A Novel Liver Surgical Navigation System Using Polyhedrons with STL-Format

Satoshi Numata(✉), Daiki Yano, Masanao Koeda, Katsuhiko Onishi, Kaoru Watanabe, Hiroshi Noborio, and Hirotaka Uoi

Osaka Electro-Communication University, Shijonawate, Osaka 5750063, Japan
numata@osakac.ac.jp

Abstract. We have developed the liver surgical navigation system composed of three subsystems: the liver position and orientation estimator, the surgical knife position estimator and the liver surgical navigator. These subsystems work separately for estimating the liver position and the knife position, and the liver surgical navigation system will use those positions to navigate surgeons accurately. The liver position is estimated by comparing two depth images; an image come from a depth camera targeting the liver and an image rendered by OpenGL using the Polyhedrons with STL-format data previously scanned from a patient. The knife position is estimated by tracking markers put at the top of the knife. The surgical navigation system holds precise data such as positions of vessels or the surgical steps, and show appropriate navigation data to the surgeons. In this paper, we describe the overview of this system and how we integrated these subsystems into the liver surgery supporting system.

Keywords: Liver surgery support · Navigation · Inter-process communication

1 Introduction

Liver surgery requires extremely delicate treatment for several types of blood vessels such as arteries or veins while they are living inside the soft organ. Surgeons often use X-ray imaging, computed tomography (CT) scanning or magnetic resonance imaging (MRI) previously to the surgery for determining precise positions of those vessels in the liver. However, those vessels are naturally located sterically in our three dimensional world and surgeons face to difficulties in matching those imaging and scanning data to the real liver of the patient. In most of the cases, surgeons have to use projected data around the liver on two dimensional monitors during the surgery. To solve the problem, some approaches are suggested such as making 3D vessel models using 3D printer [1] or matching postures of a virtual liver on a computer and a real liver of the patient in real time for the computer navigation as we have developed [2].

The goal of our system is to navigate surgeons according to the paths previously prepared using the liver models scanned from patients, and to alert surgeons visually and audibly to be careful around the high risk areas.

M. Kurosu (Ed.): HCI 2018, LNCS 10902, pp. 53–63, 2018.
https://doi.org/10.1007/978-3-319-91244-8_5

2 Liver Surgical Navigation System

2.1 System Overview

Our liver surgical navigation system is composed of two cameras located above the liver and the surgical knife and a computer that communicates with those cameras as shown in Fig. 1. These cameras have different features and each camera will be used by different program.

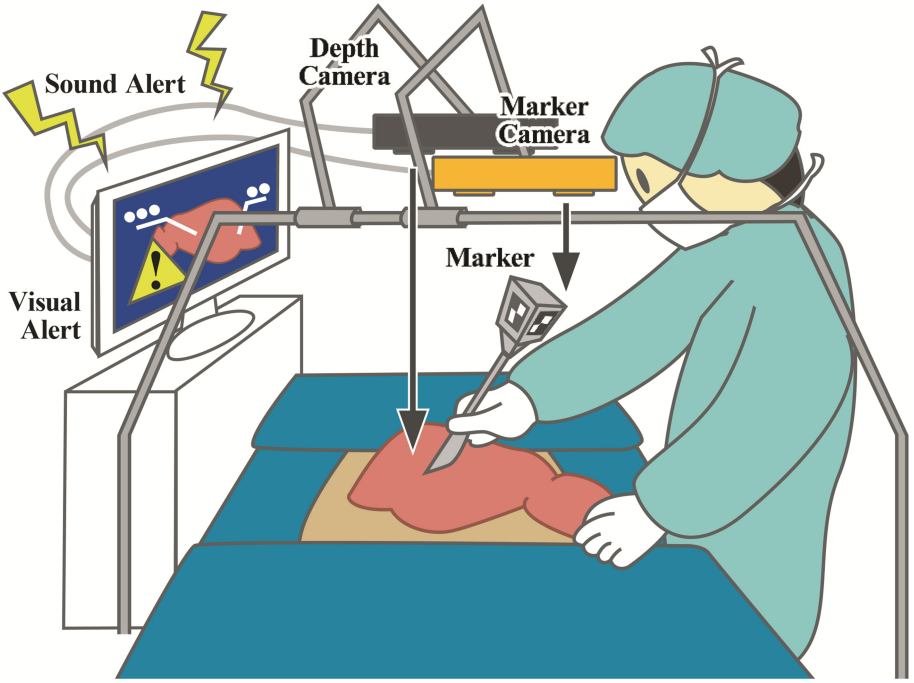

Fig. 1. System overview of our liver surgery support system

One camera (illustrated as a yellow camera in Fig. 1) is a marker tracking camera, which can track multiple two dimensional markers in real time. This marker tracking camera is used by the knife position estimator. As shown in the figure, multiple markers are put on the bottom of the surgical knife, and those markers will be captured and tracked by the camera. The knife position estimator uses position data of those markers put on the bottom to estimate the position of the knife tip.

Another camera (illustrated as a black camera in Fig. 1) is a depth camera, which can capture depth images in its measuring range. The liver position estimator repeatedly gets the depth image of the liver and perform an image processing for estimating current position and rotation of the liver.

We are currently using MicronTracker 3 as the marker tracking camera and Kinect v2 sensor as the depth camera, however, those cameras can be replaced with other cameras or sensors if precision allows.

2.2 Knife Position Estimator

The knife position estimator periodically gets marker data from the marker camera, and utilize the recognized marker information including positions and rotations for estimating the position of the knife tip. This estimator firstly calculate the average position of the markers put on the bottom of the knife, and then calculate the tip position by sliding the average position from the bottom to the top according to the preset length of the knife. Figure 2 shows how 2D markers are put on the bottom of the knife. The tip position is then converted to the depth camera coordinate, as it is well discussed in the next chapter.

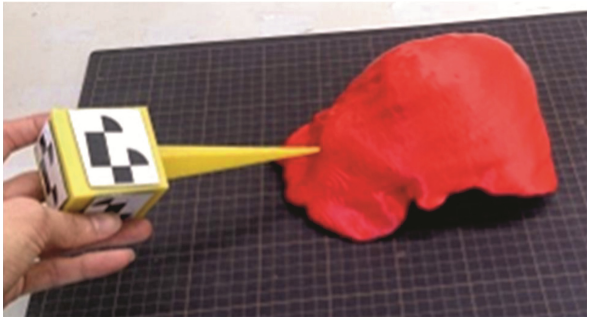

Fig. 2. A knife model with markers and a 3D printed liver model

2.3 Liver Position Estimator

The liver position estimator periodically gets a depth image using the depth camera, and then comparing the shape with Polyhedrons loaded on the memory for finding the exact location of the liver and its orientation. Figure 3 shows how the real liver model (A) is captured as a depth image (B) and how it will be compared with the depth images (C) rendered on the graphics memory using OpenGL.

Fig. 3. Real liver model (A), a depth image (B) and a rendered liver image (C)

2.4 Liver Surgical Navigator

The liver surgical navigator reads and integrates the knife position data and the liver position data from the named shared memories. It also loads the polyhedrons data with STL-format independently, for separately showing sub parts of the liver such as the hepatic artery, the portal vein, lobes and so on for the surgical navigation. In Fig. 4, the experimental implementation of the navigator is shown.

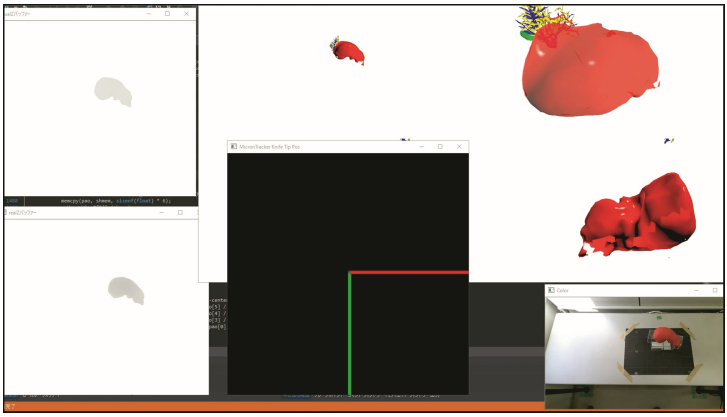

Fig. 4. The liver surgical navigator using knife and liver data from estimators

2.5 Testing Setup

Unfortunately, it is very difficult to attempt to verify our system repeatedly on human bodies. Instead, we have prepared DICOM based liver images captured by MRI, and integrated those images into polyhedrons with STL (Standard Triangulated Language) formatted data. Thereafter, we can make a 3D liver model from the STL data using a 3D printer, and the STL data also can be loaded in a computer memory to be rendered from any direction (Fig. 5).

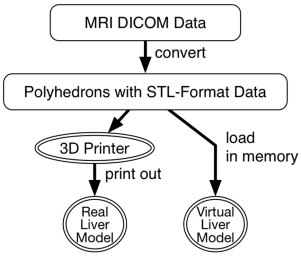

Fig. 5. A real liver model and a virtual liver model

The polyhedrons with the STL formatted data will be printed out as a real 3D model and also rendered on a graphics memory as depth images using OpenGL. The 3D liver

model will be used for moving its position and rotating its orientation, and then it is captured by a depth camera connected to the liver position estimator. The result of the liver position estimation can verify the precision of our system.

Of course, it is important not mind the influence of surgeons' shadows and the characteristics of surgical lighting (for heat reduction and shadow dilution) after the verification is well performed. But it is totally efficient way to construct the base of the liver position estimator of our system.

Figure 6 shows 3D printed surgical knife models used with the 3D printed liver model to verify the precision of the knife position estimator. The first model was introduced in [3], and it was totally made from course plastic and it had an average 2.5 mm error. An improved version was introduced in [4], and it was made from fine plastic. It had an average 2.0 mm error. The well improved version introduced in [5] is currently used and it has an average 0.8 mm error.

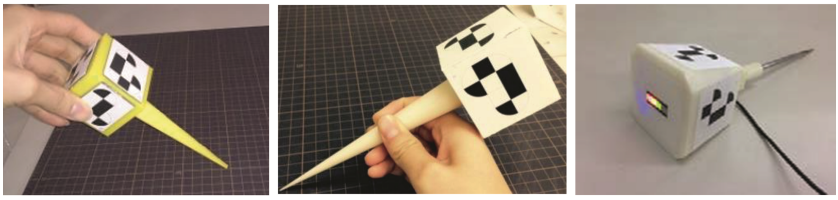

Fig. 6. Three types of 3D printed surgical knife models with bottom markers

3 Inter-process Communication

3.1 Data Flow

As we described above, three subsystems called the knife position estimator, the liver position estimator and the liver surgical navigator will work simultaneously and communicate each other. As each subsystem can be verified and improved each other, those subsystems are separately developed and are working as different programs in different processes.

Figure 5 shows how and what kind of data is flowed between the subsystems. The knife tip position is repeatedly written by the knife position estimator at a regular intervals. The position and the rotation of a liver is also repeatedly written by the liver position estimator at a regular intervals. Those data are written into separate named shared memories as illustrated in Fig. 5.

Shared memory is the memory that can be accessed by different processes simultaneously. Comparing to the other inter-process communication methods such as socket communication or named pipe, it is expected to be much faster as it is just writing and reading to/from the memories (Fig. 7).

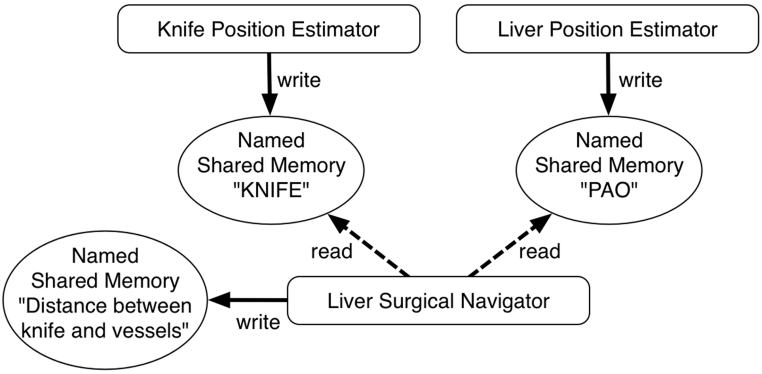

Fig. 7. Data flow in our system using named shared memories

The knife position estimator writes the knife tip position data in the shared memory named "KNIFE." The knife position is expressed as three double values (x, y, z). The unit used in the position value is discussed in the next section about the coordinate conversion.

The liver position estimator writes the liver position and the rotation in the shared memory named "PAO." (The name "PAO" for the liver position and rotation comes from "Position And Orientation," while the knife tip data has just the position.) The liver position is expressed as three float values (x, y, z) and rotation is expressed as three float values (roll, pitch, yaw). Those six float values are continuously written in position-orientation order. The unit used in the position values is discussed in the next section about the coordinate conversion. The unit used in the rotation values is degree.

The data flow design is very simple and they would not do other work. That is very important design to make subsystems so robust that they can continuously work and will not stop during the surgery.

3.2 Coordinate Conversion

In this section, the coordinate conversion between multiple data is precisely discussed. Because there are two cameras and the virtual rendered liver is located in the OpenGL space, those data should be converted in a common coordinate system.

In Knife Position Estimator. Multiple marker positions are captured by the marker tracking camera, and by calculating an average position of those positions, the knife position estimator calculates the tip position of the knife. Those calculations are performed in the coordinate system of the marker tracking camera. Because the body of the marker tracking camera and the body of the depth camera should be located in different position, and because they have different characteristics, those camera uses different coordinates. Therefore, the eight-point algorithm is used to calculate a conversion matrix after setting up those cameras. That conversion matrix is used to convert the knife tip position to the coordinate system of the depth camera, and the matrix is held as long as the same set up is used.

In Liver Position Estimator. A depth image is captured by the depth camera. As shown in Fig. 8, the STL-formatted liver model is loaded at the launch time and rendered from some different directions using an OpenGL renderer for one depth image. The number of the rendered image depends on the machine power, as the rendering process will be terminated after a certain duration has passed (currently the duration is set as ten processor time units in our implementation). Thereafter, the simulated annealing method is used to match the depth image with those rendered images to estimate the closest position and orientation of the liver. To simplify the comparison, the OpenGL rendering uses the same value of the actual depth camera characteristics for making a projection matrix: currently the field of view is set to 90 degrees and the aspect ratio is approximately 1.21. Here we can assume that the rendered images and the depth image are in the same coordinate system of the depth camera.

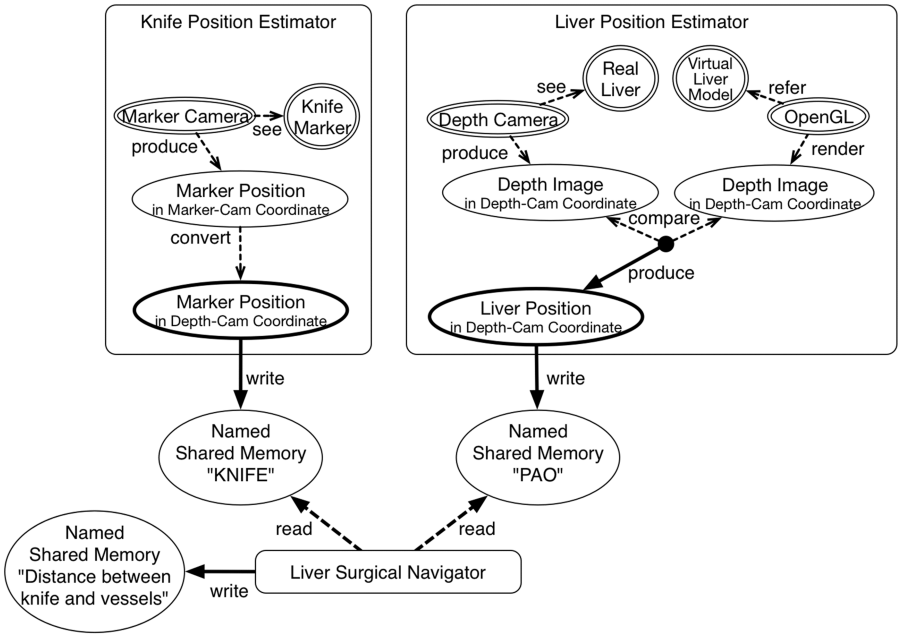

Fig. 8. Coordinate conversion between multiple camera coordinates

In Liver Surgical Navigator. As the knife position estimator and the liver position estimator now use the same coordinate system of the depth camera, the liver surgical navigator can read those data from each shared memory and combine those data for simulating the liver surgery. The liver surgery navigator independently loads STL formatted data and calculates distances between the knife tip and some vessels. Those distance data is also written in a named shared memory for the future application such as visual or sound alerts.

4 System Performance

In this chapter, performance of the whole liver navigation system is discussed, especially focusing on the knife and the liver position estimators while the navigation system is currently implemented as an experimental version. In our laboratory, a Windows PC is being used for the liver navigation system, with which the knife position estimator and the liver position estimator are simultaneously working. The PC has 6 cores of Intel Core i7-7800X CPU (@ 3.50 GHz) and 12 logical processors with 32.0 GB RAM and a graphics card of NVIDIA GeForce GTX 1080, on which Windows 10 Home edition is working. The QueryPerformanceCounter() function was used for the measurement, which can record a high resolution (under 1 ms according to the Microsoft API document) time stamp. Therefore, we assume that the measuring method gives us enough precision when counting in milliseconds.

4.1 Inter-process Communication Cost

At first, we measured the cost of writing and reading to/from the named shared memories both in the knife position estimator and the liver position estimator. However, writing or reading to/from each named shared memory took just one or two CPU clocks, that is less than 0.001 ms. As we described above, it is absolutely faster than any other inter-process communication methods like the BSD sockets.

4.2 Performance of the Knife Position Estimator

The knife position estimator is currently working on the OpenGL rendering flow with GLUT (OpenGL Utility Toolkit). The display callback function specified by glutDisplayFunc() function is repeatedly called in constant intervals, and within the callback function, marker positions captured by the marker camera are checked and the tip position is calculated by getting an average position of the markers.

An experiment was performed for measuring the performance of the knife position estimator. We firstly checked the callback intervals, secondly checked the, by changing the number of the marker captured by the marker camera from zero to two.

The callback intervals were almost constant with any number of captured markers, and the average duration of the interval was 16.668 ms that is precisely equivalent to 1/60 s, which is the basic vertical synchronizing signal interval of today's displays.

The calculation of finding the average position including coordinate conversion took 0.016 ms in average, for any number of markers captured. As Fig. 10 plots the calculation times for each frame within 500 frames (8.33 s), we can see that there is no explicit difference between the different number of captured markers for calculating the average position. Because the average 0.016 ms is definitely small comparing to 16.668 ms (= 1 frame), the calculation around the knife position estimation does not affect the callback ratio of the display function (Fig. 9).

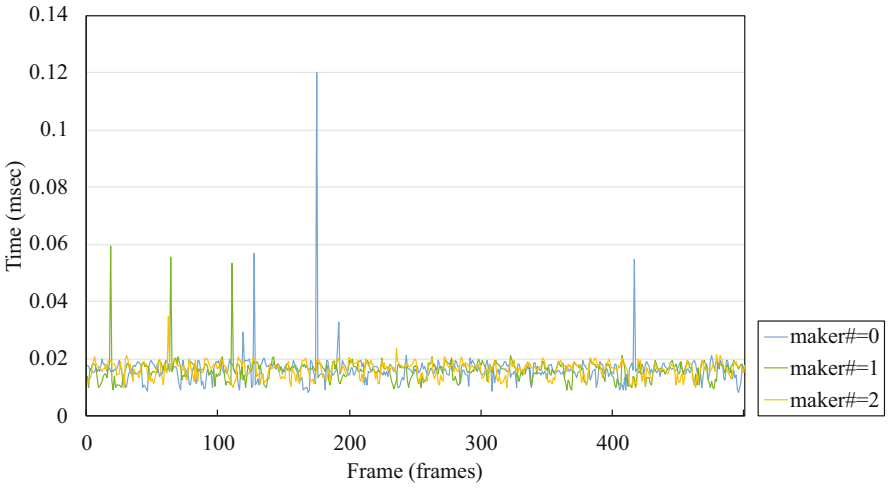

Fig. 9. Time for calculating knife tip positions

As the summary, we can say that the knife position estimator is running and writing the estimated data on the shared memory at 60 frame per second ratio, and the number of makers will not affect the rate.

4.3 Performance of the Liver Position Estimator

For the liver position estimator, we measured how many neighboring liver positions and orientations can be rendered using OpenGL and compare them to the depth image captured from the real liver. In our system, a parameter called "saTime" (prepared as a global variable) can be changed to specify the cutoff time for each simulated annealing process. The unit of the "saTime" variable is clocks per second same as the clock() function declared in time.h as a standard C language function.

Figure 10 shows how many times the liver position estimator can try neighboring liver positions and orientations at the specific saTime (shown as "saΔ" in the figure). Figure 11 shows how many simulated annealing processes (one process including multiple neighboring calculations) can be performed in a second at the specific saTime (shown as "saΔ" in the figure).

We can see that count of the neighboring calculation can be increased by increasing saTime as shown in Fig. 10. Even Fig. 11 also shows us that the simulated annealing performing count will not drastically changed with the different saTime setting, however, the performing count will not be decreased so much around the saTime between 150 to 300 clocks. It is strongly considered that the setting of this parameter should be well discussed in the future.

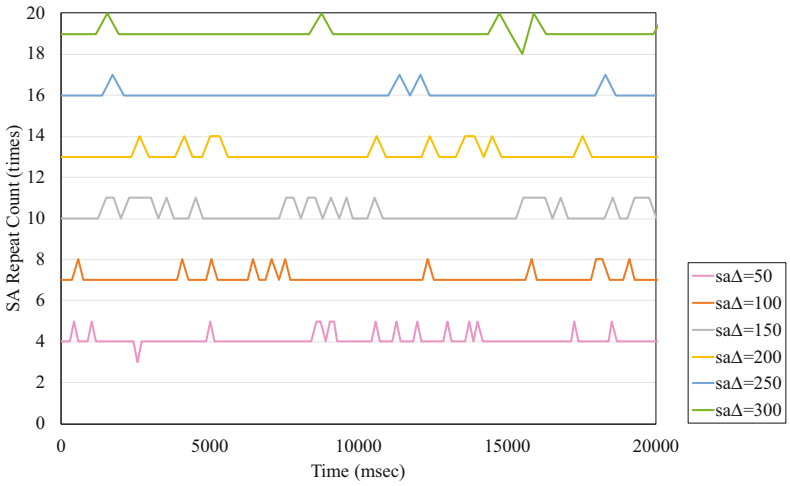

Fig. 10. Simulated annealing repeat count in one calculation

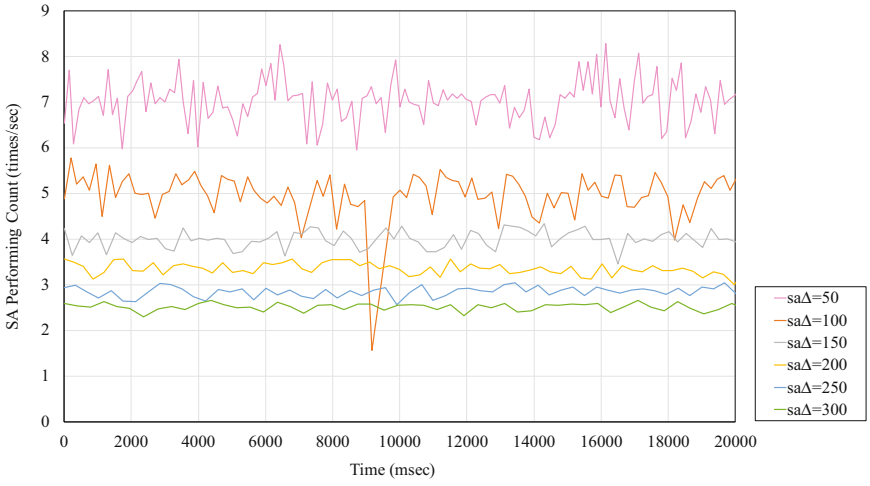

Fig. 11. Simulated annealing performing count in one second

5 Conclusion

In this paper, we described the overview of our liver surgical navigation system composed of subsystems and its data flow between different coordinate systems. The performance of our estimators are discussed and the data showed how the inter-process communication using the named shared memories is effectively transmitting the knife position and the liver position data needed in the liver surgical navigator.

For the future enhancements of our system, few things should be mentioned. First, the performance of the liver position estimator varies quickly by changing the parameter

named "saTime." That parameter setting should be well discussed with the other parameters described in [6]. Second, the way of mutual exclusions over the usage of the named shared memories should be deeply concerned because it affects the performance of the entire system. The names of the shared memories currently do not have the unity among the system (the names of "KNIFE" and "PAO" do have asymmetry just now), so it should also be improved in the future. The "KNIFE" and the "PAO" shared memories have different data types by the historical circumstances, but it should be properly measured and united in the future.

Acknowledgement. This research was supported by Grants-in-Aid for Scientific Research (No. 26289069) from the Ministry of Education, Culture, Sports, Science and Technology (MEXT), Japan.

References

1. Kuroda, S., Kobayashi, T., Ohdan, H.: 3D printing model of the intrahepatic vessels for navigation during anatomical resection of hepatocellular carcinoma. Int. J. Surg. Case Reports **41**, 219–222 (2017). ISSN 2210-2612
2. Koeda, M., et al.: Depth camera calibration and knife tip position estimation for liver surgery support system. In: Stephanidis, C. (ed.) HCI 2015. CCIS, vol. 528, pp. 496–502. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21380-4_84
3. Doi, M., Koeda, M., Tsukushi, A. et al.: Kinfe tip position estimation for liver surgery support. In: Proceedings of the Robotics and Mechatronics Conference 2015 in Kyoto (ROBOMECH 2015), 1A1-E01, May 2015
4. Doi, M., Yano, D., Koeda, M. et al.: Knife tip position estimation using multiple markers for liver surgery support. In: Proceedings of the 2015 JSME/RMD International Conference on Advanced Mechatronics (ICAM 2015), 1A2-08, pp. 74-75, Tokyo, Japan, December 2015
5. Doi, M., Yano, D., Koeda, M. et al.: Knife tip position estimation for liver surgery support system. In: Proceedings of Japanese Society for Medical Virtual Reality (JSMVR 2016), pp. 36–37, September 2016
6. Watanabe, K., Yoshida, S., Yano, D., Koeda, M., Noborio, H.: A new organ-following algorithm based on depth-depth matching and simulated annealing, and its experimental evaluation. In: Marcus, A., Wang, W. (eds.) DUXU 2017. LNCS, vol. 10289, pp. 594–607. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-58637-3_47