An educational robotic approach for improvements of learning in technology courses

Uma abordagem de robótica educacional para melhoria da aprendizagem em cursos de tecnologia

DOI:10.34117/bjdv7n1-161

Recebimento dos originais: 10/12/2020
Aceitação para publicação: 09/01/2021

João Moreno Vilas Boas
Pós-Doutorado em Engenharia Elétrica e da Computação
Instituto Federal do Rio Grande do Norte (IFRN)
Endereço completo: Rua Vega Nº 172, Loteamento Caminho do Sol. Parque das Nações – Parnamirim/RN
E-mail: joao.vilasboas@ifrn.edu.br

Plácido Antonio de Souza Neto
Pós-Doutorado em Ciência da Computação
Instituto Federal do Rio Grande do Norte (IFRN)
Av. Pref. Milton Dantas de Medeiros, Nº 316, Cond. Porto Rico, casa 26. Parque das Nações – Parnamirim-RN
E-mail: placido.neto@ifrn.edu.br

Israel Philipe Assunção Medeiros
Graduando em Tecnologia em Análise e Desenvolvimento de Sistemas
Instituto Federal do Rio Grande do Norte (IFRN)
Rua rio Tietê, Nº 7707. Pitimbu, Natal/RN
E-mail: israelphilipeassuncao@gmail.com

Vinícius Pereira da Silva Oliveira
Engenharia Mecânica
Universidade Estadual de Campinas (UNICAMP)
Endereço: Rua Felisberto Brolezze, Nº 98. Jardim Independência - Campinas/SP
E-mail: v265311@dac.unicamp.br

Eugênio Pacelly Brandão De Araújo
Mestre em ciências tecnologia e inovação
Universidade Federal do Rio Grande do Norte (UFRN)
Endereço: Rua Alexandre Câmara, Nº1915. Capim Macio, Natal/RN
E-mail: pacellyb@hotmail.com

ABSTRACT
The Rubik's cube is one of the most popular puzzles in the world. It is estimated that one in seven people have already played with the cube. There are several ways to solve the cube problem and some people solve without a technique help, however, this is a laborious and tiring practice. The existing methods range from the simplest to the most complex, and it influences the time that will be spent in the solution. A robotic solution can implement one or more of these methods to solve the Rubik's cube in an automatic way. Considering this, this work proposes a low-cost micro controlled system to solve Rubik's cube. This prototype is intended to be used in classrooms for the teaching-learning process improvement, and consequently also improve the students' performance in disciplines of logic and algorithms.

Keywords: Rubik's cube, Educational robotics, Arduino, Teaching-learning, Active Learning.

RESUMO
O cubo de Rubik é um dos quebra-cabeças mais populares do mundo. Estima-se que uma em cada sete pessoas já tenha brincado com o cubo. Existem várias maneiras de resolver o problema do cubo e algumas pessoas resolvem sem auxílio de técnicas, no entanto, essa é uma prática trabalhosa e cansativa. Os métodos existentes variam do mais simples ao mais complexo e influencia o tempo que será gasto na solução. Uma solução robótica pode implementar um ou mais desses métodos para resolver o cubo de Rubik de maneira automática. Considerando isso, este trabalho propõe um sistema microcontrolado de baixo custo para resolver o cubo de Rubik. Este protótipo destina-se a ser usado nas salas de aula para melhorar o processo de ensino-aprendizagem e, consequentemente, o desempenho em disciplinas de lógica e algoritmos.

Palavras-chave: Cubo de Rubik, Robótica Educacional, Arduino, Ensino-aprendizagem, Metodologias Ativas.

1 INTRODUCTION

In the last decade with the increasing of technological advance there is a need for a convergence between school curriculum and technological innovations. It happens because the students born in the 21st century find themselves immersed in an extremely technical world in all society domains. Additionally, the teaching-learning process of logic and basic programming in Brazil's institutions is still a huge challenge, either in technical or higher levels of education. This worrying truth, allied with the perception of the need to use innovation technologies in the school curriculum, creates an interesting scenario for the application of new approaches and pedagogical practices, especially regarding the use of robotics educational as a form of improvement in the teaching-learning process.

According to Soares et al. (2015), the school is the environment for the development of technology which contributes to the integral student formation, fostering

critical thinking, stimulating the manipulation of elements in practical activities, instigating creativity and awakening the students the desire to produce knowledge. Vilas Boas et al. (2016) affirm the learning difficulty mentioned above has been observed in the Instituto Federal do Rio Grande do Norte (IFRN) in a very expressive diagnosis in the discipline of Fundamentals of Logic, considering courses of the technical degree. From the data analyzed, it was verified that the 24% of the students failed in the Fundamentals of Logic discipline between 2012 and 2016.

Active methodology is an interesting approach to reduce the learning deficit in disciplines like Fundamentals of Logic. In this approach, the students are the most responsible for the learning process. This is the main difference compared with practices used in traditional educational institutions, where the students only follow the material taught by the professor.

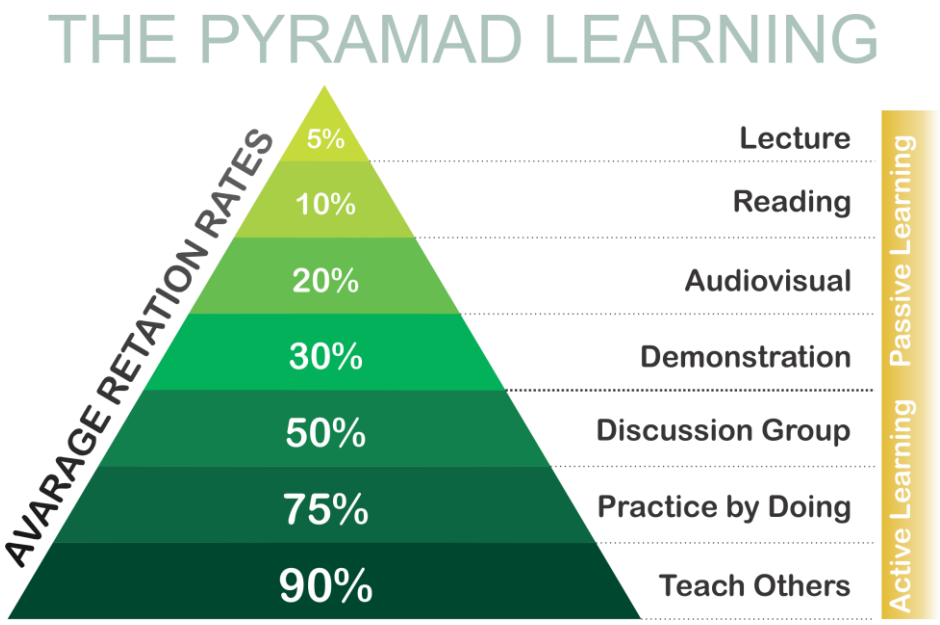
William Glasser, an American psychiatrist, explains the degree of learning in his research entitled "*How We Learn*" by using the Pyramad Learning (Figure 1). The proposal presents that the process of knowledge assimilation occurs more easily by using certain methodologies. His research also shows that when the student only listens the content in expository classes, she retains only 20%. On the other hand, in a more active learning, when the student performs some practice, this rate increases to 80% (ORSI et al., 2017; PINTO, 2020).

In this context, educational robotics is a reasonable option as active methodology which can be used as a pedagogical resource to improve the learning process in school environment, especially in disciplines from technological area. Its use started in 80's with Seymour Papert, who used "*turtle*" robots to teach the LOGO programming language to young students. The researcher observed an increase in the creativity and abstraction of the students involved in the process (ANWAR et al., 2019).

In 2013, Aroca et al. inserted a low-cost robotic kit in technological subjects in their practical activities and then, through an opinion poll, found that 83% of students agreed with the use of robots as an instrument to improve the learning process.

In 2019, Rohith et al. built an autonomous robot to solve the Rubik's Cube, using an Arduino Mega 2560. The robotic agent developed was able to recognize colors to detect the initial orientation of the cube and then, proceed the algorithm solution by using the layer method.

Figure 1: Puzzles: The pyramid learning.



Thus, the objective of this work is creating a low cost and articulated robotic mechanism, able to solve the Rubik's cube problem through the implementation of different algorithms. From this, we expect to overcome the scenario presented with an increase of interest and performance in disciplines Logic and Basic Programming using this prototype in classroom. There is also the possibility of collective construction of the solution by groups of students involved.

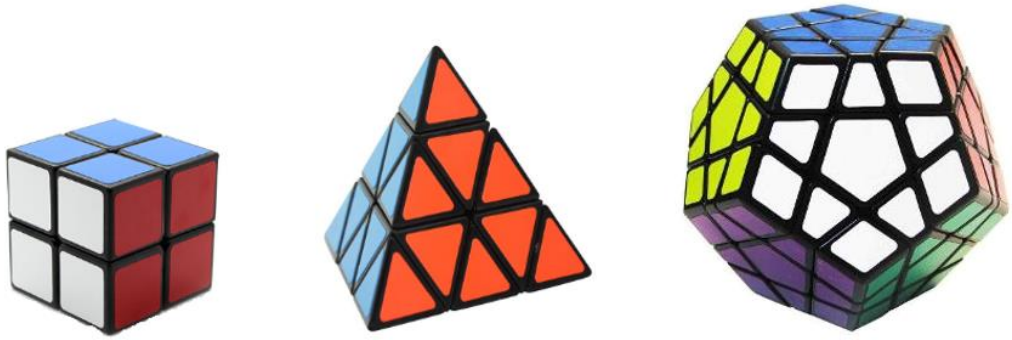
2 THE RUBIK'S CUBE

The Rubik's cube, one of the most popular puzzles in the world, came out in 1974. It is considered one of the most successful puzzles in history, mainly because it has more than 350 million units sold worldwide, making his inventor, the Hungarian Erno Rubik, become one of the richest men in his country. Erno Rubik was born in the city of Budapest, and while he was a university professor of interior design, he decided to create an object that would help him to perfectly illustrate the concept of the third dimension. From the Erno's research by the expression and the increasing intensity of some thoughts, the interest for the cube was born.

Still in the 70's, Erno was awarded for his invention, a fact that further stimulated people's interest in the toy. With the demand increasing, the puzzle began to be distributed throughout America, with a completely redesigned version, with easier faces to handle and more vivid colors. So it beings referred as Rubik's cube and nowadays, various

enthusiasts continue to produce cube-based puzzles such as *Megaminx* (dodecahedron), *Pyraminx* (tetrahedron) and 2x2x2 cube (a variation of the traditional cube 3x3x3), like as shown Figure 2.

Figure 2: Puzzles: Cube 2x2x2, *Pyramin*x and *Megaminx*.



2.1 GOD'S NUMBER

God's number is a reference to the minimum number of moves to solve any configuration of a Rubik's cube. Solving it at random can be a quite time-consuming work, where there are exactly 43.252.003.274.489.856.000 possible combinations.

According to the site Cube 20, in July 1981, the mathematician Morwen Thistlethwaite implemented a complex and very difficult algorithm to be memorized. Unlike traditional algorithms such as layering, this method does not put piece by piece in its correct position but works so that all pieces are placed in their correct positions at once. From its algorithm, Thistlethwaite has been able to prove that a total of 52 moves are needed to solve the puzzle. In December 1990, the number was reduced to 42 movements. This fact was proved by Hans Kloosterman. Two years later, Michael Reid managed to slow down the minimum number of moves, this time to 39. In the same year, Dik Winter reduced that number to 37 moves. Few years later, in 1995, Michael Reid showed that God's number was 29. It took eleven years for that number drop to 27, a fact proven by Silviu Radu. It is interesting to note that from 1981 to 2006 this number decreased by 25 movements, practically half the value that was initially proven. Still according to Cube 20, in May 2007, Dan Kunkle and Gene Cooperman showed that in fact the minimum quantity was 26 moves. The following year, Tomas Rokicki reached the number of 25 moves. After, Rokicki and John Welborn reached 23 and then in 22 moves. In 2010, Rokicki, Herbert Kociemba, Morley Davidson and John Dethridge concluded that God's number is exactly 20. So far no one has been able to prove less than that.

2.2 SOLUTION METHODS

As previously stated, the Rubik's cube has more than 40 quintiles of possible combinations, and since the 1980s this three-dimensional puzzle has challenged many people to find a solution. There is a myriad of ways to solve the cube. Some people solve without a techniques help. However, this is a laborious practice and much more difficult than just following the steps that are given in the methods. Once learning an algorithm, it is essential to repeat the movements and understand the purpose of each step until it becomes natural and intuitive. The methods range from the simplest to the most complex, and it influences the time that will be spent in the solution. Some sequences are derived from other methods or even are combinations of several of them.

There are several tutorials on worldwide websites, the most popular is the Speed Cube1, developed by Renan Cerpe, and the site belongs to one of the WCA (World Cube Association) delegates, Rafael Cinoto. The French site Francocube 2 is one of the most outstanding worldwide, for being complete and presenting tutorials for different types of different puzzles. Following, we will be expose some of the most used methods to solve the Rubik's cube (McNaughton, 1990).

2.2.1 Layer Method

The layer method is one of the most used by those who are starting to learn how to solve the cube. It is a very simple technique that does not present great complexity. It is divided into seven steps that aims to solve the cube layer by layer. This method was proposed in the 80's and performs an average of 100 movements. The algorithm will be explained in detail later.

2.2.2 Intermediate Method

The intermediate method is an extension of the traditional layer method. The difference is that it has more advanced movements to solve the last two layers. This reduces the number of movements generated and the time spent to solve the cube, a total of 30 to 40 seconds less than the previous method.

2.2.3 Fridrich Method

The Fridrich method, developed by Jessica Fridrich, is one of the most used by Rubik's cube competitors worldwide. Nowadays it is considered one of the fastest methods to solve the cube. However, it is also one of the most difficult because it uses

several algorithms. The difference to the intermediate method is that it has even more algorithms that serve as shortcuts to the traditional algorithm, which further reduces the time and the number of moves.

2.2.4 Petrus Method

The Lars Petrus method is also an advanced method. It was developed in the 80's and it was based on the resolution by block construction, where the first two layers are solved in a totally intuitive way. It is the second most popular advanced method, after Fridrich. This method is also partially used in other methods. In some situations, this method may generate fewer moves than Fridrich, but sometimes, especially for beginners, it can be difficult due to the large number of intuitive movements.

3 MECHANISM DESCRIPTION

Following, we will describe the hardware and software solution developed for the proposed system.

3.1 HARDWARE

The robot's hardware structure consists of parts made by a 3D printer, which are: (i) a base for the cube, (ii) a horizontal movement arm and an (iii) adjustment claw on the cube. These parts are arranged in an area measuring 49.5 cm in length and 15 cm in width (Figures 3 and 4). The cube used for the design is the *Shengshou Stickerless, Rainbow Stickerless* 3x3 model. The robot operation consists of the mechanical arm pushing the cube to change to the next face. To exchange each face, the arm holds the cube while the base rotates 90, 180 or 360 degrees. Then the adjusting jaw flushes the cube.

Figure 3: Articulated mechanism with servo motors and control plates.

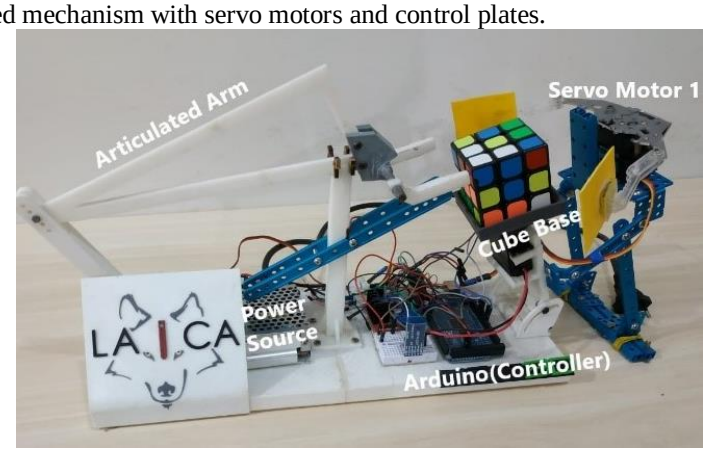
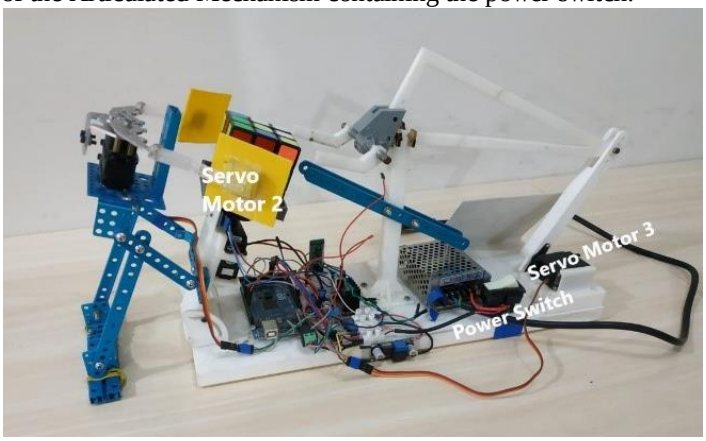


Figure 4: Back side of the Articulated Mechanism containing the power switch.

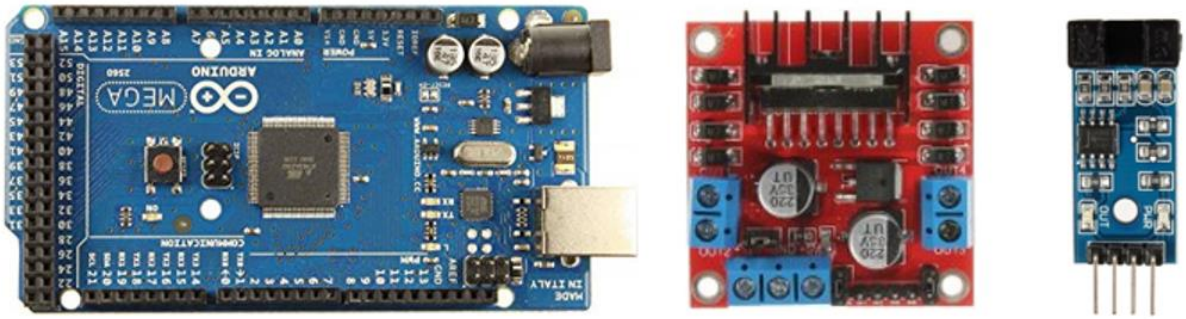


The robot arm was built with the 3D printer and has 29.5 cm. The main functions aim to push the cube in order to change the face and hold it when it is necessary to change the cube face. The mechanical arm moves through a servo motor TowerPro, MG995 model. We also use the same type of servo to move the base of the cube. The robot base was built into the 3D printer and has 6.7 cm width and 6.7 cm height and is responsible for rotating the hub at positions 90, 180 and 360 degrees. To perform these movements, coupled to the base, there is an optical encoder of 5 mm and a rotating disk that sends data to the encoder.

To give stability to the robot base, there is a 23.5 cm height metal frame made with pieces of the educational robotic platform Makeblock. At the top of this structure there is a TowerPro servo motor coupled to a modified mechanical Claw Robot Mechanical Claw H3. The claw is responsible for adjusting the sides of the cube when its parts change places. All servo motors are connected to a source Beehive Model S-30- 12 of 5 amps and 12 volts. The power supply also powers a H-bridge L298N, its function is basically to control the speed and direction of the motors. The H bridge is connected to a voltage regulator LM2596 model, responsible for regulating the reactive power division between the devices. All these equipments are connected to an Arduino Mega 2560. Figure 5 shows some of the components used in the project construction.

Figure 5: Some parts of the proposed system. (a) Arduino Mega 2560 R3, (b) Optical Encoder, (c) L298N H Bridge.

(a) (b) (c)



3.2 SOFTWARE

The software consists of two algorithms: one for the cube state acquisition through computer vision and another for controlling the robot and solving the cube using the layer method.

3.2.1 Computer Vision

In order to acquire the state of each face in the cube, it is necessary the algorithm is able to identify the colors of each cube face with some precision. For this purpose, the Open Source Computational Vision library (OpenCV). OpenCV is an open source library with optimized algorithms for computer vision and machine learning. Along with this library, the code was also developed with the help of Visual Studio 2019 IDE using the C ++ programming language.

To analyze the colors on the cube faces, an image is required. OpenCV represents it by a *Mat* container. This container stores the information of each pixel presented in the image as a matrix or set of matrices. Each position of a matrix represents an intensity value of a given pixel at this position. In the most general case, this container stores an RGB matrix, but can also receive matrices with other notations and color spaces for optimization and analysis. In the scope of this work an 8-bit RGB matrix (intensity values from 0 to 255) is used.

To identify the corresponding color, the program reads 9 pixels corresponding to each square of the cube face. These pixels are processed and then the color is defined. Each color is the result of a scalar function that receives a vector with the color components as input:

$$\vec{v} = (r, g, b) \quad (1)$$

The color space can then be considered as a 3-dimensional Euclidean space with internal product, in which the red, green and blue color components are linearly independent vectors (Euclidean internal product between vectors equal to zero) and therefore constitute an orthogonal base in the color space.

As a way to reduce the color space and make it independent of intensity, the vectors of this Euclidean space can be normalized and then each color vector will belong to a spherical surface of radius equal to one, or also called the unit sphere (equation 4), in other words:

$$\left\|\vec{v}\right\| = \sqrt{<\vec{v}, \vec{v}>} \quad (2.1)$$

$$\left\|\vec{v}\right\| = \sqrt{r^2 + g^2 + b^2} \quad (2.2)$$

$$\vec{u} = \frac{\vec{v}}{\left\|\vec{v}\right\|} \quad (3)$$

Where every $\underline{u}$ is a unit vector that satisfies the equation:

$$1 = r^2 + g^2 + b^2 \quad (4)$$

The normalized components are compared within an empirically determined range with the help of another program and a spreadsheet to measure the range where the color can be. These delimit the area of the unit sphere where the color value can be, if the color components are not in that area, the gray color is returned. Below is an example of code that represents this comparison (Table 1).

Lastly the algorithm shows the reading result on the screen and, pressing the enter key, the algorithm outputs a string with the corresponding color numbers of each square of the cube face.

Table 1: Decision structure for color identification.

Line	Code
01	if (b >= 0.60 && b <= 0.69 && g >= 0.51 && g <= 0.60 && r >= 0.49 && r <= 0.62) {
02	color[i] = white;
03	} else if (b >= 0.00 && b <= 0.28 && g >= 0.67 && g <= 0.85 && r >= 0.52 && r <= 0.74) {
04	color[i] = green;
05	} else if (b >= 0.00 && b <= 0.24 && g >= 0.49 && g <= 0.65 && r >= 0.74 && r <= 0.87) {
06	color[i] = yellow;
07	} else if (b >= 0.08 && b <= 0.36 && g >= 0.05 && g <= 0.19 && r >= 0.92 && r <= 1.00) {
08	color[i] = red;
09	} else if (b >= 0.00 && b <= 0.27 && g >= 0.08 && g <= 0.35 && r >= 0.90 && r <= 1.00) {
10	color[i] = orange;
11	} else if (b >= 0.94 && b <= 0.98 && g >= 0.24 && g <= 0.33 && r >= 0.00 && r <= 0.02) {
12	color[i] = blue;
13	} else {
14	color[i] = gray;
15	}

3.2.2 Solving Strategy

The basic method, also known as layer method, is simple and widely used by those interested in learning how to solve the rubik's cube. As its name suggests, this solution consists of solving the cube layer by layer, using only seven steps.

- Step 1: make the cross in the first face;
- Step 2: position the corner pieces of the first layer;
- Step 3: position the edge pieces of the second layer;
- Step 4: make the cross on the top face;
- Step 5: review the top corner pieces;
- Step 6: permute the corner pieces;
- Step 7: permute the edge pieces.

There are other advanced methods also based on the concept of layers. However, unlike the traditional method, they have more extensive algorithms and generate less movement.

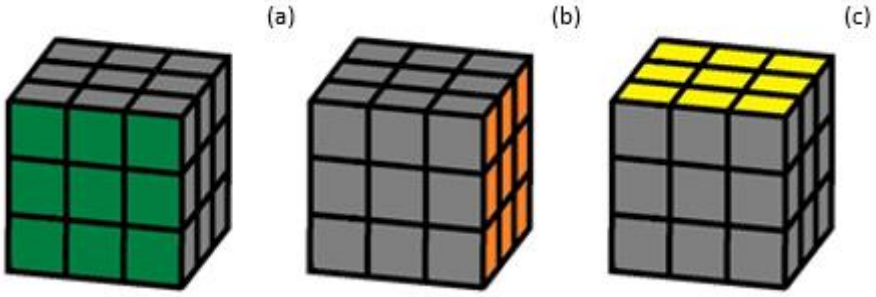
3.2.2.1 Notations

Before describing the algorithms that make up the layer method, it is essential to define some concepts and nomenclatures. Singmaster (1981), published one of the first analyses of the *Magic Cube*. He introduced the following notation:

(I) Face

The WCA, which regulates the Rubik's cube competitions in the world, establishes an official notation for each of the six faces of the traditional 3x3 cube, where each one is represented by a letter: *F (front), L (left), R (right), B (back), U (top) and D (bottom)*.

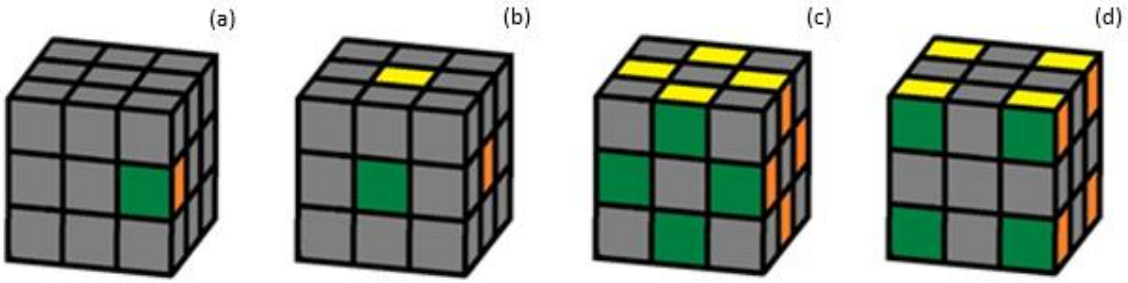
Figure 6: Rubik's cube faces. (a) Face F; (b) Face R; (c) Face U.



(II) Piece

Each piece is formed by the colors of the faces that compose it. An example is the middle piece that belongs to faces F and R, which can be called FR or RF (see Figure 6a). The cube consists of three types of pieces. One is the (i) *center piece*, which is fixed and indicates the color of a given face. The others are the (ii) *edge pieces*, which have two colors, and the (iii) *corner pieces*, which is composed of three colors (Figure 7).

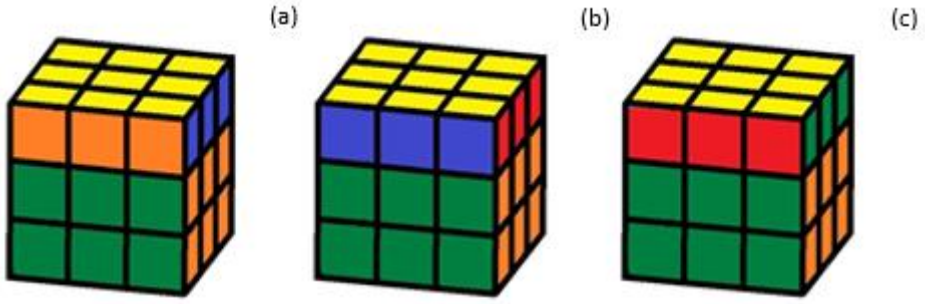
Figure 7: (a) Piece FR; (b) Centers; (c) Edges; (d) Corners.



(III) Movements

Each face produces three types of basic movements: the 90° clockwise, 180° and 90° counterclockwise rotation. The representation of these turns is done using the following symbology: if it is a rotation of 90o clockwise we write only the letter that corresponds to the face; For the rotation of 180o we put the letter plus the number 2; And for the rotation of 90° counterclockwise we write the letter plus' (apostrophe). For example, the face U produces the following rotations, *e.g.*, U, U2 and U ' (Figure 8).

Figure 8: Turns (a) U; (b) U2; (c) U'.



(IV) Permutation and Piece Orientation

The concept of permutation is related to the positioning of the pieces in a specific state. It is nothing more, but exchange one piece for another. Orientation occurs when the piece is in its correct position, but the cube is still not solved

3.2.2.2 Representation

The concept of matrices was used to represent the faces of the cube. Considering that the Rubik's cube consists of six faces, six arrays of integers were created, each with three rows and three columns. Each color was symbolized by an integer and each face was represented by a color, as shown in Table 2.

Table 2: Rubik's cube face representation.

Face	Color	Number
D	White	1
U	Yellow	2
F	Green	3
R	Orange	4
B	Blue	5
L	Red	6

3.2.2.3 Solution

The algorithm implementation for the layer method solution was made in C language. The cube and its faces were represented using the struct concept. There is a method for each type of movement. These methods take one face and spin the edge and corner pieces. Table 3 presents the code snippet for 90 ° clockwise rotation.

Table 3: Code snippet for 90 ° clockwise rotation.

Line	Code
01	void frontClock(int left_f[][3], int front_f[][3],int right_f[][3], int top_f[][3], int bottom_f[][3], int back_f[][3]) {
02	turnClock(front_f);
03	getBottom(top_f, 1);
04	getLeftI(right_f, 1);
05	getTop(bottom_f, 1);
06	getRightI(left_f, 1);
07	setBottom(top_f, rightI_1);
08	setLeft(right_f, bottom_1);
09	setTop(bottom_f, leftI_1);
10	setRight(left_f, top_1);
11	frontClockservo();
12	}

Three methods were created for each face. Each method represents the possible turns that the faces can perform. For example, for face F the following methods were created: frontClock (clockwise 90 °, or F), frontClock180 (180 ° rotation, or F2) and frontIClock (clockwise rotation 90 °, or F ').

Each method receives (Figure 6) the cube faces as a parameter and performs the expected rotation. At the end of each method the movement performed is added to the variable responsible for storing the movement history.

In addition, we also implemented seven other methods that correspond to the steps of the layered algorithm. As the objective of this algorithm is to solve through the concept of layers, then the first face to be assembled was D.

The initial step is to make the cross. Just after, the methods necessary to change the positioning of the found part are called. The purpose is illustrated in the Figure 9. Table 4 presents a code example that corresponds to the first case that the magic cube can be configured.

Table 4: Code part to verify some conditions on first step in the Layer method.

Line	Code
01	void firstStep(int left_f[][3], int front_f[][3], int right_f[][3], int top_f[][3], int bottom_f[][3], int back_f[][3]) {
02	if (bottom_f[1][2] == 1 && right_f[2][1] == 3) {
03	bottomIClock(left_f, front_f, right_f, top_f, bottom_f, back_f);
04	} else if (bottom_f[2][1] == 1 && back_f[2][1] == 3) {
05	bottomClock180(left_f, front_f, right_f, top_f, bottom_f, back_f);
06	} else if (bottom_f[1][0] == 1 && left_f[2][1] == 3) {
07	bottomClock(left_f, front_f, right_f, top_f, bottom_f, back_f);
08	}
09	}

To solve the second layer, the similar mapping is done on the matrix, this time looking for pieces of edge that correspond to the faces F, R, B and L. With the first and second layer completed, only the last layer remains.

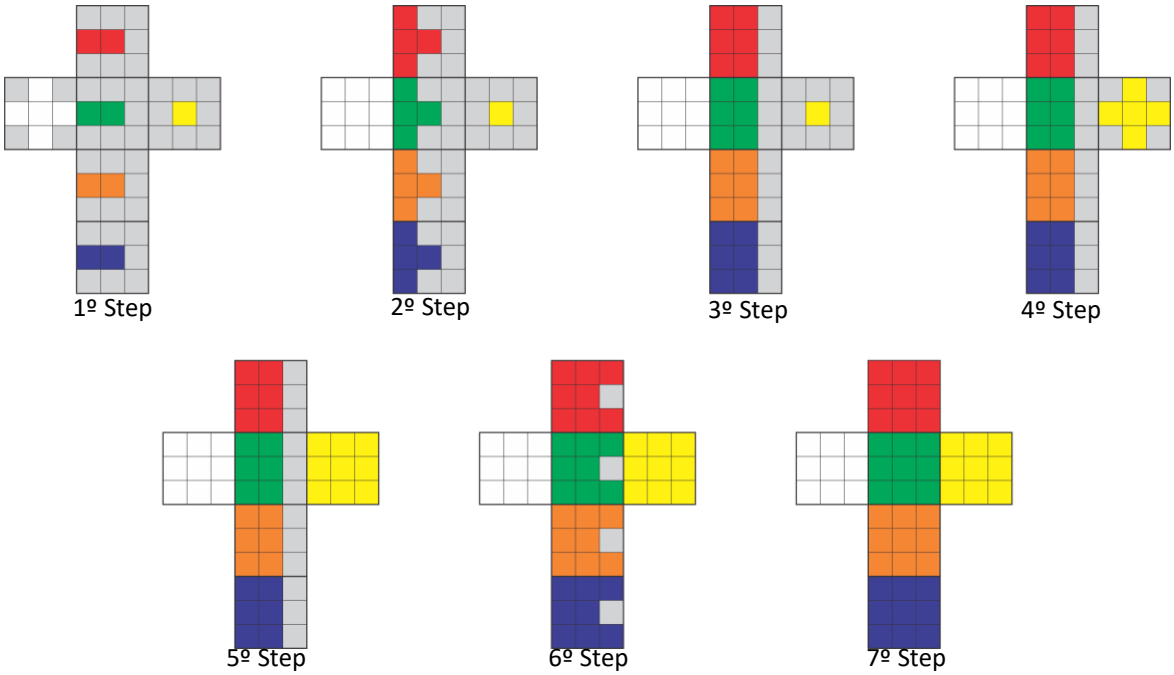
The last four steps of the algorithm are responsible for assembling the third layer. First, it is necessary to cross the U face. This mapping is done by analyzing the parts that were not in their correct position and the parts that were in the correct but inverted position.

The fifth step finishes the face U of the cube completely, raising all its pieces of steel. In this step we do not worry about the parts of faces F, R, L and B of the last layer. For this step the cube can be moved in seven different configurations. In all cases the movements will be the same, changing only the amount of times it should be applied and the position of the cube.

In the sixth step we exchanged the four corner pieces from the last layer. The algorithm locates a side that has two corners of the same color and applies the required sequence of motions.

The seventh and last step makes the permutation of the middle pieces of the last layer. This algorithm searches for one side of the cube that is fully assembled, except for the face D, and verifies the direction the permutation should be applied.

Figure 9: Rubik's cube solution steps.



All code produced to solve the Rubik's cube problem is used as example to teach either Arduino or programming logic principles in technical courses. As well as we apply the results and experiences acquired in the projects to improve the way we recruit new student to be involved in robotic projects.

4 RESULTS

To measure the performance of our solution, we are dedicated to evaluating the performance of the system in each of its subsystems, those systems being the image acquisition and processing system and the robotic mechanism that applies the solution.

4.1 IMAGE ACQUISITION AND PROCESSING SYSTEM

The algorithm was designed to work independently of ambient lighting, however in low light camera noise can cause wrong color readings, for simplicity and functionality, in this application was used RGB color space.

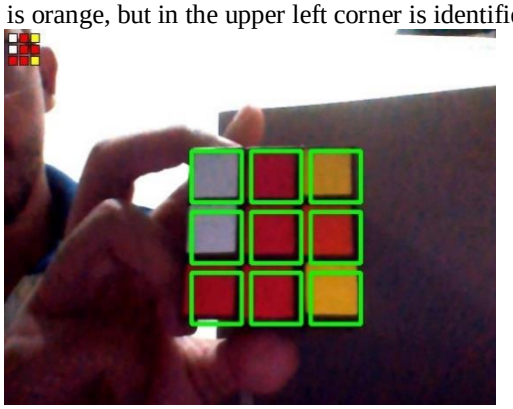
Based on this, it is worth highlighting some effects present in the algorithm output.

4.2 WRONG COLOR IDENTIFICATION

Some colors may be confused with others by the program. Looking at the ranges of green, red and blue, you can see that some colors like yellow and green, and orange and red have intersections in the domain in which they are tracked. Because colors lose

highlight the lower the illumination, they can be confused if they have nearby RGB components (Figure 10).

Figure 10: Middle right block is orange, but in the upper left corner is identified as red.



Alternative to avoid this problem involves the use of controlled lighting. The use of mathematical methods such as square error analysis or the use of inherently light-independent color space such as the human visual system (HSV) -based color system can complement the solution.

4.3 REFLECTION

The cube used has a material that reflects part of the incident light. When reading a face that contains reflex, the algorithm receives distortions in the reading region and may not be able to interpret the color or interpret a wrong color (Figure 11).

Figure 11: lower left block with reflection, reading was not possible, being shown the gray color in the upper left corner.

Reflections on the cube surface can be avoided by using a cube of matte material (incident light is reflected in all directions) or by using external illumination on the cube

face at an angle that does not cause reflection on the face and has power sufficient to blur light from other sources.

5 CONCLUSIONS

The resolution of the Rubik's cube through the mechanism proposed in this work was accomplished successfully, since the aforementioned robotic system, besides having a low cost, compared to other architectures present in the market, presented a satisfactory yield, obtaining solutions in less than 13 minutes.

Through the preliminary results obtained in this work, it can be observed that the use of the articulated mechanism to solve the problematic of the cube has shown to be promising. It should be applied in the classroom for future verification of its effectiveness regarding the improvement of teaching-learning process in disciplines of logic and programming.

In this way, in future works, this methodology will be applied in disciplines of the programming area of technical and superior courses of the IFRN. The group is also studying the development of a smartphone application that allows the user to quickly enter the values of the cube faces into the system from image analysis. This implementation aims to streamline the process and make the human-machine interface more user-friendly, which should greatly improve its application in the classroom.

REFERENCES

ANWAR, S. et al. A systematic review of studies on educational robotics. Journal of Pre-College Engineering Education Research, v. 2, 2019.

AROCA, R.V. et al. Increasing Students' Interest with Low-Cost CellBots. IEEE Transactions on Education, Piscataway, v. 56, n. 1, p. 3-8, 2013.

CUBE20, God's number is 20. Available in: <http://www.cube20.org/>. Access in: 27 mai. 2017. 2017.

JOHANSSON, F. Clique: como nascem as grandes ideias. São Paulo: Portfolio-penguin, 2013.

KORF, R. E. Finding optimal solutions to Rubik's cube using pattern databases. AAAI'97/IAAI'97: Proceedings of the fourteenth national conference on artificial intelligence and ninth conference on Innovative applications of artificial intelligence. Pages 700–705. 1997

MCNAUGHTON, D. (November 1989 - February 1990). "The Rubik Cube: A three-stage approach to mastering it". Junior News (Al-Nisr, Dubai, UAE).

ORSI, D. F. O. et al. Quando ensinamos é quando aprendemos – Elaboração de vídeos na avaliação continuada – Metodologias ativas. Revista Compartilhe Docência, 2017.

PINTO, D. O. Metodologias Ativas de Aprendizagem: o que são e como aplicá-las. Available in: < https://blog.lyceum.com.br/metodologias-ativas-de-aprendizagem/>. Access in: 09 abr. 2020. 2020.

ROHITH, S. P. et al. International Autonomous Rubik's Cube Solver Bot. Journal of Scientific Research and Engineering Development, v. 2, 2019.

SINGMASTER, D. Notes on Rubik's Magic Cube. Harmondsworth, Eng: Penguin Books. ISBN 0-907395-00-7. 1981.

SOARES, A. A. A. F. *et al.* (2015) "A preparação para a olimpíada de robótica como projeto", In: 6º Workshop of Robotics in Education, Uberlândia/MG, 2015.

VILAS BOAS, J. M. *et al.* "The use of a maze solver robot to support the teaching and learning process", In: 6º Workshop of Robotics in Education, Recife/PE, 2016.