IntEL4CoRo Master-Thesis: Visual Programming in Cognitive Robotics in an everyday manipulation

December 20, 2023

Jörn Syrbe

Introduction

Cognitive robotics is a driving force for developing AI systems that can perform complex tasks autonomously and competently. With the help of knowledge and models, these AI systems are able to interpret their environment. For example, such robots can understand vaguely formulated instructions by correctly interpreting the context. They can perform complex everyday activities such as setting the table or cleaning up.

The skills to understand and develop such AI systems can best be taught using open robot control systems that contain a broad spectrum of AI technologies. This theses topic is part of the IntEL4CoRo project. The goal of IntEL4CoRo is to create access to the field of cognitive robotics and everyday robot manipulation by making use of a learning environment for cognitive robotics that enables competence-oriented teaching.

The learning environment follows an immersive approach, containing robot control systems and photorealistic, physics-based simulation environments for experimentation. To illustrate the concepts on how everyday robotics work, it is crucial to give learners a better understanding on how manifold robot manipulation is.

Thesis Objectives

The main objective of this thesis is to add a graphical programming interface to the IntEL4CoRo robot control and simulation environment. Such graphical programming interfaces are well-known in industrial applications, smart home environments, and primary & secondary education.

Contributions

In literature, graphical programming is also described by block programming, low-code, or no-code approaches. However, each of these approaches is used to give interested laymen and learners access to programming without learning a specific programming language. The education community is discussing the pros and cons of these interfaces in different ways. The following table is a short excerpt from the discussion:

Approach	Pro	Contra
Block programming	• Instead of developing the services, the programmer can borrow the code from the available vendor in order to use and provide the services to the end-user. • The approach will result in a high decrease in the associated application development costs due to the number of	• The created blocks might not be able to satisfy all the end-user's needs. • The programmer needs to understand the code from the vendors, which could be hard to understand due to the implementation using different programming languages.

	programmers needed to develop the application. • The application allows the selected services or functions to appear that the end-users want to use or see. • The end-user is not necessarily required to have professional IT skills.	• The services might not be made available by the vendors. Therefore, the block for the unavailable services cannot be created, and this will cause failure to meet the end-users needs. • Some of the code from the vendors will change from time to time and must be manually updated by the programmer.
Graphic programming	• Dynamo is useful for designing easily describable geometries that can be broken down into simple shapes and geometric elements. • Widespread adoption in the environment of future and practicing engineers [2] • Increases efficiency in this area to an extent unmatched by CAD or BIM systems	• Rare or few coherent scientific or scientific-didactic publications in the form of academic textbooks to comprehensively present the idea of the practical applications in structural design and present at least a few elementary examples of scripts of visual programming by which a lay engineer could start working in a given environment. • Form and scope in compilations [3,4] are not adequate.
Low-code	• Can work as a part of the front-end systems supporting processes, creating interfaces that support business intelligence. • well suited for experimentation because of the quick development [5] • Accelerate software development and delivery [6] • help organizations respond more quickly to feedback after initial software releases [7] • easier to learn compared to traditional code and syntax [8] • not fragile to changes in the user interface • Functional Proof of Concepts can be developed within a short amount of time	• In unforeseen development challenges, machine code presents greater difficulties for developers in problem-solving compared to traditional code [9] • Improvements are not easy to add and are hard to maintain • offers a lack of individuality and uniqueness, with the same underlying mechanism • Lack of error details in the low-code tools and less transparency regarding debugging tools. • Slower and less efficient due to being competent in all the data serialization formats (e.g., JSON, YMAL, XML, etc.)

	• Enables business and IT to unite and create mutual understanding, support Agile development, and increase the likelihood of a successful implementation	
No-code	• Allows non-technical developers to drag and drop application components to develop mobile or web applications [10] • Minimize development and management costs [10] • Any conventional browser can run a No-Code app in place of AppOS. [10] • Simultaneously supporting iPhone and Android while it supports various templates [10]	• May have limitations when it comes to advanced customization [11] • Extensive time is required to understand the platform's capabilities before efficiently building applications [11] • No flexibility in connecting with existing traditional software infrastructure [11] • Massively dependence on a no-code platform could interfere with performance in case the platform is phased out or undergoes complex migration [11]

The intended result of this thesis is to add a perspective to this discussion on robot manipulation planning for a cognitive robot system.

Related Work

- [1]: Block-based Programming Approach: Challenges and Benefits - Mohamad et al
- [2]: Visual Programming as Modern and Effective Structural Design Technology—Analysis of Opportunities, Challenges, and Future Developments Based on the Use of Dynamo - Paweł Grzegorz Kossakowski
- [3]: Dynamo Language Manual. Available online: https://dynamobim.org/wp-content/uploads/forum-assets/colin-mccroneautodesk-com/07/10/Dynamo_language_guide_version_1.pdf (accessed on 30 May 2023)
- [4]: Dynamo: Visual Programming for Design. Available online: https://help.autodesk.com/sfdcarticles/attachments/Dynamo_Visual_Programming_for_Design.pdf (accessed on 30 May 2023).
- [5]: Osmundsen (2019). An Inquiry into Low-Code Solutions in Institutions for Higher Education
- [6]: Marvin, R. (2014). How low-code development seeks to accelerate software delivery. SD Times, 1. Retrieved from https://sdtimes.com/application-development/low-code-development-seeks-accelerate-software-delivery/
- [7]: Richardson, C., & Rymer, J. R. (2014). New Development Platforms Emerge for Customer-Facing Applications. Retrieved from Forrester Research database
- [8] Feldman, D. (2013). Drag & Drop: Think Twice. Retrieved December 18, 2019, from Medium website: https://medium.com/@dfeldman/drag-drop-think-twice-49e7bf3e6b31
- [9] (Wayner, 2019)
- [10] Young Hyun C. (2019). Design and Implementation of a Low-Code/No-Code System

- [11] Northwest Executive Education (2023). 5 Pros and Cons of No-Code Development website: https://northwest.education/insights/careers/5-pros-and-cons-of-no-code-development/