Virtual Master Device

Gábor Sziebig, Béla Takarics, Péter Korondi

Budapest University of Technology and Economics, Hungary
gabor.sziebig@gmail.com

Viliam Fedák

Technical University of Košice, Slovakia
viliam.fedak@tuke.sk

Abstract: This paper presents a virtual master device for telemanipulation systems, which transmists the operator's movement to the slave site not in a mechanical way, as in case of a real master device, but by image processing. The virtual master device uses cameras as sensors, subtracts the background from the images, and recognizes the skin color of the operator in real time. The 3D position of the operator's hand is determined by stereo vision and it becomes traceable by the color of the skin. This way the virtual master device comes to existence. The virtual master device does not have real force-feedback, but it has much more capeabilites to inform the operator as a real master device. This feedback can be any kind of human sense (visual, oral or any kind of force). Whit this capabilities exceeds its competitors (such as a joystick). The virtual master device uses many technologies to be flexiable and to create wide area of usability, which will be intoduced in this paper (OpenRTM-aist, DIMAN, Player / Stage project).

Keywords: RT-middleware, OpenRTM-aist, Telemanipulation, Master Device, Robot programming

1 Introduction

The current paper is dealing with a special kind of telemanipulation. Our aim is to control virtual autonomous robots without any physical contact, by the help of a virtual master device.

Generally telemanipulation [8] is extending the human's (operator's) sensing and acting capabilities to a remote place, where some manipulation task to do. The telemanipulation process so thus the telemanipulation system can be divided into three major parts (see in Figure 1): the master device, which creates the connection between man and machines, the slave device, which does the work at

the remote environment and the information transmitter channel between the master and slave device.

In our case, the master device is built on the concept of the 'Intelligent Space' [3]. We call a space (room, corridor, building or street) Intelligent, when it is equipped with ubiquitous sensors, which monitor the state of the space all the time. After processing the information from the sensors, an artificial intelligent based control algorithm can detect the events happening in the space and after a learning period it can categorize and evaluate them. Based on the current situation, the algorithm can autonomously decide. The concept of the Intelligent Space was developed in Japan, and it is expected to spread also in Europe. This Intelligent Space concept is used in our newly developed distributed image analyzer (DIMAN) framework, which is one of the key elements of our work. The goals of the DIMAN are more general [5] than that of the current application. DIMAN is considered as a framework for the master device. A camera's picture is used for the color skin detection and it is also the source for the position detection of the operator's hand movement [4]. These calculations are the inputs of the robot order generation, which are used in the virtual robot simulator.

In our case, the remote environment is generated by the computer for our virtual robot, what is the Player / Stage [2,6] robot simulator and controller program. In our work we present a simulation, in which the operator's goal is to push a ball into a gate whit a virtual robot.

Another key element of our work is the transmitter channel, which is based on RT-middleware (OpenRTM-aist-0.2.0) platform. The RT words stand for Robot Technology and it is not the ideal platform for image processing. RT-middlewarewas toggled because in Japan the Next-Generation Robots [7] use this platform, so thus the adaptation of our image processing system is made simpler in newly developed Japanese robots.

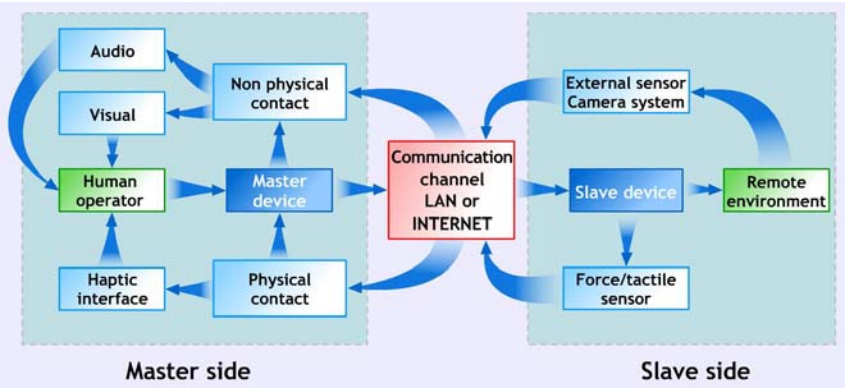

Figure 1
Information streams of the Telemanipulation

The remaining part of the paper is organized as follows: In the next section the concept of the virtual master device is introduced, among with the Distributed Image Analyzer framework, which is mentioned earlier. The evaluation of the virtual master device can be found in Section 3.

2 Concept of the Virtual Master Device

In the following sections the basis of our virtual master device will be introduced, as mentioned in the introduction. First the concept of the RT-middleware [1] is presented, which is the key element of our master device.

2.1 RT-Middleware

The Japanese Ministry of Economy, Trade and Industry (METI) in collaboration with the Japan Robot Association (JARA) and National Institute of Advanced Industrial Science and Technology (AIST) started a 3 year-national project 'Consolidation of Software Infrastructure for Robot Development' in 2002. With the intention of implementing robot systems to meet diversified users' needs, this project has pursued R&D of technologies to make up robots and their functional parts in modular structure at the software level, and to allow system designers or integrators building versatile robots or systems with relative ease by simply combining selected modular parts.

The robot technology middleware having been developed as infrastructure software for implementing the proposed robot architecture, named 'OpenRTM-aist'.

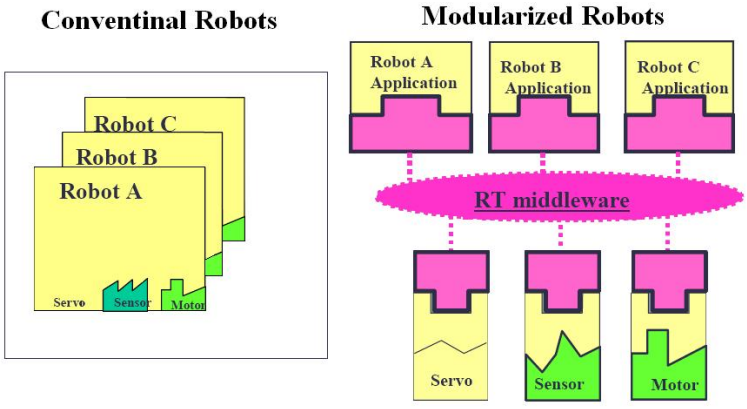

Figure 2
Difference between the conventional and modularized robot concepts

The answer of why RT-middleware is needed can be seen in Figure 2. Not only thousands of hours of robot programming could be saved, but even more the interoperability between simulation and real applications in robots is solved.

Two prototype systems have been made to ascertain the effectiveness of the developed RT-middleware.

- A robot arm control system based on real time control.
- A life supporting robot system (RT space, also known as Intelligent Space), one of promising applications.

2.2 Open Robot Technology Middleware (OpenRTM-aist)

As reference implementation software to verify the concept of RT (Robot Technology) middleware, OpenRTM-aist has been developed. OpenRTM-aist has been configured as a programming support tool to develop a robot system. To be more specific, the system includes a support tool for making RT-Component, a tool for making combined RT-Components into a new RT-Component, and a graphical user interface for linking RT-Components and making operation control.

The framework is build from components. Each component is supervised by one RT-Component Manager. The manager communicates with other RT-Component Managers and with the base RT-Component, and manages the life-cycle of the base component. The life-cycle of the components is shown in Figure 3.

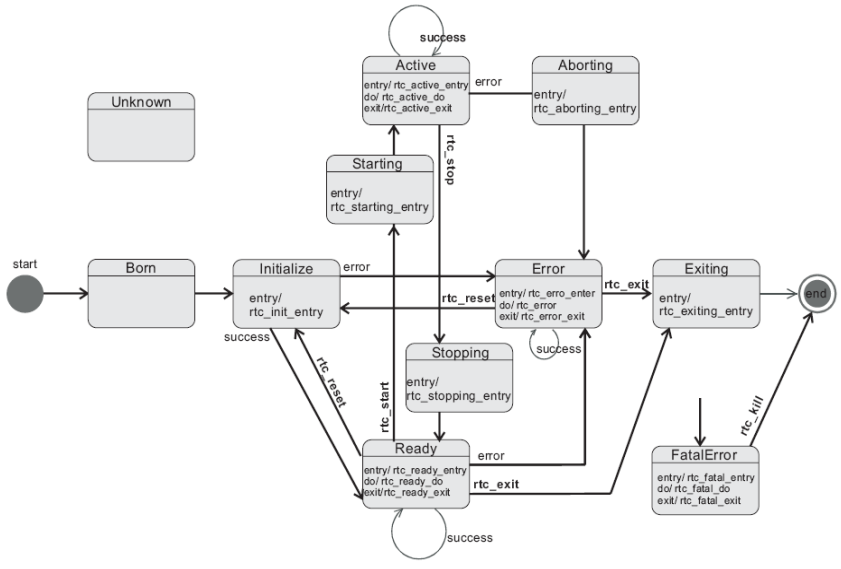

Figure 3
Life-cycle of an OpenRTM-aist component (Source: OpenRTM-aist Developers' Guide)

The Base components represent the functional part of the framework. This level is responsible for data processing, or for example moving MHI PA10 manipulator.

The communication between RTC Managers is realized with CORBA objects that are called InPort / OutPort. This hierarchical structure and the CORBA system provide the power of OpenRTM-aist.

In the conventional robot system, a slight modification has required an extensive alteration of software. With RT-middleware, it has been made possible to provide new services by creating a module of new functional part (RT-Component Manager) for providing necessary functions and posting it in the network, even when user-requested services cannot be implemented with the existing functional parts.

CORBA provides transparency of all levels of programming, from object location to protocol transparency. This means that the client doesn't need to know where an object is located; he only knows that it is reachable for the program at any time. The use of the CORBA components in RT-Components is shown in Figure 4.

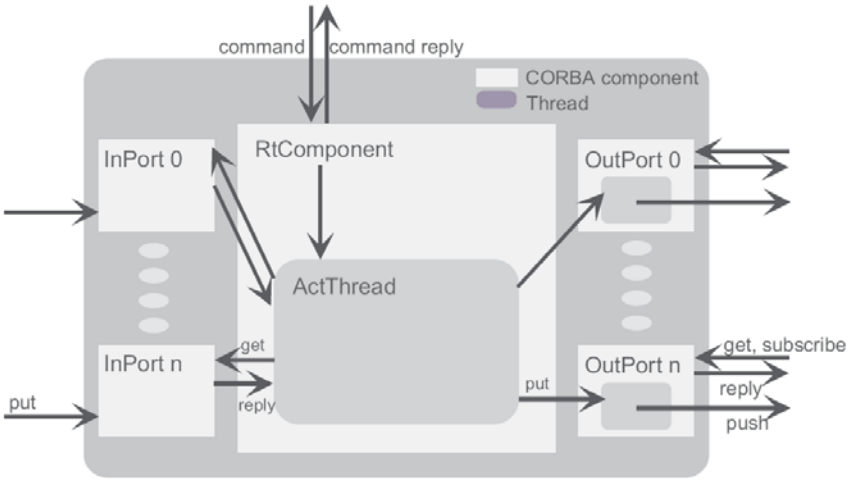

Figure 4
Use of CORBA components in RT-Components (Source: OpenRTM-aist Developers' Guide)

Concept of the OpenRTM-aist and the already developed framework made basis of our virtual master device. With inheriting this framework in DIMAN the virtual master device became usable in Intelligent Space.

2.3 Distributed Image Analyzer (DIMAN)

This software provides a framework for distributed information processing, with a special respect on visual information. The framework allows the usage of several

computers interconnected with a local area network, to enhance processing speed. The framework can accommodate several modules that communicate with each other to send and receive data to process. New modules can easily be added to the framework; only the information processing function has to be implemented. The framework is implemented entirely in Microsoft .NET 2.0.

2.3.1 Components of the System

The framework is based on two different components: the Control Unit (CU) and the Processing Unit (PU). The Processing Units holds the main basis of the framework, the modules. The module represents a logical entity of the framework. The real data processing is done in these modules. In one instance of the framework there can be only one CU, on the other hand, at least one CU is necessary otherwise the framework is unable to operate. Unlike the CU, there may be several PU in an instance of the framework. However, on one computer only one PU can run, but a PU and a CU can run on the same PC.

The components communicate with each other through the TCP(UDP)/IP protocol stack. The CU provides a GUI for the user, where the graph of module connectivity can be easily edited. The graph can be saved into or opened from a file. Most of the user interactions with the framework take place using the CU. It is the CU that instructs each PU to build up the module graph. The PU can start the simulation of the module graph (a logical interconnection of modules) after it has been configured. The communication is demonstrated in Figure 5.

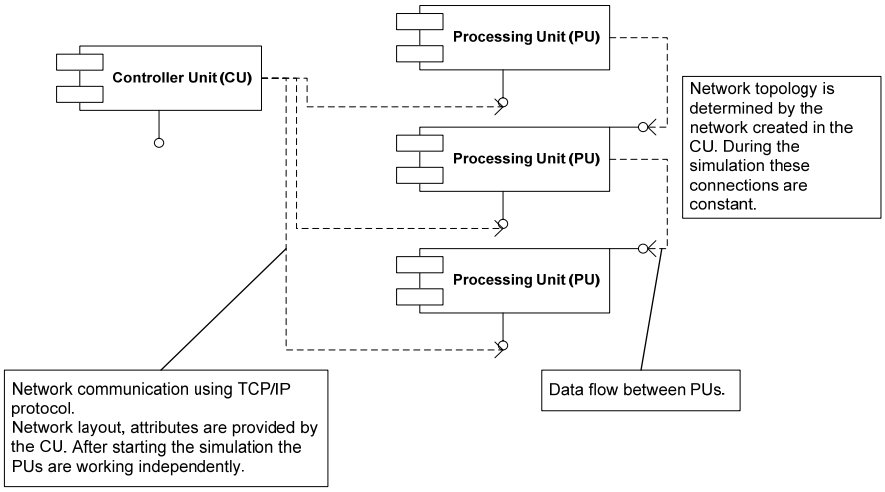

Figure 5
Communication between units (Source: DIMAN project)

2.3.2 Control Unit

The controller unit has a very simple user interface. The window is separated into 2 parts. The right pane contains the module repository. The blocks of the modules are listed there. The left pane contains the workspace. The simulation network can be assembled here. The layout of the windows is shown in Figure 6.

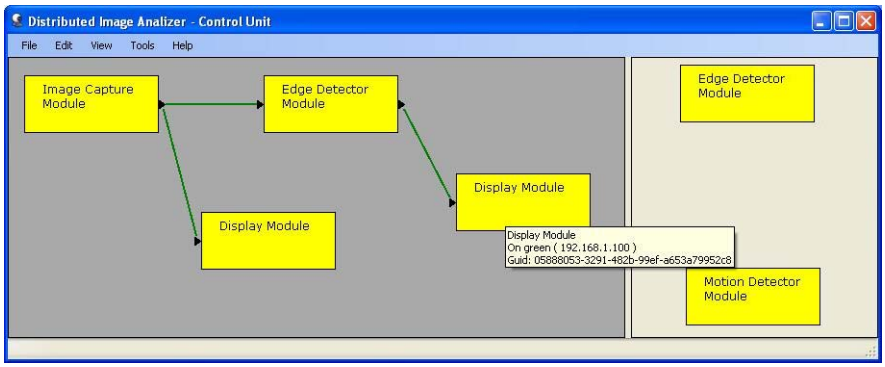

Figure 6
Control Unit's user interface (Source: DIMAN project)

2.3.3 Processing Unit

The processing unit is unique on one PC. On startup it checks if there is another instance running, and if yes, it opens a message box saying that the PU is already running, and the program exits. When a PU starts up, it starts to listen to the 11224 UDP port for broadcast messages from a CU. When a CMD_IS_ALIVE message is received from the CU, the PU sends back a CMD_ALIVE message meaning that it is ready for data processing, and also it sends back the supported modules.

Meanwhile, in the CU the user assembles a module graph, and when it is ready, it is passed to the PU, which builds up the module graph of the modules that are associated with it. On reception of a CMD_PU_START message from the CU, the PU starts to operate its modules. On the reception of a CMD_PU_STOP message, the PU stops the operation.

The PU is running in the background, and it does not require any user interaction. A small icon on the icon tray signals that a PU is running on a computer.

However, it may happen that a PU is running an output module that is supposed to render the data it was given. In this case the module opens a window and displays the data in that window. If the window is not closed and a new piece of data arrives, it is displayed in the same window that is already open.

2.3.4 Modules

The modules represent the operators for information processing in the DIMAN framework. They are designed to cooperate as a distributed network of modules, allowing a higher level of complexity. A module provides standard interfaces for communication with other modules. They also provide a standard API for the PU, which can handle each module the same way, as a black box.

The development of a new module is rather simple. A new module is derived from the Module class. This holds the communication interfaces and the API, only the data processing part of the module has to be done.

2.3.5 Communication

The components have to communicate with each other through the network, which requires the definition of the communication protocols between them. There are two basic type of communication between the components: Command messages and Data messages.

2.3.6 Command Messages

The CU is only responsible for controlling the activity and state of the PUs, and the modules inside them, but ignores the data that is passed from one module to another. The command messages thus carry commands and acknowledgements between the CU and PU. It is possible that a command or an acknowledgement have some parameters. These parameters can be the data required by a PU to build up its module network, or it can also be status data sent from the PU to the CU containing for example the progress of the modules.

2.3.7 Data Messages

Data messages are carrying information between the PUs. Their task is to transfer the data that is sent from one module of a PU to a module of another PU. Data messages are sent by the PU to other PUs. When a module within a PU has some data to send to another module, or to several of those, it passes a Data Unit to the PU it is running within. The Data Units are than serialized and sent to their destinations via the TCP/IP protocol.

In order to avoid sending data to destinations that are not ready to accept it, the PU sends an IsReady request to each destination. Those modules that respond to be ready, will be passed to the modules, which assemble the Data Units according to this information. The response about readiness is passed to the modules so that they can decide themselves if they keep the output data in their buffers waiting until it is passed to all destination modules, or they just drop that piece of data and generate a new one to be sent to the destination stub at a later time, when it may be ready to accept it.

3 Evaluation of the Virtual Master Device

To test, and to evaulate our virtual master device a sample system was assembled. This system will be introduced in the next sections.

3.1 The Hardware

To set up this system, three computers and two USB web cameras were used, as seen in Figure 7. The USB digital pan/tilt cameras are suitable for the experiments, because they are relatively cheap and easily manageable in any environment. Their resolution is 640 x 480, which is not enough for an industrial environment, but for us to test the system, this resolution is high enough.

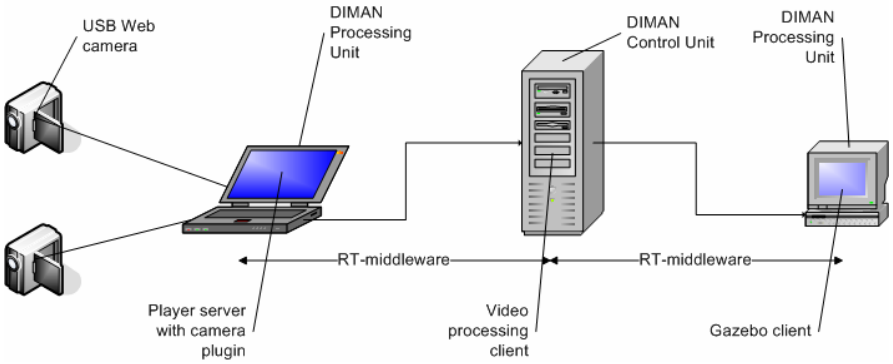

Figure 7
Set up of the test system

3.2 The Software

As seen from the previous section, the newly created telemanipulation system is built up from complex systems. From the introduced software only those parts were adopted from which we can benefit. The reader may feel little bit confused, why we spent so much time introducing DIMAN and OpenRTM-aist framework, but it is one of the key elements what we used in our research activity. The DIMAN framework was originally created in another research for image processing, and the communication was based on standard socket programming (TCP/IP), and OpenRTM-aist is a must have, to be able to communicate in Intelligent Space.

With adopting OpenRTM-aist in DIMAN framework the system is able to make manipulations in Intelligent Space. Any devices, robots, manipulators, that are supported by RT-middleware are accessible from DIMAN framework. The Player

/ Stage software provides perfect user and simulation interface for our research work. It provides also easy adoptable robot driver programming. Player supports a variety of hardware, and its modular architecture makes it easy to add support for new hardware and an active user/developer community contributes new drivers.

3.3 The Structure and Data Flow of the System

In the previous section the hierarchy among the software is introduced. The data flow of the proposed telemanipulation system is shown in Figure 8. Possibly this will help the reader understand the hierarchy more.

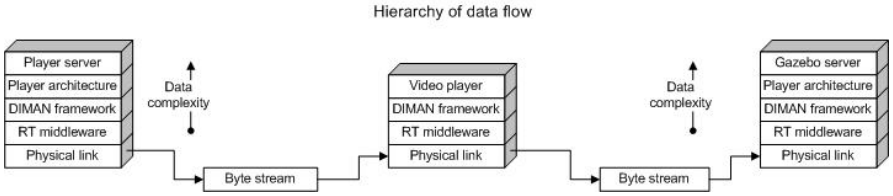

Figure 8
Figure Data flow in system

It is possible to run all the software on the same computer, but the large amount of computing tasks makes image processing very slow, so thus the process of the telemanipulation will be also very slow. This made our decision on using three separated computer, connected on local area network. The concept of the RT-middleware also supports our decision, what recommends using so called DIND (Distributed Intelligent Networked Device).

The cameras are connected to the Player Server program through their drivers, which are run on the first computer. The default drivers were modified to make the image processing, so these computers are dealing with the images, and are forwarding the data to the Player Client (Video player). The computers are connected locally at the master site, so they can send and receive large amount of data, which is a must at distributed image processing systems. This set up of components suites the concept of Intelligent Space technology and it can be used in very wide range of different fields, for example telemanipulation.

The Player Client (Video player) software displays the images of the cameras after the image processing and calculates the three-dimensional position of the master device. The third computer runs the second Player Server (Gazebo server) program at the slave site. The Player Client program can connect to the Gazebo Server via internet or local network. The bandwidth of the internet is not as large as the bandwidth of locally connected computers, but it is not a problem in this case. The images are not forwarded to the slave site, only the position data. Gazebo server program controls the robot with the help of the three-dimensional data. In our case a virtual robot is used, which is run on the robot simulator.

The whole process is shown in Figure 9.

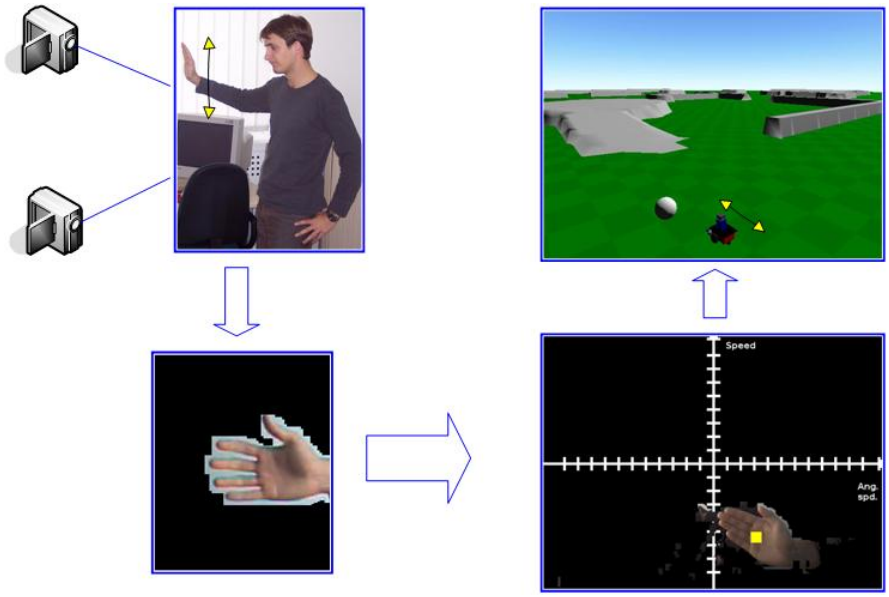

Figure 9
Process of the telemanipulation

Conclusions

The test system meets the quality requirements, which were set up against the system. It is possible to make manipulations in Intelligent Space. There is no need for physical contact between the manipulator and the slave side. Precision, which wasn't a goal, was also achieved; however it requires a little modification every time it is used. The speed of the image processing part is enough for basic telemanipulation, but with improvements it will be able to cope with higher frame rate.

The fact, that a new kind of telemanipulation is created, and also a new technology (RT-middleware) is introduced in the system, provides new industrial usability. In industrial telemanipulation it is hard to give force feedback, or it can not give force feedback at all. In this system the feedback of telemanipulation can be any kind of human sense (visual, oral or any kind of force). Whit this capabilities exceeds its competitors (such as a joystick).

Acknowledgement

The authors wish to thank the National Science Research Fund (OTKA K62836), Control Research Group and János Bolyai Research Scholarship of Hungarian Academy of Science for their financial support and the support stemming from the Intergovernmental S & T Cooperation Program.

References

[1] Noriaki ANDO, Takashi SUEHIRO, Kosei KITAGAKI, Tetsuo KOTOKU, Woo-Keun Yoon, "RT-Middleware: Distributed Component Middleware for RT (Robot Technology)", 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2005), pp. 3555-3560, 2005.08

[2] Matthias Kranz, Radu Bogdan Rusu, Alexis Maldonado, Michael Beetz, and Albrecht Schmidt, "A Player/Stage System for Context-Aware Intelligent Environments". Proceedings of the System Support for Ubiquitous Computing Workshop (UbiSys 2006), at the 8th Annual Conference on Ubiquitous Computing (Ubicomp 2006), Orange County, California, 2006. 09

[3] Peter Korondi, Hideki Hashimoto, ”INTELLIGENT SPACE, AS AN INTEGRATED INTELLIGENT SYSTEM”, **Keynote paper** of International Conference on Electrical Drives and Power Electronics, Proceedings pp. 24-31, 2003

[4] B. Reskśo, P. Baranyi, “Stereo Camera Alignment Based on Disparity Selective Cells in the Visual Cortex”, International Conference on Computational Cybernetics, pp. 285-290, 2005, Mauritius

[5] Gabor Sziebig, Andor Gaudia, Peter Korondi, Noriaki Ando, “Video Image Processing System for RT-Middleware”, 7th International Symposium of Hungarian Researchers on Computational Intelligence, pp. 461-472

[6] T. H. J. Collett, B. A. MacDonald, "Augmented Reality Visualisation for Player". IEEE International Conference on Robotics and Automation (ICRA 2006), pp. 3954-3959, Orlando, Florida, May 2006

[7] Akihiro IKEZOE, Hiroyuki NAKAMOTO, Masayuki NAGASE, “Development of RT-Middleware for Image Recognition Module”, SICE - ICASE International Joint Conference 2006, pp. 432-438, 2006. 10

[8] P. Korondi, P. Szemes, H. Hashimoto, ”Internet-based Telemanipulation” Chapter in R. Zurawski (Ed.): "The Industrial Information Technology Handbook", pp. 60-1-60-25, CRC Press, ISBN 0-8493-1985-4, June 2004