Software Impacts 8 (2021) 100057

Contents lists available at ScienceDirect

Software Impacts

journal homepage: www.journals.elsevier.com/software-impacts

Original software publication

iviz: A ROS visualization app for mobile devices ☆

Antonio Zea *, Uwe D. Hanebeck

Intelligent Sensor-Actuator-Systems Laboratory (ISAS), Karlsruhe Institute of Technology (KIT), Adenauerring 2 Geb. 50.20, 76131 Karlsruhe, Germany

ARTICLE INFO

Keywords:
Robotics
Data visualization
Augmented reality

ABSTRACT

In this work, we introduce *iviz*, a mobile application for visualizing data in the Robot Operating System (ROS). In the last few years, the popularity of ROS has grown enormously, making it the standard platform for robotic programming. However, the availability of this environment is generally restricted to PCs with the Linux operating system. Thus, users wanting to see what is happening in the system with a smartphone or a tablet are stuck with solutions such as screen mirroring or web browser versions of *rviz*, making newer visualization modalities such as Augmented Reality impossible. Our application iviz, based on the Unity engine, addresses these issues by providing a visualization platform designed from scratch to be usable in mobile platforms such as iOS, Android, and UWP, and including native support for Augmented Reality for all three platforms. If desired, it can also be used in a PC with Linux, Windows, or macOS without any changes.

Code metadata

Current code version	v1.0 devel
Permanent link to code/repository used for this code version	https://github.com/SoftwareImpacts/SIMPAC-2021-1
Permanent link to Reproducible Capsule	
Legal Code License	MIT
Code versioning system used	git
Software code languages, tools, and services used	C#
Compilation requirements, operating environments & dependencies	Unity Engine 2019.4 or greater, .NET Standard 2.0 or greater
If available Link to developer documentation/manual	https://github.com/KIT-ISAS/iviz/tree/devel/iviz
Support email for questions	antonio.zea@kit.edu

1. Introduction

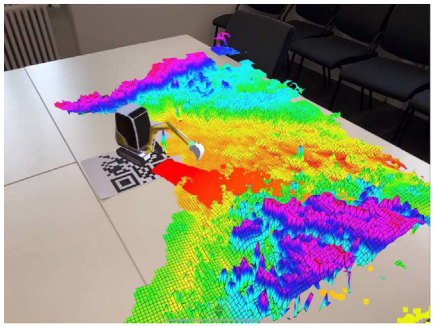
Since its inception in 2007, the Robot Operating System (ROS) [1] has been gaining ground as the premiere middleware platform for robot programming. Nowadays, ROS can be seen in every field of robotics, from tiny vacuum cleaners [2] to lawn tractors [3], from underwater vehicles [4] to deployments in outer space [5] and even for decontamination of hazardous environments [6]. Traditionally, visualization of robotic data has been limited to static two-dimensional displays, such as computer monitors. However, being able to visualize this data next to the real-world place where it was generated can provide precious contextual information that would otherwise be missed in a 2D display. This motivates an emphasis on Augmented Reality (AR) and Virtual Reality (VR) technologies, capable of displaying arbitrary three-dimensional information in any point in space.

Multiple projects in literature have combined ROS with VR [7–9] and AR [10–14], especially in an industrial setting [15]. While AR in robotics is not exactly new, it has seen an explosion in growth in the last five to seven years following the appearance of affordable, off-the-shelf devices with accurate and robust user tracking, such as the (rather pricey) Microsoft Hololens. More affordable alternatives exist in the form of Apple's ARKit and Google's ARCore, which can turn everyday smartphones and tablets into AR presentation devices. Unfortunately, mobile devices do not intersect with the platforms that ROS has traditionally supported, i.e., PCs with Linux using C++ or Python — a notable exception being [16] from 2012. However, the increasing data processing power of mobile devices, together with the advantages of AR (such as reduced cognitive load [17]), are causing a reevaluation from the ROS community, showing the need for appropriate software platforms that can take advantage of these new capabilities.

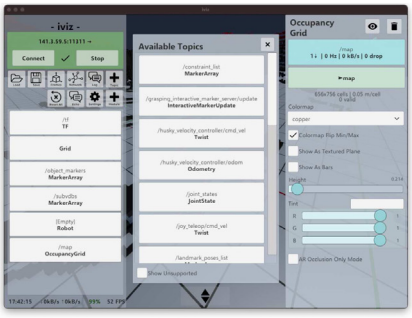
☆ Funding: This work was supported by the Federal Ministry of Education and Research of Germany (BMBF) in the framework of ROBDEKON [project number 13N14675].

* Corresponding author.
E-mail addresses: antonio.zea@kit.edu (A. Zea), uwe.hanebeck@kit.edu (U.D. Hanebeck).

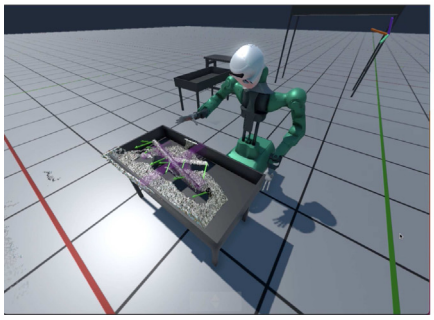
https://doi.org/10.1016/j.simpa.2021.100057
Received 1 January 2021; Accepted 23 January 2021



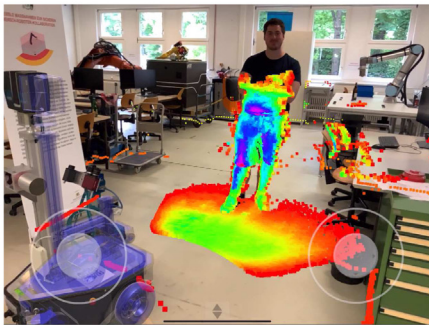
(a) AR visualization of a robotic excavator from Fraunhofer IOSB.



(b) GUI example for topic selection.



(c) Armar6 robot from KIT with mesh markers.



(d) AR teleoperation of a GammaBot robotic platform with on-screen joysticks.

Fig. 1. Examples of the iviz application in use [18,19].

In this paper, we introduce **iviz**, a new platform for ROS data visualization in mobile devices, in particular Android/iOS smartphones and tablets, and the Microsoft Hololens. We have the following goals for our platform: (i) being able to visualize a wide variety of ROS data, from point clouds to interactive markers, (ii) with fast update rate and (relatively) low latency, and (iii) with direct support for AR (as in Fig. 1(a)). Our application is based on the Unity engine and written in C#, which allows us to target many operating systems and devices with no changes in the code. The central motivation is to have a mobile app that allows a user to simply pull out a tablet or smartphone at any moment, enter a ROS network, and with a couple of taps be able to see what is going on with a robot (Fig. 1(b)), without requiring special nodes or services in the system. Additionally, by enabling the AR module, the 3D view can be projected onto the real world with just a single tap.

2. Challenges

ROS is a quasi-operating system with its own drivers, modules, and utilities. Fortunately, we do not need to rewrite them into our mobile app in order to benefit from them, as ROS provides interfaces that allow their entire functionality to be accessed remotely from the network. Thus, in order to 'talk' to a ROS system, all we need to do is to implement its network layer. However, this task yields three main challenges. First, we need a message serialization mechanism that can be used easily from C# in a mobile device. Second, we require an implementation of the ROS API that lets us talk to other ROS nodes. And third, robot visualization requires assets such as meshes and textures located on the ROS hub. Thus, we need some way to automatically transfer them to the mobile device as needed. Implementing the ROS API is not an easy task, and for this reason mobile ROS programmers have traditionally preferred the Rosbridge suite [20] instead, for example in [9,12–14]. However, the centralized architecture and JSON-based message serialization quickly lead to issues of scalability and performance. To alleviate this issue, Unity has introduced its own serialization mechanism [21], which unfortunately still requires a centralized node, thus leaving the bottleneck issue unresolved. The new version of ROS, called 'ROS 2', aims to address these issues of platform portability. However, while there are already experimental libraries for C# and Unity [22,23], ROS 2 is not backwards-compatible, and thus, most existing code will still require ROS 1 for the foreseeable future.

3. The iviz suite

Thus, we are left with the more challenging (but more rewarding) option: reimplementing the ROS API in C# from scratch, aiming for a code base that is usable not only by our visualization app, but also by any other project based on C#. Furthermore, none of the modules require a full ROS installation, and thus, installing them on a computer consists simply of copying a directory. The iviz suite is divided into the following submodules.

Message parser: Similar to the roscpp library, the *iviz_msgs_gen* module takes ROS message definition files and builds C# code from them, encapsulating the dependencies as class constants.

ROS client: The *iviz_roslib* module implements topics, services, access to the parameter server, and an experimental action client. It is based on the ROS# interface [24], widely known by Rosbridge users.

Asset loader: An important challenge when visualizing data from a robot is how to show the robot itself. We need not only the robot definition, but also its meshes, textures, pose information, and so on. While it is possible to copy this information by hand, it is preferrable to have an automated routine that transfers only the necessary files. To achieve this, we introduce the *iviz_loader_service*, an optional standalone ROS node written in C# that runs on the PC with the assets. It communicates automatically on the background with the visualization app, without requiring input from the user.

Visualization app: This is the main application, based on the Unity engine. Similar to rviz [25] (the 'standard' visualization program for

ROS), iviz relies heavily on the concept of 'displays', i.e., reusable and recyclable code modules tasked with rendering entities such as lines, point clouds, duplicated meshes, etc. For instance, a module in charge of displaying a *PointCloud2* message can also be reused to 'unproject' depth images from a *Image* topic, or to display a *Point List* marker. Implemented displays include pointclouds, laser scans, (interactive) markers (Fig. 1(c)), occupancy grids, among others. A module for on-screen joysticks (Fig. 1(d)) is also included, allowing for quick teleoperation with *Twist* messages. Of interest is the module for augmented reality. Activating it is straightforward: first, set up the scene by adding topics as necessary, and then enable AR. Look for a suitable surface, like the floor or a table, and finally, click on the 'Start' button. The scene will appear on that surface. Finally, the user can rotate, translate, and scale the scene as necessary.

4. Impact

The iviz platform is, to the authors' knowledge, the first generic ROS visualization app for mobile devices with native support for AR. While nowhere near as fully-featured as rviz, it has the advantage of being usable in every major desktop and mobile OS. This, together with AR, allows for a wide array of new, more intuitive visualization and interaction modalities. For example, planning the motion of a robotic arm does no longer require looking at a monitor. Instead, a planned path can be rendered directly on top and around the robot, together with interactive markers shown on their real 3D positions, which can be dragged with a simple finger movement. And unlike Rosbridge solutions, iviz does not require changing the ROS system by installing a new node. Thus, a user with a tablet only requires a login to the robot network, the IP of the master, and can immediately start teleoperating a robot with the joysticks. Momentarily, iviz is used primarily in the context of the ROBDEKON project [6], and is tasked with teleoperating remote tractors and humanoid robots [26] in hazardous environments. A predecessor was used for indoor localization [27] with the Microsoft Hololens, in cooperation with the German company PFW Aerospace. This work will soon be expanded into an AR-based classification system for industrial pipes, combining ROS image-processing nodes with the Hololens 2.

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.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.simpa.2021.100057.

References

[1] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A.Y. Ng, ROS: an open-source robot operating system, in: ICRA Workshop on Open Source Software, vol. 3, no. 3.2, Kobe, Japan, 2009, p. 5.

[2] A. Araújo, D. Portugal, M.S. Couceiro, R.P. Rocha, Integrating arduino-based educational mobile robots in ROS, J. Intell. Robot. Syst. 77 (2) (2015) 281–298.

[3] N.O. Lleras, S. Brennan, D. Murphy, M.J. Klena, P.M. Garvey, H. Sommer Iii, Development of an open-source tractor driving simulator for tractor stability tests, J. Agric. Saf. Health 22 (4) (2016) 227–246.

[4] K. DeMarco, M.E. West, T.R. Collins, An implementation of ROS on the yellowfin autonomous underwater vehicle (AUV), in: OCEANS'11 MTS/IEEE KONA, IEEE, 2011, pp. 1–7.

[5] J. Badger, D. Gooding, K. Ensley, K. Hambuchen, A. Thackston, ROS In space: A case study on robonaut 2, in: Robot Operating System (ROS), Springer, 2016, pp. 343–373.

[6] J. Petereit, J. Beyerer, T. Asfour, S. Gentes, B. Hein, U.D. Hanebeck, F. Kirchner, R. Dillmann, H.H. Götting, M. Weiser, et al., ROBDEKON: Robotic systems for decontamination in hazardous environments, in: 2019 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), IEEE, 2019, pp. 249–255.

[7] J.J. Roldán, E. Peña-Tapia, D. Garzón-Ramos, J. de León, M. Garzón, J. del Cerro, A. Barrientos, Multi-robot systems, virtual reality and ROS: developing a new generation of operator interfaces, in: Robot Operating System (ROS), Springer, 2019, pp. 29–64.

[8] D. Whitney, E. Rosen, D. Ullman, E. Phillips, S. Tellex, Ros reality: A virtual reality framework using consumer-grade hardware for ros-enabled robots, in: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2018, pp. 1–9.

[9] F. Muhammad, A. Hassan, A. Cleaver, J. Sinapov, Creating a shared reality with robots, in: 2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI), IEEE, 2019, pp. 614–615.

[10] L. Peppoloni, F. Brizzi, C.A. Avizzano, E. Ruffaldi, Immersive ROS-integrated framework for robot teleoperation, in: 2015 IEEE Symposium on 3D User Interfaces (3DUI), IEEE, 2015, pp. 177–178.

[11] D. Lee, Y.S. Park, Implementation of augmented teleoperation system based on robot operating system (ROS), in: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2018, pp. 5497–5502.

[12] L. Kästner, J. Lambrecht, Augmented-reality-based visualization of navigation data of mobile robots on the microsoft hololens-possibilities and limitations, in: 2019 IEEE International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM), IEEE, 2019, pp. 344–349.

[13] L. Manring, J. Pederson, D. Potts, B. Boardman, D. Mascarenas, T. Harden, A. Cattaneo, Augmented reality for interactive robot control, in: Special Topics in Structural Dynamics & Experimental Techniques, vol. 5, Springer, 2020, pp. 11–18.

[14] J. Guhl, S. Tung, J. Kruger, Concept and architecture for programming industrial robots using augmented reality with mobile devices like microsoft hololens, in: 2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), IEEE, 2017, pp. 1–4.

[15] E. Sita, C.M. Horváth, T. Thomessen, P. Korondi, A.G. Pipe, Ros-unity3d based system for monitoring of an industrial robotic process, in: 2017 IEEE/SICE International Symposium on System Integration (SII), IEEE, 2017, pp. 1047–1052.

[16] A. Zimmermann, Rviz for Android, 2012, URL http://wiki.ros.org/rviz_for_android.

[17] J. Baumeister, S.Y. Ssin, N.A. ElSayed, J. Dorrian, D.P. Webb, J.A. Walsh, T.M. Simon, A. Irlitti, R.T. Smith, M. Kohler, et al., Cognitive cost of using augmented reality displays, IEEE Trans. Vis. Comput. Graph. 23 (11) (2017) 2378–2388.

[18] T. Emter, C. Frese, A. Zube, J. Petereit, Algorithm toolbox for autonomous mobile robotic systems, ATZoffhighway Worldw. 10 (3) (2017) 48–53.

[19] T. Asfour, L. Kaul, M. Wächter, S. Ottenhaus, P. Weiner, S. Rader, R. Grimm, Y. Zhou, M. Grotz, F. Paus, et al., Armar-6: A collaborative humanoid robot for industrial environments, in: 2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids), IEEE, 2018, pp. 447–454.

[20] C. Crick, G. Jay, S. Osentoski, B. Pitzer, O.C. Jenkins, Rosbridge: ROS for non-ROS users, in: Robotics Research, Springer, 2017, pp. 493–504.

[21] C. Greene, J. Platin, M. Piñol, A. Trang, V. Vij, Robotics simulation in unity is as easy as 1, 2, 3!, 2020, URL https://blogs.unity3d.com/2020/11/19/robotics-simulation-in-unity-is-as-easy-as-1-2-3/.

[22] S. Lindgren, ROS2 for unity, 2019, URL https://github.com/DynoRobotics/UnityRos2.

[23] E. Fernandez, S. Kelly, ROS2 for .NET, 2019, URL https://github.com/ros2-dotnet/ros2_dotnet.

[24] M. Bischoff, Ros#, 2018, URL https://github.com/siemens/ros-sharp.

[25] D. Hershberger, D. Gossow, J. Faust, RViz, a 3D visualization tool for ROS, 2019, URL http://wiki.ros.org/rviz.

[26] C. Pohl, K. Hitzler, R. Grimm, A. Zea, U.D. Hanebeck, T. Asfour, Affordance-based grasping and manipulation in real world applications, in: Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2020), Las Vegas, USA, 2020.

[27] A. Zea, U.D. Hanebeck, Refining pose estimation for square markers using shape fitting, in: Proceedings of the 22nd International Conference on Information Fusion (Fusion 2019), 2019, pp. Ottawa, Canada.