BIRT: An Open-Source Integration Toolkit for Architectural Robotics

Luis Arturo Pacheco

[1] Florida International University, Miami, Florida
lpacheco@fiu.edu

Abstract. The growing role of robotic systems in various fields, including architecture calls for a more accessible and adaptable control solutions. This paper introduces the Blender Interactive Robot Control Tool (BIRT), an open-source toolkit designed to simplify hardware integration, motion control, tool pathing, and interfacing with robotic arms using an Internet of Things (IoT) approach. BIRT combines the precision of offline programming with the adaptability inherent in online programming. It comprises a Blender interface, a Python communication abstraction layer, and an IoT hardware integration interface. This toolkit seeks to simplify the integration of sensors and actuators, offer an interactive approach to tool pathing, explore non-traditional approaches to robot control, and enhance affordability. Three separate yet interconnected tools are developed and tested as part of BIRT, each dealing with a different aspect of the system. This research contributes to the field of architectural robotics by proposing a solution to make robotics more accessible and adaptable to varying operational demands.

Keywords: Interactive, Robotics, Real-time, Blender3d, Fabrication.

1 Introduction

As automation continues its rise across various industries, there's a rising need for more user-friendly and intuitive interfaces for robotic arm control. Traditional industrial robot control interfaces are often complex and outdated, confining their use to individuals with extensive technical knowledge (Briggs, 2004). This limitation impedes the adoption of robotics technology among non-technical users in creative industries, such as architecture and crafts.

Architects and creative professionals are pushing boundaries by exploring a variety of tools and solutions for programming robots and collaborating with them. These efforts typically pivot around either offline programming approaches (Braumann, 2011; Poustinchi, 2020; Soler, 2023; Schwartz, 2012) or online programming approaches (Gannon, 2018; Garcia del Castillo y Lopez, 2019; Munz, 2016). A main limitation of this approach is that online systems often lack the precision of offline, and conversely, offline lack the adaptability of online systems. This becomes particularly challenging when integrating robots in dynamic fabrication environments, such as construction sites and craftwork settings (Braumann, 2022). It is also challenging to integrate diverse

components for custom end-effectors and other robot accessories for dynamic environments. (Braumann, 2012; Jensen, 2020; Stefas , 2018; Rossi, 2022; Van Mele, 2017-2023)

To address these challenges, I present a workflow centered around the Blender Interactive Robot Control Tool (BIRT). BIRT is an open-source toolkit that streamlines hardware integration, motion control, toolpathing, and end effector control using an IoT approach. Combining the precision of offline programming with the flexibility of online programming, BIRT incorporates a Blender user interface, a Python communication abstraction layer, and an IoT hardware integration interface (Figure 1).

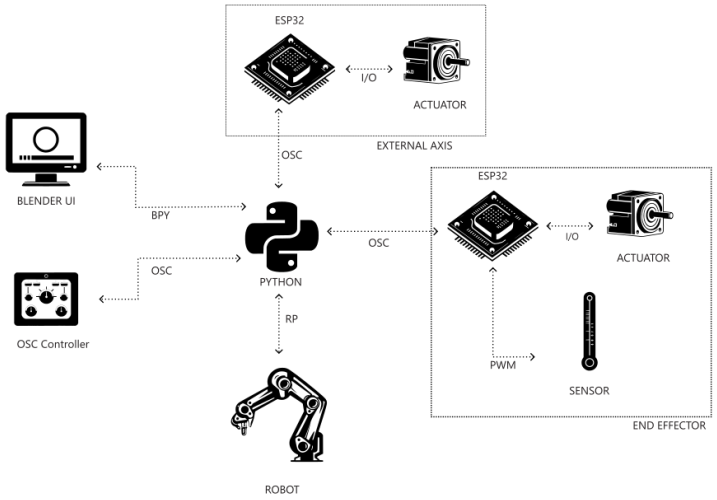

Figure 1. Systems diagram showing the relationship between the BIRT control software stack, hardware abstraction layer, and IOT hardware interface. Source: Luis Pacheco, 2023

1. **Current robot control software and their constraints**

In the past decade, there has been a significant spike in the development of software tools for programming and controlling robots, catering to different applications, ranging from traditional industrial robots to creative and architectural robotics. These tools could be broadly categorized as Industrial robotics, architectural robotics, and creative robotic tools. The following is an overview of some of the most used robot arm control and programming interfaces highlighting their features and capabilities.

1.2 Industrial robotics tools

Traditional robotic programming tools, designed primarily for industrial applications, streamline the development and implementation of robotic systems for repetitive tasks. Prominent among these are tools like ABB RobotStudio for ABB robot simulation and offline programming (abb.com, 2023), KUKA Sim for KUKA robots (kuka.com, 2023), Universal Robots' Polyscope that offers no-code programming for Universal Robots (universal-robots.com, 2013). Each one of these focuses on a specific robot brand. There are alternatives that standardize robotic development like RoboDK for simulation, offline programming, and NC conversion for multiple robot brands (robodk.com, 2023).
These software solutions provide capabilities to simulate, program, and control robots, usually for various manufacturing tasks. They typically offer graphical user interfaces for ease of use but are geared towards industry professionals with experience in robot-specific programming languages, PLC programming, and electronics. Consequently, they can present challenges when applied in non-industrial settings such as architectural robotics, where flexibility, simplicity, and ease of experimentation are key considerations.

1.3 Architectural Robotics Tools

Architectural robotics involves the integration of multi-axis robotic systems within the design and construction process, enabling architects and designers to explore innovative fabrication techniques, materials, and spatial configurations, usually incorporating digital fabrication and mass customization. These tools facilitate the creation of complex structures and geometries that push the boundaries of conventional architectural design (Kolaveric, B., 2001).

Some architectural robotics tools, include KUKA//PRC for parametric robot control in Rhino and Grasshopper initially aimed at mass customization and later incorporated real-time control tools, HAL as a robotics software package for architectural applications, Robots as a plugin for Rhino's Grasshopper interface to create parametric toolpaths and simulate robot programs, and the COMPAS framework as an open-source computational framework for collaborative research in architecture (Van Mele, 2021), cater to the specific needs of architects and designers. These tools integrate with popular design software such as Rhino and Grasshopper, enabling users to create complex robotic programs for fabrication and construction. With these tools, architects and designers can explore new ways of fabricating structures, objects, and spaces using robotic systems, pushing the boundaries of conventional architectural design. One key limitation of these architectural robotics tools is limited or restricted access to real-time control, robot state, and sensor feedback needed for interactive tool pathing and real-time control.

1.4 Creative Robotic Tools

Creative robotic tools encompass many applications outside the traditional industrial and digital fabrication context, including art, design, and human-robot interaction. These tools enable users to control and program robots in novel ways, allowing for unique expressions and interactions that push the boundaries of robotics technology.

Creative robotic tools encompass a variety of innovative solutions designed for applications beyond traditional industrial settings. One such tool is Mimic, an open-source plugin for Autodesk Maya that facilitates the control of industrial robots (mimicformaya.com, 2023). Another creative tool, Quipt, developed by Madeline Gannon, enables gestural communication and interaction between humans and robots, paving the way for more natural and intuitive human-robot collaborations (Gannon, 2016). In the domain of parametric design, Oriole Beta stands out as a tool that empowers designers to 'design' the motions of the robot based on the principles of keyframing and time-based animation, instead of merely generating pre-set motions (Poustinchi, E., 2020). Further, Robot Ex Machina, an open-source computational framework, offers real-time robot programming and control, implementing an action-state model that provides a highly interactive online robot programming interface (García del Castillo y López, J. L., 2019). They enable different means of interaction and artistic expression using robotic systems. These tools allow users to control industrial robots interactively or in real-time for creative tasks, such as animation, gestural communication, and robotic videography.

Despite the variety of creative robotic tools available, several gaps and issues still need to be addressed to enhance their usability and functionality. One major issue is the lack of real-time feedback while developing a reactive toolpath, and integration with custom end effectors and sensors which prevents users from accurately visualizing the current state of the full robotic cell system. This limitation hinders the ability to adapt and react to changes in the work cell or the workpiece during the programming process. Finally, some of these tools often involve complex workflows that can be time-consuming and difficult for users, particularly those with limited technical expertise to set up.

Simplifying the integration of sensors, reading robot states, and real-time control for reactive tool pathing is crucial to enable more interactive and adaptable robotic programming. Current tools often require advanced knowledge and complex programming to incorporate sensor data into the toolpath generation process, limiting their accessibility and adaptability.

Lastly, there is a need for a completely open-source workflow that encourages collaboration, knowledge sharing, and the development of innovative solutions. Many existing tools are proprietary or have limited open-source capabilities, hindering the potential for broader adoption and customization by the user community. Addressing these gaps and challenges will help advance the field of creative robotics and enable more accessible, flexible, and interactive robotic programming solutions.

2 BIRT Description

The Blender Interactive Robot Control tool (BIRT) aims to provide a comprehensive workflow that combines the precision and accuracy of offline programming with the parametric tool pathing workflows offered by architectural tools while integrating online features to enable a hybrid approach to interactive robotic tool pathing. This innovative workflow caters to tasks that require adaptability to changes in the work cell or the workpiece, which are commonly encountered in construction and traditional craft settings. By providing an interactive tool for hybrid fabrication, BIRT allows for the seamless integration of precise toolpath operations with the dynamic variables present on construction sites and within craftwork environments (Gannon, M., 2006). Furthermore, BIRT emphasizes the ability to integrate various sensing devices, focusing on interactive and reactive tool pathing, which enables more robust and adaptive control over robotic systems. Through this approach, BIRT can significantly enhance the capabilities of robotic programming in these industries, fostering greater innovation, efficiency, and adaptability in the creative applications of robotic technology. The following describes implementing a Minimum Viable Product (MVP) for testing this workflow.

2.1 BIRT Minimum Viable Product (MVP)

To develop the MVP, I chose Blender for its robust 3D environment, animation tools, and comprehensive Python API, which facilitates the capturing of work cell features and the generation of dynamic toolpaths. Its popularity in creative fields and open-source nature make it an excellent choice. I focused on two collaborative robot platforms: U-Factory X-Arm and Universal Robots due to their safety, user-friendly design, cost-effectiveness, and adaptability. To simplify communication and integration, I adopted OSC (Open Sound Control), which excels in transmitting data over networks using UDP. It effectively bridges software and hardware because it is human-readable, extensible, and widely supported in creative coding communities. Lastly, the ESP32 was chosen for its affordability, networking capabilities, and user-friendliness (Figure 3). The ESP32 expedites integration with various low-cost electronics, sensors, and actuators, minimizing the need for voltage conversions and physical connections.

The development of BIRT MVP consists of the creation and testing of three key tools, each handling a different aspect of the system. The first tool is the Blender Interface, a user-friendly platform for intuitive control and visualization. The second is the communication backend, which is responsible for efficient and streamlined data exchange between various system components. The third and final tool is focused on the hardware abstraction layer that interfaces with the sensors, actuators, and hardware devices.

2.2 The Blender Interface

The Blender Interface operates as a Blender addon. It offers intuitive, CAD-like control of a simulated robot arm, which acts as a controller for the physical robot (Figure 2). This provides UI elements that manage connections with the robot, simulate and visualize the robot and work cell, and establish links with various sensors, actuators, and end effectors. It also sets the toolpath from target empties (frames), curves, or edges (polylines). Users can select a toolpath and set custom parameters to control the robot's motion, speed, extrusion rate, cooling fan, etc. based on their unique setup.

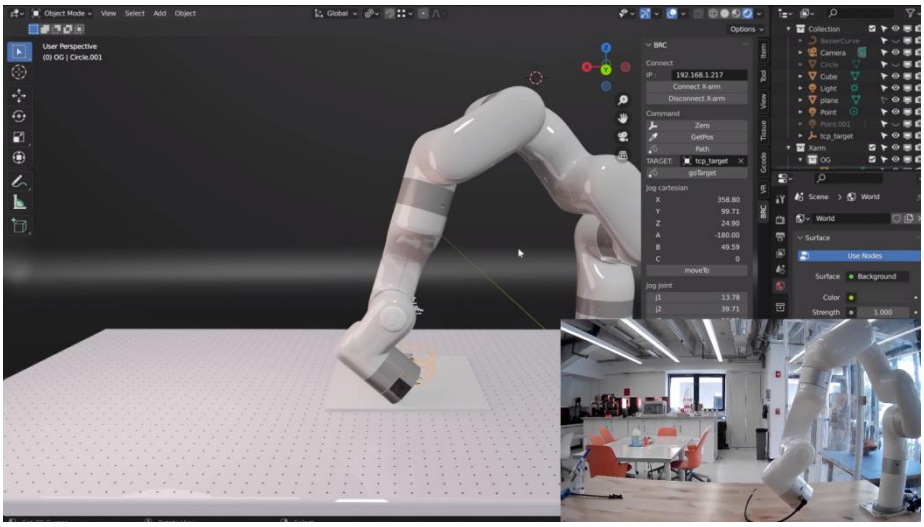

Figure 2. BIRT is built on Blender, a popular and actively developed open-source CAD tool. Here I show the control interface for moving the simulated robot, which accurately matches the real-world robot. Source: Luis Pacheco, 2023

2.3 The Python Backend

The Python backend script serves as an essential link between different systems. It leverages the OSC messaging protocol and various libraries that support robot control to facilitate communication between elements. For instance, BIRT uses the Python library ur-rtde for communication with Universal Robots, the XArm Python SDK for with XArm robots and KukavarProxy for KUKA robots.

This backend script introduces a human-readable messaging standard, promoting seamless coordination among the interface, data streams, sensors, actuators, end effectors, and other components. It forms an abstraction layer that manages the messages necessary for custom components to read or send data. Although it's designed to seamlessly integrate with Blender's API, it holds

the potential to function stand-alone or within other Python environments, such as Rhino.

2.4 The ESP32 Firmware

The ESP32 Firmware provides a streamlined communication channel that interprets OSC messages to the GPIO, PWM, I2C, and more, and vice versa. This allows for the easy integration of various low-cost sensors and actuators. Each ESP32 is wirelessly incorporated into the system, both expecting and sending messages that describe its functionality to the Python backend. This approach accelerates prototyping and experimentation by eliminating the need for voltage conversion across systems, reducing the need for physical connections and cables. Additionally, the ESP32 opens up the possibility of creating custom remote interfaces for controlling individual components, thus enhancing interactivity and ease of use.

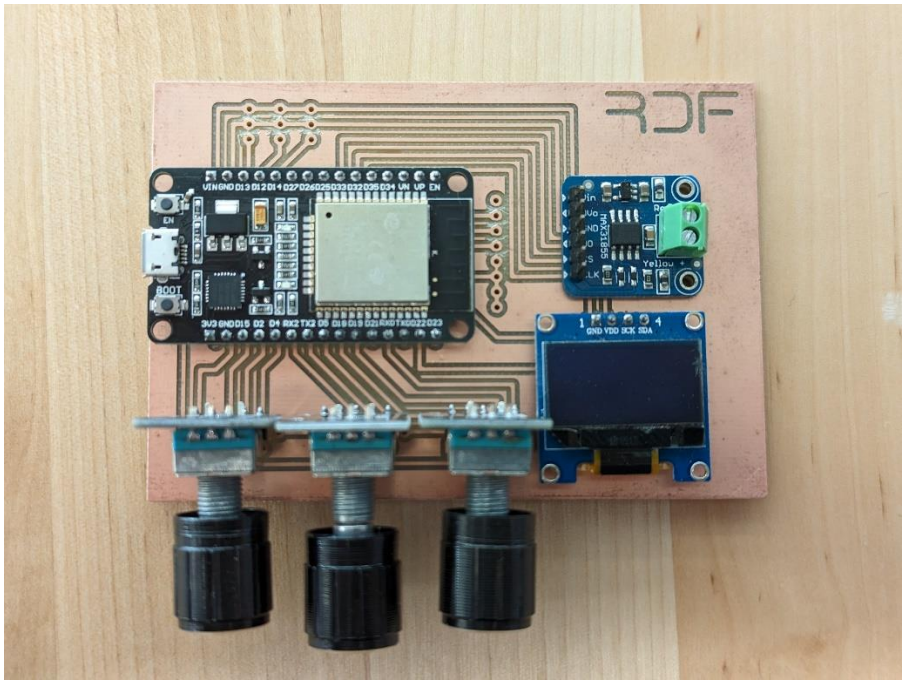

Figure 3. An example of a physical control panel for an extruder that allows to change the flow, fan speed and temperature dynamically. Source: Luis Pacheco, 2023

3 Prototypical Workflow

The prototypical workflow commences with the digitization of the work cell's features. Users employ sensors, scanning tools, or the robot itself to capture critical points, such as fixtures, tooling, and obstacles, within the workspace. The digitized features are instantaneously visualized in the 3D viewport, enabling accurate robot motion planning and interaction with the work cell. By ensuring the virtual environment aligns with the physical space, users can optimize the robotic workflow.

The second step of the workflow is the interactive development of toolpaths and their simulation. Users have the flexibility to add, modify, or update features both physically and digitally at any point in time. They can simulate the robot and various other systems before transmitting the motion paths, which allows for adjustments, safer motion, and minimizing errors. Parametric tool pathing utilizes these key points as parameters, offering a dynamically updated toolpath definition. Furthermore, toolpaths can be 'hand-drawn' by referencing various points and surfaces, thereby promoting a more intuitive and direct approach to toolpath creation.

In the final step of the prototypical workflow, users define the control flow logic for the toolpaths and parameters. At any given point, it's possible to modify the toolpath for real-time adjustments to the robot program and other systems in the cell. Logic states can be programmed to specify how the robot should respond to changing conditions within the work cell, workpiece, or user input. This enables users to create custom behaviors and interactions tailored to their needs. By incorporating real-time adjustments into the robot program, end effectors and other actuators users can create a more dynamic and adaptable system, capable of responding to the inherent variability present in construction sites and craftwork environments.

4 Potential Use Cases

Each of the following hypothetical cases could benefit from BIRT's capabilities like digitizing real-world contexts and entities, interactively developing toolpaths and interactions, converting these into actionable robotic movements in the physical world, and controlling the use of customized end-effectors or external axes, it can also be used to synchronize multiple robotic systems to work together.

4.1 Construction site work area feature digitization

BIRT tool can be employed to capture data on the features of the work area, such as walls, tubes, and other elements in a construction site. The captured data can be used to calibrate the virtual work cell in Blender, enabling users to

plan and simulate robotic tasks with a high degree of accuracy. This process reduces the risk of collisions and ensures that the robot's movements are safe and efficient in a dynamically changing work area. For example, the robot can be used to drill a specific pattern and size for holes on a wooden surface, adapting to real-world conditions and adjusting its movements accordingly, simultaneously allowing for adjusting the tool speed and penetration.

4.2 Hand-Drawn Toolpath

BIRT can be employed to capture multiple surfaces, enabling users to create "hand-drawn" toolpaths that adapt to the normals of the surfaces. This interactive approach to toolpath development allows users to design and implement custom toolpaths that accurately follow the contours of the workpiece, resulting in a more precise and tailored outcome. Some examples can include painting, milling, or 3D printing on nonplanar surfaces. Each toolpath could be dynamically generated on a tablet or drawing app, then the program would map the curve into the given surface, allowing for an interactive workflow.

4.3 Parametric Pick and Place Operations

BIRT can be used to program a robotic arm for pick and place tasks. Users can define a different source point for each object to be picked up and placed in an organized parametric array. An object could be then tracked in real-time to specify the source or target location, it is also possible to track the position of an obstacle and have a parametric toolpath that avoids collision with the obstacle. This enables the robotic arm to adapt to varying object positions, ensuring accurate and efficient object handling and placement, even in environments with dynamic object layouts.

4.5 AI Robotic Painting

BIRT can potentially integrate a workflow that involves incorporating an external "feed" of movements to paint based on an AI API integration. By leveraging AI algorithms, the robot arm can generate painting toolpaths that produce unique and visually engaging designs. The BIRT tool provides the necessary framework for connecting the AI-generated toolpaths with the robot's movements, enabling users to create AI-driven robotic painting applications.

5 Conclusion and Future Work

The development of BIRT tool demonstrates the potential for combining the powerful features of Blender with the specific needs of robotic control in various

industries. The tool provides a user-friendly, accessible, and extensible solution for both novice and experienced users to create and control robotic programs with ease. As the field of robotics continues to grow and evolve, there are numerous avenues for future research and development of the BIRT tool. Some potential directions include:

- Expanding the range of supported robot models and brands: Extending the BIRT tool to accommodate a wider variety of robot arms from different manufacturers will increase its versatility and applicability in various industries.

- Incorporating machine learning and computer vision techniques: Integrating advanced algorithms for machine learning and computer vision can further enhance the tool's capabilities for real-time control and interaction with the robot and work cell environment.

- Developing a physical user interface: Creating a physical interface for the BIRT tool, such as a handheld controller or haptic device, could improve user experience and enable more intuitive control of the robot.

- Exploring collaborative applications: Investigating the potential for the BIRT tool to facilitate human-robot collaboration in various industries can open up new possibilities for productivity and creativity.

- Enhancing the CAM and 3DP slicing capabilities: Developing more advanced CAM features and 3DP slicing tools within the BIRT tool will enable users to carry out more complex tasks and streamline the fabrication process.

By addressing these future research directions, the BIRT tool can continue to evolve and adapt to the changing needs of the robotics community, driving innovation and collaboration in various fields. As an open-source project, the BIRT tool can benefit from the collaborative nature of the open-source community, allowing individuals and organizations to contribute to its development, enhancing its features, and addressing the specific needs of various users and industries. The open-source approach encourages innovation, accelerates the development process, and ultimately results in a more robust and versatile tool.

References

Biggs, G., & MacDonald, V. (2004). A Survey of Robot Programming Systems.

Braumann, J., & Brell-Cokcan, S. (2011). Parametric Robot Control - Integrated CAD/CAM for architectural design.

Braumann, J., & Brell-Cokcan, S. (2012). Digital and Physical Tools for Industrial Robots in Architecture: Robotic Interaction and Interfaces.

Braumann, J., & Brell-Cokcan, S. (2014). Visual robot programming - Linking design simulation and fabrication.

Braumann, J., Gollob, E., & Singline, K. (2022). Visual Programming for Interactive Robotic Fabrication Processes. Process flow definition in robotic fabrication.

Gannon, M. (2016). Quipt: A gesture-based design tool for robot programming.

Garcia del Castillo y Lopez, J.L. (2019). Robot Ex Machina.

Gleadall, A. (2014). FullControl GCode Designer: open-source software for unconstrained design in additive manufacturing.

Jensen, M.B., & Das, A. (2020). Technologies and Techniques for Collaborative Robotics in Architecture - establishing a framework for human-robotic design exploration.

Kolaveric, B. (2001). Digital Fabrication: Manufacturing Architecture in the Information Age.

Mimic. (n.d.). Mimic. https://www.mimicformaya.com/

Munz, H., Braumann, J., & Brell-Cokcan, S. (2016). Direct Robot Control with mxAutomation: A New Approach to Simple Software Integration of Robots in Production Machinery, Automation Systems and New Parametric Environments.

Poustinchi, E. (2020). Oriole Beta: A Parametric Solution for Robotic Motion Design Using Animation.

Rossi, A., et al. (2022). An Open Approach to Robotic Prototyping for Architectural Design and Construction.

Schwartz, T. (2012). HAL| Extension of a visual programming language to support teaching and research on robotics applied to construction.

Soler. Robots for Rhino and Grasshopper.

Van Mele, T., et al. (2017-2021). Compass: A framework for computational research in architecture and structures.

Williams, E. (2018). Hackaday Article.