SoftwareX 11 (2020) 100426

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

Connecting the CoppeliaSim robotics simulator to virtual reality

Boris Bogaerts *, Seppe Sels, Steve Vanlanduit, Rudi Penne

Faculty of Applied Engineering, University of Antwerp, Antwerp, Belgium

ARTICLE INFO

Article history:
Received 22 July 2019
Received in revised form 8 January 2020
Accepted 18 February 2020

Keywords:
Virtual reality
Robot simulation
Human–robot interaction
Prototyping
Interface

ABSTRACT

The CoppeliaSim VR Toolbox provides a set of tools to experience CoppeliaSim robot simulation software in Virtual Reality and to return user interactions. Its primary focus is to create a platform that enables the fast prototyping and verification of robotic systems. Moreover, the generality of the toolbox ensures that it can be valuable in other contexts like robotics education, human–robot interaction or reinforcement learning. The software is designed to have a low entry threshold for moderately complex use cases, but can be extended to perform very complex visualizations for more experienced users.

Code metadata

Current code version	v2.2
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX_2019_247
Legal Code License	BSD-3 Clause
Code versioning system used	git
Software code languages, tools, and services used	C++, vtk, eigen, openvr, CoppeliaSim
Compilation requirements, operating environments & dependencies	VS2015, VTK 8.2, eigen, openvr, CoppeliaSim remote API
If available Link to developer documentation/manual	
Support email for questions	boris.bogaerts@uantwerpen.be

Software metadata

Current software version	2.2
Permanent link to executables of this version	https://github.com/BorisBogaerts/CoppeliaSim-VR-Toolbox/releases
Legal Software License	BSD-3 Clause
Computing platforms/Operating Systems	Windows
Installation requirements & dependencies	V-REP, Steam
If available, link to user manual - if formally published include a reference to the publication in the reference list	https://github.com/BorisBogaerts/CoppeliaSim-VR-Toolbox
Support email for questions	boris.bogaerts@uantwerpen.be

1. Motivation and significance

CoppeliaSim is a robot simulation environment used for the prototyping, development and verification of robot systems and algorithms. A robotic system can be a complex mechatronic system that is nontrivial to construct and use. The offline programming of the motion of a robot can furthermore be cumbersome, time-consuming and requires a high level of expertise [1]. In this work we developed a virtual reality interface for CoppeliaSim. While the importance of virtual reality in virtual prototyping and validation has long been understood [2–4], there are only very few software tools that meet the requirements of the robotics community [5].

The developed CoppeliaSim VR Toolbox aims to be more accessible to members of the robotics community than other software solutions. During the development we have focused on the following priorities: lowering the entry threshold for new

* Corresponding author.
E-mail address: boris.bogaerts@uantwerpen.be (B. Bogaerts).

https://doi.org/10.1016/j.softx.2020.100426

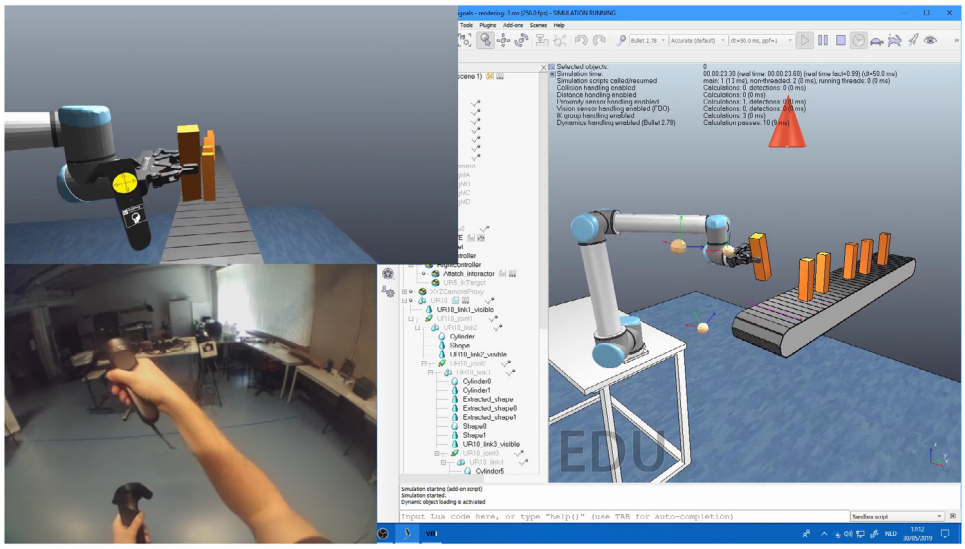

Fig. 1. The user sees the V-REP scene (right) in VR (top-left). The user can interact with a robot to teach paths.

users, simplifying the software setup, and minimizing the time to develop and test user interactions.

The software's efficacy has already been proven in the human guided design of camera networks [6] and robot path planning for optical inspections [7]. With this software, it was easy and quick to create new design scenarios that could be used to test human performance in a wider range of problems. The VR Toolbox proved its extensibility by allowing the implementation of advanced volume rendering, which was used to augment user performance.[1]

The VR Toolbox uses a commonly used robot simulator, namely CoppeliaSim. This simulator is used as usual to model the experimental scene, and to define the scene control logic. The VR Toolbox contains an application that visualizes this scene in openVR compatible devices, and returns user interactions. These can than be used in the robot simulator to modify the scene logic. The toolbox also contains a standard set of interactors in which a common set of interactions are preprogrammed. These interactors are used directly, or as building blocks for more advanced interactors or can serve as an example in the development of specific custom interactions (see Fig. 1).

Our VR Toolbox links CoppeliaSim [8], to the Visualization Toolkit (VTK) [9,10]. VTK provides an extensive library of state-of-the-art scientific visualizations and also links to openVR compatible virtual reality devices. A different approach that combines the Robot Operating System (ROS) with Unity is also available [5]. This solution, however, is less accessible to the robotics community since Unity is more challenging to set up, and is less intuitive to use for roboticists than CoppeliaSim [11]. Furthermore, the separation between scene logic in ROS and scene modeling in Unity increases the time to develop specific cases. This separation between scene logic and modeling environment is deliberately avoided in our software solution. A disadvantage that is not to be underestimated is that this extra complexity makes user studies more challenging to set up.

2. Software description and functionalities

The CoppeliaSim VR toolbox builds around a core module that converts CoppeliaSim scene data to VTK actors that can be visualized in VTK. This module is used in two separate included applications. The first application renders any CoppeliaSim scene in openVR compatible hardware, and returns user action to the simulator. We will refer to this application as the VR-Interface. The second application renders omnidirectional stereo images from CoppeliaSim scenes. These can be used to create compelling visualizations of research that was carried out with the VR-Interface.[2] We explicitly created a separate application for the VR rendering, instead of integrating these into the simulator, to guarantee a consistent and fast framerate of the VR visualization, even when the simulation speed of the simulator is inconsistent. This is important in user-studies for the comfort of subjects, and it is necessary in order to not bias results. This set of tools (i.e. module and applications) is completed with CoppeliaSim models which program standard interactions (we call these interactors). These interactors are entirely programmed in CoppeliaSim, in its native format (i.e. LUA) and also act as examples.

2.1. Software architecture

The most important component of the toolbox is the translation module (schematic version of the software architecture in Fig. 2). This module transforms CoppeliaSim data in VTK actors. To do this, the module interfaces with CoppeliaSim by calling script functions. These script files are located in a CoppeliaSim model that must be added to the scene by the user. After object loading, between renders, all information (e.g. positions) of the actors are updated, based on the scene logic in CoppeliaSim. This translation module is used to build a VTK application (in c++) which can be augmented with custom visualizations. By default, the VR Toolbox features two such applications, one which performs visualizations in VR, and one that renders omnidirectional stereo images. However, users can freely add custom visualizations based on these examples.

The easiest, and most common way of working with the VR Toolbox is by using or programming interactors. These interactors are CoppeliaSim child scripts that modify the scene logic based on interactions from the VR device. The programming of interactors is completely analogous to programming traditional logic in CoppeliaSim, or even in robot simulators in general. The graphical user interface of CoppeliaSim is designed to quickly and efficiently develop and debug such scripts.

[1] A video that shows volume rendering with the VR Interface is available online: https://youtu.be/Dsh8oyN4sD0.

[2] An example VR360 video can be found here: https://youtu.be/pFAptrCYhaQ.

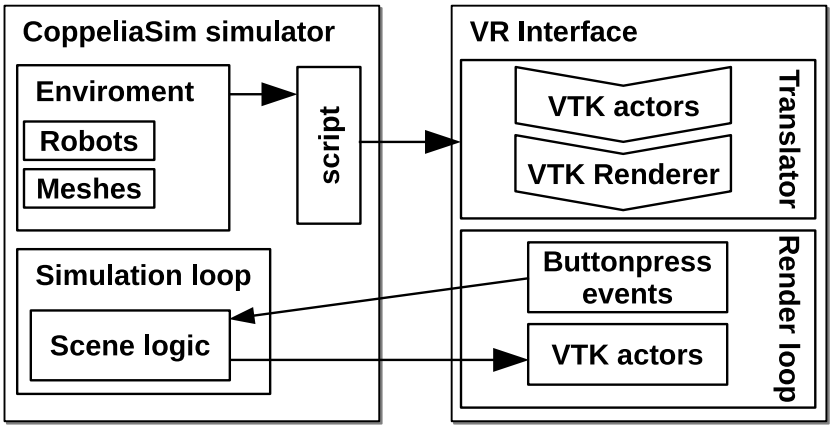

Fig. 2. A high level schematic overview of the software architecture.

```
function sysCall_actuation()
    trigger = (sim.getIntegerSignal('R_Trigger_Press')==1)
    targetHandle = sim.getObjectHandle('Robot_target')
    controllerHandle = sim.getObjectHandle('RightController')
    if trigger then
        sim.setObjectParent(targetHandle, controllerHandle, 1)
    else
        sim.setObjectParent(targetHandle, -1, 1)
    end
end
```

Fig. 3. Sample code that lets the user control the end-effector of a robot with a virtual reality controller.

2.2. Sample code snippets analysis

in Fig. 3 an interactor is programmed in a child script in CoppeliaSim (LUA). This script allows a user to control the robot end-effector with a virtual reality controller. The state of the trigger button of the right controller is stored in the integer signal *R_trigger_press*. If this button is pressed, the robot target (target of the tip–target combination of an IK object) becomes a child of the controller. This parent–child link will ensure that the relative transformation between the right controller and the robot target remains fixed. The IK calculation module in CoppeliaSim will ensure that the robot tip will match the robot target. As a result, the target will follow the movements of the controller. The parent–child link is broken if the user releases the trigger button of the right controller. This code snippet shows how easy it is to integrate virtual reality interactions in CoppeliaSim scene logic. (see Fig. 4).

3. Illustrative examples

3.1. Robotic inspection planning

We used the VR-Interface to investigate if inexperienced users are capable of generating high quality robotic inspection paths [7]. Users could move the end-effector of the robot by moving the VR controller. To help them in the process of recording inspection paths, we visualized the inspection quality on the geometry that needed to be inspected. This was achieved by programming custom vertex colors in VTK. A 360VR video was generated with the omnidirectional render tool that is also available in the VR toolbox to explain the experimental procedure.[3] This path can naturally be used for inspections on a real robot.[4]

3.2. Camera network design

Recently, we used the developed VR Toolbox to investigate how well users could design camera networks compared to automated algorithms[5] [6]. To do this we implemented an interactive volume visualization in VTK. This interactive volume showed the users which parts of the scene was invisible to a camera system, as an interactive red cloud. The users could move the cameras of the camera system by a dragging motion of the VR controllers. This is shown in Fig. 5. Upon the movement of the cameras, the interactive volume changed based on the new camera positions. With this visualization, camera network design, corresponds to placing cameras, such that there are no red areas in the environment. With this software we demonstrated that the gap between user performance, and the performance of automated algorithms is limited. The flexibility of the software allowed us to investigate this in a wide range of different (dynamic) environments.

4. Impact and conclusion

The CoppeliaSim VR Toolbox provides a set of tools that add VR support to a powerful robot system prototyping environment. CoppeliaSim itself is known for its flexibility and ability to prototype every robot system [8]. Interactors can easily be programmed in the same way and with the same flexibility.

[3] The video is available online: https://youtu.be/pFAptrCYhaQ.

[4] A video of a real inspection based on a path generated using the VR interface is available here: http://www.youtube.com/watch?v=A7E7H54v9ec&t=31m28s.

[5] An explanatory video is available online: https://youtu.be/Dsh8oyN4sD0.

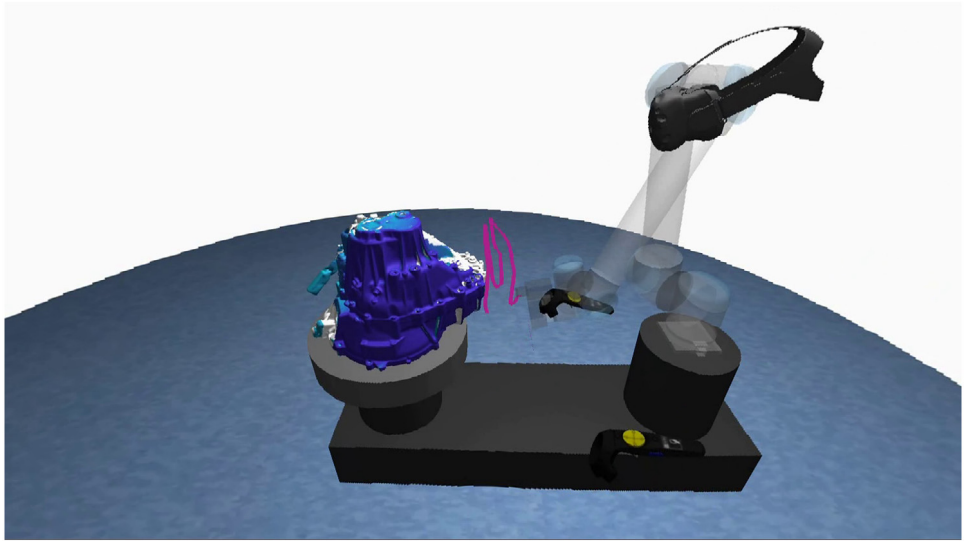

Fig. 4. The user can control the end-effector of a robot by moving a VR controller, and gets feedback on the inspection quality of an attached measurement system. This visualization allows the user to generate high quality inspection paths [7].

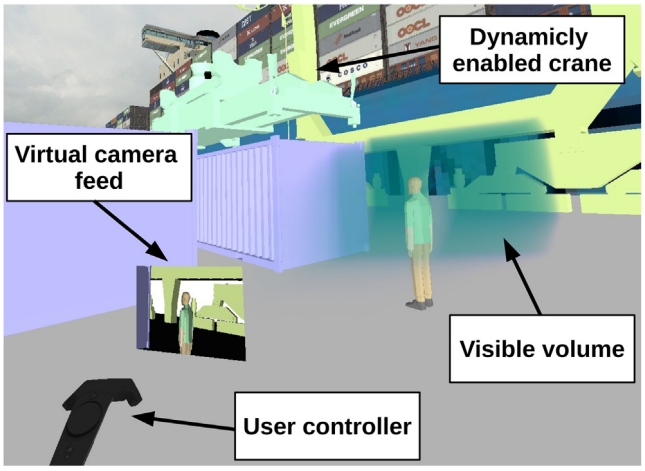

Fig. 5. The user can move virtual cameras, and see which areas of the scene are visible, here shown as a green cloud. This visualization allows users to effectively design camera networks. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

The complexity of extending the CoppeliaSim VR Toolbox is layered. The most common extension, making new interactors, can be performed with standard tools, which have a low entry threshold. However, more experienced users can build their own applications, in which they can unleash all the visualizations that VTK offers, to aid users in performing their tasks. This extensibility has already been shown with advanced volume rendering [6].

This software solution is significantly easier and faster to set up than other software solutions. This is important since it is common to perform pilot studies in human–computer interaction to guide the design process, and to eliminate experimental errors [12]. The complexity of setting up and installing all the required software is minimized which reduces the effort to set up user studies. Furthermore, all the required data that makes up a test scene, is stored in a single file. This further reduces the effort to set up user studies.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

B.B was funded by Fonds Wetenschappelijk Onderzoek (FWO, Research-Foundation Flanders), Belgium under Doctoral (PhD) grant strategic basic research (SB) 1S26216N.

References

[1] Campeau-Lecours A, Côté-Allard U, Vu D, Routhier F, Gosselin B, Gosselin C. Intuitive adaptive orientation control for enhanced human–robot interaction. IEEE Trans Robot 2019;35(2):509–20. http://dx.doi.org/10.1109/TRO.2018.2885464.
[2] Gibson I, Cobb S, Eastgate R. Virtual reality and rapid prototyping: Conflicting or complimentary? In: 1993 international solid freeform fabrication symposium, 1993.
[3] De Sa AG, Zachmann G. Virtual reality as a tool for verification of assembly and maintenance processes. Comput Graph 1999;23(3):389–403.
[4] Mujber TS, Szecsi T, Hashmi MS. Virtual reality applications in manufacturing process simulation. J Mater Process Technol 2004;155:1834–8.
[5] Krupke D, Starke S, Einig L, Steinicke F, Zhang J. Prototyping of immersive HRI scenarios. In: International conference on climbing and walking robots and the support technologies for mobile machines. World Scientific; 2017, p. 537–44.
[6] Bogaerts B, Sels S, Vanlanduit S, Penne R. Interactive camera network design using a virtual reality interface. Sensors 2019;19(5):1003.
[7] Bogaerts B, Sels S, Vanlanduit S, Penne R. Enabling humans to plan inspection paths using a virtual reality interface. 2019, arXiv preprint arXiv:1909.06077.
[8] Rohmer E, Singh SP, Freese M. V-REP: A versatile and scalable robot simulation framework. In: 2013 IEEE/RSJ international conference on intelligent robots and systems. IEEE; 2013, p. 1321–6.
[9] Hanwell MD, Martin KM, Chaudhary A, Avila LS. The visualization toolkit (VTK): Rewriting the rendering code for modern graphics cards. SoftwareX 2015;1:9–12.
[10] Schroeder W, Martin K, Lorensen B. The Visualization Toolkit–An Object-Oriented Approach To 3D Graphics. 4th ed.. Kitware, Inc.; 2006.
[11] Nogueira L. Comparative analysis between gazebo and v-rep robotic simulators. In Seminario interno de cognicao artificial, 2014, 2014, p. 5.
[12] Livatino S, Koffel C. Handbook for evaluation studies in virtual reality. In: 2007 IEEE symposium on virtual environments, human-computer interfaces and measurement systems. IEEE; 2007, p. 1–6.