Using Middleware Framework in HapticIO

Within the scope of HapticIO project[1] which aims to create a generic haptical device to promote mixed reality (MR) in workplaces, it is important to facilitate the software architecture for every constituents from different fields and background who run this project together. The project HapticIO is a joint effort by the partners

- the Department of Humanoid Robotics and Human-Technology Interaction[2] at the Berliner Hochschule für Technik (BHT) as the joint head
- the Intuity Media Lab GmbH[3]
- the Department of Media Informatics[4] at the University of Regensburg
- the Chair of Social Implications and Ethical Aspects of AI[5] at Ingolstadt University of Technology.

and supported by the KMU-Innovativ program of the Federal Ministry of Education and Research (BMBF). Since this long-term scheme includes not only designing the physical product itself in terms of its usability, ergonomics and meaningful feedbacks but also creating mixed reality set-ups and environments, it is important to combine these two in a smooth process which allows project owners focus on their research and products. It can be also said that, creating a proper working set-up decreases the user alienation from the mixed reality surrounding. For example, even if a haptic feedback tool would work properly, technical issues related to connection between MR software and the tool hardware could cause misinterpretations over proper haptic tool. Therefore, Konstruktiv, which can be introduced as the technical unit in HapticIO, offers using "Middleware Framework" which is mostly used in robotic systems.

What is Middleware Framework?

With today's technological developments and the emergence of new demands in order to helping humans for more efficient work, robot construction has been getting more complex than before. For developing a complex robotic system in vigorous, scalable and reliable way, it is necessary to connect different specialisations from different fields such as mechanical engineering, computer science and electrical engineering [15, 18]. A middleware layer between an operating system and software

[1]https://interaktive-technologien.de/projekte/hapticio
[2]https://projekt.bht-berlin.de/harmonik/
[3]https://intuity.de/
[4]https://www.uni-regensburg.de/sprache-literatur-kultur/medieninformatik/
[5]https://www.thi.de/informatik/personen-i/prof-dr-matthias-uhl/

applications (Figure 1) is used to combine components from different specialist and integrate them in one system. It can be said that middleware is a translation software that allows software applications to understand the information sent from operating system or vice versa. This layer promotes simplifying software design by providing standardised infrastructure communication between modules, as well as enhancing software application quality by giving developers more space to modify and improve their components in any time individually [2]. Even if this systems have been developed for robotics primarily, they can be also useful tools for other

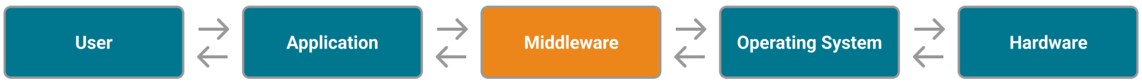

Figure 1: Basic contruction of middleware usage.

fields which includes different disciplines in themselves such as Human Computer Interaction (HCI) which also covers the HapticIO project. Since HCI is an interdisciplinary field, it can be also advocated that, for combining the knowledge and skills from different specialist like computer scientist and designers to come up with a new human-focused interface ideas through prototypes, a middleware layer would be an useful tool. For example, in the Intelligent Self-Driving Car System project of Suresh and colleagues [17], while they were prototyping a small scale autonomous car with additional sensors to define color differences between roads and roadsides, to manage software framework they have used one of the standard middlewares in robotic community: Robot Operating System (ROS). Using ROS during the project has served them to connect and control all different mechanical (cameras, sensors and other car elements) and electrical components (Raspberry Pi, Arduino and other supporter components) in one software system. Besides ROS, other examples of robotic software systems currently used could be YARP, Orocos or CARMEN.

Robotic Operating System (ROS)

As it is mentioned before, one of the most used middleware in the robotics community is ROS. It was designed through the STAIR project at Stanford University and Personal Robots Program at Willow Garage to solve challenges encountered during the forming of large-scale service robots [14] with following philosophical goals: Peer-to-peer (P2P), tool-based, multilingual, thin, and free and open-source. Additionally, with new improvements in the system, in ROS 2 project [9], it provide real-time communication which means there would be a direct path between the source and the destination. ROS officially supports C++ and Phyton with community-provided support for numerous other languages (such as Java, Android, C# or Swift) as well as can run under Ubuntu Linux, RHEL, Windows and macOS.

Open Robot Control Software (OROCOS)

OROCOS project is another free middleware framework which can be basically defined as a collection of portable C++ libraries [18]. These libraries can be divided into 3 basic tools: Kinematics and Dynamics library (supporting modelling and computation of kinematic chains such as robots or computer-animated figures [18]), Bayesian Filtering Library (providing interpretations on Dynamic Bayesian Networks [4]) and The Orocos Toolchain (creating real-time robotics applications [18]).

Yet Another Robot Platform (YARP)

YARP is an another open source library to support software development on humanoid robotics mostly [10]. In terms of humanoid robotics, it is used for visual perception, human robot interaction, dexterous manipulation, and legged locomotion [1]. It supports Windows, Linux, MacOS and Solaris as well as it is written in the C++ language.

Carnegie Mellon Robot Navigation Toolkit (Carmen)

Carmen [11], which was created in Carnegie Mellon University as a robot navigation toolkit, is an open source collection of software for mobile robot control. It includes fundamental navigation units such as sensor control, obstacle avoidance, localization and mapping [18]. Carmen runs under Linux operating system and is written in the C programming language with Java support.

Framework Name	Language(s)	Simulation	OS	Real-Time
ROS 2	Python, C++, Lisp, Java	yes	Windows, Linux, MacOS,	yes
OROCOS	C++	yes*	Windows, MacOS	yes
YARP	C++	yes	Windows, Linux, MacOS	no
Carmen	C	yes**	Linux	no
*Only with Orocos Simulink Toolbox **Only 2D simulation				

Table 1: Basic comparisons between mentioned middleware frameworks

Apart from mentioned middleware frameworks, there are other available middlewares such as Miro [3], OPRos [6] or Rock [7] which are worth mentioning.

Which Middleware Framework for HapticIO?

Within the HapticIO, Konstruktiv anticipated some future software wise necessities in terms of ongoing and incoming projects. Besides using a open source tool, firstly, since this project is carried out by professionals from different companies and universities, it is important to choose one which can be programmable by every independent group in the project. In other words, it should be language-independent, so that people do not need to learn a new language. Secondly, it is also crucial to keep operation systems (OS) what professionals ordinarily use which prevents source and time waste. Therefore, choosing a framework which supports several OSs would be helpful. Thirdly, as long as HapticIO project depends on creating prototypes through Mixed Reality tools, simulating these prototypes would taking up a big space in the project. Therefore, the middleware framework to be used should provide this service easily. Moreover, it would be a plus if the selected middleware framework could be a easy tool to integrate more than one sensor. This could serve us a big frame for trial-and-error on designing, testing and optimizing prototypes.

Finally, it is estimated that it would be possible to need a real time computing for the future steps of the HapticIO project parts. Real time computing guarantees hardware and software systems to response commands without significant delay. To maintain realistic environment and responses from it in a virtual world for the users in HapticIO , it is important to capture their gestures and behaviors simultaneously as well as giving meaningful coordinated feedback. For example, real-time processing can be used for hand gesture recognition, hand tracking as well as facial expression classification in which it is expected to get quick, accurate and natural feedbacks from dynamic environment by users [5, 8, 16, 19].

In the light of these desired features, Konstruktiv suggests to use ROS 2 for sophisticated prototypes including various hardware and software component in HapticIO. First of all, in terms of programming languages,ROS framework can be implemented in Python, C++, Lisp or Java which means it is a language-independent while supporting various OS. Moreover, one of the most advantageous parts of the ROS for making prototypes is that it provides smooth integration of more than one sensor at the same time as well as with the new ones without any hardware expertise requirements [13]. For example, for multiple sensors calibration on a autonomous vehicle, Oliveira and colleagues [12] used ROS in their study and indicated that their proposed approach ensures that multiple simultaneous sensors in smart vehicles get at least as accurate calibration performance as pairwise setups. Additionally, it provides real time performance as well as simulation.

Bibliography

[1] Miguel Aragão, Plinio Moreno, and Alexandre Bernardino. 2016. Middleware interoperability for robotics: a ros–yarp framework. *Frontiers in Robotics and AI*, 3. ISSN: 2296-9144. DOI: 10.3389/frobt.2016.00064. https://www.frontiersin.org/article/10.3389/frobt.2016.00064.

[2] Ayssam Elkady and Tarek Sobh. 2012. Robotics middleware: a comprehensive literature survey and attribute-based bibliography. *Journal of Robotics*, 2012.

[3] Stefan Enderle, Hans Utz, Stefan Sablatnög, Steffen Simon, Gerhard Kraetzschmar, and Günther Palm. 2001. Miro: middleware for autonomous mobile robots. *IFAC Proceedings Volumes*, 34, 9, 297–302. IFAC Conference on Telematics Applications in Automation and Robotics, Weingarten, Germany, 24-26 July 2001. ISSN: 1474-6670. DOI: https://doi.org/10.1016/S1474-6670(17)41721-6. https://www.sciencedirect.com/science/article/pii/S1474667017417216.

[4] Klaas Gadeyne. 2001. BFL: Bayesian Filtering Library. http://www.orocos.org/bfl. (2001).

[5] S L Happy, Anjith George, and Aurobinda Routray. 2012. A real time facial expression classification system using local binary patterns. In *2012 4th International Conference on Intelligent Human Computer Interaction (IHCI)*, 1–5. DOI: 10.1109/IHCI.2012.6481802.

[6] Choulsoo Jang and Seung-Ik Lee. 2010. Opros: a new component-based robot software platform. *ETRI Journal*, 32, (October 2010), 646–656. DOI: 10.4218/etrij.10.1510.0138.

[7] Sylvain Joyeux and Jan Albiez. 2011. Robot development: from components to systems, (May 2011).

[8] Piyush Kumar, Jyoti Verma, and Shitala Prasad. 2012. Hand data glove: a wearable real-time device for human-computer interaction. *International Journal of Advanced Science and Technology*, 43, (January 2012), 15–26.

[9] Steven Macenski, Tully Foote, Brian Gerkey, Chris Lalancette, and William Woodall. 2022. Robot operating system 2: design, architecture, and uses in the wild. *Science Robotics*, 7, 66, eabm6074. DOI: 10.1126/scirobotics.abm6074. https://www.science.org/doi/abs/10.1126/scirobotics.abm6074.

[10] Giorgio Metta, Paul Fitzpatrick, and Lorenzo Natale. 2006. Yarp: yet another robot platform. *International Journal of Advanced Robotic Systems*, 3, 1, 8.

[11] M. Montemerlo, N. Roy, S. Thrun, D. Hähnel, C. Stachniss, and J. Glover. 2002. CARMEN – the carnegie mellon robot navigation toolkit. (2002). http://carmen.sourceforge.net.

[12] Miguel Oliveira, Afonso Castro, Tiago Madeira, Eurico Pedrosa, Paulo Dias, and Vítor Santos. 2020. A ros framework for the extrinsic calibration of intelligent vehicles: a multi-sensor, multi-modal approach. *Robotics and Autonomous Systems*, 131, 103558. ISSN: 0921-8890. DOI: `https://doi.org/10.1016/j.robot.2020.103558`. `https://www.sciencedirect.com/science/article/pii/S0921889020303985`.

[13] Anis Koubaa, editor. 2019. *A ros-based framework for simulation and benchmarking of multi-robot patrolling algorithms. Robot Operating System (ROS): The Complete Reference (Volume 3)*. Springer International Publishing, Cham, 3–28. ISBN: 978-3-319-91590-6. DOI: `10.1007/978-3-319-91590-6_1`. `https://doi.org/10.1007/978-3-319-91590-6_1`.

[14] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, Andrew Y Ng, et al. 2009. Ros: an open-source robot operating system. In *ICRA workshop on open source software* number 3.2. Volume 3. Kobe, Japan, 5.

[15] Yuvraj Sahni, Jiannong Cao, and Shan Jiang. 2019. Middleware for multi-robot systems. In *Mission-oriented sensor networks and systems: Art and science*. Springer, 633–673.

[16] Toby Sharp, Cem Keskin, Duncan Robertson, Jonathan Taylor, Jamie Shotton, David Kim, Christoph Rhemann, Ido Leichter, Alon Vinnikov, Yichen Wei, Daniel Freedman, Pushmeet Kohli, Eyal Krupka, Andrew Fitzgibbon, and Shahram Izadi. 2015. Accurate, robust, and flexible real-time hand tracking. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (CHI '15). Association for Computing Machinery, Seoul, Republic of Korea, 3633–3642. ISBN: 9781450331456. DOI: `10.1145/2702123.2702179`. `https://doi.org/10.1145/2702123.2702179`.

[17] Aswath Suresh, C. P. Sridhar, Dhruv Gaba, Debrup Laha, and Siddhant Bhambri. 2019. Design and development of intelligent self-driving car using ros and machine vision algorithm. In *Robot Intelligence Technology and Applications 5*. Jong-Hwan Kim, Hyun Myung, Junmo Kim, Weiliang Xu, Eric T Matson, Jin-Woo Jung, and Han-Lim Choi, editors. Springer International Publishing, Cham, 83–98. ISBN: 978-3-319-78452-6.

[18] Emmanouil G. Tsardoulias and Pericles A. Mitkas. 2017. Robotic frameworks, architectures and middleware comparison. *CoRR*, abs/1711.06842. arXiv: `1711.06842`. `http://arxiv.org/abs/1711.06842`.

[19] Pei Xu. 2017. A real-time hand gesture recognition and human-computer interaction system. (2017). DOI: `10.48550/ARXIV.1704.07296`. `https://arxiv.org/abs/1704.07296`.