2021 International Conference on Automation, Robotics and Applications

Advanced Rubik's Cube Algorithmic Solver

Vasile Dan, Gabriel Harja, Ioan Naşcu
Department of Automation,
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
vasidan.mdt@gmail.com, Gabriel.Harja@aut.utcluj.ro, Ioan.Nascu@aut.utcluj.ro

2021 7th International Conference on Automation, Robotics and Applications (ICARA) | 978-1-6654-0469-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICARA51699.2021.9376564

***Abstract*—Rubik's cube is one of the most popular educational and challenging toys. People have tried to solve the puzzle by themselves or build robots that can do it. In this paper, has been proposed a technical solution for building an Advanced Rubik's Cube Algorithmic Solver (ARCAS), that uses a PC and an Arduino Due board as processing units. The design of the robot allows instant color recognition, by using 4 webcams positioned to capture every facet of the cube. Color image segmentation is used to find the HSV mask of the scrambled cube, thus the pattern will be obtained. The images are processed in a C# desktop application. An implementation of Kociemba's algorithm and blindfolded method, Old Pochmann, and M2, is used to solve the cube. The blindfolded method is implemented in a novel way to fit the mechanical structure of the robot and to reduce the time of solving.**

Keywords—Rubik's cube, image processing, blindfolded method, robot

I. Introduction

In a world of technology, some robots can improve the way we live. They are used in medicine, industry, sport, and army.

The popularity of the Rubik's cube was constantly growing, due to its uniqueness. Solving a Rubik's cube has been a challenge for many people, due to the complexity of the puzzle. It has 6 faces, each one containing 9 pieces, that can move individually, except the centers. The cube can be scrambled in 43 quintillion combinations by permuting the pieces.

There are multiple methods for solving the cube, of which the following will be mentioned: Fridrich, Roux, ZZ, Petrus, M2, Old Pochmann and Kociemba's algorithm. Some of these methods are used for speedsolving, others for blindfolded solving and others for solving the cube using as few moves as possible.

Herbert Kociemba's two-phase algorithm was implemented in different autonomous Rubik's solvers, due to the short given solution, which is equal to or less than 20 moves [1]. The mechanical structure of the robots makes them unique and even capable of solving puzzles with higher complexity. These puzzles can have more layers, the cube has more slices, therefore more squares on the faces. Also, the puzzles can have a different shape.

People from different fields of study have used Rubik's cube for scientific research and personal projects [2]. In [3], the authors have discussed the algorithmic structure of the Rubik's cube and the minimum number of moves for solving it. The paper [4], presents an algorithm with a low number of moves, based on Thistlethwaite's approach. In paper [5], Simulated Annealing and Genetic algorithm approaches are proposed. The process of building a robot for solving the cube using an Arduino Mega 2560 and using HSV color space for color recognition, was presented in detail in paper [6].

There are different designs of robots, that are using the optimal algorithm for gaining speed and solving the Rubik's cube. MindCuber uses Lego Mindstorms NXT kit, with a minimum number of motors and a color sensor for detecting the colors of the cube [7]. Ruku robot uses a Raspberry Pi board, an iPhone for data acquisition and steppers motors for rotating the faces of the cube [8]. A 3D printed robot, High Frequency Twister, uses an Arduino Atmel for controlling the 6 steppers that are rotating the faces of the cube and 4 webcams for color recognition [9].

Image processing is one of the most important parts of the project. Negative factors, light and its reflection were minimized, by using the HSV color space and LEDs. After identifying the initial state of the cube, the next step is to generate the solutions. This paper deals with image processing, Kociemba's algorithm and blindfolded methods, M2 and Old Pochmann. The faces of the cube are rotated by six actuators according to the chosen solution.

The paper is organized as follows: the next section gives an overall description of the hardware implementation and image recognition. Section III presents the solving algorithms, while Section IV gives details about the desktop application. The last section summarizes the results and conclusions.

II. ARCAS Structure Description

A. Hardware Implementation

The main components of the project interact with each other accordingly to the block diagram shown in Fig. 1.

The process of solving the Rubik's cube begins with the acquisition of data from 4 webcams. They are positioned in four different corners of the cube at 200 mm. Two of the webcams have been positioned at the top and the other two at the bottom, thus all the pieces are visible.

The colors on each side of the cube are identified and the two implemented methods, Kociemba's algorithm and blindfolded method, are applied. The generated solutions are sent to Arduino Due via the serial interface. Six stepper motors will rotate the faces of the cube, according to the chosen method. The 4 digits of the 7-segments display will show the solving time.

Authorized licensed use limited to: University of Ghana. Downloaded on May 17,2021 at 10:10:30 UTC from IEEE Xplore. Restrictions apply.

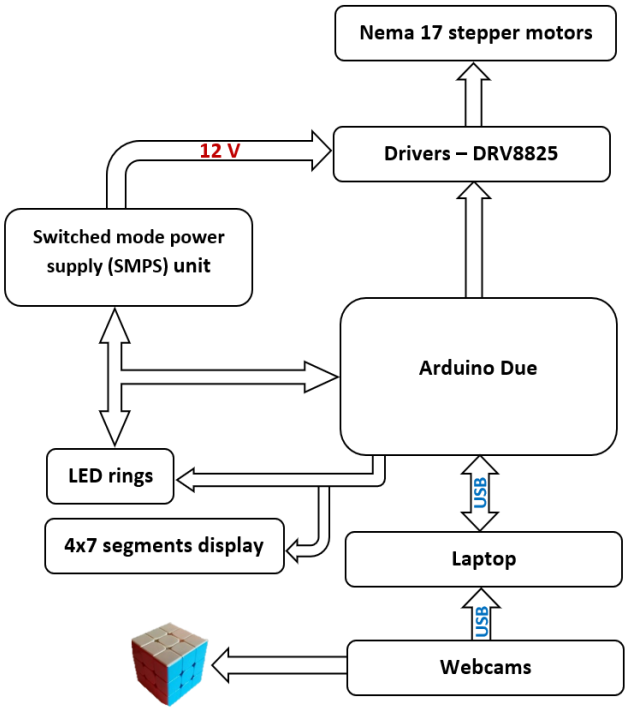

Fig. 1. Block Diagram of ARCAS.

The fastening system involves modifying the cube. The centerpiece of each face was drilled and sanded, to create four holes of 4mm x 4mm x 7mm, as shown in Fig. 2. Thus, the color of the centerpiece will be visible to the webcams.

The structure of ARCAS was design in AutoCAD and the pieces were printed on a 3D printer. The mechanism of the robot, based on rack and pinion pieces, contains two additional stepper motors that will move automatically the upper and the side arms that hold in place the cube, as shown in Fig. 3. Thus, the cube can be easily placed in the lower arm of the robot, and then, the arms of the cube can be controlled from the desktop application.

The final enclosure of ARCAS, after all the electrical connections were made, can be seen in Fig. 4.

Fig. 2. Modified Rubik's Cube.

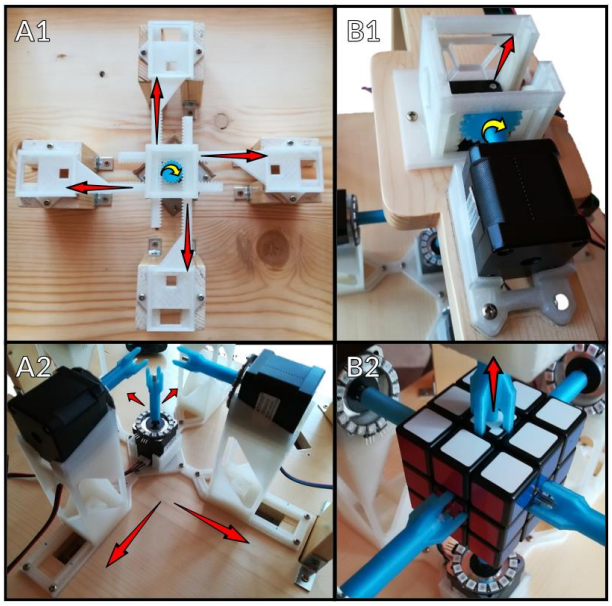

Fig. 3. Rack and pinion mechanism: (a) Side stepper motors; (b) Upper stepper motor

B. *Color Recognition*

The acquisition of data from the four webcams will determine the initial state of the cube. The color recognition algorithm can be divided into five steps:

1) Converting the color space of the image: from BGR (Blue, Green, Red) to HSV (Hue, Saturation, Value). The advantage is that the color can be modified using only one variable: Hue. The other two parameters are used for variating shadow and shine.

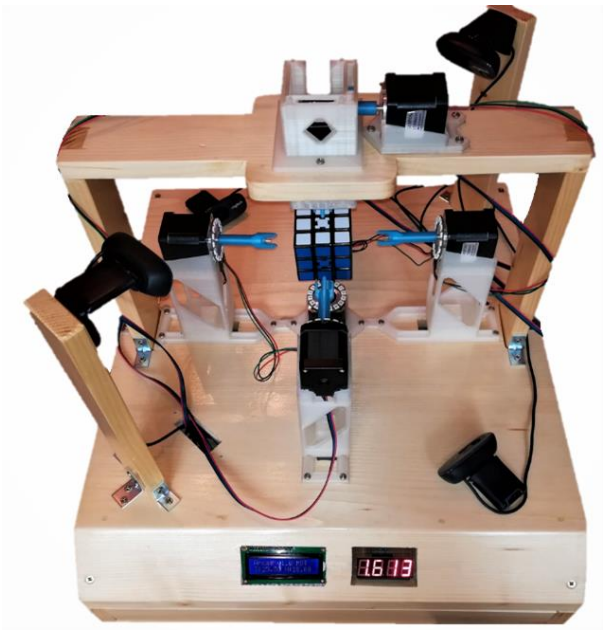

Fig. 4. ARCAS Structure.

Authorized licensed use limited to: University of Ghana. Downloaded on May 17,2021 at 10:10:30 UTC from IEEE Xplore. Restrictions apply.

2) Image filtering: using EmguCV library, SmoothMedian filter, for reducing the salt and pepper noise.

3) Selecting the regions of interest from the four images. These regions consist of one pixel on each piece of the cube, due to the previously applied filter.

4) Color identification, based on the 3 values of the pixel: hue, saturation, and value. For each of the six colors, there are 3 intervals as shown in TABLE I.

TABLE I. HSV COLOR RANGE

Color	Hue	Saturation	Value
White	[0, 180]	[0, 80]	[50, 255]
Yellow	[17, 36]	[30, 255]	[0, 255]
Orange	[0, 16]	[50, 255]	[50, 255]
Red	[120, 177]	[50, 255]	[0, 255]
Green	[40, 94]	[50, 255]	[0, 255]
Blue	[95, 160]	[50, 255]	[0, 255]

Depending on the range of the chosen pixels, the initials of the identified colors will be saved in a vector, in a predefined order, as shown in Fig. 5 (a) and (c). Thus, 4 vectors of 18 elements each will be saved, one vector for each webcam.

5) Analysis of the elements: from the 4 vectors and saving in other 6 vectors of 9 elements each, for every face of the cube. The color vectors are mapped to the cube faces orientation. For example, if the centerpiece of the upper face is red, all the red stickers will be mapped as upper stickers.

Before saving, the common pieces from each camera are checked for color matching, as shown in Fig. 5 (b) and (d). The common pieces are 3, 5 and 7.

The cube can be oriented with any face, due to the sticker on the centerpieces is visible to the webcams. In Fig. 6 the green stickers are shown in the binary image, based on the HSV color range.

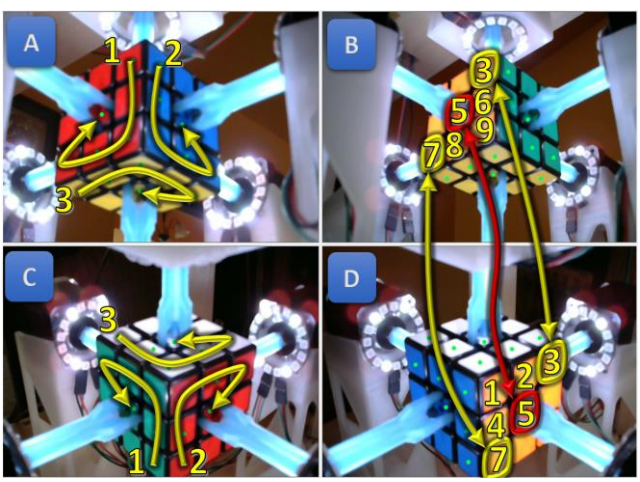

Fig. 5. Captured images from webcams: (a), (c) Front-down and front-up webcams; (b), (d) Back-down and back-up webcams.

Fig. 6. Binary image for green HSV color range.

III. SOLVING ALGORITHMS

A. Kociemba's Algorithm

The solution length of this algorithm is less or equal to 20 moves. A move is counted when a face is rotated by 90°, -90° or 180°. Faces of the cube are notated with U (up), D (down), R (right), L (left), F (front) and B (back).

Kociemba's algorithm is also called the two-phase algorithm. In the first phase of the algorithm, the cube will be brought to a specific state, and from there only certain moves will be applied. They belong to the subgroup G1 = {U, D, R2, L2, F2, B}. In the second phase, the cube will be completely solved, using only the moves from the G1 group, mentioned above.

After phase 1 has been applied, the cube will reach a state where the orientation of the edges and corners will not change, as can be seen in Fig.7 b). Even if the top face contains 2 colors (white and yellow), the orientation of the pieces will remain the same after applying the moves from group G1, although the pieces will be permuted. More information about Kociemba's algorithm can be found at [10].

The input data of the algorithm is a string that represents the initial state of the cube. The string consists of 54 characters, formed from the 6 faces of the cube, each with 9 stickers. After processing, the output is a string containing a maximum of 20 moves separated by a blank space.

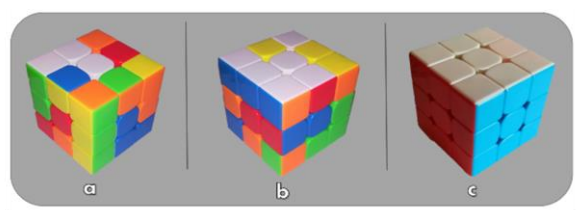

Fig. 7. Phases of the Kociemba's algorithm: (a) Initial state of the cube; (b) The cube after phase 1; (c) The cube after phase 2.

B. Blindfolded Method

The blindfolded method is used by people to solve the Rubik's cube knowing only the initial state of the cube and it was not implemented before in software. The input data is represented by a string as in the previous algorithm.

The method consists in moving each piece of the cube to the right place, without changing the initial state of the other pieces. First, will be solved the 12 edges of the cube, using the

Authorized licensed use limited to: University of Ghana. Downloaded on May 17,2021 at 10:10:30 UTC from IEEE Xplore. Restrictions apply.

M2 method. The 8 corners of the cube will be solved, using the Old Pochmann method.

The pieces of the cube are notated with letters from “A” to “X”, both for the edges and the corners, as in Fig. 8.

For permuting the pieces of the cube, a position called "buffer" will be used. Each piece will be brought to the buffer and from there, it will be moved to the corresponding position. The buffer is in a different location, for the edges and the corners.

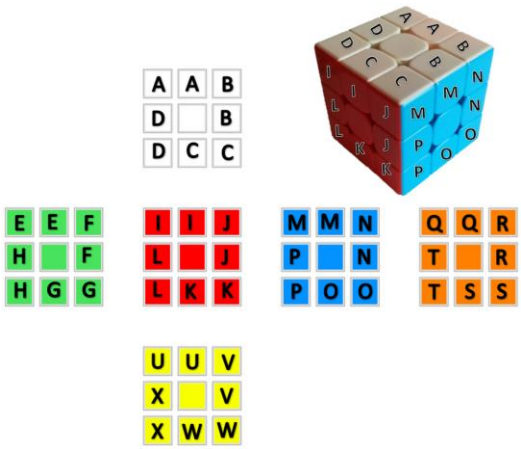

Fig. 8. Blindfolded notations.

Using the algorithms, M2 and Old Pochmann, two strings of letters are obtained. The pieces are brought to the buffer position, using specific moves, called "setup moves". These are different for each letter. A special algorithm is applied between the setup moves:

1) *Edges:* R2 L2;

2) *Corners:* R U’ R’ U’ R U R’ F’ R U R’ U’ R’ F R; noted with Y;

The mechanism of ARCAS can only rotate the outer layers of the cube. The M2 method consists in moving the middle layer of the cube by 180°, thus the “M2” move is changed to “R2 L2”. The outer layers of the cube will be moved by 180°.

The algorithm for M2 will rotate the left and right faces, by 180°, thus the position of the buffer will change. For the letters in the solution string that are in an even position, a different set of algorithms will be applied, compared to those in an odd position, as shown in Fig. 10.

Additionally, if the solution contains an odd number of letters, a parity algorithm will be applied between the two methods: D 'L2 D M2 D' L2 D. Otherwise, no additional algorithm will be applied. The state of the cube after each phase is shown in Fig. 9.

The solution consists in a string of letters from “A” to “X”, in a specific order. Each letter is replaced with the moves from Fig. 10 and sent to the microcontroller. The algorithms in green are special, due to the position of the pieces (C, I, S, W) that are in the middle layer of the cube. Also, the letters of the buffers are missing from the list: “A”, “E “and “R” for the Old Pochmann moves and M2 moves, “K” and “U”.

The actuators will rotate the parallel faces (Up – Down, Right – Left, Front – Back) simultaneously, reducing the number of moves and the time of solving.

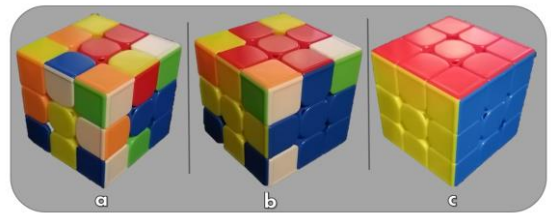

Fig. 9. Phases of the blindfolded method: (a) Initial state of the cube; (b) State of the cube after phase 1 (M2); (c) State of the cube after phase 2 (Old Pochmann).

Old Pochmann moves

```
(B) : R D' Y D R'
(C) : F Y F'
(D) : F R' Y R F'
(F) : F2 Y F2
(G) : D2 R Y R' D2
(H) : D2 Y D2
(I) : F' D Y D' F
(J) : F2 D Y D' F2
(K) : F D Y D' F'
(L) : D Y D'
(M) : R' Y R
(N) : R2 Y R2
(O) : R Y R'
(P) : Y
(Q) : R' F Y F' R
(S) : D' R Y R' D
(T) : D' Y D
(U) : F' Y F
(V) : D' F' Y F D
(W) : D2 F' Y F D2
(X) : D F' Y F D'
```

M2 (odd)

```
(A) : L2 R2
(B) : R' U R U' L2 R2 D R' D' R
(C) : U2 L R' F2 L R'
(D) : L U' L' U L2 R2 D' L D L'
(E) : B L' B' L2 R2 F L F'
(F) : B L2 B' L2 R2 F L2 F'
(G) : B L B' L2 R2 F L' F'
(H) : L' B L B' L2 R2 F L' F' L
(I) : D L R' F R2 F' L' R U R2 U' D' L2 R2
(J) : U R U' L2 R2 D R' D'
(L) : U' L' U L2 R2 D' L D
(M) : B' R B L2 R2 F' R' F
(N) : R B' R' B L2 R2 F' R F R'
(O) : B' R' B L2 R2 F' R F
(P) : B' R2 B L2 R2 F' R2 F
(Q) : B' R B U R2 U' L2 R2 D R2 D' F' R' F
(R) : U' L U L2 R2 D' L' D
(S) : L2 R2 U D R2 D' L R' B R2 B' L' R U'
(T) : U R' U' L2 R2 D R D'
(V) : U R2 U' L2 R2 D R2 D'
(W) : L' R B2 L' R D2
(X) : U' L2 U L2 R2 D' L2 D
```

M2 (even)

```
(A) : L2 R2
(B) : R' D R D' L2 R2 U R' U' R
(C) : D2 L R' B2 L R'
(D) : L D' L' D L2 R2 U' L U L'
(E) : F L' F' L2 R2 B L B'
(F) : F L2 F' L2 R2 B L2 B'
(G) : F L F' L2 R2 B L' B'
(H) : L' F L F' L2 R2 B L' B' L
(I) : U L R' B R2 B' L' R D R2 D' U' L2 R2
(J) : D R D' L2 R2 U R' U'
(L) : D' L' D L2 R2 U' L U
(M) : F' R F L2 R2 B' R' B
(N) : R F' R' F L2 R2 B' R B R'
(O) : F' R' F L2 R2 B' R B
(P) : F' R2 F L2 R2 B' R2 B
(Q) : F' R F D R2 D' L2 R2 U R2 U' B' R' B
(R) : D' L D L2 R2 U' L' U
(S) : L2 R2 D U R2 U' L R' F R2 F' L' R D'
(T) : D R' D' L2 R2 U R U'
(V) : D R2 D' L2 R2 U R2 U'
(W) : L' R F2 L' R U2
(X) : D' L2 D L2 R2 U' L2 U
```

Fig. 10. Blindfolded method algorithms.

Authorized licensed use limited to: University of Ghana. Downloaded on May 17,2021 at 10:10:30 UTC from IEEE Xplore. Restrictions apply.

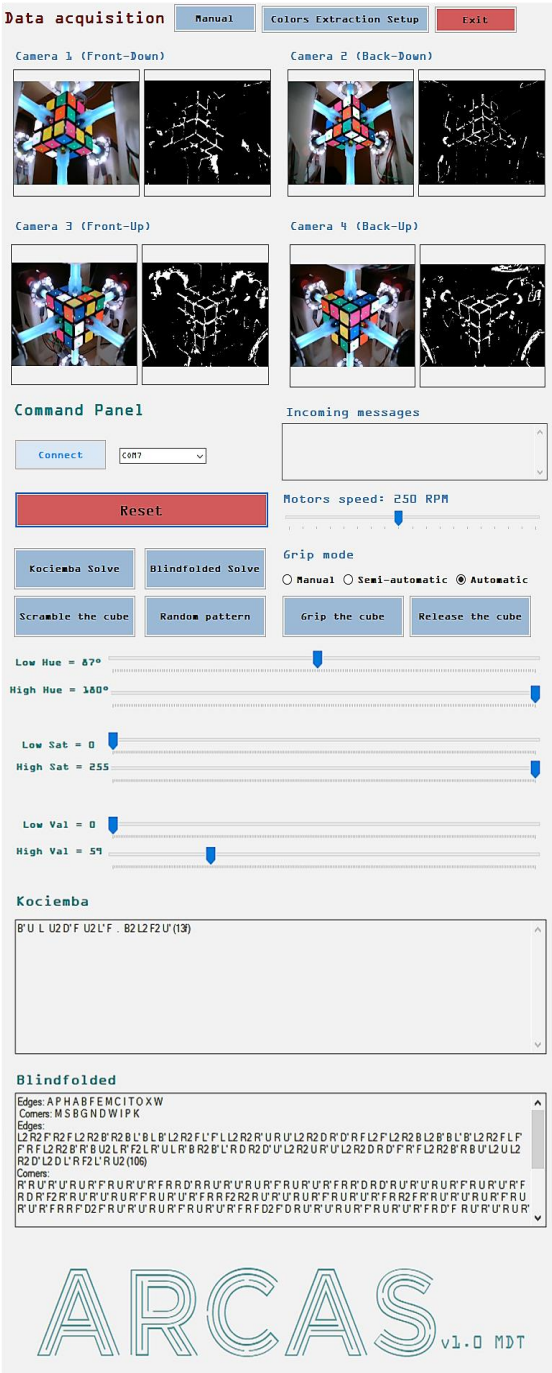

Fig. 11. Desktop application.

IV. Desktop Application

The application, implemented in C#, is responsible for image processing, finding the solutions for the cube and send them to the microcontroller. The user interface is used for choosing the solving method, scrambling the cube, generating a predefined random pattern, gripping, or releasing the cube and controlling the stepper motors speed, as shown in Fig. 11.

The communication with the Arduino due is made via the serial interface. Every button on the interface will send a command string that will be interpreted by the microcontroller. The strings will begin with "#", in the middle will be the command and will end with "#\n".

V. Conclusion

In this paper, a novel design for a Rubik's cube solving robot (ARCAS) was presented. The hardware is composed of 6 robotic arms actuated, 8 stepper motors, 6 webcams, LEDs, Arduino Due board, and PC. Based on the image processing technics the initial state of the cube is identified and a solution is proposed. On the ARCAS robot were implemented two methods for solving the Rubik's cube. The average solving time, from 10 solves, for Kobiemba's algorithm is 1.65 seconds (19.6 moves) and for the blindfolded method is 15.06 seconds (244.3 moves).

The average time of solving was reduced by rotating the parallel faces simultaneously. For Kocimeba's algorithm, from 1.75 to 1.65 seconds, and for blindfolded method, from 16.89 to 15.06 seconds.

In the future, can be added other solving methods and implement a web application, to use from other devices. The web application could be used for learning new methods or solving the cube.

References

[1] "God's Number is 20," [Online]. Available: http://www.cube20.org/. [Accessed 15 11 2020].

[2] D. Zeng, M. Li, J. Wang, Y. Hou, W. Lu and Z. Huang, "Overview of Rubik's Cube and Reflections on Its Application in Mechanism," Chinese Journal of Mechanical Engineering, vol. 31, 2018.

[3] E. D. Demaine, M. L. Demaine, S. C. Eisenstat, A. Lubiw and A. Winslow, "Algorithms for Solving Rubik's Cubes," Algorithms – ESA 2011, vol. 6942, pp. 689-700, 2011.

[4] El-Souranil, N. &. Hauke, S. &. Borschbach and Markus, "An Evolutionary Approach for Solving the Rubik's Cube Incorporating Exact Methods," Lecture Notes in Computer Science, vol. 6024, pp. 80-89, 2010.

[5] S. Saeidi, "Solving the Rubik's Cube Using Simulated Annealing and Genetic Algorithm," International Journal of Education and Management Engineering, vol. 8, 2018.

[6] S. P. Rohith, A. M. Sharif, S. Jayasankar and M. Harikrishnan, "Autonomous Rubik's Cube Solver Bot," International Journal of Scientific Research and Engineering Development, vol. 2, no. 3, pp. 146-151, 2019.

[7] D. Gilday, "MindCuber," [Online]. Available: http://mindcuber.com. [Accessed 20 November 2020].

[8] J. W. Daryl, "RukuRobot," [Online]. Available: https://rukurobot.weebly.com/. [Accessed 20 November 2020].

[9] "3dprint," [Online]. Available: https://3dprint.com/116817/3d-printed-rubiks-cube-robot/. [Accessed 20 November 2020].

[10] [Online]. Available: https://www.cs.brandeis.edu/~storer/JimPuzzles/RUBIK/Rubik3x3x3/READING/KociembaPage.pdf. [Accessed 21 November 2020].

Authorized licensed use limited to: University of Ghana. Downloaded on May 17,2021 at 10:10:30 UTC from IEEE Xplore. Restrictions apply.