RASPIED Literature Review

A Review of Robots as Pedagogical Tools in Computer Science Education

Muhummad Yunus Patel, PTLMUH006
University of Cape Town

ABSTRACT

The potential use of robots as pedagogical tools has long been recognised and much work has been done in this field. This paper reviews previous studies investigating the use of robots in education with a particular focus on their use within the context of introductory Computer Science at a tertiary education level. The studies surveyed reported that the use of robots as pedagogical tools was well received by students and generally resulted in improved student performance.

The robot hardware used in the surveyed studies can be categorised as follows: standard off-the-shelf robots, modified off-the-shelf robots, fully custom-built robots, and robots built using the Lego Mindstorms platform. These various approaches to robot hardware each have their own advantages and disadvantages. The cost of the platform and the feature set required by the curriculum will determine which approach best suits any particular use-case.

Robots used at a tertiary education level tend to favour traditional programming languages over the graphical programming languages often used for younger students. The software used to program the robots may also vary by hardware platform and this should be considered when choosing a robot. Fully software-based robot simulations have also been effectively used as a pedagogical tool both in-place-of, and alongside a physical robot.

In general, the use of robots as pedagogical tools in introductory Computer Science is effective. A key determinant of success in robot aided curricula, particularly at a tertiary education level, is adequate access to the robot for students. There is still space for work to be done to reduce the cost of educational robots while still maintaining the pedagogical effectiveness of the robot and supporting the feature set required by the Computer Science curriculum.

1. INTRODUCTION

In recent years we have seen the release of computing devices like the Arduino, and Raspberry Pi[20]. These low-cost, ultra-portable, hackable computing devices have led to an increased interest in the field of Computer Science. Aside from enabling electronics hobbyists to construct home-made 3d-printers and CNC machines, these devices could also serve as a teaching aid. In particular, one of the major selling points of the Raspberry Pi is its potential use as a teaching aid.

Tools like the Raspberry Pi, Lego Mindstorms kit, and Arduino can be used to construct low-cost educational robots. Such robots have been used with some degree of success at primary education level[21], secondary education level[1], and tertiary education level[22]. This paper surveys various aspects of previous programs utilising robots as educational tools with a focus on applications in introductory Computer Science courses at a tertiary level and the hardware and software used to build and program these robots.

2. ROBOT HARDWARE

The educational robots used in many of the studies surveyed have been custom built to suit the requirements of that specific application. For example, one study used custom-built robots connected to a Raspberry Pi controller[21], while others used robots built with the Lego Mindstorms kit[12, 18]. Key hardware considerations are the cost of the robot, and the features supported (e.g. programming method, sensors, cameras, wireless connectivity, etc.).

2.1 Lego Mindstorms

The main component of the Lego Mindstorms robotics kits is a programmable controller brick. The controller brick serves as the hub to which users connect input sensors and output components (e.g. motors and/or lights). The user can then program the controller brick to make use of the connecccted input and output devices to achieve the desired robot behaviour.

2.1.1 Features and Limitations

The Lego Mindstorms platform provides a number of input sensors including touch, light, and rotation sensors. Output devices such as servo motors are also provided. The feature set supported by the Lego Mindstorms hardware does not appear to be a limiting factor and it has been successfully used to teach such advanced concepts as Artificial Intelligence[12]. The Lego Mindstorms sensors and motors come packaged in convenient plug-and-play modules that have been made to work within the Mindstorms platform; however, the official Mindstorms modules are far more expensive than the raw electronic components. A more cost

effective solution would be to use generic electronics to build the required functionality.

The major limitations of the platform include the limited on-board memory and the lack of support for useful wireless communication protocols[12]. The Lego Mindstorms platform only supports three input ports but this limitation can be circumvented by 'stacking' sensors to the same port and interpreting the raw input in the software[12].

Another common criticism of the Lego Mindstorms platform - particularly in the context of tertiary-level courses - is that the official graphical (drag-and-drop) programming language is too simple[12]. This seems to be a non-issue as there are third-party tools allowing the controller to be programmed using many popular languages including, inter alia, Java, Ada, and C++. One of the studies surveyed overcomes this problem by using the Not Quite C (NQC)[2] programming environment[12], while another used the Ada/Mindstorms2.0[6] programming environment[8].

2.1.2 Cost

A study investigating the suitability of the Lego Mindstorms platform for Artificial Intelligence education concludes that the Lego Mindstorms platform is a cost-effective, viable option for the inclusion of a robotic element in Computer Science courses[12]; however, I would argue that the cost of the platform is still too prohibitive, especially in light of the limitations outlined above. Furthermore, an extensive year long study to measure the effectiveness of robots in teaching Computer Science found that the Lego robots were ineffective and that students using the robots performed worse than the control group[8]. This negative result was attributed mainly to a lack of access to the robots this hypothesis is supported by the fact that other studies using cheaper robots provided to every student reported more positive results[17].

2.2 Customised Hardware

Many robot-aided curricula make use of custom hardware or modified off-the-shelf robots. Motivations behind this approach include cost reduction, the inclusion of novel features, and the support of particular applications.

A major benefit of using custom hardware is the ability to control the cost of the robot and tailor its functionality. For example, one of the papers surveyed presents a custom-built, low-cost platform tailored for use with 3rd and 4th grade students[21]. The platform allowed for the entire system to be set up five minutes and consisted of five simple robots that were entirely custom built for €15 each. These five robots were then connected (via bluetooth) to a central Raspberry Pi server that allowed for the robots to be programmed wirelessly through a web-based programming interface. The robots were programmed using a customised version of Blockly, an open-source Google project allowing users to drag and drop 'blocks' to construct simple programs.

The custom-built solution described above would not scale well for larger use-cases as each robot had to be constructed by hand. By using modifiable/hackable off-the-shelf robots, larger use cases can be accommodated whilst still allowing the robot to be tailored to the specific needs of the curriculum. A study conducted at Bryn Mawr College exemplifies this approach by using the Scribbler robot from Parallax Inc. with a custom add-on for further functionality such as a camera and wireless Bluetooth connectivity[4]. The customised robot costs $110 and is now available in kit form from the Institute for Personal Robotics in Education (IPRE)[13].

Other customised hardware solutions have been implemented in a similar manner to those described above and fall into 3 main categories, viz. fully custom built robots[21], modified off-the-shelf solution[13], and standard off-the-shelf solution[14].

As previously stated, custom built robots can be tailored to suit specific curriculum and budget requirements but may require more effort to build, set up, and maintain. Modified off-the-shelf robots reduce the time investment required to build, set up, and maintain the robots, whilst still allowing some control over the functionality of the robot. The modified off-the-shelf robots, however, are generally more expensive that fully custom built robots. Standard off-the-shelf robots are generally the most expensive option but require very little to no build time.

3. SOFTWARE

When using robots as educational tools, particularly in the context of Computer Science education, the method and language used to program the robot should be carefully considered. The languages being used to program educational robots include (in descending order of popularity) Java, C++, Not Quite C (NQC), C, Python, and Scheme[15]. The language used to program the robot may be limited by the hardware platform as some platforms may have very specific requirements.

3.1 Lego Mindstorms Programming

The Lego Mindstorms robots require special tools to allow them to be programmed in languages other than the default graphical (drag-and-drop) interfaces supplied by Lego. There are many popular third-party tools allowing the Lego Mindstorms robots to be programmed using more conventional languages (or subsets thereof). For example, LeJOS lets users program the Mindstorms controller 'brick' using Java. These third-party tools allow for increasingly complex programming of the Mindstorms platform and they have been successfully used to teach such advanced Computer Science concepts as Artificial Intelligence[12].

3.2 Graphical Programming Languages

Graphical programming languages like Scratch[16] have been widely used to introduce fundamental programming concepts. Custom visual programming languages can be constructed using Google's Blockly tool as demonstrated in a study on low-cost robotics in elementary school education, where it was effectively used to design a simplified programming language that allowed 3rd and 4th grade students to learn to program their educational robots in roughly 5 minutes[21]. These graphical programming languages are well suited to younger audiences but have not been used to teach Computer Science at a tertiary level as they are too simplistic. Furthermore, introducing students to Computer Science using a graphical programming language designed around fun may leave students with a feeling of 'bait-and-switch'[4] if later languages are too different.

3.3 Traditional Programming Methods

At a tertiary level, it would be more appropriate to use actual programming languages, or subsets thereof. A common

design pattern in this space is to design an API in a common language that allows users to program the robots easily whilst still allowing sufficient complexity. One of the better known examples of this design pattern is the Arduino programming language which is based on C/C++ and is used to program Arduino micro-controller prototyping boards. It allows novice users to write trivial programs whilst still allowing more experienced users to exploit most of the features of C++. The Arduino IDE also hides most of the complexity involved in compiling the code and uploading it to the Arduino device which greatly simplifies the process for new users; however, more experienced users are free to use their preferred tools (eg. Makefiles). The Arduino platform is a good example of an easy-to-use tool that scales well with user experience, which has been identified a a key factor in engaging students[13]. The Raspberry Pi implements a similar idea by allowing novice users to control its GPIO pins using either scratch or a Python library.

Libraries like Pyro[3] (and its derivative, Myro) have been designed to program robots in a platform-independent manner by abstracting away the intricacies of the hardware specific to a robot. Users are presented with a simple interface allowing them to read in sensor values and program the robot appropriately. Whilst originally designed with a few specific off-the-shelf robotic platforms in mind, it has been shown that these libraries can be extended to support arbitrary robots as long as the required interfaces are implemented[9].

3.4 Robot Simulations

The use of robot simulations as an educational tool has also been explored. Using a software simulation of a robot greatly reduces the cost and possible hardware issues. Two examples of such simulation packages include Karel[19], and Alice[5]. A four semester study using Karel-like simulation software for the duration of an introductory Computer Science course was able to reduce the percentage of failing students by 77.4% (from 43.1% to 9.8%)[23]. Similarly positive results for simulation based approaches have also been reported in a previous review of the field[15]. Whilst a simulation based approach reduces the cost and hardware issues, it also removes the tangible aspect of using a robot as a teaching aid. A possible solution to this is a hybrid approach that uses both a physical robot, and a software simulation (eg. Using Lego Mindstorms robots and a compatible simulator for students to test their programs[7]). This allows students to work with the robot without having to worry about hardware problems whilst still retaining the tangible aspect of using an actual robot.

4. DISCUSSION

The previous sections have outlined the approaches of using robotics in education. We have examined the broad hardware choices available, viz. fully customised solution, modified off-the-shelf solution, and standard off-the-shelf solution and their associated considerations. Some of the main issues to consider when choosing the hardware include the budget/cost of the robot, the curriculum to be supported, the programming languages/software supported, the robustness of the robot, and how well it scales with the users' experience. Which of the three hardware options should be used for a specific use-case will depend mainly on the budget and curriculum requirements. A fully customised solution allows for the robot to be fully customisable but may require a more significant investment of time to build and set up the robots. Conversely, a stock standard off-the-shelf solution should require very little (if any) time to set up and build but will be more expensive.

One of the difficulties of working with robots in the context of introductory Computer Science is hardware malfunctions. Students participating in an introductory Computer Science course using robots as teaching tools reported that hardware-related issues were one of the biggest difficulties [10]. This was further supported by a similar claim from more senior students using robots to learn Artificial Intelligence concepts[12]. Hardware related difficulties could include, but are not limited to, interpretation of sensor readings (and accounting for noisy/inaccurate readings)[12], the actual building of the robot (where construction is required)[12], excessive battery usage[17], and connection problems (eg. unstable Bluetooth connections). These technical difficulties could theoretically be completely avoided by simply using a robot simulation framework but these difficulties are, to some extent, arguably beneficial. The difficulty students experience in interfacing with hardware components mirrors the difficulty they will experience in real world applications and "re-enforces the concept that programs can and will affect others"[10]. That being said, for introductory courses, it may be best to reduce hardware problems to allow students to focus on the actual Computer Science concepts being taught.

Studies have shown that insufficient access to the robot is a major source of frustration to students[11]. This is supported by a study carried out at the United States Airforce Academy where students had limited access to the robots (due to budgetary constraints) and were unable to debug their code without the robot[8]. Students in this study also reported that the lack of access to the robot was their biggest frustration, even more so than mechanical/hardware issues. A follow up study suggested that the problem of access to the robots could be solved by introducing a simulation framework alongside the physical robots[7]. This would then allow students to run and debug their code even when they do not have access to the robot. The Institute for Personal Robotics in Education (IPRE) attempts to address this issue by designing robots that can be purchased by every student, although this may still prove too costly for programs with a low budget. The hybrid approach of using a simulation framework alongside a physical robot is a low-cost, effective way of solving the issue of access to the robot.

5. CONCLUSIONS

This paper has reviewed studies relating to the use of robots as an educational aid with particular focus on applications in introductory Computer Science courses at a tertiary level. The hardware and software used to build and program the robots in the various studies considered has been discussed along with the challenges faced when using robots as teaching aids.

Much of the early work using robots in education made use of the Lego Mindstorms platform. More recent work has shifted to focus more on cheaper, custom-built robot hardware and modified off-the-shelf robots. This shift could possibly be attributed to the introduction of low-cost, hackable, computing devices like the Arduino, and Raspberry Pi. Studies using the cheaper custom-built solutions have

shown similar success rates compared to studies using the more expensive Lego Mindstorms platform.

While some work has been done to investigate low-cost robots in primary education[21], there does not appear to be much work done on low-cost robots at a tertiary education level. One reason for this may be that the increased feature set required to support the Computer Science curriculum inevitably pushes the cost up. The Institute for Personal Robotics in Education (IPRE) has produced a robot and curriculum designed for a Computer Science course where every student is provided with a robot but this may still prove too costly for departments with a low-budget (eg. in a South African context). There is still space for work to be done to reduce the cost of educational robots while still providing a feature set that supports the Computer Science curriculum.

It has been shown that insufficient access to the robot is a major frustration for students and may result in worse performance than traditional teaching methods[8]. One solution to this is to provide every student with their own personal robot. A more cost effective solution is to, rather than purchase more robots, give students access to a simulation of the robot[7]. They are then able to program and debug their solutions off-campus without needing to buy their own robot. Simulation based approaches have also been shown to be at least as successful as approaches using physical robots[15].

In conclusion, the use of robots and/or robot simulations in Computer Science education has been shown to be effective. One of the key determinants of success is sufficient access to the robot for students[8]. There is still space for work to be done to reduce the cost of educational robots while still maintaining the pedagogical effectiveness of the robot and supporting the feature set required by the Computer Science curriculum.

6. REFERENCES

[1] D. Alimisis. Robotics in education & education in robotics: Shifting focus from technology to pedagogy. In *Proceedings of the 3rd International Conference on Robotics in Education*, pages 7–14, 2012.

[2] D. Baum and J. Hansen. Nqc programmer's guide. 2003.

[3] D. Blank, D. Kumar, L. Meeden, and H. Yanco. Pyro: A python-based versatile programming environment for teaching robotics. *Journal on Educational Resources in Computing (JERIC)*, 4(3):3, 2004.

[4] D. S. Blank and D. Kumar. Assessing the impact of using robots in education, or: How we learned to stop worrying and love the chaos. In *AAAI Spring Symposium: Educational Robotics and Beyond*, 2010.

[5] S. Cooper, W. Dann, and R. Pausch. Alice: a 3-d tool for introductory programming concepts. In *Journal of Computing Sciences in Colleges*, volume 15, pages 107–116. Consortium for Computing Sciences in Colleges, 2000.

[6] B. Fagin. An ada interface to lego mindstorms. *ACM SIGAda Ada Letters*, 20(3):20–40, 2000.

[7] B. Fagin. Ada/mindstorms 3.0. *Robotics & Automation Magazine, IEEE*, 10(2):19–24, 2003.

[8] B. Fagin and L. Merkle. Measuring the effectiveness of robots in teaching computer science. In *ACM SIGCSE Bulletin*, volume 35, pages 307–311. ACM, 2003.

[9] T. Fossum and J. Snow. How platform-independent is pyro? *Department of Computer Science, Potsdam NY*, 2006.

[10] S. P. Imberman, R. Klibaner, and S. Zelikovitz. Student feedback on robotics in cs1. In *AAAI Spring Symposium: Semantic Scientific Knowledge Integration*, pages 65–68, 2007.

[11] J. S. Kay. Robots in the classroom...and the dorm room. *Journal of Computing Sciences in Colleges*, 25(3):128–133, 2010.

[12] F. Klassner. A case study of lego mindstorms' suitability for artificial intelligence and robotics courses at the college level. In *ACM SIGCSE Bulletin*, volume 34, pages 8–12. ACM, 2002.

[13] D. Kumar, D. S. Blank, T. R. Balch, K. J. O'Hara, M. Guzdial, and S. Tansley. Engaging computing students with ai and robotics. In *AAAI Spring Symposium: Using AI to Motivate Greater Participation in Computer Science*, pages 55–60, 2008.

[14] T. Lauwers, I. Nourbakhsh, and E. Hamner. Csbots: design and deployment of a robot designed for the cs1 classroom. In *ACM SIGCSE Bulletin*, volume 41, pages 428–432. ACM, 2009.

[15] L. Major, T. Kyriacou, and O. P. Brereton. Systematic literature review: teaching novices programming using robots. *Software, IET*, 6(6):502–513, 2012.

[16] D. J. Malan and H. H. Leitner. Scratch for budding computer scientists. In *ACM SIGCSE Bulletin*, volume 39, pages 223–227. ACM, 2007.

[17] S. A. Markham and K. King. Using personal robots in cs1: experiences, outcomes, and attitudinal influences. In *Proceedings of the fifteenth annual conference on Innovation and technology in computer science education*, pages 204–208. ACM, 2010.

[18] W. I. McWhorter and B. C. O'Connor. Do lego® mindstorms® motivate students in cs1? *ACM SIGCSE Bulletin*, 41(1):438–442, 2009.

[19] R. E. Pattis. *Karel the robot: a gentle introduction to the art of programming*. John Wiley & Sons, Inc., 1981.

[20] Raspberry Pi Foundation. Raspberry Pi Website. https://www.raspberrypi.org/.

[21] M. Saleiro, B. Carmo, J. M. Rodrigues, and J. H. du Buf. A low-cost classroom-oriented educational robotics system. In *Social Robotics*, pages 74–83. Springer, 2013.

[22] J. Summet, D. Kumar, K. O'Hara, D. Walker, L. Ni, D. Blank, and T. Balch. Personalizing cs1 with robots. In *ACM SIGCSE Bulletin*, volume 41, pages 433–437. ACM, 2009.

[23] A. Yadin. Reducing the dropout rate in an introductory programming course. *ACM inroads*, 2(4):71–76, 2011.