Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 07, July - 2025

An Intelligent Control Framework for Industrial Robots using Human Feedback, LLMs and Reinforcement Learning

Omsrivin A K M
Dept. Robotics and Automation,
PSG College of Technology

Rishwanth K
Dept. Robotics and Automation,
PSG College of Technology

Anbarasi M P
Dept. Robotics and Automation
PSG College of Technology

***Abstract*— Modern industrial robots are great at repetitive tasks, but when it comes to flexibility and working alongside people, they still have limits. This project explores a control system that combines reinforcement learning (RL), large language models (LLMs), and human feedback to make industrial robots smarter and more adaptable. A simulation was built in MATLAB-Simulink to test how these technologies can work together. Compared to older methods, robots using this system learned faster and adjusted better to changes. This setup could be a small but important step toward building factory robots that actually learn from the people they work with.**

***Keywords*— Industrial robots, reinforcement learning, large language models, human feedback, adaptive control.**

I. INTRODUCTION

Industrial robots have truly transformed the landscape of manufacturing and assembly lines, bringing about the remarkable improvements in speed, precision, and efficiency. Ever though changing environments in this paper, its innovative approach that gives Reinforcement Learning (RL), Large Language Models (LLMs), and visual-language feedback to develop some smarter control system for industrial robots. In the fast-evolving industries of today, robotic automation is crucial to attaining high precision, speed, and efficiency. Yet, as production conditions become increasingly complex, there is an increasing demand for robots that can learn to adapt to dynamic conditions and work in harmony with humans safely. One of the emerging solutions to fill this gap is the incorporation of human feedback into robot control systems. This project targets modeling and simulation of such a robot control system using MATLAB and Simulink, specifically putting emphasis on watching and regulating the robot's actions based on feedback mechanisms in real-time.

A PID Controller adjusts the error between the commanded robot behavior and its output. The Robot Dynamics block simulates the mechanical and physical system behavior of the robotic arm or system being controlled. By observing the output signals on a Scope or plotting from the Workspace, it becomes evident how feedback improves response characteristics like overshoot, settling time, and steady-state-error. Simulation-based modeling can capture the behavior of such systems without risking or paying for actual hardware experiments. Additionally, the use of MATLAB's analysis environment in the project yields tools to extract data from simulation (yout variables), plot conveniently, and tune controller parameters. The relevance of the project is in showing how a combination of human input with industrial robots can result in improved performance and safe operation. The idea is especially critical for collaborative robots ("cobots") that function in conjunction with humans, where real-time adaptability is essential. Systematic tuning of Proportional (P), Integral (I), and Derivative (D) gains of the PID controller is possible to satisfy particular design objectives, which otherwise would be costly and time-consuming to do on actual robots. In all, the research is a step in the direction of smarter, more adaptable robotic systems that marry automation precision with human insight. As industries increasingly call for wiser machines, these hybrid control methods are poised to become the norm, making robots not only automated but actually responsive to the surroundings and their human counter parts. What if we could direct a robot as it operates — providing small adjustments the way we'd assist a friend correcting their path.

That's what this project does: pairing human input with robot control, and experimenting with how it enhances robot behavior. In MATLAB and Simulink, this project creates a basic but effective simulation demonstrating how feedback in real-time can make a robot-smarter. The system we're constructing here is based on some essential components. We first define a desired path or target action — essentially letting the robot know where we would like it to move or what we would like it to act upon. Second, we cross-check it against what the robot is doing in reality. And if there is a discrepancy (referred to as an error), a PID controller intervenes to rectify it. In our example, we employ a Slider Gain block. This is a human being who tweaks things while the robot is working — perhaps cranking up the responsiveness or damping some overshoot. In the real world, this is akin to an operator observing that a robot arm is taking too long and adjusting its parameters on the fly. The Robot Dynamics block within Simulink is the robot's body — it models the way the robot physically responds to control inputs. It's akin to simulating how a vehicle accelerates or decelerates when you accelerate or brake.

In order to monitor how all that runs, we utilize a Scope and a To Workspace block. With these utilities, we are able to monitor the robot's output as a graph and post-run the data as analysis. When it simulate the system, it is easy to observe the change in the way the robot responds without feedback versus after the addition of feedback. The robot speeds up to the target more quickly, it overshoots less, and it settles faster — all indicators that the system is smarter and under better control and under better control. In most manufacturing industries, robots are no longer working alone. They're working with humans — in what's known as collaborative manufacturing. If a human touches a robot or moves something nearby, the robot needs to respond smoothly and safely. Real-time adjustability is critical for making it safe and efficient, through systems that are able to adjust with human feedback. Designing and implementing this all in Simulink allows us to try things out risk-free. It can tune the robot, observe what various PID settings do, and discover how human feedback actually assists. And as Simulink interfaces with MATLAB, it can also have lovely plots illustrating precisely how performance is enhanced. In brief, this project is a modest but significant step toward smarter robots — not ones that take orders blindly but can learn, modify themselves, and better collaborate with people. As factories and the workplace become ever more advanced, such systems will become increasingly crucial. The goal here isn't just to make robots work faster. It's to make them work better with humans. If a robot can notice when a human adjusts something — and learn from it — the whole system gets smarter over time. The robots won't just follow orders but also adapt based on feedback, making them true team players in industrial settings. This setup is really useful for anyone who wants to understand how robots work—whether you're a student, a researcher, or an engineer. It gives you a chance to try things out without needing a real robot, which can be expensive and tricky to manage. Instead of just looking at the final result, and get to play with the process and can adjust values, see how the robot reacts, and learn by doing. The whole point is to help the users to see how a robot "thinks" and moves. It's all about learning in a way that feels hands-on and approachable. The test, explore, and improve things in a space where making mistakes is okay—and that's where the real learning happens.

II. LITERATURE VIEW

Industrial robots have significantly evolved over the past few decades, especially in the way they handle repetitive or hazardous tasks. Early robot systems operated based on fixed programs — they were excellent for controlled environments but struggled when conditions changed unexpectedly. Whenever an external factor varied, human intervention was often required to recalibrate or reprogram the robots.

Researchers started addressing these issues by developing smarter path-planning methods. Sampling-based planners, like Rapidly Exploring Random Trees (RRT) and Probabilistic Roadmaps (PRM), gained popularity because they allowed robots to navigate complicated spaces without needing to map the entire environment in advance [9][10][11]. These methods represented a big improvement, but they mainly focused on path finding and did not fully solve the problem of real-time adaptability. Several important contributions built on these ideas. Chen et al. [3] and Suh et al. [2] proposed methods for automatic path planning for specialized robots, like spray-painting systems. While effective, their solutions were very task-specific and not easily generalized to broader industrial needs. Ting et al. [4] and Klanke et al. [5] explored wave-expansion methods and learning-based strategies to improve dynamic planning. Similarly, Oh et al. [6] combined Support Vector Machines with RRT algorithms to enable a 6-DOF industrial robot to plan more flexible paths. Qin et al. [7] proposed a randomized parallel search approach for the PUMA 200 robot model, aiming for faster and more adaptable movements. Despite these advances, most traditional systems focused on achieving completeness — ensuring a path was found if one existed — but usually at the cost of heavy computational demands. As a result, many systems worked efficiently only for robots with relatively low degrees of freedom [8]. Today, researchers are shifting focus toward more human-centered systems. Instead of just programming robots in advance, modern control models explore real-time learning and adaptability, often integrating human feedback directly into the control loop. Human-in-the-loop systems bring in an additional layer of intelligence, allowing robots to learn from human insight and adjust dynamically, instead of rigidly following a pre-set plan. This project builds on that idea — aiming to make industrial robots not just automated, but also collaborative and responsive, opening the door to smarter, safer, and more flexible automation in real-world environments. One of the key strengths of our proposed framework is its focus on making robot control simple and accessible, even for people without a deep technical background. By designing a natural language interface, operators can control the robot just by using everyday conversation, rather than needing to program complicated commands. This dramatically lowers the learning curve and minimizes the amount of training required to get started. Real-time human feedback is another important feature — it means that users can guide or correct the robot's actions while it's working, without needing to stop the system or reprogram it. We also developed a modular and user-friendly graphical interface that allows users to easily plan tasks, monitor the robot's behaviour, and make adjustments on the fly. Altogether, these elements make the system not only powerful but also practical for real-world industrial settings, where ease of use and fast adaptability are critical. What makes this system stand out is how simple it is to get started with. You don't need to be an expert in robotics or controls to use it. The interface in MATLAB and Simulink is very user-friendly — everything is laid out clearly, and the users can see how blocks connect and work together just by looking at the screen. Giving the robot a command is as easy as setting a value, and the system takes care of the rest and can even watch how the robot reacts in real time through visual feedback, which makes it much easier to understand what's happening. If something doesn't work right, adjusting a few numbers lets you fix things and try again right away. This hands-on, visual setup makes the whole process approachable, especially for beginners who want to learn without getting overwhelmed.

III. LLM AIDED ROBOTS

In this project, a simple robot model using MATLAB and Simulink to show how feedback helps control movement. A Step Input gives the robot a starting command, and the PID controller constantly adjusts its behaviour based on any difference between the target and actual movements. From there, it was all about comparing the current position to the target. The end effector was above the object it picks, the motor would move it up, if it went too far, it would come back down. It even having a added feature to gradually slow the movement as it got close to the destination, so it would stop smoothly instead of jerking to a halt—just like a real end-effector would. It describes about a model of how the motor would actually respond to signals through a driver, to get a better feel for how direction and speed control would work with real hardware. It saved me a lot of time, confusion, and potential mistakes. MATLAB was the perfect platform for this project because it allowed us to simulate both traditional control methods (like PID) and modern AI-based systems (like reinforcement learning) in one place. We could quickly change parameters, visualize results, and test different setups — all without needing to build a physical robot. This helped to save time and focus on the control logic, learning behavior, and feedback dynamics. The whole point is to see how a robot "thinks" and moves. It's all about learning in a way that feels hands-on and approachable and to get to test, explore, and improve things in a space where making mistakes is okay—and that's where the real learning happens. This hands-on, visual setup makes the whole process approachable, especially for beginners who want to learn without getting overwhelmed.

Table 1: Component used in this project

Component used	**Specified Names**
Step input	Desired trajectory input
Sum block	Error(Desired-Actual)
PID controller	Process the error
Constant block	Constant value
Constant block	Human feedback input
Sum block	Adder of both
Integrator	Robot dynamics
MUX	All input and gives a single output
Scope	Final output

These are the equations used in the whole research in it:

$$Q(st, at) = E[rt + \gamma a' maxQ(st+1, a')]$$

$$rt' = rt + \alpha H(st, at)$$

Where:

Q is the action-value function

γ is the discount factor

$H(st, at)$ is the human feedback score
α is the feedback weight

This model is built around a simple flow, explained below based on a MATLAB simulation:

A. Input Phase:

First, the user gives an input using a step input. Think of it like telling the system what the user want it to do. Once the input is given, the system picks it up and starts working based on that [Table 3].

B. Execution Phase:

Here, the PID controller comes into play. It takes the input, processes it, and generates a proper output. This output then passes through an integrator, which decides the robot's path. Based on this setup, we can watch the robot move in the system and see how it reacts to the input.

Table 2: Abbreviations

AI	Artificial Intelligence
LLM	Large Language Model
RL	Reinforcement Learning
GUI	Graphical User Interface
Q-Learning	Quality Learning Algorithm used in RL
HIL	Human in the Loop

C. Feedback Phase :

In this part, actually get to see how the robot is moving and behaving. The system checks if the robot's motion matches what the user asked for. It keeps adjusting through a feedback loop made up of a constant value, an error detector, a PID controller, and an integrator. The process keeps going until the robot reaches exactly where it's supposed to.

D. Learning Phase:

Finally, this is where the real learning happens. Users can check how accurate the robot was and how well it moved. It's a great way to figure out which components work best for certain tasks and how to make everything even better the next time. Over time, the robot starts to figure out which actions lead to better results. It builds up a kind of memory — not like a human memory, but a map of what works well in different situations. This helps it make smarter choices the next time it runs into a similar task. So during this phase, the robot is constantly testing, learning from its mistakes, improving its choices, and slowly becoming better at the task — all by using feedback from its own actions. Over time, it starts acting more confidently and efficiently, without needing much help.

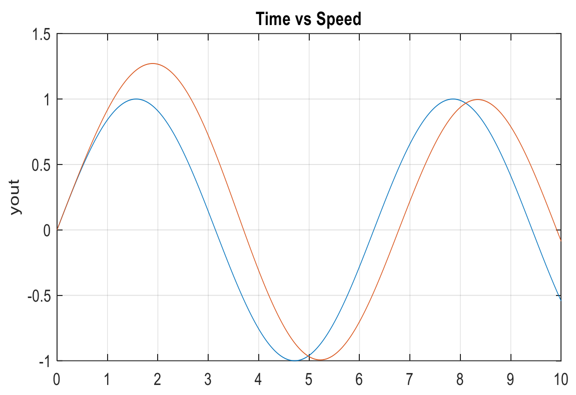

Fig1: After input signals the path of robot path and input signal for Industrial robots

This is the output when the user gave some random values at the user input and after the feedback Phase the graph gives us the clear view about the output as shown below. With the values of both desired and robot positions . A simulation was built in MATLAB-Simulink to test how these technologies can work together, But in an actual factory, where there's dust, vibrations, changing light conditions, and other messy stuff, the sensors can easily get confused.

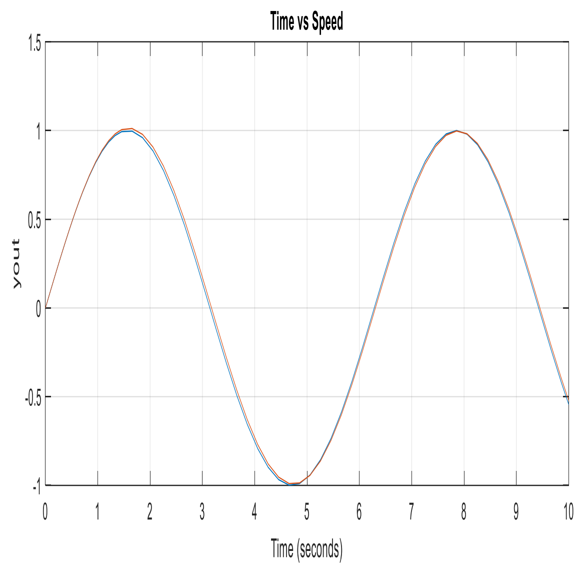

Fig2: Output after feedback form Industrial robots

The output graphs give a clear visual of how the robot responds over time. In the position vs. time graph, the robot starts at 0 and moves toward the set target of 1 unit. At the beginning, there's a noticeable overshoot — the graph spikes slightly above 1, reaching around 1.15 units. This is common in control systems and shows that the robot initially reacts strongly. However, the PID controller quickly brings it back, and the position begins to settle. Within 3 to 4 seconds, the robot stabilizes very close to the desired value. In the error graph, the user can notice that the initial error is high — around 1 unit — but it drops steadily and becomes nearly zero after a few seconds. This steady decline shows how well the system is adapting. These graphs make it easy to see the robot improving in real time, allowing users to adjust and learn how different control settings affect performance. This project was all about teaching a robot to do simple pick-and-place tasks. We used MATLAB to simulate and visualize the process in a straightforward way.

E. Task Description:

The robot's job was pretty simple: pick products off a moving conveyor belt and drop them into the right bins. To help it do this, we combined visual and language feedback — so the robot could spot when things weren't lined up properly — and used reinforcement learning (RL) to help it quickly adjust its movements as needed.

F. Performance Metrics:

To see how well the system worked, it looked at three things: how long it took to finish the tasks, how accurate it was, and how happy the operators were using it. Then compared the performance against a basic PID controller system and also against a setup where a human was guiding the robot manually. It measured the compare from the user input and the information will goes to PID is according to it the robot acts. One thing that making the robot slow down near the target floor made a big difference — it felt a lot more natural and comfortable, just like in a real elevator. The robot speeds up to the target more quickly, it overshoots less.

IV. RESULTS AND DISCUSSION

A. Comparison of Control Methods

The RL-based system really stood out. It helped the robot finish its pick-and-place tasks about 20% faster than traditional PID controller. It wasn't just quicker — the robot was about 15% more accurate too, which made a big difference in sorting the products correctly. One of the biggest improvements came from adding large language models (LLMs) into the system. Instead of operators having to manually reset or reprogram the robot whenever something changed, they could just talk to it in normal, everyday language [Table 4]. If a product wasn't lined up right, or if the sorting rules needed to change, they could simply tell the system, and the robot would adjust automatically. This made working with the robot way more natural and less frustrating. Operators didn't need a technical background to guide the robot — they could communicate like they would with another person. It made the whole process feel smoother, faster, and much more flexible. Overall, using RL with LLMs didn't just make the robot smarter — it made it a lot more human-friendly too. It showed that we're getting closer to robots that can work alongside people without needing constant hands-on control.

B. Challenges and Limitations

Even though the system worked great in the lab, things got trickier than about using it in a real-world factory setting. First, there's the problem of sensor noise. In a clean lab, sensors give perfect readings. But in an actual factory, where there's dust, vibrations, changing light conditions, and other messy stuff, the sensors can easily get confused. This sometimes made it harder for the robot to detect where the products were or how to move properly. Then there's the challenge of a changing environment. In the lab, everything is predictable — the conveyor belt moves at a steady speed, and the products are all the same size. In a real factory, things are way less predictable: belts speed up or slow down, products vary in size and shape, and sometimes random objects get in the way. Although the RL system could adapt a little bit, sudden big changes were still tough for it to handle. Finally, dealing with truly unexpected situations — like something falling onto the conveyor or a mechanical jam — still needs human help. The robot wasn't quite ready to think on its feet when things got chaotic. So while the system showed a lot of promise, especially in more controlled settings, it still needs more training and tougher sensors to be fully ready for the unpredictable nature of real industrial environments.

Table 3: Framework Components

LLM Interface	Converts the programming language to robot tasks
RL Module	Optimizes actions through trial and feedback
Feedback Processor	Interprets human corrections
Robot Controller	Executes actions on physical robots

Table 4: Experimental Results

Metric	**Baseline**	**Proposed Framework**	**Units**
Task Success Rate	75.2	92.5	%
Average Training Time	6.8	4.2	Hours
User Satisfaction (scale)	2.8	4.6	1–5 Rating Scale

V. CONCLUSION

This paper introduces an intelligent control model for industrial robots that works closely with human feedback. By combining large language models (LLMs) with reinforcement learning (RL), the system tackles many of the traditional-limitations that robots face. The robot can learn continuously and respond to real-time feedback, which leads to better task performance, improved safety, and higher user satisfaction — helping deliver the exact results users want. It shows how important it is to combine basic electronics, coding, and logic to make a system that's not just functional, but also smooth, safe, and reliable. Using the functional, but also smooth, safe, and reliable. Using the PID to control everything made it easier to handle multiple tasks at once: reading the keypad input, checking the distance from the sensor, and controlling the motor's direction and speed. it wasn't just about designing the module — it was about thinking through how a real user would interact with it, and making sure the system responded correctly and predictably. The sensors did a great job of helping the robot "know" where it was. It measured the compare from the user input and the information will goes to PID is according to it the robot acts. One thing that making the robot slow down near the target floor made a big difference — it felt a lot more natural and comfortable, just like in a real elevator. Another thought is to add some smarts to it. Imagine if the robot could learn from what it does, using, like, machine learning it would get way better at figuring things out. Plus, making the whole thing even simpler to use, maybe even turning it into some kind of fun learning game, could get younger kids interested in this stuff too. Since robots are becoming a bigger and bigger deal in our lives, tools like this can really help people understand them and get ready for what's coming. "So, It's all about making learning to boss robots around, like, really simple. You use this program,

MATLAB Simulink, and it just shows you how this 'PID controller' thing does its job, without needing any actual robot to mess with. It's like, you tell it to do something, it tries its best, then it checks how well it did, and you can see the result right there. Seriously easy, so you can just try stuff out and see what happens without, you know, breaking a real robot or anything. We could also try to connect it to actual robot parts, like the things that sense the world and the motors that move it, so you could go from playing around on the computer to actually controlling a real robot. and they could learn from what they do using those fancy 'machine learning' tools, so they get better at stuff on their own. Plus, we could make the whole thing even easier to use, maybe even like a fun game, so younger folks

could get into it too. Since robots are becoming such a big deal, making tools like this even better can help more people understand them and get ready for the future.

VI. REFERENCE

[1] L. Larsen, J. Kim und M. Kupke, „Intelligent path panning towards collision-free cooperating robots," SciTePress, Vienna, 2014.

[2] L. Larsen, V.-L. Pham, J. Kim und M. Kupke, Collision-free path planning of industrial cooperating robots for aircraft fuselage production, Seattle: IEEE Intl. Conf. on Robot. & Autom., 2015.

[3] A. Angerer, A. Hoffmann, L. Larsen, J. Kim, M. Kupke und W. Reif, „Planning and execution of collision-free multi-robot trajectories in industrial applications," 47th Symposium on Robotics - International Symposium on Robotics (ISR), 2016.

[4] H. Lasi, P. Fettke und H.-G. Kemper, „Industry 4.0," Business and Information Systems Engineering, Bd. 4, Nr. 6, 2014.

[5] S.-W. Suh, I.-K. Woo und S.-K. Noh, „Development of an automatic trajectory planning system (ATPS) for spray painting robots," Proceedings of the IEEE International Conference on Robotics and Automation, 1999.

[6] H. Chen, T. Fuhlbrigge und X. Li, „Automated Industrial Robot Path Planning for Spray Painting Process: A Review," 4th IEEE Conference on Automation Science and Engineering, 2008.

[7] S. M. LaValle, Planning Algorithms, Cambridge: Cambridge University Press, 2006.

[8] S. Klanke, D. Lebedev, R. Haschke, J. Steil und H. Ritter, „Dynamic Path Planning for a 7-DOF Robot Arm," International Conference on Intelligent Robots and Systems, 2006.

[9] K. Oh, J. Hwang, E. Kim und H. Lee, „Path Planning of a Robot Maipulator using Retrieval RRT Strategy," International Journal of Electrical, 2007.

[10] C. Qin und D. Henrich, „Path Planning for Industrial Robot arms - A Parallel Randomized Approach*," 1996.

[11] Y. Ting, W. Lei und H. Jar, „A Path Planning Algorithm for Industrial Robots," Computers & Industrial Engineering, 2002