Art Rendered using Rubik's Cubes: ARRC 256

Vasile Dan
Department of Automation, Technical University of Cluj-Napoca
Cluj-Napoca, Romania
vasidan.mdt@gmail.com,

Gabriel Harja
Department of Automation, Technical University of Cluj-Napoca
Cluj-Napoca, Romania
Gabriel.Harja@aut.utcluj.ro,

Ioan Nașcu
Department of Automation, Technical University of Cluj-Napoca
Cluj-Napoca, Romania
Ioan.Nascu@aut.utcluj.ro

***Abstract*— Mosaics are decorative images and patterns that can be created using small pieces of colored stone, ceramic, glass, or other materials. Creating a mosaic requires patience and time. This paper describes a robot that creates a mosaic using 256 Rubik's cubes. The overall process is controlled by a laptop and three Arduino Due boards. Image dithering algorithm is used to reduce the color palette, to the 6 colors of the Rubik's cube. A custom user interface is used to manually create an image or to upload an existing image. The solutions, for one face of every cube, are generated by a genetic algorithm and a permutation-based algorithm.**

Keywords-Rubik's cube, dithering, mosaic, genetic algorithm

I. INTRODUCTION

Any form of art requires hard work, patience, and time. Small pieces of colored stone, ceramic, or glass are used to create mosaics, that are lasting more than paintings [1]. Rubik's cube is a complex 3D puzzle that has different colors on each face. These are divided in 9 squares, that can be permuted and scrambled. The maximum number of moves required to solve a scrambled cube is 20 – known as God's Number [2].

Rubik's cube has determined people to create autonomous robots, to obtain shortest time for solving it. These robots can solve the cube under one second. Engineer Alber Beer from Germany built the Sub1 robot, that can solve the cube in 0.887 seconds [3]. Ben Katz built a robot that solved the cube in only 0.38 seconds [4]. In these two cases, the cubes were slightly modified for griping purpose. There are robots that can solve unmodified 3x3 cubes and cubes with more than three layers. MultiCuber 999 is a robot that can solve a cube with 9 layers [5]. In [6], the authors are using a Rubik's cube as a performance controller for music.

Many researchers have been involved in developing the algorithms that generate solutions for a scrambled Rubik's cube as a new research domain. In [7] the author has proposed an approach based on simulated annealing and genetic algorithm. In [8] the authors have discussed an algorithm that combines group theory and genetic algorithm. People have developed autonomous robots for creating mosaics from colored stone, ceramic, or glass. Robot Mosaic [9] and Colisium-SM [10] are two solution for creating a mosaic. In both cases, stepper motors and pneumatic systems are used for transporting the pieces.

One face of one Rubik's cube has 9 squares, that can be used to create a pattern. Using a predefined number of cubes, high resolution images can be created. This paper deals with the process of creating a mosaic using Rubik's cubes. The image for the mosaic can be chosen from a PC location or can be created by using the interface of the software, developed in C#. The genetic algorithm and the permutation-based algorithm will generate a solution for each cube. A specific pattern, from the mosaic, is rendered on the top face of the cube. A mechanical structure was design to manipulate the cubes and its faces, using stepper motors as actuators.

The paper is organized as follows: the next section presents an abstract of previous work. Section III gives an overall description of the hardware implementation and mosaic image processing. Section IV presents the solving algorithm, while last section summarizes the results and conclusions.

II. PREVIOUS WORK

ARCAS (Advanced Rubik's Cube Algorithmic Solver) main goal is to solve the Rubik's cube in the shortest possible time. Two solving methods are implemented: Kociemba's algorithm and blindfolded methods, M2 and Old Pochmann. Four webcams are used for color recognition of the cube pieces. For controlling the robot, a desktop application in C# was implemented. Also, besides solving the cube, ARCAS can randomize patterns on the cube faces [11].

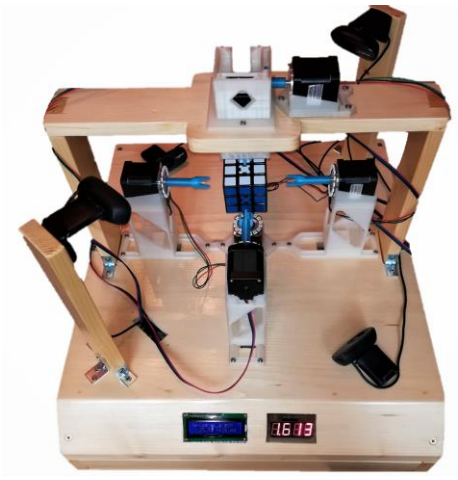

Figure 1. ARCAS structure.

The processing units of ARCAS are a PC and an Arduino Due board. To rotate the cube faces, 6 stepper motors were used. The arms of the robot can grip the cube using a rack and pinion mechanism and two additional stepper motors. The solving time is visible on the 4 digits of the 7 segments display, as shown in Figure 1.

III. ARRC Structure Description

A. *Hardware Implementation*

The robot consists of three main parts. Each of them will be controlled by a microcontroller, as shown in Figure 3.

1) The stack of cubes, that will be used to make the mosaic.

2) The robot arms, that will rotate the faces of the cube.

3) The frame, where the cubes will be placed to create the mosaic.

The 3 Arduino Due boards will communicate with each other using the RX and TX pins. The pin on which the serial data is received is RX, and the serial data is transmitted on the TX pin. An Arduino Due board that controls the robot arms, Figure 3. (2), is connected to the laptop. The solutions for each cube are sent, from the laptop to the board, in a string that will be decoded by the Arduino Due.

Hall effect proximity sensors are used to calibrate the stepper motors. This will allow the steppers that are carrying the cubes to be repositioned to the home position, with minimal errors. The Nema 17 stepper motors angle of rotation and speed are controlled by DRV8825 driver, depending on the moves that need to be made.

In Figure 2. is depicted the block diagram of the project.

The maximum size of the mosaic will be 48x48 pixels, for which 256 cubes will be used, 16 horizontally and 16 vertically, each side of the cube having 3 pixels. The cubes will be stored in a frame of 8x8x4, as shown in Figure 3. (1).

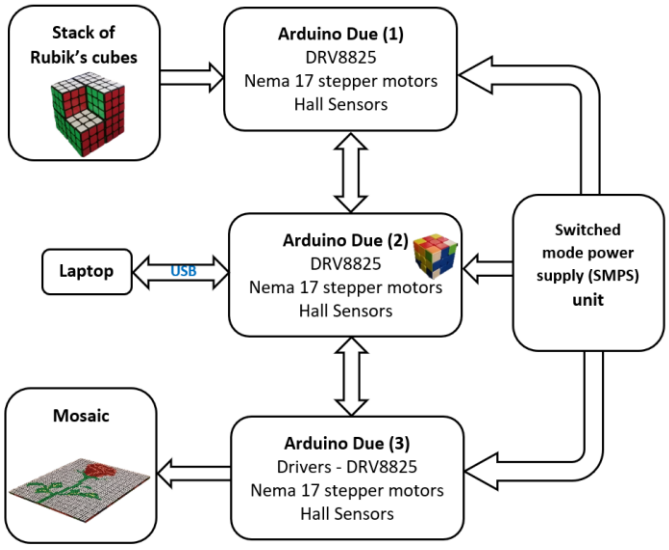

Figure 2. Block diagram of the project.

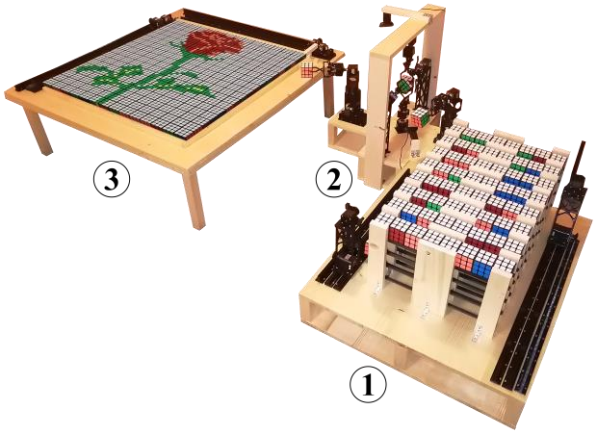

Figure 3. ARRC 256 Structure: (1) The stack of cubes, (2) The robot arms, (3) The frame for mosaic.

The extraction of the cubes from the frames will be done with the help of a mechanism, based on stepper motors. The cubes will be transported to the central part of the robot, where they will be taken over by the 6 arms, to rotate the faces.

In the central part of the robot, responsible for solving the cube, Figure 3. (2), the faces of the cube will be rotated to create a model on the top face accordingly to the model of the mosaic. The modified cubes will be placed on a frame, Figure 3. (3), where the mosaic will be made. Two mechanisms will slide the cubes on the horizontal axes. A group of cubes, up to 16, will be slid to create the image, layer by layer.

The final structure of the robot, can be seen in Figure 3.

B. *Mosaic Image Processing*

To create a mosaic, a software graphical user interface has been implemented to upload an existing image or to draw an image. The window where the image will be drawn is represented by a matrix of 48x48 pixels, as in Figure 4.

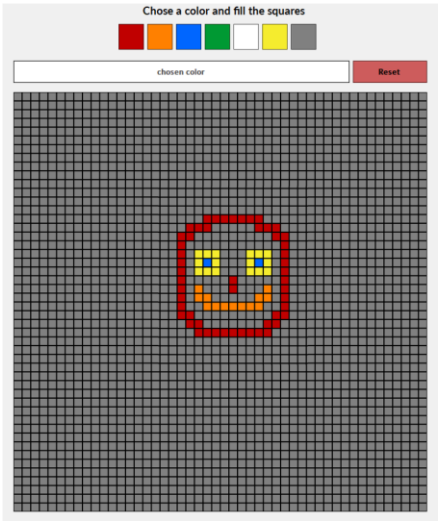

Figure 4. ARC 256 Drawing Interface.

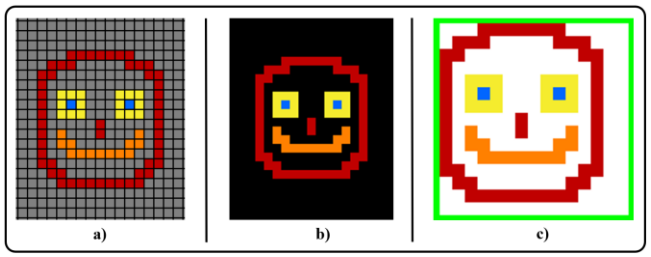

Figure 5. Image conversion: (a) Initial drawn image; (b) Unused pixels conversions; (c) Final image.

The color of the pixel can be modified by selecting one of the 6 available colors and then pressing left click. Further, multiple pixels can be changed by pressing and holding the left click and hovering the mouse over them. The pixels will change according to the selected color. The drawn image will be prepared by converting the unused gray pixels, into black pixels. There will be a high contrast to the cube colors and the image will be extracted by generating the bounding box. Thus, it will be possible to create a mosaic smaller than 48 x 48 pixels.

After the bounding box is found, the length and the width of the image are checked, so that the number of pixels is a multiple of 3. Otherwise, extra layers (rows or columns) of white pixels will be added. The gray pixels inside the bounding box will also be modified into white pixels. In Figure 5. a layer of white pixels has been added on the rows, at the bottom, and two layers of white pixels on the columns, on the right side of the image.

The mosaic can be created using an existing image, that can be uploaded and then converted, according to the color palette of the cube. The uploaded image is resized to 300x300 pixels. To preserve the aspect ratio of the image, the longest edge will be resized to 300 pixels, and the other edge gaps will be filled with black pixels. The entire image or a region of interest (ROI) can be used, as shown in Figure 6. The cropped image is converted so that the maximum size is 48 x 48 pixels. The color palette can have between 2 and 6 colors. To narrow the color palette, a filtering process, called "dithering", was applied [12]. This process consists of reading the RGB (red, green, blue) values of each pixel and modifying them with the closest values from the color palette. The value of the pixels is modified according to the threshold distance from the selected colors, as in Figure 7.

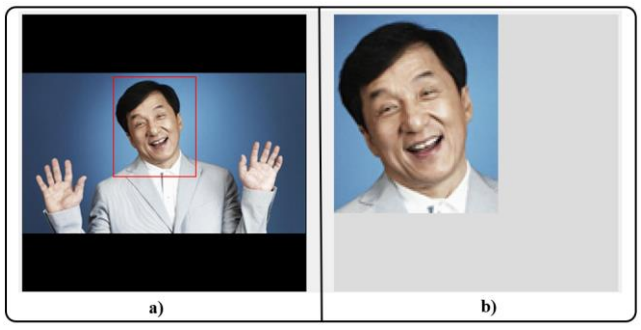

Figure 6. (a) Uploaded image; (b) Cropped image.

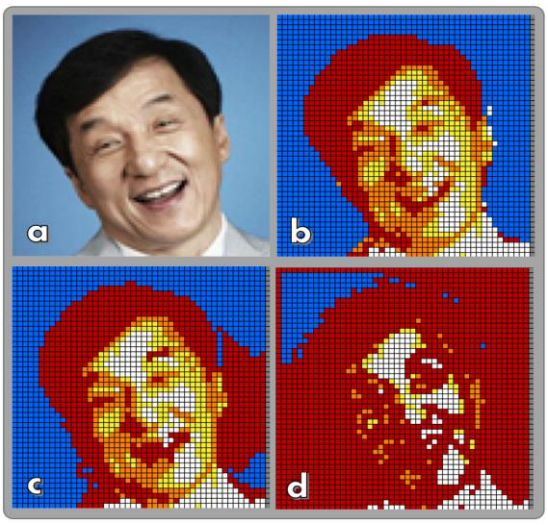

Figure 7. Dithering: (a) Uploaded image; (b) Filtered image – threshold 270; (c) Filtered image – threshold 230; (d) Filtered image – threshold 200.

The threshold can be modified from the user interface using a slider or typing the desired value in a text box. The filtered image is updated instantly, thus the best value of the threshold can be chosen. The converted image can be adjusted manually by changing the color of the pixels.

IV. One Face Algorithm

The top face of the cube represents 9 pixels of the image. The position of other pieces of the cube is irrelevant. The solutions, for each top face of the cubes, are generated by the one face algorithm.

To identify each piece, they are marked with letters from "A" to "X". The corners are marked in capital letters and the edges, in lower case letters, from "a" to "x". The centers are not marked, as shown in Figure 8. The faces of the cube are notated with letters: U (Up), D (Down), F (Front), B (Back), L (Left) and R (Right). The faces can be rotated as follow:

- *90°, clockwise rotation (ex. F)*
- *-90°, counterclockwise rotation (ex. F')*
- *180°, two 90° rotations (ex. F2)*

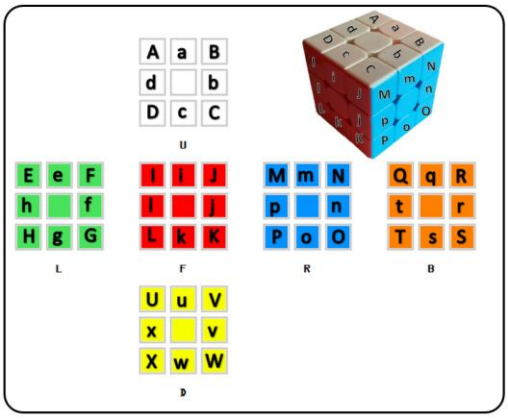

Figure 8. Cube notations.

The solution of the algorithm will be a string of letters, depending on the top face pattern of the cube. For example: L' D2 L, will permute the front-bottom-right corner to the top-back-left position. The other pieces of the top face are in the same position, before and after applying the three moves, as shown in Figure 9.

To create a model on one face, the pieces of the cube will be permuted (the center is fixed). Each of these pieces can have one of the 6 colors of the cube. The position of the pieces differs and for a certain color, there will be a different set of moves. The faces of the cube have 4 corners and 4 edges. For each sticker of one face, there are maximum 4 different set of moves, with a certain length. These sets of moves are predefined in the code. Thus, the order of going through the 8 stickers, determines the length of the final solution.

The one face algorithm is a combination of a genetic algorithm and an algorithm based on permutation, which will be presented in the following sections.

A. Genetic Algorithm

The stickers of the top face, are noted from 0 to 7 starting from the top left corner, as shown in Figure 10. The chromosome consists of 8 genes, the stickers on the top of the cube, each represented in 3 bits. In total, the chromosome will have 24 bits. The genotype consists of the numbers 0, 1, 2, 3, 4, 5, 6 and 7.

When the genetic algorithm is initialized, several different chromosomes are created. The mapping is realized by assigning a set of moves to the 8 genes. For each gene, there are maximum 4 variants. The variant with the fewest moves is chosen. If there are more variants with the same number of minimum moves, a random variant is chosen, as show in Figure 11.

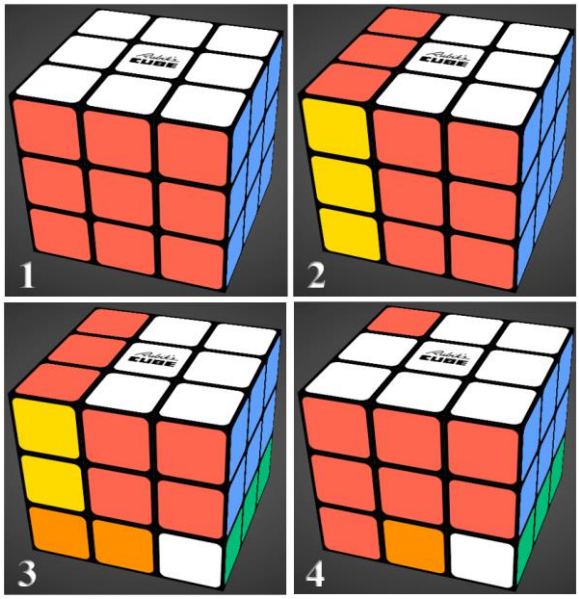

Figure 9. Corner permutation: 1. Initial cube; 2. Cube after the L' move; 3. Cube after the D2 move; 4 Cube after the L move.

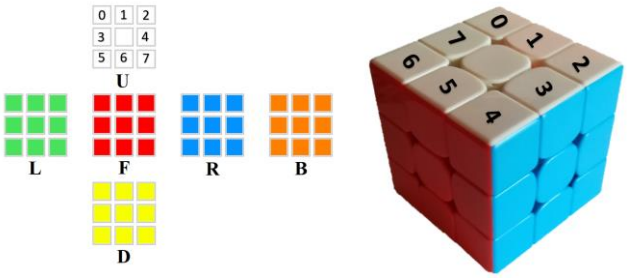

Figure 10. Genetic algorithm top face notations.

In the example above, the chosen pattern is red-blue-red-white-blue-white-white-white (genes from 0 through 7 and the center). The last set of variants is permuting the front-top-right (red-white-blue) corner to the top-back-right corner (gene 2), using the following moves: (F B') D (B F'). While permuting the red-white-blue corner, in his place came the blue-orange-yellow corner. In this example, for gene 4 was needed a blue color and no further moves are required. If the gene 4 should have been yellow, a new set of variants would have been generated. The algorithm is excluding previous genes, 0 and 1. For that reason, there are only 3 variants in the last set.

After mapping, the performance of the individuals is calculated as follows:

$$S_{c_i} = \sum_{k=0}^{7} I[k] \tag{1.1}$$

Where, S_{c_i} is the sum of the chromosome, I represents the individual, and k represents the gene, the set of moves for permuting the stickers. The performance function, which calculates the minimum length of the chromosome, is as follows:

$$F = \mathrm{Min}(S_{c_i}) \tag{1.2}$$

The selection of individuals is made based on the performance function. Elitism selection, where 2 or 3 elites are chosen: individuals with the shortest length, move on to the next stage.

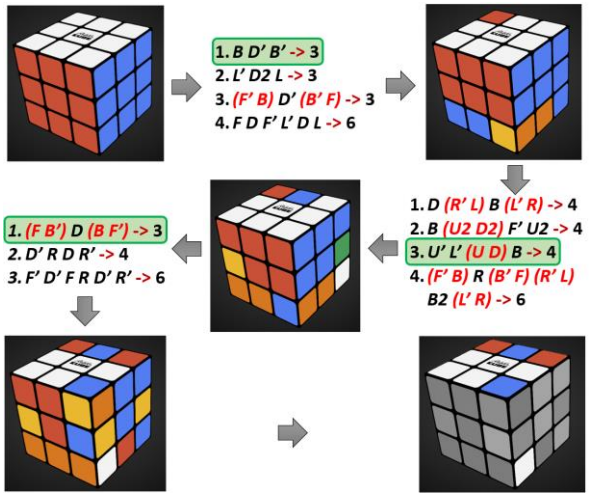

Figure 11. Genetic Algorithm Mapping Example.

Roulette wheel selection: the probability of the individuals being chosen to create offspring is directly proportional to the performance given by the F function.

The genetic operators applied to create offspring are:

1) Crossover, where two parents are divided, and a viable solution is generated. In this case, no duplicate genes are allowed.

2) Mutation, where the chromosome changes one or more values. In this case, two values are swapped.

In Figure 12. , the genetic algorithm has generated 10 solution. The first generation has 38 moves towards the 11 generation, that has 33 moves.

B. Permutation-based Algorithm

The steps of the permutation-based algorithm are as follows:

1) Calculating the solution for the edges: the 4 edges of the top face are selected, all 24 solutions (4!) are calculated and the solution with the fewest moves is chosen.

2) Calculating the solution for the corners: the procedure is the same as at the edges.

3) The solutions from step 1 and 2 are merged.

The difference from the genetic algorithm is, the edges are solved first and then the corners. This approach was chosen because, the length of the algorithm for solving the edges varies between one move and 4 moves. The disadvantage is that the corners of the upper face will also be permuted. Thus, with this approach it can happen that certain corners are exchanged correctly.

The one face algorithm consists of the two algorithms: the genetic algorithm, presented above, and the permutation-based algorithm. Each of these algorithms are generating one solution. The solution with the fewest moves is chosen and used to create the pattern of the face.

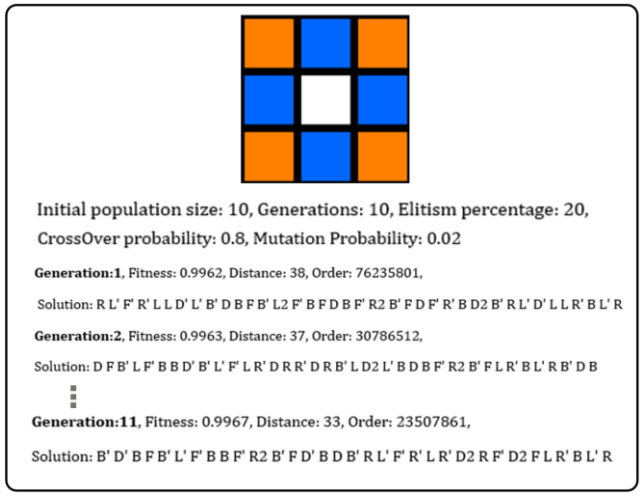

Figure 12. Genetic algorithm solution for a specific pattern.

The genetic algorithm offers the shortest solution when only the edges need to be permuted. The permutation-based algorithm offers the optimal solution when both corners and edges need to be permuted. Both algorithms will generate an optimal solution, when only corners need to be permuted.

V. CONCLUSIONS

In this project, the aim was to make a mosaic using several Rubik's cubes. The mosaic image can be drawn by the user or an existing image can be uploaded. A genetic algorithm and a permutation-based algorithm were used to generate the solution for the mosaic.

The physical assembly was realized successfully. The components used to create the robot are: 26 stepper motors, 6 hall effect sensors, 3 Arduino Due boards, a power supply unit and a laptop. ARRC 256 can be used with the desktop application, developed in C#.

The algorithms implemented in the application are developed independently of the physical assembly, so that the application can be used without connecting it to the robot. Thus, the algorithms implemented for this work can be incorporated in other applications such as a webpage or a mobile application.

REFERENCES

[1] F. Stanco, S. Battiato and G. Gallo, Digital Imaging for Cultural Heritage Preservation, Boca Raton: CRC Press, 2011.

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

[3] M. Matisons, "3Dprint," 17 Februarie 2016. [Online]. Available: https://3dprint.com/120112/sub1-robot-rubiks-cube/7/. [Accessed 03 December 2020].

[4] [Online]. Available: http://build-its-inprogress.blogspot.com/2018/03/the-rubiks-contraption.html. [Accessed 03 December 2020].

[5] A. Liszewski, "Gizmodo," 22 Decembrie 2016. [Online]. Available: https://gizmodo.com/a-lego-contraption-that-solves-giant-9x9x9-rubiks-cubes-1790397289. [Accessed 03 December 2020].

[6] R. Polfreman and B. Oliver, "Rubik's Cube, Music's Cube," pp. 493-494, 2017

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

[8] A. Darbandi and S. A. Mirroshandel, "A Novel Rubik's Cube Problem Solver by Combining Group Theory and Genetic Algorithm," SN Computer Science, vol. 1, no. 1, 2019.

[9] "Robot Mosaic," [Online]. Available: https://robot-mosaic.com/en/. [Accessed 04 December 2020].

[10] [Online]. Available: https://grabcad.com/library/mosaic-tile-layer-robot-1. [Accessed 02 December 2020].

[11] V. Dan, G. Harja and I. Nascu, "Advanced Rubik's Cube Algorithmic Solver," unpublished.

[12] L. Velho, A. Frery and J. Gomes, "Image Processing for Computer Graphics and Vision," Springer, January 2008.

AUTHORS' BACKGROUND

Your Name	Title*	Research Field	Personal website
Vasile Dan	Phd candidate	Embedded systems, Robotics, Control Engineering	
Gabriel Harja	Assistant Professor	Embedded systems, Modelling and Control Engineering, Wastewater Treatment	
Ioan Nascu	Professor	Adaptive Control, Self-Tunning, Model Based Predictive Control.	