White Rose Research Online

eprints@whiterose.ac.uk
https://eprints.whiterose.ac.uk
Universities of Leeds, Sheffield and York

Deposited via The University of York.

White Rose Research Online URL for this paper:
https://eprints.whiterose.ac.uk/id/eprint/134019/

Version: Published Version

Article:
Millard, Alan G., Redpath, Richard, Jewers, Alistair et al. (2018) ARDebug: An augmented reality tool for analysing and debugging swarm robotic systems. Frontiers Robotics AI. 87. ISSN: 2296-9144

https://doi.org/10.3389/frobt.2018.00087

Reuse
This article is distributed under the terms of the Creative Commons Attribution (CC BY) licence. This licence allows you to distribute, remix, tweak, and build upon the work, even commercially, as long as you credit the authors for the original work. More information and the full terms of the licence here: https://creativecommons.org/licenses/

Takedown
If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.

UNIVERSITY OF LEEDS
University of Sheffield
UNIVERSITY of York

frontiers in Robotics and AI

CODE
published: 24 July 2018
doi: 10.3389/frobt.2018.00087

ARDebug: An Augmented Reality Tool for Analysing and Debugging Swarm Robotic Systems

Alan G. Millard[1,2]*, *Richard Redpath*[1,2], *Alistair M. Jewers*[1,2], *Charlotte Arndt*[1,2], *Russell Joyce*[1,3], *James A. Hilder*[1,2], *Liam J. McDaid*[4] and *David M. Halliday*[2]

[1] York Robotics Laboratory, University of York, York, United Kingdom, [2] Department of Electronic Engineering, University of York, York, United Kingdom, [3] Department of Computer Science, University of York, York, United Kingdom, [4] School of Computing and Intelligent Systems, Ulster University, Derry, United Kingdom

OPEN ACCESS

Edited by:
Carlo Pinciroli,
Worcester Polytechnic Institute,
United States

Reviewed by:
Andreagiovanni Reina,
University of Sheffield,
United Kingdom
Lenka Pitonakova,
University of Bristol, United Kingdom
Jerome Guzzi,
Dalle Molle Institute for Artificial Intelligence Research, Switzerland

***Correspondence:**
Alan G. Millard
alan.millard@york.ac.uk

Specialty section:
This article was submitted to Multi-Robot Systems, a section of the journal Frontiers in Robotics and AI

Received: 01 March 2018
Accepted: 29 June 2018
Published: 24 July 2018

Citation:
Millard AG, Redpath R, Jewers AM, Arndt C, Joyce R, Hilder JA, McDaid LJ and Halliday DM (2018) ARDebug: An Augmented Reality Tool for Analysing and Debugging Swarm Robotic Systems. Front. Robot. AI 5:87. doi: 10.3389/frobt.2018.00087

Despite growing interest in collective robotics over the past few years, analysing and debugging the behaviour of swarm robotic systems remains a challenge due to the lack of appropriate tools. We present a solution to this problem—ARDebug: an open-source, cross-platform, and modular tool that allows the user to visualise the internal state of a robot swarm using graphical augmented reality techniques. In this paper we describe the key features of the software, the hardware required to support it, its implementation, and usage examples. ARDebug is specifically designed with adoption by other institutions in mind, and aims to provide an extensible tool that other researchers can easily integrate with their own experimental infrastructure.

Keywords: swarm robotics, augmented reality, debugging, open-source, cross-platform, code:c++

1. INTRODUCTION AND RELATED WORK

Debugging robotic systems is an inherently complex task, and the traditional software debugging tools currently available are of limited use. Erroneous behaviour can be attributed to either bugs in a robot's control code, or faults in its hardware, and determining the cause of the problem is often challenging due to the lack of feedback regarding the robot's internal state. Simple hardware such as on-board LEDs or LCD screens may be used to signal data to an experimenter (McLurkin et al., 2006), however such methods are only able to convey small quantities of information. Remotely connecting to a robot may allow an experimenter to capture detailed debug messages, but this kind of output can be difficult to correlate with the behaviour of the robot in real-time.

These problems are exacerbated when working with swarm robotic systems (Brambilla et al., 2013), where the quantity of debugging information required to locate and resolve a bug increases with the number of robots in the system. Large volumes of text-based debug messages concurrently transmitted by multiple robots are difficult for a human experimenter to interpret in real-time, and can easily become overwhelming. Moreover, the behaviour of an individual robot in a swarm robotic system is a product of not only its hardware, software, and interactions with its environment, but also its interactions with other robots. This increases the number of variables that may affect a specific robot's behaviour, which an experimenter must keep track of in order to isolate a bug.

Various debugging tools already exist for single- and multi-robot systems, but very few have been created specifically for use with robot swarms. Perhaps the most general-purpose tools are the `rviz`[1] 3D visualiser and `rqt`[2] GUI development framework for the Robot Operating System (ROS) (Quigley et al., 2009). However, due to its centralised architecture, ROS has not yet been widely adopted by the swarm robotics community. Following the recent official release of ROS 2.0, which features a distributed architecture, ROS may eventually gain traction with swarm roboticists. Until then, new tools and techniques are required to make the task of analysing and debugging the behaviour of swarm robotic systems easier for researchers.

It has been shown that *augmented reality*—the process of superimposing computer-generated graphics onto a view of the real world, to create a composite perspective (Azuma, 1997)—can be used to overcome some of the limitations of traditional debugging tools when engineering robotic systems (Collett and MacDonald, 2010). For example, Ghiringhelli et al. (2014) demonstrated a promising approach to debugging multi-robot systems, by augmenting a video feed with real-time information obtained from the robots being observed. Although the generalisability of their system was somewhat limited due to dependence on specialised robotic hardware, they showed that augmented reality tools can be used effectively for debugging multi-robot systems.

This paper presents ARDebug—a novel augmented reality tool that is specifically designed for analysing and debugging the behaviour of swarm robotic systems, which builds on the success of Ghiringhelli et al. (2014), offering a more generalised tool that can meet the requirements of a wider variety of swarm systems. It provides an experimenter with a single intuitive interface through which they can view the internal state of robots in a swarm in real-time, making the process of identifying, locating, and fixing bugs significantly easier. Similar *mixed reality* (Hoenig et al., 2015) applications exist that allow robots to perceive augmented reality environments through the use of virtual sensors (Reina et al., 2015, 2017; Antoun et al., 2016), which can aid the debugging process through the creation of reproducible virtual environments, but ARDebug differs from these tools in its focus on presenting detailed debugging information to a human experimenter.

2. SYSTEM ARCHITECTURE

ARDebug works in real-time, tracking each of the robots within a video feed and combining their position information with other internal data obtained wirelessly from the robots. By fusing these data sources, researchers are provided with a new tool for identifying bugs and diagnosing faults in robot swarms. Users are able to compare the internal data that defines each robot's "understanding" of its environment, against a view of that same environment, thus making any perception anomalies apparent.

An experimenter is also able to observe the behaviour of a swarm robotic system while simultaneously being informed of changes to each robot's state. ARDebug uses graphical augmented reality techniques alongside more traditional textual/visual data presentation to make this possible, with the aim of reducing the time required for a user to identify bugs or faults. The primary use case for the system is in lab-based development and debugging of new robot behaviours, prior to their use in experimental work or deployment in the field.

The experimental infrastructure required for the debugging system presented in this paper comprises three main components: the robots themselves, a tracking system, and the core ARDebug software (described in section 3). A host machine collects data sent wirelessly from the individual robots, as well as position and orientation (or *pose*) data from a tracking system. The application then combines these sources of information and presents them to the user.

2.1. Tracking Infrastructure

In our own experimental setup, the position and orientation of each robot is tracked using a JAI GO 5000C-PGE 5-megapixel camera (maximum resolution of 2,560 × 2,048), and the ArUco fiducial marker detection library (Garrido-Jurado et al., 2014). The camera is positioned 2.5 m above the centre of the arena, which is approximately 2.5 m square. A 5.2 cm square ArUco marker, is attached to the top of each robot and tracked by the camera using ArUco image processing algorithms built into OpenCV (Bradski, 2000). The image coordinates of each tag can be easily converted into real-world coordinates using a simple camera back-projection model (Sturm et al., 2011).

The use of a passive marker-based tracking algorithm means that no additional hardware is required to track the robots; any robot can be tracked as long as a fiducial marker can be attached to it. The minimum tracking hardware required to use ARDebug is incredibly low cost—ArUco tags that can simply be printed on paper, and a single visible-light camera of sufficient resolution to decode the fiducial markers at the desired distance. Inexpensive low-resolution cameras can be used if necessary, either by printing larger ArUco tags, or by generating a set of markers that are easier to distinguish at longer distances.

Alternative tracking systems can also be integrated into the system with relative ease, by sending tracking data to the core ARDebug application in JSON[3] format. For example, an infra-red (IR) motion capture system such as OptiTrack (Millard et al., 2014) or Vicon[4] could be used instead in conjunction with a visible-light camera.

2.2. Robot Platform

ARDebug is designed to be agnostic to the robot platform used—any robot that can transmit its internal state in JSON format via Wi-Fi or Bluetooth can be integrated with the software. We provide example code for integrating the system with two different swarm robotic platforms: the widely-used e-puck (Mondada et al., 2009) via Wi-Fi, and the Psi Swarm (Hilder et al., 2016) via Bluetooth.

[1] http://wiki.ros.org/rviz
[2] http://wiki.ros.org/rqt
[3] https://www.json.org
[4] https://www.vicon.com

In order to enable Wi-Fi communication with ARDebug, we have equipped each of our e-puck robots with a Linux extension board (Liu and Winfield, 2011) featuring an ARM processor and a Wi-Fi adapter. Robot control code was written for the ARM processor using the ARGoS framework (Garattoni et al., 2015), which communicates with the e-puck's dsPIC microcontroller that interfaces with the robot's sensors and actuators. This extension board could equivalently be replaced with the popular Gumstix Overo COM turret[5], or the new Pi-puck extension board (Millard et al., 2017), which extends the e-puck's capabilities by interfacing it with a Raspberry Pi single-board computer.

The Psi Swarm robot instead communicates with ARDebug via Bluetooth, using control code written for its mbed microcontroller. Despite using a different communication protocol, the data transmitted to/from the robot conforms to the same JSON interface. These two sets of example code are provided to make it easy for other users to integrate their own robot platforms with ARDebug.

3. ARDEBUG SOFTWARE

ARDebug displays data retrieved from the robots superimposed onto a video feed of the arena, in conjunction with robot pose data obtained via a tracking system. The design of the user interface focuses on presenting information to the user in a structured manner that is readily understood. High-level information is made available by means of graphical and textual overlays, with the ability to access more detailed information about the selected robot(s) via data tables and real-time charts. Previous research into interfaces for interacting with multi-robot systems has shown that users prefer this type of design (Rule and Forlizzi, 2012). Due to the volumes of data present in swarm systems, these filtering capabilities are critically important for focusing on the key data elements relevant to the task at hand.

3.1. Key Features

The ARDebug GUI, shown in **Figure 1**, is split into four regions. The visualiser (top-left) displays the augmented video feed, and can generate overlays showing a robot's position, orientation, ID, and any data that the robot reports via the JSON interface. Each of these overlays can be individually set to display for all of the selected robots in the system, or for only one robot. The connection pane (top-right), is used to display a list of the robots currently known to the system, and tabs allow access to network controls, Bluetooth device management, and data logging settings. The details pane (bottom) is used to display data transmitted by the selected robot(s), which can be overlaid onto the video feed, or displayed visually as a real-time chart in the fourth region of the application. A video demonstrating the use of these features to debug a swarm robotic system can be found on our website[6].

The *Data Visualisation* tab lists data transmitted by the selected robot(s) to the ARDebug application. This data is formatted as a series of key/value pairs, allowing the user to report any information from within their robot code that they deem important to the debugging process. For example, the user can define custom data fields such as the robot's current battery voltage, control code iteration, or random walk timer value. Each of these custom data elements can be simultaneously displayed on the visualiser; battery voltage could even be rendered as a floating bar next to each robot if desired. Users can easily add their own visualisations to the software by subclassing an abstract visualisation element class, and drawing with geometric primitives.

These key/value pairs can also be visualised using real-time charts. For example, ARDebug will display any array of numerical values as a bar chart. This feature can be used to graphically represent a robot's sensor readings, such as ambient and reflected infra-red light. Strings, such as the robot's current state, are instead displayed as a pie chart showing the distribution of values across the selected robots, which are assigned colours in the visualiser in relation to the segments of the chart (as shown in **Figure 1**). This information can be useful to determine whether a robot is getting stuck in a particular state under certain conditions. Finally, single numerical values, such as a robot's battery voltage, are visualised as a line chart displaying the recent history of reported values over time.

ARDebug also offers a number of convenience features, including: data logging for post-experiment analysis; selection of robots via the visualiser, so they can be more easily differentiated or flagged for analysis; and the inclusion of ArUco tag detection as a core feature of the application, making it easier for users to get started with the software. The mapping between ArUco tags and robot IDs can be configured by simply modifying a JSON configuration file (a default mapping file is generated if one is not found).

3.2. Implementation

ARDebug has been implemented following a model-view-controller (MVC) architecture (Krasner and Pope, 1988), and is designed to be highly modular. **Figure 2** shows a breakdown of the software architecture, including the key modules, organised according to both MVC layer and threading.

The *Model* layer contains data describing the current state of the robots, which is dealt with by the *data model*. New data from the robots or tracking system is received and parsed in the *Controller* layer, on task-specific threads, before being passed to the *application controller*, which handles updating the model. The *View* layer contains the user interface and is updated regularly to display the contents of the model, as well as receiving and passing user input messages to the central application controller. Using an MVC architecture in this way ensures that only one copy of the data describing the robots is maintained, and state information is not entangled within the user interface. Combined with the modular design, this allows individual elements of the software to be easily replaced without disrupting other elements or the overall structure.

The software is currently known to work under Ubuntu 16.04 or later, and macOS 10.13.5. OpenCV and the Qt application

[5]http://www.gctronic.com/doc/index.php/Overo_Extension
[6]https://www.york.ac.uk/robot-lab/ardebug/

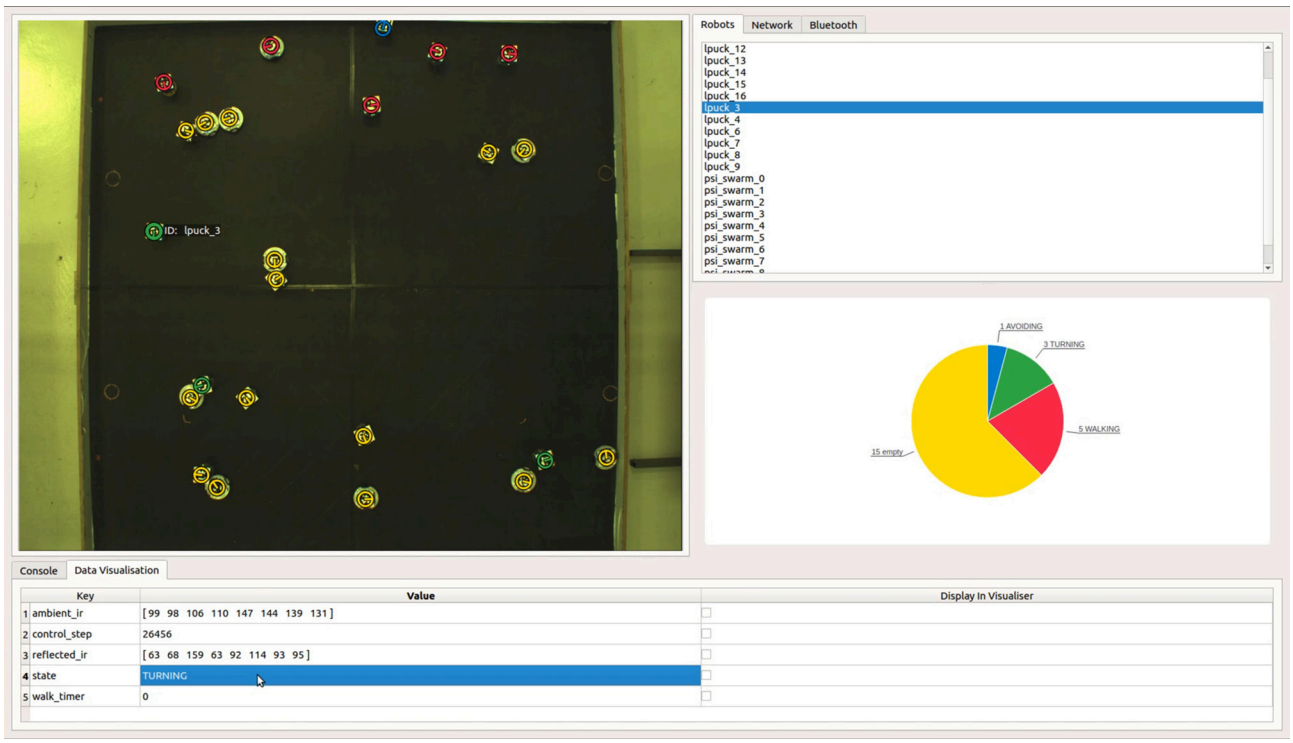

FIGURE 1 | The ARDebug user interface, showing a top-down view of a heterogeneous swarm comprising Linux-enhanced e-pucks and Psi Swarm robots. The *Data Visualisation* tab shows internal data received from the selected robot that can optionally be overlaid onto the video feed, along with the robot's recent trajectory history and graphical representations of its sensor readings. In this particular example, a real-time pie chart is also used to visualise the distribution of internal state data across the swarm.

framework[7] were chosen as the base upon which to build the ARDebug software, as they are mature, cross-platform libraries, with refined APIs and extensive documentation. They are also free, open-source software—Qt is released under GNU Lesser General Public License v3.0; and OpenCV, along with its implementation of the ArUco library, is released under the 3-Clause BSD License. It is hoped that the use of these libraries will enhance the maintainability and longevity of ARDebug.

3.3. Scalability

We have tested ARDebug with heterogeneous swarms of up 25 robots (15 e-pucks and 10 Psi Swarm robots), communicating with the application via a combination of Wi-Fi and Bluetooth in 100 ms intervals. With the software running on a server housing a 16-core Intel Xeon E5520 (2.27 GHz) and 16 GB RAM, ARDebug was able to track all of the robots and visualise their internal state in real-time. OpenCV's ArUco library implementation supports multi-threaded execution, so scales well as the number of robots in a swarm increases. We have verified that the software is able to track up to 50 ArUco tags in real-time, which is representative of the number of robots used in large-scale swarm experiments [excluding miniature robotic platforms such as Kilobots (Rubenstein et al., 2012) or Droplets (Farrow et al., 2014), which are not supported by ARDebug].

If the experimental arena covers a large spatial area, then it may be necessary to stitch together images from multiple overhead cameras in order to track an entire swarm. If high resolution cameras are used, the computational load can be distributed by performing ArUco tag detection on multiple servers, which then send tracking data to the ARDebug application via the JSON interface.

The performance of the application (post-tracking) is largely independent of number of robots, the amount of data they send, and the frequency they send it at. This is because ARDebug collates received data over a short time window before triggering an update to the GUI, instead of updating the user interface upon receipt of each JSON packet.

3.4. Current Limitations

ARDebug is currently limited to tracking robots in a 2D plane, but could be extended for use with flying robots moving in three dimensions if integrated with a tracking system such as OptiTrack or Vicon. The application is also only able to integrate with robotic platforms that can communicate with a server via Bluetooth or Wi-Fi. This therefore excludes Kilobots (Rubenstein et al., 2012) and Droplets (Farrow et al., 2014), which are only able to communicate via IR light.

An inherent limitation of the Bluetooth specification means that only 7 robots can simultaneously connect to a single Bluetooth adapter. In order to use ARDebug with more than 7 Bluetooth-connected robots, it is necessary to use

[7]https://www.qt.io

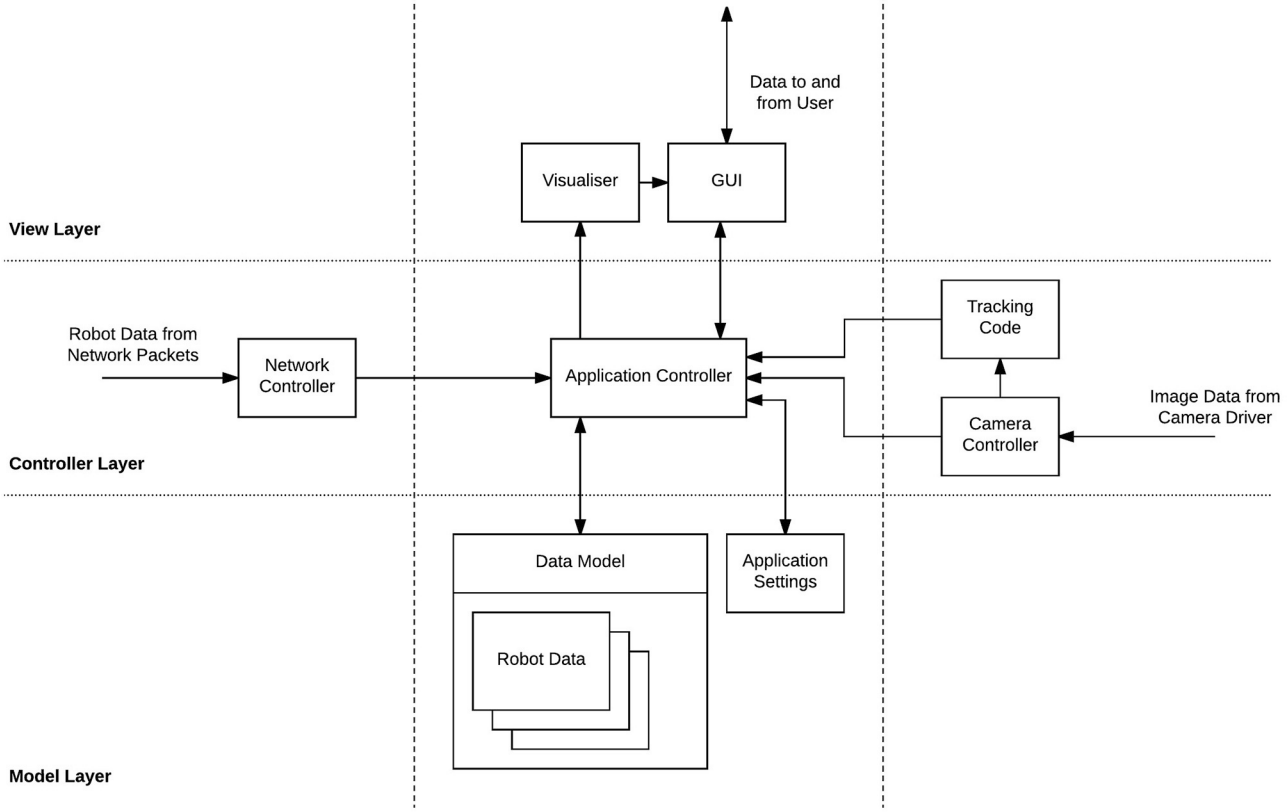

FIGURE 2 | ARDebug software architecture. Components are split across Model, View, and Controller layers, and Main, Network, and Camera threads.

multiple adapters that communicate with up to 7 robots each. If Wi-Fi/Bluetooth communication is used for inter-robot communication as part of an experiment, then also sending data to ARDebug may create a communication bottleneck. However, this can be mitigated by reducing the amount and frequency of data transmitted to the application.

4. CONCLUSIONS

Appropriate debugging tools are necessary if the behaviour of swarm robotic systems are to be analysed and debugged effectively. In particular, tools are needed that allow an experimenter to visually inspect and interpret the data held by each robot in the system. This paper has presented ARDebug—a tool for monitoring a robot swarm by displaying its internal data in real-time. Augmented reality techniques are employed to visualise the data, with the aim of making it readily understandable, and therefore quicker to parse than numerical or textual data alone.

ARDebug aims for minimal hardware requirements, cross-platform compatibility, and is implemented in a modular fashion to allow easy modification and integration with different hardware. The software is open-source (released under the GNU General Public License v3.0), and has been made freely available online in the hope that it will contribute a useful tool to the field of swarm robotics research: 10.5281/zenodo.1283488 (Datasheet 1 in Supplementary Material).

Scenarios in which ARDebug is envisioned to be useful include: diagnosing robot control code bugs, identifying sensor hardware faults or calibration issues, and verifying actuator hardware operation. Access to internal state information and decision-making variables could aid in debugging robot control code, for instance, by checking that state transitions are occurring in response to stimuli, or by checking that variables are being updated correctly.

A number of future extensions to the system are planned: support for scripting custom video augmentations, to allow users to more easily tailor the system to their own needs; video recording and data log replay, for more in-depth post-experiment analysis and debugging; and bidirectional communication between ARDebug and the robots, allowing the software to be used as a central experiment controller.

AUTHOR CONTRIBUTIONS

AM, RR, and AJ wrote sections of the paper. AJ wrote the initial version of the software and documentation, which was then developed further by AM, RR, and CA. AM, RJ, and JH worked on experimental infrastructure that enabled the development of the software. LM and DH supervised the project.

FUNDING

This work is funded by EPSRC grants EP/N007050/1 and EP/L000563/1.

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/frobt.2018.00087/full#supplementary-material

REFERENCES

Antoun, A., Valentini, G., Hocquard, E., Wiandt, B., Trianni, V., and Dorigo, M. (2016). "Kilogrid: a modular virtualization environment for the Kilobot robot," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Daejeon: IEEE), 3809–3814.

Azuma, R. T. (1997). A survey of augmented reality. *Presence Teleoperat. Virtual Environ.* 6, 355–385.

Bradski, G. (2000). The OpenCV Library. *Dobbs J. Softw. Tools* 25, 120–123. Available online at: http://www.drdobbs.com/open-source/the-opencv-library/184404319

Brambilla, M., Ferrante, E., Birattari, M., and Dorigo, M. (2013). Swarm robotics: a review from the swarm engineering perspective. *Swarm Intell.* 7, 1–41. doi: 10.1007/s11721-012-0075-2

Collett, T. H. J., and MacDonald, B. A. (2010). An augmented reality debugging system for mobile robot software engineers. *J. Softw. Eng. Robot* 1, 18–32. Available online at: http://hdl.handle.net/2292/8956

Farrow, N., Klingner, J., Reishus, D., and Correll, N. (2014). "Miniature six-channel range and bearing system: algorithm, analysis and experimental validation," in *IEEE International Conference on Robotics and Automation (ICRA)* (Hong Kong), 6180–6185.

Garattoni, L., Francesca, G., Brutschy, A., Pinciroli, C., and Birattari, M. (2015). *Software Infrastructure for e-puck (and TAM)*. Technical Report TR/IRIDIA/2015-004, Université Libre de Bruxelles.

Garrido-Jurado, S., noz Salinas, R. M., Madrid-Cuevas, F., and Marín-Jiménez, M. (2014). Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recogn.* 47, 2280–2292. doi: 10.1016/j.patcog.2014.01.005

Ghiringhelli, F., Guzzi, J., Di Caro, G. A., Caglioti, V., Gambardella, L. M., and Giusti, A. (2014). "Interactive augmented reality for understanding and analyzing multi-robot systems," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Chicago, IL), 1195–1201.

Hilder, J. A., Horsfield, A., Millard, A. G., and Timmis, J. (2016). "The Psi swarm: a low-cost robotics platform and its use in an education setting," in *Conference Towards Autonomous Robotic Systems* (Sheffield: Springer), 158–164.

Hoenig, W., Milanes, C., Scaria, L., Phan, T., Bolas, M., and Ayanian, N. (2015). "Mixed reality for robotics," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Hamburg), 5382–5387.

Krasner, G. E., and Pope, S. T. (1988). A description of the model-view-controller user interface paradigm in the smalltalk-80 system. *J. Object Oriented Program.* 1, 26–49.

Liu, W., and Winfield, A. F. T. (2011). Open-hardware e-puck Linux extension board for experimental swarm robotics research. *Microprocess. Microsyst.* 35, 60–67. doi: 10.1016/j.micpro.2010.08.002

McLurkin, J., Smith, J., Frankel, J., Sotkowitz, D., Blau, D., and Schmidt, B. (2006). "Speaking swarmish: human-robot interface design for large swarms of autonomous mobile robots," in *AAAI Spring Symposium: To Boldly Go Where No Human-Robot Team Has Gone Before* (Palo Alto, CA), 72–75.

Millard, A. G., Hilder, J. A., Timmis, J., and Winfield, A. F. T. (2014). "A low-cost real-time tracking infrastructure for ground-based robot swarms," in *Proceedings of the 9th International Conference on Swarm Intelligence (ANTS)* (Brussels: Springer), 278–279.

Millard, A. G., Joyce, R., Hilder, J. A., Fleşeriu, C., Newbrook, L., Li, W., et al. (2017). "The Pi-puck extension board: a Raspberry Pi interface for the e-puck robot platform," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Vancouver, BC), 741–748.

Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C., Klaptocz, A., et al. (2009). "The e-puck, a robot designed for education in engineering," in *Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions, Vol. 1,* (Castelo Branco), 59–65.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., et al. (2009). "ROS: an open-source robot operating system," in *ICRA Workshop on Open Source Software, Vol. 3* (Kobe), 5.

Reina, A., Cope, A. J., Nikolaidis, E., Marshall, J. A., and Sabo, C. (2017). ARK: augmented reality for Kilobots. *IEEE Robot. Autom. Lett.* 2, 1755–1761. doi: 10.1109/LRA.2017.2700059

Reina, A., Salvaro, M., Francesca, G., Garattoni, L., Pinciroli, C., Dorigo, M., et al. (2015). "Augmented reality for robots: virtual sensing technology applied to a swarm of e-pucks," in *NASA/ESA Conference on Adaptive Hardware and Systems (AHS)* (Montreal, QC: IEEE), 1–6.

Rubenstein, M., Ahler, C., and Nagpal, R. (2012). "Kilobot: A low cost scalable robot system for collective behaviors," in *IEEE International Conference on Robotics and Automation (ICRA)* (Saint Paul, MN: IEEE), 3293–3298.

Rule, A., and Forlizzi, J. (2012). "Designing interfaces for multi-user, multi-robot systems," in *Proceedings of the Seventh Annual ACM/IEEE International Conference on Human-Robot Interaction* (Boston, MA), 97–104.

Sturm, P., Ramalingam, S., Tardif, J.-P., Gasparini, S., and Barreto, J. (2011). "Camera models and fundamental concepts used in geometric computer vision," in *Foundations and Trends in Computer Graphics and Vision*, 36–89. Available online at: https://hal.archives-ouvertes.fr/inria-00590269/

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Copyright © 2018 Millard, Redpath, Jewers, Arndt, Joyce, Hilder, McDaid and Halliday. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY). The use, distribution or reproduction in other forums is permitted, provided the original author(s) and the copyright owner(s) are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.