

Intelligent Rubik's Cube Solver

Chun-Fei Hsu, Shih-An Li, Szu-Hao Lien, Han-Sheng Chang, and Yu-Cheng Lai

Department of Electrical Engineering, Tamkang University

E629, No.151, Yingzhuan Rd., Tamsui Dist., New Taipei City 25137, Taiwan (R.O.C.)

fei@ee.tku.edu.tw

lishyhan@ee.tku.edu.tw

Abstract—This paper describes using an FPGA chip to solve a Rubik's Cube. A CMOS camera is used to sense the colours of Rubik's Cube. NXT motors and a Lego building blocks construct a hand-like mechanical structure to push and turn the Rubik's Cube. We build a SOPC system and a Nios II processor is used to recognise the colors of Rubik's Cube and calculate the resolving actions of Rubik's Cube.

Keywords— Rubik's Cube; Artificial Intelligence; FPGA; NXT

I. INTRODUCTION

Nowadays emphasis on computerized machines, intelligence era, the application of automated machinery and equipment rather attention, because is the application of automation equipment can reduce people need to pay labor force, increasing production volume, and engage in hazardous work.

We use an FPGA-based system architecture, using the found formulas of calculation, CMOS camera for the sense of sight, NXT motor and Lego to build the structure and achieve turning around the magic square and solve this equation.

We hope artificial intelligence can be embedded in robots in the future and let them get more abilities to solve more problems for servicing humans.

II. MECHANISM DESIGN

Design and implementation of FPGA-based to solutions for Rubik's Cube Robot, and using the Lego design portfolio. Therefore, to reduce cost and time of development, as shown in Fig.1., using the CMOS camera to solution six sides for the Rubik's cube. The FPGA based image processing detects colors on the face of a Rubik's cube, by using image processing techniques to analyse the cube's color distributes of condition, and the steps required to calculate the solution Rubik's Cube.



Fig. 1 Rubik's cube solving machine

Because the restriction of machine set design, the robot can only do to turn-over of push action and situ rotation of changing face of transfer action, and arm grip box to Spin box of the bottom, as shown in Fig. 2 and Fig. 3 and Fig. 4. For matching with a solution the action produced by machine, if the certain color revolves however many degrees, then we have to move the color's surface to the bottom, then clip the first surface of revolving. The robot's actual problem-solving action must be designed to solve the Rubik's Cube.

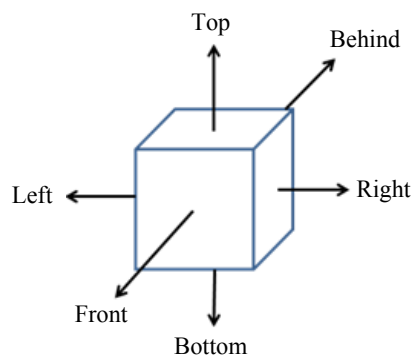


Fig. 2 Rubik's Cube six definitions of positions

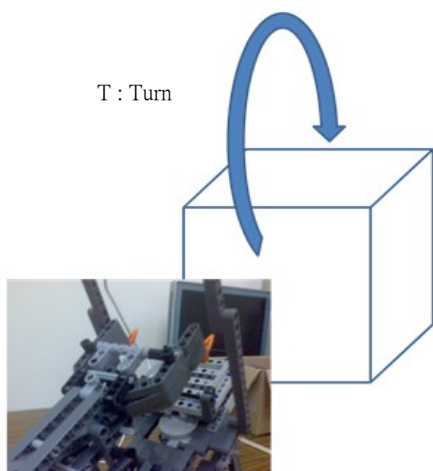


Fig. 3 Flip the Rubik's Cube

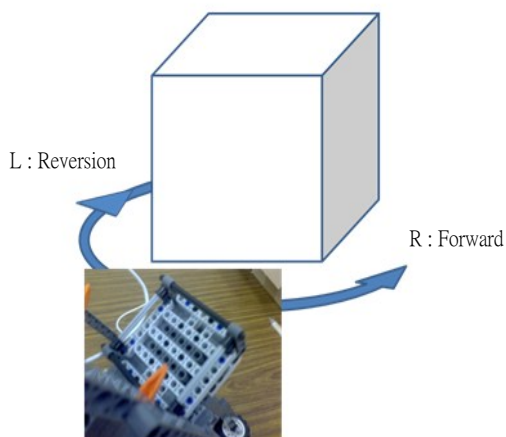


Fig. 4 Rotate the Rubik's Cube

III. SYSTEM ARCHITECTURE

FPGA-based system architecture ,using the found formulas of calculation, CMOS camera for the sense of sight , NXT motor and Lego to build the structure and achieve turning around the magic square and solve this equation

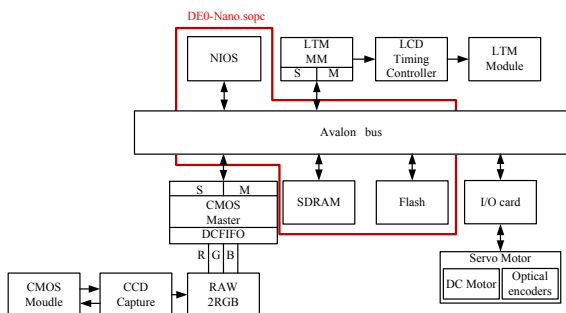


Fig. 5 System Architecture

IV. SERVO MOTOR

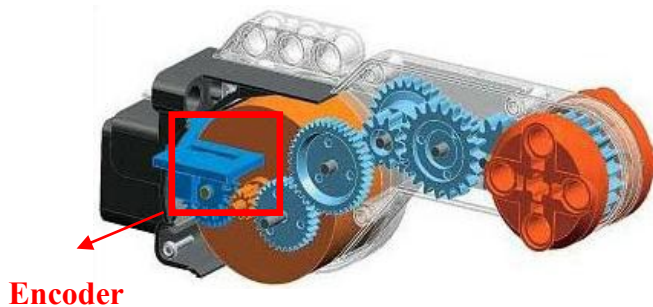


Fig. 6 Internal structure of the motor

This motor is NXT motor, includes a rotation encoder, returning to the NXT the position of the shaft with 1° resolution. Because of the special connector of this motor (non-standard phone plug type), a cable adapter is required to drive this motor with regular 9V sources. Not recommended for use with a RCX which can't deliver the high current that this motor can consume. Slow rotation speed, minimizing the need of external gear train.

Speed is controlled using Pulse Width Modulation, and the AVR controller handles PWM generation. Feedback is provided by a quadrature encoder attached to the motor, and allows detection of rotation direction and amount

Each LEGO Smart Motor includes a built in encoder that can be used to program the robot to move around in a very precise manner. The built-in encoder has a resolution of 360 counts per revolution of the axle or one tick per degree. Therefore every time the robot wheel spins one wheel rotation the robot counts 360 encoder counts.

The gear reduction alone involves eight different gears, and because the Rotation Sensor is way back by the motor end, it has one degree of revolution on the output shaft.

Encoder wheel and enclosure back parts. We have 12 slits in encoder, and motor to encoder gear reduction is 10:32. So for 1 turn of output hub, encoders turns $48 \cdot 10 / 32 = 15$ turns, optical detectors sees $15 \cdot 12 = 180$ slits. Using both sides of slits gives nominal 360 ticks per turn resolution.

The encoders work by sending out a pulse every time you rotate the shaft. There are two encoders in each motor. The two output waveforms are 90 degrees out of phase, these signals are decoded to produce a count up pulse or a count down pulse. When rotating the shaft, the pulses will look like this:

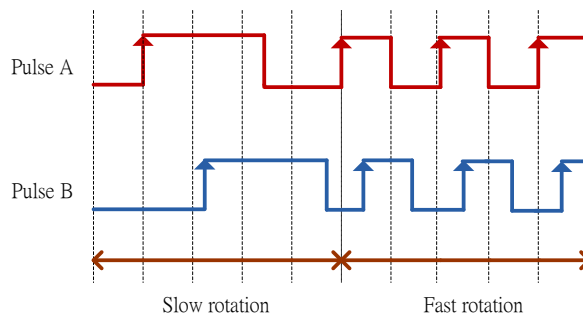


Fig. 7 Forward movement

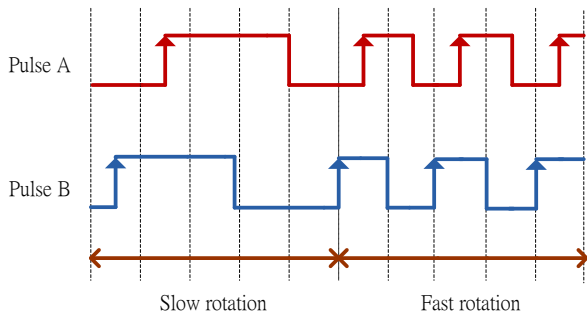


Fig. 8 Backwards movement.

V. I/O CARDS

In this paper, use the Home-made I/O Card, as shown in Fig. 9 and Fig. 10. Include rotary encoder of decoder count, motor drive IC, 4-channel D/A , 2-Channel ADC and power supply. DC-rated voltage of 15V, and the output voltage 7.5V to supply other sensors and control panel. In the custom-designed I/O Card below, there is a 2-channel motor transmission port that transport PWM signals to control DC motor and receive motor feedback signal.

After the optical decoder, by the I/O card export to DE0 NANO board. The DAC made observed signal with TLV5636IC after conversion to the oscilloscope, observation data within the system status changes.

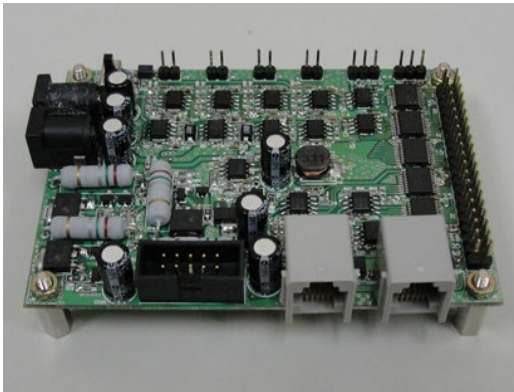


Fig. 9 Home-made I/O Card

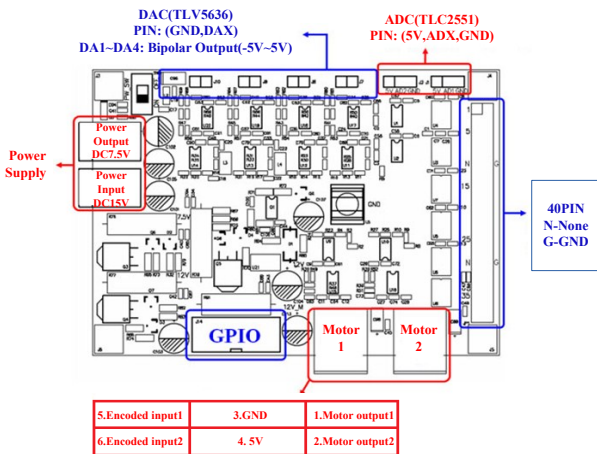


Fig. 10 Illustration of Home-made I/O Card

A. Motor drive IC

In this I/O card, use TA7291 of the motor drive IC, where the schematic is shown Fig.3. TA7291 use to control the DC

motor,IN1 and IN2 which can be as forward ,backward, stop, and brake control and direction of such action., Vref is in accordance with the size of the input voltage to change the DC motor speed.

The higher the voltage Vref, the quicker the DC motor speed is. The lower voltage Vref, the slower the DC motor is.

The DC motor positioning control system architecture diagram is shown Fig. 11.

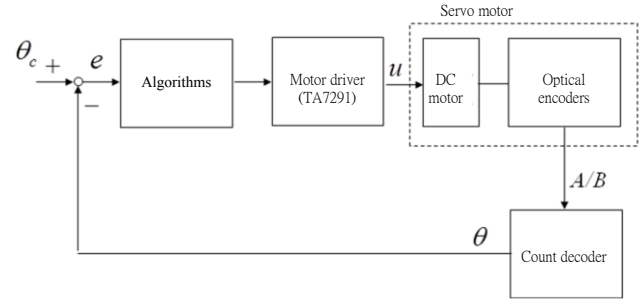


Fig. 11 DC motor positioning control system architecture diagram

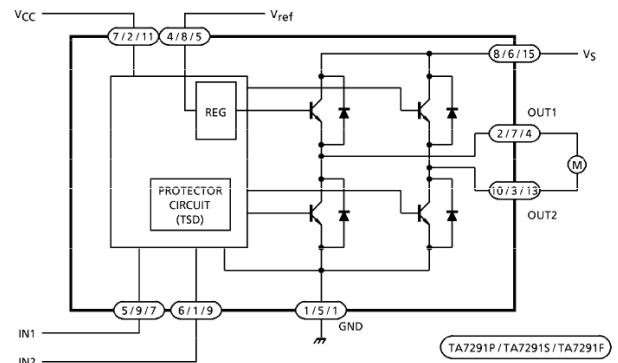


Fig. 12 TA7291 Schematic

B. Analog-to-digital converters (ADC)

The TLC2551 is a 5-V supply, 12-bits low-power, miniature, CMOS analog-to-digital converters (ADC) with auto power down, as shown Fig. 13.

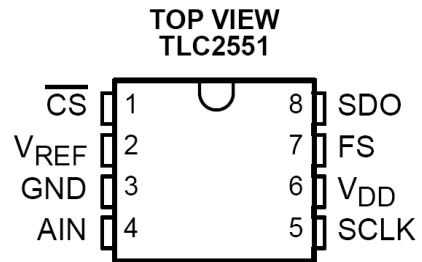


Fig. 13 TLC2551 IC

C. Digital-to-analog converters DAC

The TLV5636 is a 2.7-V to 5.5V, 12-bit low-power, digital-to-analog converters (DAC) with internal reference and power down, as shown Fig.14.

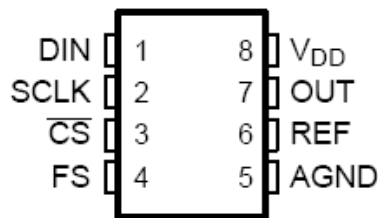


Fig. 14 TLV5636 IC

D. Count decoder

The rotary encoder is an optoelectronic method to convert mechanical rotation angle of the shaft for precision sensor of the digital signal output, include incremental rotary encoders and absolute rotary encoders.

Using the optoelectronic incremental encoder method is the uniform grating on the pulse Transducer rotates together with the shaft and the transducer will distribute evenly in light transmission section and blackout section.

Incremental encoders have not a fix zero point, the output is proportional between the increment of angle and pulse and the pulse number counter. It will send a pulse signal in each turn photic zone and then the will count one pulse to calculate the increment of angle.

The single-channel incremental encoder has only one pair of optical couplers, so it can only send one pulse sequence.

The AB phase encoder has two pairs of internal opt couplers; it outputs the phase difference of 90 between two sets of pulses. Forward and reverse pulses will be leading or lagging in same time.

B phase pulse is rising and A phase pulse is just opposite then forward and reverse. So by using AB phase encoder we can confirm the direction by rotation of the shaft easily.

VI. DE0-NANO

The DE0-Nano board introduces a compact-sized (49 mm x 75 mm) FPGA development platform suited for prototyping circuit designs such as robots and "portable" projects. The board is designed to be used in the simplest possible implementation targeting the Cyclone IV device up to 22,320 LEs.

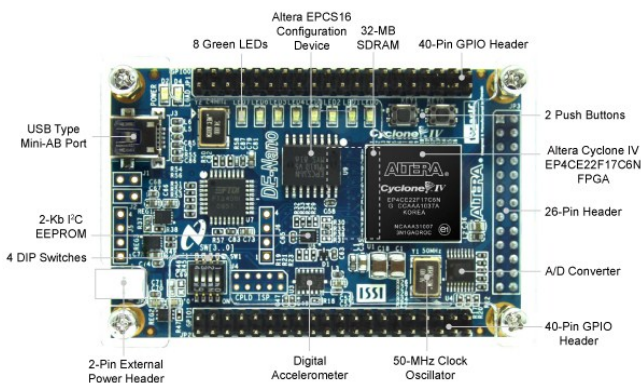


Fig. 15 DE0-Nano

The DE0-Nano has a collection of interfaces including two external GPIO headers to extend designs beyond the DE0-Nano board, on-board memory devices including SDRAM

and EEPROM for larger data storage and frame buffering, as well as general user peripheral with LEDs and push-buttons.

The advantages of the DE0-Nano board include its size and weight, as well as its ability to be reconfigured without carrying superfluous hardware, setting itself apart from other general purpose development boards. In addition, for mobile designs where portable power is crucial, the DE0-Nano provides designers with three power scheme options including a USB mini-AB port, 2-pin external power header and two DC 5V pins.

DE0-Nano Board Features

Cyclone® IV EP4CE22F17C6N FPGA

- 22,320 Logic elements (LEs)
- 594 Embedded memory (Kbits)
- 66 Embedded 18 x 18 multipliers
- 4 General-purpose PLLs
- 153 Maximum FPGA I/O pins

Configuration Status and Set-Up Elements

- On-board USB-Blaster circuit for programming
- FPGA Serial Configuration Device (EPCS)

Expansion Header

- Two 40-pin Headers (GPIOs) provides 72 I/O pins
- Two 5V power pins, two 3.3V power pins and four ground pins
- One 26-pin header provides 16 digital I/O pins and 8 analog input pins to connect to analog sensors, etc

Memory Devices

- 32MB SDRAM
- 2Kb I2C EEPROM

General User Input/Output

- 8 green LEDs
- 2 debounced push-buttons
- 4 dip switches

G-Sensor

- ADI ADXL345, 3-axis accelerometer with high resolution (13-bit)

A/D Converter

- NS ADC128S022, 8-Channel, 12-bit A/D Converter
- 50 kps to 200 kps

Clock System

- On-board 50MHz clock oscillator

Power Supply

- USB Type mini-AB port (5V)
- Two DC 5V pins of the GPIO headers (5V)
- 2-pin external power header (3.6-5.7V)

VII. NIOS II IMAGE CAPTURE

We process color image via the Nios II system. Using CMOS camera (shown in Fig. 17) to capture color image, processing by FPGA and transform to image by VGA controller. We use 8 bits of data in the FPGA operation, which range of gray is from 0 to 255. The size of the image is 800x500. It means the image has 800 rows of pixels and 500 columns of pixels, totalling 400000 pixels in every image. System work step is as follows: Using a CMOS camera for capturing an image, there is an output 8-bits integer signal to DE0-Nano (shown in Fig. 18) to collect signal, preprocess

image and save image after collecting image data and then wait for FPGA to process. After the FPGA processes successfully, the input image signal VGA controller displays on the in LTM (shown in Fig. 19) monitor and the system uses JTAG to download the image. Fig 1 is the actual pin map.

A. Avalon Bus

Avalon Bus connects to the processor and peripheral devices, support for peripheral devices and transfer operations support multiple bus master device and access.

Fig. 20 is the system architecture of Avalon Bus.



Fig. 16 actual pin map



Fig 19 LTM

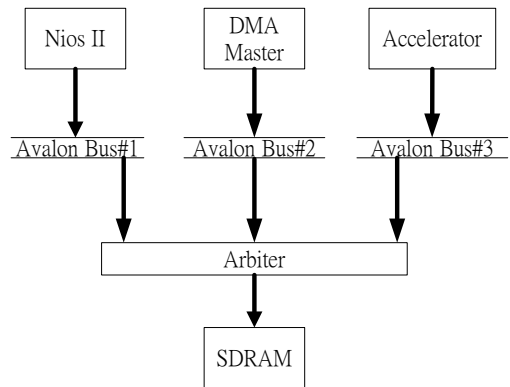


Fig. 20 Avalon Bus Master System



Fig. 17 CMOS camera

VIII. NIOS II IMAGE PROCESS

In this section, we will catch each surface of Rubik's Cube and analysis the RGB value of each surface. As shown in Fig.21, we first catch each surface of Rubik's Cube, and save to disk as a bmp file. Because the center square of each surface will never move. This is because the square is the axis of the cube. In addition to, it's also the basis when detect the color of each surface. Table I. is the color Sample of Rubik's Cube.

TABLE I

THE COLOR SAMPLE OF RUBIK'S CUBE

Color	R	G	B
Red	148	0	0
Orange	255	64	0
Yellow	180	135	24
Green	0	47	54
Blue	0	0	94
white	186	182	192

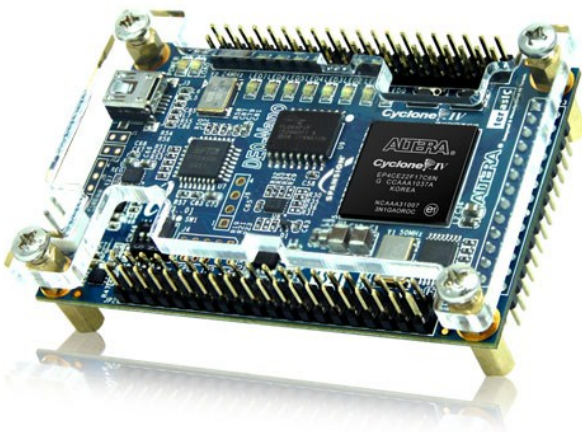
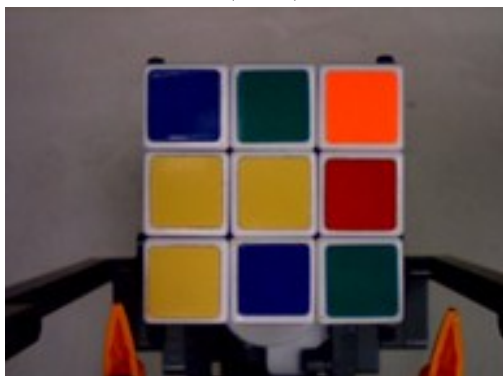


Fig. 18 DE0- Nano



a.(white)



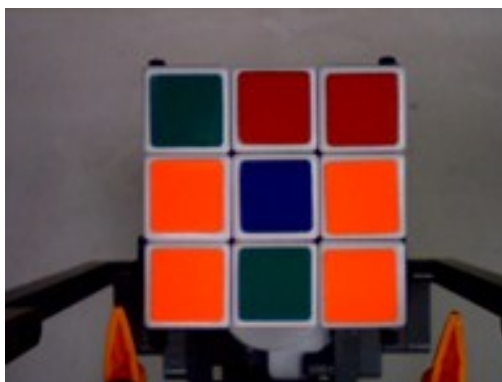
b.(yellow)



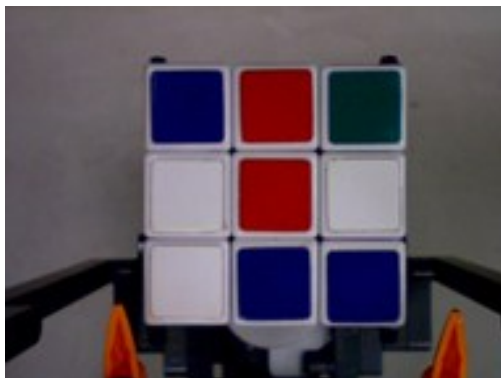
c.(orange)



d.(green)



e.(blue)



f.(red)

Fig. 21 Six surface image of the Rubik's Cube

After getting the information of Rubik's Cube, we need to distinguish the background image and cube image. As shown in Fig. 22. We drawing a red rectangle along on the edge of cube image and define the range as the detection in range.



Fig. 22 Sample of Image Cpture

According to the RGB sample result of Table I, we test against to nine square of each surface. Here we use white surface (1) to detect, getting the average pixel of nine squares. The result is shown in Table II. White square has bigger deviation because of the trademark. In order to solve the problem, we increase the detection range of center square.

TABLE II
THE RGB VALUE OF (1).

R : 0 G : 26 B : 30	R : 255 G : 56 B : 0	R : 166 G : 162 B : 172
R : 0 G : 24 B : 26	R : 159 G : 154 B : 151	R : 0 G : 0 B : 75
R : 0 G : 0 B : 71	R : 140 G : 0 B : 0	R : 170 G : 164 B : 174

Through the predicate of program to classify the color of (1). yellow classify to 1, green classify to 2, red classify to 3, orange classify to 4 blue classify to 5 , white classify to 6.

Use 3x3 matrix to illustrate we will obtain as follows:

$$\text{WhiteSurface} \begin{bmatrix} 2 & 4 & 6 \\ 2 & 6 & 5 \\ 5 & 3 & 6 \end{bmatrix} (1)$$

$$\text{RedSurface} \begin{bmatrix} 5 & 3 & 2 \\ 6 & 3 & 6 \\ 6 & 5 & 5 \end{bmatrix} (2)$$

$$\text{OrangeSurface} \begin{bmatrix} 3 & 6 & 1 \\ 4 & 4 & 1 \\ 1 & 5 & 6 \end{bmatrix} (3)$$

$$\text{YellowSurface} \begin{bmatrix} 5 & 2 & 4 \\ 1 & 1 & 3 \\ 1 & 5 & 2 \end{bmatrix} (4)$$

$$\text{GreenSurface} \begin{bmatrix} 3 & 2 & 4 \\ 1 & 2 & 1 \\ 3 & 6 & 1 \end{bmatrix} (5)$$

$$\text{BlueSurface} \begin{bmatrix} 2 & 3 & 3 \\ 4 & 5 & 4 \\ 4 & 2 & 4 \end{bmatrix} (6)$$

If we make the value of each surface as an unfolded map, we will find a more obvious color distribution. As shown in Figure 8., the axis is fixed and it is the color basis of the origin surface. If we observe other four adjacent surfaces, Red, green, orange and blue must be arranged in a clockwise, while the other side must be yellow.

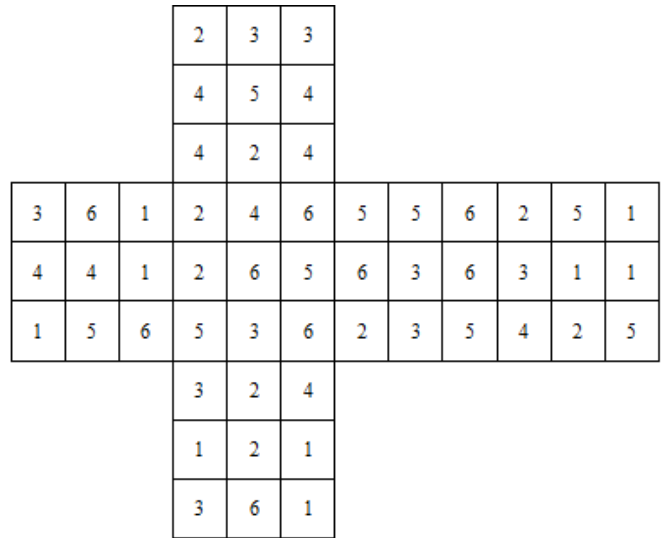


Fig. 23 unfolded map of Rubik's Cube

IX. RUBIK'S CUBE SOLUTION

First, we must define the Rubik's cube six faces of the code and its corresponding color, shown in Fig. 24.



Fig. 24 Rubik's Cube six face color and code

While rotating the cube, the top face colors are scanned using the CMOS camera. Now we can begin solve Rubik's cube.

Most people would think that a Rubik's Cube is a plane surface solution, but the solution is actually Rubik's Cube layer by layer to the solution, the basic solution is make the first layer's color the same and the second layer, the final layer. This basic transfer method called LBL, which is Layer By Layer.

But the Rubik's Cube using LBL solution required too many rotations, so we opted to change to use CFOP quick solution. The improvement is enhanced by the basic solution made using about 80 algorithms, rotating number of steps can be reduced to about 50-70 steps, mainly can be divided into the following four steps, shown in Fig. 25.

A. Cross

The cross is done intuitively. The cross is done in eight moves or less, though in over 99% of the cases it can be done in 6 moves.

B. First two layers (F2L)

The strategy here is to join a corner of the first layer with the edge that goes above it, and then insert that pair.

C. Orient last layer (OLL)

OLL consists of 57 different algorithms. This stage involves manipulating the top layer so that all the top cubes have the same colour on top.

D. Permutation of Last Layer (PLL)

The final stage involves moving the pieces of the top layer while preserving their orientation. There are a total of 21 algorithms for this stage.

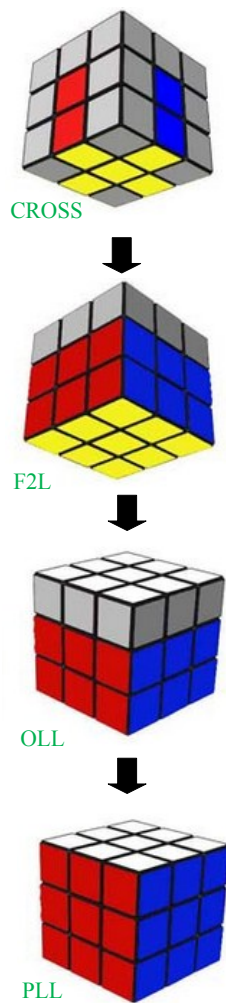


Fig. 25 CFOP solution Process

REFERENCES

[1] Bob Burton (2005-2012) Cubewhiz.com. Available: <http://www.cubewhiz.com>

[2] Hans Andersson (2011) Tilted Twister 2.0. Available: <http://tiltedtwister.com/tiltedtwister2.html>

[3] Unifish unicube.tw. Available: <http://www.unicube.tw/>

[4] Terasic Technologies Inc (2013). Available: <http://www.terasic.com.tw>

[5] Po-nien Pao. (2010) Design of a Rubik's Cube Solver Robot using Matlab RWTH toolbox[Online]. Available: <http://ndltd.ncl.edu.tw/cgi-bin/gs32/gsweb.cgi?o=dncldr&s=id=%22098CHPI5442034%22.&searchmode=basic>

[6] http://en.wikipedia.org/wiki/Lego_Mindstorms_NXT#Technical_specifications, Wikipedia NXT introduction

[7] Rubik's Cube Solution Methods by Josef Jelinek Available: <http://rubikscube.info/>

[8] Youtube: Robot shows how to solve Rubiks Cube Available: <http://www.youtube.com/watch?v=bNAnUygqOYc&feature=related>

[9] Youtube: Danny's fully automated Lego Rubik's cube solver Available: <http://www.youtube.com/watch?v=htnL1KTpaY8&feature=related>

[10] Herbert Kociemba. Cube Explorer. <http://kociemba.org/cube.htm>, 2006.

[11] <http://www.erobot.com.tw> (K-Kingdom)