Published in final edited form as:
Int J Comput Assist Radiol Surg. 2010 May ; 5(3): 211–220. doi:10.1007/s11548-009-0388-9.

Open core control software for surgical robots

Jumpei Arata,
Nagoya Institute of Technology, Gokiso-cho, Showa-ku, Nagoya 466-8555, Japan

Hiroaki Kozuka,
Nagoya Institute of Technology, Gokiso-cho, Showa-ku, Nagoya 466-8555, Japan

Hyung Wook Kim,
Nagoya Institute of Technology, Gokiso-cho, Showa-ku, Nagoya 466-8555, Japan

Naoyuki Takesue,
Tokyo Metropolitan University, 6-6 Asahigaoka, Hino-shi, Tokyo 191-0065, Japan

B. Vladimirov,
Nagoya Institute of Technology, Gokiso-cho, Showa-ku, Nagoya 466-8555, Japan

Masamichi Sakaguchi,
Nagoya Institute of Technology, Gokiso-cho, Showa-ku, Nagoya 466-8555, Japan

Junichi Tokuda,
Harvard Medical School, Brigham and Women's Hospital, 75 Francis Street, Boston, MA 02115, USA

Nobuhiko Hata,
Harvard Medical School, Brigham and Women's Hospital, 75 Francis Street, Boston, MA 02115, USA

Kiyoyuki Chinzei, and
National Institute Advanced Industrial Science and Technology, AIST Tsukuba East, 1-2-1 Namiki, Tsukuba, Ibaraki 305-8564, Japan

Hideo Fujimoto
Nagoya Institute of Technology, Gokiso-cho, Showa-ku, Nagoya 466-8555, Japan

Jumpei Arata: jumpei@nitech.ac.jp

Abstract

Object—In these days, patients and doctors in operation room are surrounded by many medical devices as resulting from recent advancement of medical technology. However, these cutting-edge medical devices are working independently and not collaborating with each other, even though the collaborations between these devices such as navigation systems and medical imaging devices are becoming very important for accomplishing complex surgical tasks (such as a tumor removal procedure while checking the tumor location in neurosurgery). On the other hand, several surgical robots have been commercialized, and are becoming common. However, these surgical robots are not open for collaborations with external medical devices in these days. A cutting-edge “intelligent surgical robot” will be possible in collaborating with surgical robots, various kinds of sensors, navigation system and so on. On the other hand, most of the academic software developments for surgical robots are “home-made” in their research institutions and not open to the public. Therefore, open source control software for surgical robots can be beneficial in this field. From

Correspondence to: Jumpei Arata, jumpei@nitech.ac.jp.

these perspectives, we developed Open Core Control software for surgical robots to overcome these challenges.

Materials and methods—In general, control softwares have hardware dependencies based on actuators, sensors and various kinds of internal devices. Therefore, these control softwares cannot be used on different types of robots without modifications. However, the structure of the Open Core Control software can be reused for various types of robots by abstracting hardware dependent parts. In addition, network connectivity is crucial for collaboration between advanced medical devices. The OpenIGTLink is adopted in Interface class which plays a role to communicate with external medical devices. At the same time, it is essential to maintain the stable operation within the asynchronous data transactions through network. In the Open Core Control software, several techniques for this purpose were introduced. Virtual fixture is well known technique as a "force guide" for supporting operators to perform precise manipulation by using a master–slave robot. The virtual fixture for precise and safety surgery was implemented on the system to demonstrate an idea of high-level collaboration between a surgical robot and a navigation system. The extension of virtual fixture is not a part of the Open Core Control system, however, the function such as virtual fixture cannot be realized without a tight collaboration between cutting-edge medical devices. By using the virtual fixture, operators can pre-define an accessible area on the navigation system, and the area information can be transferred to the robot. In this manner, the surgical console generates the reflection force when the operator tries to get out from the pre-defined accessible area during surgery.

Results—The Open Core Control software was implemented on a surgical master–slave robot and stable operation was observed in a motion test. The tip of the surgical robot was displayed on a navigation system by connecting the surgical robot with a 3D position sensor through the OpenIGTLink. The accessible area was pre-defined before the operation, and the virtual fixture was displayed as a "force guide" on the surgical console. In addition, the system showed stable performance in a duration test with network disturbance.

Conclusion—In this paper, a design of the Open Core Control software for surgical robots and the implementation of virtual fixture were described. The Open Core Control software was implemented on a surgical robot system and showed stable performance in high-level collaboration works. The Open Core Control software is developed to be a widely used platform of surgical robots. Safety issues are essential for control software of these complex medical devices. It is important to follow the global specifications such as a FDA requirement "General Principles of Software Validation" or IEC62304. For following these regulations, it is important to develop a self-test environment. Therefore, a test environment is now under development to test various interference in operation room such as a noise of electric knife by considering safety and test environment regulations such as ISO13849 and IEC60508. The Open Core Control software is currently being developed software in open-source manner and available on the Internet. A communization of software interface is becoming a major trend in this field. Based on this perspective, the Open Core Control software can be expected to bring contributions in this field.

Keywords

Surgical robot; Open source software; Virtual fixture

Purpose

In recent advanced neurosurgery and other surgeries, it is becoming common to collaborate with navigation systems to deal with a lot of surgical information such as tool/tumor location and pre-operative medical images. The improvements of the outcomes of these advanced surgeries (such as a tumor removal rate) are reported in many studies. Therefore,

the collaborations between such advanced medical devices can be a key technology in future operations.

In these days, there are a lot of medical devices in operation room. In addition, several surgical master–slave robots have been commercialized, and are becoming common. However, most of these advanced medical devices are independently working and not working in collaborations with each other.

By introducing surgical robots, it is possible to perform more precise surgery. Moreover, collaborations with surgical robots and these advanced medical devices can be a further strong advancement in this field. The da Vinci surgical system (Intuitive Surgical Inc.) [1] is the most widely used surgical master–slave robot. A communication interface of the da Vinci surgical system has been announced recently [2], however, the data stream of the API is still limited and the whole system is a “black box”. Therefore, a high network connective control software of surgical robots, which can be developed in a open source manner will be beneficial in this field.

In this paper, The Open Core Control software for surgical robots is presented. The software has following features:

- Open source
- High network connectivity
- Robustness
- Reusability on different types of surgical robots
- Safety design as a real-time networked software

The Open Core Control software consists of a hierarchical structure including a kinematic model, a position/force control system, and network extensions. The aims of the study are the safety enhancement and further developments of surgical robots. In addition, by realizing a software such as the Open Core Control software, it is possible to realize a high-level collaboration between a surgical robot and external medical devices through network.

In this paper, virtual fixture is implemented to demonstrate an idea of collaborations between surgical robot and navigation system. Virtual fixture is a well-known technique for supporting operators to perform a precise manipulation by using master–slave robot as a “force guide” [3–7]. In the prototype implementation, operators can pre-define an accessible area on a navigation system, and the area information can be transferred to the robot. In this manner, the master robot generates a reflection force when the operator tries to get out from the pre-defined accessible area during surgery. Therefore, a function such as virtual fixture is one of the highest-level collaborations with surgical robot system and navigation system. The extension of virtual fixture is not a part of the Open Core Control system, however, the function such as virtual fixture cannot be realized without a tight collaboration between cutting-edge medical devices.

As previous researches on robot control software, Robot Technology (RT)-middleware [8] was proposed by AIST, Japan. The RT-middleware could establish various networked robot control systems by using distributed component technology. In medical fields, MRC-II control software for surgical robots was proposed by Taylor et al. [9]. Kragic et al. [10] have proposed a structure and analytical models of a human–machine collaborative systems (HMCS) including a method of guidance virtual fixture for microsurgical applications by using an admittance controlled robot. A communication software for system integration of the surgical devices using CORBA was studied [11]. As studies on virtual fixture, Abbott et

al. [4] proposed tele-operation algorithm using a forbidden-region virtual fixture (FRVF) and compared nine FRVFs on each of four common tele-manipulator control architectures. Aarno et al. [5] studied adaptive virtual fixture to cope with a degraded performance in cases of unexpected obstacles or incorrect fixture models. Kikuuwe et al. [6] studied control algorithms of a new class of virtual fixtures that is based on simulated plasticity. Li et al. [7] proposed a new method to generate virtual fixtures for surgical robot control, which provides sophisticated ways to assist the surgeon.

Methods

Software architecture

In general, control softwares have hardware dependencies based on actuators, sensors and various kinds of internal devices. Therefore, these control softwares cannot be used on different types of robots without fundamental modifications. However, the structure of the Open Core Control software can be reused in different types of surgical robots by abstracting hardware dependent parts. The structure of the Open Core Control software is shown in Fig.1. The software consists of a hierarchical structure including a kinematic model, a position/force control system, and network extensions.

In the software architecture, a robot class is defined. The robot class consists of an interface manager class, a frame class, (a) joint class(es) and (a) driver class(es). The interface manager class deals with all network data transactions with external devices. In the interface manager class, multiple interface classes can be declared and used depending on the number of communication channels with external devices. This part is abstracted from the hardware dependent part and completely reusable in different types of robots. On the other hand, the frame class, the joint class(es) and the driver class(es) are hardware dependent parts. These classes deal with the kinematic model and I/O of robot and must be partially modified depending on the robot structure. Therefore, by the proposed architecture, it is possible to separate the hardware dependent part and non-dependent part in terms of the reusability. The software is implemented by using C++ and tested on VxWorks 5.5.1 and ART-Linux-2.4.36. The source code of the Open Core Control software is available on the Internet [19].

Network connectivity

Network connectivity is crucial for collaborations among advanced medical devices. In previous studies [8,11], distributed object models such as CORBA (Common Object Request Broker Architecture) were introduced. However, it is essential to keep the simplicity of implementation to be widely used platform. Moreover, the overhead of data transactions on such distributed object models can be an obstacle of implementation. Therefore, OpenIGTLink is adopted in the interface class which plays a role to communicate with external medical devices [12]. The OpenIGT-Link structure is shown in Fig.2. The OpenIGTLink is a protocol which is intended to communicate among medical devices such as trackers, robots, and scanners in a simple manner.

By using the OpenIGTLink, sequences of bytes are transmitted through the application layer of the internet protocol. For simple implementations and ensuring compatibilities, all necessary information for decoding of communication data is included in the header field of the OpenIGTLink protocol. In the Open Core Control software, the OpenIGTLink is implemented by using TCP/IP and UDP/IP. The data transaction threads are independently implemented in the main control routine. Therefore, it is possible to communicate with medical devices which have different communication frequencies in real-time.

Virtual fixture

Virtual fixture is defined as “abstract sensory information overlaid on top of reflected sensory feedback from a remote environment” [3]. For example, when asked to draw a straight line in the real world, human performance can be greatly enhanced by using a simple tool such as a ruler. A ruler is essentially a “perceptual overlay” designed to enhance line drawing performance. The overlaid sensory information represents a single rigid surface to be perceived visually by the user. By overlaying this sensory information on top of the workspace, the user has reduced the mental and physical demands of the straight line drawing task and has thus enhances task performance.

In surgical applications, the virtual fixture can be a key technology for avoiding unexpected damage on fragile tissues during the surgical procedure. By defining an accessible area before surgery, it is possible to keep the surgical tool out of the fragile area such as a nerve system.

From these perspectives, the virtual fixture was implemented as an extension of the Open Core Control software. The extension of virtual fixture is not a part of the Open Core Control software, however, the function such as virtual fixture cannot be realized without a tight collaboration between cutting-edge medical devices. Therefore, the virtual fixture was implemented as one of the most suitable applications for showing a performance of collaboration of these devices.

The editor interface of accessible area was implemented on the 3D Slicer Version 3.21.0 (Fig.3). An operator specifies the accessible area considering the figure and position of fragile tissues such as vessels and nerve systems. The location and figure information of the accessible area, then, are transmitted to the controller of master robot by using the OpenIGTLink protocol. During the operation, in case the slave robot approaches the boundary of the accessible area, the master robot generates interaction force to prohibit the slave robot from approaching to the fragile tissues. Therefore, the operator can safely access the pre-planned surgical area. In this implementation, the accessible area was defined as a simple spherical model by a spring model. The stiffness of the sphere is defined in the protocol and it can be modified (Fig. 4). The reflection force in simple sphere model can be given as follows:

$$\mathbf{D}=\mathbf{P_s}-\mathbf{P_e}$$

$$\begin{cases} f_m=k(\|\mathbf{D}\|-r) & (\|\mathbf{D}\|-r>0) \\ f_m=0 & (\|\mathbf{D}\|-r\le 0) \end{cases}$$

$$\mathbf{F}=f_m\frac{\mathbf{D}}{\|\mathbf{D}\|}$$

where

$\mathbf{P_e}$: Position of the end-effecter

$\mathbf{P}_s$: Position of the center of sphere

$\mathbf{D}$: Distance between the end-effecter and the center of sphere

$\mathbf{F}$: Force to be displayed

r: Radius of the sphere

k: Stiffness of the sphere

f_m: Intermediate parameter of the force calculation

In this case, proxy-base method was introduced for the calculation of entry point of the virtual fixture. The method can be extended for the accessible area of multiple spheres for the complex surgical area definition. For example, implicit function can be utilized for dealing with the multiple spheres. The implementation of the virtual sphere with multiple spheres is a further study of this work frame.

Safety design

As a real-time networked software, it is essential to maintain the stable operation within the asynchronous data transactions through network. In the Open Core Control software, several techniques for this purpose were introduced.

The Open Core Control software is implemented by using watch dog timer and message queue for stable operations. The watch dog timer monitors control transaction time (sampling time). In case the monitored task is not finished in the predefined sampling time, a control interrupt occurs for protecting the control transactions. It is important to maintain the control frequency properly in the critical systems such as surgical robots. In the developed software, the watchdog timer monitors the sampling time to guarantee a stability of the system. The message queue is a mechanism for asynchronous communications by using queue, which is a list structure of first in first out (FIFO). In the developed software, a message queue is used to communicate between the interface class and a main routine of real-time transaction. In addition, the records of robot motion can be stored or given through network in real-time as a data log of the operations. For the network transmission, CRC, Body size and time stamp are described in every packet header in the OpenIGTLink. These header data was checked in every data arrival for robust data communications.

Results

To test the performance of the Open Core Control software, the software was implemented on a prototype of surgical master–slave robot (Fig. 5). The experiments were conducted by a collaborating motion test with a surgical master–slave robot, the Optotrak (Northern Digital Inc.) [13] and the 3D Slicer [14]. In addition, the virtual fixture was implemented to demonstrate a high-level collaboration between a surgical master–slave robot and a navigation system.

Figure 5 shows the structure of the prototype system. Both master and slave robots were controlled by using the Open Core Control software which were implemented on VxWorks 5.5.1. All data was transmitted by using the OpenIGTLink (illustrated as allows in Fig. 6). For the robot control, the position data was sent from the master robot to the slave robot in 500Hz. The position of slave robot was displayed on a test image in the 3D Slicer by using both position of the Optotrak (robot base) and the slave robot (tip position). By combining these two position data, it is possible to take advantage of precision, because robot position data is generally more precise than optical 3D position sensors, and an occlusion problem does not occur on the robot position data. In addition, the boundary position data of the accessible area is sent from the 3D Slicer to the master robot for virtual fixture.

Slave robot

Figure 7 shows the overview of the slave robot. In the slave robot, a parallel mechanism was introduced for its remote center mechanism, which has significant features including low inertia and high stiffness. The robot has 3 DOF in total, two rotational motions around the remote center and an axial translational motion. The robot was designed as a motion base for various kinds of minimally invasive surgery [15]. Various types of surgical instruments can

be attached on the tip of the slave robot. In the prototype system, a dummy surgical tool (a simple bar) was attached for the motion test.

Master robot

The overview of the master robot is shown in Fig. 8. The master robot is a haptic device using parallel link mechanism, which consists of a 3 DOF translation mechanism, a 3 DOF rotation mechanism and a 1 DOF grasping motion [16]. In the motion test, the 3 DOF translation mechanism was used by applying its motions to the 3 DOF motions of slave robot. The 3 DOF translation mechanism was realized by a redundant parallel mechanism, which realizes 3 DOF translational motions by 4 actuators fixed on the base plate. By actuating these 4 motors, it is possible to display an interaction force to the operator.

Controller implementation

In the prototype system, the controllers of the robots were implemented by using Compact PCI form factor boards (Tables 1, 2). VxWorks 5.5.1 was installed on CPU boards on both master and slave robot controllers. As it is mentioned above, the modifications of hardware dependent part are minimized in Open Core Control software. The software was implemented on both master and slave controllers in different I/O configurations.

Structure of the prototype system

Figure 9 shows the data transaction of whole system. In the prototype system, navigation system and master–slave surgical robots can be tightly collaborated. The system consists of the master robot, the slave robot, the 3D Slicer and the Optotrak. The network of these devices was established by connecting all devices in a network hub (100Mbps). The OpenIGTLink within UDP/IP was used between master–slave robots, and other network connections were established by the OpenIGTLink within TCP/IP.

At first, the virtual fixture information can be edited and generated in the 3D Slicer, and it is transmitted to the master robot by using the OpenIGTLink within TCP/IP. In the master robot controller, the virtual fixture information is received in a asynchronous interface loop, and it is send to a robot control routine through a message queue. The message queue is an internal data transmission process, which enables the asynchronous data exchange among different tasks in the operating system. The robot control routine in the master robot obtains its position data in every routine of 1kHz by accessing the driver class. In the same time, the robot control routine generates the reflective force by using the virtual fixture information and the kinematic calculations. Then the reflective force can be properly displayed on the master robot according to the virtual fixture information. At last, the position data of the master robot is sent to another message queue for sending to the slave robot in 500 Hz. A watchdog timer (WDTimer) was introduced for observing these procedures in the robot control routine. If these procedures have unexpected delay, the WDTimer sends an alert message to the system operator.

In the slave robot controller, the position command is received in an asynchronous interface loop, and it is sent to a robot control routine through a message queue. Then the robot control routine generates the slave motion by the inverse kinematics of the slave robot by the driver class in 1 kHz. The WDTimer was introduced on the slave robot controller as well as the master robot. Finally, the slave tip position data is transmitted to the 3D Slicer in 10 Hz. The position data of the slave robot consists of the tip position and the base position. The tip position is given by the precise robot internal data, and the base position is obtained by the Optotrak for a patient-robot registration. The base position data is transmitted through the network from the Optotrak to the 3D Slicer by using the OpenIGTLink within TCP/IP in 10 Hz as well.

Preliminary motion test

A preliminary motion test was performed by introducing the developed software to the prototype system, and evaluated in a quantitative manner. The position of the slave robot was updated in the 3D Slicer. The pre-defined virtual fixture (a sphere with diameter of 200 mm) was displayed as a force guide on the master robot. The stiffness of the sphere was set as 1 Nmm in this test. Figure 9 shows the trajectory of the master robot during the motion test. The motion scale between the master robot and the slave robot was set as 10:1. The sphere illustrated in the graph represents the pre-defined accessible area. From this result, it is observed that the robot position remained in the accessible area, and the spherical boundary of the accessible area was properly displayed as a virtual force wall. The result showed that the most of trajectory was kept in the sphere. The data log revealed that the maximum of 10 N force was displayed during the operation. In addition, the system showed stable performance in a duration test of several hours. Thus, from the motion test, the Open Core Control software showed the expected performance such as collaborated motions and high robustness (Fig. 10).

Stability test in network disturbance

An evaluation test was conducted for testing the stability of the developed software.

The test was conducted on the prototype system, the configuration was same as the preliminary motion test. In this test, the packet loss was randomly generated on the position data transmission between the master robot and the slave robot. Two consecutive packet losses were configured with the rate of 2% of the 500 Hz data transmission between the master robot and the slave robot. Thus, the packets were randomly lost 10 times in every 1 s. In case more packet loss is observed in the same time (such as three consecutive packet loss), it can be considered as a serious network problem. In this case, the system immediately stops the operations and sends an alert message to the system operator.

Figure 11 shows the interval of packet arrival on the slave robot. When two consecutive packets are lost, the interval of the packet arrival is increased from 2 to 6 ms. In the test, the position data was transmitted from the master robot to the slave robot as: *X*-Sine wave (0.1 Hz frequency, 50 mm of amplitude), *Y*-Cosine wave (0.1 Hz of frequency, 50 mm of amplitude) and *Z*-20 mm (fixed value). The motion scale between master and slave was set as 10:1, and the experiment was conducted during 12 h for testing the robustness of the system. Figure 12 shows the trajectories of the input and output motions of the slave robot. The trajectories of the input and output motions are in accordance with the average error of 0.094 and 0.097 mm in *X* and *Y*, respectively. These errors were stable and major variation was not observed during the test. In addition, no unstable behavior of the system was observed during the experiment. From these results, the robustness of the software in terms of the network stability was showed.

Conclusions

In this paper, a design of the Open Core Control software for surgical robots with high connectivity was presented. In collaborating motion tests, the prototype system showed stable performance.

- The Open Core Control software was designed in open source manner.
- Reusability was taken into account in terms of further wide use of the software
- Several techniques such as watch dog timer and message queue were introduced for stable operations

- Records of robot motion can be stored in local or through network.
- OpenIGTLink was implemented for network connectivity.
- A prototype system with a surgical master–slave robot, the Optotrak and the 3D Slicer was implemented.
- Virtual fixture was implemented on the prototype system to demonstrate a high-level collaboration between the advanced medical devices.
- A collaborating motion test quantitatively showed a stable performance of the system.
- A stability test also showed a stable performance of the system in terms of the network disturbance.

Safety issues are essential for control software of these complex medical devices. It is important to follow the global specifications such as a FDA requirement “General Principles of Software Validation” or IEC62304 for the further practical realizations [17,18]. For following these regulations, a test environment is now under development to test various interferences in operation room such as a noise of electric knife, in considering the safety and test environment regulations such as ISO13849 and IEC60508. In addition, documentation of the source code for wide reusability and further network test (such as multiple device connection) are currently on-going. In vivo/vitro tests of effectiveness of virtual fixture in surgical applications are also on-going process. The software was tested on two different prototype systems including the presented system in this paper, up to the present time. The Open Core Control software is currently being developed software in open-source manner and available on the Internet [19].

A communization of software interface is becoming a major trend in this field. Based on this perspective, The Open Core Control software can be expected to bring contributions in this field.

Acknowledgments

The authors gratefully acknowledge the helpful comments of anonymous reviewers. This study was funded by NEDO Intelligent Surgical Instruments Project. This publication was made possible by Grant Number 5U41RR019703, 5P01CA067165, 5U54EB005149 from NIH.

References

1. Intiutive Surgical Inc. 2009. http://www.intuitivesurgical.com/
2. DiMaio, SP.; Hasser, CJ. The da Vinci Research Interface. Proceedings of international conference on medical image computing and computer assisted intervention 2008: Workshop-S5 systems and architectures for computer assisted interventions; 2008.
3. Rosenberg, L. Virtual fixtures: perceptual tools for telerobotic manipulation. Proceedings of IEEE virtual reality annual international symposium; 1993. p. 76-82.
4. Abbott, JJ.; Okamura, AM. Virtual fixture architectures for telemanipulation. Proc IEEE Int Conf Robot Autom 2003; 2003. p. 2798-2805.
5. Aarno, D.; Ekvall, S.; Kragic, D. Adaptive virtual fixtures for machine-assisted teleoperation tasks. Proc IEEE Int Conf Robot Autom 2005; 2005. p. 1151-1156.
6. Kikuuwe R, Takesue N, Fujimoto H. A control framework to generate nonenergy-storing virtual fixtures: use of simulated plasticity. IEEE Trans Robot 2008;24(4):781–793.
7. Li, M.; Kapoor, A.; Taylor, RH. A constrained optimization approach to virtual fixtures. Proc IEEE Int Conf Intell Robots Syst 2005; 2005. p. 2904-2929.

8. Ando, N.; Suehiro, T.; Kitagaki, K.; Kotoku, T. RT (robot technology)-middleware towards component based networked robot systems development. Proc Int Conf Ubiquitous Robots Ambient Intell 2005; 2005. p. 101-104.

9. MRC-II. 2009. http://www.cisst.org/~pkaz/mrc/Overview.htm

10. Kragic D, Marayong P, Li M, Okamura AM, Hager GD. Human–machine collaborative systems for microsurgical applications. Int J Robot Res 2005;24(9):731–741.

11. Aoki, E.; Suzuki, T.; Kobayashi, E., et al. Modular design of master–slave surgical robotic system with reliable real-time control performance. Proc IEEE RAS-EMBS Int Conf Biomed Robot Biomechatronics 2006; 2006. p. F119

12. Tokuda, J.; Ibanez, L.; Csoma, C., et al. Software and hardware integration strategy for image guided therapy (IGT) using OpenIGTLink. Proceedings of international conference on medical image computing and computer assisted intervention 2008: Workshop-S5 systems and architectures for computer assisted interventions; 2008.

13. Northern Digital Inc. 2009. http://www.ndigital.com/

14. 3D Slicer. 2009. http://www.slicer.org/

15. Arata, J.; Ikemoto, J.; Sakaguchi, M.; Fujimoto, H. Development of the surgical motion base system using a parallel link mechanism. Proceedings of Asian conference on computer aided surgery; 2007. p. 110076

16. Arata, J.; Kondo, H.; Sakaguchi, M.; Fujimoto, H. A haptic device DELTA-4: kinematics and its analysis. Proc World Haptics 2009; 2009. p. 452-457.

17. General Principles of Software Validation. 2009. http://www.fda.gov/cdrh/comp/guidance/938.html

18. Medical device software—software life cycle processes. IEC62304. 2006

19. Open Core Control software. 2009. http://svn.na-mic.org/NAMICSandBox/trunk/IntelligentSI/Control/

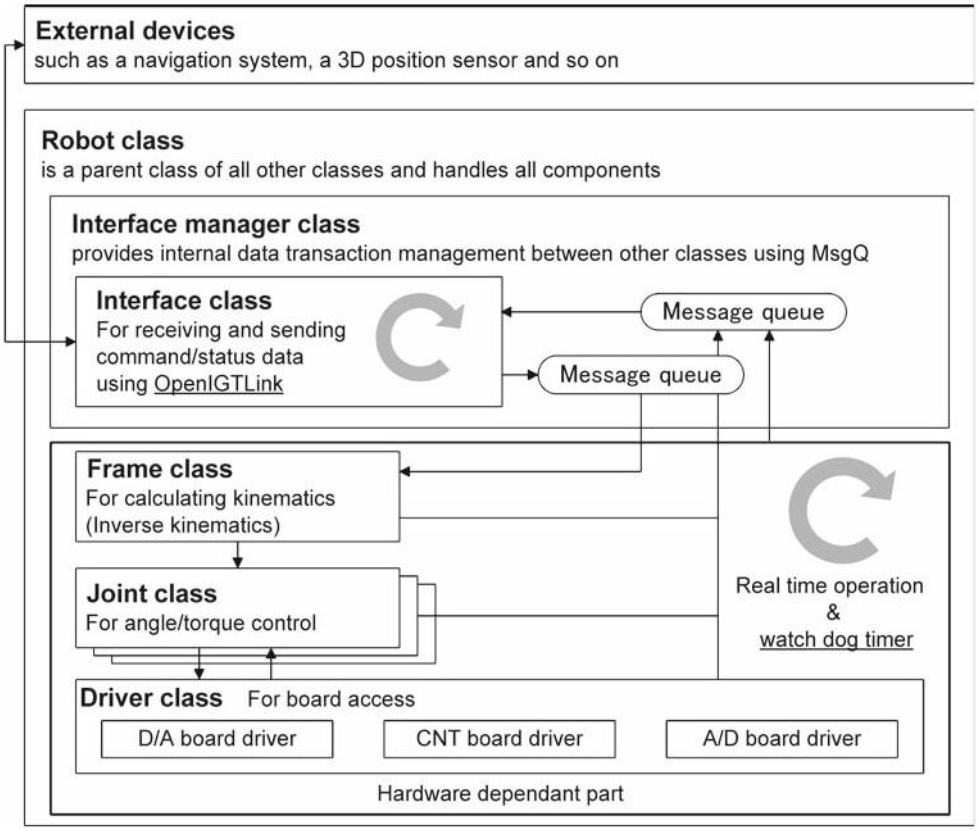

Fig. 1.
Structure of Open Core Control software

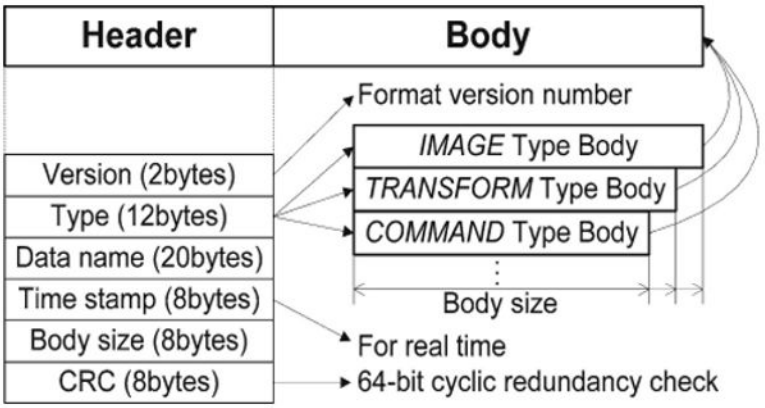

Fig. 2.
OpenIGTLink structure

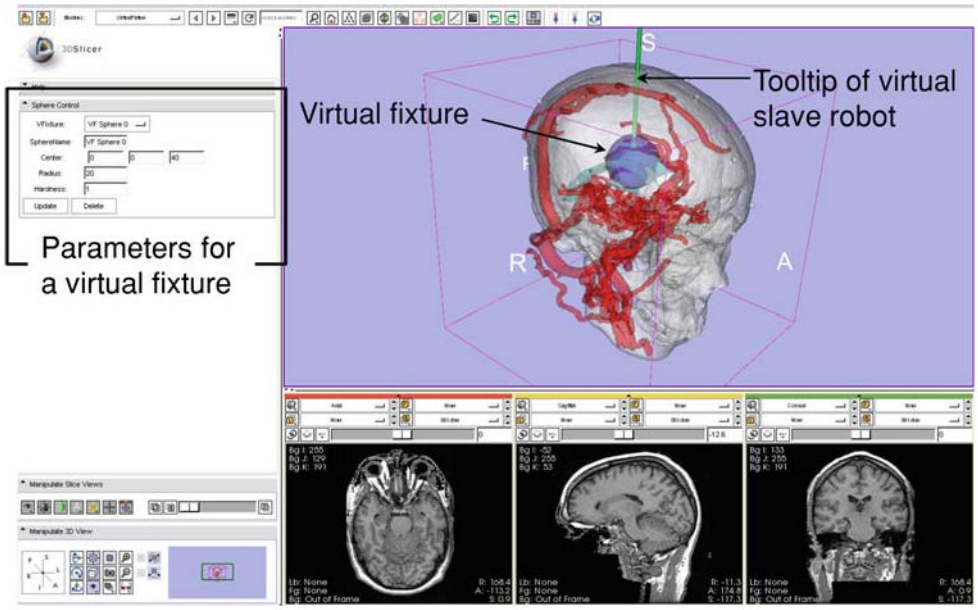

Fig. 3.
Editor interface of virtual fixture on 3D Slicer

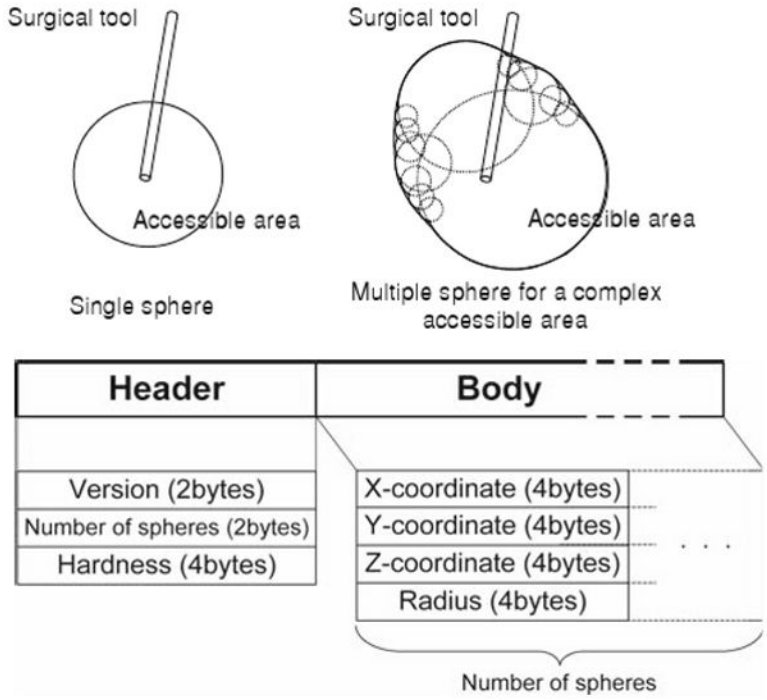

Fig. 4.
Data structure of virtual fixture

Fig. 5.
Overview of the prototype system

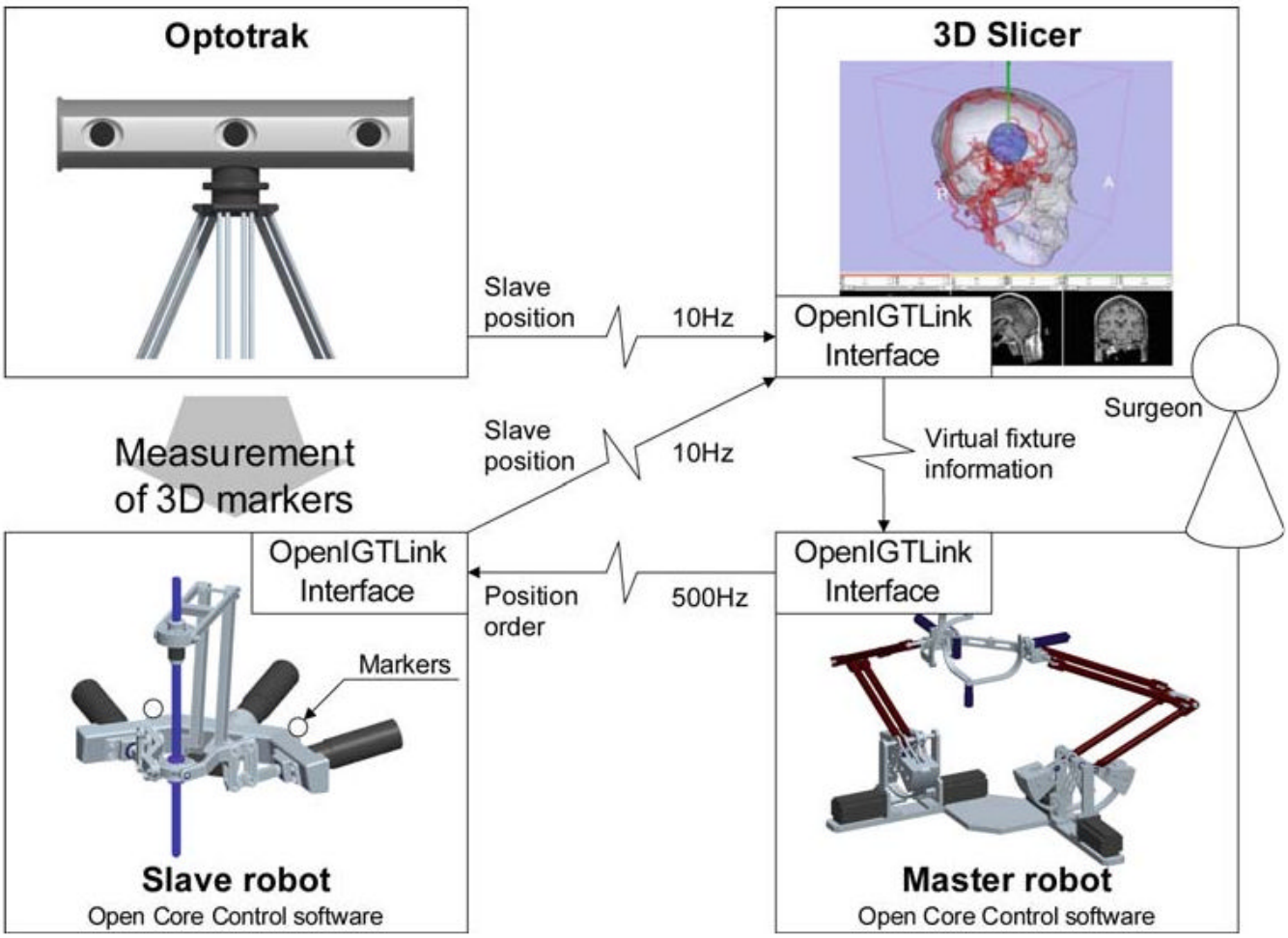

Fig. 6.
Network configuration

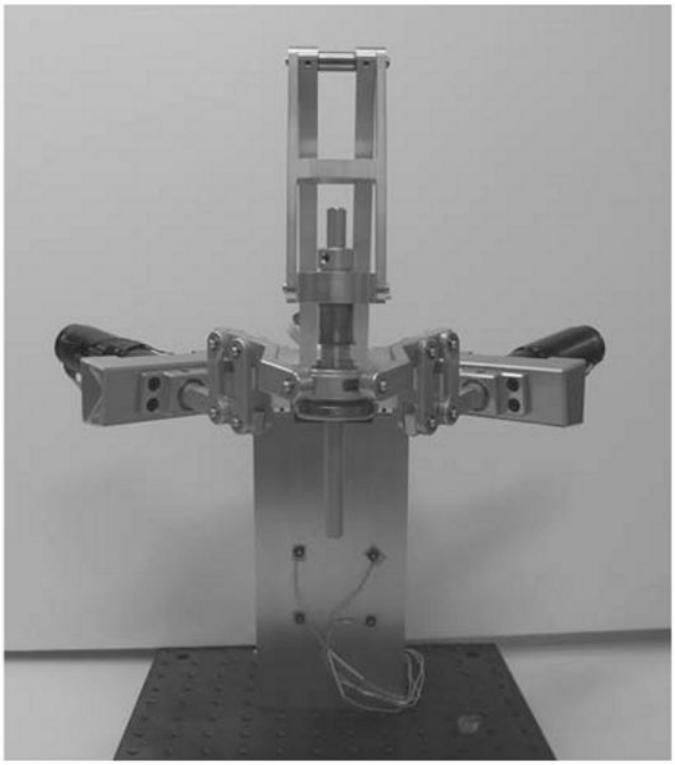

Fig. 7.
Slave robot

Fig. 8.
Master robot

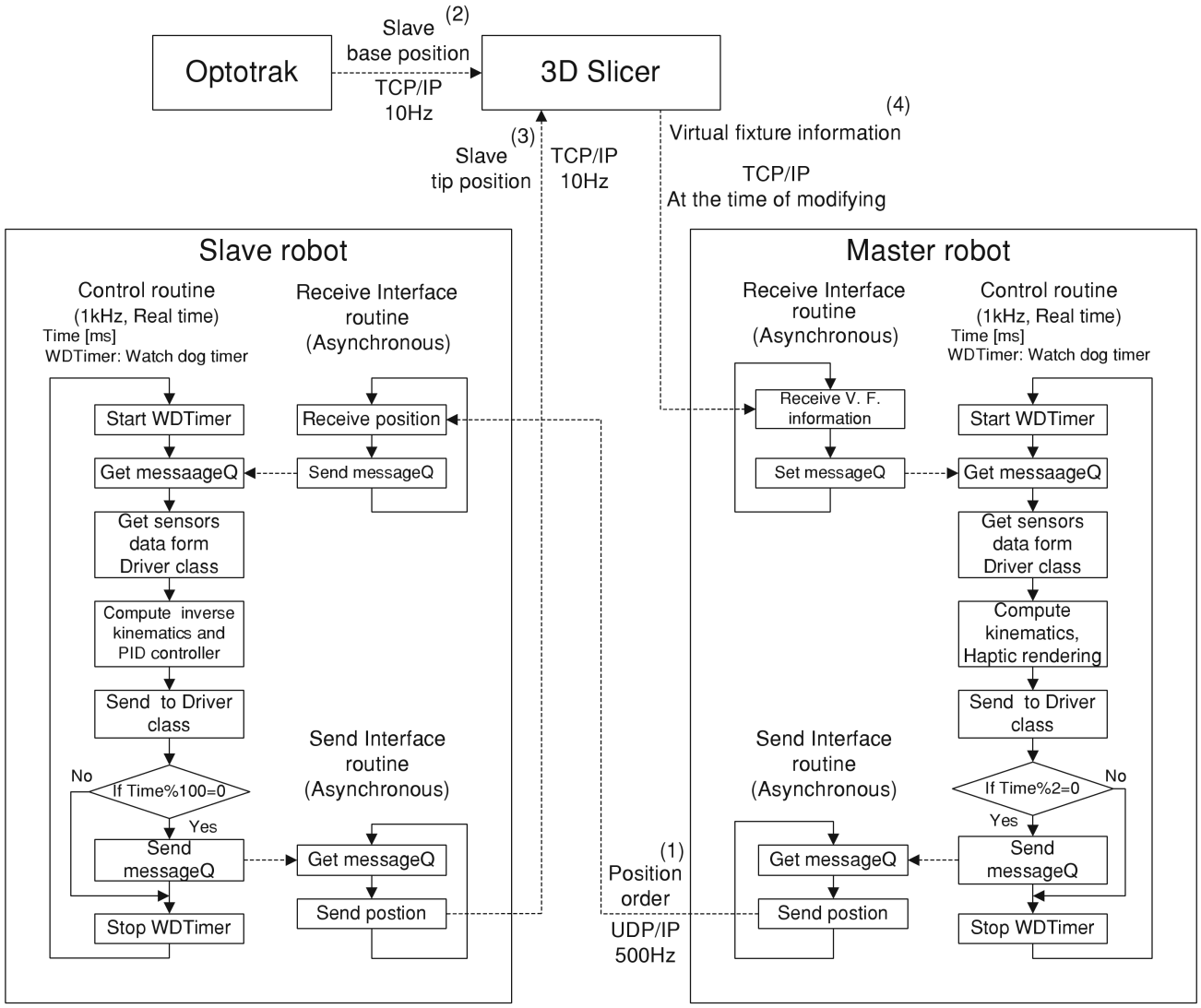

Fig. 9.
Structure of the prototype system

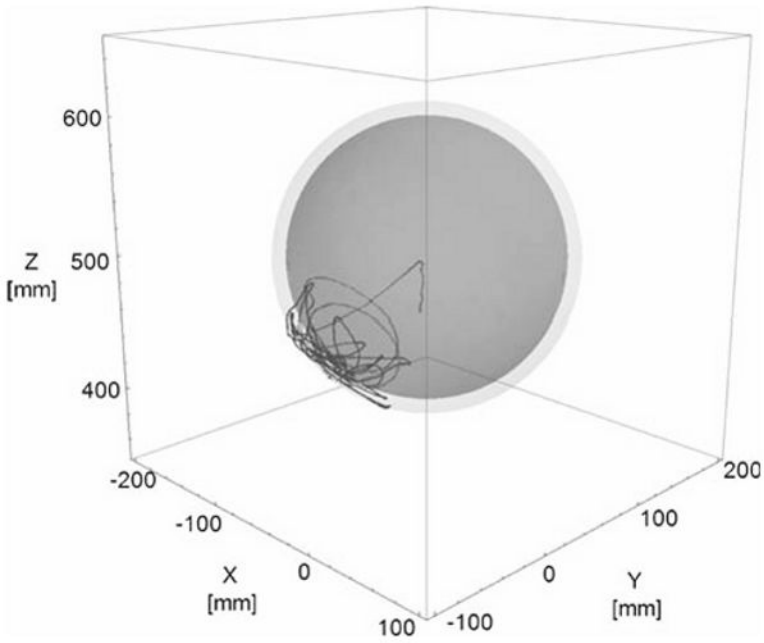

Fig. 10.
Trajectory of the master robot

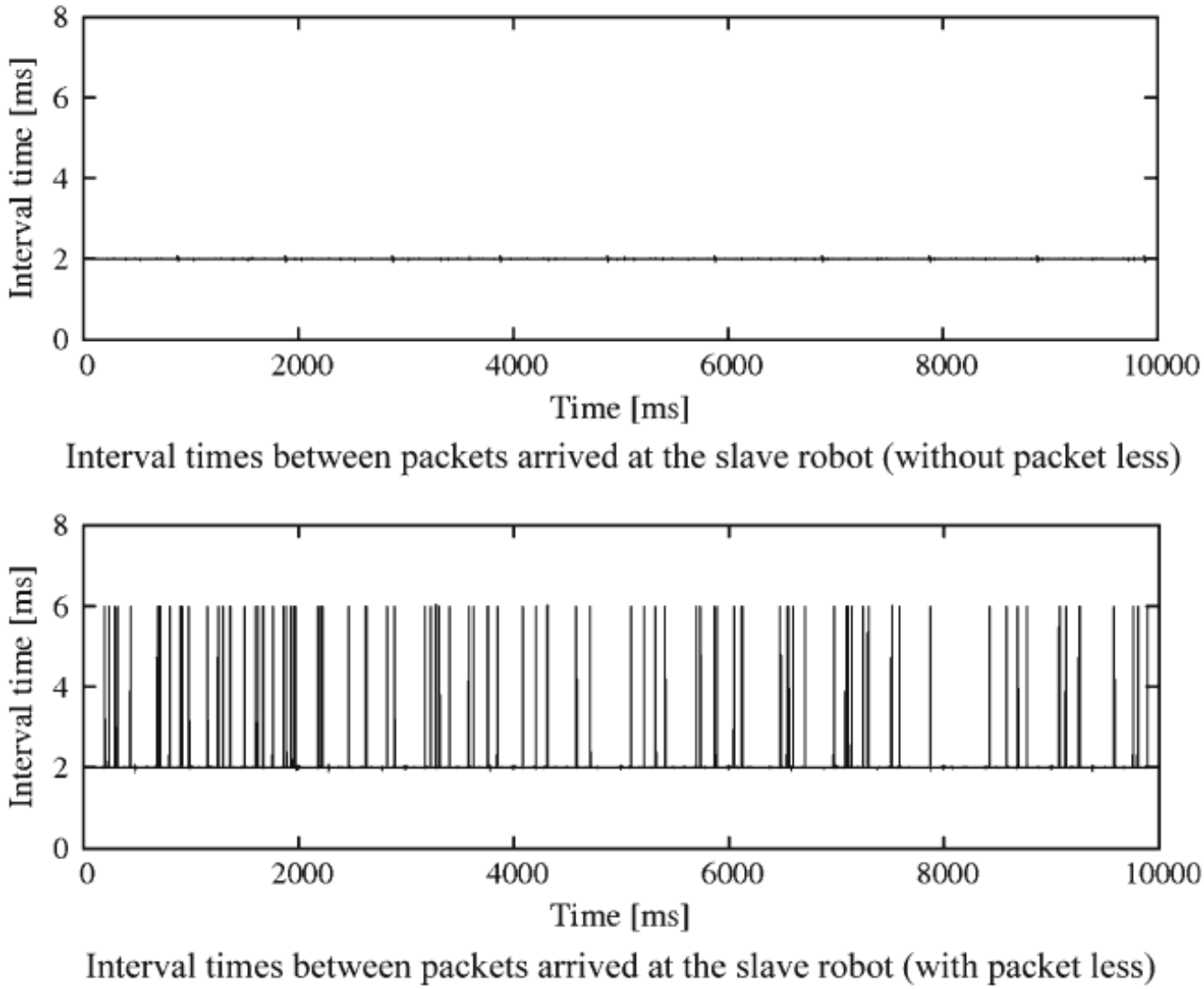

Fig. 11.
Effect of network disturbance

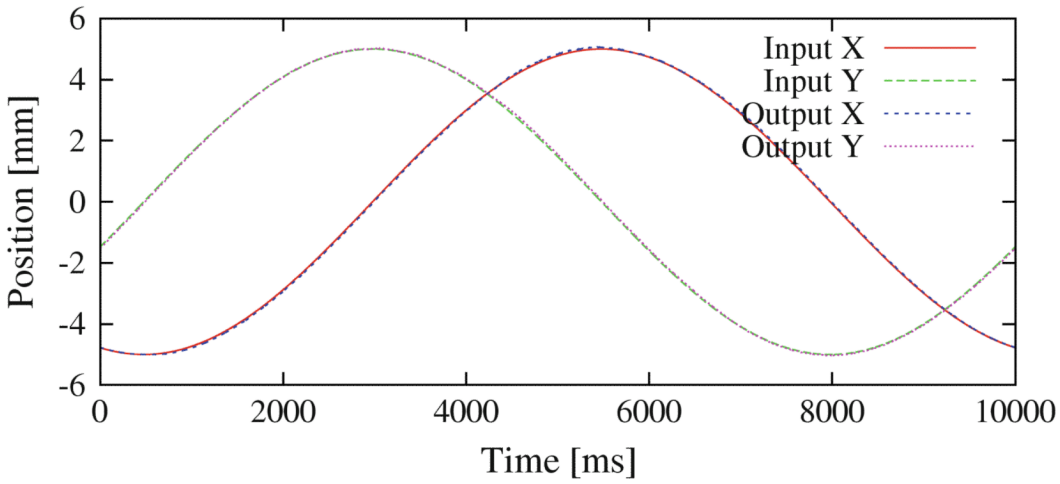

Fig. 12.
Experimental result of stability test

Table 1

Specifications of controller

CPU	MOTOROLA, CPCI-6020-500, 500MHz MPC7410 processor, CompactPCI 64Bit/33/66 MHz/3.3V
Chassis	AVAL DATA, ACP-900, CompactPCIbus IEEE1101.10 compatible PICMG REM2.1 compatible backboard, 8 slot (6U)
Counter	(Master) Interface, CTP-6205, 24bit Counter, 8Ch, differential input, CompactPCI, 32bit/33MHz/3.3V (Slave) AVAL DATA, ACP-420, 28Bit Counter, 4Ch, differential/open corrernter, CompactPCI 32bit/33MHz/5V
D/A	(Master) Interface, CTP-3351, Resolution 12Bit, Range ±10V, 8Ch, CompactPCI 32bit/33MHz/3.3V (Slave) AVAL DATA, ACP-560, Resolution 16Bit, Range ±10V, 4Ch, CompactPCI 32bit/33MHz/5V

Table 2

Network transmissions

No.	Data	Frequency	Used for
1	IGTL Header (52bytes) + Position command (24bytes)	500Hz	Robot control
2	IGTL Header (52bytes) + Current base position (24bytes)	10Hz	Navigation
3	IGTL Header (52bytes) + Current tip position (24bytes)	10Hz	Navigation
4	IGTL Header (52bytes) + Virtual Fixture (8 +16xn bytes)	Event driven	Virtual fixture