a client uses the service by sending the request message and awaiting the reply. ROS client libraries generally present this interaction to the programmer as if it were a remote procedure call.
- Bags: Bags are a format for saving and playing back ROS message data. Bags are an important mechanism for storing data, such as sensor data, that can be difficult to collect but is necessary for developing and testing algorithms.

SYSTEM ARCHITECTURE OF CLOUD ROBOTICS

Networked robotics can be seen as transition state between pre-programmed robots to cloud-enabled robots [6]. As previouslymentioned, cloudroboticsaimattransferringthehigh complexity of the computing process to the cloud platform through communication technology. This greatly reduces the computational load on individual robots. Figure 4 describes the main architecture of the robotic cloud.


FIG 3.SYSTEM ARCHITECTURE OF CLOUD ROBOTICS.

The architecture of cloud robotics is mainly composed of two parts: the cloud platform and its related equipment and the bottom facility. The bottom facilities usually include all types of mobile robots, unmanned aerial vehicles, machineryandotherequipment. Accordingly, thecloudplatform is composed of a large number of high-performance servers, proxy servers, massive spatial databases and other components. Multi-robot cooperative works, such as SLAM and navigation [7] networks, are typical applications of cloud robots. The term ''networked robotics'' refers to the communication mode of cloud robotics and multi-robot systemsarecomposedascooperativecomputingnetworksusing wireless communication technologies. The major advantages of cooperative computing networks are the following: 1) a collaborative computing network can gather computing and storage resources, and can dynamically allocate

theseresourcesaccordingtospecificworkrequirements; and 2) because of the exchange of information, decisions can be made cooperatively between machines. The nodes' computing and storage capabilities' deficiency in networked robotic systems may lead to large delays, while in cloud robotic systems the nodes can collaborate with spare nodes by transferringcomputingorstoragetasks.

.

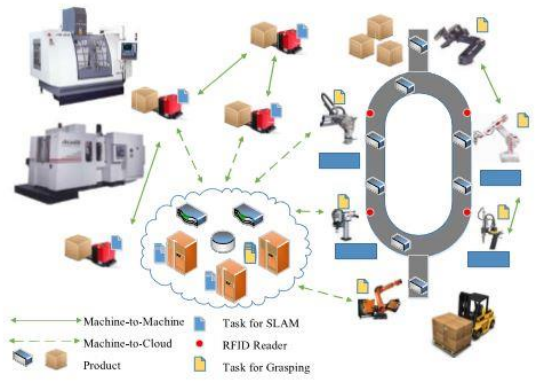
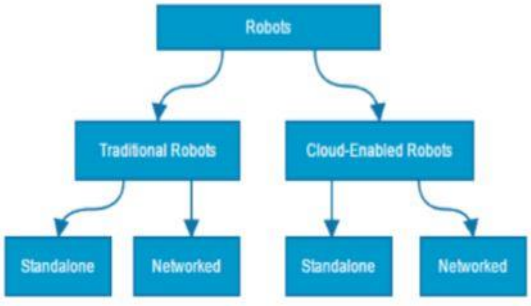



FIGURE 4:IMPLEMENTATION OF CLOUD ROBOTICS IN INDUSTRIAL ENVIRONMENT

connected to the cloud resource can connect to the cloud through other nodes that have already established links to thecloud. Theapplicationofthismechanismgreatlyexpands the manageable complexities of tasks accomplished through multi-robot cooperative work and enhances the efficiency of specific work tasks. Othertasksthatdo notinvolveaneedforadditionalrobots but require complex operations, such as grasping, are also hot research areas. Due to the fact that the target object is usually unknown, several researchers have so far proposed the incorporation of a large number of sensing devices on the grabbing body [8], aiming to improve the accuracy of grasping. However, in actual industrial environments, a demand for high accuracy and fewer sensors is more in line with production practices [9]. At the same time, due to the limitation of physical space and materials, the equipment and storage facilities are limited. This leads to a bottleneck of development and research. With the introduction of big data, with a small amount of sensor and other characteristic data uploaded, researchers can determine a match in the cloud. Then the characteristic data of the grab motion are downloaded and sent to the mechanical equipment for execution of the operation. An example of the applications of cloud robotics including SLAM and Grasping is shown in Figure 2. In summary, the main features of the cloud robotics architecture are as follows.
1) In the cloud infrastructure, where computing tasks are dynamic and resources are elastic and availableon-demand.
2) Thecloudrobotics''brain''isin the cloud. Theresultsofprocessingcanbeobtainedthroughnetworkingtechnologies, whiletasksareprocessedindividually.
3) Computingworkcanbedelegatedtothecloud, whichleads to a smaller robot load and greatly extended battery life.

CLOUD ROBOTICS PLATFORM

Developing software solutions for robots is difficult, because of varying hardware and non-standardized APIs. Robotics researchers, have created a variety of frameworks to manage complexity and facilitate rapid prototyping of software for experiments, resulting in the many robotic software systems currently used in academia and industry [14].Stanford University and Willow Garage developed a generalized open source operating system called Robot Operating System (ROS) for robots. ROS is not only an operating system; rather, it provides a structured communications layer above the host operating systems of a heterogeneous compute cluster.Rapyuta is an open source cloud robotics platform. It serves a platform-as-a-service (PaaS) framework for robots. Rapyutaarchitecture depends on LxCcontainers. It provides an environment to access RoboEarth. Knowledge Repository. Massively parallel computation, allowing humans to monitor or intervene robots and serving as a global repository to store and share object models, environment maps, and actions recipes between various robotic platforms are some of specifications of Rapyuta. It is a competitor of Rosbridgein terms of communication. Survivable Cloud MultiRobotics (SCMR) Framework is designed, implemented and evaluated for heterogeneous environments. One of the challenges for cloud robotics is the inherent problem of cloud disconnection.




The SCMR framework provides the combination of a virtual Ad-hoc network formed by robot-to-robot communication and a

FIG 5: CLASSIFICATION OF ROBOTS

physical cloud infrastructure formed by robot-to cloud communications. The design trade-off for SCMR is between the computation energy for the robot execution and the offloading energy for the cloud execution. The SCMR framework uses Web Sockets protocol for communication between the individual robots and the cloud server. In case of

cloud disconnection a virtual ad hoc cloud is created between the individual robots and the robot leader and the individual robots communicate with one another through the gossip protocol.Distributed Agents with Collective Intelligence (DAvinCi) is a software framework that provides the scalability and parallelism advantages of cloud computing for service robots in large environments. It is implemented as a system around the Hadoop cluster with ROS as the messaging framework.

## CLOUD-ENABLED ROBOTS

Robots have some constraints in terms of computational capacity, memory and storage. CR help them to overcome these challenges. Opportunity to use cloud allows cost effective robots to be produced. Robots can be classified as traditional robots and cloud-enabled robots. This paper focuses on cloud-enabled robots. A cloud technology not only empowers robots but also it allows them to network each other regardless of distance. Cloud-enabled robots are divided into two categories as standalone robots and networked robots. Classification of robots is shown in Figure 2.

Standalone robots can benefit from cloud in terms of computation power, storage capacity and memory. However, networked robots can make networks, share their information through cloud and can perform collaborative works. CR infrastructure with standalone robots and networked robots is presented in Fig. 6.

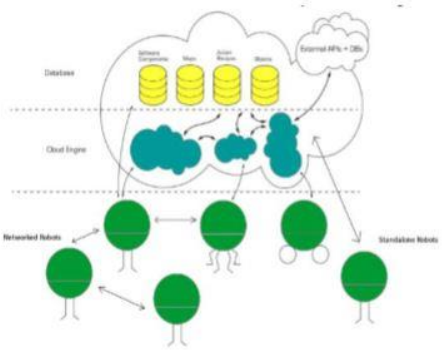


FIG.6: STANDALONE AND NETWORKED ROBOTS

in Fig. 6. Robots can do a wide variety of works such as grasping, identifying objects, SLAM, monitoring, networking and some other actuating works. Robots can grasp formerly known objects easily. They can also grasp novel objects with the help of cloud. In, a study aboutgrasp planning in the presence of shape uncertainty and how cloud computing can facilitate parallel Monte Carlo sampling is presented. Kehoe et al focus on parallel-jaw push grasping for the class of parts that can be modelled as extruded 2-D polygons with statistical tolerancing. SLAM refers to a technique for a robot or an autonomous vehicle to build a map of the environment without a priori knowledge, and to simultaneously localize itself in the unknown environment. SLAM is important in robotics and there are plenty of researches. It consists of statistical techniques such as Kalman filters, mapping and sensing. Riazuelo et al develop a cloud framework which name is Cloud framework for Cooperative Tracking and Mapping (C2TAM). This is a visual SLAM system based a distributed framework where the CPU-intensive map optimization and storage is allocated as a service in the Cloud, while a light camera runs on robots for tracking. The robots need only internet connection for tracking and cooperative repositioning. C2TAM providesa database consisting maps can be built and stored, stored maps can be reused by other robots. A robot can fuse its map online with a map already in the database, and several robots can estimate individual maps and fuse them together if an overlap is detected. Virtual monitoring technology has been applied in more and more fields such as military, education, medical science, manufacturing engineering, and so forth. In order to realize resource sharing among all collaborating robots in a virtual monitoring system, cloud computing is proposed by combining professional computing equipment as a super virtual computing center. Zhang et al, proposed 3D virtual monitoring system based on CR. This system‟s architecture consist of communication language for agent communication, algorithm for cooperative working and conflict resolution. Prototype system is applied for the monitoring of fully mechanized coal-mining equipment. Networking robots overcome the limitations of stand-alone robots by having robots, environment sensors, and humans communicate and cooperate through a network. Mateo et al, presented a work to decrease message overhead occurred because of communication. The proposed an information sharing model for group communication based on Brownian agent approach. In presented work they grouped robots in clusters with a cluster head to overcome message overhead . Kamei et al, proposed prototype infrastructure of cloud networked robotics enables multi-location robotic services for life support. Their study focuses on requirements in typical daily supporting services through example scenarios that target senior citizens and the disabled.

## ADVANTAGES

- Offloads the heavy computing tasks to the cloud.
- Lower the barrier to entry for robotics
- Scalable CPU, memory and storage
- Shared knowledge database
- Hardware upgrades are invisible & hassle free
- Longer battery life.
- Lighter and easier to maintain hardware.
- Robot experiences/history/behavior outcomes/learned skills can all be published or data mined.
- Expanding the knowledge

beyond “physical body”.

DISADVANTAGES

Cloud robotics is still taking baby steps, so will have to wait for the platforms to develop.
Cloud based applications can get slow or simply become unavailable leaving the robot "brainless".
Tasks that involve real time execution require onboard processing.

ROBOTIC APPLICATIONS

Future robotic applications will benefit from cloud robotics, which provides the following advantages over traditional networked robots. • Ability to offload computation-intensive tasks to the cloud. The robots only have to keep necessary sensors, actuators, and basic processing power to enable real-time actions (e.g., real-time control). The battery life is extended, and the robotic platform becomes lighter and less expensive with easier to maintain hardware. The maintenance of software onboard with the robots also becomes simpler, with less need for regular updates. As the cloud hardware can be upgraded independently from the robotic network, the operational life and usefulness of the robotic network can be easily extended. • Access to vast amounts of data. The robots can acquire information and knowledge to execute tasks through databases in the cloud. They do not have to deal with the creation and maintenance of such data. • Access to shared knowledge and new skills. The cloud provides a medium for the robots to share information and learn new skills and knowledge from each other. The cloud can host a database or library of skills or behaviors that map to different task requirements and environmental complexities. The RoboEarth project [12] is trying to turn this into a reality. Due to these advantages, cloud robotics has a wide range of potential applications in data-intensive or computationintensive tasks in the areas of intelligent transportation, environment monitoring, health care, smart home, entertainment, education, and defense. In this section, we discuss the opportunities and challenges that cloud robotics brings to traditional robotic applications. Specifically, we focus on three robotic applications: SLAM, grasping, and navigation SLAM

SLAM

SLAM refers to a technique for a robot or an autonomous vehicle to build a map of the environment without a prioriknowledge, and to simultaneously localize itself in the unknown environment. SLAM, especially vision-based SLAM and cooperative SLAM, are both data intensive and computation intensive. The steps such as map fusion and filtering for state estimation can be processed in a parallel fashion. Thus, these tasks can be offloaded to the cloud. For example, a grid based FastSLAM is implemented in a cloud computing framework as reported in . As demonstrated in , the cloud can substantially improve the implementation speed of SLAM

GRASPING

Robotic grasping has been an active research topic over a few decades. If the full 3-D model of the object is precisely known, then various methods can be applied to synthesize the grasp. If the object is unknown or not precisely known, the problem is much more challenging, and involves the access and pre-processing of vast amounts of data and can be computationally intensive. Recently, information-based or data-driven grasping methods have been developed to enable robotic grasping for any hand and any object. These methods requires access to large databases. By offloading this task to the cloud, grasping can be facilitated without requiring vast amounts of computing power, data, and storage space on the robotic platform. In addition, model knowledge of new objects learned by different robots can be shared in the cloud for future usage by other robots.

NAVIGATION

Robotic navigation refers to a robot's activity to determine its own position with respect to a certain reference and then plan a path to reach a desired location. It can involve a combination of tasks such as localization, path planning, and mapping. Basically, there are two types of approaches: map-less approaches and map-based approaches . Map-less approaches rely on the observations of the perception sensors for navigation. Due to the limited onboard resources, these approaches usually suffer from reliability issues. Map-based robotic navigation is relatively reliable if a precise map is available. It can either use a known map or build a map during the navigation. However, the process of building the map requires large amounts of storage space and is computationally intensive. On the other hand, the process of searching a map requires access to large amounts of data, which is challenging if the navigation area is large. Cloud robotics provides a very promising solution for future cloud-enabled navigation that avoids these two challenges. The cloud can not only provide storage space to store the large amount of map data, but also provide processing power to facilitate the building and searching of the map quickly. Through the cloud, commercially available maps (e.g., Google maps) can also be leveraged to develop reliable, agile, and long-range autonomous navigation solutions.

CONCLUSION

This paper presents cloud computing, cloud robotics and cloud interaction of robots. It surveys cloud platforms and cloud-enabled robotics studies. Standalone robots can benefit cloud technologies and networked robots can perform collaborative works. Networked cloud-enabled robots can

share computation resources, information and data with each other and can access new knowledge and skills not learned by them. This is a new paradigm in robotics that webelieve leads to exciting future developments. Future works can focus on reliable connection, data offloading methods and ubiquitous networking among robots and cloud services

## ACKNOWLEDGMENT

I take this opportunity to express my profound gratitude and deep regards to my guide Dr.AsitBaran Bhattacharya, Professor, Department of Electronics and Communication, Techno India University for his exemplary guidance, monitoring and encouragement throughout the course of this paper.

## REFERENCES

[1] "Cloud Robotics and Automation A special issue of the IEEE Transactions on Automation Science and Engineering". IEEE. Retrieved 7 December 2014.
[2] "RoboEarth".
[3] Goldberg, Ken. "Cloud Robotics and Automation".
[4] Li, R. "Cloud Robotics-Enable cloud computing for robots". Retrieved 7 December 2014
[5] CLOUD ROBOTICS AND AUTOMATION Srujikanth Das1, Rekam Saikiran2, R.VenkataRamana IJEEE Vol.No.8 Issue 01 January-June 2016
[6] G. Hu, W. P. Tay, and Y. Wen, ''Cloud robotics: Architecture, challenges and applications,'' IEEE Netw., vol. 26, no. 3, pp. 21–28, May/Jun. 2012.
[7] I. M. Rekleitis, G. Dudek, and E. E. Milios, ''Multi-robot cooperative localization: A study of trade-offs between efficiency and accuracy,'' in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., vol. 3. Sep. 2002, pp. 2690–2695.
[8] J. Felip and A. Morales, ''Robust sensor-based grasp primitive for a threefinger robot hand,'' in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., Oct. 2009, pp. 1811–1816.
[9] J.Liu,Q.Wang,J.Wan,andJ.Xiong,''Towardsreal-timeindoorlocalization in wireless sensor networks,'' in Proc. 12th IEEE Int. Conf. Comput. Inf. Technol., Chengdu, China, Oct. 2012, pp. 877–884.
[10]B. Siciliano and O. Khatib, Eds.: Springer Handbook of Robotics,Springer, 2008.
[11]Guoqiang Hu, Wee Peng Tay, Yonggang Wen: Cloud robotics:architecture, challenges and applications, Network, IEEE , vol.26, no.3, pp.21,28, May-June 2012.
[12]A. Chibani, Y. Amirat, S. Mohammed, E. Matson, N Hagita, M.Barreto: Ubiquitous robotics: Recent challenges and future trends. Robotics and Autonomous Systems ,2013.
[13]Wired Blog, Robot Cops to Patrol Korean Streets [Online].
[14]Robot Waiters [Online]. Available:http://www.technovelgy.com/ct/Science-Fiction.
[15]Robot Pets [Online]. Available: http://en.wikipedia.org/wiki/AIBO
[16] Robot to be added at Hoag Hospital Irvine [Online]. Available: http://www.intouchhealth.com/
[17]J. Kramer and M. Scheutz, "Development environments for autonomous mobile robots: A survey," Autonomous Robots, vol. 22, no. 2, pp. 101–132, 2007.
[18]M. Quigley, E. Berger, and A. Y. Ng, "STAIR: Hardware and Software Architecture," in AAAI 2007 Robotics Workshop, Vancouver, B.C, August, 2007.
[19]K. Wyobek, E. Berger, H. V. der Loos, and K. Salisbury, "Towards a personal robotics development platform: Rationale and design of an intrinsically safe personal robot," in Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA), 2008.
[20]M. Montemerlo, N. Roy, and S. Thrun, "Perspectives on standardization in mobile robot programming: The Carnegie Mellon Navigation (CARMEN) Toolkit," in Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS), Las Vegas, Nevada, Oct. 2003, pp. 2436–2441.
[21]A. Makarenko, A. Brooks, and T. Kaupp, in Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS), Nov. 2007.
[22]R. T. Vaughan and B. P. Gerkey, "Reusable robot code and the Player/Stage Project," in Software Engineering for Experimental Robotics, ser. Springer Tracts on Advanced Robotics, D. Brugali, Ed. Springer, 2007, pp. 267–289.
[23]G. Bradski and A. Kaehler, Learning OpenCV, Sep. 2008.
[24]R. Diankov and J. Kuffner, "The robotic busboy: Steps towards developing a mobile robotic home assistant," in Intelligent Autonomous Systems, vol. 10, 2008.
[25] J. Jackson, "Microsoft robotics studio: A technical introduction," in IEEE Robotics and Automation Magazine, Dec. 2007, http://msdn. microsoft.com/en-us/robotics.
[26]O. Michel, "Webots: a powerful realistic mobile robots simulator," in Proc. of the Second Intl. Workshop on RoboCup. Springer-Verlag, 1998.M. King, B. Zhu, and S. Tang, "Optimal path planning," *Mobile Robots*, vol. 8, no. 2, pp. 520-531, March 2001.
[27]H. Simpson, *Dumb Robots*, 3rd ed., Springfield: UOS Press, 2004, pp.6-9.
[28]M. King and B. Zhu, "Gaming strategies," in Path Planning to the West, vol. II, S. Tang and M. King, Eds. Xian: Jiaoda Press, 1998, pp. 158-176.
[29]B. Simpson, et al, "Title of paper goes here if known," unpublished.
[30]J.-G. Lu, "Title of paper with only the first word capitalized," *J. Name Stand. Abbrev.*, in press.
[31]Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," *IEEE Translated J. Magn. Japan*, vol. 2, pp. 740-741, August 1987 [*Digest 9th Annual Conf. Magnetics Japan*, p. 301, 1982].
[32]M. Young, *The Technical Writer's Handbook*, Mill Valley, CA: University Science, 1989.