Chapter 24
Networked-, Cloud- and Fog-Robotics

Dezhen Song, Ajay Kumar Tanwani, and Ken Goldberg

Summary

Almost all robots have access to computer networks that offer extensive computing, memory, and other resources that can dramatically improve performance. Robots can use networks such as the Internet to communicate with remote resources such as human teleoperators or central servers for memory and computation. In contrast, swarm robots coordinate locally with neighbouring robots and resources. Networked robots trace their origin to telerobots (remotely controlled robots). Telerobots are widely used to explore undersea terrains and outer space, to defuse bombs, and to clean up hazardous waste. Early telerobots were accessible only to trained and trusted experts through dedicated communication channels, but today's networked robots are more ubiquitous and are operated by less experienced users for a variety of tasks. This chapter will describe relevant network and communication technology, the history of networked robots, architecture and properties of networked robots, how to build a networked robot, example systems, and recent developments in Cloud- and Fog-Robotics.

24.1 Introduction

Networked Robots combine robotics with communication networks. This sub-field focuses on robot system architectures, interfaces, hardware, software, and applications that use networks (primarily the Internet / Cloud). This chapter considers how robots can communicate and enhance its abilities in sensing/perception, planning/decision-making, and action by using resources provided over the network.

Networked robots originate from teleoperation and telerobots, where an operator remotely controls a robot to perform real world task such as robot manipulation, minimally invasive surgery, flying a drone, security/surveillance, remote patient monitoring, inspection/exploration in deep underwater or space missions and so on. The rise of Internet along with cloud and edge computing has shifted the paradigm to offload robot computation and storage and facilitate multi-agent robot systems. In this chapter, we first review the history and related work in Section 24.2. In Section 24.3,

Dezhen Song is with Department of Computer Science and Engineering, Texas A&M University, College Station, TX 77845, United States of America. Ajay Tanwani and Ken Goldberg are with Department of Industrial Engineering and Operations Research, and Department of Electrical Engineering and Computer Sciences, UC Berkeley, CA 94720, United States of America.

we review enabling technologies (e.g network and communication) to provide necessary background for the following two main Sections 24.4 and 24.5. Section 24.4 focus on traditional networked robots while Section 24.5 summarize emerging developments in Cloud and Fog robotics. Section 24.6, we conclude the chapter with recent applications and future directions.

24.2 History

Recent evolution of the Internet and wireless networks has enabled networked robots to quickly expand their scope from the traditional master-slave teleoperation relationship to an integration of robots, human, agents, off-board sensors, databases, and clouds across the globe. To review the history of networked robots, we trace back to remotely controlled devices.

24.2.1 Teleoperation Control Spectrum: from Direct Control to Full Autonomy

Like many technologies, remotely controlled devices were first imagined in science fiction. In 1898, Nicola Tesla [2] demonstrated a radio-controlled boat in New York's Madison Squares Garden. The first major experiments in teleoperation were motivated by the need to handle radioactive materials in the 1940s. Goertz demonstrated one of the first bilateral simulators in the 1950's at the Argonne National Laboratory [3]. Nowadays teleoperated robots are everywhere including personal telepresence robots [4, 5], underwater remote operated vehicles (ROVs) [6], planetary rovers [7], surgical robots [8], etc. Over a century in development, teleoperation and telerobotics encompass a spectrum of remotely controlled devices. The book from Sheridan [1] provides a detailed review on teleoperation and telerobotics research (see Fig. 24.1), while "Beyond Webcams" gives an overview of Internet telerobots across different engineering examples [9]. As an example, consider a teleoperated vehicle:

- Type (a) is direct control: we can envision a vehicle equipped with radio-based remote control (RC) and servos to actuate throttle, brake, and steering. It also has a television camera for feedback. There is no computer involved, as all signals are analog and directly transmitted over the radio. The operator sends the control radio signals and receives television signals as feedback display. The human operator is in direct control without any assistance.
- Type (b) is still direct control but computers are introduced at both operator and robot ends. At operator end, it is named as human machine interface (HMI) computer. The computer system digitizes signals into variables and transmits them over the communication channel. In early days, the communication channels were the dedicated point-to-point links but they have become computer networks as communication technology has evolved. Here we have explicit configuration $\mathbf{x}$ in variable format but the system simply passes it around instead of providing any assistance.
- Type (c) enters supervisory control domain. Here the telerobot computer starts to close the loop locally to provide assistance. At this stage, the control loop is simple feedback control for key state variables. No advanced perception and recognition are involved. For example, a vehicle may have cruise control to help maintain its speed. However, the human operator still has to closely maintain direction control.
- Type (d) advanced supervisory control by providing more loop closure capabilities through advanced intelligent algorithms. The human operator does not need to monitor the state of the robot continuously. For example, the vehicle may have collision avoidance, lane-keeping, adaptive cruise control, and global positioning system (GPS)

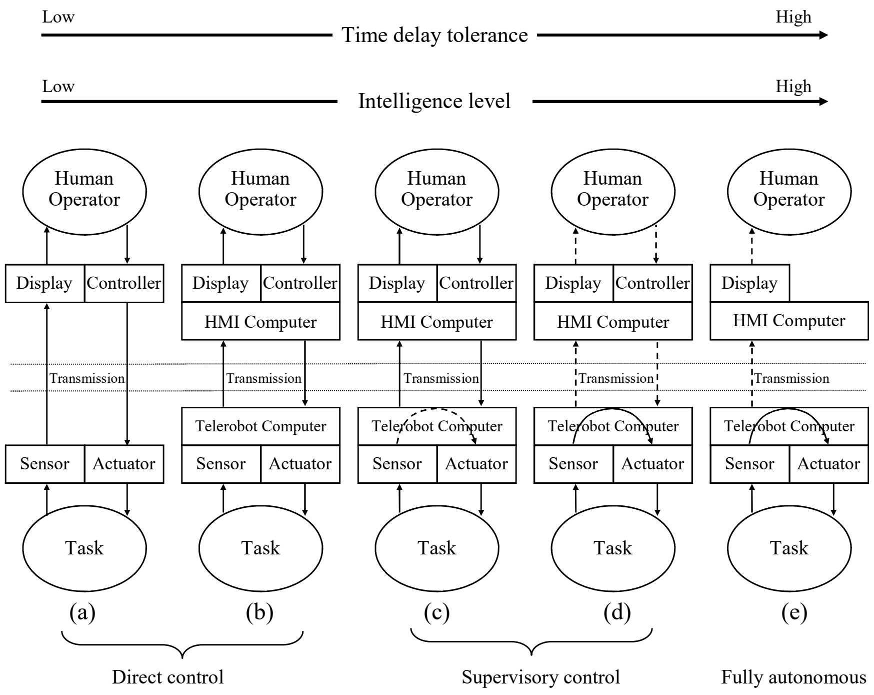


Fig. 24.1: A spectrum of teleoperation control modes adapted from Sheridan's text [1]. We label them a-e, in order of increasing robot autonomy. At the far left it would be a mechanical linkage where the human directly operates the robot from another room through sliding mechanical bars, and on far right it would be a system where the human role is limited to observation/monitoring. In c-e, the dashed lines indicated that communication may be intermittent. As the telerobot increases its intelligence level, its tolerance to time delay also increases.

navigation capabilities. The human operator only needs to update way points, **u**, intermittently. Locally, the vehicle determines its configuration **x** by constantly evaluating it against **u** while considering landmarks L and obstacles B.

- Type (e) is fully autonomous. Humans are out of the control loop. The high level decision-making algorithms are introduced to generate a plan for the robot. It does all steps in (d) but with more intelligent perception and decision making capabilities. Robot configuration **x** and environment variables L and B are used mostly at telerobot end and only transmit to the human end for occasional monitoring purposes. In vehicle terms, this is level 4 or level 5 autonomous driving mode according to National Highway Traffic Safety Administration and Society of Automotive Engineers.

It is worth noting that there is no clear boundary between adjacent types. Networked telerobots are a special case of "supervisory control" telerobots. Under supervisory control, a local computer plays an active role in closing the feedback loop. Most networked robots are supervisory control systems somewhere between type (c) and type (d).

24.2.2 Teleoperation: from Manipulators to Drones and Unmanned Aerial Vehicles

Almost all robots can be teleoperated, either as a primary working mode or as a fall-back solution in case of automation failure. At General Electric, Mosher [10] developed a two-arm teleoperator with video cameras. Teleoperation is also applied to prosthetic hands [11]. More recently, teleoperation is being considered for medical diagnosis [12], manufacturing [13] and micromanipulation [14]. These early teleoperators are mostly telemanipulators for material handling or assembly jobs which are either in inhospitable environments or require augmented skills. An example is Canadarm2 [15], a 17 metre-long robotic arm that assembled the International Space Station (ISS) while in space. Another example is the Da Vinci surgical robot [16], a vital tool for minimally invasive surgeries.

While fixed manipulators often live in structured and static environments [17], mobile robots are often required to be able to respond to more dynamic environments. At the beginning of the 21st century, new sensing and computation capabilities significantly enable mobile robots to face real world environments. As a result, we witness the fast growing of teleoperated mobile robots including ground robots, underwater remotely controlled vehicles (ROV), drones and aerial vehicles. Applications of these telerobots include planetary exploration [7], deep sea observation [18, 19], and traffic monitoring [20]. Networked robots provide a new medium for people to interact with remote environments. A networked robot can provide more interactivity beyond what a normal videoconferencing system [21] is able to. The physical robot not only represents the remote person but also transmits multi-modal feedback to the person, which is often referred as “telepresence” in literature [22]. Paulos and Canny’s Personal ROving Presence (PRoP) robot [23], Jouppi and Thomas’ Surrogate robot [22], Takayama et al.’s Texai [24], and Lazewatsky and Smart’s inexpensive platform [25] are representative examples. Generally speaking, teleoperated mobile robots require more local intelligence to overcome challenges brought by time delay in the communication channel and dynamic environments.

24.2.3 Architecture: from Single-Operator-Single-Robot to Socially-Assisted Multi-Robot System

The communication networks also enable a variety of architectures for networked robots. Although a majority of networked telerobotic systems consist of a single human operator and a single robot [22,26–32], Chong et al. [33] propose a useful taxonomy: Single Operator Single Robot (SOSR), Single Operator Multiple Robot (SOMR) [34,35], Multiple Operator Single Robot (MOSR), and Multiple Operator Multiple Robot (MOMR) [36,37]. These frameworks greatly extend the system architecture of networked robots. In fact, human operators can often be replaced with autonomous agents, off-board sensors, expert systems, and programmed logic, as demonstrated by Xu et al. [38] and Sanders et al. [39]. The extended network connectivity also allows us to employ techniques such as crowd sourcing and collaborative control for demanding applications such as nature observation and environment monitoring [40, 41]. Note that such collaborative robots or cobots are simple, smart, lightweight, and easily manipulable than the classic industrial robots. Hence networked telerobots fully evolve into networked robots: an integration of robots, humans [42], computing power, off-board sensing, and databases over the Internet.

This “open access” to hardware connects robots to the general public, who may have little to no knowledge of robots, with opportunities to understand, learn, and operate robots, which were expensive scientific equipment limited to universities and large corporate laboratories before. In fact, one of the earliest networked telerobot systems [43] originates from the idea of a remote laboratory. Built on networked telerobots, online remote laboratories [44, 45] greatly improves distance learning by providing an interactive experience. For example, teleoperated telescopes help students to understand astronomy [46]. Teleoperated microscopes [47] help student observe micro-organisms. The

Tele-Actor project [48] allows a group of students to remotely control a human tele-actor to visit environments that are normally not accessible, such as clean-room environments for semi-conductor manufacturing and DNA analysis laboratories.

24.2.4 Cloud, Edge and Fog Computing: from Local to Remote Compute and Storage

The recent development of cloud computing provides new means and platforms for networked robots. In 2010, James Kuffner at Google introduced the term "Cloud Robotics" [49] to describe a new approach to robotics that takes advantage of the Internet as a resource for massively parallel computation and real-time sharing of vast data resources. The Google autonomous driving project exemplifies this approach: the system indexes maps and images that are collected and updated by satellite, Streetview™, and crowd-sourcing from the network to facilitate accurate localization. Another example is Amazon Kiva™ System's new approach to warehouse automation and logistics using large numbers of mobile platforms to move pallets which uses a local network to coordinate platforms and updates tracking data. These are just two new projects that build on resources from the Cloud. Steve Cousins of Willow Garage aptly summarized the idea: "No robot is an island."

Cloud Robotics recognizes the wide availability of networking, incorporates elements of open-source, open-access, and crowdsourcing to greatly extend earlier concepts of "Online Robots" [50] and "Networked Robots" [51, 52]. New resources range from software architectures [53] [54] [55] [56] to computing resources [57]. For example, the RoboEarth project [58] aims to develop "a World Wide Web for robots: a giant network and database repository where robots can share information and learn from each other about their behaviors and their environments". [59].

Edge computing optimizes Cloud computing systems by processing data or services to the other logical extreme (the "edge") of the Internet in proximity with the physical world, sensors or end users [60, 61]. This can significantly reduce the amount of data and the distance the data must travel, resulting in better quality of service (QoS) with reduced latency. Fog computing, in contrast, is a decentralized architecture that distributes resources and services of computing, storage, control and networking anywhere along the continuum from Cloud to Edge [62–65]. Later in the Chapter, we motivate the need of a 'Fog Robotics' approach for robot learning and control at a large scale.

24.3 Enabling Technologies

Networked robots are enabled by emergent technologies including communication networks, sensor networks, Internet of things (IoT), mobile computing, and Cloud and Fog computing.

24.3.1 Communications and Networking

Below is a short review of relevant terminologies and technologies on networking. For details, see the texts by [66].

A communication network includes three elements: *links*, *routers/switches*, and *hosts*. Links refer to the physical medium that carry bits from one place to another. Examples of links include copper or fiber-optic cables and wireless (radio frequency or infrared) channels. Switches and routers are hubs that direct digital information between links. Hosts are communication end points such as browsers, computers, and robots.

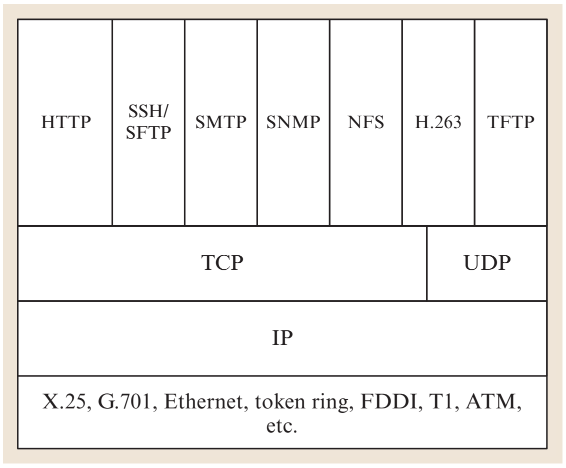


Fig. 24.2: A four-layer model of internet protocols (after [66])

Networks can be based in one physical area (local-area network, or LAN), or distributed over wide distances (wide-area network, or WAN). Access control is a fundamental problem in networking. Among a variety of methods, the *ethernet* protocol is the most popular. Ethernet provides a broadcast-capable multiaccess LAN. LANs are interconnected with each other via routers/switches. The information transmitted is in packet format. A packet is a string of bits and usually contains the source address, the destination address, content bits, and a checksum. Routers/switches distribute packets according to their routing table. Routers/switches have no memory of packets, which ensures scalability of the network. Packets are usually routed according to a first-in first-out (FIFO) rule, which is independent of the application. The packet formats and addresses are independent of the host technology, which ensures extensibility. This routing mechanism is referred to as packet switching in networking literature.

The creation of the Internet can be traced back to US Department of Defense's (DoD) ARPANET network in the 1960s. There are two features of the ARPANET network that enable the successful evolution of the Internet [67]. One feature is the ability for information (packets) to be rerouted around failures. Originally this was designed to ensure communication in the event of a nuclear war. Interestingly, this dynamic routing capability also allows the topology of the Internet to grow easily. The second important feature is the ability for heterogeneous networks to interconnect with one another. Heterogeneous networks, such as X.25, G.701, Ethernet, can all connect to the Internet as long as they can implement the Internet protocol (IP). The IP is media, operating system (OS), and data rate independent. This flexible design allows a variety of applications and hosts to connect to the Internet as long as they can generate and understand IP.

Fig. 24.2 illustrates a four-layer model of the protocols used in the Internet. On the top of the IP, we have two primary transport layer protocols: the transmission control protocol (TCP) and the user data protocol (UDP). TCP is an end-to-end transmission control protocol. It manages packet ordering, error control, rate control, and flow control based on packet round-trip time. TCP guarantees the arrival of each packet. However, excessive retransmission of TCP in a congested network may introduce undesirable time delays in a networked telerobotic system. UDP behaves differently; it is a broadcast-capable protocol and does not have a retransmission mechanism. Users must take care of error control and rate control themselves. UDP has a lot less overhead compared to TCP. UDP packets are transmitted at the sender's preset rate and the rate is changed based on the congestion of a network. UDP has great potential, but it is often blocked by firewalls because of a lack of a rate control mechanism. It is also worth mentioning that the widely accepted term TCP/IP refers to the family of protocols that build on IP, TCP, and UDP.

In the application layer of the Internet protocols, the hypertext transfer protocol (HTTP) is one of the most important protocols. HTTP is the protocol for the World Wide Web (WWW). It allows the sharing of multimedia information among heterogeneous hosts and OSs including text, image, audio, and video. The protocol has significantly contributed to the boom of the Internet. It also changes the traditional client/server (C/S) communication architecture to a browser/server (B/S) architecture. A typical configuration of the B/S architecture consists of a web server and clients with web browsers. The web server projects the contents in hypertext markup language (HTML) format or its variants, which is transmitted over the Internet using HTTP. User inputs can be acquired using the common gateway interface (CGI) or other variants. The B/S architecture is the most accessible because no specialized software is needed at the client end.

The concept of hypertext (linked references) was proposed by Vannevar Bush in 1945 and was made possible by subsequent developments in computing and networking. In the early 1990's, Berners-Lee introduced HTTP. A group of students led by Marc Andreessen developed an open source version of the first graphical user interface, the "Mosaic" browser, and put it online in 1993. The first networked camera, the predecessor of today's "webcam", went online in November 1993 [68].

Approximately nine months later, the first networked telerobot went online. The "Mercury Project" combined an IBM industrial robot arm with a digital camera and used the robot's air nozzle to allow remote users to excavate for buried artifacts in a sandbox [69,70]. Working independently, a team led by K. Taylor and J. Trevelyan at the University of Western Australia demonstrated a remotely controlled six-axis telerobot in September 1994 [71,72]. See [73–81] for other examples.

The robots and the software system communicate to each other with a middleware. Majority of the robots today use *Robot Operating System (ROS)* as a middleware that provides a reusable set of libraries and tools across different platforms [82]. ROS has become the de-facto standard in robotics for communication with sensors, actuators and drivers in a wide variety of applications including research, manufacturing, agriculture, home and space robots. The communication is based on three levels: 1) filesystem level, containing packages, metapackages, repositories, etc; 2) computation graph level, defining peer-to-peer network of processes comprising of nodes, master, parameter server, messages, services, topics, and bags. The ROS master provides name registration and lookup to the rest of the computation graph to facilitate communication between nodes by passing messages and services, while the parameter server allows data to be stored by key in a central location. The messages are sent/received with a publish/subscribe mechanism from a node to a given topic; 3) community level, including a collection of software distributions and repositories. Use of ROS has largely standardized robot communication across a wide range of robots in academia and industry.

24.3.2 Mobile Computing, Augmented Reality and Virtual Reality

One significant event in 21st century is the rise of mobile computing. Cellphones, tablets, wearable devices, and augmented reality devices are part of this fast growing technology trend. As an evidence of being primary data terminal, mobile data browsing surpassed desktop data browsing in October of 2016, according to web analytics firm StatCounter™. In the subsequent month, Google started its mobile-first index, which primarily looks at the mobile version of your website for its ranking signals and fall back on the desktop version when there is no mobile version. All of these indicate the significant shift of computation platform from desktop computing to mobile computing. Networked telerobots are inevitably being significantly changed by the new technology. Mobile devices often come with an array of embedded sensors and are inherently connected to cloud services. The former significantly changes the way that humans can interact with telerobots and the latter invokes a more powerful computation architecture to support robots. A mobile phone goes beyond a nimble desktop computer replacement with broadband and widely accessible mobile communication capability [83]. The on-board camera and motion sensors can be used as additional sensors to assist the robot if the mobile phone is employed at the teleoperator side. At the human side, its motion sensors can also behave as a haptic interface [84] and its augmented reality display can further enhance the teleoperation experience [85].

Mobile computing also significantly accelerates virtual reality (VR) and augmented reality (AR) applications. VR and AR provide new interfaces for human operators to interact with telerobots. Initially developed for desktop computer gaming, modern VR renders 3D environments in headsets. Appearing more and more realistic, most of these 3D environments are the results from game rendering engine in VR applications. As of 2018, popular VR headsets include Oculus Rift™, Google Daydream View™, HTC Vive™, etc. AR technology is more complex. It usually overlays virtual object display with camera inputs. It requires the system can accurately compute camera pose with respect to the environment in real time and track eye gaze to allow insertion/overlay of virtual objects in the display. As of 2018, popular AR headsets include Microsoft HoloLens™and Magic Leap One™. AR can also be directly implemented in mobile phones such as Apple ARKit™, Google ARCore™, etc. VR applications are often used to simulate robot motions and can be used as a predictive display for teleoperation.

24.3.3 Internet of Things (IoT)

Recent developments in IoT are relevant to networked robots. IoT refers to a network of physical devices, smart home appliances, sensors, actuators that facilitates these devices to communication and exchange data. IoTs are mostly employed for environment monitoring and control applications. From planetary scale radio telescope arrays [86] to human body area networks [87], IoTs can have many shapes and formats. IoTs can assist networked robots in three aspects: mission-related information, communication, and navigation.

Mission-related information refers to environmental variables that a robot needs to accomplish their missions. For example, motion sensors may provide intruder location for security robots [88]. Thermal sensors may provide temperature field information for robots deployed to assist first responders [89, 90]. Humidity sensors may provide soil moisture information [91] for irrigation robots in precision agriculture. Underwater acoustic sensors may assist autonomous underwater vehicles (AUVs) to hunt for enemy submarines [92]. Robots may also behave as mobile nodes in IoT [93, 94]. IoT may provide communication capabilities to assist networked robots by serving as routers to extend communication coverage [95]. Robots may also serve as data mules to assist communication [96]. IoT can also directly help a robot in navigation tasks by serving as beacons/landmarks [97–99]. In these cases, the end devices become the landmark set and the robot has this set pre-stored in its memory for quick indexing/localization.

This growth in the number of active devices in IoT presents a unique challenge with issues related to privacy, security, scalability, latency, bandwidth, availability and durability control. Edge and Fog Computing addresses these challenges by moving the computation and storage using resources that are closer to the devices.

24.4 Properties of Networked Robots

Terminology

Consider a networked robot described by its state $\mathbf{x} \in \mathbb{C}$, where $\mathbb{C}$ is the robot configuration space. The dimensionality of a robot configuration space is the degrees of freedoms (DoFs) that the robot's mechanism provides. A typical industrial manipulator may have 6 DoFs due to the standard six-joint open-chain configuration. For manipulator, $\mathbf{x}$ here denotes a joint-space representation. Forward kinematics can be applied to convert $\mathbf{x}$ to end-effector pose (position and orientation) $\mathbf{x}_e \in SE(3)$

$$\mathbf{x}_e = f(\mathbf{x}), \tag{24.1}$$

while the inverse kinematics does the opposite transformation

$$\mathbf{x} = f^{-1}(\mathbf{x}_e). \tag{24.2}$$

where function $f(\cdot)$ depends on robot mechanism (see [100]). A mobile robot without arms in 3D space has $\mathbb{C}$ reduced to $SE(3)$.

The world coordinate system or work space is defined as $W \subset \mathbb{R}^3$, with $\mathbf{w} \in W$ denoting a point in the work space. The human input vector is $\mathbf{u} \in \mathbb{R}^{d_u}$ where d_u is input dimension. Ideally, human inputs $\mathbf{u}$ are in frame W. However, due to the limitation introduced by the input device, it may need to be estimated or remapped to the $SE(3)$ as controller inputs for the robot. For a mobile robot, its environment can be characterized by the map that it navigates in. It takes either a prior map or a map constructed by its on-board simultaneous localization and mapping (SLAM) algorithm. The map is often represented as a set of l landmark points $L := \{\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_l\} \subset W$. There are often obstacles in the environment as characterized by the obstacle set $B \subset W$. These variables provide a minimalist view of the robot and the world that the robot lives. The networking and communication paradigms enable the robot to exchange these variables with other robots, humans, external sensors, and computation resources which drastically increases robot capabilities and robustness.

24.4.1 Overall Structure

Fig. 24.3 illustrates how networked robots, such as automatic forklift, ground vehicles, and manipulators, are employed in a manufacturing factory. Networked robots have the following properties

- The physical world is affected by one or more robots that are locally controlled by a network *server*, which connects to the Internet to communicate with remote human users, databases, agents, and off-board sensors, which are referred to as *clients* of the system.
- Human decision making capability is often an integral part of the system. If so, humans often access the robot via web browsers, such as Firefox™, Chrome™, Safari™or Edge™, or apps in mobile devices. As of 2018, the standard protocol for network browsers is the HTTP, a stateless transmission protocol.
- Most networked robots are continuously accessible (online), 24 hours a day, 7 days a week.
- Networks may be unreliable or have different speed for clients with different connections.
- Since billions of people now have access to the Internet, mechanisms are needed to handle client authentication and contention. System security and privacy of users are important in networked robots.

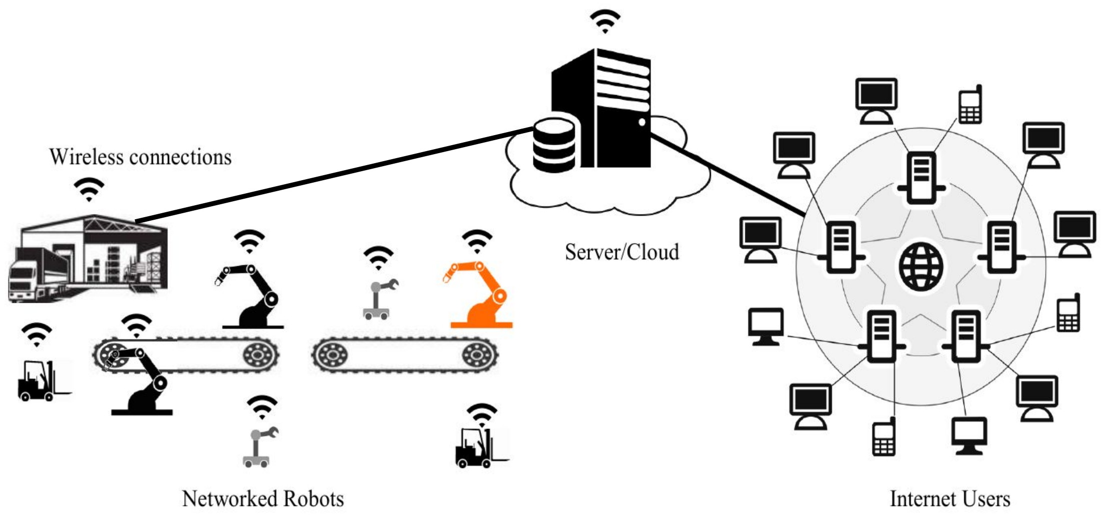


Fig. 24.3: A sample architecture of networked robots used in manufacturing. Automatic forklifts, autonomous ground vehicles, and manipulators are all connected to a server or a cloud server through wireless connections. Workers, managers, and relevant clients can teleoperate or interact with the robots at various authorization levels.

- Input and output for human users for networked robots are usually achieved with mobile devices or desktop computers with the standard computer screen, mouse, and keyboard.
- Clients may be inexperienced or malicious, so online tutorials and safeguards are generally required.
- Additional sensing, databases and computing resources may be available over the network.

As defined by Mason, Peshkin, and others [101, 102], in *quasistatic* robot systems, accelerations and inertial forces are negligible compared to dissipative forces. In quasistatic robot systems, motions are often modeled as transitions between discrete atomic *configurations*.

For exmaple, in quasistatic telerobotics (QT), robot dynamics and stability are handled locally. After each atomic motion, a new state report is presented to the remote user, who sends back an atomic command. The atomic state describes the status of the robot and its corresponding environment. Atomic commands refer to desired robotic actions.

Several issues arise

- *State-command-environment representation*: At time t, how should robot state $\mathbf{x}(t)$, available commands $\mathbf{u}(t)$, and environment information (e.g. L and B for mobile robots) be presented to remote human operators using their display?
- *Command execution/state generation*: How should commands $\mathbf{u}(t)$ be executed locally at each robot to ensure that the desired state $\mathbf{x}(t+1)$ is achieved and maintained by the robot?
- *Information fusion*: How should robots exchange their state $\mathbf{x}(t)$ and environment understandings (e.g. L and B) to facilitate collaboration and coordination among robots and human operators given that these variables may have different uncertainties and perspective limitations?
- *Command coordination*: How should commands be resolved when there are multiple human operators and/or agents? How to synchronize and aggregate commands issued by users/agents with different network connectivity, background, responsiveness, error rate, etc. to achieve the best possible system performance?

- *Error prevention and state correction*: How should the system prevent the wrong commands that may lead the robot to collision or other undesirable states?

Before we detail these issues, let us walk through how to build an example of a networked robot system. A reader can follow the example below to build his/her own networked robot system as well as understand challenges and issues.

24.4.2 A Example Networked Robot System

A simple networked telerobotic system allows a group of users to access a single robot manipulator (i.e. the orange robot in Fig. 24.3) via web browsers. This typical or minimal networked telerobotic system includes three components:

- users: anyone with an Internet connection and a web browser or equivalent apps that understand HTTP.
- web server: a computer running a web server software.
- robot: a robot manipulator, a mobile robot, or any device that can modify or affect its environment.

Users access the system via their web browsers. Any web browser that is compatible with W3C's HTML standard can access a web server. A mobile browser may be implicitly called by mobile apps in mobile devices such as Apple iPads™, Apple iPhones™, and Google Andriod-based tablets and smart phones.

A web server is a computer that responds to HTTP requests over the Internet. Depending upon the operating system of the web server, popular server software packages include Apache™, Nginx™, and Microsoft Internet Information Services (IIS)™. Most servers can be freely downloaded from the Internet.

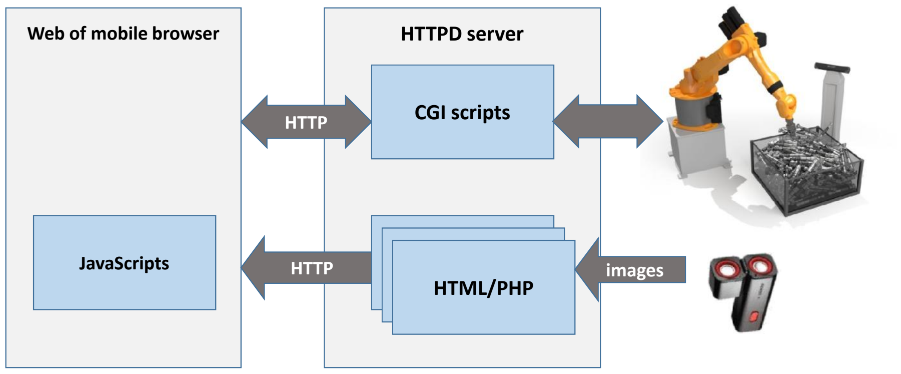


Fig. 24.4: A sample software architecture of a networked telerobot.

To develop a networked telerobot, one needs a basic knowledge of developing, configuring, and maintaining web servers. As illustrated in Fig. 24.4, the development requires knowledge of HTML and at least one local programming language such as C, C#, CGI, Javascript, Perl, PHP, Python, .Net, or Java.

It is important to consider compatibility with the variety of browsers. Although HTML is designed to be compatible with all browsers, there are exceptions. For example, Javascript, which is the embedded scripting language of web browsers, is not completely compatible across different browsers. One also needs to master the common HTML components such as forms that are used to accept user inputs, frames that are used to divide the interface into different functional regions, etc. An introduction to HTML can be found in [103].

User commands are usually processed by the web server using CGI, the common gateway interface. Most sophisticated methods such as PHP, Java Server Pages (JSP), and socket-based system programming can also be used. CGI is invoked by the HTTP server when the CGI script is referred in the Uniform Resource Locator (URL). The CGI program then interprets the inputs, which is often the next robot motion command, and sends commands to the robot via a local communication channel. CGI scripts can be written in almost any programming language. The most popular ones are Perl and C. A simple networked telerobotic system can be constructed using only HTML forms and CGI. However, if the robot requires a sophisticated control interface, Javascript offers a better solution.

Most telerobotic systems also collect user data and robot data. Therefore, database design and data processing program are also needed. The most commonly used databases include MySQL™ and PostgresSQL™. Both are open-source databases and support a variety of platforms and operation systems. Since a networked telerobotic system is online 24 hours a day, reliability is also an important consideration in system design. Website security is critical. Other common auxiliary developments include online documentation, online manual, and user feedback collection.

24.4.3 State, Command, and Environment Representations

Robots listen to commands given by humans. To generate a correct and high-quality command depends on how effectively the human operator understands the state feedback and the working environment. Also, an efficient representation can help exchange environmental understanding and enhance collaboration among robots. However, the representations of the three type of information are often limited by hardware capabilities and communication bandwidth. Let us begin with state displays.

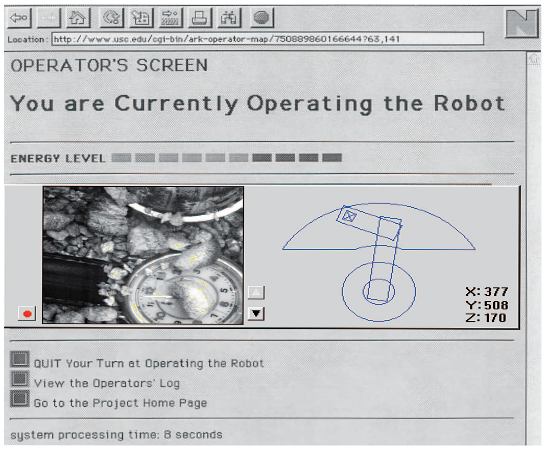
24.4.3.1 State Display

State display visualizes robot state $\mathbf{x}$ in the human operator's display device. It serves as a feedback mechanism for a human operator to understand robot pose with respect to its target or object of interest. Therefore, the human can generate new commend $\mathbf{u}$ accordingly.

Unlike traditional point-to-point teleoperation, where specialized training and equipment are available to operators, networked telerobots offer wide access to the general public. Designers can only assume common consumer level hardware. There are two most commonly-used types: traditional 2-D screen displays or 3-D AR/VR headsets.

Early networked telerobotic systems often display the robot state on a 2-D screen display because users are mostly desktop computer users. The states of the teleoperated robot are often characterized in either world coordinates or robot joint configuration, which are either displayed in numerical format or through a graphical representation. Fig. 24.5a lists robot *XYZ* coordinates on the interface and draws a simple 2-D projection to indicate joint configurations. Fig. 24.5b illustrates another example of the teleoperation interface that was developed by *Taylor* and *Trevelyan* [43]. In this interface, *XYZ* coordinates are presented in a sliding bar near the video window.

The state of the robot is usually displayed in a 2-D view as shown in Figs. 24.5a and 24.5b. In some systems, multiple cameras can help the human operator understand the spatial relationship between the robot and the objects in the surrounding environment. In fact, video feedback is a primary way for the human operator to observe robot relative pose due to its intuitiveness. Fig. 24.5c shows an example with four distinct camera views for a six-degree-of-freedom


Location : http://www.usc.edu/cgi-bin/ark-operator-map/7508898601666447?63,141

OPERATOR'S SCREEN

You are Currently Operating the Robot

ENERGY LEVEL

X: 377
Y: 508
Z: 170

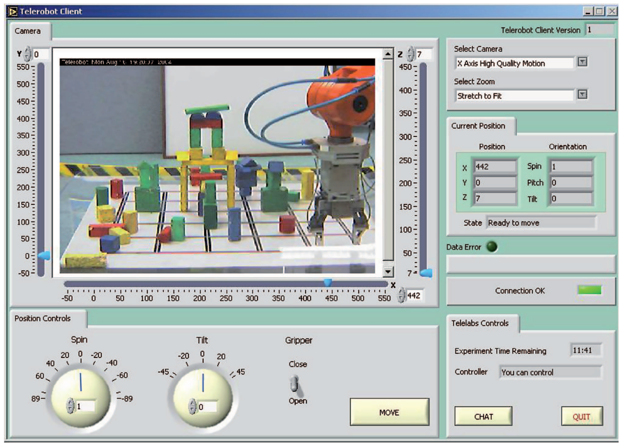
QUIT Your Turn at Operating the Robot

View the Operators' Log

Go to the Project Home Page

system processing time: 8 seconds

(a)



Telerobot Client Version 1

Select Camera: X Axis High Quality Motion

Select Zoom: Stretch to Fit

Current Position

	Position		Orientation
X	442	Spin	1
Y	0	Pitch	0
Z	7	Tilt	0

State: Ready to move

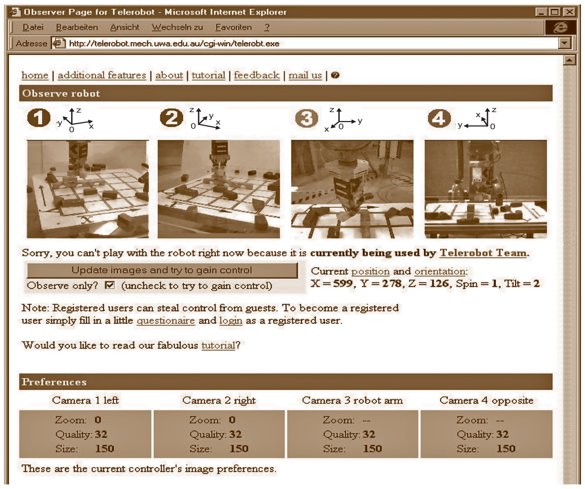
Data Error

Connection OK

Experiment Time Remaining: 11:41

Controller: You can control

(b)


Observer Page for Telerobot - Microsoft Internet Explorer

Adresse http://telerobot.mech.uwa.edu.au/cgi-win/telerobt.exe

home | additional features | about | tutorial | feedback | mail us |

Observe robot

Sorry, you can't play with the robot right now because it is **currently being used by** Telerobot Team.

Update images and try to gain control

Observe only? ☑ (uncheck to try to gain control)

Current position and orientation:
X = 599, Y = 278, Z = 126, Spin = 1, Tilt = 2

Note: Registered users can steal control from guests. To become a registered user simply fill in a little questionaire and login as a registered user.

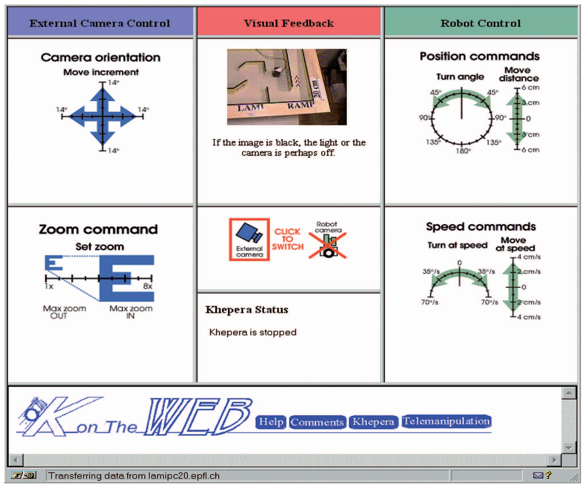
Would you like to read our fabulous tutorial?

Preferences

Camera 1 left	Camera 2 right	Camera 3 robot arm	Camera 4 opposite
Zoom: 0	Zoom: 0	Zoom: --	Zoom: --
Quality: 32	Quality: 32	Quality: 32	Quality: 32
Size: 150	Size: 150	Size: 150	Size: 150

These are the current controller's image preferences.

(c)



If the image is black, the light or the camera is perhaps off.

Khepera Status

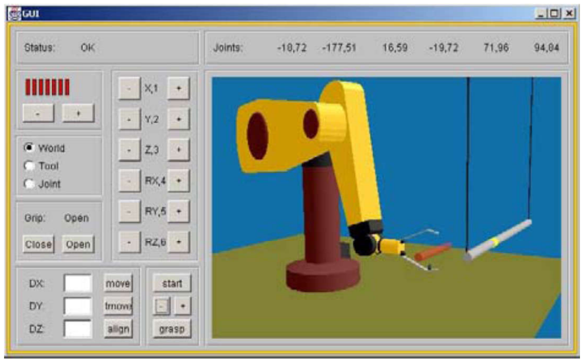
Khepera is stopped

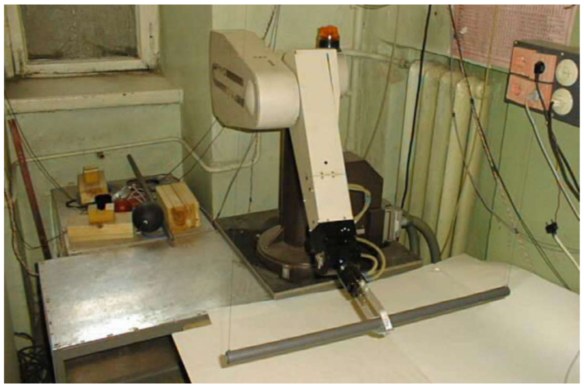
Transferring data from lamipc20.epfl.ch

(d)

Fig. 24.5: **(a)** Browser's view of the first networked telerobot interface [104]. The schematic at *lower right* gives an overhead view of position of the four-axis robot arm (with the camera at the end marked with X), and the image at the *lower left* indicates the current view of the camera. The *small button marked with a dot at the left* directs a 1 s burst of compressed air into the sand below the camera. **(b)** Browser interface to the Australian networked telerobot which was a six-axis arm that could pick up and move blocks [72]. **(c)** Use of a multi-camera system for multi-viewpoint state feedback [105]. **(d)** Camera control and mobile robot control in Patrick Saucy and Francesco Mondada's Khep on the web project.

industrial robot. Fig. 24.5d demonstrates an interface with a pan–tilt–zoom robotic camera. The interface in Fig. 24.5d is designed for a mobile robot.





(a) (b)

Fig. 24.6: A web-based teleoperation system that allows a robot to capture a fast-moving rod [27] **(a)** user interface and **(b)** system setup.

The fast development of AR and VR technology makes it possible to visualize robot and environment states in 3D displays and generate synthetic eco-centric views (a.k.a. a third person views). VR requires a complete 3D model of the robot and its working environments. It usually employs a game engine to model environment changes as the robot interacts with the environment. VR often functions as a great predictive display to examine a command before it is sent to the real robot. When directly interacting with the teleoperator, AR is a better choice than VR because it is difficult to synchronize environment changes in VR. To enable AR, it often requires the robot to be equipped with multiple cameras and laser range finders to quickly reconstruct the remote environment [106, 107] and to estimate its real time pose with respect to the environment. The reconstructed sensory information can be superimposed on previously known 3D information which drastically increases telepresence and performance [107].

Developers may also use sound feedback and text-to-speech capabilities to augment state displays with audio channels. Audio channels are very useful when users' visual channel is pre-occupied. However, it is not a good idea to readout coordinates at low level as it dramatically increases cognitive workload. Task level feedback or warnings are the good audio messages.

24.4.3.2 Human Operator Input

Most networked telerobotic systems only rely on commonly available hardware to generate control inputs **u**. For personal computers, this is often limited to a keyboard and a 2D pointing device such as a mouse, a touch-pad, a track stick, etc. 3D pointing devices exist but are rare. For mobile devices, its touch screen is a 2D pointing device. However, its motion sensors may also be used as the advanced input devices. A new development is voice input as voice recognition becomes more and more prevalent in mobile devices.

We start with 2D pointing devices and keyboards for input because they are the most commonly used input mechanisms. The design problem is what to click on in the interface. Given the fact that user commands can be quite different, we need to adopt an appropriate interface for inputs; for example, inputs could be Cartesian *XYZ* coordinates in world coordinate system or robot configurations in angular joint configurations. For angular inputs, it is often suggested to use a round dial as a control interface, as illustrated in bottom left of Fig. 24.5b and the right-hand side of Fig. 24.5d.

For linear motion in Cartesian coordinates, arrows operated by either mouse clicks or the keyboard are often suggested. Position and speed control are often needed, as illustrated in Fig. 24.5d. Speed control is usually controlled by mouse clicks on a linear progress bar for translation and a dial for rotation.

The most common control type is position control. The most straightforward way is to click on a projected plane (e.g. the video image) directly. To implement the function, the software needs to translate the 2-D click inputs into robot or end effector configuration in $SE(3)$. To simplify the problem, the system designer usually assumes that the clicked position is on a fixed plane; for example, a mouse click on the interface of Fig. 24.5a assumes the robot moves on the X–Y plane. For orientation inputs, entering rotation angles in numeric format is often confusing. A more desirable way is to employ a 3D model rendering (e.g. Fig. 24.6a) to visualize and adjust the input. Forward and inverse kinematics Eqs. 24.1 and 24.2 are often included in the model to facilitate the shift between joint mode and world mode controls. Mobile devices equipped with an inertial measurement unit (IMU) can directly measure device orientation and use that as a convenient and intuitive orientation input.

Combined with an object recognition algorithm, a mouse click on the recognized objects in an image can enable abstract task-level command. The example in Fig. 24.10b uses mouse clicks to place votes on an image to generate a command that directs a robot to pick up a test agent. More sophisticated spatial reasoning can eliminate the need for humans to provide low-level control by automatically generating a sequence of commands after it receives task-level commands from the human operator. This is particularly important when the robotic system is highly dynamic and requires a very fast response. In such cases, it is impossible to ask the human to generate intermediate steps in the robot control; for example, *Belousov* et al. adopt a shared autonomy model to direct a robot to capture a moving rod [27] as shown in Fig. 24.6. *Fong* and *Thorpe* [108] summarize vehicle teleoperation systems that utilize these supervisory control techniques. *Su* et al. develop an incremental algorithm for better translation of the intention and motion of operators into remote robot action commands [31].

Voice-based control has become another intuitive input mechanism for task level commands. Recent developments in voice recognition algorithms and their widespread use in mobile devices make it a natural input channel for applications where hands may not be available. A number of existing digital assistants including Amazon Alexa™, Apple Siri™, Microsoft Cortana™, and Google Assistant™ show the potential of the voice-based input. While understanding generic conversation is still difficult, today's voice recognition engines are reasonably accurate when working with a limited set of vocabulary under a known context.

24.4.3.3 Environment Representation

Humans need to understand the working environment for robots through on-board sensors. An efficient and accurate environment representation is important because it determines situational awareness and consequently, task performance. Constraints such as communication bandwidth and computation power often limit the choice of environment representation.

Environments can be classified as a combination of two parts: stationary or dynamic. For an industrial manipulator performing an assembly job in a fixed work cell, its environment is largely stationary. A representation of a stationary environment can be priorly generated and shared among all robots and human operators. Therefore, minimal communication bandwidth is required. As another example, Google Maps™ and Streetview™ is one of the largest prior environment representations for ground robots.

When prior environment representation does not exist or facing a dynamic environments, the robot has to generate its own environment representation and share it over the communication network. Such cases often happen to indoor mobile robots. What type of environment representation the robot can generate also depends on its on-board sensor types. A lidar, a regular camera, and/or a RGB-D camera are frequently used sensors. In general there are

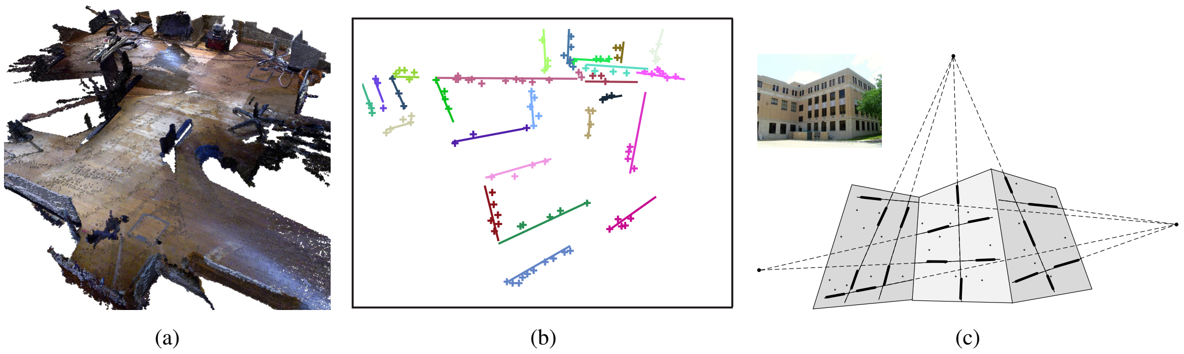


Fig. 24.7: **(a)** Dense environment representation [109]. **(b)** Line-based environment representation [110]. **(c)** Multi-layered feature graph [111, 112].

sparse representations and dense representations. A dense representation often refers to detailed 3D reconstruction with dense point clouds with color information. Fig. 24.7a shows an example of the dense representation generated by a robot with a RGB-D camera. Dense representations are easy for human operators to understand the environment but communication bandwidth requirement is high. If communication bandwidth is limited, sparse representations can be used. Fig. 24.7b shows a resulting map from a line-based map representation. This representation can be easily shared among robots but may be difficult for human to understand. Fig. 24.7c illustrates a trade-off approach, a multi-layered sparse 3D reconstruction consisting of points, parallel lines, and planes. More specifically, the sparse representation is just the extension of the landmark set L. If we also share the obstacle set B, then B and L are usually aligned in the same coordinate system.

24.4.3.4 Information Fusion

The network also allows robots to exchange their states $\mathbf{x}$ and commands $\mathbf{u}$ to achieve collaborative tasks. Chapter 23 (Multiple Robots) covers robot teams for various tasks such as coverage, mapping, etc. Here we want to emphasize that robots can also combine their sensory data to achieve better joint understandings of the environment. We name this as *information fusion*. Fig. 24.8 illustrates a case where a low-cost robot equipped with monocular camera can share information with a robot equipped with a 2D lidar. The robot with monocular vision has a vision-based environment representation. Monocular visual SLAM algorithms often generate low-quality maps due to scale and angular drift, as illustrated in the figure. Lidars are more expensive and power hungry than cameras. According to [113], fusion of the information can allow low costs robot to improve their map quality significantly. An effective information fusion can allow robots to share their sensory capabilities and reduce overall cost of the robot team. Of course, it is necessary to address issues caused by data correspondence and synchronization issues in this heterogeneous sensory information fusion [114].

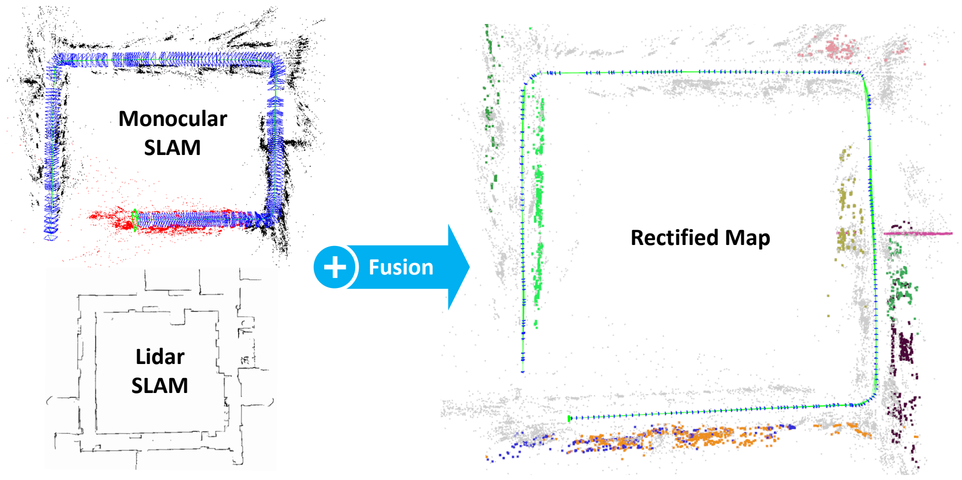


Fig. 24.8: An illustration of information fusion between a robot with a monocular camera and a robot with a 2D lidar.

24.4.4 Level of Autonomy

In a networked telerobot example, the human operator may drive the remote robot using either direct control, or using shared or supervised autonomy. Direct control lacks the autonomy/intelligence to assist the operator and the remote robot simply mimics the movement of the operator. Shared autonomy fine-tunes/complements the continuously streamed operator data by local sensory feedback on the remote side. For constrained tasks, *virtual fixtures* can be used to reduce the operator workload by influencing the robot motion along desired paths [115, 116]. Supervised autonomy enables the operator to control the robot by issuing high-level commands only. We briefly review the assistive autonomy modes here (see Fig. 24.9 for an overview) [117].

24.4.4.1 Shared Autonomy

When a robot receives a command, it executes the command and a new state is generated and transmitted back to the human operator. However, commands may not arrive in time or may get lost in transmission. Also, because users are often inexperienced, their commands may contain errors. Over the limited communication channel, it is impossible to ask the human to control a fast moving robot directly. It is important to equip the robot with sufficient local intelligence to realize the robot control through the *shared autonomy*.

Belousov and colleagues demonstrated a system that allowed a web user to capture a fast rod that is thrown at a robot manipulator [27]. The rod is on bifilar suspension, performing complicated oscillations. Belousov et al. designed a shared-autonomy control to implement the capture. First, an operator chooses the desired point for capture on the rod and the capture instant using a 3-D online virtual model of the robot and the rod. Then, the capturing operation is performed automatically using a motion prediction algorithm that is based on the rod's motion model and two orthogonal camera inputs, which perceive the rod's position locally in real time. This *shared autonomy* approach is often required when the task execution requires much faster response than the Internet can allow. Human commands have to remain at task level instead of directing the movements of every actuator. The root of this approach can be traced back to the “Tele-Autonomous” concept proposed by *Conway*, *Volz*, and *Walker* [118] in 1990. In the paper, two important notions including time clutch and position clutches are introduced to illustrate the shared autonomy approach. The time clutch disengages the time synchronization between the human operator and the robot. The human operator verifies his/her commands on a predictive display before sending a set of verified commands to remote robots. The robot can then optimize the intermediate trajectory proposed by the human operator and disengage the position correspondence, which is referred to as the position clutch. Recent work [119] uses a similar idea to guide load-haul-dump vehicles in underground mines by combining human inputs with a tunnel following behavior.

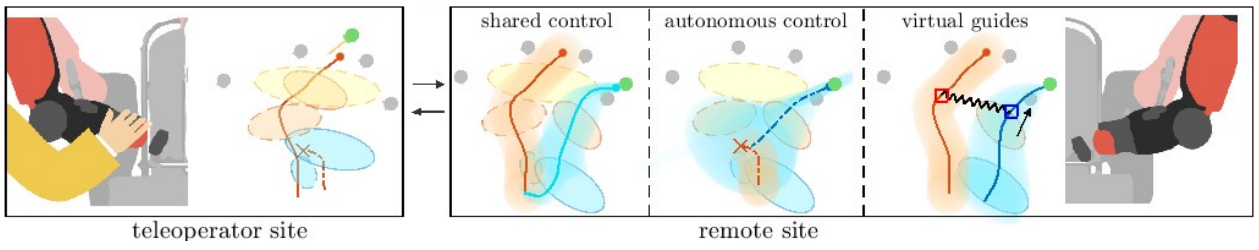


Fig. 24.9: Networked telerobot grasps an object: *(left)* the teleoperator performs the imprecise movement (in orange) to grasp the perceived object in green, *(right)* a model learned from a few demonstrations (shown as ellipses) is used to assist the teleoperator under different autonomy levels: the shared autonomy corrects the movement of the robot (in blue) locally in accordance with the actual object position on the remote site; the virtual fixture guides the movement of the remote arm along the desired path; while the supervised autonomy generates the movement to the object after the teleoperator switches to the autonomous mode (marked with a cross) [117, 120].

24.4.4.2 Virtual Fixtures

Due to time delay, lack of background, and possible malicious behavior, human errors are inevitably introduced to the system from time to time. Erroneous states may be generated from incorrect commands. If unchecked, robots or objects in the environment may be damaged. Sometimes, users may have an good intention, but are not able to generate accurate commands to control the robot remotely. For example, it is hard to generate a set of commands to direct a mobile robot to move along the wall and maintain a distance of 1 meter to the wall at the same time.

Virtual fixtures are designed to cope with these challenges in teleoperation tasks. Proposed by Rosenberg [115], virtual fixtures are defined as an overlay of abstract sensory information on a robot work space in order to improve the telepresence in a telemanipulation task. To further explain the definition, Rosenberg uses a ruler as an example. It is very difficult for a human to draw a straight line using bare hands. However, if a ruler, which is a physical fixture, is provided, then the task becomes easy. Similar to a physical fixture, a virtual fixture is designed to guide robot motion through some fictitious boundaries or force fields, such as virtual tubes or surface, generated according to sensory data. In fact, the ruler can be easily described by an inequality in configuration space, $v^{\mathrm{T}}\mathbf{x} \leq 0$, where v is the line coefficients. More sophisticated virtual fixtures are often implemented using control laws [121, 122] based on a “virtual contact” model.

Virtual fixtures serve two main purposes: avoid operation mistakes and guide robots along desirable trajectories. This is also a type of shared autonomy that is similar to that in Section 24.4.4.1 where both the robot and the human share control in the system. It is worth noting that virtual fixtures should be visualized in the display to help operators understand the robot state to maintain situation awareness. This actually turns the display to *augmented reality* [123], as we discussed in Section 24.4.3.2.

24.4.4.3 Supervised Autonomy

Continuous control of the remote robot for routine tasks can be cumbersome for the human operator, especially in the presence of communication latency. Supervisory or autonomous control gives local autonomy to the remote robot to execute tasks in the presence of large communication delays. It makes use of predictive displays and high-level symbolic commands of atomic structure (such as reach, grasp, etc.) to breakdown a task in smaller subtasks [124, 125]. The supervised autonomy allows the operator to execute the task for the next time horizon in an autonomous manner. When the task is accomplished or the communication channel is re-established, the operator may switch back to the direct/shared control.

24.4.5 Collaborative Control and Crowd Sourcing

When more than one human is sharing control of the device, command coordination is needed. According to [127], multiple human operators can reduce the chance of errors, cope with malicious inputs, utilize operators' different expertise, and train new operators. In [128, 129], a collaboratively controlled networked robot is defined as a telerobot simultaneously controlled by many participants, where input from each participant is combined to generate a single control stream.

When group inputs are in the form of direction vectors, averaging can be used as an aggregation mechanism [130]. When decisions are distinct choices or at the abstract task level, voting is a better choice [48]. As illustrated in Fig. 24.10a, *Goldberg* and *Song* develop the Tele-Actor system using spatial dynamic voting. The Tele-Actor is a human equipped with an audio/video device and controlled by a group of online users. Users indicate their intentions by positioning their votes on a voting image during the voting interval. Votes are collected at the server and used to determine the Tele-Actor's next action based on the most requested region on the voting image.

Another approach to collaboratively control a networked robot is the employ a optimization framework. *Song* and *Goldberg* [126, 131] developed a collaboratively controlled camera that allowed many clients to share control of its camera parameters, as illustrated in Fig. 24.10b. Users indicate the area they want to view by drawing rectangles on a panoramic image. The algorithm computes an optimal camera frame with respect to the user satisfaction function, which is defined as the frame selection problem [132, 133]. Xu etc. [38, 134] further the optimization framework to p-frames that allows multiple cameras to be controlled and coordinated whereas human inputs can also be replaced by autonomous agents and other sensory inputs. These developments have been applied to a recent project, the Collaborative Observatory for Nature Environments (CONE) project [135], which aims to design a networked robotic camera system to collect data from the wilderness for natural scientists.

One important issue in collaborative control is the disconnection between individual commands and the robot action, which may lead to loss of situation awareness, less participation, and eventual system failure. Inspired by engaging power in scoring systems in computer games, Goldberg et al. [136] design scoring mechanism for the collaborative control architecture by evaluating individual leadership level. The early results show great improvement in group performance. The resulting new architecture can be viewed as a crowd sourcing [40, 137] approach to networked robots that combines human recognition and decision making capabilities with robot execution at a different scale and depth than a regular teleoperation system.

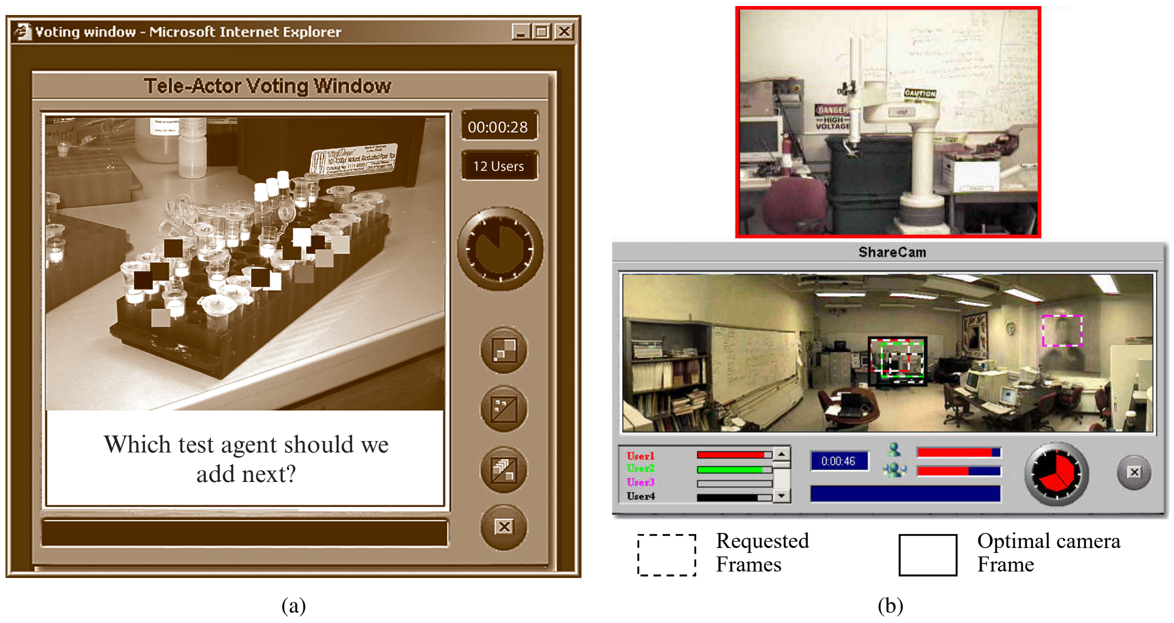


Fig. 24.10: **(a)** Spatial dynamic voting interface for the Tele-Actor system [48]: the spatial dynamic voting (SDV) interface as viewed by each user. In the remote environment, the Tele-Actor takes images with a digital camera, which are transmitted over the network and displayed to all participants with a relevant question. With a mouse click, each user places a color-coded marker (a *votel* or voting element) on the image. Users view the position of all votels and can change their votel positions based on the group's response. Votel positions are then processed to identify a *consensus region* in the voting image that is sent back to the Tele-Actor. In this manner, the group collaborates to guide the actions of the Tele-Actor. **(b)** Frame selection interface [126]. The user interface includes two image windows. The lower window displays a fixed panoramic image based on the camera's full workspace (reachable field of view). Each user requests a camera frame by positioning a *dashed rectangle*. Based on these requests, the algorithm computes an optimal camera frame (shown with a *solid rectangle*), moves the camera accordingly, and displays the resulting live streaming video image in the upper window.

24.5 Cloud Robotics

Cloud Robotics encompasses any robot or automation system that relies on either data or code from a network to support its operation, i.e., where not all sensing, computation, and memory is integrated into a single standalone system as defined in the survey article [138]. Cloud computing provides robots with vast resources in computation, memory, programming. Each robot accesses the resources over a centralized network that enables sharing of data across applications and users. The use of modern remote data centers provides the means to cut down the size and cost of the robots, while making them more intelligent to deal with uncertainties in the real world. In addition, the cloud removes overheads for maintenance and updates, and reduces adhoc dependencies on custom middleware. The cloud services can be provided in three ways: 1) Infrastructure as a Service (IaaS): cloud resources including servers, networking, storage are provided on a pay-per-use basis as an Internet service, 2) Platform as a Service (PaaS): a standalone cloud-based environment is provided to support complete lifecycle of developing cloud applications, 3)

Fig. 24.11: The Cloud has potential to enable a new generation of robots and automation systems to use wireless networking, Big Data, Cloud Computing, statistical machine learning, open-source, and other shared resources to improve performance in a wide variety of applications including assembly, inspection, driving, warehouse logistics, caregiving, package delivery, house- keeping, and surgery.

Software as a Service (SaaS): cloud applications run on distant computers and users typically connect over a web browser.

24.5.1 Potential Benefits of Cloud Robotics

Here we review five ways that cloud robotics and automation can potentially improve robots and automation performance: 1) providing access to global libraries of images, maps, and object data, eventually annotated with geometry and mechanical properties, 2) massively-parallel computation on demand for demanding tasks like optimal motion planning and sample-based statistical modeling, 3) robot sharing of outcomes, trajectories, and dynamic control policies, 4) human sharing of "open-source" code, data, and designs for programming, experimentation, and hardware construction, and 5) on-demand human guidance ("call centers") for exception handling and error recovery. Updated information and links are available at [139].

24.5.1.1 Big Data

The term "Big Data" describes data sets that are beyond the capabilities of standard relational database systems, which describes the growing library of images, maps, and many other forms of data relevant to robotics and automation on the Internet. One example is grasping, where online datasets can be consulted to determine appropriate grasps. The Columbia Grasp dataset [140] and the MIT KIT object dataset [141] are available online and have been widely used to evaluate grasping algorithms [142] [143] [144] [145]. Brekeley robotics and automation as a service (BraaS) [146] and Dex-Net as a Service (DNaaS) [147] are recent efforts to provide online access to robust grasp planning systems.

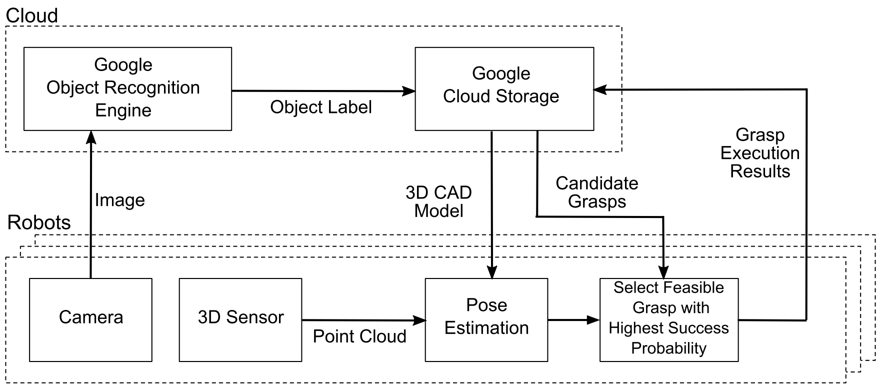


Fig. 24.12: System Architecture for cloud-based object recognition for grasping. The robot captures an image of an object and sends via the network to the Google object recognition server. The server processes the image and returns data for a set of candidate objects, each with pre-computed grasping options. The robot compares the returned CAD models with the detected point cloud to refine identification and to perform pose estimation, and selects an appropriate grasp. After the grasp is executed, data on the outcome is used to update models in the cloud for future reference [148].

Related work explores how computer vision can be used with Cloud resources to incrementally learn grasp strategies [149] [150] by matching sensor data against 3D CAD models in an online database. Examples of sensor data include 2D image features [151], 3D features [152], and 3D point clouds [153]. Google Goggles [154], a free network-based image recognition service for mobile devices, has been incorporated into a system for robot grasping [148] as illustrated in Fig 24.12.

Dalibard et al. attach "manuals" of manipulation tasks to objects [155]. The RoboEarth project stores data related to objects maps, and tasks, for applications ranging from object recognition to mobile navigation to grasping and manipulation (see Fig. 24.14(a)) [58].

As noted below, online datasets are effectively used to facilitate learning in computer vision. By leveraging Google's 3D warehouse, [156] reduces the need for manually labeled training data. Using community photo collections, [157] created an augmented reality application with processing in the cloud.

24.5.1.2 Cloud Computing for Robotics

As of 2018, Cloud Computing services like Amazon's EC2 elastic computing engine provide massively-parallel computation on demand [158]. Examples include Amazon Web Services [159] Elastic Compute Cloud, known as EC2 [160], Google Compute Engine [161], Microsoft Azure [162]. These provide a large pool of computing resources that can be rented by the public for short-term computing tasks. These services were originally used primarily by web application developers, but have increasingly been used in scientific and technical high performance computing (HPC) applications [163] [164] [165] [166].

Cloud computing is challenging when there are real-time constraints [167]; this is an active area of research. However there are many robotics applications that are not time sensitive such as decluttering a room or pre-computing grasp strategies.

There are many sources of uncertainty in robotics and automation [168]. Cloud computing allows massive sampling over error distributions and Monte Carlo sampling is “embarrassingly parallel”; recent research in fields as varied as medicine [169] and particle physics [170] have taken advantage of the cloud. Real-time video and image analysis can be performed in the Cloud [156] [171] [172]. Image processing in the cloud has been used for assistive technology for the visually impaired [173] and for senior citizens [174]. Cloud computing is ideal for sample-based statistical motion planning under uncertainty, where it can be used to explore many possible perturbations in object and environment pose, shape, and robot response to sensors and commands [175]. Cloud-based sampling is also being investigated for grasping objects with shape uncertainty [176] [177] (see Fig. 24.13). A grasp planning algorithm accepts as input a nominal polygonal outline with Gaussian uncertainty around each vertex and the center of mass to compute a grasp quality metric based on a lower bound on the probability of achieving force closure.

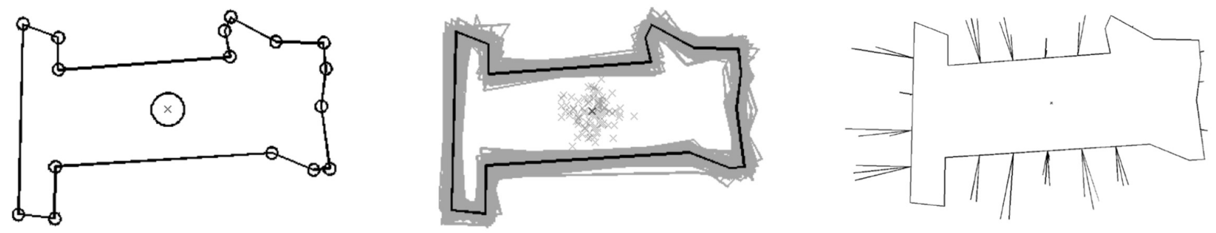


Fig. 24.13: A cloud-based approach to geometric shape uncertainty for grasping [176] [177].

24.5.1.3 Collective Robot Learning

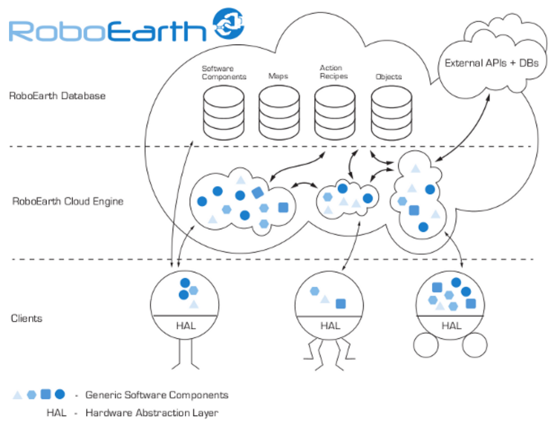
The Cloud allows robots and automation systems to “share” data from physical trials in a variety of environments, for example initial and desired conditions, associated control policies and trajectories, and importantly: data on performance and outcomes. Such data is a rich source for robot learning.

One example is for path planning, where previously-generated paths are adapted to similar environments [179] and grasp stability of finger contacts can be learned from previous grasps on an object [143].

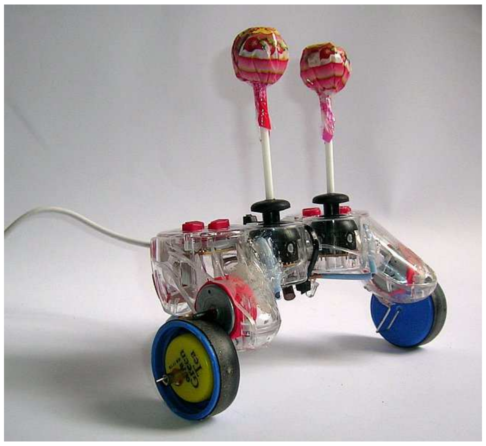
The MyRobots project [180] from RobotShop proposes a “social network” for robots: “In the same way humans benefit from socializing, collaborating and sharing, robots can benefit from those interactions too by sharing their sensor information giving insight on their perspective of their current state” [181].

24.5.1.4 Open-Source and Open-Access

The Cloud facilitates sharing by humans of designs for hardware, data, and code. The success of open-source software [182] [183] [184] is now widely accepted in the robotics and automation community. A primary example is ROS which is used by software developers to create robot applications [185]. ROS has also been ported to Android devices [186]. ROS has become a standard akin to Linux and is now used by almost all robot developers in research and many in industry.



(a)



(b)

Fig. 24.14: **(a)** RoboEarth architecture [58]. **(b)** Suckerbot, designed by Tom Tilley of Thailand, a winner of the $10 Robot Design Challenge [178].

Additionally, many simulation libraries for robotics are now open-source, which allows students and researchers to rapidly set up and adapt new systems and share the resulting software. Open-source simulation libraries include Bullet [187], a physics simulator originally used for video games, OpenRAVE [188] and Gazebo [189], simulation environments geared specifically towards robotics, OOPSMP, a motion-planning library [190], and GraspIt!, a grasping simulator [191].

Another exciting trend is in open-source hardware, where CAD models and the technical details of construction of devices are made freely available [192] [193]. The Arduino project [194] is a widely-used open-source microcontroller platform, and has been used in many robotics projects. The Raven [195] is an open-source laparoscopic surgery robot developed as a research platform an order of magnitude less expensive than commercial surgical robots [196].

The Cloud can also be used to facilitate open challenges and design competitions. For example, the African Robotics Network with support from IEEE Robotics and Automation Society hosted the “$10 Robot” Design Challenge in the summer of 2012. This open competition attracted 28 designs from around the world including a winning entry from Thailand (see Fig. 24.14b) that modified a surplus Sony game controller, adapting its embedded vibration motors to drive wheels and adding lollipops to the thumb switches as inertial counterweights for contact sensing, which can be built from surplus parts for US $8.96 [178].

24.5.1.5 Crowdsourcing and Call Centers

In contrast to automated telephone reservation and technical support systems, consider a future scenario where errors and exceptions are detected by robots and automation systems, which then access human guidance on-demand at remote call centers. Human skill, experience, and intuition are being tapped to solve a number of problems such as image labeling for computer vision [198] [149] [56] [49]. Amazon’s Mechanical Turk is pioneering on-demand “crowdsourcing” that can draw on “human computation” or “social computing systems”. Research projects are exploring how this can be used for path planning [199], to determine depth layers, image normals, and symmetry from

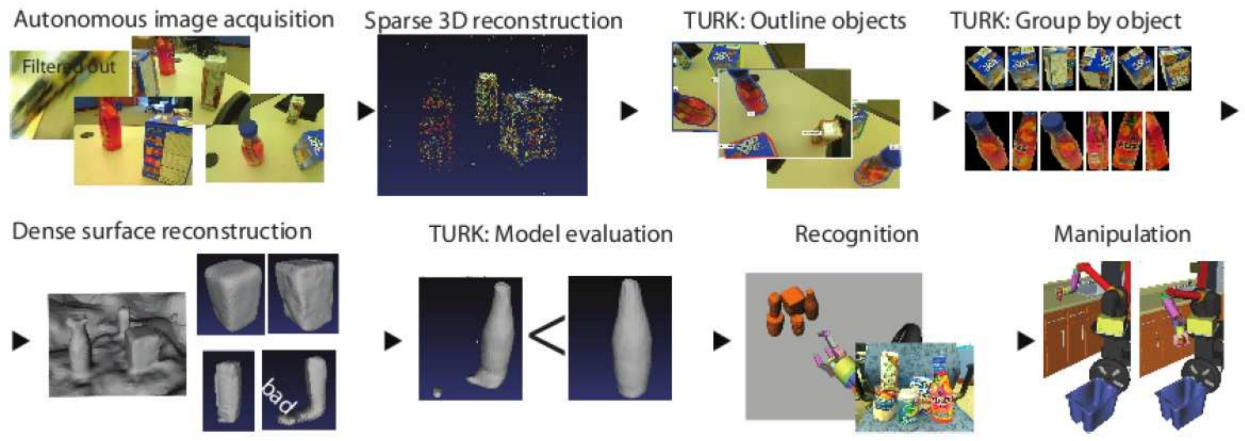


Fig. 24.15: A cloud robot system that incorporates Amazon's Mechanical Turk to "crowdsource" object identification to facilitate robot grasping [197].

images [200], and to refine image segmentation [201]. Researchers are working to understand pricing models [202] and apply crowdsourcing to grasping [197] (see Fig. 24.15).

24.5.2 Fog Robotics

'Fog Robotics' is an extension of Cloud Robotics that balances storage, compute and networking resources between the Cloud and the Edge in a federated manner [203, 204]. The robot uses Edge resources for performing tasks that need low latency and high bandwidth, and defers to Cloud for more intensive tasks. Note that the Fog complements the Cloud, not replace it. The concept of Fog Computing was introduced by Cisco Systems in 2012 [62] (see [63–65] for details). Other closely related concepts to Fog Computing are Cloudlets [205] and Mobile Edge Computing [206].

The rapid growth of service robots and IoT applications poses a challenge to the large-scale adoption of Cloud Robotics. One of the important factors is security of the data sent and received from heterogeneous sources over the Internet. The correctness and reliability of information has direct impact on the performance of robots. The robots collect sensitive information (e.g., images of home, proprietary warehouse and manufacturing data) that needs to be protected. Many robotics applications assume by default that they operate in a trusted, non-malicious environment – an assumption that is at odds with their widely distributed nature and multi-tenant usage. As an example, the widely used Robot Operating System (ROS) as a middleware is not designed for multicast subscribers and the variable latency along with the security concerns make it unsuitable for many real world applications [207] (ROS2 is an ongoing effort that is build upon DDS [208] with real-time capabilities and configurable QoS [209]). At the same time, the sheer volume of sensory data continues to increase, leading to a higher latency, variable timing, limited bandwidth access than deemed feasible for modern robotics applications. Moreover, stability issues arise in handling environmental uncertainty with any loss in network connectivity.

Fog Robotics addresses these issues by exploring a range of resources for compute and storage that are onboard a robot and throughout the network (from the the Edge of the network to the distant Cloud data centers). The Edge of the

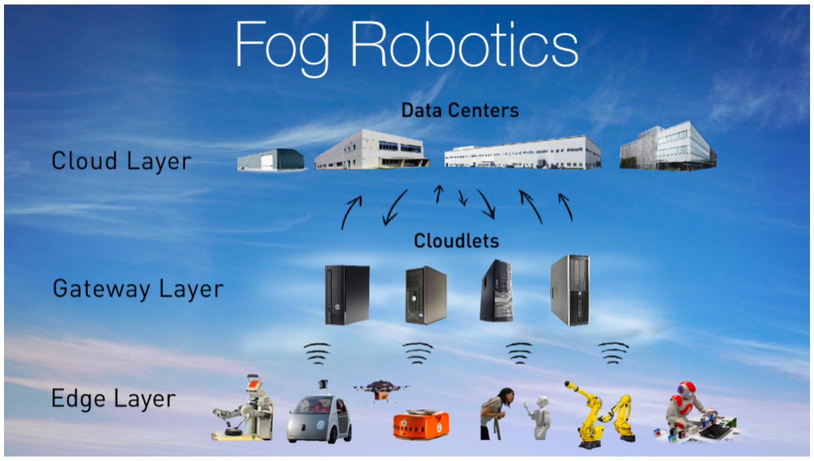


Fig. 24.16: A Fog Robotics architecture that uses the resources on the Cloud and the Edge of the network to meet lower latency requirements, while preserving the privacy and security of the data [204].

network consists of a large number of geo-distributed devices owned and operated by various administrative entities. As opposed to the practically infinite pool of homogeneous resources in the Cloud, resources at the Edge come in various sizes, e.g. light-weight micro servers, networking devices such as gateway routers, switches and access points. These devices communicate with the sensors in a Peer to Peer (P2P) manner or form a cluster depending upon the location or type of the device. The central idea behind Fog Robotics is to utilize the resources at the Edge which are in close proximity and save the time in communication with the Cloud. The goal is to manage the resources such that the Edge devices can process most of the data from the robot, without making the data traverse through untrusted domains while communicating with the Cloud. Fog Robotics enable [204]: 1) sharing of data and distributed learning with use of resources in close proximity instead of exclusively relying on Cloud resources, 2) security and privacy of data by restricting its access within a trusted infrastructure, and 3) resource allocation for load balancing between the Cloud and the Edge. As an example, a number of battery powered WiFi-enabled mobile robots can use resources from a close-by fixed infrastructure, such as a relatively powerful smart home gateway, while relying on far away Cloud resources for non-critical tasks. Similar deployments of robots with a fixed infrastructure can be envisioned for self-driving cars, flying drones, socially aware cobots and so on.

Fog robotics applications are well-suited for multi-agent connected environments where privacy and security are concerns, such as households, factories, connected vehicles, smart cities, and personalized devices. In [210], the authors present use cases of fog computing in urban surveillance, smart power grid, and drones surveillance. The authors in [211] use a multi-tier fog and cloud computing approach for pervasive brain monitoring system that can reliably estimate brain states and adapt to track users' brain dynamics. Aazam and Huh [212] presented a resource allocation model for fog computing. Bonomi et al. present provision for resource constrained IoT devices in their fog computing platform [213]. The authors in [214] present Mobile Fog to distribute IoT applications from the edge devices to the cloud in an hierarchical manner. The authors in [215] propose a framework to minimize service delays in fog applications by load sharing. Some groups recently advocated the need of a Fog robotics approach for industrial automation [216] and human robot collaboration [217]. Tanwani and Goldberg presented a Fog Robotics architecture and its application to simulation to reality transfer for surface decluttering [204]. Non-private (public) synthetic images of cluttered floors are used for training deep models on the Cloud, and private real images are used to adapt the deep models by learning invariant feature representations with an adversarial discriminator at the Edge. The trained models are deployed as a service for inference with a fleet of robots. Deploying the service on the Edge significantly reduces the inference time, while maintaining the privacy of data by restricting control within a trusted domain.

Recently, Amazon released the Cloud Robotics platform RoboMaker to develop and test robotic applications in simulation by extending ROS connectivity with their Cloud services [218]. Google has also announced the release of a new Cloud Robotics platform to provide Cloud-connected services to the robots [219]. Designing custom, secure and adaptive algorithms to manage compute and storage from Cloud to Edge is an exciting and open area of research.

24.6 Conclusion and Future Directions

Networked-, Cloud-, and Fog-Robotics are a vibrant area in both application and research. Networks connect autonomous agents, robots, humans, and their residing environments. Four waves of network technologies, namely the Internet, WWW, mobile computing, and cloud computing, have significantly pushed forward the area of the networked robots in past decades. At present, artificial intelligence including computer vision, voice recognition, and machine learning provide new techniques to make robots smart and easy to interact with their human users. Looking forward, Fast 5G communication and AR/VR can provide tremendous growth space for this area. Embracing the new technology wave, many new research challenges remain.

- *New algorithms*: Scalable algorithms that are capable of handing large amounts of data such as video/sensor network inputs and utilize fast-evolving hardware capability such as distributed and parallel computation will become increasingly important in the networked robotics, especially in Cloud and Fog robotics.
- *New performance metrics*: As more and more robots enter service, it is important to develop metrics to quantify the performance of the robot-human team. As we are more familiar with metrics developed to assess robot performance or task performance [220], recent progresses on using the robot to assess human performance [221, 222] shed light on new metrics. Standardizing these metrics will also be an important direction.
- *New Applications*: Recent successful applications include environment monitoring [41, 223], agriculture and manufacturing [224, 225], and infrastructure inspection and maintenance [226, 227]. The fast development of networked robot systems is worldwide. Many new applications are emerging in areas such as security, inspection, education, and entertainment. Application requirements such as reliability, security, and modularity will continuous to pose new challenges for system design.

Exercise 1. We have discussed teleoperation control spectrum in Section 24.2.1. For the cases below, please discuss which type of teleoperation control would be the best for each case. In each type, we tell you its scenario, tasks, along with communication types. Please take cost into consideration.

- **Case 1:** There is a bus with its driver who has lost consciousness. The bus is fully equipped with a global positioning system (GPS), a front facing camera, a full 360 degrees field of view radar, a drive-by-wire system, and a satellite link that connects the bus to a control center. The urban street is very busy with other vehicles, pedestrians, and bicycles. The visibility is okay but weather may deteriorate soon. The satellite link has some latency (about 200ms delay) in communication. The control center want to teleoperates this bus to safety. What is the best teleoperation control type here?
- **Case 2:** There is a mobile rover on the Mars transporting equipment between a landing zone and a mining zone. It is connected to a ground station on the Earth through an inter-planet radio link. The one-way time delay in communication is about 25 minutes due to the relative position between the Mars and the Earth. How should the ground station teleoperate the rover?
- **Case 3:** A surgeon wants to perform an open heart surgery for a patient with cardiovascular issues. The surgery needs to be performed when the patient's heart is still beating. A 6-DOF surgery robot system is available and it is capable of performing high speed visual servoing. The robot can synchronize the motion of the cutting knife/camera feed with the heart beating motion so that the heart appears to be stationary in the camera view. The stabilized images are sent to the doctor in the adjacent room via a high speed 1Gbs fiber optic link. The doctor sends commands to the robot using a joystick in real time. The robot combines commands from the doctor and its sensory data to plan for and execute knife motion. What is the best teleoperation control type here?
- **Case 4:** Tom's grandparents live by themselves and have a robot vacuum in their household. It is connected to WiFi and can be teleoperated via mobile apps. Tom wants to teleoperate this robot to reach his grandparents to check their status since they have not answered Tom's phone call. Tom wants to use the robot's onboard camera, microphone, and speaker to dialog with them. Tom wants to teleoperate the robot to search for his grandparent first. Tom has a home WiFi with broadband connection to this robot (about 25Mbs). What is the best teleoperation control type here?

Exercise 2. What are the mathematical state-command-environment representations for each case in Exercise 1? At time t, robot states are $\mathbf{x}(t)$, available commands are $\mathbf{u}(t)$, and environment information (e.g. L and B for mobile robots) may be specific to tasks. Please explain these variables in each dimension in the context of each case.

Exercise 3. For Cases 1-4 in Exercise 1, how would you visualize/display states for the human operators? Please note that states are different across those cases.

Exercise 4. For Case 2 in Exercise 1, there are many identical rovers on the Mars. Also, there are many operators on the Earth. Assume there are more rovers than operators. All rovers are identical and perform different transportation service at the same time. However, rovers may run into issues from time to time and need human assistance. For example, it may get stuck. How would you design an MOMR architecture to improve task efficiency? More specically, what is your strategy in dynamically allocating rover-human pairs?

Exercise 5. For Case 4 in Exercise 1, Tom has many cousins who also want use the same system to visit the grandparents. It is possible that more than one person may want to teleoperate the robot at the same time. How would you resolve the conflict by designing an SOMR system?

References

1. Thomas B. Sheridan. *Telerobotics, Automation, and Human Supervisory Control*. MIT Press, 1992.
2. N. Tesla. Method of and apparatus for controlling mechanism of moving vessels or vehicles. *http://www.pbs.org/tesla/res/613809.html*, 1898.
3. Raymond Goertz and R. Thompson. Electronically controlled manipulator. *Nucleonics*, 1954.
4. Annica Kristoffersson, Silvia Coradeschi, and Amy Loutfi. A review of mobile robotic telepresence. *Adv. in Hum.-Comp. Int.*, 2013:3:3–3:3, January 2013.
5. Iwaki Toshima, Shigeaki Aoki, and Tatsuya Hirahara. Sound localization using an acoustical telepresence robot: Telehead ii. *Presence: Teleoperators and Virtual Environments*, 17(4):392–404, 2008.
6. Veerle A.I. Huvenne, Katleen Robert, Leigh Marsh, Claudio Lo Iacono, Tim Le Bas, and Russell B. Wynn. *ROVs and AUVs*, pages 93–108. Springer International Publishing, Cham, 2018.
7. Paul Voosen. Mars rover steps up hunt for molecular signs of life. *Science*, 355(6324):444–445, 2017.
8. Huu Minh Le, Thanh Nho Do, and Soo Jay Phee. A survey on actuators-driven surgical robots. *Sensors and Actuators A: Physical*, 247:323 – 354, 2016.
9. K. Goldberg and R. Siegwart. *Beyond Webcams: An Introduction to Online Robots*. MIT Press, 2002.
10. R. S. Mosher. Industrial manipulators. *Scientific American*, 211(4), 1964.
11. R. Tomovic. On man-machine control. *Automatica*, 5, 1969.
12. A. Bejczy, G. Bekey, R. Taylor, and S. Rovetta. A research methodology for tele-surgery with time delays. In *First International Symposium on Medical Robotics and Computer Assisted Surgery*, Sept. 1994.
13. Matthew Gertz, David Stewart, and Pradeep Khosla. A human-machine interface for distributed virtual laboratories. *IEEE Robotics and Automation Magazine*, December 1994.
14. T. Sato, J. Ichikawa, M. Mitsuishi, and Y. Hatamura. A new micro-teleoperation system employing a hand-held force feedback pencil. In *IEEE International Conference on Robotics and Automation*, May 1994.
15. Scott B. Nokleby. Singularity analysis of the canadarm2. *Mechanism and Machine Theory*, 42(4):442 – 454, 2007.
16. Brain Davies. Robotic surgery – a personal view of the past, present and future. *International Journal of Advanced Robotic Systems*, 12(54):1 – 6, 2015.
17. A. K. Bejczy. Sensors, controls, and man-machine interface for advanced teleoperation. *Science*, 208(4450), 1980.
18. R. D. Ballard. A last long look at titanic. *National Geographic*, 170(6), December 1986.
19. J. Yuh, G. Marani, and D. Blidberg. Applications of marine robotic vehicles. *Intelligent Service Robotics*, 4:221 – 231, 2011.
20. Konstantinos Kanistras, Goncalo Martins, Matthew J. Rutherford, and Kimon P. Valavanis. *Survey of Unmanned Aerial Vehicles (UAVs) for Traffic Monitoring*, pages 2643–2666. Springer Netherlands, Dordrecht, 2015.
21. Katherine M. Tsui, Munjal Desai, Holly A. Yanco, and Chris Uhlik. Exploring use cases for telepresence robots. In *Proceedings of the 6th International Conference on Human-robot Interaction*, HRI '11, pages 11–18, New York, NY, USA, 2011. ACM.
22. N. P. Jouppi and S. Thomas. Telepresence systems with automatic preservation of user head height, local rotation, and remote translation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 62–68, 2005.
23. Eric Paulos, John Canny, and Francesca Barrientos. Prop: Personal roving presence. In *SIGGRAPH Visual Proceedings*, page 99, August 1997.
24. L. Takayama, E. Marder-Eppstein, H. Harris, and J. Beer. Assisted driving of a mobile remote presence system: System design and controlled user evaluation. In *IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China*, pages 1883 –1889, may 2011.
25. D. Lazewatsky and W. Smart. An inexpensive robot platform for teleoperation and experimentation. In *IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China*, pages 1211 –1216, may 2011.
26. D. Aarno, S. Ekvall, and D. Kragi. Adaptive virtual fixtures for machine-assisted teleoperation tasks. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1151–1156, 2005.
27. I. Belousov, S. Chebukov, and V. Sazonov. Web-based teleoperation of the robot interacting with fast moving objects. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 685–690, 2005.
28. Z. Cen, A. Goradia, M. Mutka, N. Xi, W. Fung, and Y. Liu. Improving the operation efficiency of supermedia enhanced internet based teleoperation via an overlay network. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 691–696, 2005.
29. B. Ricks, C. W. Nielsen, and M. A. Goodrich. Ecological displays for robot interaction: a new perspective. In *International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 2855 – 2860, 2004.
30. D. Ryu, S. Kang, M. Kim, and J. Song. Multi-modal user interface for teleoperation of robhaz-dt2 field robot system. In *International Conference on Intelligent Robots and Systems (IROS)*, volume 1, pages 168–173, 2004.

31. J. Su and Z. Luo. Incremental motion compression for telepresent walking subject to spatial constraints. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 69–74, 2005.
32. I. Toshima and S. Aoki. Effect of driving delay with an acoustical tele-presence robot, telehead. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 56–61, 2005.
33. N. Chong, T. Kotoku, K. Ohba, K. Komoriya, N. Matsuhira, and K. Tanie. Remote coordinated controls in multiple telerobot cooperation. In *IEEE International Conference on Robotics and Automation*, volume 4, pages 3138–3343, April 2000.
34. P. Cheng and V. Kumar. An almost communication-less approach to task allocation for multiple unmanned aerial vehicles. In *IEEE International Conference on Robotics and Automation (ICRA), Pasadena, California*, pages 1384 –1389, may 2008.
35. X. Ding, M. Powers, M. Egerstedt, S. Young, and T. Balch. Executive decision support. *IEEE Robotics Automation Magazine*, 16(2):73 –81, june 2009.
36. J. Liu, L. Sun, T. Chen, X. Huang, and C. Zhao. Competitive multi-robot teleoperation. In *IEEE International Conference on Robotics and Automation Barcelona, Spain*, April 2005.
37. Z. Zhang, Q. Cao, L. Zhang, and C. Lo. A corba-based cooperative mobile robot system. *Industrial Robot: An International Journal*, 36(1):36 – 44, 2009.
38. Y. Xu and D. Song. Systems and algorithms for autonomous and scalable crowd surveillance using robotic ptz cameras assisted by a wide-angle camera. *Autonomous Robots*, 29(1):53–66, July 2010.
39. D. Sanders, J. Graham-Jones, and A. Gegov. Improving ability of tele-operators to complete progressively more difficult mobile robot paths using simple expert systems and ultrasonic sensors. *Industrial Robot: An International Journal*, 37(5):431 – 440, 2010.
40. S. Faridani, B. Lee, S. Glasscock, J. Rappole, D. Song, and K. Goldberg. A networked telerobotic observatory for collaborative remote observation of avian activity and range change. In *the IFAC workshop on networked robots, Oct. 6-8, 2009, Golden, Colorado*, Oct. 2009.
41. R. Bogue. Robots for monitoring the environment. *Industrial Robot: An International Journal*, 38(6):560 – 566, 2011.
42. R. Murphy and J. Burke. From remote tool to shared roles. *IEEE Robotics Automation Magazine*, 15(4):39 –49, Dec. 2008.
43. K. Taylor and J.P. Trevelyan. Australiaś telerobot on the web. In *26th Symposium on Industrial Robotics, Singapore*, pages 39–44, Oct. 1995.
44. A. Khamis, D. M. Rivero, F. Rodriguez, and M. Salichs. Pattern-based architecture for building mobile robotics remote laboratories. In *IEEE Internetional Conference on Robotics and Automation (ICRA), Taipei, ROC*, pages 3284–3289, Sep. 2003.
45. C. Cosma, M. Confente, D. Botturi, and P. Fiorini. Laboratory tools for robotics and automation education. In *IEEE Internetional Conference on Robotics and Automation (ICRA), Taipei, ROC*, pages 3303–3308, Sep. 2003.
46. K. W. Dorman, J. L. Pullen, W. O. Keksz, P. H. Eismann, K. A. Kowalski, and J. P. Karlen. The servicing aid tool: A teleoperated robotics system for space applications. In *The Seventh Annual Workshop on Space Operations Applications and Research (SOAR 1993), Johnson Space Center, Houston, TX*, volume 1, Jan. 1994.
47. C. Pollak and H. Hutter. A webcam as recording device for light microscopes. *Journal of Computer-Assisted Microscopy*, 10(4):179–83, 1998.
48. K. Goldberg, D. Song, and A. Levandowski. Collaborative teleoperation using networked spatial dynamic voting. *The Proceedings of The IEEE*, 91(3):430–439, March 2003.
49. James J. Kuffner. Cloud-Enabled Robots. In *IEEE-RAS International Conference on Humanoid Robots*, Nashville, TN, 2010.
50. Ken Goldberg and Roland Siegwart, editors. *Beyond Webcams: An Introduction to Online Robots*. MIT Press, 2002.
51. Ken Goldberg, Michael Mascha, Steve Gentner, Nick Rothenberg, Carl Sutter, and Jeff Wiegley. Desktop teleoperation via the World Wide Web. In *IEEE International Conference on Robotics and Automation*, volume 1, pages 654–659. IEEE, 1995.
52. G. McKee. What is Networked Robotics? *Informatics in Control Automation and Robotics*, 15:35–45, 2008.
53. Rajesh Arumugam, V.R. Enti, Liu Bingbing, Wu Xiaojun, Krishnamoorthy Baskaran, F.F. Kong, A.S. Kumar, K.D. Meng, and G.W. Kit. DAvinCi: A Cloud Computing Framework for Service Robots. In *IEEE International Conference on Robotics and Automation*, pages 3084–3089. IEEE, 2010.
54. Zhihui Du, Weiqiang Yang, Yinong Chen, Xin Sun, Xiaoying Wang, and Chen Xu. Design of a Robot Cloud Center. In *International Symposium on Autonomous Decentralized Systems*, pages 269–275. IEEE, March 2011.
55. Guoqiang Hu, WP Tay, and Yonggang Wen. Cloud Robotics: Architecture, Challenges and Applications. *IEEE Network*, 26(3):21–28, 2012.
56. Koji Kamei, Shuichi Nishio, Norihiro Hagita, and Miki Sato. Cloud Networked Robotics. *IEEE Network*, 26(3):28–34, 2012.
57. Dominique Hunziker, Mohanarajah Gajamohan, Markus Waibel, and Raffaello D Andrea. Rapyuta: The RoboEarth Cloud Engine. 2013.
58. Markus Waibel, Michael Beetz, Javier Civera, Raffaello D'Andrea, Jos Elfring, Dorian Gálvez-López, Kai Häussermann, Rob Janssen, J.M.M. Montiel, Alexander Perzylo, Björn Schieß le, Moritz Tenorth, Oliver Zweigle, and René De Molengraft. RoboEarth. *IEEE Robotics & Automation Magazine*, 18(2):69–82, June 2011.

59. What is RoboEarth? `http://www.roboearth.org/what-is-roboearth`.
60. Luis M. Vaquero and Luis Rodero-Merino. Finding your way in the fog: Towards a comprehensive definition of fog computing. *SIGCOMM Comput. Commun. Rev.*, 44(5):27–32, October 2014.
61. W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5):637–646, 2016.
62. Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16. ACM, 2012.
63. Shanhe Yi, Cheng Li, and Qun Li. A survey of fog computing: Concepts, applications and issues. In *Proceedings of the 2015 Workshop on Mobile Big Data*, Mobidata '15, pages 37–42. ACM, 2015.
64. A.V. Dastjerdi, H. Gupta, R.N. Calheiros, S.K. aGhosh, and R. Buyya. Chapter 4 - fog computing: principles, architectures, and applications. In Rajkumar Buyya and Amir Vahid Dastjerdi, editors, *Internet of Things*, pages 61 – 75. Morgan Kaufmann, 2016.
65. Slavica Tomovic, Kenji Yoshigoe, Ivo Maljevic, and Igor Radusinovic. Software-defined fog network architecture for iot. *Wireless Personal Communications*, 92(1):181–196, 2017.
66. J. Walrand and P. Varaiya. *High-Performance Communication Networks, 2nd Edition*. Morgan Kaufmann Press, 2000.
67. Robert H. Thomas. A resource sharing executive for the arpanet. In *Proceedings of the National Computer Conference and Exposition*, AFIPS, pages 155–163. ACM, 1973.
68. FirstWebcam. `http://www.cl.cam.ac.uk/coffee/qsf/timeline.html`. 1993.
69. K. Goldberg, M. Mascha, S. Gentner, N. Rothenberg, C. Sutter, and J. Wiegley. Robot teleoperation via www. In *IEEE International Conference on Robotics and Automation*, May 1995.
70. K. Goldberg, M. Mascha, S. Gentner, N. Rothenberg, C. Sutter, and Jeff Wiegley. Beyond the web: Manipulating the physical world via the www. *Computer Networks and ISDN Systems Journal*, 28(1), December 1995. Archives can be viewed at http://www.usc.edu/dept/raiders/.
71. B. Dalton and K. Taylor. A framework for internet robotics. In *IEEE International Conference On Intelligent Robots and Systems (IROS): Workshop on Web Robots, Victoria, Canada*, 1998.
72. K. Taylor and J. Trevelyan. `http://telerobot.mech.uwa.edu.au/`. 1994.
73. H. Hu, L. Yu, P. W. Tsui, and Q. Zhou. Internet-based robotic systems for teleoperation. *Assemby Automation*, 21(2):143–151, May 2001.
74. R. Safaric, M. Debevc, R. Parkin, and S. Uran. Telerobotics experiments via internet. *IEEE Transactions on Industrial Electronics*, 48(2):424–31, April 2001.
75. S. Jia and K. Takase. A corba-based internet robotic system. *Advanced Robotics*, 15(6):663–673, Oct 2001.
76. S. Jia, Y. Hada, G. Ye, and K. Takase. Distributed telecare robotic systems using corba as a communication architecture. In *IEEE International Conference on Robotics and Automation (ICRA), Washington, DC, United States*, 2002.
77. J. Kim, B. Choi, S. Park, K.Kim, and S. Ko. Remote control system using real-time mpeg-4 streaming technology for mobile robot. In *IEEE International Conference on Consumer Electronics*, 2002.
78. T. Mirfakhrai and S. Payandeh. A delay prediction approach for teleoperation over the internet. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2002.
79. K. Han, Y. Kim, J. Kim, and S.Hsia. Internet control of personal robot between kaist and uc davis. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2002.
80. L. Ngai, W.S. Newman, and V. Liberatore. An experiment in internet-based, human-assisted robotics. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2002.
81. R.C. Luo and T. M. Chen. Development of a multibehavior-based mobile robot for remote supervisory control through the internet. *IEEE/ASME Transactions on Mechatronics*, 5(4):376–385, 2000.
82. Morgan Quigley and Brian Gerkey. ROS: an open-source Robot Operating System. In *ICRA Workshop on Open Source Software*, number Figure 1, 2009.
83. Alar Kuusik, Takeo Hisada, Satoshi Suzuki, and Katsuhisa Furuta. Cellular network telecontrolled robot vehicle. *IFAC Proceedings Volumes*, 37(7):99 – 104, 2004. 1st IFAC Symposium on Telematics Applications in Automation and Robotics (TA 2004), Espoo, Finland, 21-23 June 2004.
84. Karlin Bark, William McMahan, Austin Remington, Jamie Gewirtz, Alexei Wedmid, David I. Lee, and Katherine J. Kuchenbecker. In vivo validation of a system for haptic feedback of tool vibrations in robotic surgery. *Surgical Endoscopy*, 27(2):656–664, Feb 2013.
85. R. Marin, P. J. Sanz, and J. S. Sanchez. A very high level interface to teleoperate a robot via web including augmented reality. In *IEEE International Conference on Robotics and Automation (ICRA), Washington DC, May*, volume 3, pages 2725–2730, 2002.
86. U. Rau, S. Bhatnagar, M. A. Voronkov, and T. J. Cornwell. Advances in calibration and imaging techniques in radio interferometry. *Proceedings of the IEEE*, 97(8):1472–1481, Aug 2009.

87. M. Seyedi, B. Kibret, D. T. H. Lai, and M. Faulkner. A survey on intrabody communications for body area network applications. *IEEE Transactions on Biomedical Engineering*, 60(8):2067–2079, Aug 2013.
88. R. C. Luo, T. Y. Hsu, T. Y. Lin, and K. L. Su. The development of intelligent home security robot. In *IEEE International Conference on Mechatronics, 2005. ICM '05.*, pages 422–427, July 2005.
89. V. Kumar, D. Rus, and S. Singh. Robot and sensor networks for first responders. *IEEE Pervasive Computing*, pages 24–33, October-December 2004.
90. George Kantor, Sanjiv Singh, Ronald Peterson, Daniela Rus, Aveek Das, Vijay Kumar, Guilherme Pereira, and John Spletzer. *Distributed Search and Rescue with Robot and Sensor Teams*, pages 529–538. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.
91. A. Cano, E. Lopez-Baeza, J. L. Anon, C. Reig, and C. Millan-Scheding. Wireless sensor network for soil moisture applications. In *2007 International Conference on Sensor Technologies and Applications (SENSORCOMM 2007)*, pages 508–513, Oct 2007.
92. Ian F. Akyildiz, Dario Pompili, and Tommaso Melodia. Underwater acoustic sensor networks: research challenges. *Ad Hoc Networks*, 3(3):257 – 279, 2005.
93. G. T. Sibley, M. H. Rahimi, and G. S. Sukhatme. Robomote: a tiny mobile robot platform for large-scale ad-hoc sensor networks. In *IEEE International Conference on Robotics and Automation (ICRA), Washington DC*, volume 2, pages 1143–1148, 2002.
94. S. Kamath, E. Meisner, and V. Isler. Triangulation based multi target tracking with mobile sensor networks. In *IEEE International Conference on obotics and Automation (ICRA), Roma, Italy*, April 2007.
95. M. Batalin and G. Sukhatme. Coverage, exploration and deployment by a mobile robot and communication network. *Telecommunication Systems*, 26(2-4):181–96, Oct. 2004.
96. O. Tekdas, V. Isler, J. H. Lim, and A. Terzis. Using mobile robots to harvest data from sensor fields. *IEEE Wireless Communications*, 16(1):22–28, February 2009.
97. Maxim A. Batalin, Gaurav S. Sukhatme, and Myron Hattig. Mobile robot navigation using a sensor network. In *IEEE International Conference on Robotics and Automation (ICRA), New Orleans, LA,*, pages 636 – 642, Ari-May 2004.
98. N. Patwari, J. N. Ash, S. Kyperountas, A. O. Hero, R. L. Moses, and N. S. Correal. Locating the nodes: cooperative localization in wireless sensor networks. *IEEE Signal Processing Magazine*, 22(4):54–69, July 2005.
99. Jesus Capitán Fernández, Jose Ramiro Martinez de Dios, Ivan Maza, Fabresse Felipe Ramon, and Anibal Ollero. Ten years of cooperation between mobile robots and sensor networks. *International Journal of Advanced Robotic Systems*, 12(6):70, 2015.
100. J. Craig. *Introduction to Robotics: Mechanics and Control*. Pearson, 4th edition, 2017.
101. Michael A. Peshkin and Art C. Sanderson. Minimization of energy in quasi-static manipulation. *IEEE Transactions on Robotics and Automation*, 5(1), February 1989.
102. Matthew T. Mason. On the scope of quasi-static pushing. In O. Faugeras and G. Giralt, editors, *The Third International Symposium on Robotics Research*. MIT Press, 1986.
103. Eric Ladd and Jim O'Donnell. *Using Html 4, Xml, and Java 1.2*. QUE Press, 1998.
104. K. Goldberg, M. Mascha, S. Gentner, N. Rothenberg, C. Sutter, and J. Wiegley. Desktop tele-operation via the world wide web. In *IEEE International Conference on Robotics and Automation, Nagoya, Japan.*, May 1995.
105. H. Friz. *Design of an Augmented Reality User Interface for an Internet based Telerobot using Multiple Monoscopic Views*. PhD thesis, Institute for Process and Production Control Techniques, Technical University of Clausthal Clausthal-Zellerfeld, Germany, 2000.
106. A .Birk, N. Vaskevicius, K. Pathak, S. Schwertfeger, J. Poppinga, and H. Buelow. 3-d perception and modeling. *IEEE Robotics Automation Magazine*, 16(4):53 –60, Dec. 2009.
107. A. Kellyo, N. Chan, H. Herman, D. Huber, R. Meyers, P. Rander, R. Warner, J. Ziglar, and E. Capstick. Real-time photorealistic virtualized reality interface for remote mobile robot control. *The International Journal of Robotics Research*, 30(3):384–404, 2011.
108. T. Fong and C. Thorpe. Vehicle teleoperation interfaces. *Autonomous Robots*, 11:9–18, 2001.
109. F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard. An evaluation of the rgb-d slam system. In *2012 IEEE International Conference on Robotics and Automation*, pages 1691–1696, May 2012.
110. T. Lemaire and S. Lacroix. Monocular-vision based slam using line segments. In *IEEE International Conference of Robotics and Automation*, pages 2791–2796, Roma, Italy, April 2007.
111. H. Li, D. Song, Y. Lu, and J. Liu. A two-view based multilayer feature graph for robot navigation. In *IEEE International Conference on Robotics and Automation (ICRA), St. Paul, Minnesota*, May 2012.
112. Y. Lu and D. Song. Visual navigation using heterogeneous landmarks and unsupervised geometric constraints. In *IEEE Transactions on Robotics (T-RO)*, volume 31, pages 736 —- 749, June 2015.
113. Y. Lu, J. Lee, S. Yeh, H. Cheng, B. Chen, and D. Song. Sharing heterogeneous spatial knowledge: Map fusion between asynchronous monocular vision and lidar or other prior inputs. In *International Symposium on Robotics Research (ISRR), Puerto Varas, Chile*, Dec. 2017.

114. C. Chou, A. Kingery, D. Wang, H. Li, and D. Song. Encoder-camera-ground penetrating radar tri-sensor mapping for surface and subsurface transportation infrastructure inspection. In *IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia*, May 2018.
115. Louis B. Rosenberg. Virtual fixtures: Perceptual tools for telerobotic manipulation. In *IEEE Virtual Reality Annual International Symposium (VRAIS), Seattle, Washington*, pages 76–82, Sep. 1993.
116. Jake J. Abbott, Panadda Marayong, and Allison M. Okamura. *Haptic Virtual Fixtures for Robot-Assisted Manipulation*, pages 49–64. 2007.
117. A. K. Tanwani and S. Calinon. A generative model for intention recognition and manipulation assistance in teleoperation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pages 43–50, 2017.
118. L. Conway, R. A. Volz, and M. W. Walker. Teleautonomous systems: Projecting and coordinating intelligent action at a distance. *IEEE Transactions on Robotics and Automation*, 6(20):146–158, Appl. 1990.
119. J. Larsson, M. Broxvall, and A. Saffiotti. An evaluation of local autonomy applied to teleoperated vehicles in underground mines. In *IEEE International Conference on Robotics and Automation (ICRA), Anchorage, Alaska*, pages 1745 –1752, may 2010.
120. A. K. Tanwani and S. Calinon. Learning robot manipulation tasks with task-parameterized semitied hidden semi-markov model. *Robotics and Automation Letters, IEEE*, 1(1):235–242, 2016.
121. P. Marayong, M. Li, A. Okamura, and G. Hager. Spatial motion constraints: theory and demonstrations for robot guidance using virtual fixtures. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1954 – 1959, sept. 2003.
122. A. Bettini, P. Marayong, S. Lang, A. Okamura, and G. Hager. Vision-assisted control for manipulation using virtual fixtures. *IEEE Transactions on Robotics*, 20(6):953 – 966, dec. 2004.
123. R. Azuma. A survey of augmented reality. *Presence*, 6(4):355 – 385, Aug. 1997.
124. Thomas B. Sheridan. *Telerobotics, Automation, and Human Supervisory Control*. MIT Press, Cambridge, MA, USA, 1992.
125. D. Yoerger and J. J. Slotine. Supervisory control architecture for underwater teleoperation. In *IEEE International Conference on Robotics and Automation.*, volume 4, pages 2068–2073, Mar 1987.
126. D. Song, A. Pashkevich, and K. Goldberg. Sharecam part II: Approximate and distributed algorithms for a collaboratively controlled robotic webcam. In *IEEE/RSJ International Conference on Intelligent Robots (IROS), Las Vegas, NV*, volume 2, pages 1087 – 1093, Oct. 2003.
127. K. Goldberg, B. Chen, R. Solomon, S. Bui, B. Farzin, J. Heitler, D. Poon, and G. Smith. Collaborative teleoperation via the internet. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 2019–2024, April 2000.
128. D. Song. *Systems and Algorithms for Collaborative Teleoperation*. PhD thesis, Department of Industrial Engineering and Operations Research, University of California, Berkeley, 2004.
129. D. Song. *Sharing a Vision: Systems and Algorithms for Collaboratively-Teleoperated Robotic Cameras*. Springer, 2009.
130. K. Goldberg and B. Chen. Collaborative teleoperation via the internet. In *International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2001.
131. D. Song and K. Goldberg. Sharecam part I: Interface, system architecture, and implementation of a collaboratively controlled robotic webcam. In *IEEE/RSJ International Conference on Intelligent Robots (IROS), Las Vegas, NV*, volume 2, pages 1080 – 1086, Oct. 2003.
132. D. Song and K. Goldberg. Approximate algorithms for a collaboratively controlled robotic camera. *IEEE Transactions on Robotics*, 23(5):1061–1070, Nov. 2007.
133. D. Song, A. F. van der Stappen, and K. Goldberg. Exact algorithms for single frame selection on multi-axis satellites. *IEEE Transactions on Automation Science and Engineering*, 3(1):16–28, January 2006.
134. Y. Xu, D. Song, and J. Yi. An approximation algorithm for the least overlapping p-frame problem with non-partial coverage for networked robotic cameras. In *IEEE International Conference on Robotics and Automation (ICRA), Pasadena, CA*, May 2008.
135. D. Song, N. Qin, and K. Goldberg. Systems, control models, and codec for collaborative observation of remote environments with an autonomous networked robotic camera. *Autonomous Robots*, 24(4):435–449, May. 2008.
136. K. Goldberg, D. Song, I. Y. Song, J. McGonigal, W. Zheng, and D. Plautz. Unsupervised scoring for scalable internet-based collaborative teleoperation. In *IEEE International Conference on Robotics and Automation (ICRA), New Orleans, LA, US*, April 2004.
137. J. Rappole, S. Glasscock, K. Goldberg, D. Song, and S. Faridani. Range change among new world tropical and subtropical birds. In K.-L. Schuchmann, editor, *Tropical vertebrates in a changing world*, Bonner Zoologische Monographien, Nr 57, pages 151–167. Bonn: Zoologisches Forschungsmuseum Alexander Koenig, Germany, 2011.
138. B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg. A survey of research on cloud robotics and automation. *Automation Science and Engineering, IEEE Transactions on*, 12(2):398–409, April 2015.
139. Cloud Robotics. `http://goldberg.berkeley.edu/cloud-robotics/`. Accessed: 2018-11-29.
140. C. Goldfeder, M. Ciocarlie, and P.K. Allen. The Columbia Grasp Database. In *IEEE International Conference on Robotics and Automation*, pages 1710–1716. IEEE, May 2009.

141. A. Kasper, Z. Xue, and R. Dillmann. The KIT object models database: An object model database for object recognition, localization and manipulation in service robotics. *The International Journal of Robotics Research*, 31(8):927–934, May 2012.
142. Hao Dang, Jonathan Weisz, and Peter K. Allen. Blind grasping: Stable robotic grasping using tactile feedback and hand kinematics. In *IEEE International Conference on Robotics and Automation*, pages 5917–5922. Ieee, May 2011.
143. Hao Dang and Peter K. Allen. Learning grasp stability. In *IEEE International Conference on Robotics and Automation*, pages 2392–2397. IEEE, May 2012.
144. Jonathan Weisz and Peter K. Allen. Pose error robust grasping from contact wrench space metrics. In *IEEE International Conference on Robotics and Automation*, pages 557–562. IEEE, May 2012.
145. Mila Popovic, Gert Kootstra, Jimmy Alison Jorgensen, Danica Kragic, and Norbert Kruger. Grasping unknown objects using an Early Cognitive Vision system for general scene understanding. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 987–994. IEEE, September 2011.
146. N. Tian, M. Matl, J. Mahler, Y. X. Zhou, S. Staszak, C. Correa, S. Zheng, Q. Li, R. Zhang, and K. Goldberg. A cloud robot system using the dexterity network and berkeley robotics and automation as a service (brass). In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1615–1622, 2017.
147. Li Pusong, B. DeRose, J. Mahler, J. A. Ojea, A. K. Tanwani, and K. Golberg. Dex-net as a service (dnaas): A cloud-based robust robot grasp planning system. In *14th International Conference on Automation Science and Engineering (CASE)*, pages 1–8, 2018.
148. B. Kehoe, A. Matsukawa, S. Candido, J. Kuffner, and K. Goldberg. Cloud-Based Robot Grasping with the Google Object Recognition Engine. 2013. Submitted to IEEE International Conference on Robotics and Automation, 2013.
149. Matei Ciocarlie, Caroline Pantofaru, Kaijen Hsiao, Gary Bradski, Peter Brook, and Ethan Dreyfuss. A Side of Data With My Robot. *IEEE Robotics & Automation Magazine*, 18(2):44–57, June 2011.
150. M.A. Moussa and M.S. Kamel. An experimental approach to robotic grasping using a connectionist architecture and generic grasping functions. *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, 28(2):239–253, May 1998.
151. Kai Huebner, Kai Welke, Markus Przybylski, Nikolaus Vahrenkamp, Tamim Asfour, and Danica Kragic. Grasping Known Objects with Humanoid Robots: A Box-Based Approach. In *International Conference on Advanced Robotics*, 2009.
152. Corey Goldfeder and Peter K. Allen. Data-Driven Grasping. *Autonomous Robots*, 31(1):1–20, April 2011.
153. Matei Ciocarlie, Kaijen Hsiao, E. G. Jones, Sachin Chitta, R.B. Rusu, and I.A. Sucan. Towards Reliable Grasping and Manipulation in Household Environments. In *Intl. Symposium on Experimental Robotics*, pages 1–12, New Delhi, India, 2010.
154. Google Goggles. `http://www.google.com/mobile/goggles/`.
155. Sebastien Dalibard, Alireza Nakhaei, Florent Lamiraux, and Jean-Paul Laumond. Manipulation of documented objects by a walking humanoid robot. In *IEEE-RAS International Conference on Humanoid Robots*, pages 518–523. Ieee, December 2010.
156. K. Lai and D. Fox. Object Recognition in 3D Point Clouds Using Web Data and Domain Adaptation. *The International Journal of Robotics Research*, 29(8):1019–1037, May 2010.
157. Stephan Gammeter, Alexander Gassmann, Lukas Bossard, Till Quack, and Luc Van Gool. Server-side Object Recognition and Client-side Object Tracking for Mobile Augmented Reality. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, number C, pages 1–8. Ieee, June 2010.
158. Michael Armbrust, Ion Stoica, Matei Zaharia, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, and Ariel Rabkin. A View of Cloud Computing. *Communications of the ACM*, 53(4):50, April 2010.
159. Amazon Web Services. `http://aws.amazon.com`.
160. Amazon Elastic Cloud (EC2). `http://aws.amazon.com/ec2/`.
161. Google Compute Engine. `https://cloud.google.com/products/compute-engine`.
162. Microsoft Azure. `http://www.windowsazure.com`.
163. Gideon Juve, Ewa Deelman, G. Bruce Berriman, Benjamin P. Berman, and Philip Maechling. An Evaluation of the Cost and Performance of Scientific Workflows on Amazon EC2. *Journal of Grid Computing*, 10(1):5–21, March 2012.
164. Piyush Mehrotra, Jahed Djomehri, Steve Heistand, Robert Hood, Haoqiang Jin, Arthur Lazanoff, Subhash Saini, and Rupak Biswas. Performance evaluation of Amazon EC2 for NASA HPC applications. In *Proceedings of the 3rd workshop on Scientific Cloud Computing Date - ScienceCloud '12*, page 41, New York, New York, USA, 2012. ACM Press.
165. Radu Tudoran, Alexandru Costan, Gabriel Antoniu, and Luc Bougé. A performance evaluation of Azure and Nimbus clouds for scientific applications. In *Proceedings of the 2nd International Workshop on Cloud Computing Platforms - CloudCP '12*, pages 1–6, New York, New York, USA, 2012. ACM Press.
166. TOP500. `http://www.top500.org/list/2012/06/100`.
167. Nitesh Kumar Jangid. Real Time Cloud Computing. In *Data Management & Security*, 2011.
168. Jared Glover, Daniela Rus, and Nicholas Roy. Probabilistic Models of Object Geometry for Grasp Planning. In *Robotics: Science and Systems*, Zurich, Switzerland, 2008.
169. Henry Wang, Yunzhi Ma, Guillem Pratx, and Lei Xing. Toward real-time Monte Carlo simulation using a commercial cloud computing infrastructure. *Physics in medicine and biology*, 56(17):N175–81, September 2011.

170. Martin Sevior, Tom Fifield, and Nobuhiko Katayama. Belle monte-carlo production on the Amazon EC2 cloud. *Journal of Physics: Conference Series*, 219(1):012003, April 2010.
171. D Nister and H Stewenius. Scalable Recognition with a Vocabulary Tree. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 2161–2168. IEEE, 2006.
172. James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Object Retrieval with Large Vocabularies and Fast Spatial Matching. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. Ieee, June 2007.
173. Bharat Bhargava, Pelin Angin, and Lian Duan. A Mobile-Cloud Pedestrian Crossing Guide for the Blind. In *International Conference on Advances in Computing & Communication*, 2011.
174. Javier J. Salmerón Garcia. Using cloud computing as a hpc platform for embedded systems. 2012.
175. Jur van den Berg, Pieter Abbeel, and Ken Goldberg. LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information. *The International Journal of Robotics Research*, 30(7):895–913, June 2011.
176. Ben Kehoe, D Berenson, and K Goldberg. Estimating Part Tolerance Bounds Based on Adaptive Cloud-Based Grasp Planning with Slip. In *IEEE International Conference on Automation Science and Engineering*. IEEE, 2012.
177. Ben Kehoe, Dmitry Berenson, and Ken Goldberg. Toward Cloud-based Grasping with Uncertainty in Shape: Estimating Lower Bounds on Achieving Force Closure with Zero-slip Push Grasps. In *IEEE International Conference on Robotics and Automation*, pages 576–583. IEEE, May 2012.
178. The African Robotics Network (AFRON). "Ten Dollar Robot" Design Challenge Winners. `http://robotics-africa.org/design_challenge.html`.
179. Dmitry Berenson, Pieter Abbeel, and Ken Goldberg. A Robot Path Planning Framework that Learns from Experience. *IEEE International Conference on Robotics and Automation*, pages 3671–3678, May 2012.
180. MyRobots.com. `http://myrobots.com`.
181. What is MyRobots? `http://myrobots.com/wiki/About`.
182. Laura Dabbish, Colleen Stuart, Jason Tsay, and Jim Herbsleb. Social coding in GitHub: transparency and collaboration in an open software repository. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, page 1277, New York, New York, USA, 2012. ACM Press.
183. Alexander Hars. Working for free? Motivations of participating in open source projects. In *Proceedings of the 34th Annual Hawaii International Conference on System Sciences*, volume 00, page 9. IEEE Comput. Soc, 2001.
184. Daniel Nurmi, Rich Wolski, Chris Grzegorczyk, Graziano Obertelli, Sunil Soman, Lamia Youseff, and Dmitrii Zagorodnov. The Eucalyptus Open-Source Cloud-Computing System. In *IEEE/ACM International Symposium on Cluster Computing and the Grid*, pages 124–131. IEEE, 2009.
185. ROS (Robot Operating System). `http://ros.org`.
186. rosjava, an implementation of ROS in pure Java with Android support. `http://cloudrobotics.com`.
187. Bullet Physics Library. `http://bulletphysics.org`.
188. OpenRAVE. `http://openrave.org/`.
189. Gazebo. `http://gazebosim.org`.
190. Erion Plaku, Kostas E. Bekris, and Lydia E Kavraki. OOPS for Motion Planning: An Online, Open-source, Programming System. In *IEEE International Conference on Robotics and Automation*, number April, pages 3711–3716. IEEE, April 2007.
191. A.T. Miller and P.K. Allen. GraspIt! A Versatile Simulator for Robotic Grasping. *IEEE Robotics & Automation Magazine*, 11(4):110–122, December 2004.
192. S. Davidson. Open-source hardware. *IEEE Design and Test of Computers*, 21(5):456–456, September 2004.
193. Erik Rubow. Open Source Hardware. Technical report, 2008.
194. Arduino. `http://www.arduino.cc`.
195. H Hawkeye King, Lei Cheng, Philip Roan, Diana Friedman, Sina Nia, Ji Ma, Daniel Glozman, Jacob Rosen, and Blake Hannaford. Raven II™: Open Platform for Surgical Robotics Research. In *The Hamlyn Symposium on Medical Robotics*, 2012.
196. An open-source robo-surgeon. *The Economist*, 2012.
197. A Sorokin, D Berenson, S S Srinivasa, and M Hebert. People helping robots helping people: Crowdsourcing for grasping novel objects. *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2117–2122, October 2010.
198. Luis von Ahn. Human Computation. In *Design Automation Conference*, page 418, 2009.
199. Juan Camilo Gamboa Higuera, Anqi Xu, Florian Shkurti, and Gregory Dudek. Socially-Driven Collective Path Planning for Robot Missions. *2012 Ninth Conference on Computer and Robot Vision*, pages 417–424, May 2012.
200. Yotam Gingold, Ariel Shamir, and Daniel Cohen-Or. Micro perceptual human computation for visual tasks. *ACM Transactions on Graphics*, 31(5):1–12, August 2012.
201. Matthew Johnson-Roberson, Jeannette Bohg, Gabriel Skantze, Joakim Gustafson, Rolf Carlson, Babak Rasolzadeh, and Danica Kragic. Enhanced visual scene understanding through human-robot dialog. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3342–3348. IEEE, September 2011.

202. Alexander Sorokin and David Forsyth. Utility Data Annotation with Amazon Mechanical Turk. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, number c, pages 1–8. IEEE, June 2008.
203. Fog Robotics. `http://goldberg.berkeley.edu/fog-robotics/`. Accessed: 2018-11-29.
204. A. K. Tanwani, N. Mor, J. Kubiatowicz, and K. Goldberg. A fog robotics architecture for distributed learning of surface decluttering. 2018. (under review).
205. Tim Verbelen, Pieter Simoens, Filip De Turck, and Bart Dhoedt. Cloudlets: Bringing the cloud to the mobile user. In *Proceedings of the Third ACM Workshop on Mobile Cloud Computing and Services*, MCS '12, pages 29–36, 2012.
206. Rodrigo Roman, Javier López, and Masahiro Mambo. Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges. *CoRR*, abs/1602.00484, 2016.
207. Nicholas DeMarinis, Stefanie Tellex, Vasileios Kemerlis, George Konidaris, and Rodrigo Fonseca. Scanning the internet for ROS: A view of security in robotics research. *CoRR*, abs/1808.03322, 2018.
208. G. Pardo-Castellote. Omg data-distribution service: Architectural overview. In *23rd International Conference on Distributed Computing Systems Workshops*, 2003.
209. Yuya Maruyama, Shinpei Kato, and Takuya Azumi. Exploring the performance of ros2. In *Proceedings of the 13th International Conference on Embedded Software*, EMSOFT '16, pages 5:1–5:10, New York, NY, USA, 2016. ACM.
210. Prateeksha Varshney and Yogesh Simmhan. Demystifying fog computing: Characterizing architectures, applications and abstractions. *CoRR*, abs/1702.06331, 2017.
211. John K. Zao, Tchin-Tze Gan, Chun-Kai You, Cheng-En Chung, Yu-Te Wang, Sergio Jose Rodriguez Mendez, Tim Mullen, Chieh Yu, Christian Kothe, Ching-Teng Hsiao, San-Liang Chu, Ce-Kuen Shieh, and Tzyy-Ping Jung. Pervasive brain monitoring and data sharing based on multi-tier distributed computing and linked data technology. *Frontiers in Human Neuroscience*, 8:370, 2014.
212. M. Aazam and E. Huh. Fog computing and smart gateway based communication for cloud of things. In *2014 International Conference on Future Internet of Things and Cloud*, pages 464–470, 2014.
213. Flavio Bonomi, Rodolfo Milito, Preethi Natarajan, and Jiang Zhu. *Fog Computing: A Platform for Internet of Things and Analytics*, pages 169–186. Springer International Publishing, 2014.
214. Kirak Hong, David Lillethun, Umakishore Ramachandran, Beate Ottenwälder, and Boris Koldehofe. Mobile fog: A programming model for large-scale applications on the internet of things. In *Proceedings of the Second ACM SIGCOMM Workshop on Mobile Cloud Computing*, pages 15–20. ACM, 2013.
215. A. Yousefpour, G. Ishigaki, and J. P. Jue. Fog computing: Towards minimizing delay in the internet of things. In *IEEE International Conference on Edge Computing (EDGE)*, pages 17–24, 2017.
216. P. Pop, M. L. Raagaard, M. Gutierrez, and W. Steiner. Enabling fog computing for industrial automation through time-sensitive networking (tsn). *IEEE Communications Standards Magazine*, 2(2):55–61, 2018.
217. Siva Leela Krishna Chand Gudi, Suman Ojha, Benjamin Johnston, Jesse Clark, and Mary-Anne Williams. Fog robotics for efficient, fluent and robust human-robot interaction, 2018.
218. Amazon RoboMaker. `https://aws.amazon.com/robomaker/`. Accessed: 2018-11-29.
219. Google Cloud Robotics Platform. `https://cloud.google.com/cloud-robotics/`. Accessed: 2018-11-29.
220. W. Fung, N. Xi, W. Lo, B. Song, Y. Sun, Y. Liu, and I. H. Elhajj. Task driven dynamic qos based bandwidth allcoation for real-time teleoperation via the internet. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, Las Vegas, NV, US*, October 2003.
221. J. Chen, E. Haas, and M. Barnes. Human performance issues and user interface design for teleoperated robots. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 37(6):1231 –1245, nov. 2007.
222. Y. Jia, N. Xi, Y. Wang, and . Li X. Online identification of quality of teleoperator (qot) for performance improvement of telerobotic operations. In *IEEE International Conference on Robotics and Automation (ICRA), St. Paul, Minnesota*, pages 451 –456, may 2012.
223. G. Podnar, J. Dolan, A. Elfes, S. Stancliff, E. Lin, J. Hosier, T. Ames, J. Moisan, T. Moisan, J. Higinbotham, and E. Kulczycki. Operation of robotic science boats using the telesupervised adaptive ocean sensor fleet system. In *IEEE International Conference on Robotics and Automation (ICRA), , Pasadena, California*, pages 1061 –1068, may 2008.
224. Y. Kwon and S. Rauniar. E-quality for manufacturing (eqm) within the framework of internet-based systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 37(6):1365 – 1372, nov. 2007.
225. Lihui Wang. Wise-shopfloor: An integrated approach for web-based collaborative manufacturing. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 38(4):562 –573, july 2008.
226. P. Debenest, M. Guarnieri, K. Takita, E. Fukushima, S. Hirose, K. Tamura, A. Kimura, H. Kubokawa, N. Iwama, and F. Shiga. Expliner - robot for inspection of transmission lines. In *IEEE International Conference on Robotics and Automation (ICRA), Pasadena, California*, pages 3978 –3984, may 2008.
227. N. Pouliot and S. Montambault. Linescout technology: From inspection to robotic maintenance on live transmission power lines. In *IEEE International Conference on Robotics and Automation (ICRA), Kobe, Japan*, pages 1034 –1040, may 2009.