Mixed Reality Robotics for STEM Education

Gordon Stein
Institute for Software Integrated Systems
Vanderbilt University
Nashville, Tennessee 37212-2328
Email: gordon.stein@vanderbilt.edu

Ákos Lédeczi
Institute for Software Integrated Systems
Vanderbilt University
Nashville, Tennessee 37212-2328
Email: akos.ledeczi@vanderbilt.edu

***Abstract*—This work presents an overview of a new system to provide motion tracking and mixed reality integration of robots for use in K12 STEM education. The robots, commanded by students through a block-based programming environment, are placed in a virtual environment visible as an overlay on the physical space. The system's control loop allows the robots to interact with virtual objects as if they are physically present within easily reconfigurable environments. This virtual space allows for lower cost robots to be used by incorporating virtual sensors and actuators, and creates the potential for a wider range of scenarios for students to work with.**

I. Introduction

Mixed reality robotics opens a multitude of new possibilities in STEM education. Off-the-shelf mobile robots can be integrated with sophisticated virtual sensors, actuators and communication devices deployed in diverse and dynamic simulated scenarios. Position and orientation of the remote controlled robots are determined by tracking them with a camera above a controlled area. Interchangeable virtual maps and other dynamic objects including virtual robots are rendered on the surface by an overhead projector and/or a mobile app with augmented reality (AR) capabilities. The platform enforces map features, such as obstacles and differences in terrain, and other interactions with the virtual world by filtering/modifying the control signals sent to the robots on-the-fly. By providing a virtual equivalent to almost all aspects except robot mobility, the cost and time required to create expensive and complex physical platforms can be reduced. For even deeper immersion in the ongoing experiment, real-time AR and virtual reality (VR) capabilities are possible. Students program the robots, both real and virtual, with a block-based programming environment.

The platform will open up completely new dimensions for learning. While robot use is already well established [1], including block-based programming for robotics [2] [3], this system will improve upon existing methods by providing sophisticated virtual sensors and actuators, allowing coordination and communication between robots, placing these robots in a mixed reality environment simulating various physical settings and additional virtual robots, and providing a collaborative platform for learning. In addition, the platform will be a perfect vehicle to introducing advanced computer science concepts in K12 education. The most critical challenge is simulating interactions between the physical and virtual entities accurately. Synchronizing the distributed state information of physical and virtual objects and correctly predicting future trajectories of the robots are essential to reach this goal.

Mixed reality creates new opportunities for robotics in the classroom. While the closely related virtual reality can bring students to totally new worlds, it has very large costs for both equipment and content development. A mixed reality approach using real robots to interact with a virtual world viewed through a projector allows for novel concepts in the classroom while providing more accessible means to do so. A constant connection to the "real world" and a reminder of practical applications is provided through the robots. While it would be possible to display the virtual environment on a standard screen, a projector displaying a "virtual overlay" on top of the physical space provides students with the sense that their robots are more than just characters in a video game, and are genuinely interacting with the virtual objects. Costs for robots and other equipment can be reduced by simulating portions of the environment, such as expensive LIDAR sensors or GPS sensors. Setup time for using the robots may also be reduced by having the task take place in the interaction between the robot and the virtual space, allowing for automation of evaluation and consistent resetting of both robots and obstacles.

II. Related Work

In recent years, a summer program has been run using the block-based programming environment NetsBlox with the same robots used in this work to teach cybersecurity concepts [4], [5]. During a one-week course, the students are introduced to block-based programming, instructed on how to control a robot through blocks, and then given a series of challenges where they must use each new topic they learn about to either protect or disrupt a robot attempting to accomplish a task. This program demonstrates cybersecurity issues such as unencrypted traffic, insecure channels, replay attacks, and denial of service attacks, while also including mitigations to protect a cyber-physical system against such attacks. From an education perspective, these summer programs also demonstrate the ability of robots in an instructional setting to capture student interest and help facilitate education on complex topics.

Mixed reality approaches with robots have been created previously, such as the PiTanks [6] and Phygital [7] games. These existing systems typically use a similar projector and camera setup, with robots tracked by the camera using markers

978-1-7281-4849-6/19/$31.00 ©2019 IEEE

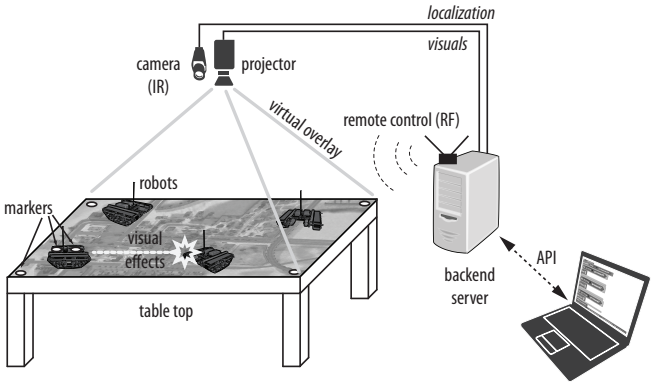


Fig. 1: Overall system architecture.

and a virtual overlay provided by the projector. However, these existing systems do not include advanced simulation models of sensors or complex virtual environments, and participants in the experiences are not required to program a robot to control it. These systems instead focus primarily on human-computer interaction in games over tight robot control loops or on applications within STEM education.

Block-based programming interfaces focusing on robotics typically either are made for a purely physical robot (e.g. Lego Mindstorms), or for a purely simulated robot. Robot Blockly [2] provides software to control a simulated model of a robot arm. Students using Robot Blockly were noted as having difficulty interpreting the simulated robot's positioning, which is not an issue for a mixed reality system as the robots' position is easily discernible in the physical space. Open Roberta [3] provides a simulation of a robot with an online block-based programming interface, but its simulation is not designed to integrate physical robots into it, is limited to a single robot at a time, and does not include advanced sensors or actuators.

III. System Architecture

A. System Overview

The system, as illustrated in Figure 1, consists of six main components:

- A "restricted area" for the robots to operate within, in a location where it is visible to students using the system
- A projector, to create a "virtual overlay" showing the virtual space on top of the existing space.
- The camera looking downward at the restricted area, used to track the positions of the robots and the edges of the space. The camera does not need to be perfectly aligned with the normal axis of the restricted area as the vision software would be able to correct for the camera orientation and position.
- A computer acting as the server, with wireless communication capabilities to command the robots. The server computer must also be powerful enough to run the simulation and maintain the robots' control loop in real time.
- Computers for students to use the web-browser based programming interface.
- The robots themselves, placed in the restricted area and connected to the server. Each robot has a printed fiducial marker attached to the top of it to enable tracking in a more straightforward way.

The simulation running on the server computer is key to the meaningful use of this system. Each of the robots is also present within the simulation's virtual environment, with the virtual location corresponding to its physical location. A control loop exists to send the correct signals to the robots to match both the commands issued to them by the students and the constraints of the virtual environment around them. For example, the virtual terrain could feature a large region of sand, where the robot will not be able to drive at as great a speed. Or the robot could slow down while pushing a virtual object, to represent the increased force required, or the robot could be stopped when it collides with another robot (either real or virtual). At the same time, the virtual models of the robots may be equipped with sensors not necessarily present on their physical counterparts. This system opens up many new possibilities by replacing all robot sensors with virtual versions. Sensors that would otherwise not be feasible to demonstrate at this scale or within a classroom, such as radar or a poison gas sensor for a simulated disaster rescue mission, are now within the realm of possibility.

Virtual environments for use in this system are not restricted to only robotics-related tasks. With the virtual overlay in place, the robots and the space may be used to represent many different concepts. For example, the robots could be used to represent predators and prey in agent-based modelling, or they could represent planetoids and have gravity simulated between them for a physics demonstration. Beyond STEM topics, the robots could become artists' tools, painting virtual colors onto their environment. With the visual component and the locomotion capabilities of the robots totally virtual and the robots' behavior easily programmable, a wide range of new possibilities become realizable. The number of robots used is limited primarily by how many can fit within the space. The size of the area used for the environment is limited by the field of view and resolution of both the projector and camera, and also the ability of the attached computer to process the larger area in real-time.

B. Robots Used

To allow for the greatest accessibility and to simplify the technical requirements of development, a low-cost commercial off-the-shelf (COTS) robot platform should be targeted. During the development process, this choice will prevent software components from being delayed while a new robot platform is designed and constructed. Once the system is deployed, use of COTS robots will prevent issues procuring additional robots to expand the system, reduce the barrier to entry for establishing the system in a new location, and provide preexisting support documentation and parts availability for maintenance. As the sensors on the robots are all simulated, the robots only need

to be programmable and capable of communicating over an internet connection. This creates a reduced barrier to entry for students being able to work with advanced sensors not commonly found on low-cost robot platforms.

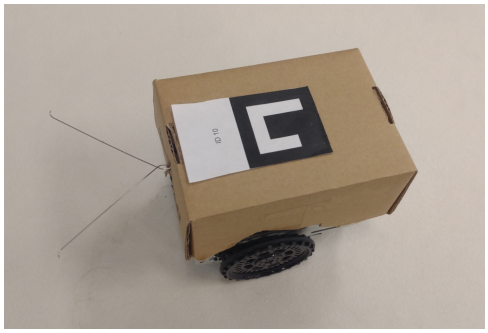


Fig. 2: Parallax ActivityBot robot with ArUco marker attached.

In current preliminary work on implementing this system, the robots used are the Parallax ActivityBot 360° platform with an attached wireless communications module. This specific platform has been used previously in conjunction with the NetsBlox programming environment [5]. Ultrasonic range and "whisker" bumper sensors are installed on the robot for observing its environment. On top of the robots, a platform has been attached to allow for an ArUco marker [8] to be temporarily affixed to each robot. A modified version of the RoboScape [4] firmware is used to provide additional features, such as allowing the range sensor's value be replaced with a distance to a virtual object. Figure 2 shows one of the robots with the tracking marker attached.

Commands are sent to the robots as UDP messages from a central server, instead of being directly programmed by students. This allows the interface students use with the robots to be greatly simplified with blocks, and reduces the electronics knowledge required to teach other subjects with the robots. It also allows the robots themselves to be more tightly controlled, for example creating an even playing field of available measures for students in a cybersecurity class.

C. Programming Interface

The current programming environment for students to interact with the robots is NetsBlox [9], [10]. NetsBlox builds upon Snap! by adding a suite of additional network features, including multi-user programs, cloud-synced collaboration, message passing and remote procedure calls (RPCs) to services running on the server. These RPCs allow for students to have easy access to various web-based APIs and server-based software through a simple block-based interface, such as Google Maps, weather and climate data, and graphing using gnuplot. In this work, the robot commands for interacting with the mixed reality environment are implemented using the existing RoboScape RPC. The existing RPC contains robot-agnostic cybersecurity education features, such as rate limiting, sequence numbers, and encryption, which are compatible with the mixed reality system as well. This simplifies the process of using that curriculum with this system, or using this system to teach similar topics. Only minimal modifications to the NetsBlox platform will be required to interact with simulations of sensors or actuators not already present in its command set. That is, only the RoboScape module will change.

Using a block-based programming environment with strong networking features contributes greatly to this project. Using blocks eases students into programming, and the RoboScape RPC in NetsBlox specifically has previously been demonstrated to be very straightforward for students to use to control a robot with. Students were able to learn to program robots through these blocks even while learning cybersecurity concepts simultaneously [4]. Students using the collaboration tools are able to work as a team, allowing even a limited number of robots to be used effectively. The network communication features present in NetsBlox allow students to experiment with robots that send messages to each other, opening up the possibility of modelling vehicle-to-vehicle and vehicle-to-infrastructure communications entirely within a block-based interface. In addition, the virtual environments may be integrated with the other existing RPCs.

D. Computer Vision

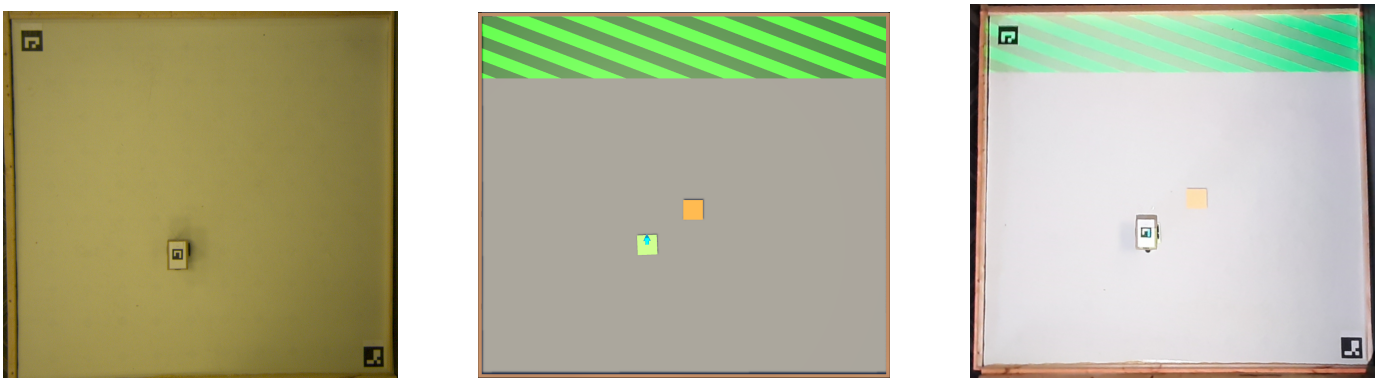
Robots on the field are tracked using ArUco markers [8] attached to them, with a camera viewing the scene from above. ArUco markers are noted for detection speed and robustness [11]. The robots are placed on a white vinyl surface, however this is not required, as the space's corners are detected using additional ArUco markers to make a reconfigurable, adaptable space. The camera is mounted to the ceiling above the space. This software was implemented using C# and Emgu CV [12], a .NET wrapper for the OpenCV library [13].

E. Game Engine

In this work, virtual environments are implemented using the Unity game engine [14]. Using an existing game engine allows for rapid development of a three-dimensional environment with attractive graphics and real-time physics simulation. In addition, Unity was selected for the potential to enable the virtual environment to be viewed using a head-mounted display, cross-platform development and deployment, and for the potential to share C# code between the computer vision system and virtual environment components. Development of new environments will be simpler for educators, as the Unity editor required to create them may be either available for free or already purchased through a school IT department, and many asset packages exist to provide both functionality and content for Unity projects. The view of this environment is then projected onto the physical area used by the robots. This creates a greater sense of the robots existing in both spaces simultaneously.

F. Architecture

The software components of the system are distributed across multiple computers. The computer vision systems along



(a) (b) (c)

Fig. 3: A robot (a) in the physical space, (b) the virtual space, and (c) with the overlay projected onto the physical space.

with the virtual environment simulation share computer hardware. This local server also runs the software handling commands from the programming interface and managing control loops for the robots. These components sharing hardware reduces latency between them due to communication, allowing for quicker response from the robots. The web-based programming interface is currently hosted on the main NetsBlox server, as the system was designed to be compatible with an existing installation of NetsBlox instead of requiring a separate one. This choice was to reduce cost by allowing for the same web server to be shared between multiple RoboScape deployments. Students creating programs to interact with the robots access the programming interface through web browsers on personal devices. The final software component is the firmware used for the robots. This firmware is the existing RoboScape firmware, with a modification for communicating directly with the the local server instead of the remote web server, allowing the computer to provide the robot with commands that are appropriate given the current state of the virtual environment, possibly resulting from an interaction with a virtual object rather than a student's command.

IV. Example Virtual Environment

A simple virtual environment was created as a prototype of the full system. This virtual world is a simple walled room of similar shape and size to the physical space. The robots appear in this room, along with a simulated box and a striped green target area. As a challenge, the robot must be programmed to autonomously locate the box and push it into the target area from an unknown starting orientation in a position directly behind the box. Once the box reaches the target area, an effect is played and the box is reset. A visual indicator off the side of the virtual space displays the number of times this task has been completed successfully. Figure 3 shows a robot in the physical space and its representation in the virtual space.

The corresponding simplified NetsBlox program can be seen in Figure 4. This short program rotates the robot slowly (using the "set speed 10 0" command) until the box is detected by the range sensor (retrieving its value with the "get range" command). When the virtual box has been located, it then drives forwards (using the "set speed 100 100" command) to push it towards the goal region. If the box is no longer in front of the robot, it begins searching again. With the sensor data simulated, the real robot is able to "see" the virtual box as if its real sensors detected it and move it around by coming into contact with it. Existing NetsBlox programs made for use without the mixed reality system continue to work, albeit with the limitation of being restricted to a smaller space.

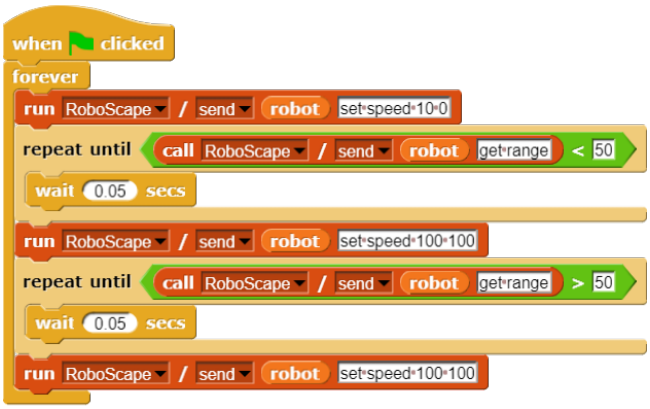


Fig. 4: Simplified example use of RoboScape blocks in NetsBlox for autonomously finding and pushing a virtual box.

V. Conclusion

Mixed reality robots have a promising future for STEM education. Using off-the-shelf robots reduces costs and technical knowledge required by instructors. By simulating sensor data and environment dynamics, the robots can act as stand-ins for many types of vehicle or other platforms that would be infeasible to actually use in a classroom, with advanced sensors and differing physical capabilities. At the same time, using a mixed reality approach, with physical robots, improves the experience over a purely simulated approach. The use of a block-based programming language creates a low barrier to entry for students with varying levels of programming experience to interact with the system, making it suitable for use in disciplines outside of computer science.

REFERENCES

[1] L. P. E. Toh, A. Causo, P.-W. Tzuo, I.-M. Chen, and S. H. Yeo, "A review on the use of robots in education and young children," *Journal of Educational Technology & Society*, vol. 19, no. 2, pp. 148–163, 2016. [Online]. Available: http://www.jstor.org/stable/jeductechsoci.19.2.148

[2] D. Weintrop, D. C. Shepherd, P. Francis, and D. Franklin, "Blockly goes to work: Block-based programming for industrial robots," in *2017 IEEE Blocks and Beyond Workshop*, pp. 29–36, Oct 2017.

[3] B. Jost, M. Ketterl, R. Budde, and T. Leimbach, "Graphical programming environments for educational robots: Open Roberta – yet another one?" in *2014 IEEE International Symposium on Multimedia*, pp. 381–386, Dec 2014.

[4] A. Lédeczi, M. MarÓti, H. Zare, B. Yett, N. Hutchins, B. Broll, P. Völgyesi, M. B. Smith, T. Darrah, M. Metelko, X. Koutsoukos, and G. Biswas, "Teaching cybersecurity with networked robots," in *Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE '19)*, pp. 885–891, 2019. [Online]. Available: http://doi.acm.org/10.1145/ 3287324.3287450

[5] A. Lédeczi, H. Zare, and G. Stein, "NetsBlox and wireless robots make cybersecurity fun," in *Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE '19)*, pp. 1290–1290. [Online]. Available: http://doi.acm.org/10.1145/ 3287324.3293749

[6] H. Costa, P. Cebola, T. Cunha, and A. Sousa, "A mixed reality game using 3Pi robots — "PiTanks"," in *2015 10th Iberian Conference on Information Systems and Technologies (CISTI)*, pp. 1–6, June 2015.

[7] M. L. Lupetti, G. Piumatti, and F. Rossetto, "Phygital play - HRI in a new gaming scenario," in *Proceedings of the 7th International Conference on Intelligent Technologies for Interactive Entertainment*. IEEE, 2015. [Online]. Available: https://doi.org/10.4108/icst.intetain.2015.259563

[8] F. Romero Ramirez, R. Muñoz-Salinas, and R. Medina-Carnicer, "Speeded up detection of squared fiducial markers," *Image and Vision Computing*, vol. 76, pp. 38–47, Aug. 2018.

[9] B. Broll, A. Lédeczi, P. Volgyesi, J. Sallai, M. Maroti, A. Carrillo, S. L. Weeden-Wright, C. Vanags, J. D. Swartz, and M. Lu, "A visual programming environment for learning distributed programming," in *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education,* pp. 81–86.

[10] NetsBlox website, https://netsblox.org, cited 2019 July 17.

[11] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and R. Medina-Carnicer, "Generation of fiducial marker dictionaries using mixed integer linear programming," *Pattern Recognition*, vol. 51, pp. 481–491, Mar. 2016.

[12] Emgu Corporation, "Emgu CV," http://emgu.com/, 2019.

[13] G. Bradski, "The OpenCV library," *Dr. Dobb's Journal of Software Tools*, 2000.

[14] Unity Team, *Unity Engine*, Unity Technologies, San Francisco, CA, 2019. [Online]. Available: http://www.unity.com/