[MoMa 2023] Design and Implementation of a State Machine and GUI for the Megatron Robot System

Yaxun Yang, Fujing Xie, Bowen Xu
{yangyx12022, xiefj, xubw}@shanghaitech.edu.cn

November 2023

Abstract. This project introduces a comprehensive solution for robot control and task coordination of the Megatron robot utilizing the FlexBe state machine within the ROS (Robot Operating System) framework. The primary focus is on enabling efficient management of essential robotic tasks of the Megatron robot such as Navigation, Pickup Panel, and Place Panel activities. A significant innovation of this project is the development of a user-centric Graphical User Interface (GUI), tailored to operators unfamiliar with the complexities of state machines. This GUI allows users to easily monitor and control the robot's status and operations, enhancing the interaction between human operators and robotic systems.

1. Introduction

The project involves the integration of FlexBE (Flexible Behavior Engine)[1] with ROS (Robot Operating System) to create an advanced robot control system, complemented by a user-friendly Graphical User Interface (GUI). FlexBE is a behavior engine for ROS that facilitates the creation of complex robot behaviors without the necessity of manually coding them. It employs state machines which can be easily composed via a drag-and-drop editor, providing an intuitive approach to defining robot behaviors. Once these behaviors are designed, FlexBE allows for their execution and real-time monitoring, offering a high degree of collaborative autonomy. This means the operator can influence the execution during runtime, such as forcing transitions, and the robot can request help or confirmation from the operator when needed. The FlexBE behavior engine is a meta-package encompassing various packages that collectively enable the definition and running of robotic behaviors. This includes components like flexbe_core, flexbe_input, flexbe_mirror, and others, each serving specific roles in the behavior creation and execution process. Complementing FlexBE, the project also integrates a user-friendly GUI developed through QT Creator[2]. The FlexBe GUI serves as an interface for both editing and runtime control of the FlexBE behavior engine. And the user GUI this project developed is tailored to users who may not be familiar with the intricacies of state machines, allowing them to monitor and control the robot's status and behavior in a more accessible manner. The GUI incorporates essential control features such as 'STOP', 'Pause/Unpause',

[1] http://wiki.ros.org/flexbe
[2] https://github.com/qt-creator/qt-creator

and 'Confirm', facilitating user interaction and operational safety. Moreover, it supports different operational modes, like Semi-Autonomy and Full Autonomy, to cater to varying levels of robot autonomy and human intervention. This combination of FlexBE's state machine capabilities with an user-friendly GUI interface aims to enhance the efficiency and usability of Megatron robot systems, making sophisticated robotic operations more accessible to a broader range of operators.

2. State of the Art

FlexBE represents the state of the art in robotic behavior engines due to its sophisticated features that enable the creation of complex robot behaviors without the need for manual coding. It offers a user-friendly interface through a drag-and-drop editor, allowing seamless composition of state machines based on the robot's capabilities and standard functionalities. One of the defining characteristics of FlexBE is its facilitation of collaborative autonomy, which means that human operators can influence the behavior execution in real-time, for instance, by manually triggering transitions or by the robot requesting operator confirmation for certain actions. This flexibility is especially beneficial in scenarios like rescue operations or remote manipulation tasks, where adaptability to unforeseen circumstances is crucial. FlexBE's hierarchical state machine model ensures that states correspond to actionable behaviors and transitions are clearly defined in response to outcomes, setting it apart from other behavior engines. This approach allows for excellent operator integration and a comprehensive user interface. The system is designed to support both full autonomy and restricted manual intervention, empowering operators to modify the structure of behaviors dynamically during execution without the need for a restart. Moreover, the FlexBE framework is developed to ensure non-blocking state execution, which is crucial for allowing remote operator interactions with the robot behavior during runtime, enhancing the flexibility and responsiveness of robotic systems. For an understanding of how FlexBE compares with other state machines and behavior engines in terms of capabilities and applications, additional information can be found on the ROS Wiki page for FlexBE and the ROS Index page detailing FlexBE's functionality and usage. These resources provide insights into the benefits of using FlexBE over other state machines, as well as tutorials and examples of its application in various robotic systems.

3. System Description

3.1. FlexBe

The state machine depicted in the image outlines a robotic system's workflow for handling solar racks. The initial state is not visible, but the flow moves into a decision point, which likely determines the subsequent action based on certain conditions or inputs. One branch leads to a "Waypoint Navigation Robot," suggesting an automated navigation process where the robot moves to predefined coordinates. From the decision point, another branch extends to a series of actions involving the robot's interaction with solar racks. It starts with "y: Drive robot," where the robot moves to a specific location. The next step is "Pickup Solar Rack from truck," indicating a robotic arm or mechanism that grabs the solar rack. Following this, the robot "Place Solar Rack on rods," which implies placing the rack into a setup

position, possibly on a solar farm. The final action in this sequence is "Place Solar Rack on truck," suggesting that the robot can also load the racks back onto the truck, likely for repositioning or storage. This sequence suggests a cyclical process where the robot can both unload and load solar racks as needed. Throughout the process, the robot's actions are likely governed by a state machine inside, which is not fully shown. This internal state machine would handle the detailed decision-making required at each step to ensure the robot performs the correct action based on its current state and external inputs. The state of our project is shown in Figure. 1

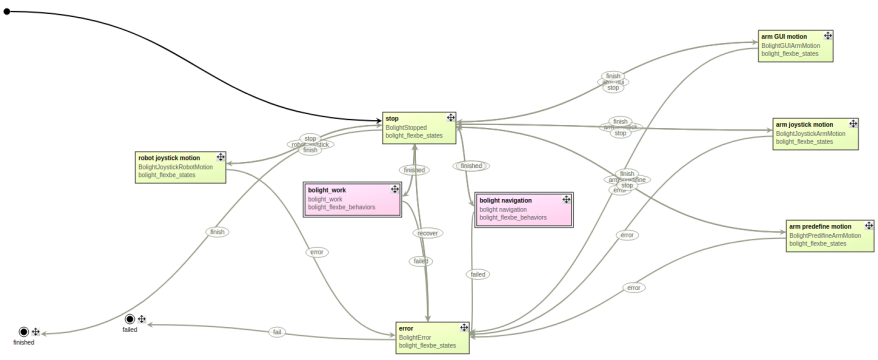

Figure 1. FlexBe of Megatron Robot

3.2. GUI

The GUI is tailored to allow a human operator to monitor and control the robot with varying levels of autonomy, providing both manual control and automated behavior adjustments. The interface is divided into several sections, each with a distinct function, shown in Fig.2:

3.2.1. Control Buttons

1. STOP: A prominent red button, to immediately halt all robot operations.
2. Pause: A button to temporarily stop robot movements.
3. Confirm: Used to confirm certain actions or decisions made by the robot.
4. Semi-Autonomy: The yellow button switch the robot to a mode requiring more operator input.
5. Next State: Progresses the robot to the next step in its task sequence when in Semi-Autonomy mode.
6. Full Autonomy: A green button, a mode where the robot operates independently of operator input.

3.2.2. Operational Buttons

1. Drive Joy: Joystick control for driving the robot.
2. Arm Joy: Joystick control for the robot's arm.

3. Various Arm Movement Buttons: Actions such as "Arm home," "Arm truck pre," and "Arm truck pick," performing precise control over the robot's arm for various tasks.

3.2.3. Status Panels

1. FlexBE State: Displays the current state of the FlexBE state machine, with a description area below.
2. Robot Status: Shows the robot's status, which includes wall time and other system information.
3. RViz Panel: Part of the RViz visualization tool used in ROS for visualizing the robot's environment, navigation paths, and sensor data.

3.2.4. Record Functionality

1. Record Bagfile: This botton is a feature to record the robot's data streams for later analysis or replay.

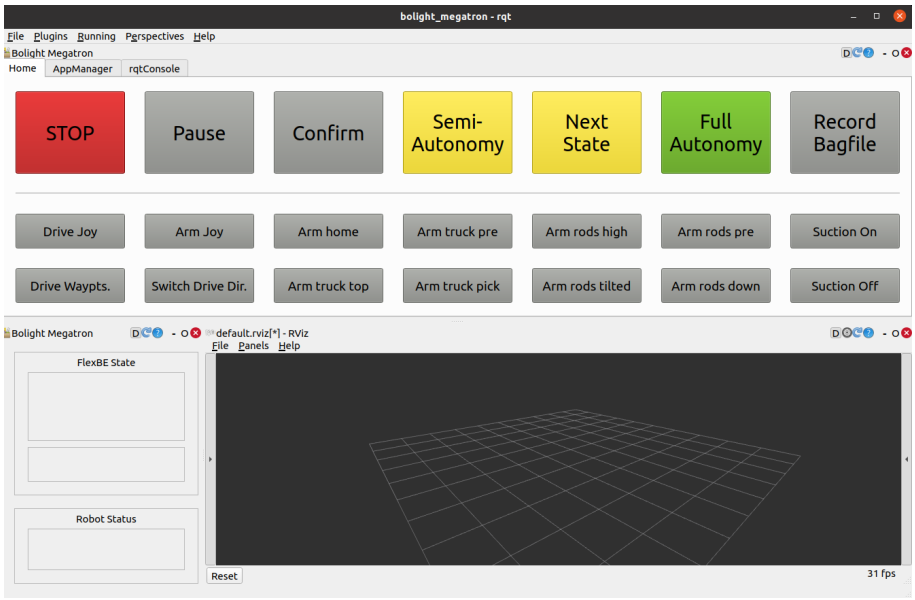

Figure 2. GUI of Megatron Robot

4. Conclusions

In conclusion, this project has successfully demonstrated the effective integration of FlexBE within the ROS framework to enhance the control and task coordination capabilities of the Megatron robot. Through the implementation of a user-friendly GUI, the project has addressed the common barrier of complexity associated with state machine operations, making it accessible to operators without advanced technical knowledge. The GUI's design facilitates not only real-time monitoring but also interactive control, contributing significantly to the robot's operational safety and flexibility. Moreover, the project's use of FlexBE's state-of-the-art behavior engine has allowed for the seamless creation and execution of complex robotic behaviors,

while also providing the ability to dynamically alter behavior execution in response to real-world variables. Future work could focus on expanding the robot's capabilities and further refining the interface to ensure that the Megatron robot continues to set the benchmark in autonomous robotic solutions.