HTC Vive MeVisLab integration via OpenVR for medical applications

Jan Egger[1,2]*, Markus Gall[1], Jürgen Wallner[3], Pedro Boechat[3], Alexander Hann[4], Xing Li[5], Xiaojun Chen[5], Dieter Schmalstieg[1]

1 Institute of Computer Graphics and Vision, Graz University of Technology, Inffeldgasse 16c/II, Graz, Austria, 2 BioTechMed-Graz, Krenngasse 37/1, Graz, Austria, 3 Medical University of Graz, Department of Oral and Maxillofacial Surgery, Auenbruggerplatz 5/1, Graz, Austria, 4 Department of Internal Medicine I, Ulm University, Albert-Einstein-Allee 23, Ulm, Germany, 5 Shanghai Jiao Tong University, School of Mechanical Engineering, Shanghai, China

* egger@tugraz.at

Abstract

Virtual Reality, an immersive technology that replicates an environment via computer-simulated reality, gets a lot of attention in the entertainment industry. However, VR has also great potential in other areas, like the medical domain, Examples are intervention planning, training and simulation. This is especially of use in medical operations, where an aesthetic outcome is important, like for facial surgeries. Alas, importing medical data into Virtual Reality devices is not necessarily trivial, in particular, when a direct connection to a proprietary application is desired. Moreover, most researcher do not build their medical applications from scratch, but rather leverage platforms like MeVisLab, MITK, OsiriX or 3D Slicer. These platforms have in common that they use libraries like ITK and VTK, and provide a convenient graphical interface. However, ITK and VTK do not support Virtual Reality directly. In this study, the usage of a Virtual Reality device for medical data under the MeVisLab platform is presented. The OpenVR library is integrated into the MeVisLab platform, allowing a direct and uncomplicated usage of the head mounted display HTC Vive inside the MeVisLab platform. Medical data coming from other MeVisLab modules can directly be connected per drag-and-drop to the Virtual Reality module, rendering the data inside the HTC Vive for immersive virtual reality inspection.

Introduction

Virtual Reality (VR) places a user inside a computer-generated environment. This paradigm is becoming increasingly popular, due to the fact that computer graphics have progressed to a point where the images are often indistinguishable from the real world. The computer-generated images previously presented in movies, games and other media are now detached from the physical surroundings and presented in new immersive head-mounted displays (HMD), like the Oculus Rift, the PlayStation VR or the HTC Vive (Fig 1). Virtual reality not only places a user inside a computer-generated environment – it is able to completely immerse a user in a virtual world, removing any restrictions on what a user can do or experience [1–3]. Beside movies, games, and other media, the medical area has great potential for the newly VR devices,

Fig 1. Illustration of a person using the HTC Vive and controllers (photo filtered with PRISMA 2.2.1 under an iPhone 6s running iOS 9.3.3).

https://doi.org/10.1371/journal.pone.0173972.g001

because they are also now able to process and display high-resolution medical image data acquired with modern CT (Computed Tomography) and MRI (Magnetic Resonance Imaging) scanners [4]. Examples are surgery trainers and simulators, and preoperative surgical planning [5], [6]. Others already working in the area of medical Virtual Reality are Nunnerley et al. [7], who tested the feasibility of an immersive 3D virtual reality wheelchair training tool for people with spinal cord injury. They designed a wheelchair training system that used the Oculus Rift headset and a joystick. Newbutt et al. [8] studied the usage of the Oculus Rift in autism patients. Their study explores the acceptance, willingness, sense of presence and immersion of participants with autism spectrum disorder (ASD). The examination of digital pathology slides with the virtual reality technology and the Oculus Rift has been explored by Farahani et al. [9]. They applied the Oculus Rift and a Virtual Desktop software to review lymph node cases for digital pathology. A usability comparison between HMD (Oculus Rift) and stereoscopic desktop displays (DeepStream3D) in a VR environment with pain patients has been performed by Tong et al. [10]. Twenty chronic pain patients assessed the severity of physical discomforts by trying both displays, while watching a virtual reality pain management demo in a clinical setting. An interactive 3D virtual anatomy puzzle for learning and simulation has been designed and tested by Messier and collages [11]. The virtual anatomy puzzle is supposed to help users to learn the anatomy of various organs and systems by manipulating virtual 3D data. It was implemented with an Oculus Rift. A computer-based system, which can record a surgical procedure with multiple depth cameras and reconstruct in three dimensions the dynamic geometry of the actions and events that occur during the procedure, has been introduced by Cha et al. [12]. Equipped with a virtual reality headset, such as the Oculus Rift, the user was able to

walk around the reconstruction of the procedure room, while controlling the playback of the recorded surgical procedure with simple VCR controls (e.g., play, pause, rewind, and fast forward). Xu et al. [13] studied the accuracy of the Oculus Rift during cervical spine mobility measurement by designing a virtual reality environment to guide participants to perform certain neck movements. Subsequently, the cervical spine kinematics was measured by both the Oculus Rift tracking system and a reference motion tracker. The Oculus Rift has been applied by Kim et al. [14] as a cost-effective tool for studying visual-vestibular interactions in self-motion perception. The vection strength has been measured in three conditions of visual compensation for head movement (1. compensated, 2. uncompensated and 3. inversely compensated). King [15] developed an immersive VR environment for diagnostic imaging. The environment consisted of a web server acquiring data from volumes loaded within the 3D Slicer platform [16] and forwarding them to a Unity application (https://unity3d.com/) to render the scene for the Oculus Rift. However, to the best of the authors' knowledge, no work described the integration the HTC Vive into MeVisLab (http://www.mevislab.de/) platform [17] yet. We developed and implemented a new module for the medical prototyping platform MeVisLab that provides an interface via the OpenVR library to head mounted displays, enabling the direct and uncomplicated usage of the HTC Vive in medical applications. Unlike the Oculus Rift, the room-scale tracking offered by the HTC Vive allows walking around virtual objects, which enables a more advanced immersion and inspection.

This contribution is organized as follows: Section 2 introduces details of the methods, Section 3 presents experimental results and Section 4 concludes the paper and gives an outlook on future work.

Methods

In this section, the materials and methods that have been used for the integration of the HTC Vive into the MeVisLab environment via the OpenVR library are introduced.

Datasets

For testing and evaluating our software integration, we used multiple high-resolution CT (Computed Tomography) scans from the clinical routine. The resolution of the scans consisted of 512x512 voxels in the x- and the y-direction with an additional few hundred slices in the z-direction. The scans varied in anatomy and location of pathology, including datasets from patient skulls with cranial defects. In Fig 2, 3D visualizations of patient skulls with cranial defects on the left (left) and right side (right) are shown. The medical scans are freely available for download and usage in own research projects, however, we kindly ask users cite our work [18], [19]. All relevant data are hosted at the public repository Figshare. Please see data hosted at Figshare at the following URL:

https://figshare.com/articles/Cranial_Defect_Datasets/4659565.

Note that the datasets from our clinical partners have not been altered or downsampled in any way. Hence, we assess the visual quality and evaluate the frames per second (fps) when displaying original sized scans inside the HTC Vive.

MeVisLab

This paragraph describes the medical imaging platform MeVisLab (Medical Visualization Laboratory), which includes a software development kit (SDK) used to interface with the HTC Vive. The MeVisLab platform provides basic and advanced algorithms for medical image processing and visualization. It also includes an environment for programming new modules in C++, as well as an environment for implementing user interfaces with MDL script (MeVis Description
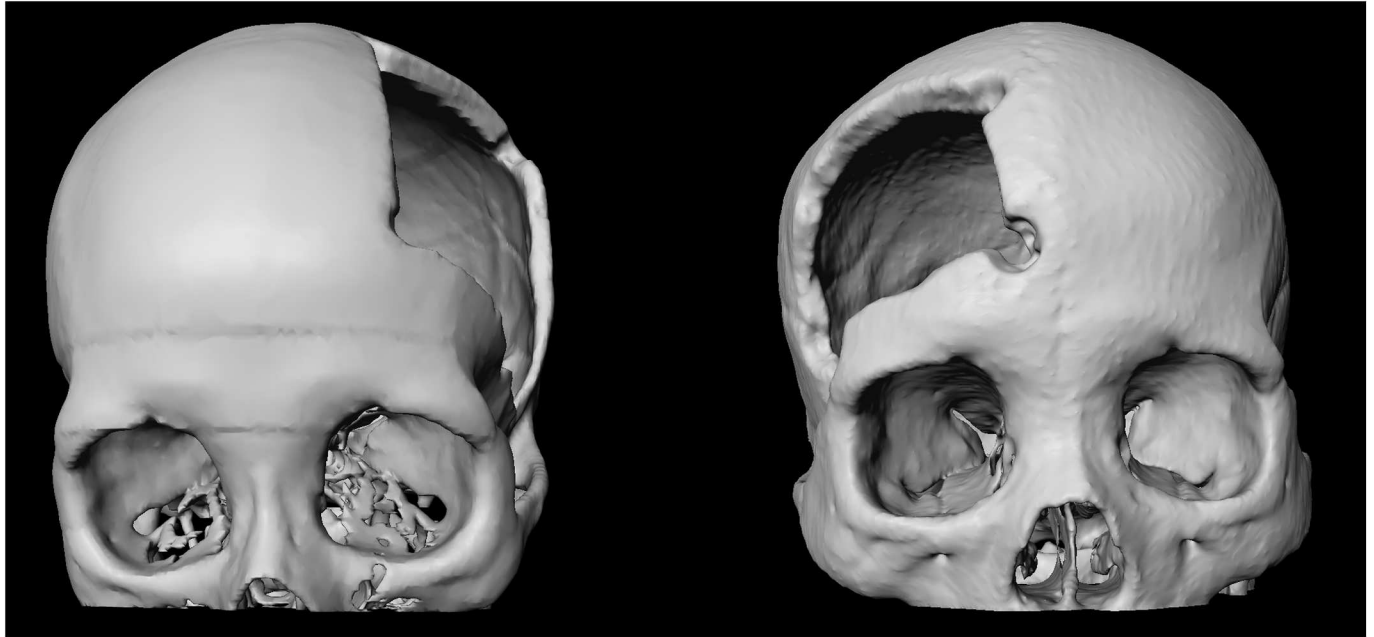
Fig 2. 3D visualization of a patient skull with a cranial defects on the left side (left) and 3D visualization of a patient skull with a cranial defects on the right side (right).

Language). MeVisLab allows to construct dataflow networks using a variety of image processing modules. Developers can employ visual programming by compositing pre-installed modules into complex networks using the graphical user interface. Alternatively, the Python scripting language can be used. Finally, C++ programmers generate individually designed modules. Furthermore, it is possible to design user interfaces with MDL script, hiding the complexity of user-interface programming from non-technical users. Fig 3 shows a simple MeVisLab network with a small amount of modules: a module for loading the image data, an algorithm module, which applies a threshold function in this example, and a viewer module for the visualization.

In summary, the MeVisLab network concept is based on a graphical representation of modules with specific functions for image processing. MeVisLab uses three different types of modules (Fig 4). ML modules (blue) are processing modules, Open Inventor modules (green) provide scene graphs in 3D, and Macro modules (brown) combine other modules. Module connectors located on the bottom of the module are inputs, and connectors located at the top of the modules are outputs. The shape of the inputs and outputs define the connection types: A triangle for ML images, a rectangle for a base object indicating pointers to data structures, and a half circle for an Open Inventor scene (see Figs 3 and 4). A data transmission between connectors is enabled by a connection, visually represented by a blue line, like shown in Fig 3 on the right side. In addition to these data connections, a parameter connection can also be established. This enables the connection of single parameters between modules and is indicated by a two-sided arrow. For further information, please see the MeVisLab Reference manual (last accessed in January 2017):

http://mevislabdownloads.mevis.de/docs/current/MeVisLab/Resources/Documentation/Publish/SDK/MeVisLabManual/index.html.

HTC Vive

This paragraph states the technical specifications of the VR headset HTC Vive, which is developed by HTC and Valve Corporation and was released in April 2016. In contrast to the current
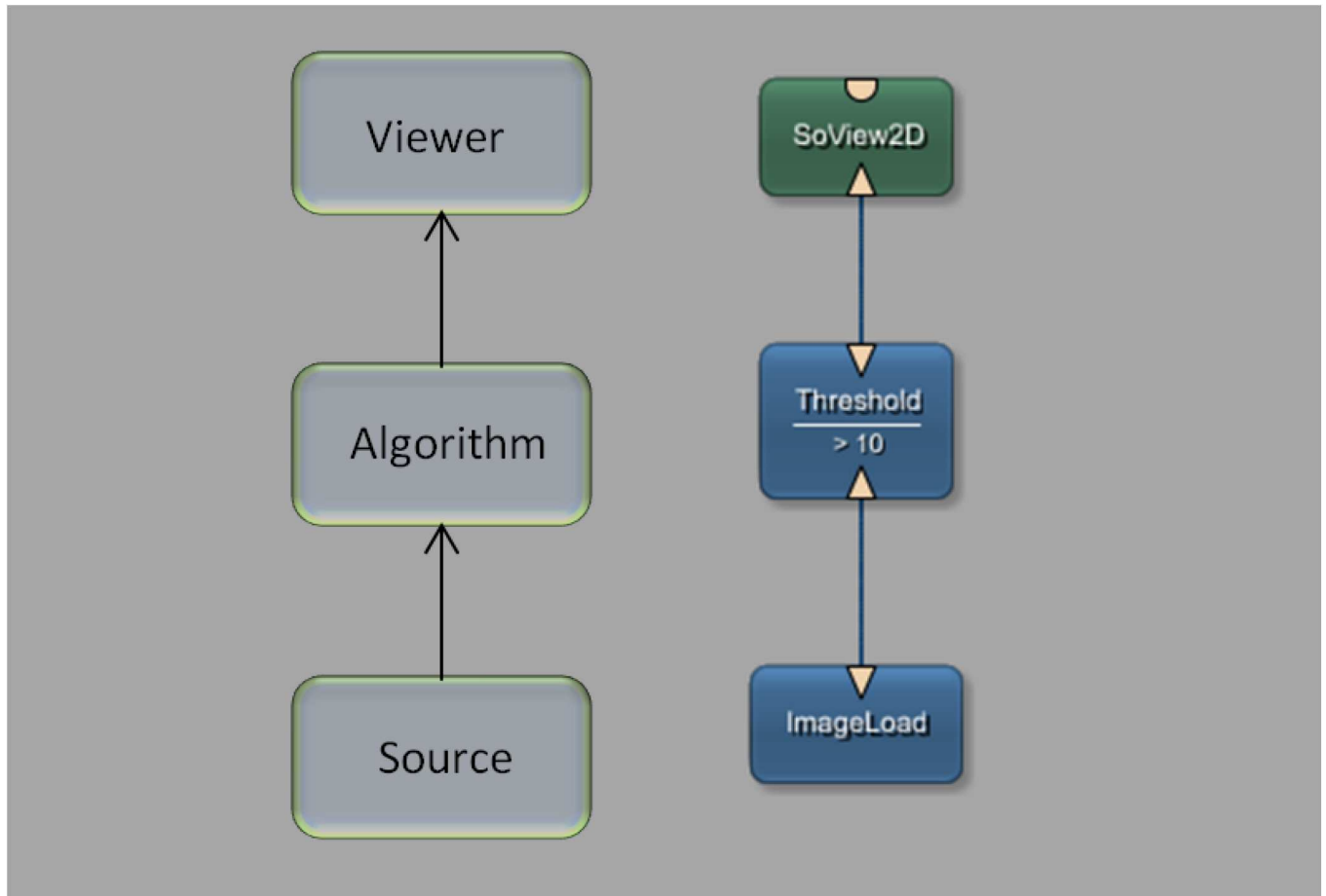
Fig 3. Basic module processing pipeline with a network of three modules: An image source for loading the data set into the network, an algorithm module to process the data and a viewer module that enables the visualization of the image.

version of the Oculus Rift, the HTC Vive is designed to turn a room into 3D space. Two stationary reference units track the user's head and handheld controllers, while the user is free to walk around and manipulate virtual objects. The HTC Vive uses an organic light-emitting diode (OLED) display and provides a combined resolution of 2160x1200 (1080x1200 per eye) with a refresh rate of 90 Hz and a field of view (FOV) of about 110 degrees. The head mounted display weights about 555 grams and has HDMI 1.4, DisplayPort 1.2 and USB 2.0 connections.
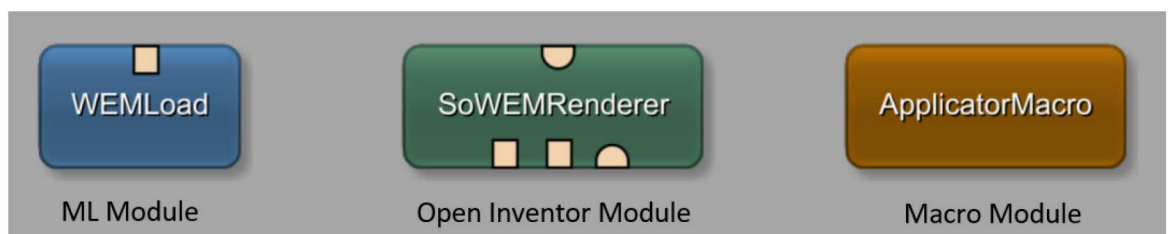

Fig 4. Types of modules under MeVisLab: ML modules (blue) are page/-patch based and demand-driven, Open Inventor modules (green) provide scene graphs in 3D and Macro modules (brown) consist of a combination of modules.

Moreover, the HTC Vive has a 3.5 mm audio jack for headphones and a built-in microphone. Finally, the HTC Vive has a front-facing camera for blending real-world elements into the virtual world. For further information, please see the HTC Vive website (last accessed in January 2017):

https://www.htcvive.com

OpenVR

This paragraph gives a short description of OpenVR, which is a software development kit and application programming interface developed by Valve. The C++ library supports the HTC Vive and other VR headsets via the SteamVR software platform (http://steamvr.com). Thus, the OpenVR API provides a way to connect and interact with VR displays without relying on a specific hardware vendor's SDK. OpenVR is implemented as a set of C++ interface classes with virtual functions supposed to support also future versions of the hardware. For further information, please see the corresponding GitHub repository (last accessed in January 2017):

https://github.com/ValveSoftware/openvr

The current OpenVR GitHub repository comes for Microsoft Windows, Linux and MacOS. Moreover, it includes examples in C++ for Visual Studio from Microsoft.

Workflow

Fig 5 shows a high-level workflow diagram of the communication and interaction between MeVisLab and the HTC Vive device via OpenVR. In doing so, the MeVisLab platform provides several modules to import and load medical data, e.g., in DICOM format (http://dicom.nema.org/). The medical image data can be processed and visualized with a variety of modules under MeVisLab. Options include the visualization in 2D and 3D, or medical image analysis algorithms, like the segmentation of anatomical or pathological structures. Moreover, MeVisLab modules that derive directly from ITK (Insight Segmentation and Registration Toolkit, https://itk.org/) and VTK (The Visualization Toolkit, http://www.vtk.org/) can be applied.

Our new HTCVive module communicates with the HTC Vive via OpenVR. However, our MeVisLab module may also be able to communicate with another VR device, like the Oculus Rift (https://www.oculus.com/), because the OpenVR API provides a way to connect and interact with VR displays without relying on a specific hardware vendor's SDK. The HTCVive module transfers the image date to the HMD and receives the position and the orientation of the HMD in real-world to MeVisLab. The position and orientation enable further visualizations under MeVisLab, like rendering the user's view in a second 3D viewer on a conventional screen, so users without HMD can see the VR view, too. For our implementation, we followed the hellovr_opengl source code example from OpenVR (https://github.com/ValveSoftware/openvr/tree/master/samples/hellovr_opengl). In doing so, using an initialization method to start the SteamVR runtime and render a distortion view of the right and left eye.

Network

Fig 6 shows the overall MeVisLab network with our HTCVive module (blue) in the lower left surrounded by a yellow rectangle. In this example network, the (medical) data is imported via the WEMLoad module (here named DataLoad) and directly passed via its output (rectangle) to the HTCVive module (rectangle input at the bottom of the HTC Vive module). Furthermore, the loaded data is passed via a WEMModify module (blue, lower right side), an SoWEMRenderer module (green, right side in the middle) and an SoSeparator module (green, upper right side) to another SoSeparator module (named 3DUserView at the top). The window on the right side belongs to the 3DUserView module at the top and displays what the user
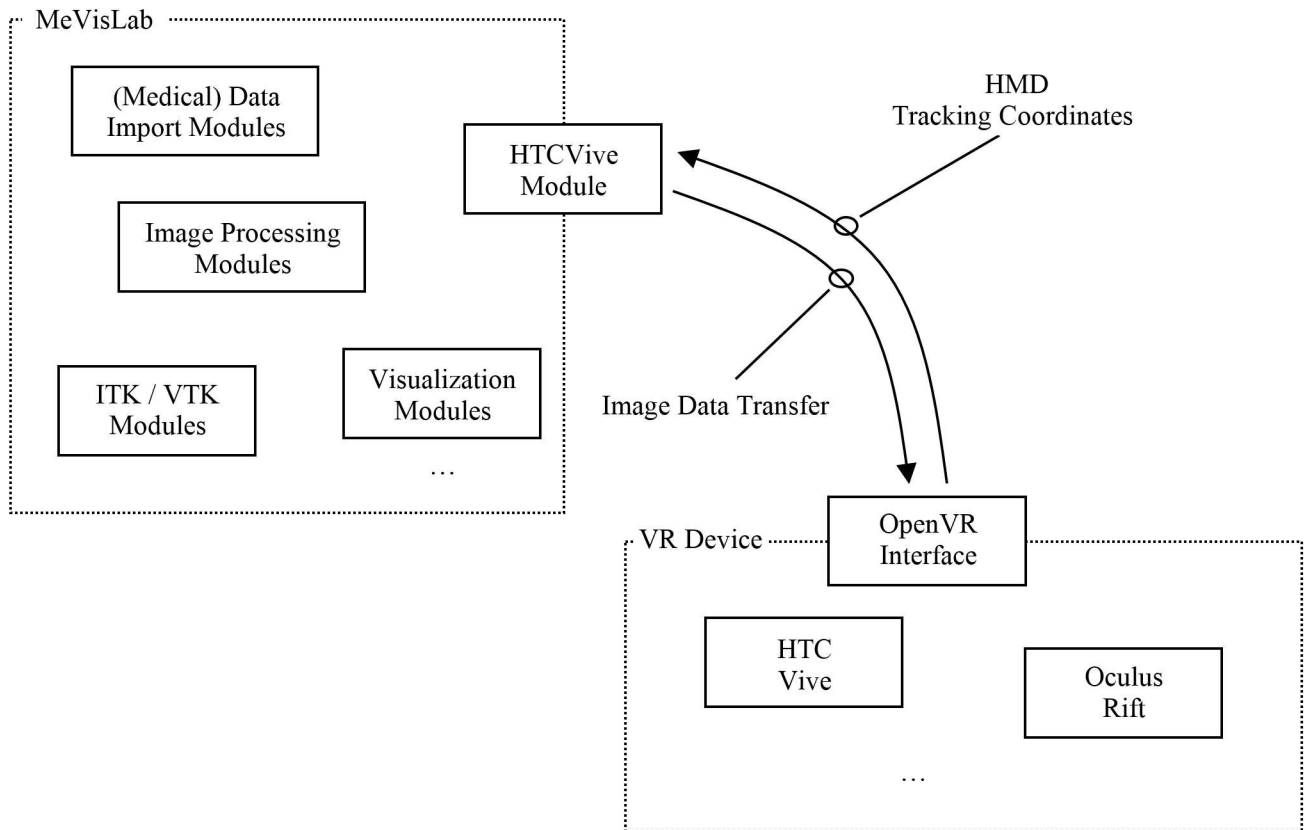
Fig 5. High level workflow diagram that demonstrates the communication and interaction between the MeVisLab platform and the HTC Vive device via the OpenVR library.

wearing the HTC Vive headset sees. Note that the HMD tracking coordinates are transferred via direct parameter connections in our example. The remaining modules in the network example from Fig 6 are mainly arithmetic and matrices modules to decompose (Decompose-Matrix and DecomposeMatrix2) and compose (ComposeMatrix) the rotation and translation for a correct visualization. The interface and parameters of our HTCVive module are shown on the left side (Panel HTCVive). Finally, a status window of the SteamVR is presented on the lower right corner (black), indicating that the HMD, the two HTC Vive controllers and the two HTC Vive Lighthouse base stations are ready and running (green).

Results

The aim of this contribution was to investigate the feasibility of using the Virtual Reality device HTC Vive under the medical prototyping platform MeVisLab. We present the direct rendering and visualization of (medical) image data in the head mounted device using the publicly available OpenVR library. The software integration was achieved under Microsoft Windows 8.1 with MeVisLab version 2.8.1 (21-06-2016) for Visual Studio 2015 x64 (http://www.mevislab.de/download/) and the OpenVR SDK version 1.0.2 (https://github.com/ValveSoftware/openvr). The MeVisLab module was implemented as an image processing module in C++ using the MeVisLab Project Wizard and the Microsoft Visual Studio 2015 Community Edition. The HTC Vive MeVisLab module (HTCVive) has an input for the medical image data, which is transferred and displayed in the HTC Vive. Additionally, the HTCVive module provides the data at
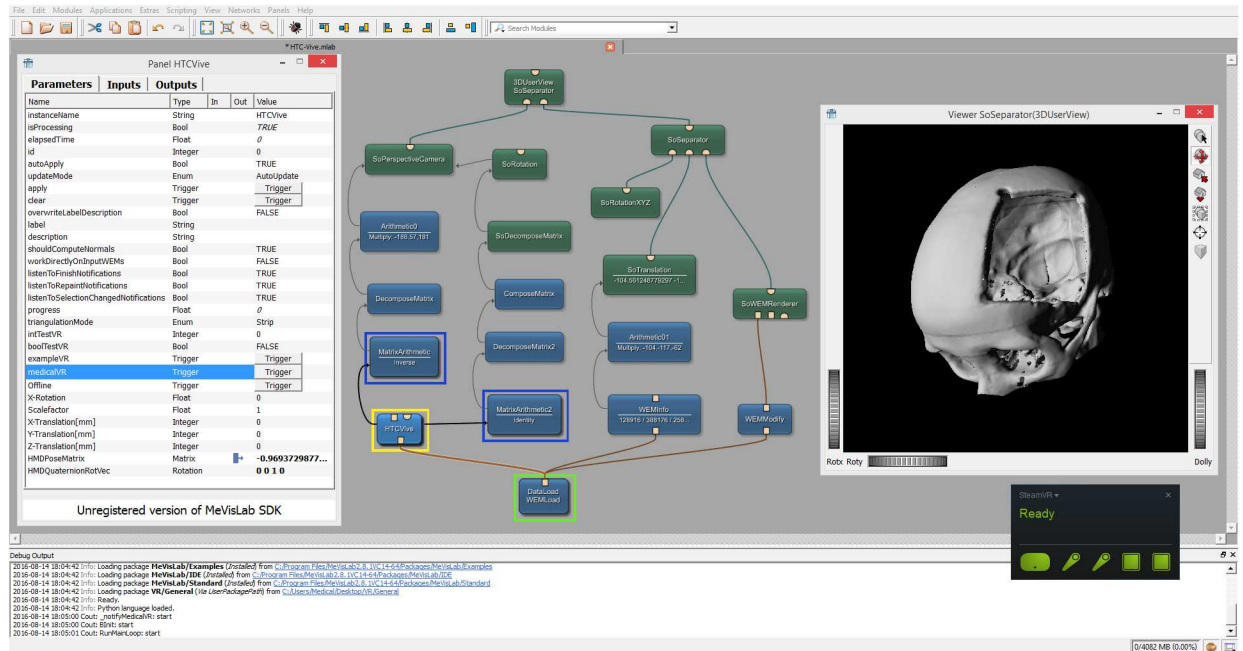
Fig 6. Screenshot of the comprehensive MeVisLab network with the HTCVive module (blue, lower left corner surrounded by a yellow rectangle) and its interface and parameters on the left side (Panel HTCVive). Additionally, the module provides the HMD tracking coordinates of the HTC Vive, which can be used to have the same position and viewing direction as the actual user wearing the HTC Vive in an extra 3D view (Viewer SoSeparator(3DUserView) on the right side).

https://doi.org/10.1371/journal.pone.0173972.g006

an output (rectangle) for visualization in a standard undistorted 3D view, and the module provides the HMD tracking coordinates of the HTC Vive (Fig 6, right window). Finally, an extra window can be created for a distorted view of the virtual reality input for the left and the right eye (Fig 7 without texture and Fig 8 with texture). For an evaluation, we tested several medical datasets, like patient skulls with cranial defects and scans from gastrointestinal tracts for usage in virtual colonoscopy (Fig 9). As a hardware setting, we applied two computers: a desktop PC and a MacBook Pro laptop, which both had Windows 8.1 Enterprise installed. The hardware
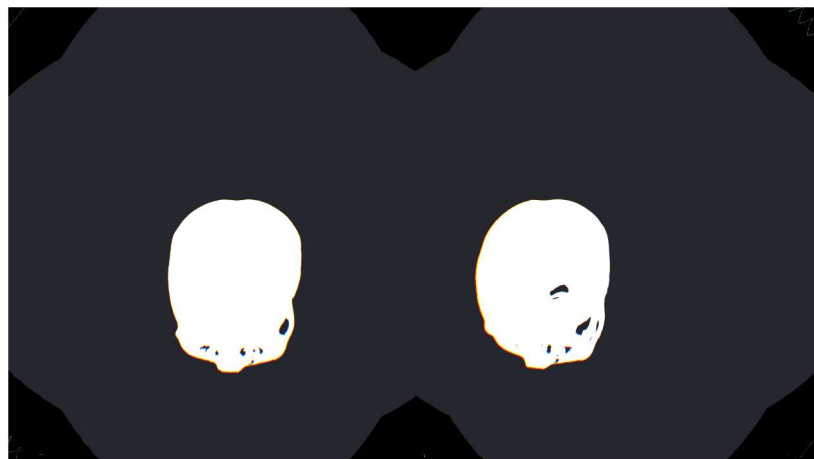

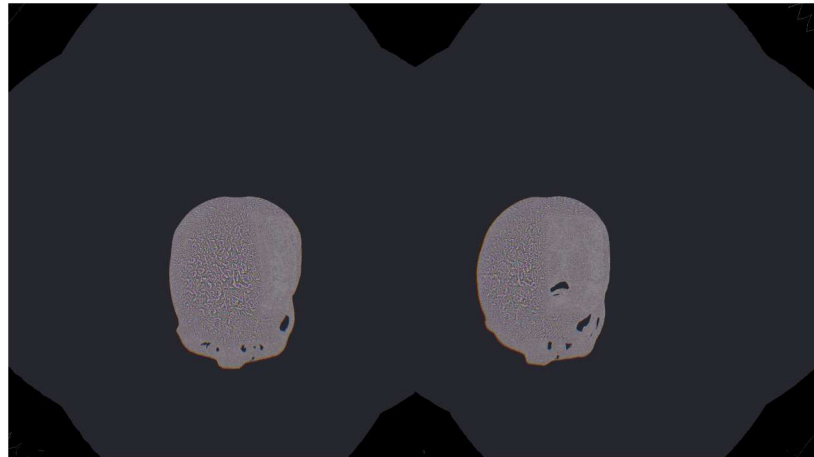**Fig 7. The distorted view of the VR input for the right and left eye without texture.**

https://doi.org/10.1371/journal.pone.0173972.g007

Fig 8. The distorted view of the VR input for the right and left eye with texture.

https://doi.org/10.1371/journal.pone.0173972.g008

Fig 9. The figure shows a scan of the gastrointestinal tract that has been imported and displayed in the HTC Vive for usage in virtual colonoscopy.

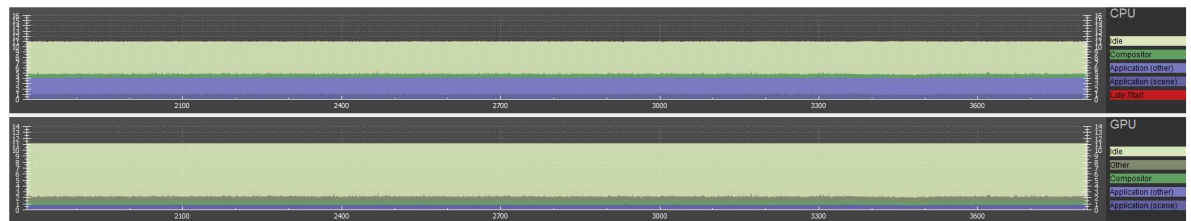https://doi.org/10.1371/journal.pone.0173972.g009

Fig 10. Frame timing result diagrams of the CPU (top) and the GPU (bottom) for a Windows desktop PC with Intel Core i7-3770 CPU @ 3.40GHz, 16 GB RAM and a NVIDIA GeForce GTX 970 graphic card.

configuration of the desktop PC consisted of an Intel Core i7-3770 CPU @ 3.40GHz, 16 GB RAM and a NVIDIA GeForce GTX 970 graphics card, and the hardware configuration of the laptop computer consisted of an Intel Core i7-4850HQ CPU @ 2.30GHz, 16 GB RAM and a NVIDIA GeForce GT 750M graphics card. Figs 10 and 11 show the frame timing results of both configurations for the CPU (top) and the GPU (bottom). The frame timing results come directly from the SteamVR and can be activated and displayed via the SteamVR status window. As seen, the default view splits the CPU and GPU performance in a pair of stacked graphs: The blue sections represent the time spent by the application, which is further spitted between application-scene and application-other. Application-scene is the amount of work performed in between, when WaitGetPoses (returns pose(s) to use to render scene) returns and the second eye texture is submitted. Application-other is any time spent after this for rendering the application's companion window, etc. Note that the CPU timing does not capture any parallel work being performed, for example, on the application's main thread. The brown section (other) in the GPU timing reflects any GPU bubbles, context switching overhead, or GPU work from other application getting scheduled in between other segments of work for that frame (to examine this in more detail requires the use of gpuview). For more information, please see also (last access January 2017):

https://developer.valvesoftware.com/wiki/SteamVR/Frame_Timing.

As shown in the frame timing diagram in Fig 10, a sufficient framerate for the desktop PC configuration could be achieved, because ten frames per second are already considered as real-time or interactive in computer graphics applications [20]. Hence, the medical datasets could be displayed very pleasant to the human eye inside the HTC Vive device. In contrast, the framerates on the laptop were in general not sufficient, resulting in a flickering inside the HTC Vive. For a short period of time, this is acceptable as a temporary (mobile) solution, but for a longer working period, this is currently definitely not suitable. However, the laptop and its
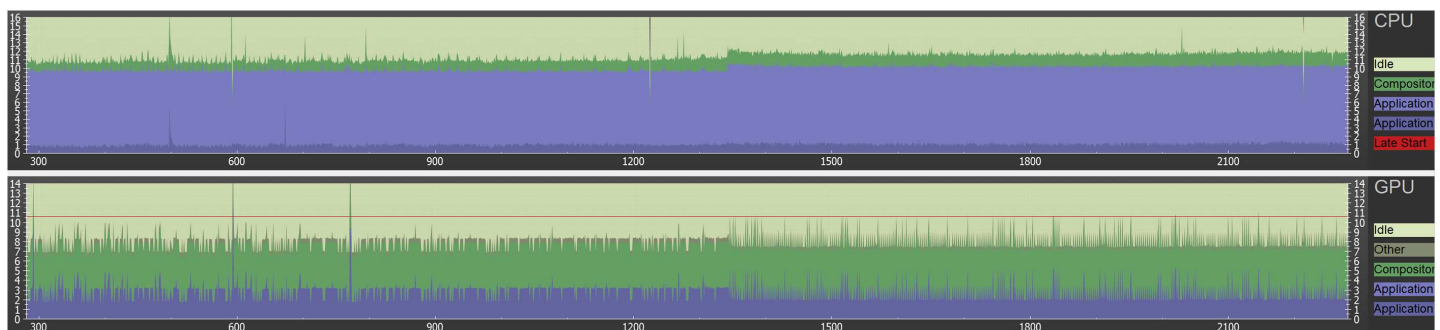

Fig 11. Frame timing result diagrams of the CPU (top) and the GPU (bottom) for a Windows laptop with Intel Core i7-4850HQ CPU @ 2.30GHz, 16 GB RAM and a NVIDIA GeForce GT 750M graphic card.

hardware is also not considered as HTC Vive ready, but we expect to see soon laptops that can handle the HTC Vive requirements, so it can also be used as a portable application.

Discussion

In this open access article, we demonstrated the successful integration and evaluation of the HTC Vive headset into the research prototyping platform MeVisLab via the OpenVR library. OpenVR is a software development kit and application programming interface developed by Valve (http://www.valvesoftware.com/) for supporting the SteamVR (HTC Vive) and other VR headsets. Hence, our integration is device and vendor independent and can also be used with other devices. The proposed software integration has been carried out with the C++ implementation of a MeVisLab image processing module that provides inputs for medical datasets in different formats. Thus, our module is easy to use and can be added to existing MeVisLab networks or applied in new ones. Per drag and drop, data outputs under MeVisLab can be directly connected to an input of our module to transfer the data into the HTC Vive headset and display it there. Furthermore, dataset operations and manipulations, like translation, rotation, segmentations, etc. that have been performed in advance with corresponding MeVisLab modules will also directly be shown inside the headset. Highlights of the proposed contribution are as follows:

- Successful integration of OpenVR into MeVisLab

- Solution enables MeVisLab networks to connect to VR headsets

- Real-time visualization of medical data in VR under MeVisLab

- Proof of concept evaluation with the HTC Vive headset

- HTC Vive MeVisLab module can be added to existing MeVisLab networks.

Areas of future work include the evaluation of our integration with a greater amount of medical data and formats. Although if we did not encounter motion sickness in our tests, this might be an issue for some users and needs to be explored systematically. In addition, we plan a MeVisLab module communicating with the Oculus Rift head mounted display, or, as an alternative, the extension of our VR module supporting also the Oculus Rift, when fully supported by the OpenVR library, and the evaluation on non-Windows-Setups like Linux or Mac OS. Moreover, we work on applying our solution to support computer-aided reconstructions of facial defects [21], [22] with photorealistic rendering in VR [23]. This will enable an even more realistic assessment of pre-operative planning results. Furthermore, we investigate options for performing image-guided therapy tasks inside VR, for example, the planning of facial interventions using so called miniplates [24] and virtual stent simulations to treat aneurysms [25–29]. Capturing and reconstruction of 3D models of tumors in the gastrointestinal tract during diagnostic endoscopic procedures will ease the planning for the endoscopic removal without the need for prolonged or repeated anesthesia [30]. Using segmentation algorithms implemented in MeVisLab [31–33] will help identify areas of tumor spread in the 3D model due to surface characteristics. Interactive segmentation [34], [35] might even be done with the HTC Vive controllers. Additionally, reviewing 3D models of tumors in virtual space will provide a platform for training to improve recognition of these lesions from different angles. We would like to pursue the development of an Augmented Reality (AR) module for MeVisLab, to support novel devices like the Microsoft HoloLens, because optical see-through head-mounted displays seem to be very promising during surgical navigation [36]. Last but not least, we plan to integrate OpenVR into other research platforms, like OsiriX [37], 3D

Slicer [38] or the two Medical Imaging (Interaction) Toolkits (MITK) from Beijing in China (www.mitk.net) [39] and Heidelberg in Germany (www.mitk.org) [40], respectively.

Acknowledgments

Author Contributions

Conceptualization: JE MG.

Data curation: JW XC XL.

Formal analysis: JE MG.

Funding acquisition: JE DS XC.

Investigation: JE MG.

Methodology: JE MG.

Project administration: JE DS.

Resources: JE MG JW PB AH XL XC DS.

Software: JE MG PB.

Supervision: JE DS.

Validation: JE MG.

Visualization: JE MG.

Writing – original draft: JE.

Writing – review & editing: JE DS.

References

1. Schmalstieg D, Höllerer T. Augmented Reality: Principles and Practice. Addison-Wesley Professional; 1st ed., 2016; 1–528.

2. Bowman DA, McMahan RP. Virtual reality: How much immersion is enough? IEEE Computer 2007; 40 (7): 36–43.

3. Rheingold H. Virtual Reality: Exploring the Brave New Technologies. Simon & Schuster Adult Publishing Group 1991; 1–384.

4. McCloy R, Stone R. Virtual reality in surgery. BMJ 2001; 323(7318): 912–915. PMID: 11668138

5. Reitinger B, Bornik A, Beichel R, Schmalstieg D. Liver surgery planning using virtual reality. IEEE Comput Graph Appl. 2006; 26(6): 36–47. PMID: 17120912

6. Gor M, McCloy R, Stone R, Smith A. Virtual reality laparoscopic simulator for assessment in gynaecology. BJOG 2003; 110(2): 181–7. PMID: 12618163

7. Nunnerley J, Gupta S, Snell D, King M. Training wheelchair navigation in immersive virtual environments for patients with spinal cord injury—end-user input to design an effective system. Disabil Rehabil Assist Technol. 20016; 4:1–7.

8. Newbutt N, Sung C, Kuo HJ, Leahy MJ, Lin CC, Tong B. Brief Report: A Pilot Study of the Use of a Virtual Reality Headset in Autism Populations. J Autism Dev Disord. 2016; 46(9): 3166–76. https://doi.org/10.1007/s10803-016-2830-5 PMID: 27272115

9. Farahani N, Post R, Duboy J, Ahmed I, Kolowitz BJ, Krinchai T, et al. Exploring virtual reality technology and the Oculus Rift for the examination of digital pathology slides. J Pathol Inform. 2016; 7: 22. https://doi.org/10.4103/2153-3539.181766 PMID: 27217972

10. Tong X, Gromala D, Gupta D, Squire P. Usability Comparisons of Head-Mounted vs. Stereoscopic Desktop Displays in a Virtual Reality Environment with Pain Patients. Stud Health Technol Inform. 2016; 220: 424–31. PMID: 27046617

11. Messier E, Wilcox J, Dawson-Elli A, Diaz G, Linte CA. An Interactive 3D Virtual Anatomy Puzzle for Learning and Simulation—Initial Demonstration and Evaluation. Stud Health Technol Inform. 2016; 220: 233–40. PMID: 27046584

12. Cha YW, Dou M, Chabra R, Menozzi F, State A, Wallen E, et al. Immersive Learning Experiences for Surgical Procedures. Stud Health Technol Inform. 2016; 220: 55–62. PMID: 27046554

13. Xu X, Chen KB, Lin JH, Radwin RG. The accuracy of the Oculus Rift virtual reality head-mounted display during cervical spine mobility measurement. J Biomech. 2015; 48(4): 721–4. https://doi.org/10.1016/j.jbiomech.2015.01.005 PMID: 25636855

14. Kim J, Chung CY, Nakamura S, Palmisano S, Khuu SK. The Oculus Rift: a cost-effective tool for studying visual-vestibular interactions in self-motion perception. Front Psychol. 2015; 6: 248. https://doi.org/10.3389/fpsyg.2015.00248 PMID: 25821438

15. King F. An immersive virtual reality environment for diagnostic imaging. Thesis (Master, Computing), Queen's University, Kingston, Ontario, Canada, 2015; 1–60.

16. Egger J, Kapur T, Nimsky C, Kikinis R. Pituitary Adenoma Volumetry with 3D Slicer. PLoS ONE 7(12): e51788. https://doi.org/10.1371/journal.pone.0051788 PMID: 23240062

17. Egger J, Tokuda J, Chauvin L, Freisleben B, Nimsky C, Kapur T, et al. Integration of the OpenIGTLink network protocol for image-guided therapy with the medical platform MeVisLab. Int J Med Robot. 2012; 8(3): 282–90. https://doi.org/10.1002/rcs.1415 PMID: 22374845

18. Gall M, Li X, Chen X, Schmalstieg D, Egger J. Cranial Defect Datasets. ResearchGate 2016.

19. Gall M, Li X, Chen X, Schmalstieg D, Egger J. Computer-Aided Planning and Reconstruction of Cranial 3D Implants. IEEE Engineering in Medicine and Biology Society 2016; 1179–1183.

20. Ai Z, Evenhouse R, Leigh J, Charbel F, Rasmussen M. Cranial implant design using augmented reality immersive system. Stud Health Technol Inform. 2007; 125: 7–12. PMID: 17377223

21. Gall M, Li X, Chen X, Schmalstieg D, Egger J. Computer-Aided Planning of Cranial 3D Implants. Int J CARS 11 2016; (Suppl 1): S241.

22. Li X, Xu L, Zhu Y, Egger J, Chen X. A semi-automatic implant design method for cranial defect restoration. Int J CARS 11 2016; (Suppl 1): S241–S243.

23. Vogelreiter P, Wallner J, Reinbacher K, Schwenzer-Zimmerer K, Schmalstieg D, Egger J. Global Illumination Rendering for High-Quality Volume Visualization in the Medical Domain. Face 2 Face – Science meets Arts, Medical University of Graz, Department of Maxillofacial Surgery, 2015.

24. Gall M, Wallner J, Schwenzer-Zimmerer K, Schmalstieg D, Reinbacher K, Egger J. Computer-aided Reconstruction of Facial Defects. IEEE Engineering in Medicine and Biology Society 2016; Posters, Orlando, FL, USA.

25. Egger J, Mostarkić Z, Maier F, Kaftan JN, Grosskopf S, Freisleben B. Fast self-collision detection and simulation of bifurcated stents to treat abdominal aortic aneurysms (AAA). 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society 2007; 6231–6234.

26. Lu J, Egger J, Wimmer A, Großkopf S, Freisleben B. Detection and visualization of endoleaks in CT data for monitoring of thoracic and abdominal aortic aneurysm stents. SPIE Medical Imaging 2008; 69181F.

27. Renapurkar RD, Setser RM, O'Donnell TP, Egger J, Lieber ML, Desai MY, et al. Aortic volume as an indicator of disease progression in patients with untreated infrarenal abdominal aneurysm. Eur J Radiol. 2012; 81: e87–93. https://doi.org/10.1016/j.ejrad.2011.01.077 PMID: 21316893

28. Greiner K, Egger J, Großkopf S, Kaftan JN, Dörner R, Freisleben B. Segmentation of aortic aneurysms in CTA images with the statistic approach of the active appearance models. Bildverarbeitung für die Medizin (BVM) 2008; 51–55.

29. Egger J, Mostarkic Z, Grosskopf S, Freisleben B (2007) Preoperative Measurement of Aneurysms and Stenosis and Stent-Simulation for Endovascular Treatment. IEEE International Symposium on Biomedical Imaging (ISBI): From Nano to Macro 2007; 392–395.

30. Alcantarilla PF, Bartoli A, Chadebecq F, Tilmant C, Lepilliez V. Enhanced imaging colonoscopy facilitates dense motion-based 3D reconstruction. Conf Proc IEEE Eng Med Biol Soc. 2013; 7346–9.

31. Egger J, Kapur T, Dukatz T, Kolodziej M, Zukic D, Freisleben B, et al. Square-Cut: A Segmentation Algorithm on the Basis of a Rectangle Shape. PLoS ONE 7(2): e31064. https://doi.org/10.1371/journal.pone.0031064 PMID: 22363547

32. Schwarzenberg R, Freisleben B, Nimsky C, Egger J. Cube-Cut: Vertebral Body Segmentation in MRI-Data through Cubic-Shaped Divergences. PLoS ONE 9(4): e93389. https://doi.org/10.1371/journal.pone.0093389 PMID: 24705281

33. Egger J. PCG-Cut: Graph Driven Segmentation of the Prostate Central Gland. PLoS ONE 8(10): e76645. https://doi.org/10.1371/journal.pone.0076645 PMID: 24146901

34. Egger J, Lüddemann T, Schwarzenberg R, Freisleben B, Nimsky C. Interactive-cut: real-time feedback segmentation for translational research. Comput. Med. Imaging Graphics 2014; 38(4): 285–295.

35. Egger J, Busse H, Brandmaier P, Seider D, Gawlitza M, Strocka S, et al. Interactive Volumetry of Liver Ablation Zones. Sci Rep. 2015; 5: 15373. https://doi.org/10.1038/srep15373 PMID: 26482818

36. Chen X, Xu L, Wang Y, Wang H, Wang F, Zeng X, et al. Development of a surgical navigation system based on augmented reality using an optical see-through head-mounted display. J Biomed Inform. 2015; 55: 124–31. https://doi.org/10.1016/j.jbi.2015.04.003 PMID: 25882923

37. Rosset A, Spadola L, Ratib O. OsiriX: an open-source software for navigating in multidimensional DICOM images. J Digit Imaging 2004; 17(3): 205–16. https://doi.org/10.1007/s10278-004-1014-6 PMID: 15534753

38. Egger J, Kapur T, Fedorov A, Pieper S, Miller JV, Veeraraghavan H, et al. GBM Volumetry using the 3D Slicer Medical Image Computing Platform. Sci Rep. 2013; 3: 1364. https://doi.org/10.1038/srep01364 PMID: 23455483

39. Tian J, Xue J, Dai Y, Chen J, Zheng J. A novel software platform for medical image processing and analyzing. IEEE Trans Inf Technol Biomed. 2008; 12(6): 800–12. https://doi.org/10.1109/TITB.2008.926395 PMID: 19000961

40. Klemm M, Kirchner T, Gröhl J, Cheray D, Nolden M, Seitel A, et al. MITK-OpenIGTLink for combining open-source toolkits in real-time computer-assisted interventions. Int J Comput Assist Radiol Surg. 2017; 12(3): 351–361. https://doi.org/10.1007/s11548-016-1488-y PMID: 27687984