Cloud-based Robots and Intelligent Space Teleoperation Tools

Tomáš Cádrik[1], Peter Takáč[1], Jaroslav Ondo[1], Peter Sinčák[1], Marián Mach[1], Filippo Cavallo[2], Manuele Bonaccorsi[2]

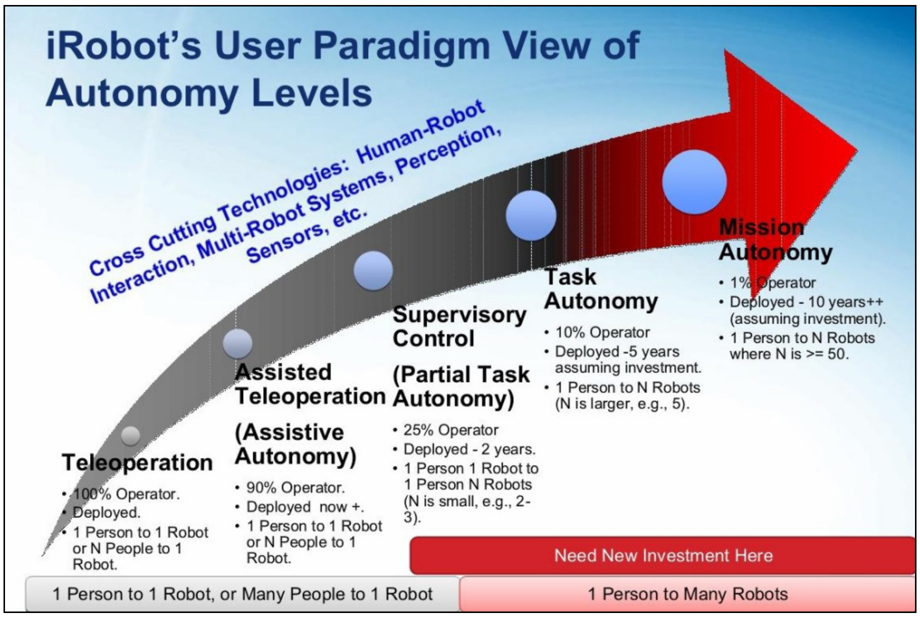
[1] Center for Intelligent Technologies, Department of Cybernetics and Artificial Intelligence, Faculty of Electrical Engineering and Informatics, Technical University of Kosice, Košice, Slovakia
{tomas.cadrik, peter.takac.3, jaroslav.ondo, peter.sincak, marian.mach}@tuke.sk
http://www.ai-cit.sk

The BioRobotics Institute, Scuola Superiore Sant'Anna, Pontedera, Pisa – ITALY
{f.cavallo, m.bonaccorsi}@sssup.it

Abstract. Despite an idea of robotic systems teleoperation, is a relatively old concept, here we present its enhancements heading to an interconnection of teleoperation and collecting relevant information from the environment where robots act. This environment should be an intelligent space featured with various devices and sensors, which allows to obtain, preprocess and stores data in the cloud. Those data should provide relevant information for teleoperator or directly for robots, which act autonomously. For this purpose, we developed cloud-based tools, named Telescope v2. It is a platform-independent system for remote monitoring and controlling various systems. In this paper, we introduce this system, its abilities, and compare it with its network-based ancestor, Telescope v1. We analyze measurements of latency and response time when our new system is used for teleoperation in different places equipped with various internet bandwidth.

1 Introduction

Teleoperation is an important factor of today's service and social robots. The basic aim of teleoperation is being able to operate a robot in remote or non-approachable environment. The human person operating a remote robot is using all the information to be able to control the robot by all the means of technology. The latest results in Cloud Computing are giving more and more conditions to fulfil the idea of remote brain concept [1]. The practical aim is the during teleoperating a robotic system "would be nice to learn" and offload the human which is teleoperating a robot from more routine work and this possible by creating intelligent agent on the Cloud environment which is doing gradual replacement of human which is operating a robot towards anonomous robots. The figure 1 shows the concept of the U.S. IROBOT Company of creating an intelligent machine using tele-operation and engaging learning procedure into the process. The application potential of this approach is rather large, since a number of autonomous machines are expected to appear on the market in the near future.



iRobot's User Paradigm View of Autonomy Levels
Cross Cutting Technologies: Human-Robot Interaction, Multi-Robot Systems, Perception, Sensors, etc.
Teleoperation
• 100% Operator.
• Deployed.
• 1 Person to 1 Robot or N People to 1 Robot.
Assisted Teleoperation (Assistive Autonomy)
• 90% Operator.
• Deployed now +.
• 1 Person to 1 Robot or N People to 1 Robot.
Supervisory Control (Partial Task Autonomy)
• 25% Operator
• Deployed - 2 years.
• 1 Person 1 Robot to 1 Person N Robots (N is small, e.g., 2-3).
Task Autonomy
• 10% Operator
• Deployed -5 years assuming investment.
• 1 Person to N Robots (N is larger, e.g., 5).
Mission Autonomy
• 1% Operator
• Deployed - 10 years++ (assuming investment).
• 1 Person to N Robots where N is >= 50.
Need New Investment Here
1 Person to 1 Robot, or Many People to 1 Robot
1 Person to Many Robots

Fig. 1. The approach of Tele-operation toward Mission Autonomy operation [2]

One of the major problems related to Intelligent Machines are the difficulties in measuring the autonomy of the system. There is no universal approach to this problem. We can state that it is a task or mission oriented approach and therefore we write it as:

$$\mathbf{GTI = HTI + MTI} \tag{1}$$

where

GTI (Global Task Intelligence) is always value 1, is a sum of Human Intelligence „HTI (Human Task Intelligence) from interval <0,1> and Machine Intelligence MTI (Machine Task Intelligence) from interval <0,1>. Also, we can define a Machine Task Intelligence Autonomity"(MTIA) as follows:

$$\mathbf{MTIA = MTI / HTI} \tag{2}$$

So, when MTIA is 0, we are describing a manual, fully human-made process, since HTI is 1 and MTI is 0. If MTIA is a very large number, HTI is very small close to „0“ and MTI is close to „1“. This can be considered as an autonomous mission where a human is the only observer. The further consideration of MIQ Machine Intelligent Quotients related to machines have been studied in the past [3]. The MIQ should be domain-oriented and could be used in future for commercial advantage by selling various machines to humans.

The goal of teleoperation is to allow a human to control a robot in a situation where it is inconvenient or unsafe to place a human and difficult to program a robot to perform autonomously complex operations [4]. Nowadays, teleoperation tasks variate from very basic like a toy car teleoperation by remote controller, teleoperation of firefighter drones equipped with cameras, up to high-tech teleoperation task in the field of medicine, army or space exploration. Many of those tasks demand the robot to perform more complex and difficult works [4].

With the development of high-speed Internet networks, it has formed a network-based robot teleoperation with a combination of the advantages of network technology and robot technology [5]. We used just this technology to build the Telescope v1 system.

Along with the development of high-speed Internet networks, the cloud services have become popular. Cloud technology have brought a whole new concept of software programming, operation, and maintenance. Cloud providers guarantee a worldwide 24/7 availability, offer autoscaling of deployed services in the case that demand exceeds limits of service and the last but not least on-demand computational and storage sources [6][7]. On the other hand, it requires to such applications be deployed somewhere on the Internet, more precisely in cloud providers data centers. The cloud computing has become popular also in research. An evidence of that is the emergence of a new approach, named cloud robotics, which James Kuffner defined in 2010 in the paper [8].

This new approach inspired us to the development of a cloud-base system for robot teleoperation. Moreover, we enhanced typical teleoperation systems with the possibility of operation different robotic systems and also with a possibility of acquiring relevant information not only from robots but also from the environment, where the robot acts. As was mentioned; this environment should be an intelligent space. By intelligent space, we mean a room equipped with many cameras and sensors, which enable the space to perceive and understand what is happening in them [8]. Such space also can provide additional information for the operator and also directly for robots.

The paper structure is as follows. Chapter 2 briefly describes Telescope v1 system, as an ancestor of cloud-based Telescope v2 system. Then the attention is focusing on Telescope v2 description, which consists of robot teleoperation module and intelligent space devices control module. We compare old and new system and highlight the advantages of each solution. In Chapter 4 we focus on experiments and their evaluations.

2 Server-based teleoperation tool – Telescope v1

Telescope v1 was developed as a server-based modular system, which allows operating connected devices through the Internet all over the world. Simply, it allows teleoperation of connected devices. In our understanding, we consider a device as any electronic appliance that has an ability to communicate over the network and can be accessed programmatically (using API).

We designed the system for various independent operators, who can teleoperate their connected devices at the same time. We made a special effort to implement Telescope v1 to be platform independent and so solve a problem how to connect and operate devices with heterogeneous environments. By heterogeneous environment, we mean devices with different interfaces for programmers (API), programmable with various inconsistent programming languages and feature with diverse user's interfaces. We solved that using a wrapper program (similar to a device driver), which translates Telescope's commands into the commands understandable for the appropriate device. Such wrappers were implemented for each type of connected device.

In the Fig. 2 is an architecture of Telescope v1 system. It consists of two integrated modules – Teleoperation module and Telediagnostics module, which provides a full set of information about the state of the operated device and its possible faults or errors for the operator. It is necessary for successful completion of the task remotely.

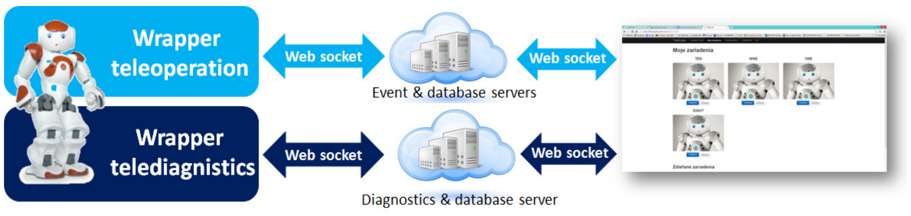


Fig. 2 The basic architecture of the Telescope v1 system.

The essential part of the system is Event server. This server interconnects messages coming from the appropriate operator and device(s), which the operator teleoperates. Web server hosts Telescope and Telediagnostics web portals, and provides controls and overview of connected devices for operators [10]. Each device can connect and communicate with Event server only in the case, that there is installed wrapper program. Simultaneously, we implemented Telediagnostics server, which provides a service for acquiring and storing data of healthy state, faults, and errors of connected devices. By comparing up to date data and stored historical trends, we can identify arisen problem of the device. For all ongoing communication, the web socket [10] technology is used.

The test bed for the Telescope v1 was a Nao robot, which is a complex robotic system, consists of various sensors and actuators. The Telescope v1 allows to operate almost all functions of Nao robot, including single joints movements, stiffness settings, text to speech, launching pre-programmed behaviors, like sitting, walking, etc. Moreover, the system allows adding, removing and sharing devices with others users, what makes possible to operate one robot by various operators, or operate several robots by one operator.

3 Toward to cloud-based teleoperation – Telescope v2

Here, we describe more deeply a cloud-based solution for teleoperation, named Telescope v2. This system evolved from Telescope v1 and enhanced its functionality by gathering relevant information from devices installed in the intelligent space. It allows to gain supporting information for robot teleoperation. Since the Telescope v2 consists of robot teleoperation module and a module for data acquiring from intelligent space, this section is divided into two subsections, which describe each module separately.

3.1 Robot teleoperation module

Telescope v2 is a platform for multiple robot types teleoperation, which is based on the Microsoft Azure cloud platform. This system has a big advantage according to his predecessor, which used only a server architecture. The advantage when using the cloud is you do not need to care about the physical machine. Also, the scalability can serve for the processing of many requests of the users. If many users want to connect to the system, we can simply increase the computational capabilities of the machine.

The system consists of two main parts. One is for the interaction with the user, and the second one is for the whole communication and calculation logic. This architecture is shown in the Fig. 3

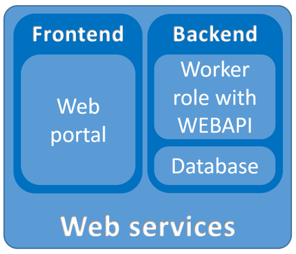


Fig. 3 The basic structure of the cloud-based Telescope v2 system.

The front end is a simple web page that can be used by the user for teleoperation. The user can use the web page for adding a new type of robot, or adding a movement for the robot. For this frontend was the MVC.ASP.Net used [11] which can be simply explain as a combination of the C# programming language and HTML.

The worker role serves for starting and controlling the WEBAPI [10]. Because WEBAPI usually runs on the frontend size of the cloud service, we needed to find a way how to start it on the worker role. The result was the Owin self-host library which serves exactly for our purpose. The WEBAPI is used for the communication between the robots and the service. It is a technology, which allows sending and receiving messages via http protocol. That means that you can use an Http client to call the specific function in our WEBAPI controller using a simple url. In this system is the WEBAPI controller used for sending a request when a user clicks on the specific movement and for sending movements when a robot requests for a movement.

The last part of the system is the databases. At the beginning, only two of them are created. The first one is the table with the list of added robots, and the second one is the blob container (for storing raw data) for the files with the description of an added movement. When the user adds a new robot new table and blob container is created. This table is then used for storing the movements of the robots.

The movement is represented by a python script. This python script is specific for each robot. For instance, the movement for the NAO robot has code using naoqi and the robots based on the robotic operating system are using subprocess module for calling the rostopic commands.

When we need to connect a robot to a cloud service, we cannot do it directly. It is possible with using VPN but we wanted to solve this without it. So we created a wrapper which is universal for all Linux based robots. This wrapper is able to communicate with the WEBAPI service and can also save the time response between the robot and the cloud service.

The paring mechanism in this system is very simple. The user adds a unique string on the website and when he chooses the movement, the combination between the type of the robot with the unique number and the selected movement is saved to a dictionary. And when the wrapper sends a request to the cloud service, it also sends the type of the robot and the unique number of this robot. If the dictionary contains a movement for this concrete robot, it is send to it.

There are three possible scenarios how to use this cloud service. First one is that one user can control one robot. The second one is that multiple users can control one robot. At one time the robot can do only one action, but we can change it in the future. The last way how the system can be used is that one user can control multiple robots which can move at the same time.

WEBAPI was also chosen because it is easy to create another interface which is using the cloud service. The application can simply use http calls to the WEBAPI via URL. So the developer can create an application which uses for instance some sensors for controlling the robots.

3.2 Module for acquiring data from the intelligent space

While the teleoperation schemes of Telescope v1 and Telescope v2 can handle the user robot communication and control it can collect data only acquired by the robot itself. In case of an indoor environment the robot would have to rescan the given environment each time it makes an action. This approach is time and also resource consuming. Hence, we have used an approach gathering pre-processed data from intelligent space with the ability of storing the relevant data directly in the cloud environment.

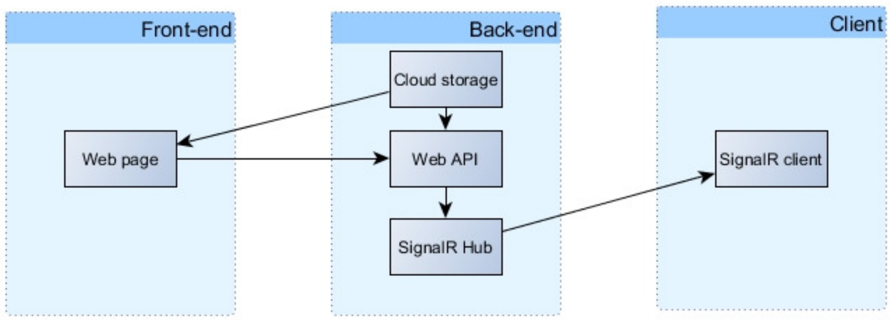


Fig. 4 The basic architecture of the teleoperation in intelligent space.

The proposed solution relies on the ability of Microsoft AZURE Cloud Storage to store multiple types of data, which can be accessed efficiently in real-time. Furthermore, the Microsoft AZURE Cloud is used as the main platform for the user device communication, ensures the commands routing, keeps track of processes and to some extent ensures data gathering.

The system consist of three main parts (see Fig. 4). The first part serves as the frontend for the interaction with the users, the second part is the main cloud-client communication module and the third part is the client application in the desired local environment, where the devices are placed.

For the front-end the ASP.NET MVC 5 [11] was used, which offers the user control capabilities in form of graphical interface, where all the clients and devices are shown. This front-end was placed on AZURE Web Role and is interconnected with the back-end by WEBAPI technology, which offers making simple method-like calls from one Role to another in real-time.

The second part is the back-end which is represented by AZURE Worker Role, which holds the WEBAPI controller for front-end to back-end communication and the SignalR Hub, which further communicates with the client application. The SignalR technology [10] is based on Web socket communication where the server-side logic (Hub) is responsible for connection registration and in/out connection management, whereas the client-side logic is only able to receive incoming connection and call methods from the Hub. For more complex mechanism for returning messages from the client a number of modifications had to be made. These modifications mainly influenced the Hub, where several methods were created for the client, which can this way notify the Hub about any relevant information or even send data back.

Together, five buffers were created: Client buffer, Device buffer, Process buffer, Data buffer and Device-related data buffer. The Client and the Device buffer are in the form of simple dictionary, which hold the data about the connected client applications and about the devices connected to them. The Process buffer is the most crucial part of the system, while it holds the information about all the running processes i.e. running data gathering. It ensures that only one type of action is performed on a single device, while there can be multiple accesses to the given client. The last two buffers were created especially for data upload, which can by mainly used for sending data to other Roles for further processing. The reason behind why there are two buffers is to ensure simplicity and flexibility within the Hub, while multiple types of devices could be feeding data to the Hub and this buffer stores them in a joined queue and then distributes them to their appropriate containers within the Device-related data buffer, where the exact types can be stored.

The third and the last part is the client application, which runs on Windows machine to which the devices in the environment are connected. It is a simple console application where the SignalR client is running. The client is able to communicate with devices and forward the user input from the cloud to the device the way it would be controlled locally using device controllers. For each type of device there is a unique controller native for the communication with the devices firmware. Moreover, this client is also responsible for its registration and also the registration of the devices connected to it by uploading information about itself and the devices to the AZURE Cloud Table, which serves as a simple database. In some cases it is important to upload only raw data to the cloud with no pre-processing needed and in this case the client application is also able to upload the gathered data directly to the AZURE Cloud Storage.

For being able to control the devices from the cloud at least one client with a controllable device have to be connected to it and running. The user can then easily select the desired client with the device and perform action with this device. The input from the user is then forwarded to the process buffer, where it is noted that a certain process started after the device has started performing the given action and only after then the user is notified by the system about the action success. Usage of the Web socket technology and the request sizes ensure that this whole action is only matter of milliseconds.

4 Experiments

We have done preliminary experiments focused on latency and response time measurements. Specifically, we measured reaction time in the teleoperation module and latency in the module for acquiring data from the intelligent space.

We focused on the experiments that show whether the used technologies (WEBAPI and SignalR) are a good choice for this purpose. In the first case, we tested the WEBAPI service that is the main component in the robot teleoperation module. The object of those experiments was to measure the time between the movement when the operator entered the request on the teleoperation website and the moment when the robot started to act. The times were saved in local log files in the robot and then they were sent to the cloud storage.

Till now, we measured a reaction time during the teleoperation of the Nao, Qbo and Hanson robots. The operator and the robots were located in Košice, Slovakia, but the teleoperation was done through the cloud service. The results of the experiments are shown in Fig. 5.

The experiments show that the fastest connection was with using the Qbo robot. The most unstable connection was using the Nao robot. When we do not count the measurements that are untraditional, the average was between 86 and 259 milliseconds. The most stable connection was on the Qbo robot. For now we do not know whether the connection deviations are because the WEBAPI technology or the device. More measurements need to be done to analyze this.

In the case of experiments of teleoperation in intelligent space the measurements were based on the time that took to send the request to the cloud and receive the response. This measurement is the key to be able to tell if this system is usable in real-time applications. The experiment was carried out as follows:

1. The SignalR client created a request, which consisted of simple binary data containing a simple string message notificating the SignalR Hub about the available information, i.e. the name of the client, client's connection ID and the connected devices information.
2. The SingalR Hub receives the information, checks wheather it is in a right format and if there are no duplicate requests, it proceeds to respond back to the user.
3. The SignalR client receives the boolean information and if the connection was succcesful (true value of the received boolean variable) it stores the final time measurement.

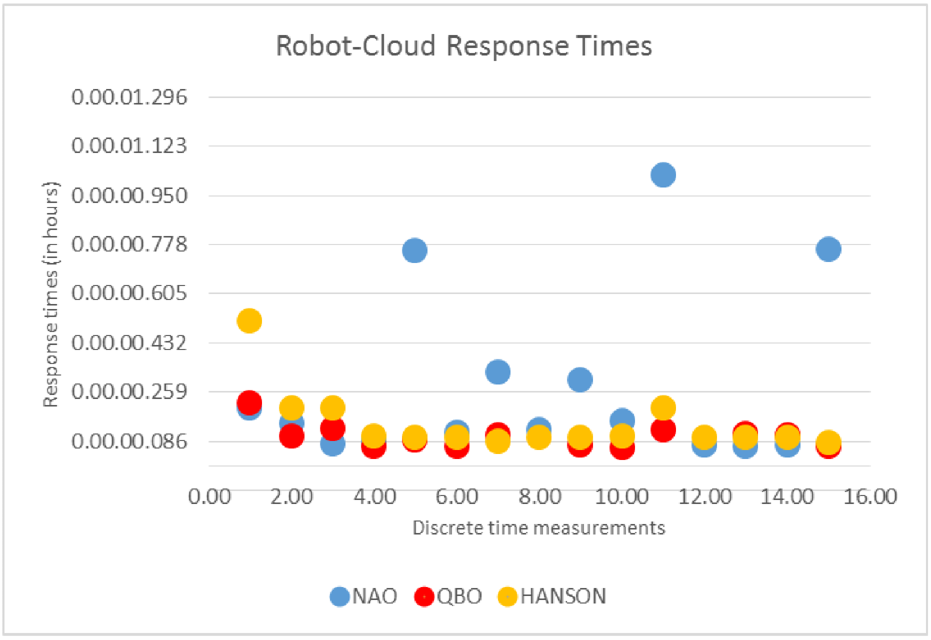


Fig. 5 The graph shows the results of Robot-Cloud response times from the latency measurements.

During our experiments there was no failure in sending data in the proper format and while there was also no conflict, we managed to get the time measurement of every request made by the SignalR client. The results of the experiment are shown in Fig. 6. We can see that our system managed to get the response from the cloud on average 63 milliseconds, which is a very good result, if we consider that the basic latency measurements of our network were around 15 milliseconds. The best latency measurement was 36 milliseconds and the worst measurement was 103 milliseconds. These data leave sufficient time window for further processing of the gathered data and also for data larger in size to be uploaded by this system.

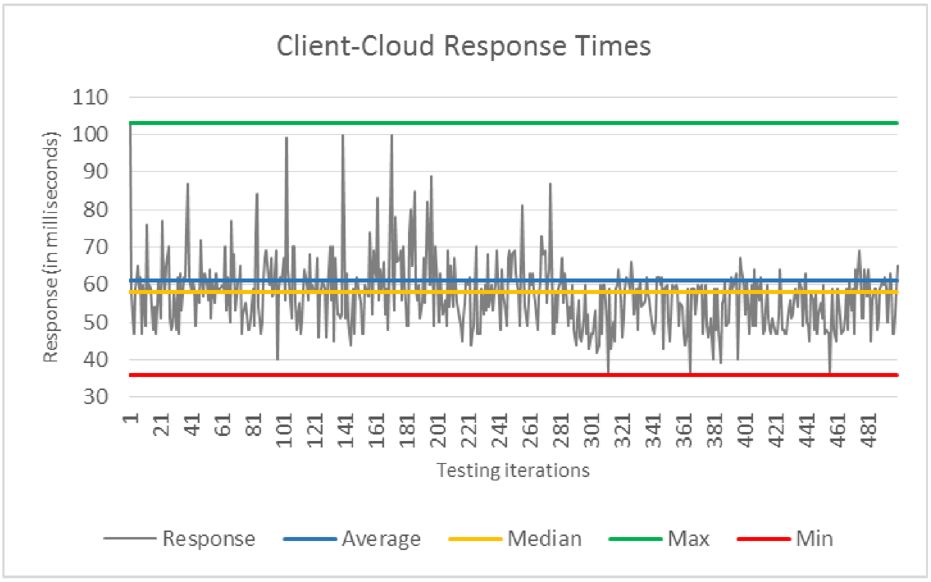


Fig. 6 The graph shows the results of Client-Cloud response times from the latency measurements.

5 Conclusion

Taking into consideration that all experiments were performed locally, where the operator and also the robot were in the same place, although the control was done via the cloud service. We cannot claim, that the used methods are the right ones for teleoperation purpose. The results of our experiments were good enough, but we need to approve our hypothesis based on much more measurements. This way, we would like to ask volunteers to perform tests for us with our system, which is available on [14]. We need to collect teleoperation data generated during the robot teleoperation from different places all around the world, connected to the Internet using various bandwidth and by several Internet providers. Then, we will able to analyze acquired data, upgrade our system and conclude the results. If someone is interested to participate on this, please contact us using the email address peter.takac.3@tuke.sk.

6 Acknowledgment

This publication is the result of the following Projects implementation: University Science Park TECHNICOM for Innovation Applications Supported by Knowledge Technology, ITMS: 26220220182 and also partially by the National Research and Development Project Grant 1/0667/12 “Incremental Learning Methods for Intelligent Systems” 2012-2015.

Bibliography

[1] Inaba M. at al. : A Platform for Robotic Research Based on the Remote-Brained Robot Approach, The International Journal of Robotic Research, Vol 19, No 10, October 2000, pp. 933-954

[2] iRobot's user paradigm view of autonomy levels. Robotics Summit, Virtual Conference & Expo, June 2011

[3] Park H., Kim B.,Kye L. : Measuring the Machine intelligence Quotient (MIQ) of Human-Machine Cooperative Systems, IEEE Trans. On SMC / Part A Systems and Humans, Vol 31, No 2, March 2011

[4] Hu, H. H. H., Li, J. L. J., Xie, Z. X. Z., Wang, B. W. Bin, Liu, H. L. H., & Hirzinger, G. (2005). A robot arm/hand teleoperation system with telepresence and shared control. Proceedings, 2005 IEEE/ASME International Conference on Advanced Intelligent Mechatronics., 24–28.Song, W., Zhao, S., Jiang, F., Zhu,

[5] K., Cao, L., & Shi, Y. (2012). Teleoperation Robot Control System Based on Mindband Sensor. 2012 Fifth International Symposium on Computational Intelligence and Design, 299–302.

[6] NIST. (2011). The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology. Nist Special Publication, 145, 7.

[7] Lorenčík, D., Cádrik, T., Mach, M., & Sinčák, P. (2014). Cloudová robotika; Vplyv cloudového computingu na budúcnosť robotiky. ATP Journal, 3(1), 40–42.

[8] J. J. Kuffner, "Cloud-enabled robots," in IEEE-RAS International Conference on Humanoid Robotics, 2010.

[9] Lee, J.-H., & Hashimoto, H. (2002). Intelligent Space — concept and contents. Advanced Robotics, 16(3), 265–280.

[10] Miženko, L., Lorenčík, D., Ondo, J., & Sinčák, P. (2014). Telescope: System Overview. In B. Sobota & S. Balász (Eds.), Developments in Virtual Reality Laboratory for Factory of the Future (p. 100-105). Miskolc: Bay Zoltán Non-profit Ltd.

[11] ASP.NET MVC, Available online: http://www.asp.net/mvc.

[12] ASP.NET WebAPI, Available online: http://www.asp.net/web-api.

[13] ASP.NET SignalR, Available online: http://www.asp.net/signalr.

[14] Telescope v2 website, Available online: http://wizardofoz.cloudapp.net/.