W.T.P. TEACH-IN

Ankush Ahuja, Simon Chladek, Leopold Kernegger, Emily Kralik, Oliver Przewlocki, Danko Vukoja
Technologisches Gewerbemuseum (TGM)
Information Technology
Vienna, Austria
Contact: lkernegger@student.tgm.ac.at

***Abstract*— This paper examines the application of Teach-In programming to Obelix, a robot designed for competition, focusing on addressing the challenges of precision loss through simplified command execution. Employing the Create-3 Robot with a modified arm, this study departs from traditional Teach-In methodologies by recording commands instead of precise positional data to navigate the lack of accurate sensor information. Experiments demonstrate successful navigation but limited arm accuracy, underscoring the complexities of servo calibration and sensor integration. The study concludes that while Teach-In programming holds theoretical benefits for robotic programming, practical limitations regarding sensor feedback and mechanical precision render it suboptimal for competitive use.**

I. Introduction

In past endeavors, manually coding various values and calculating circular rotations to ascertain our robot's delays emerged as formidable challenges. This year, our robot, Obelix, benefits from our augmented knowledge. We previously grappled with escalating offset issues and implemented routines that inadequately corrected these offsets, resulting in collisions and a loss of precision. To address these issues this year, we sought to employ more sophisticated control methods for the robot.

A. Teach-In Rationale

Teach-In programming was chosen for its methodological simplicity, allowing a robot to be manually guided through a sequence of actions, then memorizing these actions to autonomously replicate them later. Appreciated for its ease of use, this approach facilitates an intuitive, hands-on programming experience. By leveraging Teach-In, we aimed to bypass the complexities associated with manual coding and precise position control, instead focusing on straightforward programming that could directly influence the robot's behavior. The robot would subsequently replay the input actions, mirroring the exact sequence and manner in which they were performed, enabling a precise and adaptable response to various tasks and challenges. This methodology promised a significant advantage in competitive settings, where the ability to swiftly adjust and execute strategies with high precision is essential.

II. Concept

To implement Teach-In for our robot, we focused on control and manipulation capabilities.

A. Robot

The Create-3 Robot, part of this year's Botball kit, was selected for its reliable motor functions and the capacity to support a heavier arm due to its size and weight.

B. Arm

The arm was designed to manipulate our environment. In an effort to avoid structural issues encountered last year, we adapted HTL-Anichstraße's design, simplifying it by eliminating unnecessary features for our experiment and competition needs, such as the rotating claw.

C. Program

Our program fundamentally differs from traditional Teach-In applications by saving commands instead of coordinates, due to our inability to accurately determine positions. This approach, while potentially less reliable, simplifies implementation in the absence of precise sensor data. Initially planning to use a controller, we ultimately opted for keyboard input to simplify the implementation and align with our model of movement execution.

III. Implementation

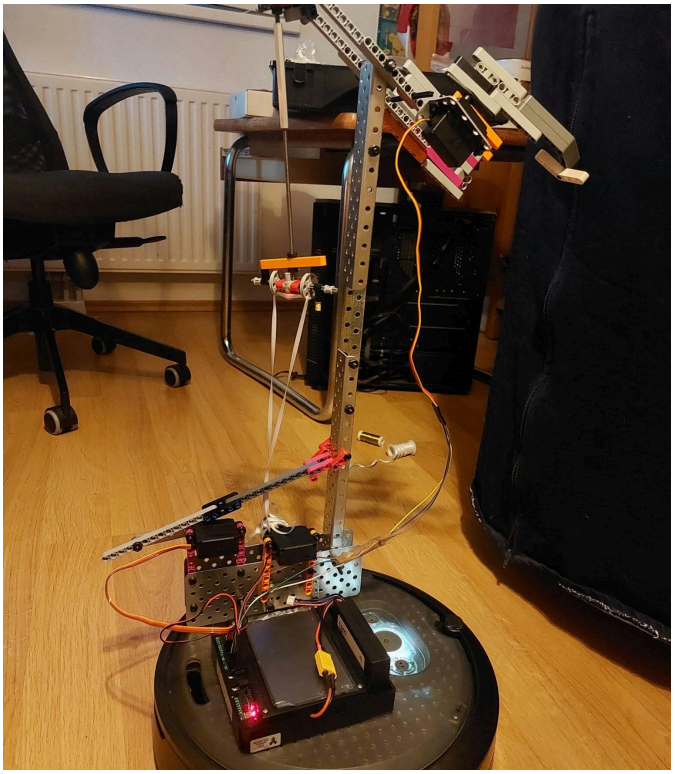
Our implementation slightly deviated from the original concept, particularly in the method of input, transitioning from a controller to a keyboard for easier integration and single-command execution, thereby avoiding the complexities of case handling for minimal performance improvement.

Pseudo code of our implementation:

```
def control_devices_and_robot(command_file):
    with open(command_file, 'a') as file:
        while True:
```

```
            command = input()
              # If 'x' is pressed, the loop stops,
marking the recording of steps as complete.
            if command == 'x':
                break
            execute_command(command)
            file.write(command + "\n")

def replay_commands(command_file):
    with open(command_file) as file:
        for command in file:
            command = command.strip()
            if command:
                execute_command(command)
                time.sleep(0.1)

def main():
    if not is_robot_connected():
        print("Failed to connect to robot.")
        return

     replay_option = input("Replay commands from a
file? (y/n): ")
    command_file = input("Enter command file name: ")

    if replay_option.lower() == 'y':
        replay_commands(command_file)
    else:
        control_devices_and_robot(command_file)
```

Photo of our finished robot with its arm:



IV. Experiment

A. *Create*

This experiment evaluated the reliability of programming Obelix (the Create Robot) without using the arm.

1) *Setup:*

A random routine was programmed into our Create, which started and ended at a fixed point (in this case, the docking station). We conducted several test runs from various starting locations to assess consistency.

2) *Results:*

Obelix reliably returned to the docking station in all five test runs, irrespective of the starting location.

3) *Conclusion:*

The built-in movement functions of the Create proved highly reliable, necessitating no further adjustments. This outcome demonstrates that with effective error correction, as seen in the Create3, the approach is viable in a competitive setting.

B. *Create + Arm*

This experiment tested the robot's ability to pick up an improvised object, introducing more complexity.

1) *Setup:*

The robot, positioned in front of a chair with an object on it, was tasked with picking up the object, returning to its starting position, and retaining the object in its claw.

Photo of the object:

2) *Results:*

Multiple attempts with varying starting positions and commands were unable to accurately replicate the movements for the robot to consistently pick up the object. Many attempts involved a plethora of individual commands for fine-tuning purposes, often engaging the servos, which proved less reliable than the Create3′s built-in movement functions. This led to compounded errors, such as the arm colliding with the chair on which the object was placed.

3) *Conclusion:*

The variability of the test setup precludes a definitive assessment of the code's efficacy. However, the consistent inaccuracies indicate the need for improvements in the design of the arm/claw and the integration of sensor-based correction.

V. Conclusion

Our approach was theoretically sound, but practical limitations and issues with the test setup resulted in a less reliable arm for competitive use. Implementing Teach-In at a competitive level would require comprehensive sensor integration and servo calibration, making the approach impractical. Consequently, we are unlikely to utilize this programming methodology in future competitions.

VI. Sources

[1] The concept of Virtual Teach-In in industrial robotics. (https://ieeexplore.ieee.org/abstract/document/8444250)

[2] Automated robot programming through teach-by-showing. (https://ieeexplore.ieee.org/abstract/document/933255)

[3] HTL-Anichstraße's Arm design. (https://www.youtube.com/watch?v=Wzuv4gFlMTU)

[4] Teach-In programming advantages. (https://www.learntechlib.org/p/35741/)

References