A Lightweight and Portable Communication Framework for Multimodal Image-Guided Therapy

Release 1.00

A. Schoch[1], B. Fuerst[1], F. Achilles[1] S. Demirci[1] and N. Navab[1]

June 13, 2013

[1] Computer Aided Medical Procedures, Technische Universität München, Germany

Abstract

In today's medicine a lot of research revolves around the combination of different imaging and tracking modalities to form a new system for Image-Guided Therapy. Such systems can help to overcome shortcomings of the individual techniques or create new kinds of possible use cases. To ease the development of a hybrid modality we have created a generic, portable, lightweight, and easily extensible, client-server based framework, implemented in C++, to control the communication between multiple imaging and tracking devices. Our framework *Computer-Aided Medical Procedures Communication* (CAMPCom), allows developers of hybrid systems to focus on the most important implementation aspects by freeing them from the more generic tasks of data serialization and exchange.

Latest version available at the Insight Journal [http://hdl.handle.net/10380/3408]

Contents

1 Introduction

Recently, a new trend towards fusing various imaging and tracking modalities, to create hybrid systems can be observed. Such systems encounter common problems: data exchange between individual devices and software solutions. Therefore, we propose a new communication framework CAMPCom, which contributes a solution to this problem.

1.1 Multimodal Image-Guided Therapy

In multi-modal Image-Guided Therapy (IGT) data sources are merged in order to combine individual modalities and to create a completely new solution for physicians. An example for that is freehandSPECT, a combination of an optical tracking system with a gamma probe to create a [10]. This provides 3D SPECT data, which was previously only available at pre- or postoperative stage, to the intra-operative domain and can even further be combined with Ultrasound (US) or Magnetic Resonance (MR) data [5].

Multimodal and Intra-Operative Imaging: The *Endoscopic Time-Of-Flight Position Emission Tomography and Ultrasound* (EndoTOFPET-US[1]) project is an EU FP7 cooperation project between six research institutes, three clinical centers and three companies. The project presents a hybrid imaging technology, which combines an endoscopic PET head inside the patient, featuring also a new type of scintillation crystal, with a PET plate outside the patient to generate high resolution PET ($\sim 1\,mm$) by incorporating Time-Of-Flight information using (temporal resolution $< 200\,ps$). This functional data is further combined with anatomical data from an US device. Hybrid tracking (optical and electromagnetic) is required for both the PET incident detection and the 3D compounding of US. This improvement to PET technology might contribute to the detection of certain tumor types like pancreatic carcinoma in a much earlier stage [2].

While such hybrid modalities often provide a lot of benefits they also introduce new challenges: To combine the information, different data sources have to be registered with each other. But before being able to do that, data has to be exchanged. This can be quite simple when the communication only takes place within one software solution or on a single computation platform but can become a very difficult task once it has to be performed between several computers, which are e.g. connected via local area network (LAN) or the internet, use different operating systems or different programming languages.

1.2 State of the Art Software Solutions

There is a range of commercial and open-source frameworks for Computer Assisted Interventions (CAI) or IGT, such as the Image-Guided Surgery Toolkit (IGSTK) [3], OpenIGTLink [3], 3D Slicer [4] or SIGN [8], each with its individual focus and set of strengths and weaknesses.

The *Image-Guided Surgery Toolkit* (IGSTK) is a recently developed and constantly updated open source `C++` framework. Its key aspects are safety and robustness. While it enables its user to read and visualize images, it also offers an interface for several different tracking devices and features logging and a Graphic User Interface (GUI) among other. However, it does not contain any functionality for network communication by itself and thus also no inter-device control flow [1, 3].

OpenIGTLink, on the other hand, emphasizes on inter-device communication providing a network protocol for data exchange in combination with a multitude of data exchange types (e.g. for tracking and image data) and serialization functions as well as support for a range of tracking devices.

[1] https://endotofpet-us.web.cern.ch/

While it helps in the development of a client/server architecture by providing specialized sockets it does not contain a pre-build architecture unlike our framework, and, thus, leaves the setup of it and the control flow to the developer. It also does not feature visualization options [9].

3D Slicer is an open-source framework, which is well known for delivering great tools for visualization and image analysis on multiple platforms. Building upon libraries like *Insight Segmentation and Registration Toolkit* (ITK)[2] and *Visualization Toolkit* (VTK)[3] and offering a multitude of functions it is very powerful but far from being lightweight [7]. It also offers no network communication for inter-device connection by default but, due to its plugin mechanism, there exist solutions which add this functionality [4].

Finally, the *Slicer Image-Guided Navigator* (SIGN) shares a lot of its properties (e.g. being powerful but not very lightweight) with 3D Slicer. Likewise, a lot of established open-source libraries like VTK, ITK, OpenTracker[4] and ACE[5] for network functionality as well as the same modeling language MRML[6] for its data model are used [8].

1.3 Proposed Solution

In this paper, we propose a framework to overcome the challenges mentioned. It features a platform-independent, lightweight, and easily expandable network communication framework using a client/server architecture, to connect various devices with each other and to control the message and data flow between them. Compression and prioritization are native functions, which are highly relevant when multicasting image data. Additional features include a lightweight and thread-safe logging library (CAMPLog), a GUI front-end, fast and meaningful visualization of image data due to the close connection to the novel CAMPVis[7] library and an automatic serialization and de-serialization of data exchange types.

2 Methods

2.1 Communication Framework Design Goals

Among other design decisions like a close connection to the novel visualization framework CAMPVis we emphasized on the following goals:

1. **Control and Data Flow:** Besides data communication, it should be possible to control the clients or the server by transmitting commands and warnings. Additionally, data should be distributed in a controlled way.
2. **Efficiency:** The framework should be lightweight and the communication efficient.
3. **Platform Independency:** Data serialization and transmission should be compatible with all types of operating systems and hardware architectures.
4. **Abstraction and Extensibility:** The framework has been designed to support extensions, by defining a general interface.

[2] http://www.itk.org/
[3] http://www.vtk.org/
[4] http://www.opentracker.net/
[5] http://www.cs.wustl.edu/~schmidt/ACE.html
[6] http://www.slicer.org/slicerWiki/index.php/Modules:DataModule-3.4
[7] http://campar.in.tum.de/Main/ChristianSchulteZuBerge

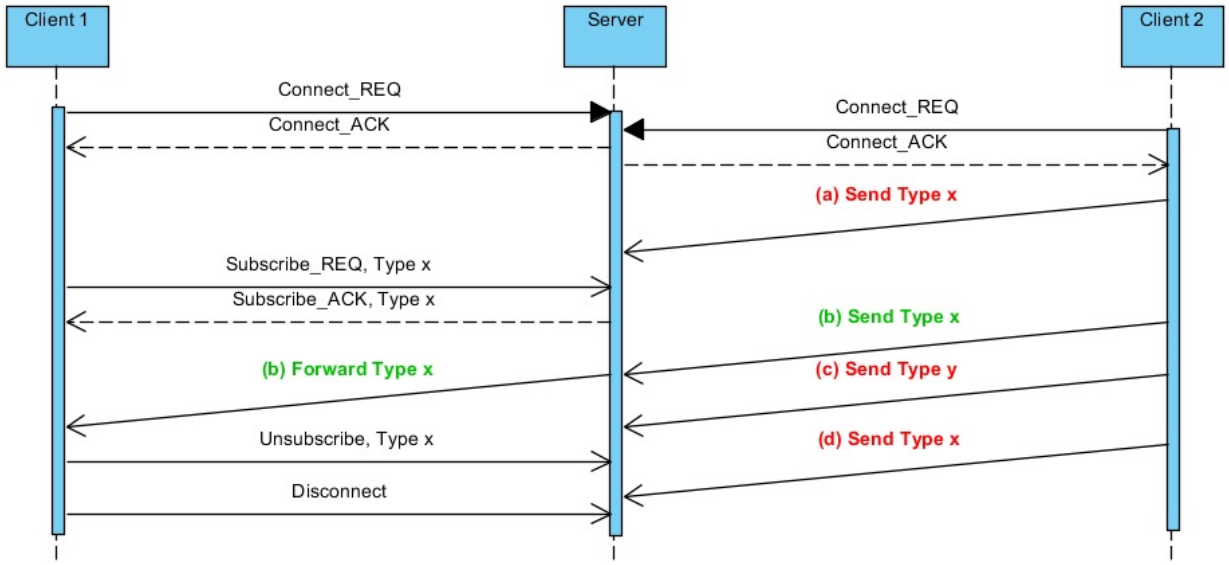

Figure 1: A typical communication sequence including the subscription model. Out of the 4 messages (a-d) only (b) is forwarded. During the time (a) and (d) reach the server, client 1 has no active subscription for Type x and message (c) is of a different type.

2.2 General Design

To offer a centralized exchange of messages and data between the devices we utilized a client/server architecture. Both the client and the server are offered as `C++` libraries and can either be easily integrated as plugin in an existing software solution or wrapped in a simple (console or GUI) application for a non-intrusive approach. Furthermore, most of the communication functions are provided as synchronous and asynchronous versions. A network protocol was implemented which defines the structure of the header and its serialization as well as different data exchange types, which can be extended by the user.

Control and Data flow: To control data flow, we used a subscription model similar to the known observer pattern. When connected to the server, a client can subscribe for any message type that has been defined (e.g. tracking data) and will (only then) receive a multicast of that type by the server (see Fig. 1). Of course, such a subscription can be revoked at any time. Using a function pointer or function object as callback, a developer can then decide how to handle the received data. Regarding the control flow we implemented a reserved system message type that can be used to relay commands or information (like warnings) to the server or other clients (by adding a special multicast command).

Efficiency: Keeping in mind that a high server load can easily slow down a centralized communication, the interaction of the server and a data transmission of a client was reduced to a minimum. Despite control messages, all other messages will not be de-serialized by the server. Instead only the small header of the message, which is part of our communication protocol, will be opened to determine the type of the message to forward it.

Prioritization: Another feature of the client library is a special query handling and priority mechanism for asynchronous functions to support efficiency (see Fig. 2). Upon execution, those functions are wrapped into a function object together with their parameters until the object is consumed in order to delay as much computation as possible until needed. Each of those queries is then placed into a priority queue so that they can be executed by a dispatcher based on their priority. In case of a huge number of messages, it is possible to limit the maximum number of parallel worker threads that are used as targets by the dispatcher in order

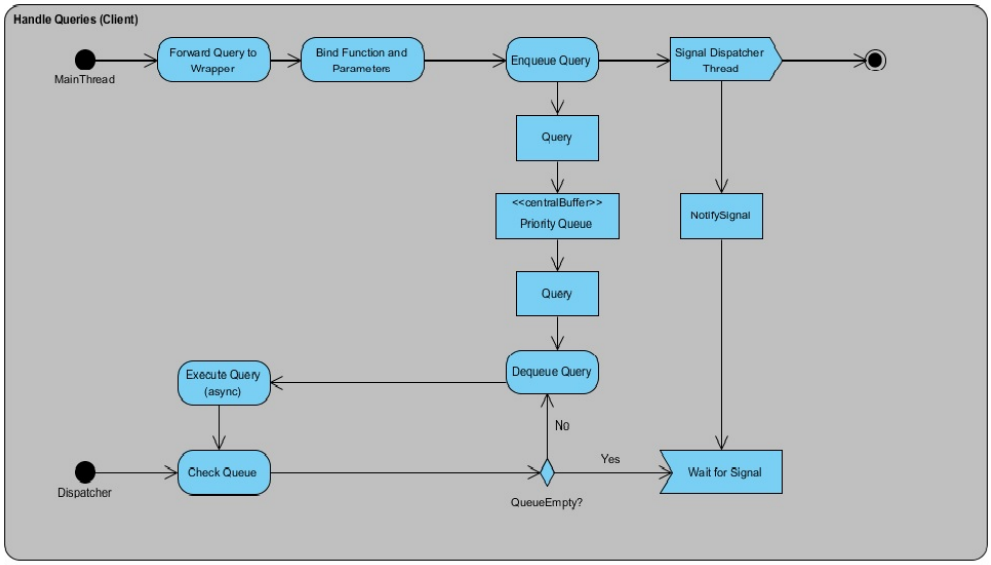

Figure 2: An activity diagram of the query handling and prioritisation of the client library.

to prevent the distribution of computation time between too many threads

Platform Independence: Using the boost::Serialization library, we provided an easy and efficient way to serialize data independent of the operating system or defined sizes of data types. We also provided different serialization functions, which offer a different grade of platform independence and performance. Compression is performed by integrating zlib[8]. Via CMake[9] build options, a user can chose which of these two properties to focus on.

Abstraction and Extensibility: To introduce a new data exchange struct to our serialization and communication functions, a developer just has to provide a single header (and optionally a source file) containing the declarations and definitions of the data type (member variables), a template specialization for a single function of the TypeHandler class, as well as a generic method for serialization. Using a template specialization for compile-time polymorphism, we also provide very simple and abstract functions e.g. for sending a data type, which perform most of the work automatically.

Frontend and other Features: Both for the client and server library, we build a GUI front-end using the Qt framework[10], which focuses on simplicity and only the most important functions while providing a console widget if the user needs more specialized commands or feedback. For visualization of 2D and volumetric image data, CAMPCom is inter-operable and may optionally include the novel `C++` visualization library CAMPVis[7], which was created and is constantly maintained by CAMP. Also, the lightweight and thread-safe logger CAMPLog is included, which is used in both libraries and can also be used independently in other projects.

3 Testing and Evaluation

To evaluate the framework we analyzed the communication between the modules (clients) and the server in terms of connection issues (e.g. dropped packages) and transfer rates.

[8] http://zlib.net/
[9] http://www.cmake.org/
[10] http://qt-project.org

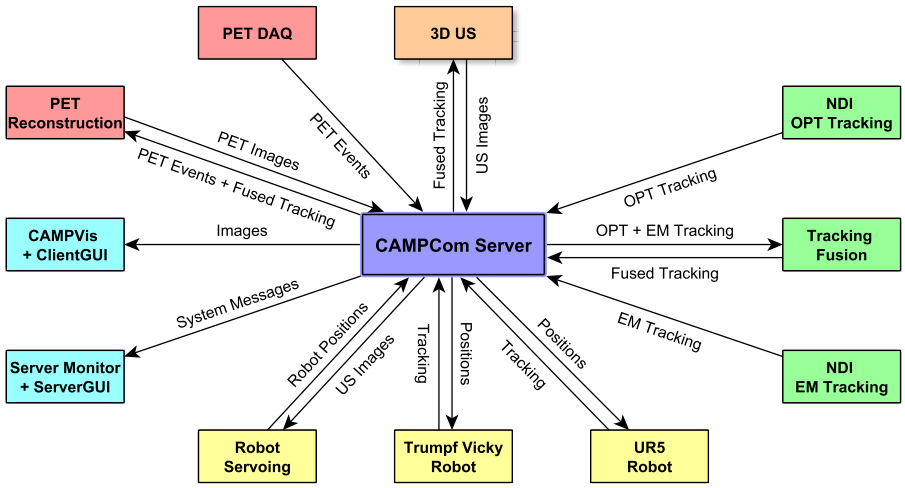

Figure 3: An overview of the inter-device communication which is used for the EndoTOFPET-US project.

	Loading data	Serialization	Sending	Receiving	Deserialization	Total
OpenIGTLink						
Duration [sec]	16.464	0.728	8.837	11.258	0.052	37.311
Transfer rate [MB/sec]	6.074	137.269	11.316	8.883	4035.152	2.680
CAMPCom (Compression and Prioritization off)						
Duration [sec]	14.026	0.666	9.272	12.247	0.882	37.094
Transfer rate [MB/sec]	7.129	150.039	10.785	8.165	113.399	2.696

Table 1: By averaging over six experiments, the transfer rates of CAMPCom and OpenIGTLink when sending a 100MB raw data package on a round trip (from client to server and back) are compared. Both frameworks are built in the simplest implementation.

Client/Server Communication: In order to test the subscription model, we connected several clients to the server (Fig. 3) and transmitted random and static test data. In both scenarios we could not observe any data loss or connection issues. We therefore reason that the incorporation of boost libraries is of great benefit.

Transfer rates: We performed several experiments in order to evaluate the transfer capacity and rates, in which we have observed an average transfer rate of 90 % of the constant Microsoft Windows file transfer rate between multiple clients using SMB[11]. In order to compare our framework to state of the art software, we performed benchmarks using the simplest implementation of both CAMPCom and OpenIGTLink. For CAMPCom, this already includes all possible data types, subscription possibilities of all data, and multicasting, prioritization and compression of data packages. In six independent experiments we could observe that our framework has very similar transfer rates as OpenIGTLink, see Tab. 1.

Prioritization and Compression: We evaluated the performance of other native functions, such as prioritization and compression, but did not compare it to state of art software as these features are not natively integrated in OpenIGTLink. The priority queue approach after serialization allows packages of the same type (e.g. system message) to be sent out faster. Besides, we evaluated the compression[12] approach using MRI data from the BITE database [6]. We can achieve an average compression rate of 67,60 % (at

[11] Server Message Block, see https://en.wikipedia.org/wiki/Server_Message_Block
[12] using deflate from zlib, http://zlib.net/feldspar.html

26.16MB/sec), resulting in an overall transfer rate of 3.35MB/sec, which is an improvement of 19.52% (2.69MB/sec) compared to data sent without compression. We are convinced that the integration of prioritization and compression are absolutely necessary for progressing the state of art in IGT systems and are therefore the most crucial contributions of our proposed framework.

4 Conclusion and Outlook

We have presented a stable, reliable, and fast framework for research purposes in the field of image guided interventions. The CAMPCom framework provides an efficient way to connect various devices in an intra-operative environment. Its lightweight nature and extensibility allows easy integration. At the same time it is equipped with various features and hides most of the complexity of the communication process from the user.

We have shown that the framework reaches performance similar to the state of the art software solution, while offering additional features, such as prioritization in communication, and compression of data to reduce the required bandwidth. We have provided our evaluation results and most of the implementation details in this paper, and will make the entire framework openly available[13] through our webpage[14]. We believe that the framework will progress the state of the art in IGT systems and therefor provides a valuable contribution to the community. Furthermore, we encourage fellow researchers to adapt the framework and contribute new features to CAMPCom.

We are planing to add new features like XML parser in combination with a settings manager to ease during run-time, as well as a code generator to easily create new compile-time data exchange types and automatically add them to the system.

Acknowledgements: This research was performed in the course of the EndoTOFPET-US project funded within the EU FP7 framework (FP7/2007-2013, Grant Agreement No. 256984). The authors would like to thank Christian Schulte zu Berge for providing and maintaining CAMPVis.

References

[1] Andinet Enquobahrie, Patrick Cheng, Kevin Gary, Luis Ibanez, David Gobbi, Frank Lindseth, Ziv Yaniv, Stephen Aylward, Julien Jomier, and Kevin Cleary. The image-guided surgery toolkit IGSTK: an open source C++ software toolkit. *Journal of Digital Imaging*, 20(1):21–33, 2007. 1.2

[2] Erika Garutti. EndoTOFPET-US: a Novel Multimodal Tool for Endoscopy and Positron Emission Tomography. *arXiv preprint arXiv:1303.4503*, 2013. 1.1

[3] Kevin Gary, Luis Ibanez, Stephen Aylward, David Gobbi, M Brian Blake, and Kevin Cleary. IGSTK: an open source software toolkit for image-guided surgery. *Computer*, 39(4):46–53, 2006. 1.2

[4] András Lassó, Junichi Tokuda, Siddharth Vikal, Clare M Tempany, Nobuhiko Hata, and Gabor Fichtinger. A generic computer assisted intervention plug-in module for 3D Slicer with multiple device support. 2009. 1.2

[13] Please find a summary of CC BY 3.0 DE on http://creativecommons.org/licenses/by/3.0/de/

[14] http://campar.in.tum.de/view/Main/CAMPCom

[5] Philipp Matthies, Asli Okur, Thomas Wendler, Nassir Navab, and Michael Friebe. Combination of intra-operative freehand SPECT imaging with MR images for guidance and navigation. In *IEEE Engineering in Medicine and Biology Conference (EMBC)*, 2013. to appear. 1.1

[6] Laurence Mercier, Rolando F Del Maestro, Kevin Petrecca, David Araujo, Claire Haegelen, and D Louis Collins. Online database of clinical MR and ultrasound images of brain tumors. *Medical Physics*, 39:3253, 2012. 3

[7] Steve Pieper, Michael Halle, and Ron Kikinis. 3D Slicer. In *Biomedical Imaging: Nano to Macro, 2004. IEEE International Symposium on*, pages 632–635. IEEE, 2004. 1.2

[8] Eigil Samset, Arne Hans, Jochen von Spiczak, Simon DiMaio, Randy Ellis, Nobuhiko Hata, and Ferenc Jolesz. The SIGN: A dynamic and extensible software framework for Image-Guided Therapy. 2006. 1.2

[9] Junichi Tokuda, Gregory S Fischer, Xenophon Papademetris, Ziv Yaniv, Luis Ibanez, Patrick Cheng, Haiying Liu, Jack Blevins, Jumpei Arata, Kapur Tina Golby, Alexandra J, Steve Pieper, Everette C. Burdette, Gabor Fichtinger, Clare M. Tempany, and Nobuhiko Hata. OpenIGTLink: an open network protocol for image-guided therapy environment. *The International Journal of Medical Robotics and Computer Assisted Surgery*, 5(4):423–434, 2009. 1.2

[10] Thomas Wendler, Ken Herrmann, Andreas Schnelzer, Tobias Lasser, Joerg Traub, Olivier Kutter, Alexandra Ehlerding, Klemens Scheidhauer, Tibor Schuster, Marion Kiechle, et al. First demonstration of 3-D lymphatic mapping in breast cancer using freehand SPECT. *European journal of nuclear medicine and molecular imaging*, 37(8):1452–1461, 2010. 1.1