Herbert
Autonomous Robotic Rubik's Cube Solver

Jonathan Whitaker, Dylan Lytle, Matt Frandsen, Li Lao
Dept. of Electrical and Computer Engineering, University of Utah

Abstract

Project Herbert is an autonomous robotic Rubik's Cube solver that is composed of a complex network of mechanical and electrical devices. In this project our team will be interfacing with these complex mechanical and electrical devices so as to design and create an autonomous robot that is capable of achieving record-breaking Rubik's Cube solution sequences. This project will be an integration of various technologies including: mechanical actuators, electrical stepper motors, single-board computers, field-programmable gate arrays, and video cameras.

I. Introduction

The purpose of this project is to create an autonomous robotic Rubik's Cube solver through the integration of several components. The main components needed in our project include video cameras, electrical stepper motors, mechanical actuators, a single-board computer (SBC), and field-programmable gate arrays (FPGAs). The video cameras connect to the computer through standard universal serial bus (USB) 2.0 protocol [1]. The video cameras are responsible for capturing the initial configuration of the Rubik's Cube, and the single board computer is responsible for processing the video frame information and generating a matrix model for the initial state of the cube. After generating a matrix model for the initial cube, the computer will then apply Kociemba's algorithm [2] (an optimal algorithm used to solve a Rubik's Cube) to generate a solution sequence that can be processed sequentially. The solution sequence that Kociemba's algorithm will return will be in the standard notation used in Rubik's Cube discussion and theory (see Appendix A). As each solution sequence is processed, the computer will communicate through an RS232 serial connection to an FPGA board which will drive the mechanical actuation and stepper motor rotations needed to physically manipulate the cube. The FPGA will act as a system control board and will be responsible for controlling the actions of two motor control boards and a relay board used to trigger the mechanical actuators. An overview of this process can be found below in Fig. 1 of section II.

Upon completion of the robotic Rubik's Cube solver, if time permits, we hope to take Project Herbert one step further and obtain a Guinness World Record for the fastest robot to solve a Rubik's Cube. CubeStormer3 is the current record holder [3]. The uniqueness of this project is rooted in the optimizations that we will have to make in each component of our system, especially the electro-mechanical stepper motors and the mechanical actuators. We will have to fine-tune these mechanical components to perform physical operations quicker than the human eye in order to obtain record-breaking speed. In this process we will have to maintain the precision needed to rotate the cube.

II. Project Design

A. Capturing the Cube with OpenCV

1) Camera Orientation: As seen in Fig. 1, two cameras will be connected to a SBC through a standard USB 2.0 connection. Each one of these cameras is responsible for capturing exactly three of the six faces of the cube. One camera will capture the front, top, and left faces of the cube. The other camera will be responsible for capturing the back, bottom, and right faces of the cube. To do this, each camera will

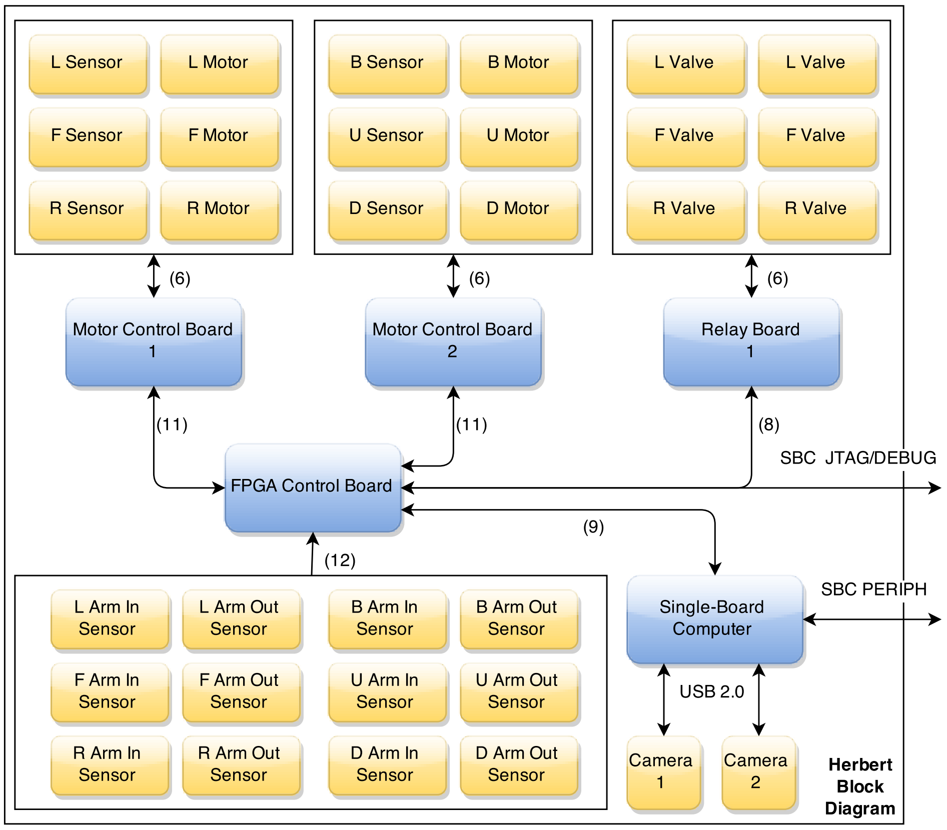

Fig. 1. Herbert Block Diagram

be positioned so that it faces the corner piece that connects the edges of the faces it is responsible for capturing. An example of this cube orientation is shown in Fig. 2.

2) Utilizing OpenCV for Image Processing: OpenCV is an open source computer vision and machine learning software library [4]. We will be leveraging this library to perform image analysis on a Rubik's Cube, which will allow us to generate a matrix model for any cube orientation. The OpenCV library will be installed on the SBC, which will enable us to capture and analyze video frames from the USB connected cameras.

Capturing the faces of a cube requires three primary operations:

- **Grayscale conversion** - A video frame will be captured and converted to Grayscale using OpenCV's *cvtColor* function [5]. Grayscale conversion transforms RGB pixel values into black and white intensity values [6]. Converting to Grayscale allows an easy transformation to a binary (black and white) image, which is used to filter out features that are not important.
- **Canny edge detection** - Canny edge detection is a multi-stage algorithm used to detect a wide range of edges in an image [7], [8]. We will be using OpenCV's *Canny* function [9] to identify the edges of the Rubik's Cube, which, when combined with contour filtering, will allow us to dynamically identify the planes in the frame that represent the three faces of the cube.

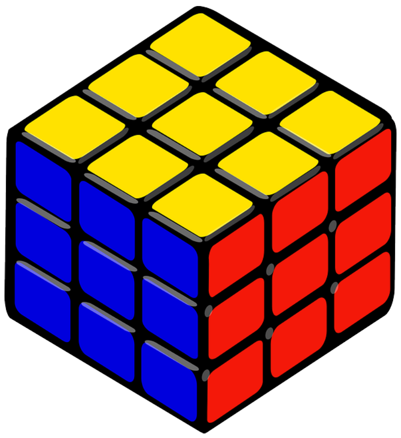
Fig. 2. Camera Positioning

- **Contour filtering** - Contour filtering will be used to identify the contours of the Rubik's Cube within the image. OpenCV's *findContours* function [10], [11] will allow us to identify the region of space in the image where the Rubik's Cube resides. Likewise, it will enable us to identify individual cubelets on each of the three faces of the cube.

After the individual cubelets have been identified within the image, a matrix model like that shown in Fig. 3 will be generated. Each individual position (U1, U2, F1, F2, etc.) can be represented as a single ASCII character indicating the color at that cubelet position. Table I defines the mapping scheme we will employ for the Rubik's Cube matrix model. Once this matrix model has been generated for the cube, we will pass it on to an application written by Greg Schmidt known as *Kcube* [13]. Kcube will process the cube permutation (see II-B), apply Kociemba's algorithm, and generate a solution sequence.

TABLE I
CUBELET COLOR TO ASCII CHARACTER MAPPING

COLOR	CHARACTER
WHITE	'W'
RED	'R'
BLUE	'B'
GREEN	'G'
ORANGE	'O'
YELLOW	'Y'

B. Kcube and the Solution Sequence

Kcube is a C++ application developed by Greg Schmidt that utilizes Kociemba's two-phase algorithm which uses two stages of an iterative depth first search algorithm [13]. The Kcube application will be used to generate the solution sequence needed to solve the Rubik's Cube that was captured during the image processing phase. The matrix model generated from the image processing phase will allow us to provide Kcube's command-line interface with the cube representation needed to generate a solution sequence. Kcube's command-line interface takes six parameters, one for each face of the cube. The values for these parameters are the color characters at each of the cubelet locations for that face (as seen in Fig.

```
             |************|
             |*U1**U2**U3*|
             |************|
             |*U4**U5**U6*|
             |************|
             |*U7**U8**U9*|
             |************|
|************|************|************|************|
|*L1**L2**L3*|*F1**F2**F3*|*R1**R2**F3*|*B1**B2**B3*|
|************|************|************|************|
|*L4**L5**L6*|*F4**F5**F6*|*R4**R5**R6*|*B4**B5**B6*|
|************|************|************|************|
|*L7**L8**L9*|*F7**F8**F9*|*R7**R8**R9*|*B7**B8**B9*|
|************|************|************|************|
             |************|
             |*D1**D2**D3*|
             |************|
             |*D4**D5**D6*|
             |************|
             |*D7**D8**D9*|
             |************|
```

Fig. 3. Rubik's Cube matrix representation

3). For example, to solve the scrambled cube shown in Fig. 4 you would invoke Kcube with the following command:

```
c:>kcube L:GGWWOWBRB F:GWGBGYWBO U:YOOOWYROY D:ORGWYYYRB R:OGBBRYWRR B:YBROBGWGR
```

Kcube will then process the input parameters and generate a sequence of twenty-three or less moves (see B) that, when applied to the cube, will solve the cube. Each move will be mapped to a unique integer value (see Table II), and these values will be transmitted sequentially over an RS232 serial connection to the FPGA control board, at which point the control board will take responsibility for controlling the electro-mechanical stepper motors and mechanical actuators needed to physically manipulate the cube.

TABLE II
CUBE MOVES TO INTEGER MAPPING

MOVE	**VALUE**	**MOVE**	**VALUE**	**MOVE**	**VALUE**
F	1	R	7	D	13
F’	2	R’	8	D’	14
F2	3	R2	9	D2	15
L	4	U	10	B	16
L’	5	U’	11	B’	17
L2	6	U2	12	B2	18

C. Mechanical Actuators

To physically manipulate the cube, Herbert will employ a six arm design. One arm for each face of the cube. In order to achieve a six arm design, each arm must actuate in and out so as to avoid conflict with

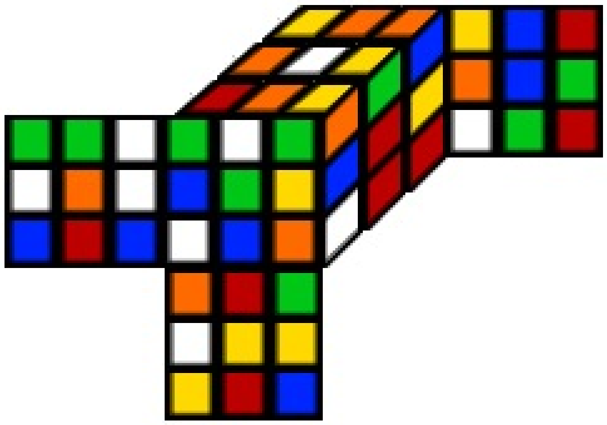

Fig. 4. A scrambled cube

the other arms. This actuation process is a time critical component of the design, and therefore needs to be as fast as possible. We initially planned on implementing the arm actuation with motors. However, preliminary testing has showed that using motors to convert rotary motion into linear motion is too slow, and using a linear motor actuator is too costly. Our design requires high speed and affordable cost. We believe pneumatic actuation is the solution to this problem. Each of the arms will be attached to a double action pneumatic air cylinder as show in Fig. 5.

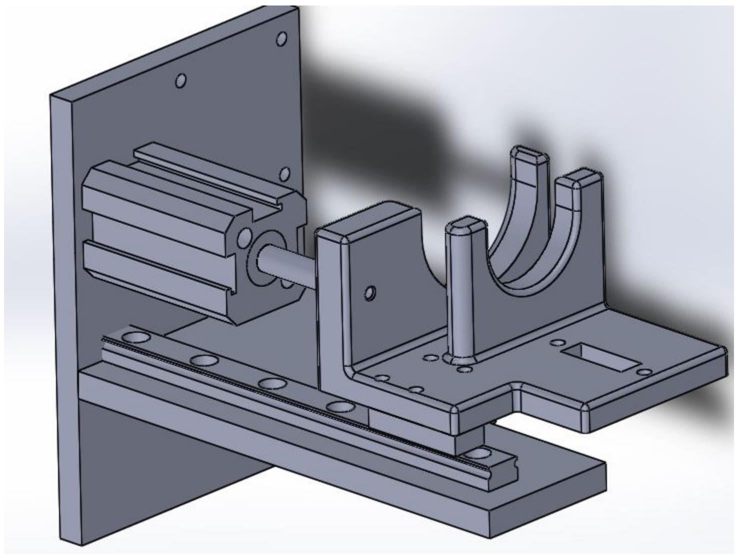

Fig. 5. Pneumatic Air Cylinder

The FPGA control board will be responsible for controlling a relay control board (see Fig. 1) which will control coaxial pairs of air cylinders. The actuation distance for each control arm will be fixed. The relay switches coaxial pairs of arms will either be in an extended or retracted position. Potential optimizations and more stable cube manipulations are obtained by simultaneously extending and retracting coaxial pairs of arms. The linear actuation motion allows two coaxial pairs of arms to extend, thus encasing two sides of the cube in the sockets of the arms. After a pair of arms extend, a stepper motor will spin the arm corresponding to the appropriate move from the solution sequence (see section II-D). Each air cylinder will be provided approximately 80-100 psi supplied from an air compressor. To protect against any arm collisions only one pair of arms will be in the extended position at any given time.

D. Electro-mechanical Stepper Motors

Each actuating arm will have a stepper motor which will be responsible for rotating a single face. A stepper motor will rotate a face either 90 degrees or 180 degress clockwise or counter-clockwise based on the solution move that is being processed (as specified in Appendix A). A square axle (as seen Fig. 6) will be fastened to the stepper motors. This axle will twist the arm pieces in order to spin the faces of the Rubik's Cube.

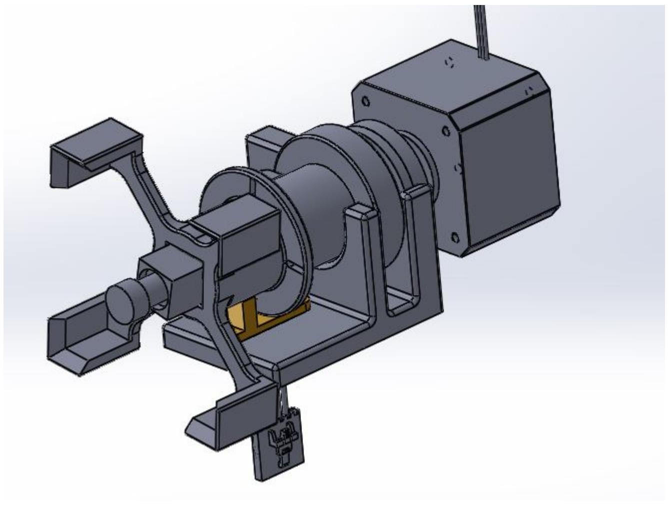

Fig. 6. Stepper Motor Arm

Each stepper motor will be driven by a motor control board which will be controlled by the FPGA control board (see Fig. 1). The motor control boards will contain a motor driver chip for each stepper motor. The FPGA control board will be responsible for controlling the angular and temporal timing of each stepper motor rotation.

III. Schedule

Our work will be distributed over the upcoming summer months and Fall 2015 semester, spanning May 13-December 2015. Table III outlines the milestones we hope to have completed by the end of this upcoming summer. A brief description of each one of these milestones is presented in Section III-A. Come Fall semester 2015, we hope to be integrating each component and testing the robustness of our system.

TABLE III
HERBERT ITEMIZED TIMELINE

Tasks	Start	Finish	Milestone
	2015-05-13	2015-05-20	OpenCV Setup
Setup Windows Embedded VM using Vagrant			
Install Camera Drivers and Interfaces			
Download and Install OpenCV			
	2015-05-20	2015-06-19	Basic Cube Capture
Research Object Detection Algorithms			
Experiment with Canny Edge Detection			
Experiment with Contour Identification			
	2015-06-19	2015-07-02	Finalize Object Detection Software
Implement Object Detection Algorithm			
Generate Output Representation of Cube			
	2015-05-13	2015-05-26	Mechanical Tasks
Build The Herbert Frame			
Determine fastener hardware			
Gather Hardware			
Print 3D Arms			
Attach Hardware to Frame			
	2015-05-13	2015-06-12	Firmware for Solution Sequence
Implement firmware for solution execution			
Create Offline Command in comm.c			
	2015-06-15	2015-07-10	Herbert Test Application
Add tab for Basic Movements			
Add tab for Manual Solution Input			
Add tab for testing motors and sensors			
	2015-12-16	2015-07-11	Integration and Testing
T.B.D.			

A. *Milestones*

- **OpenCV Setup**
  In this milestone we will configure a stable development environment for OpenCV, which will include setting up a development box, installing camera device drivers, and installing and configuring OpenCV.
- **Basic Cube Capture**
  In this milestone we will familiarize ourselves with the OpenCV technologies and experiment with Canny edge detection and contour identification algorithms. This will give us the background we need to implement our cube capture application.
- **Kcube and the Solution Sequence**
  By the end of this milestone we will be able to capture a cube, process it using the Kcube application, and generate a solution sequence that can be passed on to the FPGA control board for further processing.
- **Mechanical Tasks**
  This milestone is dedicated to getting all of the mechanical parts integrated. This includes fastening the electro-mechanical stepper motors to the actuating arms and connecting the arms to the main chassis. Upon completion of this milestone we should have functioning mechanical components and a working robot skeleton that can be operated by the FPGA control board.

- **Firmware for Solution Execution**
  By the end of this milestone all FPGA control board firmware should be finalized and tested. When given a solution from Kcube, Herbert should be able to carry out the operations needed to solve the cube.
- **Herbert Test Application**
  This goal of this milestone is to create a test application that allows us to interact with the mechancial components through a software interfaces communicating through the firmware on the FPGA control board. At the end of this milestone we should be able to manually control the rotation and actuation of the mechanical arms.
- **Integration and Testing**
  Every part should be integrated together and tested for robustness and speed by the end of this milestone. If there are any errors or optimizations needed, then they are done upon completing this milestone as well. This milestone is tentatively scheduled after every other milestone is completed.

We believe that parallelizing these tasks by distributing them across our team will allow us to design this system in a time efficient manner. We hope to stay close to the timeline we present above, however the timeline is tentative, and will likely change over the course of summer and Fall semester.

IV. Required Resources

Table IV is the bill of materials (BOM) needed to realize this project. This BOM defines the main materials needed to realize the system as outlined in Fig. 1. Many of the components that we will be using are being donated to us by industry sponsors. The components being donated to us include: the stepper motors, FPGA system control board, the motor control boards, and the Chameleon USB2.0 cameras. The stepper motors, motor control boards, and the FPGA system control board are “in-house” proprietary boards developed by BioFire Defense Systems. The FPGA board embeds a Xilinx Spartan3 with a softcore Microblaze processor. The Chameleon USB2.0 cameras donated to us by Point Grey Research are 1.3 megapixel cameras with a Sony ICX445 CCD, 1/3”, 3.75 micron sensor.

As a team we will be distributing the cost of the pneumatic air cylinders, pneumatic solenoid valves, the 8 channel relay board needed to drive the valves, and any other unforseen costs that may arise. We are aware that additional costs may arise for standard industrial items such as screws and wiring. We hope to minimize these additional costs by utilizing the resources available to us as students in the ECE department at the University of Utah.

TABLE IV
Main component BOM

Part Description	**Quantity**	**Vendor**	**Vendor PN**	**Price/Unit (dollars)**
Stepper Motor	6	BioFire Defense	NA	DONATED
Pneumatic 12mmx25mm Double Action Thin Air Cylinder	6	Amico	A12030500UX0057	7.86
24V 2 Position 5 Way Pneumatic Solenoid Valve	6	Uxcell	A11102700UX0130	10.31
FPGA System Control Board	1	BioFire Defense	NA	DONATED
Motor Control Board	2	BioFire Defense	NA	DONATED
8 Channel 5V Relay Board	1	SainSmart	20-018-102	11.99
Chameleon USB2.0 Camera	2	Point Grey Research	CMLN-13S2C-CS	DONATED

V. Risk Assessment

The main risk associated with this project is the image processing. The likelihood of error in the cube capture phase is high. Capturing the cube orientation is error-prone because of changes in environmental lighting conditions. Object detection algorithms present a steep learning curve. Likewise, interfacing with camera device parameters to aid in eliminating changes in ambient lighting can further increase the complexity of obtaining the initial cube configuration. Capturing the initial cube state is a core component of this project. Without accurate cube capture, the autonomous aspect of our project is void. As a result of this, we will put a large emphasis on accurate cube capture. If adapting to various lighting conditions proves too difficult and becomes too time costly, we will design our system under ideal lighting conditions to eliminate this risk altogether.

Another high risk component of project Herbert is the chance of mechanical failure. There are various mechanical components that could cause total system failure if they were to break. If mechanical failures occur, the entire project may be jeopardized. In order to mitigate this risk, we have made sure that spare hardware parts can be obtained quickly from our industry sponsors.

VI. Summary

Project Herbert is a project that integrates various technologies and domains of engineering into one complete package. Working on this project will help expose us to a real-world application of system integration and, most importantly, teamwork. Project Herbert will be broken down into three primary parts: cube detection, solution generation, and the physical manipulation of the cube using mechanical components. Each of these will present their own set of challenges. Implementing the cube detection application will require a lot of research in the field of image processing/computer vision, especially if we attempt to operate the system under various lighting conditions. The mechanical aspect of this project will require a fundamental understanding of mechatronics and robotics. The mechanical aspect of this proejct will also pose the greatest limitations. We will have to optimize the mechanical components by reducing frictional effects so as to ensure operation timings that are within the window of time needed to solve a cube in record breaking time.

The workload associated with this project is ambitious. As a result of this, the workload will be distributed and worked on in parallel over the upcoming summer and Fall 2015 term. Working in parallel over the summer will allow us to get the bulk of the individual components implemented before Fall term, which will allow us to focus on integration, testing, and optimizations over the Fall semester. Thanks to many industry sponsors we are confident that the risks associated with our project will be mitigated in the case of component failure. This will allow our team to focus the majority of our time on design, integration, and testing.

VII. Acknowledgements

Our team would like to thank BioFire Defense LLC, Point Grey Research Inc., and Futura Industries for all the support and resources they have provided us. Your contributions are greatly appreciated.

BioFire Defense has given us the various control boards and mechanical components needed for this project. They have also given us access to various prototyping tools including high precision 3D printers and laser cutters. A special thanks goes out to BioFire engineers Logan Taylor (Mechanical), Pat Riley

(Electrical/Systems), Matt Murdock (Electrical), and David Nielsen (VP of Product Development). These individuals have provided invaluable time and knowledge to our team.

We'd also like to thank Vladimir Tucakov of Point Grey Research. He has provided our team with their Chameleon CMLN-13S2M-CS camera which we will use for precise and fast image acquisition. Vladimir has also discussed our project and ideas with his colleagues at Point Grey, and they have mentioned featuring our project in their newsletter.

We couldn't put all these components together without a nice chassis to house them all. For this, we would like to thank Futura Industries. They have helped us in the design and construction of the aluminum frame we will use to build Herbert. Another special thanks goes out to Futura's Kenton Frandsen (Mechanical/Manufacturing Engineer) who has assisted in the mechanical design of the arm and frame of our project.

References

[1] *USB 2.0 Specification* [Online]. Available: http://www.usb.org/developers/docs/usb20_docs/
[2] Herbert Kociemba. *The Two-Phase Algorithm* [Online]. Available: http://kociemba.org/cube.htm
[3] Guinness World Records. *Fastest robot to solve a Rubik's Cube* [Online]. Available: http://www.guinnessworldrecords.com/world-records/fastest-robot-to-solve-a-rubiks-cub
[4] OpenCV Developers Team. *About OpenCV* [Online]. Available: http://opencv.org/about.html
[5] OpenCV Developers Team. *Miscellaneous Image Transformations* [Online]. Available: http://docs.opencv.org/modules/imgproc/doc/miscellaneous_transformations.html#cvtcolor
[6] Wikipedia contributors. *Grayscale* [Online]. Available: http://en.wikipedia.org/w/index.php?title=Grayscale&oldid=652694198
[7] Wikipedia contributors. *Canny edge detector* [Online]. Available: http://en.wikipedia.org/w/index.php?title=Canny_edge_detector&oldid=655662708
[8] Thomas Moeslund. *Canny Edge Detection* [Online]. Available: http://www.cse.iitd.ernet.in/~pkalra/csl783/canny.pdf
[9] OpenCV Developers Team. *Feature Detection* [Online]. Available: http://docs.opencv.org/modules/imgproc/doc/feature_detection.html?highlight=canny#canny
[10] OpenCV Developers Team. *Structural Analysis and Shape Descriptors* [Online]. Available: http://docs.opencv.org/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html?highlight=findcontours#findcontours
[11] S. Suzuki and K. Abe. *Topological Structural Analysis of Digitized Binary Images by Border Following*, pp. 473-480, 1985
[12] Joseph Converse. *Basic Notation* [Online]. Available: http://astro.berkeley.edu/~converse/rubiks.php?id1=basics&id2=notation
[13] Herbert Kociemba *The Two-Phase Algorithm* [Online]. Available: http://kociemba.org/twophase.htm

Appendix

In order to solve a cube, it is standard to define the terminology and orientation layout used in Rubik's Cube theory and analysis. This section describes the basic notation that is used throughout this document.

A. Faces

A Rubik's Cube is composed of six faces: right (**R**), left (**L**), up (**U**), down (**D**), front (**F**), and back (**B**) (see Fig. 7). The exact color of each face is relative to the orientation in which you are holding the cube. For example, if you align the blue face towards you then the blue face is defined as the front face. Each face can be rotated in two different directions: *clockwise* or *counter-clockwise*. These rotations are defined as the direction of rotation when looking directly at that face.

B. Fundamental Moves

The most fundamental moves are 90-degree clock-wise rotations for each of the faces outlined above. These moves are outlined below [12]:

- **R** - Indicates a 90-degree clockwise rotation of the right face such that the side on top rotates towards the back.
- **L** - Indicates a 90-degree clockwise rotation of the left face such that the side on top rotates towards the front.

- **U** - Indicates a 90-degree clockwise rotation of the upper face such that the side in front moves to the left.
- **D** - Indicates a 90-degree clockwise rotation of the downward face such that the side in front moves to the right.
- **F** - Indicates a 90-degree clockwise rotation of the front face such that the side on top moves to the right.
- **B** - Indicates a 90-degree clockwise rotation of the back face such that the side on top moves to the left.

C. Modifiers

For each of the fundamental moves above, there are modifiers that can be appended to the move to change the rotation of the face. My example below uses **L** as the base move, but these modifiers can be applied to any of the fundamental moves.

- **L'** - Indicates a 90-degree counter-clockwise rotation of the left face such that the side on top rotates towards the back (opposite direction as that defined above).
- **L2** - Indicates a 180-degree rotation of the left face (two rotations).

D. Cubelets

A cubelet refers to a particular piece on the cube. Cubelets are categorized based on their position. There are three types of cubelets: center cubelets, edge cubelets, and corner cubelets (see Fig. 8). A center cubelet is unique. All other cubelets revolve around the center cubelets, they never move (go ahead, try and move the center piece). Edge cubelets connect two face pieces together at an edge. A corner cubelet connects three pieces together at the corner of the cube.

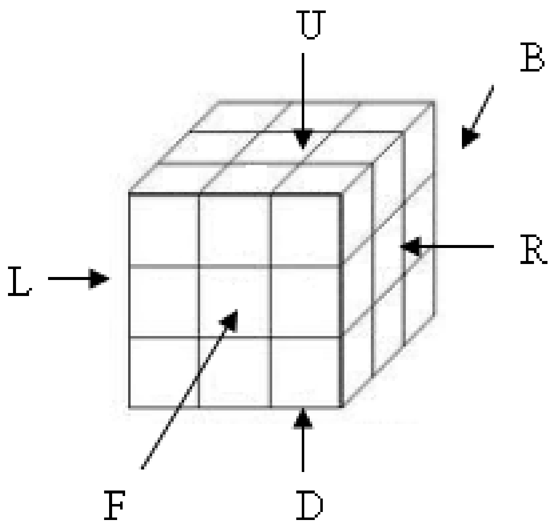

Fig. 7. Cube orientation

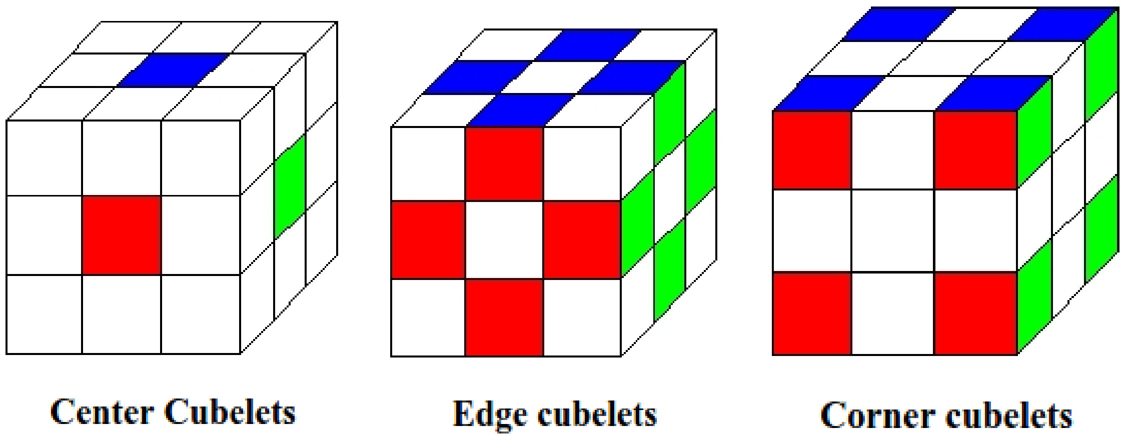

Fig. 8. Cubelet categories