VCVRC: Voice-Controlled Virtual Rubik's Cube

Andrew Bae
Stony Brook University
andrew.bae@stonybrook.edu

ABSTRACT

The Rubik's cube is a popular 3-D combination puzzle that requires the correct order of face-turns to be solved. Speedcubing, a competitive sport, has gained much popularity over the past decade. However, people with physical impairments may not have the fine motor skills necessary to manipulate the cube. While virtual cubes that use a keyboard as an alternative have been developed, not everyone has access to a keyboard or the ability to type. To address these limitations, we have created VCVRC (Voice-Controlled Virtual Rubik's Cube), a program that allows users to control the cube using voice commands. This innovation has the potential to revolutionize how people engage with Rubik's cubes. Future developments will focus on improving the voice-recognition system and implementing more customized voice commands. *Our code is available at: https://github.com/werdnabae/Voice-Controlled-Virtual-Rubiks-Cube*

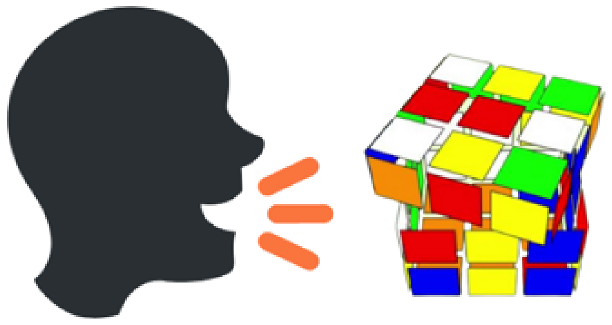


Figure 1: Our program interprets voice commands to turn the sides of a virtual Rubik's cube

1 INTRODUCTION

The Rubik's Cube is a three-dimensional combination puzzle that was invented in 1974 by Hungarian sculptor and professor of architecture Ernő Rubik. The cube consists of 26 smaller cubes that can be rotated independently on a central axis, with the goal being to align each of the cube's six faces with a single color. It has over 43 qutintillion possible combinations. The Rubik's Cube has become one of the world's most popular toys, with millions of people attempting to solve it every day.

There are various methods for individuals to engage with a Rubik's cube, such as solving it with one or two hands, utilizing virtual Rubik's cubes on a computer or phone, or providing instructions to another person in the form of the Team Blind event. However, each of these methods has inherent limitations that restrict access to the puzzle for some individuals. For instance, solving a physical or virtual Rubik's cube demands fine motor skills, which some individuals with physical impairments may not have. Furthermore, Team Blind requires the assistance of another cuber, which may not be available to everyone, impeding their ability to enjoy the activity. As a result, the challenges of current Rubik's cube interaction methods restrict the number of individuals who can engage with this enjoyable puzzle.

Previous academic research on the Rubik's cube has primarily centered on robot solving [4, 6] or cube theory [5]. Meanwhile, advancements in speedcubing methodology are typically pioneered by world-class speedcubers outside of academia. Hardware advancements, on the other hand, are usually produced by the CubicleLabs Research Division or private Chinese cube companies. However, to the best of our knowledge, there has been no research conducted on human interaction with the Rubik's cube that specifically addresses the issues we mentioned earlier.

We introduce a novel program called VCVRC, short for Voice-Controlled Virtual Rubik's Cube, that enables users to solve a virtual Rubik's cube using voice commands. Our work offers a unique solution to the limitations of current Rubik's cube interaction methods. The main contributions of our paper can be summarized as follows:

- Our program introduces a novel approach to Rubik's cube interaction by utilizing voice commands to control a virtual Rubik's cube, that addresses many of the issues that exist in current human-to-Rubik's cube interaction methods.
- Our program incorporates various optimizations that are tailored to enhance the solving speed of the Rubik's cube using voice commands.
- We conduct an analysis of the solving performance achieved through our program and identify potential areas for future improvement.

In section 2, we discuss previous human-to-Rubik's cube interaction methods. In section 3, we dive deeper into our new program which allows control of a Rubik's cube through one's voice. In section 4, we compare the solving performance of our new program compared to other solving methods and discuss future work.

2 BACKGROUND

Since the invention of the original Rubik's cube, there have been several variations of the cube and people have solved them in various ways. Currently, there are 17 official events recognized by the World Cube Association (WCA) that people can compete in. We discuss some of these events and some more unofficial events, as well as discuss their limitations in this section.

Supervised by Prof. Xiaojun Bi.

Project in Human-Computer Interaction, CSE 323, Stony Brook University
2023.

2.1 Two-Handed Solving

The CFOP solving method, also known as the Fridrich method, is the most popular and widely used way to solve the Rubik's cube, and it is the standard event in official competitions. The current world record, set by Yusheng Du from China, is an astonishing 3.47 seconds. However, this method requires precise fine-motor skills to be able to turn the cube quickly and accurately, which can be a limitation for some individuals who may not have the physical ability to turn the cube. In addition to this, high-quality Rubik's cubes can be expensive, and this cost may be a barrier for some cubers who are just starting out or who cannot afford to purchase expensive cubes.

2.2 One-Handed Solving

One-handed solving is a variation of two-handed solving where the cuber can only use one hand to manipulate the cube. It is also an official WCA event and has its own world record, currently held by Max Park from the USA with a time of 6.20 seconds. Compared to two-handed solving, this method only requires fine-motor skills in one hand, which makes it a viable option for people with limited hand functionality in the other hand. However, it still requires a functional hand, and the cost of purchasing a Rubik's cube can still be a barrier.

2.3 Virtual Rubik's Cube

A virtual Rubik's cube is a digital representation of the cube that can be controlled through various methods, such as keyboard commands or swiping on a touchscreen device. Although it is not an official event recognized by the WCA, many virtual cubes exist, including the popular one found on CStimer.net. Experienced speedcubers have been able to achieve times on par with their average two-handed solving times using the virtual cube. This could potentially solve the cost issue of physical cubes, as cubers can use their existing computer or mobile device. However, even with a virtual cube, fine motor skills are still required, whether using a keyboard or swiping on a touchscreen.

2.4 Team Blind

Team Blind is an unofficial speedcubing event that involves two people. One person is blindfolded while holding a Rubik's cube, and the other person gives verbal instructions on how to solve it. While it is not recognized by the World Cube Association (WCA), many world-class cubers such as Patrick Ponce and Rami Sbahi have achieved impressive times of around 10 seconds or less. Successful Team Blind duos often have customized and specific verbal commands for each other. THis event eliminates the need for fine motors skills as the person giving the commands does not need to turn the cube. However, this method still requires a physical cube and the presence of a second person.

3 METHOD

Our new program, VCVRC, addresses several limitations commonly associated with other Rubik's Cube solving methods. It essentially works by having the user say verbal commands to a computer. The computer then interprets these voice commands and turns the different faces of a virtual cube accordingly. Unlike traditional techniques, VCVRC doesn't necessitate fine motor skills, a physical Rubik's Cube, or a partner's help. It revolutionizes how people engage with Rubik's Cube by offering a unique solving approach. The full list of voice commands included in our program can be found in Appendix A.

3.1 Basic Voice Commands

We base our basic voice commands for VCVRC based off the universal Rubik's cube notation [2]. The basic moves are **U**p, **D**own, **R**ight, **L**eft, **F**ront, and **B**ack. Each move means to turn that side clockwise as if you were facing that side. An apostrophe (pronounced as prime) means to turn the face in the opposite direction (counterclockwise). The number 2 means to turn that face twice.

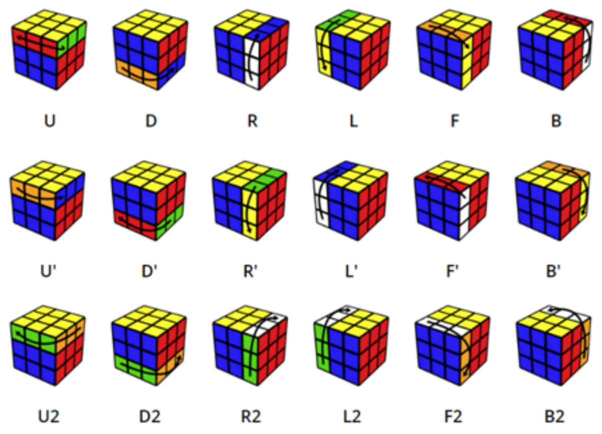


Figure 2: Standard notation for turning faces of Rubik's cube.

In addition to the fundamental moves, rotations play a critical role in Rubik's cube solving. Cube rotations are typically denoted by the letters x, y, and z. x corresponds to the R direction, y to the U direction, and z to the F direction. However, these letter notations for rotations can be challenging to remember. To make it more intuitive, we use simpler and more natural language such as "rotate up" for x, "rotate down" for x', "rotate left" for y, and "rotate right" for y'. Since the z rotation is not commonly used in 3x3 solving, we do not include commands for it in our program.

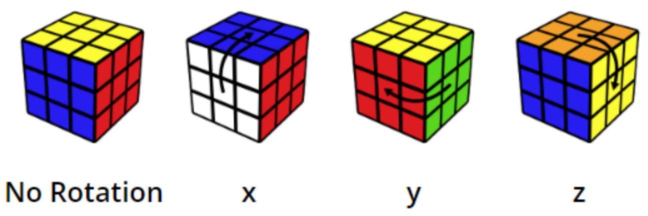


Figure 3: Standard notation for rotating a Rubik's cube.

Furthermore, we incorporate essential trigger moves like the sexy move (R U R' U') and the sledgehammer (R' F R F') into our program to enable smoother and more efficient solving. While there are additional moves such as wide moves (turning two layers at once) and slice moves (turning the middle layer), we choose not to include voice commands for them, as they are not commonly utilized in regular solving.

3.2 Optimized Commands for CFOP

The most widely used advanced speedcubing technique globally is CFOP [1]. The acronym CFOP is derived from each step of the method. C refers to Cross, which involves creating a cross pattern on the bottom layer of the cube with the edges matching each center face. F stands for First 2 Layers (F2L), which entails solving the corners of the first layer and the edges of the second layer simultaneously. O stands for Orient Last Layer (OLL), which aims to switch and orientate the pieces on the top face. Finally, P refers to Permute Last Layer (PLL), which completes the cube after the top face is solved.

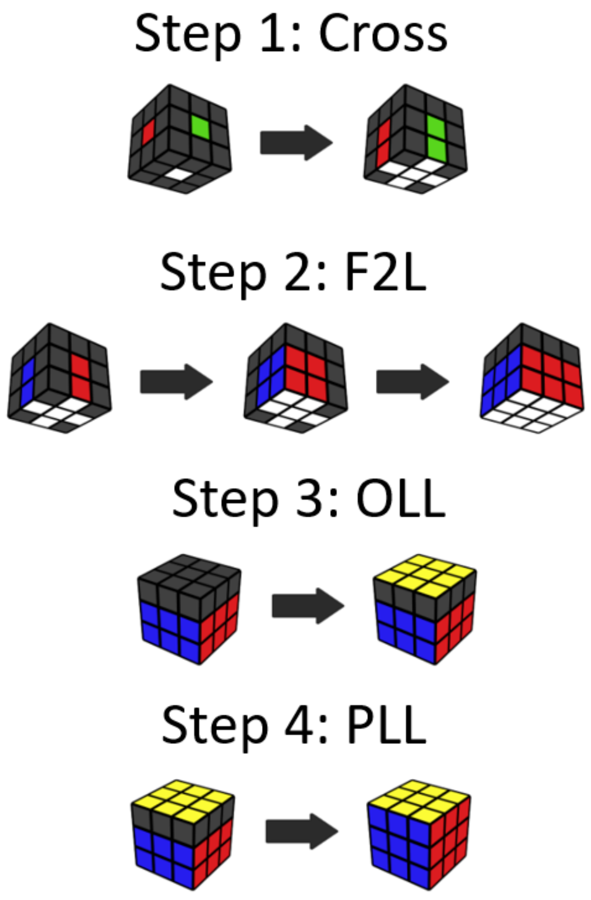
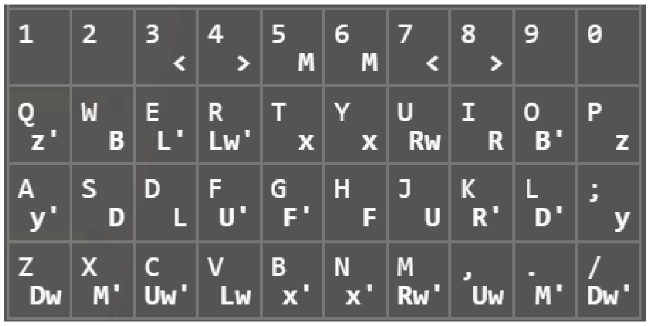


Figure 4: Steps of the CFOP method

The first two steps, cross and F2L are intuitive, meaning that they are solved mostly using cube theory knowledge. The last two steps, OLL and PLL are solved purely using algorithms. There are 57 OLL cases and 21 PLL cases in total. In order to make the OLL step more manageable for users, our program uses a 2-look approach, breaking down the step into two sub-steps. First, the user makes a cross on the top layer, and then they solve the top layer. While the PLL cases are identified by letters, the OLL cases are numbered, making it difficult to identify them verbally. Additionally, there is less standardization in OLL algorithms compared to PLL algorithms, so users may have preferences for different algorithms. Furthermore, different algorithms for the same OLL case may need to be executed from different angles, making it impossible to know which algorithm the user prefers. Therefore, we opted for the 2-look OLL approach instead of using all 57 algorithms to simplify the process. In addition, our program includes customized cases for all 21 PLL algorithms, allowing the user to simply say the name of the algorithm instead of reciting each move. For example, instead of saying "R U R' U' R' F R2 U' R' U' R U R' F'" for the T-perm PLL case, the user can simply say "t-perm".

3.3 Implementation

We use the virtual cube available at CStimer.net as a base for our program. We use Selenium [3], a Python library designed for automated testing of web applications, to input keyboard presses. These keyboard presses turn the virtual cube on CStimer. The virtual cube key mapping of the CStimer virtual cube is shown in Figure 5. The Voice commands are interpreted using Google Cloud Speech-to-Text API.



1	2	3 <	4 >	5 M	6 M	7 <	8 >	9	0
Q z'	W B	E L'	R Lw'	T x	Y x	U Rw	I R	O B'	P z
A y'	S D	D L	F U'	G F'	H F	J U	K R'	L D'	; y
Z Dw	X M'	C Uw'	V Lw	B x'	N x'	M Rw'	, Uw	. M'	/ Dw'

Figure 5: CStimer virtual cube key map

4 RESULTS AND DISCUSSION

In this section, we compare solve times using our program VCVRC and other solving methods, as well as discuss future work to further improve our program.

4.1 VCVRC Performance

To evaluate the performance of our program, we compared solving times using VCVRC with more traditional methods such as two-handed and one-handed physical solving, as well as the CStimer virtual cube. As shown in Table 1, VCVRC consistently produced lower solving times compared to the other methods. It's important to note that these results were obtained solely from testing with the author, Andrew Bae. Although Andrew performed fewer practice solves using VCVRC, he noted that his solving times were unlikely to improve significantly with additional practice. He felt that the limitations in solving times were more likely to be caused by the software itself rather than a lack of practice.

Table 1: Average Rubik's cube solve times and estimated number of practice solves done for different solving methods for the author Andrew Bae

Method	Avg Solve Time	Solves
2-handed	8 sec	50,000
1-handed	12 sec	50,000
virtual	13 sec	1000
VCVRC (our program)	5 min	10

4.2 Future Work

Our program is the first of its kind, and as expected it is far from a polished product. Further versions will hopefully improve on what we have done in this work. We point out potential improvements in this section.

Our program's most significant weakness lies in its voice recognition system. While we use Google's Text-To-Speech API, it is not suitable for this project. Although it excels at recognizing normal sentences, it struggles to identify the commands for VCVRC, which are composed of more arbitrary words. As a result, users often have to repeat a command multiple times before the voice detector picks it up. To address this issue, we propose training a separate voice recognition system specifically designed for our task. By doing so, we can ensure that the system is optimized to recognize the specific words and phrases used in speedcubing, improving its accuracy and response time. While training a new voice recognition system will require additional resources, we believe it is a necessary step to improve the usability and accessibility of our program. With a more accurate and responsive voice recognition system, users will be able to interact with the virtual cube more seamlessly, reducing solve times, and improving the overall user experience.

Customized voice controls offer another potential area for improvement in our program. While we have made several optimizations for the CFOP method, there is still room for more customization. We can draw inspiration from top-level TeamBlind solvers, who use a vast number of customized commands for all stages of the CFOP method. To improve our program's voice controls, we plan to include some of the most popular commands used by top TeamBlind solvers. Furthermore, we aim to give users the ability to create their own voice commands and personalize the program to match their solving styles. By doing so, our program will provide a more intuitive and efficient way for users to interact with the virtual cube.

While our program has shown promising results with one user, Andrew Bae, who practiced only ten solves using the program, we recognize that more extensive testing is needed. It's important to assess our program's performance with a broader range of speedcubers, including both world-class cubers and beginners, to gain a better understanding of its strengths and weaknesses. Therefore, in future work, we plan to evaluate our program with a larger sample size of speedcubers. By doing so, we hope to identify areas for improvement and make necessary changes for future versions of our program. Through these evaluations, we aim to create a tool that is useful and accessible to a wide range of speedcubers, regardless of their skill level or experience with technology.

5 CONCLUSION

We present VCVRC, a voice-controlled virtual Rubik's Cube that provides a novel way for individuals to engage with this iconic puzzle. This technology overcomes several obstacles that some people encounter when attempting to learn speedcubing, including limited fine motor skills and the high cost of purchasing a high-quality Rubik's cube. VCVRC is optimized for the CFOP solving method, incorporating command shortcuts for all PLL algorithms and 2-look OLL. To evaluate VCVRC's performance, we conducted a study with one user, Andrew Bae, who achieved faster solving times using various established methods, including physical 2-handed and 1-handed solving, as well as the virtual cube. In future versions, we plan to enhance VCVRC's voice-to-text interpretation and add more customized voice commands. Our goal is to continue improving VCVRC's performance and accessibility, so that more people can experience the satisfaction of mastering this complex puzzle.

REFERENCES

[1] [n. d.]. CFOP Speedsolving Method. https://jperm.net/3x3/cfop

[2] [n. d.]. How To Solve The 3x3 Rubik's Cube. https://jperm.net/3x3

[3] S. Nyamathulla, Dr P. Ratnababu, Nazma Sultana Shaik, and Bhagya Lakshmi N. 2021. A Review on Selenium Web Driver with Python. *Annals of the Romanian Society for Cell Biology* (June 2021), 16760–16768. https://www.annalsofrscb.ro/index.php/journal/article/view/7087

[4] OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. 2019. Solving Rubik's Cube with a Robot Hand. https://doi.org/10.48550/arXiv.1910.07113 arXiv:1910.07113 [cs, stat].

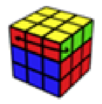
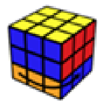
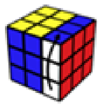
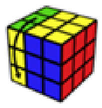
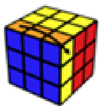
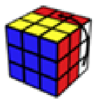
[5] Tomas Rokicki, Herbert Kociemba, Morley Davidson, and John Dethridge. 2014. The Diameter of the Rubik's Cube Group Is Twenty. *SIAM Rev.* 56, 4 (Jan. 2014), 645–670. https://doi.org/10.1137/140973499 Publisher: Society for Industrial and Applied Mathematics.

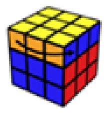
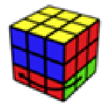
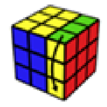
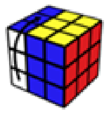
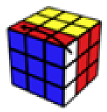
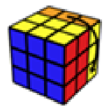
[6] Boling Yang, Patrick E. Lancaster, Siddhartha S. Srinivasa, and Joshua R. Smith. 2020. Benchmarking Robot Manipulation With the Rubik's Cube. *IEEE Robotics and Automation Letters* 5, 2 (April 2020), 2094–2099. https://doi.org/10.1109/LRA.2020.2969912 Conference Name: IEEE Robotics and Automation Letters.

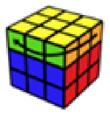
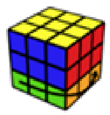
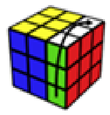
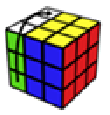
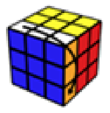
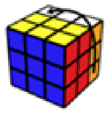
A APPENDIX - VOICE COMMANDS

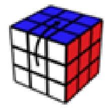
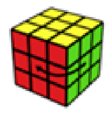
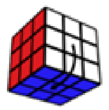
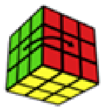
We include all voice commands included into our program in this section. To start the inspection of the cube, say "start." To quit at any point, say "quit."

BASIC COMMANDS

Move	Visual	Voice Command
U		"turn up"
D		"turn down"
R		"turn right"
L		"turn left"
F		"turn front
B		"turn back"

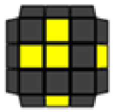
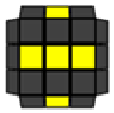
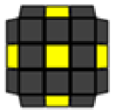
Move	Visual	Voice Command
U'		"turn up prime"
D'		"turn down prime"
R'		"turn right prime"
L'		"turn left prime"
F'		"turn front prime"
B'		"turn back prime"

Move	Visual	Voice Command
U2		"turn up two"
D2		"turn down two"
R2		"turn right two"
L2		"turn left two"
F2		"turn front two"
B2		"turn back two"

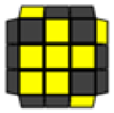
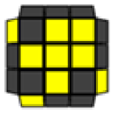
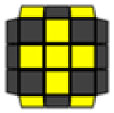
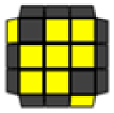
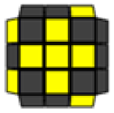
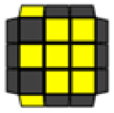
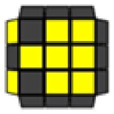
MOVE	Visual	Voice Command
X		"rotate up"
y		"rotate left"
x'		"rotate down"
y'		"rotate right"

2-look OLL

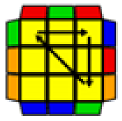
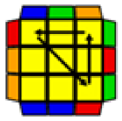
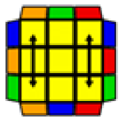
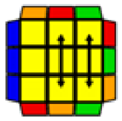
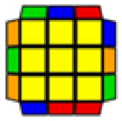
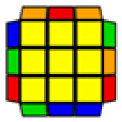
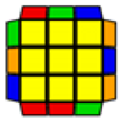
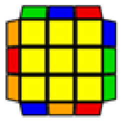
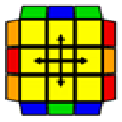
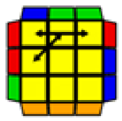
Step 1

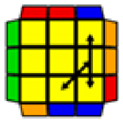
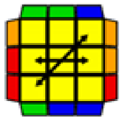
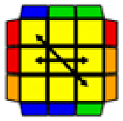
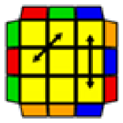
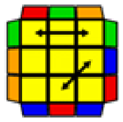
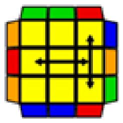
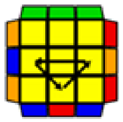
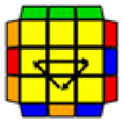
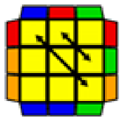
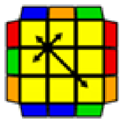
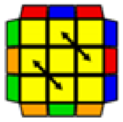
NAME	CASE	Algorithm	Voice Command
Adjacent		F U R U' R' F'	"no edges"
Bar		F R U R' U' F'	"opposite edges"
No Edges		F R U R' U' F'	"adjacent edges"

Step 2

NAME	CASE	Algorithm	Voice Command
Sune		R U R' U R U2 R'	"sune"
Antisune		R U2 R' U' R U' R'	"antisune"
H		R U2 R' U' R U R' U' R U' R'	"H"
L		F' r U R' U' r' F R	"bowtie"
Pi		R U2 R2 U' R2 U' R2 U2 R	"pi"
T		r U R' U' r' F R F'	"chameleon"
U		F R U R' U' F'	"headlights"

PLL

Name	Case	Algorithm	Command
Aa		x R' U R' D2 R U' R' D2 R2	"aa perm"
Ab		x R2 D2 R U R' D2 R U' R	"ab perm"
E		x' R U' R' D R U R' D' R U R' D R U' R' D'	"e perm"
F		R' U' F' R U R' U' R' F R2 U' R' U' R U R' U R	"f perm"
Ga		R2 U R' U R' U' R U' R2 U' D R' U R D'	"ga perm"
Gb		F' U' F R2 u R' U R U' R u' R2	"gb perm"
Gc		R2 F2 R U2 R U2 R' F R U R' U' R' F R2	"gc perm"
Gd		R U R' U' D R2 U' R U' R' U R' U R2 D'	"gd perm"
H		M2 U M2 U2 M2 U M2	"h perm"
Ja		R' U L' U2 R U' R' U2 R L	"ja perm"

Jb		R U R' F' R U R' U' R' F R2 U' R'	“jb perm”
Na		R U R' U R U R' F' R U R' U' R' F R2 U' R' U2 R U' R	“na perm”
Nb		R' U R U' R' F' U' F R U R' F R' F' R U' R	“nb perm”
Ra		R U' R' U' R U R D R' U' R D' R' U2 R'	“ra perm”
Rb		R' U2 R U2 R' F R U R' U' R' F' R2	“rb perm”
T		R U R' U' R' F R2 U' R' U' R U R' F'	“t perm”
Ua		R U' R U R U R U' R' U' R2	“ua perm”
Ub		R2 U R U R' U' R' U' R' U R'	“ub perm”
V		R' U R' U' y R' F' R2 U' R' U R' F R F	“v perm”
Y		F R U' R' U' R U R' F' R U R' U' R' F R F'	“y perm”
Z		M' U' M2 U' M2 U' M' U2 M2	“zed perm”