Computer Science and Application 计算机科学与应用, 2019, 9(2), 406-413
Published Online February 2019 in Hans. http://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2019.92045

Hans汉斯

Flowchart Design of Layer-by-Layer Method for Rubik's Cube

Runchuan Liu

High School Attached to Xi'an University, Xi'an Shaanxi
Email: liurunchuan@yeah.net

Received: Feb. 6th, 2019; accepted: Feb. 19th, 2019; published: Feb. 26th, 2019

Abstract

There are many methods to solve Rubik's cube manually, such as layered method, edge method, bridge method, angle method, CFOOP method, CFOP method, and smiling face tiger method. The exhaustive search method for computer restoring Rubik's Cube requires calculating all states of Rubik's cube. The computational load is huge and the method's efficiency is low. In this paper, a layer-by-layer algorithm for computer restoration of third-order Rubik's cube is studied. First, a mathematical model is established; second, Rubik's Cube is judged hierarchically, and then squares of each layer are restored by cyclic mode. On this basis, the flowchart for restoring each layer of Rubik's cube is designed. The simulation results prove the feasibility of the method.

Keywords

Rubik's Cube, Layer Method, Flow Chart, Mathematical Model

魔方按层求解算法设计

刘润川

西安电子科技大学附中，陕西 西安
Email: liurunchuan@yeah.net

收稿日期：2019年2月6日；录用日期：2019年2月19日；发布日期：2019年2月26日

摘　要

手动求解魔方方法很多，分层方法、棱方法、桥式方法、角方法、CFOOP方法、CFOP方法、笑面虎方法。计算机复原魔方的穷举搜索法要求获知魔方的所有状态，计算量巨大，算法效率低。本文研究了计算机复原三阶魔方的按层求解算法，首先建立数学模型，对魔方进行分层、判断继而用循环模式复原每

文章引用: 刘润川. 魔方按层求解算法设计[J]. 计算机科学与应用, 2019, 9(2): 406-413.
DOI: 10.12677/csa.2019.92045

一层方块；在此基础上设计了每一层复原算法的流程图，图例演示结果说明本文算法是可行的。

关键词

魔方，层次法，流程图，数学模型

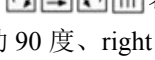
1. 引言

魔方又叫鲁比克方块，于 1974 年匈牙利布达佩斯建筑学院鲁比克教授发明。它是一个由 26 个小方块连结在一起组成的大立方体[1]。小方块分为 3 类：6 个中心块(只有一个颜色面，决定了魔方表面的颜色)、8 个角块(3 个颜色面)和 12 个边缘块(2 个颜色面)，中心块绕着本身旋转，角块和边缘块绕着中心块旋转。魔方有 6 种水平方向转动()、6 种垂直方向转动()、前表面有顺时针和逆时针 2 种转动() [2]。小方块有 3 种状态：归位、定向、复原。归位是指小方块的位置不正确，需翻转到合适位置；定向是指小方块位置已正确，但其各表面颜色指向不对；复原是指小方块的位置及其各表面颜色指向都正确。

魔方手动求解方法很多，分层方法、棱方法、桥式方法、角方法、CFOOP 方法、CFOP 方法、笑面虎方法等。其中，CFOP 方法是世界上使用最广泛的快速复原魔方方法。此方法使用了 119 个公式，分 4 个步骤复原魔方[3]。计算机求解魔方，就是用编程方法，让计算机复原任意一种打乱状态的魔方。计算机求解魔方有两种方法：穷举法和层次法。穷举搜索法[4]需要解决三个难题：如何用公式描述魔方状态、魔方状态的存储、魔方状态和魔方的操作对应。由于魔方状态千变万化，穷举搜索法求解效率极低。

本文分析计算机求解魔方的经典算法：层次法。该复原方法[5]包含七个步骤：第一步，形成上层十字，任选一面，归位四个角块的位置和颜色；第二步，把上层四个边缘块的位置和颜色归位；第三步，归位中层四个边缘块的位置和颜色；第四步，下层形成十字，把下层四个角块的位置归位；第五步，归位下层四个角块的颜色；第六步，把下层四个边缘块的位置归位；第七步，把下层四个边缘块的颜色归位。本文设计实现了这七个步骤的算法流程，并用图例详细解释每一步复原过程。

2. 建立数学模型

首先建立图 1 所示的三维坐标：每个小方块用其右前上角的坐标表示，小方块的面颜色用垂直于该面的坐标轴表示，即每个小方块用四元组表示，如图 1 上表面为蓝色的角块，表示为(3, 3, 3, z)。魔方的前后左右上下各个面分别用 front、behind、left、right、upper、bottom 表示。一连串拧转魔方的操作是通过若干次(upper 或 bottom 面)水平方向左右转动 90 度、(left 或 right 面)垂直方向上下转动 90 度、front 面顺时针或逆时针转动 90 度实现。如串操作  表示，魔方 front 面首先顺时针转动 90 度、bottom 面水平向右转动 90 度、front 面逆时针转动 90 度、right 面垂直向上转动 90 度这四个动作。

3. 求解算法框图设计

完整求解算法框架如下所示：

Input：待复原的魔方

Output：已复原的魔方
1 Stage 1：**上层小方块复原**
2 上层角块复原，算法 1 流程
3 上层边缘块复原，算法 2 流程
4 Stage 2：**中层小方块复原**
5 中层边缘块复原，算法 3 流程
6 Stage 3：**下层小方块复原**
7 下层角块归位，算法 4 流程
8 下层角块定向，算法 5 流程
9 下层边缘块归位，算法 6 流程
10 下层边缘块定向，算法 7 流程
11 Result：魔方复原

3.1. 魔方上层复原

魔方上层复原就是使这一层所有小方块的上表面颜色与中心块颜色一致，同时，同一侧面的各个小方块颜色也要一致。魔方初始状态如图 1 所示，魔方整体向左旋转 90 度，之后寻找和角块(3, 1, 3)一样的目标角块(3 个颜色面，一面为蓝色，一面为(3, 1, 3, x)的颜色)，使角块目标处于图 2(a)标示的位置(黑色编号所示位置)，执行算法 1 (图 3 所示)流程，上层角块复原(图 2(b)所示)。在本文所示的算法流程图中，例如图 3，Switch (3, 1, 3, x)表示位于 x 轴上 3 位置、y 轴上 1 位置、z 轴上 3 位置的小方格，面的颜色为垂直于 x 轴的面颜色；Case (3, 3, 1, y)表示位于 x 轴上 3 位置、y 轴上 3 位置、z 轴上 1 位置的小方格，面的颜色为垂直于 y 轴的面颜色。算法 1 中 ❶ 分支表示若图 2(a)中所示的编号①标记的方块，且面颜色为 y 轴指向的面颜色(红色)与 Switch (3, 1, 3, x)指向的面颜色一致，就执行系列拧转操作()，则(3, 3, 3)角块复原。魔方整体向右再次旋转 90 度，若此时转到(3, 3, 3)位置的角块已复原，则上层角块复原完毕。

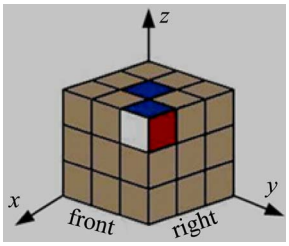
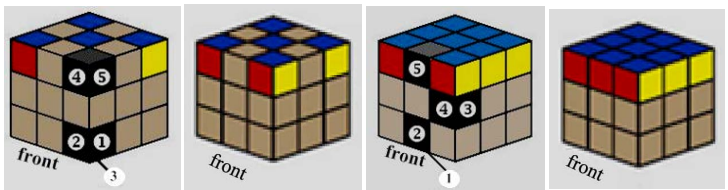


Figure 1. The coordinates of the cubes
图 1. 魔方各个小方块的坐标



(a) (b) (c) (d)

Figure 2. Initial and target states of upper layer. (a) Initial state of module 1, (b) Target state of module 1, (c) Initial state of module 2, (d) Target state of module 2
图 2. 上层复原状态图及目标图。(a) 算法 1 状态图，(b) 算法 1 目标图，(c) 算法 2 状态图，(d) 算法 2 目标图

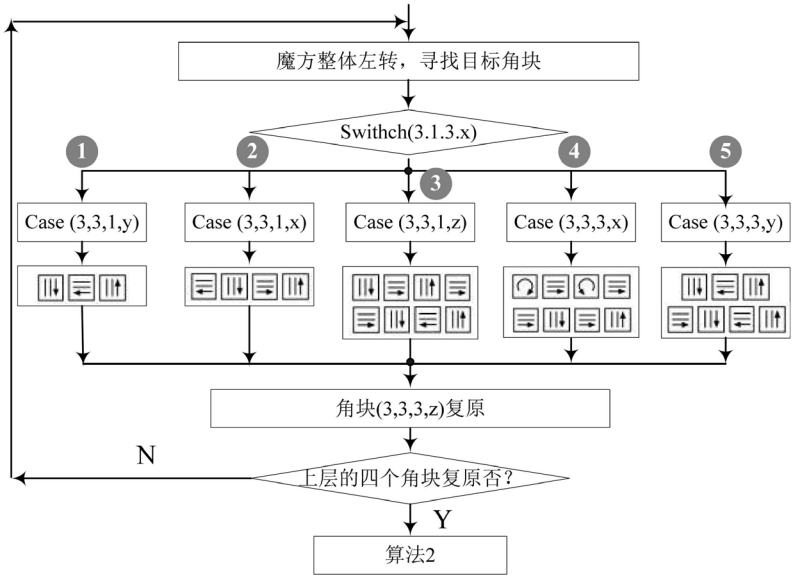


Figure 3. Flow chart of module 1
图 3. 算法 1 流程图

寻找目标边缘块(两个颜色面，一面为蓝色，一面为(3, 1, 3, x)的颜色)，旋转中层及底层，使目标边缘块位于图 2(c)标示出的位置(黑色编号所示位置)，执行算法 2 (图 4 所示)流程，上层边缘块复原，结果如图 2(d)所示。

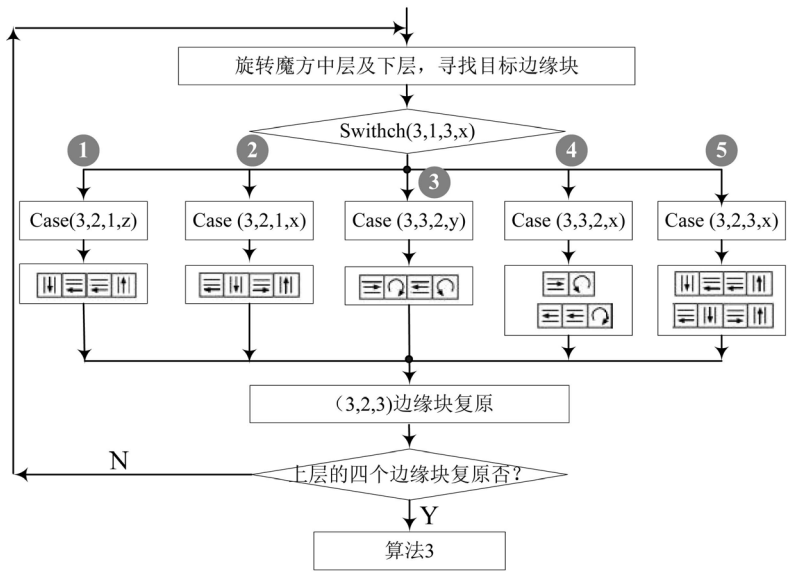
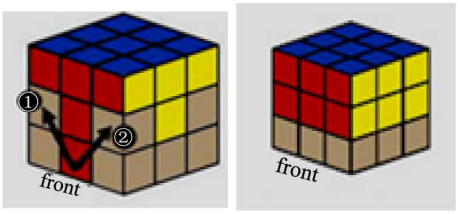


Figure 4. Flow chart of module 2
图 4. 算法 2 流程图

3.2. 魔方中层复原

在算法 2 目标图(图 3(d)所示)基础上，旋转魔方中层，使中层中间方块颜色(3, 2, 2, x)与顶层方块侧面颜色(3, 2, 3, x)一致；旋转魔方下层，得到图 5(a)状态，执行算法 3 (图 6 所示)流程，将中层边缘块复原(图 5(b)所示)。



(a) (b)

Figure 5. Initial and target states of middle layer. (a) Initial state of module 3, (b) Target state of module 3
图 5. 中层复原状态图及目标图。(a) 算法 3 状态图，(b) 算法 3 目标图

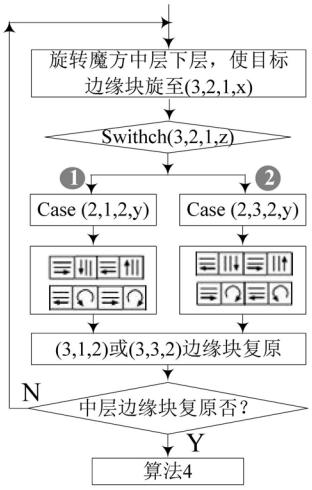


Figure 6. Flow chart of module 3
图 6. 算法 3 流程图

3.3. 魔方下层复原

第一步复原四个角块，第二步复原四个边缘块。将算法 3 的目标图翻转，下层翻转到上层，蓝色面翻到底面，白色面翻到左侧，利用算法 4 (图 7 所示)流程归位四个角块。该流程将图 8(a)中②或③编号位置上为红绿白三色的角块转到①编号位置，即(3, 3, 3)位置。

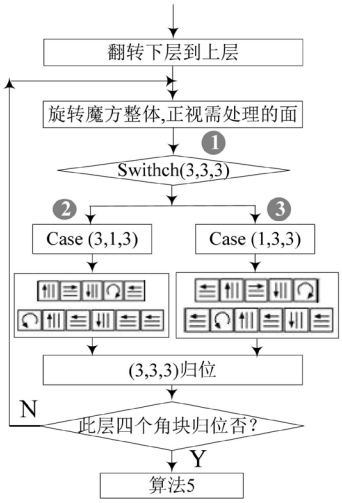
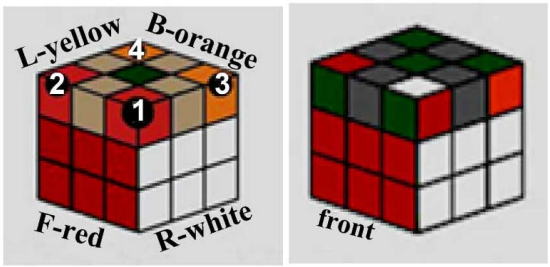


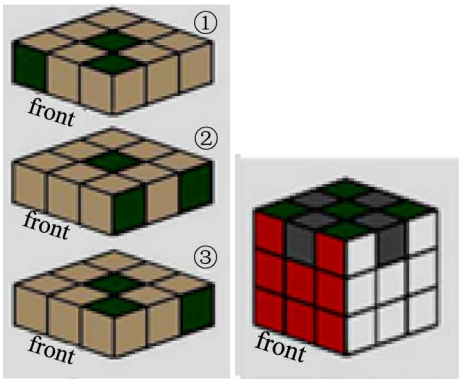
Figure 7. Flow chart of module 4
图 7. 算法 4 流程图



(a) (b)

Figure 8. Initial and target states of corner cubes of bottom layer. (a) Initial state of module 4, (b) Target state of module 4
图 8. 下层角块归位状态图及目标图。(a) 算法 4 状态图，(b) 算法 4 目标图

角块归位后，还需定向操作。无论这四个角块是否出现图 9(a)所示的三种情况，都执行算法 5 (图 10 所示)流程。执行算法 5 流程时，需要轮流操作 9(a)所示的三种情况，直至四个角块全部复原(见图 9(b) 所示)，算法 5 结束。



(a) (b)

Figure 9. Initial and target states of corner cubes of bottom layer. (a) Initial state of module 5, (b) Target state of module 5
图 9. 下层角块定向状态图及目标图。(a) 算法 5 状态图，(b) 算法 5 目标图

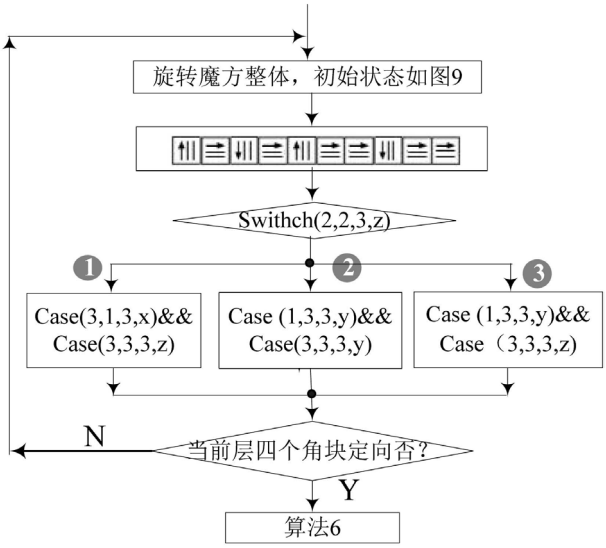
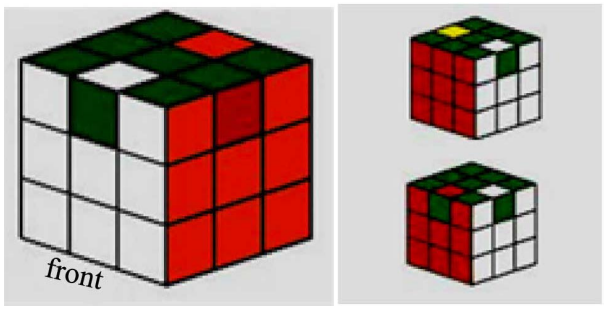


Figure 10. Flow chart of module 5
图 10. 算法 5 流程图

魔方下层复原的第二步：复原四个边缘块。若算法 5 执行完毕，目标图中没有如图 11 所示的归位边缘块，先执行一次算法 6 (图 12 所示)，之后将归位的边缘块所在的面转到前面；然后继续执行算法 6 流程，直到四个边缘块全部归位。



(a) (b)

Figure 11. Initial and target states of edge cubes of bottom layer. (a) Initial state of module 6, (b) Target state of module 6
图 11. 下层边缘块归位状态及目标图。(a) 算法 6 状态图，(b) 算法 6 目标图

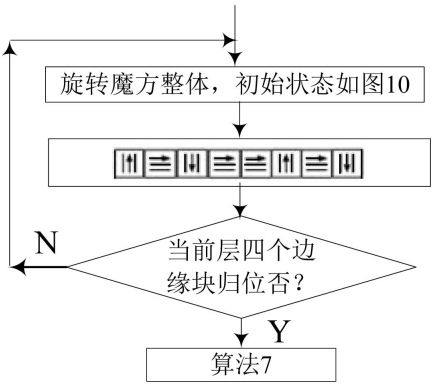


Figure 12. Flow chart of module 6
图 12. 算法 6 流程图

下面完成四个边缘块定向。算法 6 流程结束时，至少有两个边缘块定向，结果会出现图 13 所示的两种情况。执行算法 7 (图 14 所示)流程，将剩下的两个边缘块定向，至此，魔方复原成功。

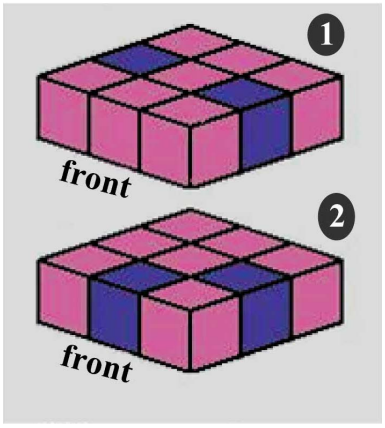


Figure 13. Initial state of edge cubes of bottom layer
图 13. 算法 7 初始状态

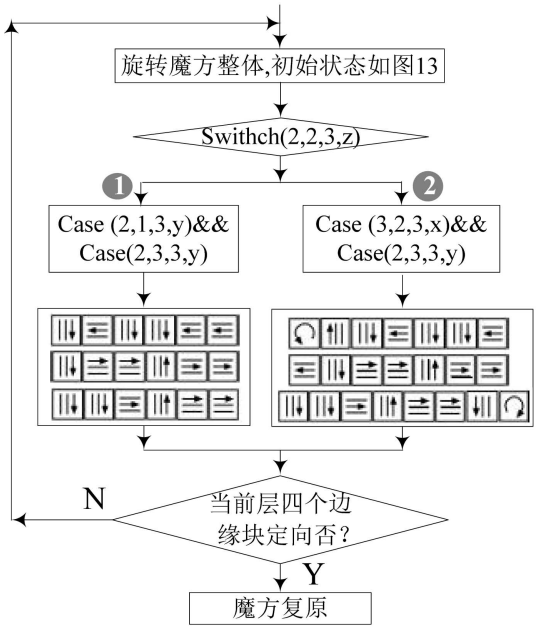


Figure 14. Flow chart of module 7
图 14. 算法 7 流程图

4. 结束语

魔方设计更好地理解了三维空间的各种运动。对于计算机求解，要求具有一定的智能判断能力；在对魔方状态进行判断后，根据算法设定，匹配最佳操作。本文研究的对象是三阶魔方，图例结果说明，本文算法适用于打乱状态的三阶魔方，算法设计是可行的。文献[6]利用 android 手机中的 DroidCam 软件和 LabVIEW 软件实现魔方复原。文献[7]从分析的角度给出了基于 Arduino 电子平台的一种魔方复原方案。而本文的研究工作旨在 Visual studio 2010 开发环境下编程实现三阶魔方复原，并结合 OpenGL 提供的强大三维绘图功能，开发一款 3D 虚拟魔方软件，处理 Windows 输入消息，通过操作键盘复原魔方。

致　谢

衷心的感谢郝航行同学，他的学士学位论文《基于 OpenGL 的魔方游戏的设计与实现》中的图片，对阐明本文算法流程图帮助很大。

参考文献

[1] 魔方[EB/OL]. https://baike.baidu.com/item/魔方/5275, 2018-11-07.

[2] 郝航行. 基于 OpenGL 的魔方游戏的设计与实现[D]: [学士学位论文]. 西安:西安理工大学, 2012.

[3] CFOP [EB/OL]. https://baike.baidu.com/item/CFOP/10504857, 2018-11-08.

[4] 孟坤, 王俊, 闫桐. 一种基于经验知识的军旗博弈算法设计与实现[J]. 智能计算机与应用, 2017, 7(2): 66-69.

[5] 层先法[EB/OL]. https://baike.baidu.com/item/层先法/5233102, 2018-11-07.

[6] 盛庆华, 杜永均, 罗飞, 李辰龙, 何凯. 基于 STM32 机械臂解魔方算法研究[J]. 实验室研究与探索, 2017, 36(4): 20-32.

[7] 李泽萱, 滕旭阳, 郑艺彬, 唐日成, 徐欢潇. 基于 Arduino 的两臂解魔方机器人——算法设计[J]. 电脑知识与技术, 2018, 14(17): 254-256.

知网检索的两种方式：

1. 打开知网页面 http://kns.cnki.net/kns/brief/result.aspx?dbPrefix=WWJD
   下拉列表框选择：[ISSN]，输入期刊 ISSN：2161-8801，即可查询
2. 打开知网首页 http://cnki.net/
   左侧“国际文献总库”进入，输入文章标题，即可查询

投稿请点击：http://www.hanspub.org/Submission.aspx
期刊邮箱：csa@hanspub.org