3×3 Rubik's Cube Solver Using Computer Vision, ROS2 and ESP32 Controlled Mechanism

Md. Sheikh Sadi
Computer Science and Engineering
United International University
Dhaka, Bangladesh
msadi223646@bscse.uiu.ac.bd

Badhon Dalbot
Computer Science and Engineering
United International University
Dhaka, Bangladesh
bdalbot223660@bscse.uiu.ac.bd

Mominul Islam
Computer Science and Engineering
United International University
Dhaka, Bangladesh
mislam223425@bscse.uiu.ac.bd

Sayed Hasan Sami
Computer Science and Engineering
United International University
Dhaka, Bangladesh
ssami2230939@bscse.uiu.ac.bd

Md. Mahmud Hasan
Computer Science and Engineering
United International University
Dhaka, Bangladesh
mhasan2230995@bscse.uiu.ac.bd

***Abstract*—This paper presents an automated system for solving a standard 3×3 Rubik's Cube using a combination of computer vision, robotics middleware and a custom mechanical mechanism. The system captures cube face colors with a mobile/web interface, computes an optimal solution using the Kociemba algorithm, and actuates eight servo motors via an ESP32 to manipulate the cube—all coordinated via ROS2. Experimental results demonstrate complete cube solves within ∼96 s, with 98% color detection accuracy in various lighting conditions.**

***Index Terms*—Rubik's Cube solver, ROS2, ESP32, computer vision, 3D printed mechanism, Kociemba algorithm**

I. Introduction

Problem Statement. Solving a 3×3 Rubik's Cube quickly and automatically remains a challenging task that combines computer vision, planning algorithms and precise robotic manipulation. Manual solving is accessible, but reliable automation requires robust color detection, efficient solution computation and well-synchronized actuation. **Literature Review.** Prior work includes robotic cube solvers using stepper motors and simple webcam setups [1], [2]. Some systems rely on dedicated microcontrollers and proprietary drivers [3]. These often trade off modularity or network-based control in favor of pure hardware speed. **Innovative features added to our project.** Our implementation integrates a mobile/web front-end (for image capture) with the ROS2 middleware, enabling the cube to be manipulated via a publish-subscribe architecture. We also employ an ESP32 controlling eight MG90 servo motors and a fully 3D-printed rack & clamp mechanism—offering cost-effective, scalable hardware.

II. Proposed Method

A. Block diagram of the overall system

The system comprises three major layers: the user interface (mobile or web), the backend computing & ROS2 broker, and the mechanical actuation layer. The UI captures six cube faces, the backend uses OpenCV to detect sticker colors and computes a solution via the Kociemba algorithm. That solution is published on a ROS2 topic, subscribed by a node running on a laptop. The laptop forwards individual moves serially to the ESP32, which triggers eight servo motors to manipulate the cube via a 3D-printed rack, clamp and gear system.

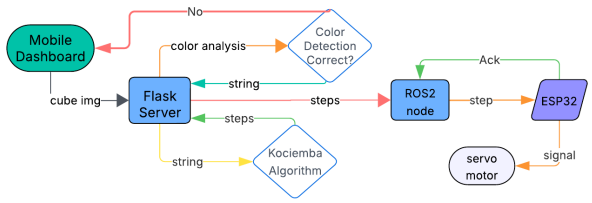

Fig. 1. Block diagram of the automated cube solving system.

III. Implemented Hardware System

A. List of the hardware and software environment

Hardware: ESP32 microcontroller, eight MG90S 9 g servo motors, 3 D-printed parts including rack, rack top, clamp, rack base and spur gear assembly, standard 3×3 Rubik's Cube.
Software: Ubuntu 22.04 LTS with ROS2 Humble, Python 3.13, Flask (backend), OpenCV, Kociemba cube solver library, Web frontend using HTML, CSS, JavaScript.

B. Figures of the implemented hardware system

C. Configure Tables

Table I presents the hardware specifications used in the project, while Table II shows the HSV color detection thresholds used to identify each sticker color during cube face recognition.

The project was carried out as part of the microprocessor & microcontroller lab course requirement in the Department of Computer Science and Engineering at United International University.

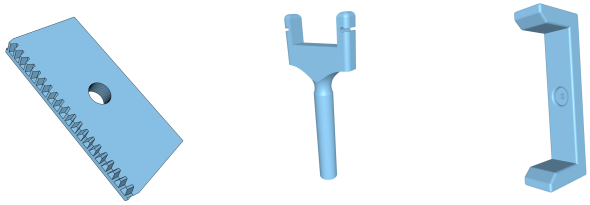

Fig. 2. Left: rack. Center: servo holder Right: clamp cube holder.

Fig. 3. Left: rack top. Center: rack base. Right: spur gear

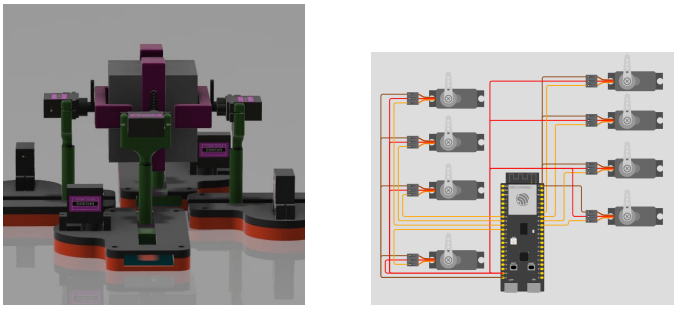

Fig. 4. Left: MG90S servo mounted on position. Right: ESP32 and wiring interface.

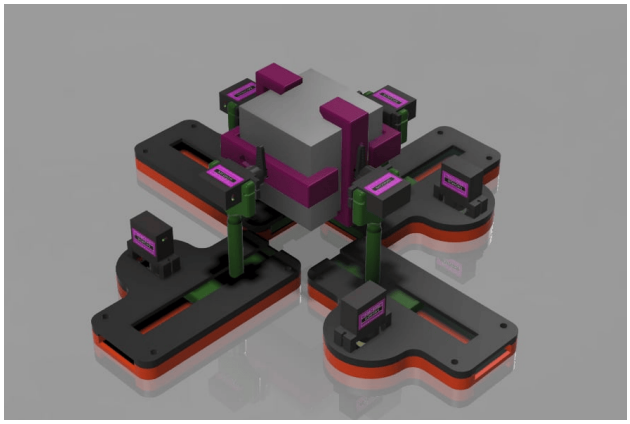

Fig. 5. final setup

TABLE I
SERVO MOTOR AND HARDWARE SPECIFICATIONS

Component	Specification	Value
Servo Type	MG90S Micro-Servo	9 g, 180° range
Microcontroller	ESP32	Dual-core, WiFi/Bluetooth
3D Printed Parts	Material	PLA+, custom rack, spur and clamp design

TABLE II
HSV COLOR DETECTION RANGES FOR EACH CUBE FACE

Side / Color	Lower HSV Range	Upper HSV Range
White	(0, 0, 120)	(180, 50, 255)
Yellow	(20, 50, 80)	(35, 255, 255)
Red (1)	(0, 120, 120)	(8, 255, 255)
Red (2)	(165, 120, 120)	(180, 255, 255)
Orange	(8, 100, 100)	(25, 255, 255)
Green	(40, 50, 50)	(85, 255, 255)
Blue	(90, 50, 50)	(130, 255, 255)
Black	(0, 0, 0)	(180, 255, 80)

The HSV values in Table II were calibrated manually under controlled lighting conditions. The ranges allow reliable separation of similar hues (e.g., red and orange) and maintain stability across different brightness levels.

IV. RESULTS

The system was evaluated by capturing 20 random cube configurations under varying lighting and executing the full solve sequence. Color detection achieved an average accuracy of 98.2%, and the full mechanical solve completed in 85–110 s depending on re-grip frequency. Table III summarizes the key performance metrics.

TABLE III
PERFORMANCE SUMMARY OF RUBIK'S CUBE SOLVER

Metric	Result
Average Color Detection Accuracy	98.2%
Average Solve Time	96.4 s
Fastest Solve Time	85 s
Slowest Solve Time	110 s
ROS2–ESP32 Communication Delay	$\approx$ 8.7 ms
Servo Motor Positioning Error	$\pm 3.8^\circ$

The system demonstrated reliable communication and consistent solving accuracy even under moderate lighting variations.

V. DISCUSSION

The project demonstrates a cost-effective and modular approach to automated cube solving. However, limitations include the latency of sequential servos (90° rotations followed by 180° doubling) and physical tolerance in the 3D-printed parts which cause ±2° mis-alignment. Alternative designs could employ stepper motors or continuous servo systems to reduce repositioning latency (P3). From an ethical viewpoint, the system uses open-source software and reused electronic components, aligning with sustainable design practices (P5). The hardware and software layers are decoupled via ROS2 topics, but they remain inter-dependent when timing constraints require hardware state to be communicated (P7).

P1	P2	P3	P4	P5	P6	P7
✓		✓	✓	✓		✓

TABLE IV
COMPLEX ENGINEERING PROBLEM MAPPING.

VI. CONCLUSION

This project presents an integrated hardware-software system capable of solving a standard 3×3 Rubik's Cube automatically. The system achieved high color-detection accuracy and reliable mechanical performance. In future work we plan to incorporate parallel actuation of multiple faces and explore machine-learning-based sticker detection to adapt to more cubes and lighting conditions.

ACKNOWLEDGMENT

The authors would like to express their sincere gratitude to the course faculty of the Department of Computer Science and Engineering, United International University, for their continuous guidance, supervision, and encouragement throughout the project. Special thanks are also extended to the UMRT members for their assistance with fabrication support.

REFERENCES

[1] M. Hoffmann, "Cube Explorer — Optimal Solver for Rubik's Cube," 2021. [Online]. Available: https://kociemba.org/cube.htm

[2] A. Kociemba, "Two-Phase Algorithm for Rubik's Cube," 2025. [Online]. Available: https://www.jaapsch.net/puzzles/cubic3.htm

[3] J. T. Tørresen and J. E. Gustavsen, "Rubik's Cube Solving Robot with Computer Vision and Kociemba Algorithm," in *Proc. IEEE Int. Conf. Mechatronics and Automation*, 2017, pp. 134–139.

[4] S. P. Rohith, A. C. Varghese, J. Thomas, and M. M. Mathew, "Autonomous Rubik's Cube Solver Bot," *Int. J. of Scientific Research and Engineering Development*, vol. 2, no. 3, pp. 1–6, 2019. [Online]. Available: https://www.academia.edu/41627834/Autonomous_Rubiks_Cube_Solver_Bot

[5] S. Sadi, B. Dalbot, S. H. Sami, M. Hasan, and M. Islam, "GitHub Repository of Project Source Code," *GitHub*, 2025. [Online]. Available: https://github.com/5S4D1/cube_solver_workspace