

# 浙江工业大学

## 本科毕业设计说明书 (论文)

(2022 届)



论文题目 基于 Arduino 的三阶魔方复原系统的实现

作者姓名 林 泽

指导教师 潘 翔

学科(专业) 计算机科学与技术 1802

所在学院 计算机科学与技术学院

提交日期 2022 年 06 月

## 摘要

魔方是一种具有八面体结构的魔术方块。人们对魔方研究的不断深入，伴随着魔方还原技术的日益进化，魔方已经成为一种易上手、门槛低的益智类玩具。但同时又考虑到人脑在记忆魔方块颜色并求解时存在一定局限性，使用手指转动魔方面相比于电机带动魔方面转动在旋转速度上存在根本性的差异等问题。针对这一问题，本论文提出一套通过软硬件协同设计的方案，实现还原效率既高，还原速度又快的三阶魔方还原系统。具体研究内容如下：

1. 设计并实现了面向魔方复原的六轴架构。传统二轴和四轴的魔方复原系统存在步骤冗余等缺点，本论文提出的架构采用增加旋转轴数的设计，提升魔方复原效率。在此基础上，本论文自主设计了与下位机、步进电机相连接的 PCB 电路板，从而能够搭载下位机并传递电机旋转信号。
2. 提出了并行复原的魔方旋转算法，在较为成熟的魔方复原算法——Kociemba 算法的基础上，将魔方复原步骤序列中互不干扰的旋转操作并行化，与现有采用串行原则依次进行旋转完成复原的方法相比，本论文的方法能有效提升魔方复原的速度。
3. 设计并实现了系统可视化界面。提供矫正魔方块颜色、验证还原可行性等功能，针对系统发生类似于颜色识别错误的偶然性偏差时能通过人工手动修正的方式有效保证系统可行性。

实验结果表明，该系统能平均在 2 秒内还原一个任意打乱状态下的魔方，超越大多数现有的魔方复原系统。

**关键词：** 三阶魔方，魔方还原，软硬件协同设计，并行优化

## Abstract

Rubik's cube is a rubik's cube with an octahedral structure. With the continuous deepening of research on the Rubik's cube and the increasing evolution of the rubik's Cube reduction technology, the Rubik's cube has become an easy-to-use, low-threshold puzzle toy. But at the same time, considering the human brain in memory cube color and solving the existence of certain limitations, the use of fingers to rotate magic compared to the motor driven magic rotation in rotation speed there are fundamental differences.

In view of this problem, this paper proposes a set of software and hardware co-design scheme to achieve high efficiency and fast restoration of the three-order Rubik's cube restoration system. The specific research contents are as follows:

1. Designed and implemented a six-axis architecture for Rubik's Cube restoration. Traditional two-axis and four-axis rubik's Cube restoration systems often have disadvantages such as redundancy of restoration steps. The architecture proposed in this paper adopts the design of increasing the number of rotation axes to improve the efficiency of rubik's cube restoration. On this basis, this paper independently designed the PCB which is connected with the lower machine and the stepper motor, so as to carry the lower machine and transmit the motor rotation signal.

2. The rubik's cube rotation of parallel restoration algorithm is proposed, and the more mature - Kociemba rubik's cube recovery algorithm is presented on the basis of the rubik's cube recovery sequence of steps in each other the rotation of the parallel operation, with the existing serial principle compared to rotate in order to complete the recovery method, the method of this paper can effectively improve the speed of the rubik's cube recovery.

3. Designed and realized the visual interface of the system. It provides functions such as correcting the color of rubik's cube and verifying the feasibility of restoration. It

can effectively ensure the feasibility of the system by manually correcting the accidental deviation similar to color recognition error.

The experimental results show that the system can restore a rubik's cube in an arbitrarily disturbed state in 2 seconds on average, which exceeds most existing rubik's cube restoration systems.

**Keywords:** rubik's 3x3 cube, rubik's cube restoration, software and hardware co-design, parallel optimization

# 目录

<b>摘要 .....</b>	<b>I</b>
<b>Abstract .....</b>	<b>II</b>
<b>第一章 绪论 .....</b>	<b>1</b>
1.1 研究背景及意义 .....	1
1.2 国内外研究现状 .....	2
1.3 本文的主要工作 .....	4
1.4 本文的创新点 .....	5
1.5 本文的组织结构 .....	6
1.6 本章小结 .....	7
<b>第二章 硬件框架的设计与搭建 .....</b>	<b>8</b>
2.1 硬件设备介绍 .....	8
2.1.1 Arduino Nano 开发板 .....	8
2.1.2 SERVO42A 闭环步进电机 .....	10
2.1.3 DS3120 伺服电机 .....	11
2.2 六轴架构设计 .....	12
2.3 PCB 电路板设计 .....	14
2.4 三维建模结果图 .....	15
2.4.1 零件仿真图 .....	15
2.4.2 装配仿真图 .....	18
2.4.3 实际搭建效果图 .....	20
2.4.4 本章小结 .....	21
<b>第三章 魔方还原算法的研发与优化 .....</b>	<b>22</b>
3.1 开发环境简介 .....	22
3.1.1 Ubuntu 20.04 操作系统 .....	22
3.1.2 PyCharm 集成开发环境 .....	23
3.1.3 PyQt5 库 .....	23
3.1.4 串口通信模块 .....	23
3.2 系统目录规划表 .....	24
3.3 系统算法流程图 .....	24
3.4 系统实现 .....	25
3.4.1 可视化界面设计 .....	25

3.5 魔方块颜色识别 .....	27
3.5.1 Kociemba 算法 .....	31
3.5.2 并行旋转优化算法 .....	34
3.6 系统界面截图 .....	35
3.6.1 初始界面 .....	35
3.6.2 颜色矫正 .....	36
3.6.3 还原可行性验证 .....	37
3.6.4 魔方块还原 .....	37
3.7 本章小结 .....	38
<b>第四章 系统测试与分析 .....</b>	<b>39</b>
4.1 魔方块颜色识别实验 .....	39
4.2 魔方复原算法对比分析 .....	39
4.3 串并行复原实验 .....	41
4.4 本章小结 .....	42
<b>第五章 总结 .....</b>	<b>43</b>
5.1 完成的工作 .....	43
5.2 存在的问题及下一步工作 .....	43
<b>参考文献 .....</b>	<b>45</b>
<b>致谢 .....</b>	<b>49</b>
<b>附录 .....</b>	<b>50</b>
附录 1 毕业设计文献综述 .....	50
附录 2 毕业设计开题报告 .....	50
附录 3 毕业设计外文翻译(中文译文与外文原文) .....	50

## 图 目 录

图 1-1 标准三阶魔方.....	1
图 2-1 Arduino Nano 实物图 .....	9
图 2-2 引脚说明图 .....	9
图 2-3 SERVO42A 电机尺寸图 .....	10
图 2-4 SERVO42A 电机实物图 .....	10
图 2-5 伺服电机.....	12
图 2-6 两臂二轴魔方机器人 .....	13
图 2-7 带坐标轴的魔方示意图.....	14
图 2-8 魔方中心块示意图与装配图.....	14
图 2-9 电路板设计图与实物图.....	15
图 2-10 铝型材仿真图 .....	16
图 2-11 滑轨仿真图.....	16
图 2-12 滑块仿真图 .....	17
图 2-13 电机滑块连接件仿真图.....	17
图 2-14 魔方托槽仿真图 .....	18
图 2-15 魔方与托槽装配仿真图 .....	19
图 2-16 系统装配仿真图(侧视图).....	19
图 2-17 系统装配仿真图(背视图).....	20
图 2-18 系统侧视图 .....	20
图 2-19 系统俯视图 .....	21
图 3-1 系统流程图 .....	25
图 3-2 可视化界面设计图.....	26
图 3-3 HSV 颜色空间示意图 .....	28
图 3-4 魔方块颜色识别流程图.....	29
图 3-5 左上方视角原图(左)与处理后图片(右) .....	30
图 3-6 右后方视角原图(左)与处理后图片(右) .....	30
图 3-7 右下方视角原图(左)与处理后图片(右) .....	31
图 3-8 右下方视角原图(左)与处理后图片(右) .....	31
图 3-9 并行优化算法流程图 .....	34

图 3-10 系统初始界面 .....	35
图 3-11 颜色矫正界面 .....	36
图 3-12 还原可行性效果图 .....	37
图 3-13 魔方块还原结果界面 .....	38
图 4-1 还原可行性效果图 .....	40
图 4-2 单位步骤耗时图 .....	42

## 表 目 录

表 2-1 主要技术参数.....	8
表 2-2 电机参数.....	11
表 3-1 系统目录规划表 .....	24
表 3-2 详细函数说明表 .....	27
表 3-3 HSV 颜色分量范围表 .....	28
表 3-4 函数清单.....	30
表 3-5 子群状态数变化表.....	32
表 4-1 颜色识别实验结果.....	39
表 4-2 颜色误识别次数表.....	39
表 4-3 串并行复原实验数据表.....	41

# 第一章 绪论

## 1.1 研究背景及意义

魔方作为凝聚了人类智慧的玩具，不仅可以培养人们的空间想象力、记忆力等，更可以锻炼手脑协同工作能力，其还原过程代表着人类三维空间想象及逻辑推理运算等方面的智力行为<sup>[1][2]</sup>。再者，如今出现以不同方式还原魔方的竞技运动，例如单手还原魔方、双手还原魔方等。可见在解魔方这个过程中，人们有各式各样的办法与招式。而标准三阶魔方可称得上是所有魔方类型中的经典。如下图 1-1 是一个标准的三阶魔方实例图。



图 1-1: 标准三阶魔方

近年来，研究人员开始考虑研究魔方还原的机器人自动化设备以追求魔方复原速度上的极致。对于魔方还原的机器人自动化设备，其技术关键是如何设计复原操作装置、颜色识别算法以及魔方复原算法来可靠地完成复原任务。这正是本论文研究所需要解决的问题。本论文借助 Arduino 嵌入式系统开发魔方还原系统<sup>[3][4]</sup>，不仅可以极大地简化开发步骤，而且能够提高还原效率。本系统经过复杂的计算与推理，寻找还原步骤简化规律，可快速建立最优的还原操作；同时依次通过颜色识别模块、还原求解模块、串口通讯模块和电机转动模块可实现对任意打乱状态下的魔方进行精准、快速地还原。

## 1.2 国内外研究现状

早在 1978 年,匈牙利的数学家们把魔方介绍给当年国际数学家代表会的专家学者,这是魔方在数学家面前的首次正式亮相,同时也引起了极大的反响,受到了人们的广泛关注。在随后的一段时间里英国便出现了历史上第一批的魔方理论研究小组。在 1981 年 3 月,魔方首次登上了《科学美国人》的封面。在魔方领域的众多概念中,魔方复原所需的最小转动次数称为“上帝之数”,也就是至少需要经过多少旋转操作能够复原一个打乱的魔方<sup>[1]</sup>。“上帝之数”的具体数值吸引了非常多的魔方研究者<sup>[5]</sup>,其中有相当一部分是数学家。数学家们用几十年的研究证明,任何打乱都可以在二十步内解决,即上帝之数为二十<sup>[6]</sup>。

近年来,魔方复原系统机器人引起了国内外研究人员的广泛兴趣。在国外,最早出现的魔方复原系统是在 2010 年世界制造者博览会上展出的名为“The Cubinator”的双臂魔方还原机器人,这款机器人是皮特-雷蒙德设计的,其工作原理是由相互垂直、呈九十度角的两个机械手臂旋转配合,通过一系列机械手臂对魔方的松紧操作,从而完成魔方复原。2011 年,斯威本科技大学的一个学生小组研制了一款名为 Ruby 的魔方复原机器人,被人们誉为“最快魔方机械手”,该魔方机器人以 10.96 s 的成绩打破当时机器人复原魔方的最快纪录。2014 年,由 ARM 的两位工程师完成的 CubeStormer3 机器人还原魔方仅需 3.253 秒,而上一代产品 CubeStormer2 需要 5.27 秒,并且当时人类复原魔方的记录则为 5.5 秒,机器人复原魔方的水平已远远领先人类的水平,该机器人的旋转部分采用的是四个与乐高机器人配对的伺服电机,机器人采用 ARM 驱动的三星 Galaxy S4 智能手机,由三星 Exynos 5 Octa 应用处理器驱动,分析立方体,并指导四个机器人手进行旋转操作,ARM9 处理器还为八块乐高 MINDSTORMS EV3 积木提供动力,执行电机排序和控制。2019 年,日本工程师 Human Controller 制作了一款自还原的魔方,该工程师把魔方内部“魔改”了一番,把原本是简单连接杆的核心改成了电子机械结构<sup>[7]</sup>,其结构包含了电机、电线、电池等复杂机械结构,使得魔方拥有了自动化的基础,魔方在被打乱的过程中,其内部芯片会存储打乱的顺序,当魔方检测到一段时间没有发生人为转动时,则会自动开始按照打乱的顺序倒序进行相反方向旋转的

操作。除此之外,随着科技水平的进步,德国工程师 Albert Beer 设计的名为“Sub 1 Reloaded”的魔方机器人是目前世界上魔方复原速度最快的机器人,能够在 0.637 秒的时间内完成任务,其主要结构为空间中相互垂直的 6 个旋转轴,该魔方机器人通过两张照片来识别魔方的色块样式,然后借助 Herbert Kociemba 发明的两阶段算法<sup>[8]</sup> 来计算出一套解决方案,最终在英飞凌处理器的指挥下,让机械臂在 1s 内完成最后的操作。相比于文献[9]中介绍的魔方复原的算法,两阶段算法具有在复原步骤上有很明显的优勢。

在国内,也有许多魔方爱好者设计了不同复原方式的魔方复原系统,例如用双臂解魔方机器人系统<sup>[10][11]</sup>、四臂魔方还原系统<sup>[12]</sup> 等,除此之外还有基于不同嵌入式平台的系统<sup>[13]</sup>,例如基于 FPGA 异构平台的魔方还原系统<sup>[14]</sup>、基于 ARM9 的魔方机器人系统<sup>[15]</sup>、基于 STM32 的解魔方系统<sup>[11][16]</sup> 等。南通理工学院的解魔方机器人<sup>[17]</sup> 可以在七十秒之内完成魔方的复原。北京理工大学珠海学院自动化学院设计的魔方还原系统采用双气动控制的方法,其魔方还原成功率可达到 97%<sup>[10]</sup>。济南大学的田田<sup>[18][19]</sup> 设计了一款同样基于气动的解魔方组合机械手,该方法通过采用带有 CMOS 图像传感器的摄像头来捕获各个魔方块的颜色图像,再通过 RGB 颜色模型完成魔方色块的颜色识别,以本地计算机为上位机,采用 Thistlethwaite 算法计算出复原步骤,最后通过控制气动机械手进行魔方复原。大连民族学院的董海洋<sup>[20]</sup> 设计了一款通过四轴还原魔方的类人机械臂机器人,以安卓操作系统的智能机作为上位机,通过摄像头完成图像采集,并计算出还原指令,通过嵌入式控制板连接舵机,再接收客户端发出的还原指令,最终串行调度相互垂直的四个机械手臂,从而执行魔方复原动作。东北大学的张雪娇<sup>[21]</sup> 设计了一款硬件部分由乐高组件组成的魔方机器人,该机器人采用能将光学影像转化为数字信号的 CCD 半导体作为图像采集模块,并使用笔记本电脑作为上位机对捕捉到的图像进行预处理与魔方块颜色识别。除此之外,国产魔方品牌 GAN 设计的全球消费级智能魔方机器人 GAN Robot 能在五秒内复原任意打乱的魔方。该机器人采用的是五轴伺服系统,由动力台、动力臂及 X 旋爪组成,四向卡位底座,固定动力臂。若要进行复原操作,则需要使用配套的魔方,并且在相应的 APP 平台上,其复原的原理是魔方中心

轴记录打乱的顺序并通过蓝牙将其顺序发送到 APP 平台上,通过 APP 平台进行还原算法的推理,得出还原的顺序及各自的转动方向后将其发送到四向卡位底座进行还原。

随着上述各种魔方还原机器人的出现,也有越来越多的各界人士包括数学家们研究魔方还原算法,追求魔方还原在速度、效率上的极致。并且由于解魔方本身具有趣味性及可观赏性,研究解魔方的人也越来越多。

### 1.3 本文的主要工作

本文主要的研究目标是基于 Arduino 单片机设计并实现一款三阶魔方的复原系统。在开发本系统的前期准备工作中需要对系统的整体硬件框架进行三维建模,并深入学习研究 Kociemba 提出的两阶段算法。系统的建模工作完成后,便需要开始系统的编码工作。通过上下位机的串口通信实现上位机的指令发送以及下位机的指令接收。在上位机编程中包含结合 QT 进行可视化界面设计等部分。本文阐述具体的研究工作主要包括以下几个方面:

(1) 对魔方系统的硬件框架建模并搭建。考虑到本系统结合软硬件协同设计,并且为了硬件框架搭建的可行、有效,因此需要借助三维建模对系统的每一个零部件进行精确设计。建模完毕后方可实际搭建该系统。

(2) 深入研究魔方复原的两阶段算法,提高魔方复原的效率。首先确定当下主流的魔方复原算法,再对比不同算法间的优缺点,选择最适合本课题系统的魔方复原算法。待确定算法后,再深入理解算法其中的数学原理及操作步骤,确定算法中不同阶段的输入输出从而进一步对算法进行微调。

(3) 上位机编程。使用 Python 语言在上位机编写程序。其中包括可视化界面设计部分,魔方块颜色识别部分,串口通信部分,人机交互部分等。在每次识别魔方块颜色后,为防止小概率识别错误的发生,另外还增加魔方块颜色纠错功能。待复原算法得出还原步骤及顺序后,再通过串口将指令发送至下位机,进而驱动电机带动魔方面旋转。

(4) 下位机编程。该部分的主要工作是将上位机发送的魔方复原解法转换成电机的转动操作。针对不同指令,电机进行不同的旋转操作。通过下位机编程能有

效、迅速地将上位机的复原解法传递至电机部分。此外在这一部分中，系统还实现了部分并行还原的算法，可进一步缩小还原操作的用时。

(5) 系统测试。在所有工作基本完成后，需要对系统进行大量测试。主要目的是发现系统中隐藏的一些难以被发现的问题。通过测试才能发现一些在实现本系统时忽略的问题。

在本课题初期的资料查阅阶段，通过阅读大量中、英文文献综述发现，从魔方被发明至今的时间里，魔方复原领域的研究者们产出了大量的成果，设计并实现了许多速度快、效率高、复杂度低的算法<sup>[22]</sup>，这些算法相较传统方法都有巨大的提升。但是结合本课题在魔方复原系统方向的实际需求，仍有以下难点等待后续工作的解决：

1) 针对不同光照环境下的魔方块颜色识别方面的性能有待提升。现有研究的普遍特点是基于摄像头对魔方颜色块进行识别，大部分现有研究并不能摆脱对摄像头的依赖，但是摄像头对图片颜色的识别又受光照等环境因素的影响。因此，如何在不同的环境因素的影响下能稳定识别魔方块的颜色是本课题的重要技术难点。

2) 魔方复原算法需要进一步改进。本课题拟采用 Kociemba 提出的两阶段算法，针对任意打乱状态的魔方都可在二十步内得到解法。现有的算法<sup>[23]</sup> 仍存在一些问题，特别是在转动操作十分单一的时候，此时该算法得到的结果并不是真正意义上的最优解。

3) 电机与魔方中心块的契合。就目前资料来看，有一部分魔方系统是采用 3D 打印技术打印出将魔方中心块与电机相连接的零部件，而也有一部分魔方系统是使用类似夹子的工具将单个魔方面夹紧进行旋转。不同的策略所导致魔方旋转的速度也大不相同。因此在电机与魔方中心块连接的设计部分，也是该方向的难点之一。

## 1.4 本文的创新点

1. 在硬件结构方面，本论文提出的基于闭环步进电机的六轴复原架构采用电机与魔方中心块一对一紧扣的设计，使得每个魔方面的旋转操作由单个电机负责，

不再出现多个旋转操作的结果最后只是旋转一个面的现象,从而极大程度上提升魔方复原效率。相比于二轴、四轴等魔方复原系统,该架构的翻动操作更加灵活迅速、更具稳定性,在还原速度上具有根本的优势。

2. 在现阶段还原步骤最少的 Kociemba 魔方复原算法的基础上提出了并行还原优化算法,将魔方复原步骤序列中存在互不干扰的对立面旋转操作并行化,既不影响魔方复原序列,也能减少魔方复原的总步骤数,可进一步缩短魔方复原的时间。测试结果表明,与串行复原相比,使用并行还原优化算法的效果十分显著。

## 1.5 本文的组织结构

本文一共分为以下四个章节,各章具体内容如下所示:

第一章简要介绍魔方复原系统的研究背景,综合对比国内外研究现状及成果的优缺点,并对现有魔方还原系统进行分析与简要归类判断,除此之外还介绍了本文的主要工作和组织架构。

第二章首先介绍在硬件层面开发本魔方复原系统所应用到的设备等相关内容。紧接着介绍了自主设计的 PCB 电路板,其目的是追求高度集成化,同时也为上、下位机通信打下硬件基础。最后列举展示在计算机中设计的三维建模仿真与实际搭建后的结果图。

第三章首先是简要介绍了在软件层面开发魔方复原系统的所涉及到的操作系统,集成开发环境以及部分重要模块,接着从数学的群论角度<sup>[24][25]</sup> 出发详细说明了两阶段算法的原理及实现步骤,再接着提出通过并行的旋转操作对还原算法进行进一步优化的想法,最后列举系统中已有的功能并附上截图加以文字说明。

第四章选取了本系统的部分模块进行对比实验。实验表明,本系统使用的魔方块颜色识别算法、复原算法及并行算法均可有效提高系统复原魔方的速度。

第五章对本篇文章的所有内容进行总结,针对本系统现有功能的不足提出问题以及列举下一步的改进工作。

## 1.6 本章小结

本章简要阐述了基于 Arduino 的三阶魔方复原系统研究的意义以及目前国内  
外本领域已有的一些研究成果，并介绍了本研究课题的主要工作以及技术难点，本  
章的最后列举了本篇论文的组织架构。

## 第二章 硬件框架的设计与搭建

本章详细描述了本课题硬件框架的设计与搭建,本模块是系统实际落地的硬件基础。本模块的主要工作包括电路设计,模型设计与搭建,轴间对齐,布线优化等。

### 2.1 硬件设备介绍

#### 2.1.1 Arduino Nano 开发板

本魔方复原系统使用 Arduino Nano 开发板作为下位机。Arduino Nano 是一款基于 ATmega328P 的开发板,与 Arduino Uno 十分类似。它与 Uno 的区别是 Nano 没有直流电压供电接口同时 Nano 通过 Mini-B USB 接口与电脑连接,并且 Nano 的尺寸更加小巧。表 2-1、图 2-1 与图 2-2 分别是 Arduino Nano 开发板的参数表、实物图和引脚说明图。

表 2-1: 主要技术参数

属性	属性值
工作电压	5 伏特
Flash Memory(闪存)	32 KB (ATmega328P)
SRAM(静态存储器)	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
模拟输入引脚	8 个
输入/输出引脚直流电流	40 毫安
输入电压	7-12 伏特
数字输入输出引脚	22 个
PWM 引脚	6 个
3.3V 引脚电流	50 毫安
长	45 mm
宽	18 mm
重	7 克
时钟频率	16 MHz

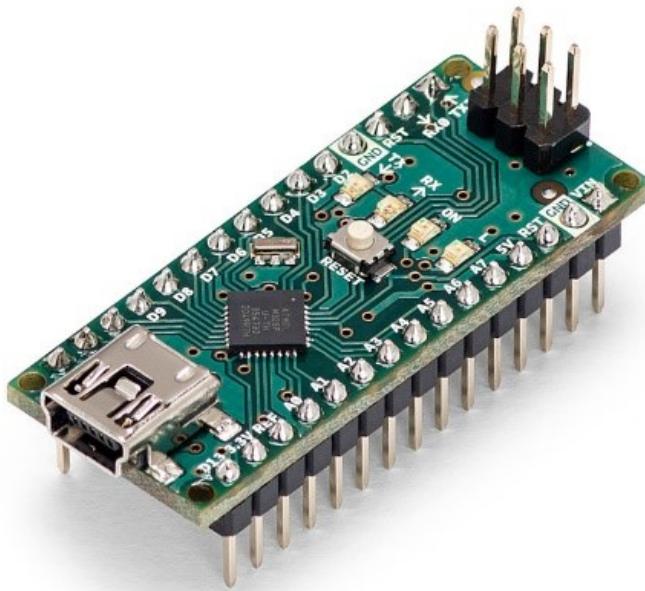


图 2-1: Arduino Nano 实物图

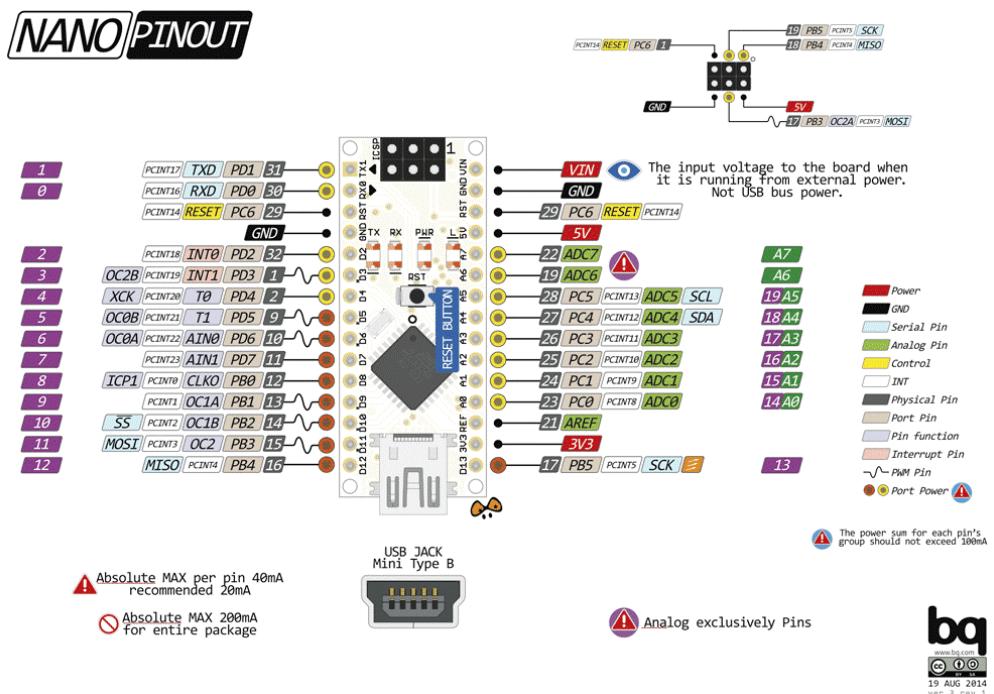


图 2-2: 引脚说明图

### 2.1.2 SERVO42A 闭环步进电机

步进电机是通过脉冲信号来进行控制，每输入一个脉冲信号，步进电机前进一步。步进电机旋转的步距角，是在电机结构的基础上等比例控制产生的，如果控制电路的细分控制不变，那么步进旋转的步距角在理论上是一个固定的角度。这正好符号了魔方面旋转需要固定角度旋转的特性。

闭环步进电机与普通步进电机的区别在于闭环步进电机的闭环控制采用位置反馈或速度反馈<sup>[26]</sup> 的方法来确定与转子位置相适应的相位转换<sup>[27]</sup>，其目的是检测旋转的每一步是否都顺利完成，换言之是为了防止丢步。闭环电机如果出现丢步的情况，自身会通过反馈回电路的信息再补上，原用于生产精度要求比较高的3D 打印机，考虑到其精度高，因此将其用于控制魔方面的旋转。

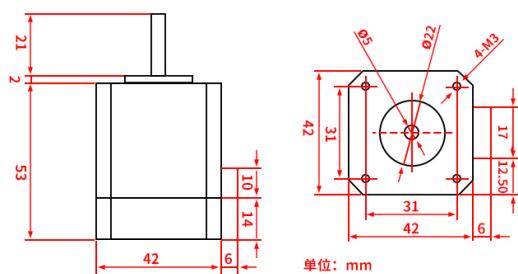


图 2-3: SERVO42A 电机尺寸图



图 2-4: SERVO42A 电机实物图

闭环控制的步进电机系统或可在具有给定精确度下跟踪和反馈时,扩大工作速度范围,或可在给定速度下提高跟踪和定位精度,或可得到极限速度指标和极限精度指标。步进电机的闭环驱动很有意义,可实现完全无过冲的位置定位,在中低速应用中超越传统电机而取得一定的应用优势,最简单的闭环控制是在常规驱动器外加一个闭环控制器,使磁极磁通与电流的相位关系保持一致,产生能带动负载转矩的电磁转矩,除此之外,更加严格的闭环控制是以控制激磁磁通与电流相位角的方式对步进电机实施矢量控制,称作功率角闭环控制方法。不论哪种闭环控制方法,都可以大幅度提高顺滑性,降低功耗和固有震动。下表 2-2 是 SERVO42A 闭环步进电机的参数表。

表 2-2: 电机参数

属性	属性值
输入电压	12V – 24V
峰值输出电流	$\pm 2A$
闭环反馈频率	6kHz
精度	$> 0.1125^\circ$
细分步数	16, 32, 64, 128, 256
电机相数	2
保持转矩	$\geq 400mN \cdot m$
转动惯量	$62.5g \cdot cm^2$
步距角	$1.8^\circ \pm 5\%$
额定电流	DC 1.0A/相

本系统选用闭环步进电机作为魔方一对一旋转的硬件部分,即一台闭环步进电机与一个魔方面相连接,同时,电机实现对魔方面的旋转控制意味着电机的旋转需要精确的位置控制。与开环步进电机相比,闭环步进电机最大的优势在于其高精度性,可稳定地进行旋转操作,具有更高的运行速度,更稳定、更光滑的转速。

### 2.1.3 DS3120 伺服电机

伺服电机是一种传统的电机,在自动控制系统中用作执行元件。该电机可以控制速度,位置精度非常准确,可以将电压信号转化为转矩和转速以驱动控制对象。

伺服电机的转子转速受输入信号控制,并能快速反应,在自动控制系统中,用

作执行元件,且具有机电时间常数小、线性度高等特性,可把所收到的电信号转换成电动机轴上的角位移或角速度输出。其主要作用是随着电压的变化控制转速均匀稳定。

伺服电机之所以精度非常准确,是因为靠脉冲来定位,当接受到一个脉冲电流,就会相应的旋转一个脉冲的对应角度,因为伺服电机本身也具有发出脉冲电流的功能,每当旋转一个角度都会发出对应数量的脉冲,和伺服电机接受的脉冲形成了呼应,或者叫闭环,因此能够精确的控制电机的转动,精确的定位可以达到0.001mm。



图 2-5: 伺服电机

与上一部分提到的闭环步进电机相比,伺服电机的精度更高,但是因为考虑到本系统需要同时考虑旋转速度,而伺服电机的旋转速度与闭环步进电机相差较大,因此并不能用于魔方面旋转。在本魔方还原系统中,伺服电机的作用在于将六个步进电机同时向位于中心的魔方移动,稳定魔方的位置,使其紧扣魔方中心块。

## 2.2 六轴架构设计

现有双臂解魔方机器人系统<sup>[10][11]</sup>、四臂魔方还原系统<sup>[12]</sup> 分别采用的是二轴、四轴的方式还原,而非六轴旋转还原的方式存在着单个旋转步骤需要分解为多个电机的旋转操作的缺点。两臂二指魔方机器人如下图 2-6 所示。

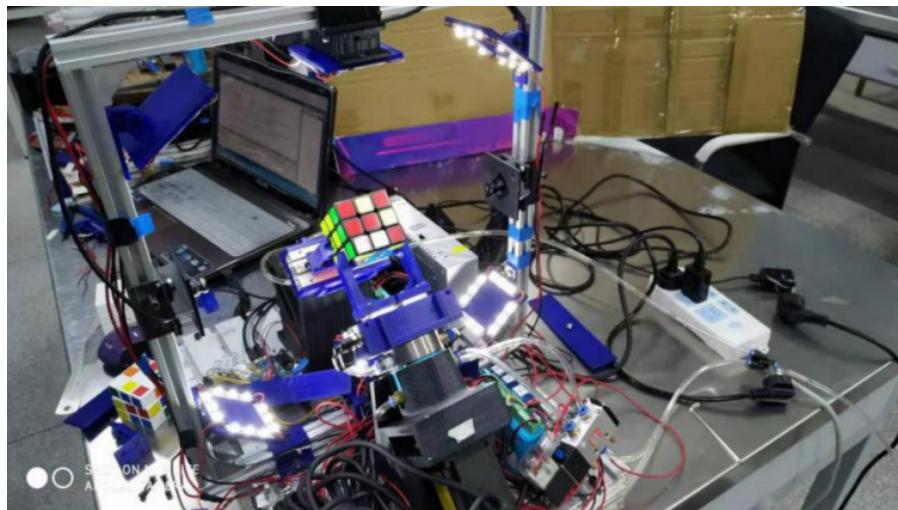


图 2-6: 两臂二轴魔方机器人

下图 2-7 是魔方建立三维坐标系后的示意图。以上述二轴魔方复原系统为例，假设该系统的两个旋转臂分别处于 X 轴正方向与 Y 轴正方向。此时若要还原魔方需要对 Z 轴正方向顺时针旋转  $90^\circ$ 。假设在二轴复原架构下，首先需要将 X 轴正方向的旋转臂顺时针旋转  $90^\circ$ ，将白色魔方面旋转至 Y 轴正方向，接着再将顺时针旋转 Y 轴正方向的旋转臂，两步操作后可复原魔方。但在本论文所设计的六轴复原架构下，只需要将位于 Z 轴正方向的电机顺时针旋转  $90^\circ$  一步操作即可完成魔方复原。

也就是说，当需要旋转除与两臂直接接触的魔方面时，都需要进行额外的旋转操作，这将会大大增加魔方复原所消耗的时间，而这些冗余的时间也正是本论文提到的六轴复原架构所能节省的复原时间。

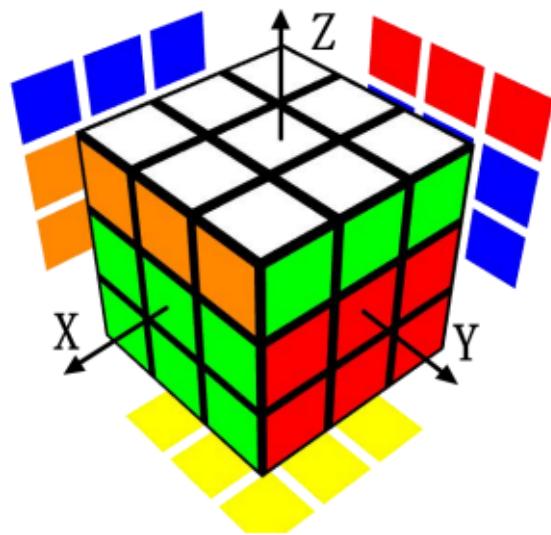


图 2-7: 带坐标轴的魔方示意图

六轴复原架构的具体实现方法是通过“爪子”将魔方中心块与电机一对一相连接，“爪子”与中心块的凹陷处紧扣后即可通过“爪子”的旋转带动魔方面的旋转。下列左图展示的是魔方中心块示意图，图中中心有四个小凹槽，分别与“爪子”四个凸起部分相匹配，右图是凹槽与“爪子”与红色魔方面相扣时的装配图。



图 2-8: 魔方中心块示意图与装配图

本论文提出的六轴复原架构将魔方的六个面均采用上述“爪子”与魔方块一对一对扣的方式相连接，该架构大大增加了系统的稳定性以及复原速度。

### 2.3 PCB 电路板设计

为了使本魔方复原系统高度集成化，因此自主设计 PCB 电路板。

PCB 电路板是将 Arduino Nano 下位机与闭环步进电机联结为一体的中间桥梁。该电路板外接 12V 适配器并搭配降压模块可同时分别向舵机提供 5V 电压, 向步进电机提供 12V 电压。

此外, 通过定义引脚以及电路板连线, 将 Arduino Nano 发送旋转信号的引脚与相应电机的引脚相连接。电路板设计图与实物图如下图 2-9 所示, 其中实物图左侧的蓝色模块为 12V 降压模块, 可将 12V 电压降至 5V 电压, 给舵机提供稳定 5V 电压。

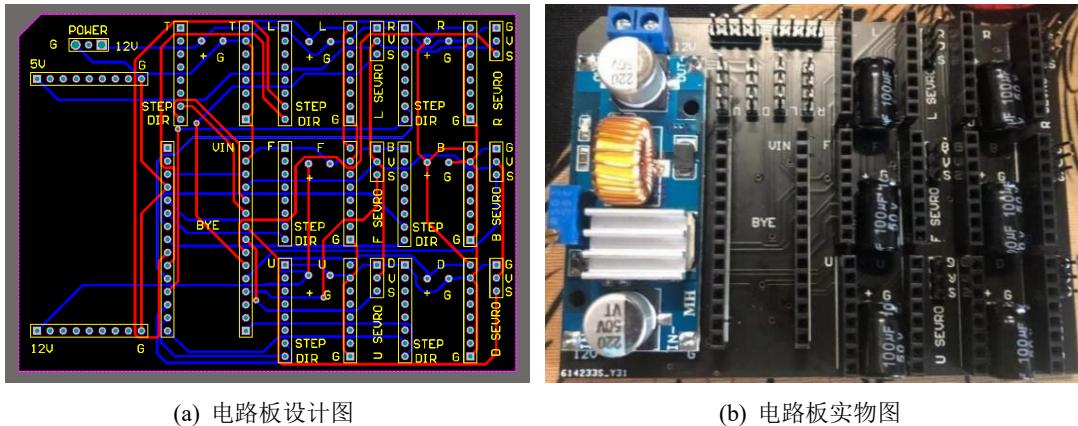


图 2-9: 电路板设计图与实物图

## 2.4 三维建模结果图

### 2.4.1 零件仿真图

本系统在三维建模阶段所用到的软件为 SolidWorks2016, 以下图 2-10 至图 2-14 为三维建模的仿真结果图。每个零件在系统中都有其独一无二、不可取代的作用。其中, 图 2-10 所示的铝型材作为系统骨架, 用于支撑整个系统; 图 2-11 所示的滑轨用于平行移动步进电机, 在伺服电机的驱动下可划动使其夹紧魔方; 图 2-12 所示的滑块用于承载闭环其上方的步进电机, 与图 2-11 的滑轨相配合; 图 2-13 为步进电机与滑块的连接件, 用于固定步进电机与滑块; 图 2-14 为魔方托槽, 用于放置本系统未运行状态时的魔方。

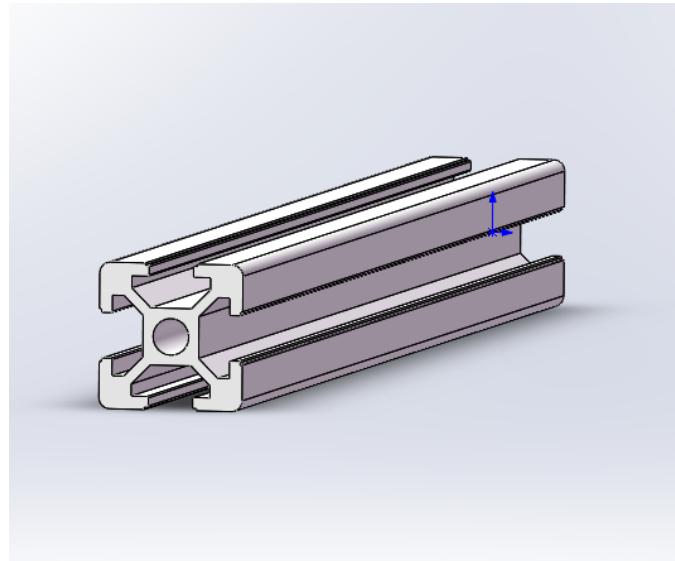


图 2-10: 铝型材仿真图

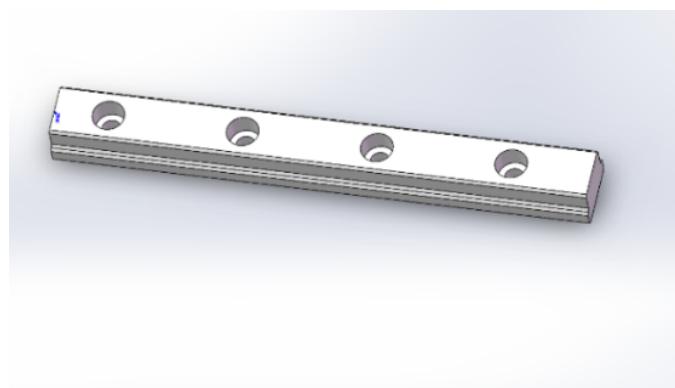


图 2-11: 滑轨仿真图

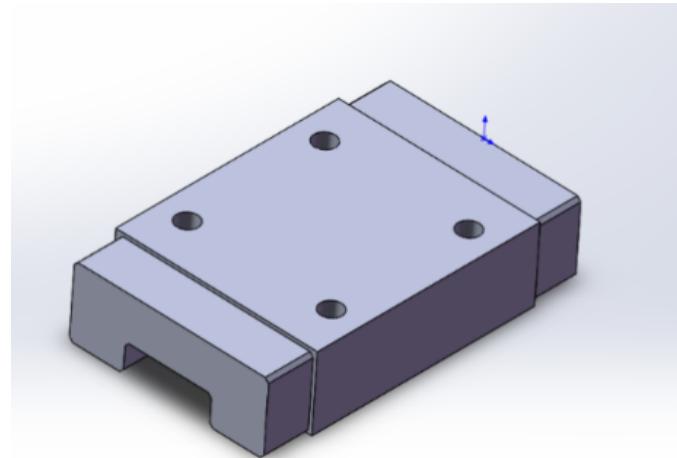


图 2-12: 滑块仿真图

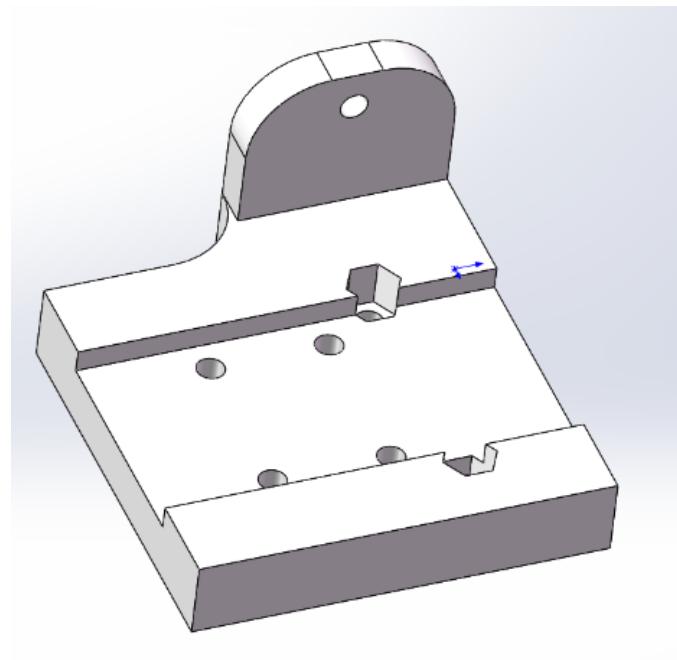


图 2-13: 电机滑块连接件仿真图

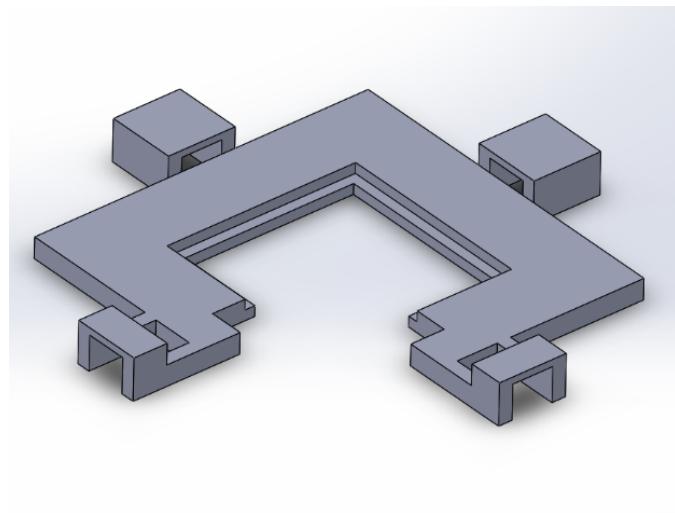


图 2-14: 魔方托槽仿真图

#### 2.4.2 装配仿真图

对于本论文提出的魔方复原系统，其装配结构对整体性能具有非常重要的影响。整个魔方还原系统需要设计多个方面的仿真结构，包括魔方与托槽装配仿真图、系统装配仿真图。

其中，下图 2-15 给出的魔方与托槽装配仿真图用于模拟仿真系统空闲时魔方放置在托槽上的状态；图 2-16 与图 2-17 的系统装配仿真图用于模拟仿真整个系统的硬件框架，在实际装配的过程中以该仿真图为参照物搭建整个系统，为整体系统搭建提供极大的便利。

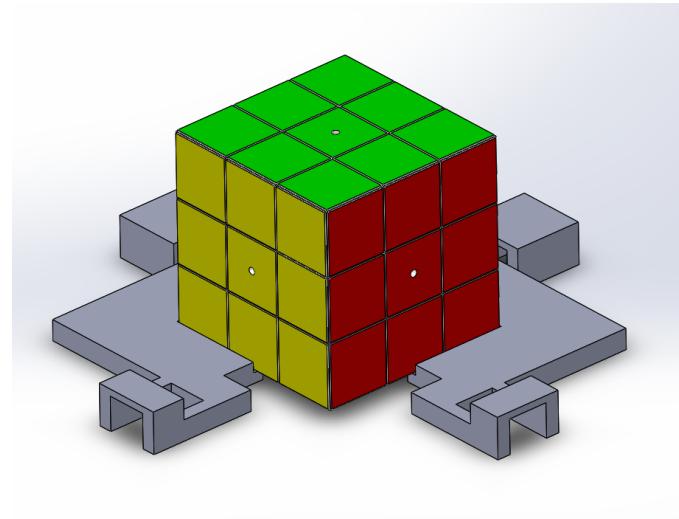


图 2-15: 魔方与托槽装配仿真图

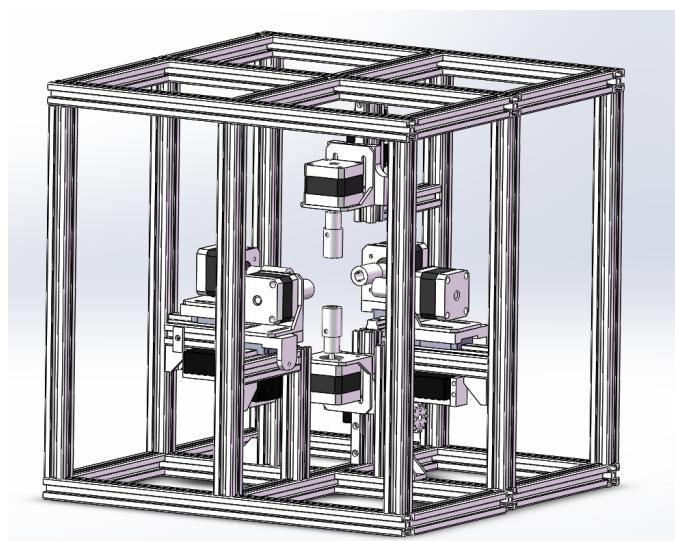


图 2-16: 系统装配仿真图(侧视图)

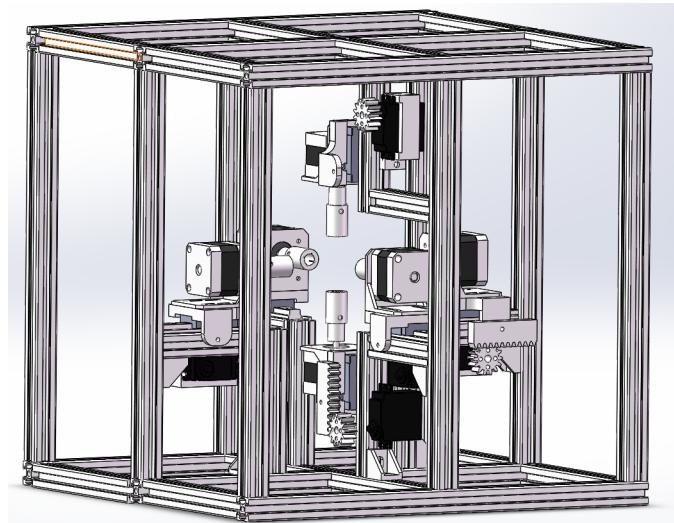


图 2-17: 系统装配仿真图(背视图)

### 2.4.3 实际搭建效果图

在本系统硬件部分的设计中,有一部分零件结合 3D 打印技术打印而成,也有一部分是使用现有相同规格的材料。图 2-18、图 2-19 为实际搭建的效果图,该框架是以上部分所示的仿真图为参照物进行系统搭建。

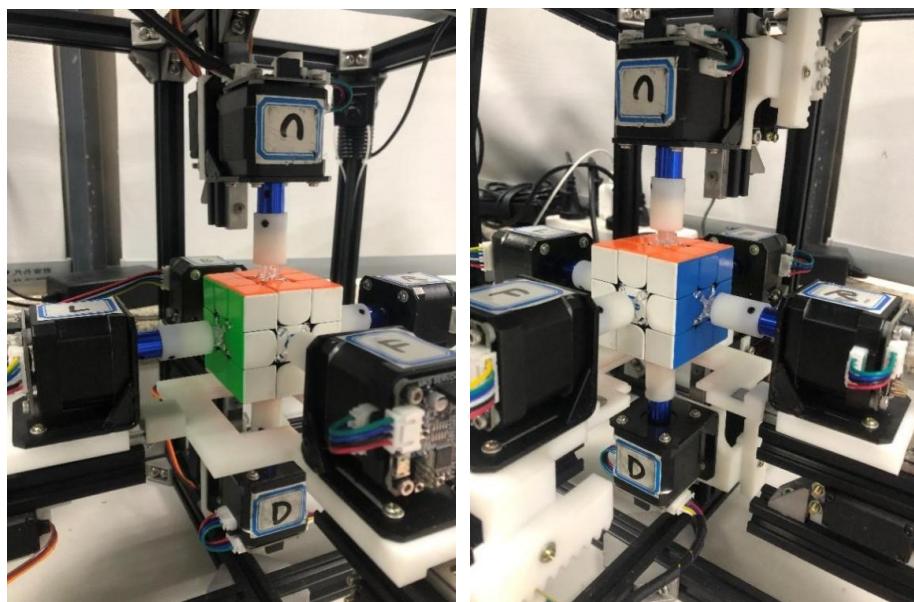


图 2-18: 系统侧视图

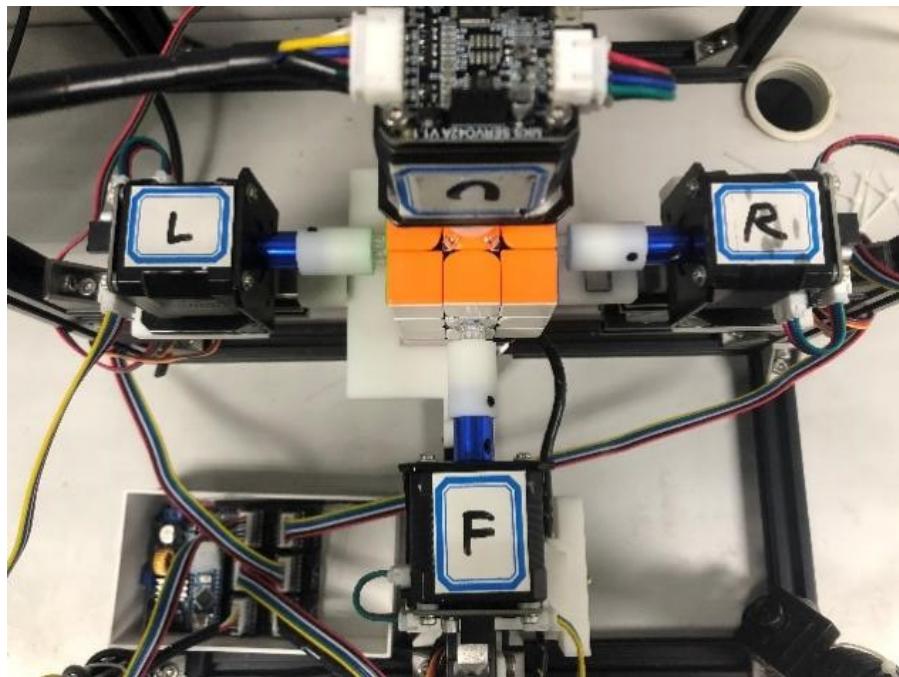


图 2-19: 系统俯视图

#### 2.4.4 本章小结

本章首先是介绍搭建本系统的硬件设备，接着简要阐述自主设计 PCB 电路板的思路，随后展示借助三维建模软件 SolidWorks 对魔方复原系统的框架整体搭建的建模结果图以及实际搭建后的效果图。本章列举本系统中硬件框架部分的内容，是本系统中最基础、最底层的一部分，为下一章系统软件算法的开发打下扎实的基础。

## 第三章 魔方还原算法的研发与优化

本章详细阐述了本课题在软件层面的工作与成果,主要体现在魔方还原算法的并行还原优化方面,本章是系统算法的核心所在。本章模块的主要工作包括两阶段算法的应用,并行还原优化等。

### 3.1 开发环境简介

本系统的开发在软件层次分为上位机和下位机两部分。其上位机程序部分是在 Ubuntu 20.04.3 LTS 的操作系统上使用 Python 语言并在 JetBrains 公司的 Pycharm 集成开发环境下进行魔方还原算法的开发;下位机程序部分首先在 Windows10 的操作系统上使用基于类 C++ 语言在 Arduino 开发环境下进行指令接收、电机旋转等相关功能的开发,待开发结束后将代码烧录至 Arduino Nano 开发板上。

#### 3.1.1 Ubuntu 20.04 操作系统

Ubuntu 20.04.3 LTS (Focal Fossa)。Ubuntu 20.04 是 Ubuntu 的第 8 个 LTS 版本,代号为”Focal Fossa”,此次版本将会获得 5 年的技术支持,直至 2025 年 4 月,本次长期支持版本包含了诸多增强的安全特性,包括可防止低层攻击和包括可防止 rootkit 和低级攻击的安全启动。Ubuntu 20.04 LTS Beta 已将大部分核心软件包和工具链升级至新版本,包括正在使用的 Linux 5.4 内核,GNOME 3.36 桌面环境,文件系统升级至 ZFS 0.8.3。

以下是 Ubuntu20.04 的特点:GNOME 3.36 是默认的桌面系统;性能大幅度提高;拥有华丽的 Yaru 主题,同时也具有黑暗模式;改进的 ZFS 支持可以获得最新的 Linux 内核 5.4(LTS);增加了对 exFAT 的支持;改进硬件和图形支持;更新软件 Python 3.8.2;Wireguard 已被移植到 Linux 内核 5.4,可以在 Ubuntu 20.04 上使用等。

### 3.1.2 PyCharm 集成开发环境

PyCharm 是一种 Python 语言的集成开发环境。该集成开发环境在先进代码分析程序的支持下,使其成为 Python 专业开发人员和刚起步人员使用的有力工具。PyCharm 是用于 Python 脚本语言的最流行的 IDE。除此之外,该集成开发环境提供了以下高级功能:代码补全、项目代码导航、代码分析、Git 可视化、代码覆盖率、包装管理、本地历史、重构等。总的来说,PyCharm 还是一款比较强大的 IDE。

### 3.1.3 PyQt5 库

Qt 框架用 C/C++ 语言编写的,PyQt 是 Qt 框架的 Python 语言实现,是最强大的 GUI 库之一。在本系统中该库主要用于制作 UI 界面。其特性有以下几点:能够跨平台运行;基于高性能的 Qt 的 GUI 控件集;对 Qt 库进行完全封装;使用信号槽机制进行通信;提供一整套种类齐全的窗口控件等<sup>[28]</sup>。

PyQt5 中的类包含很多模块与功能,其中 QtWidgets 类包含了一系列创建桌面应用的 UI 元素;QtTest 提供了测试 PyQt5 应用的工具;QtXml 包含了处理 xml 的类,提供了 SAX 和 DOM 等 API 的工具;QtSql 提供了处理数据库的工具;QtMultimedia 包含了处理多媒体的内容和调用摄像头 API 的类;QtPositioning 包含了定位的类,可以使用卫星、WiFi 甚至文本。

上述功能模块中,本系统只选用 QtWidgets 类,目的是制作一个简易美观的 UI 界面。除此之外还有一些人机交互功能更是凸显了本系统的实用性,对于系统使用者来说大大提高了使用便捷性。

### 3.1.4 串口通信模块

串口是计算机上一种非常通用的设备通信协议。而串口通信是指外设与计算机之间通过数据信号线等方式,设置波特率、停止位、校验位、字节大小等相关参数后按位进行传输数据的一种通讯方式。

本系统的串口通信模块使用 USB2.0 接口连接上位机与下位机,负责两端的通信<sup>[29]</sup>,传输电机旋转的操作指令。PySerial 模块封装了 Python 语言对串行端口的访问。

## 3.2 系统目录规划表

本系统将所有文件与目录放置在 rubik 文件夹下, 该文件夹包含 src 文件夹以及系统运行的必要配置文件, src 文件夹包含魔方块识别图片的 images 文件夹与系统运行的 Python 文件, images 文件夹里包含魔方块颜色识别前后的 marked 和 unmarked 文件夹, marked 文件夹存放识别并标记后的图片, unmarked 文件夹存放识别前的图片。系统目录规划的具体情况如下表 3-1 系统目录规划表所示:

表 3-1: 系统目录规划表

目录	名称及说明
/rubik	总目录, 此目录存放系统的所有文件, 包括环境配置文件等
/rubik/src	用于存放上位机所有运行文件, 包括业务逻辑代码等
/rubik/src/images	存放魔方块颜色识别图片的目录
/rubik/src/images/marked	存放识别并标记后的图片
/rubik/src/images/unmarked	存放识别前保存的图片

## 3.3 系统算法流程图

本系统的整体运作流程如下: 首先通过摄像头获取魔方六个面的图像, 再根据魔方块颜色识别算法对每个魔方块进行颜色识别, 如果识别完全正确得到魔方块颜色序列后并计算出可行的解法, 那么可直接执行魔方复原的操作。如果识别不完全正确, 则需要根据魔方块位置和魔方块颜色的信息手动矫正识别结果, 待矫正完所有错误颜色结果并得到可行还原解法之后即可执行魔方复原操作。该算法的流程图如下图 3-1 所示:

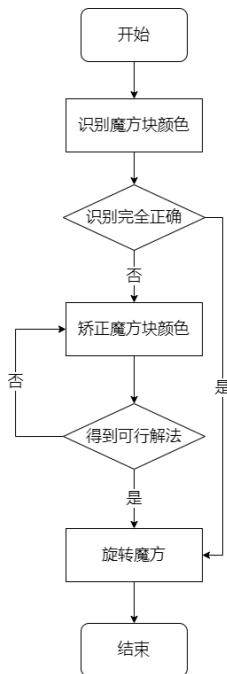


图 3-1: 系统流程图

## 3.4 系统实现

### 3.4.1 可视化界面设计

本论文中的界面设计部分首先通过 QT Creator 设计界面的大致结构,例如按钮、窗体等构件的分布,设计图如下图 3-2 所示,将设计图所生成的 ui 文件转换至 py 文件并微调参数后即是本系统的初始界面。

该设计图大致分为三个模块,界面最上方是图片显示模块,在系统运行后在该模块会显示摄像头捕捉到的四张图片;中间类似魔方展开图部分是魔方可视化模块,该模块将颜色识别后的结果与其背景颜色相结合,系统使用者可通过相应魔方块位置的背景颜色直观地了解魔方块颜色识别的结果,为后续魔方块颜色矫正过程提供了便捷。设计图界面最下方是操作模块,在该模块中提供识别、颜色矫正、还原可行性验证、魔方复原的功能,极大程度上减少魔方复原出错的可能性。

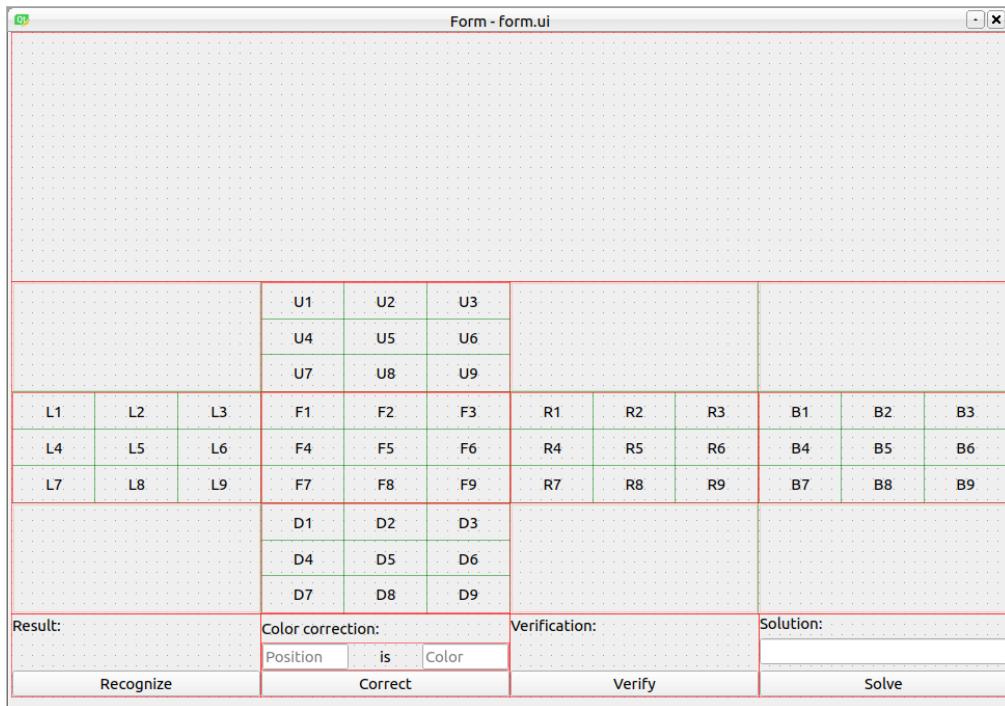


图 3-2: 可视化界面设计图

在可视化界面设计中, 不同函数有不同的功能, 其中包含初始化参数、设置构件布局及命名、配置构件颜色、设置按钮触发器等。详细的函数说明表如下表 3-2 所示:

表 3-2: 详细函数说明表

函数名称	主要功能
<code>__init__</code>	初始化参数
<code>setupUi</code>	设置构件布局及命名
<code>setStyleSheet</code>	配置构件颜色
<code>retranslateUi</code>	设置构件显示的内容
<code>click_connecting</code>	设置按钮触发器
<code>push_recognize_button</code>	魔方块颜色识别
<code>push_correct_button</code>	修正魔方块颜色结果
<code>push_verify_button</code>	验证还原可行性
<code>push_solve_button</code>	复原魔方
<code>solve_rubik</code>	求解魔方复原步骤
<code>rec_color</code>	魔方块颜色划分
<code>setLabel</code>	设置可视化界面中魔方块颜色
<code>mark_img</code>	标记识别结果
<code>communicate</code>	串口通信
<code>push_solve_button</code>	复原魔方
<code>solve_rubik</code>	求解魔方复原步骤

### 3.5 魔方块颜色识别

本系统的魔方块颜色识别模块基于 OpenCV 库,采用 HSV 颜色模型对魔方块进行颜色的判断。与 RGB 颜色模型相比,HSV 颜色模型对亮度并不敏感,反而是 RGB 颜色空间的三个分量都与亮度密切相关,即只要亮度改变,三个分量都会随之相应地改变,因此使用 RGB 颜色模型在某种程度上无法彻底摆脱环境亮度对颜色识别的影响。下图 3-3 表示的 HSV 颜色空间,圆柱体的横截面可以看作是一个极坐标系,其中色调用极坐标的极角表示,饱和度用极坐标的极轴长度表示,明度用圆柱中轴的高度表示。

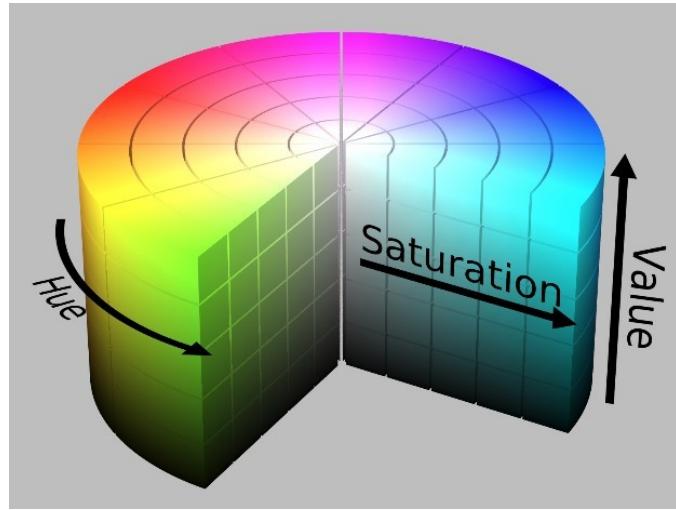


图 3-3: HSV 颜色空间示意图

HSV 颜色空间比 RGB 颜色空间更容易跟踪某种颜色的物体,因此 HSV 颜色空间常用于分割指定颜色的物体。考虑到自然环境下获取的图像容易受自然光照、遮挡和阴影等情况的影响,因此采用 HSV 颜色模型对魔方块进行颜色识别。文献[30][31][32]是在 HSV 颜色空间下进行图像检索,例如电影中的镜头检测等,证明该颜色空间也可用于图像处理。文献[33]是基于 HSV 的机器人船舶导航系统,并且在低光照的情况下也可以很好的进行船舶导航。因此,周围环境光照的亮度强弱在 HSV 颜色空间下对系统的整体影响较小。

表 3-3: HSV 颜色分量范围表

	黑	灰	白	红	橙	黄	绿	青	蓝	紫	
$H_{min}$	0	0	0	0	156	11	26	35	78	100	125
$H_{max}$	180	180	180	10	180	25	34	77	99	124	155
$S_{min}$	0	0	0	43	43	43	43	43	43	43	43
$S_{max}$	255	43	30	255	255	255	255	255	255	255	255
$V_{min}$	0	46	221	46	46	46	46	46	46	46	46
$V_{max}$	46	220	255	255	255	255	255	255	255	255	255

同时,考虑到本系统的摄像头角度及位置固定,因此在判断每个魔方块的颜色时并不需要做定位魔方块等工作,只需要人为标定每张图像中不同魔方块的固定范围即可。上表是 HSV 颜色分量范围表,对图像在 HSV 颜色空间的变量范围判断时,先是计算固定范围内像素点的分量平均值,接着判断其平均值在哪个范围内

即可粗略判断颜色,最后根据不同的运行环境进行多次实验测试并修改具体分量边界值参数。该魔方块颜色识别的算法流程图如下图 3-4 所示:

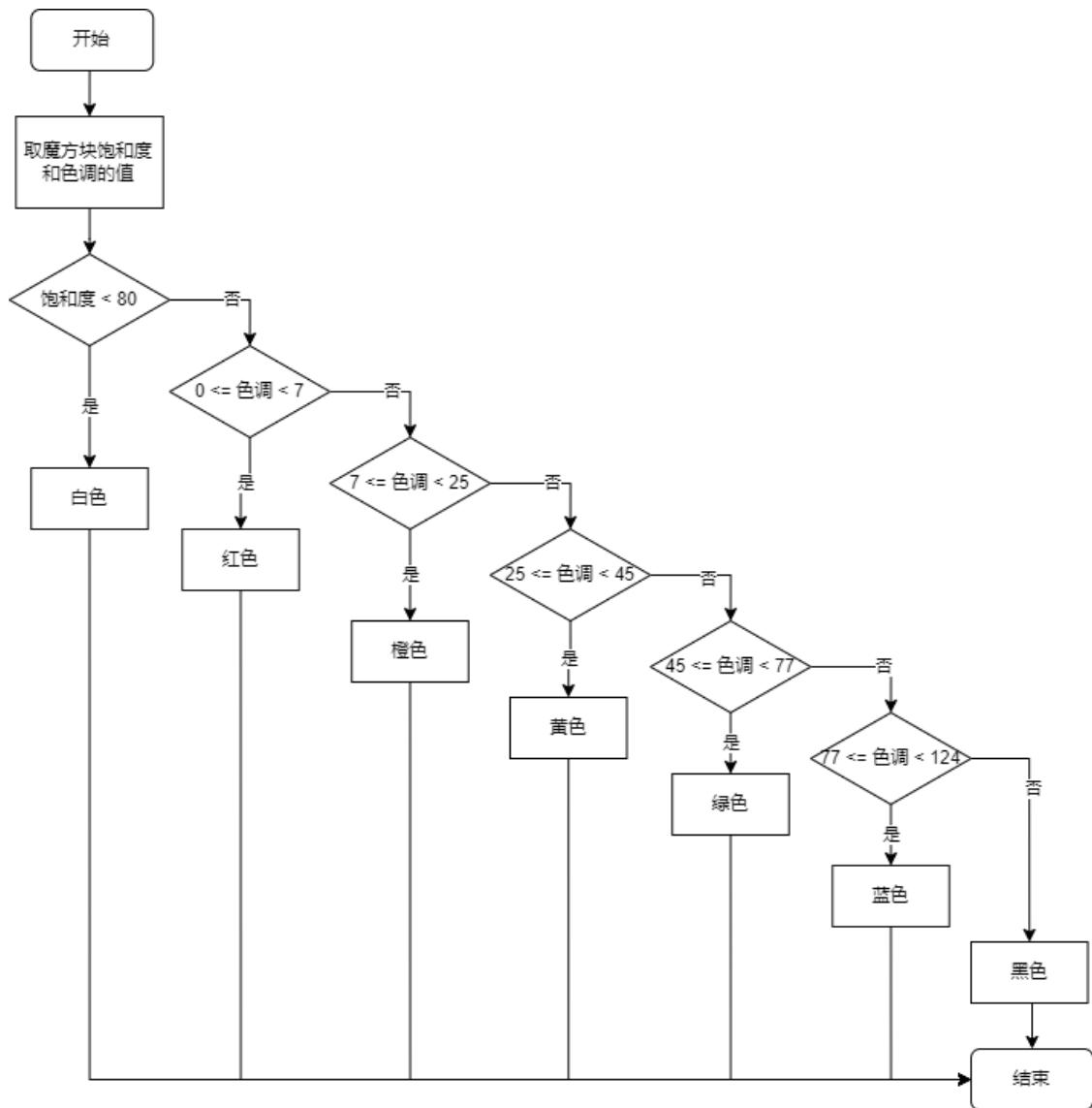


图 3-4: 魔方块颜色识别流程图

在魔方块颜色识别中,包括捕捉魔方面图像、计算 HSV 平均值及颜色模型转换等函数,详细函数说明如下表 3-4 所示:

表 3-4: 函数清单

函数名称	主要功能
catch_frames	捕捉魔方面图像
cal_hsv	计算 HSV 各项平均值
detect	颜色模型转换
print_hsv	输出 HSV 颜色模型结果

下列四组图展示的是不同视角的摄像头处理前后的图片。每组右侧图片中的绿色矩形代表的是每个魔方块的像素范围，下列四组图片中，一共有 48 个绿色矩形，对应六个面的八个魔方块。

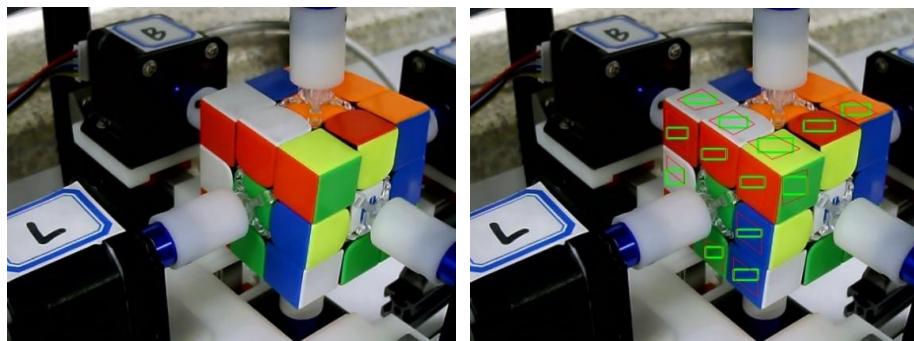


图 3-5: 左上方视角原图(左)与处理后图片(右)

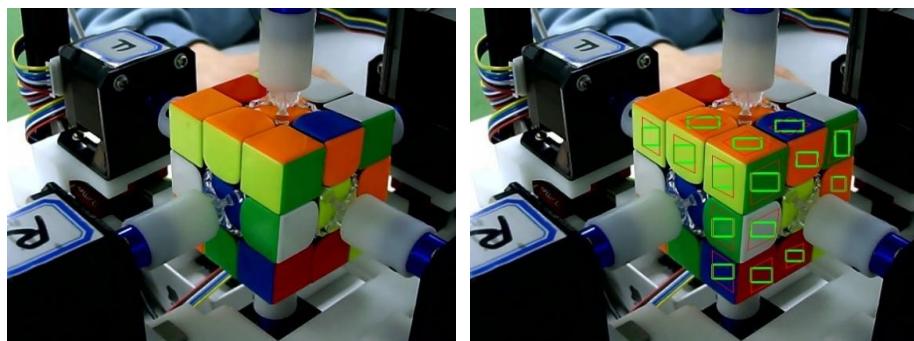


图 3-6: 右后方视角原图(左)与处理后图片(右)

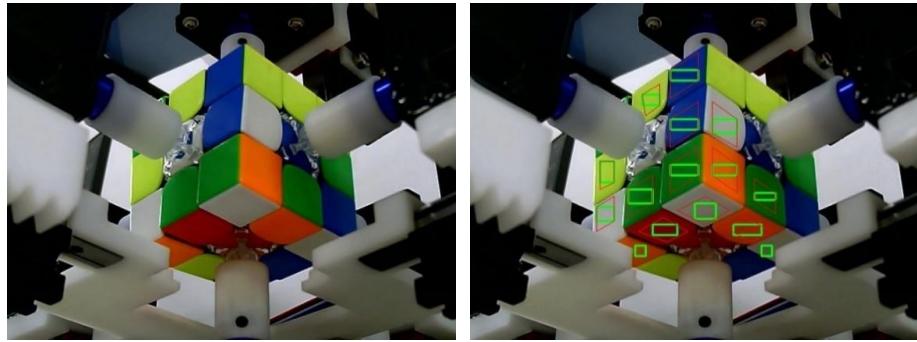


图 3-7: 右下方视角原图(左)与处理后图片(右)

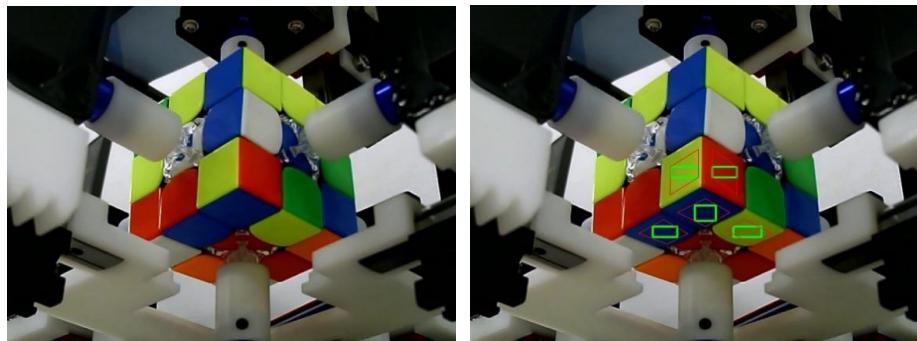


图 3-8: 右下方视角原图(左)与处理后图片(右)

### 3.5.1 Kociemba 算法

Kociemba 算法, 即两阶段算法, 是由 Thistlethwaite 算法演变而来的。该算法采用降解子群<sup>[34]</sup>的核心思想将魔方复原分为四个步骤:

$$G0 = \langle U, D, L, R, F, B \rangle, \quad (3-1)$$

$$G1 = \langle U, D, L, R, F2, B2 \rangle, \quad (3-2)$$

$$G2 = \langle U, D, L2, R2, F2, B2 \rangle, \quad (3-3)$$

$$G3 = \langle U2, D2, L2, R2, F2, B2 \rangle, \quad (3-4)$$

$$G4 = \langle I \rangle \text{ (还原态).} \quad (3-5)$$

其中 U,D,L,R,F,B 分别代表顶面、底面、左面、右面、前面或后面旋转  $90^\circ$ , 如果字母后面紧跟数字 2 , 则代表旋转  $180^\circ$ , 例如 U2 意为顶面旋转  $180^\circ$ 。

以公式 3-2 为例, G1 子群代表的是由魔方顶面、底面、左面或右面旋转  $90^\circ$  与前面或后面旋转  $180^\circ$  任意组合后可还原魔方的魔方状态。换言之, 当 G1 子群状态的魔方复原过程中, 不允许出现魔方前面、后面旋转  $90^\circ$  的情况。

上述各个子群降解过程中, 其子群状态数变化如下表所示, 影响因子意为降解至当前子群状态数缩小的倍数。

表 3-5: 子群状态数变化表

群	状态数	影响因子
$G0 = \langle U, D, L, R, F, B \rangle$	$4.33 \times 10^{19}$	/
$G1 = \langle U, D, L, R, F2, B2 \rangle$	$2.1110^{16}$	$2048(2^{11})$
$G2 = \langle U, D, L2, R2, F2, B2 \rangle$	$1.9510^{10}$	$1082565 \left( \frac{12!}{8! \times 4! \times 3^7} \right)$
$G3 = \langle U2, D2, L2, R2, F2, B2 \rangle$	$6.6310^5$	$29400 \left( \frac{8!}{4! \times 4!} \times 2 \times 3 \right)$
$G4 = \langle I \rangle$ (还原态)	1	$663552 \left( \frac{4!^5}{12} \right)$

而 Kociemba 算法是分别将上述 Thistlethwaite 算法的前两个阶段合并为一个阶段, 后两个阶段合并为另外一个阶段, 在前后两个阶段合并的过程中都使用了在 A\* 算法基础上加入迭代加深思想的 IDA\* 搜索算法<sup>[35][36]</sup>。迭代加深<sup>[37]</sup>的操作可以保证搜索树上面的任何一个节点都可以获得精确的启发函数值。

本课题研究的算法内容是搭建借助两阶段算法应用至本魔方复原系统并采用部分并行的方式优化其还原步骤。

第一阶段: 如果在拧魔方的时候不使用  $\{R, R', L, L', F, F', B, B'\}$ , 那么会生成一个状态子集。这个子集我们用  $G1 = \langle U, D, R2, L2, F2, B2 \rangle$  表示。在这个子集里面, 角块和边块的方向是不会改变的, 并且这点可以在盲拧中体会到。那就是说, 对于一个块(边块或角块)而言, 其方向是不会改变的。而魔方顶面与底面之间夹心的那四块边块仍然会保留在魔方顶面与底面之间。自然, 魔方顶面与底面的边角块仍然会处在魔方顶面与底面。在第一阶段的搜索中, 本算法会查找能把一个打乱的魔方变成 G1 状态的步法。那就是说, 做完该步后, 整个魔方边/角块方向都被纠正。在这个抽象的空间里, 移动魔方一步会把代表魔方状态的三维组  $(x, y, z)$

变成  $(x', y', z')$ , 所有 G1 状态的魔方都拥有相同的三位组  $(x0, y0, z0)$ , 而这就是第一阶段搜索的目标。在 Cube Explorer 2 中, 它会给出需求解的魔方确切的达到 G1 状态的最少步数。这种启发式的算法可以在产生解法的时候提前剪枝, 不需要等待一段非常非常长的时间来等结果。这种启发式的算法 h1 使用的是基于内存的查表方法<sup>[38]</sup>, 最多允许提前 12 步做出判断剪枝。

第二阶段: 在该阶段搜索中, 会使用 G1 步法  $\{U, D, R2, L2, F2, B2\}$  来复原魔方。实际上就是复原八个角块, 魔方顶面与底面的八个边块和魔方顶面与底面夹心的那四块的位置。启发式函数  $h2(a, b, c)$  只对复原六面的步法长度内的情况做出评估, 因为 G1 子集里面的情况实在太多了。本算法不会在找到上面的解法后停止, 而是会继续的在第一阶段的结果基础上继续展开第二阶段的搜索。举一个例子, 如果上面的解法找到第 1 阶段需要 10 步, 第 2 阶段需要 12 步, 但后面搜索的结果可能是第 1 阶段 11 步, 而第 2 阶段变成了 5 步。第 1 阶段的步法长度增加了, 但第 2 阶段的步法减少了。如果第 2 阶段的步法长度减少到 0。那么这个解法是优化完毕的, 算法结束。但是当前的两阶段搜索算法并不能在所有的情况下都找到最优解, 在这种情况下我们必须倒回头, 继续做一次二阶段搜索。这会增加相当多大的时间。

在该两阶段算法中, A\* 算法的启发函数进行初始化操作的同时会生成一个映射表, 然后以魔方的原始状态为起点进行若干次操作, 组成一颗操作树, 显而易见, 其根节点为魔方的原始状态。遍历操作树上面的所有节点, 也就是每一个状态, 从表上可以映射为一个启发函数值, 其值的大小就为节点的深度。因为此时魔方的状态就是魔方在原始状态下通过其树的深度值次数的操作结果, 经过相反的操作即可还原魔方。

同时, 操作树上每个节点向上搜索必然存在状态相同的不同节点, 因为启发函数的值应该为最小值才可以保证精度, 因此, 我们在遍历操作树的时候, 使用广度优先搜索, 保证浅层的节点先被记录, 而深层同状态的节点因为已经存在映射关系而会被忽略。

得到映射表之后, 在调用启发函数的时候, 根据当前的魔方状态, 经过查表可

以获得启发函数的值, 使用 A\* 算法判断魔方的下一个判断即可, 同时因为迭代加深的缘故, 节点的判断函数都是准确的, 而且不需要记录对未搜索的可达的节点。

本系统使用基于 Python 语言编程的 Kociemba 算法, 以顶面、右面、前面、底面、左面与背面的顺序依次输入其颜色所代表的方位, 即可得到魔方复原的步骤。

### 3.5.2 并行旋转优化算法

由魔方还原算法得出魔方求解步骤后, 上位机将还原指令通过 USB 串口发送至下位机, 下位机接收到其指令后, 对该指令进行并行优化处理, 根据不同情况执行不同旋转步骤。该算法的算法流程图如下图 3-9 所示:

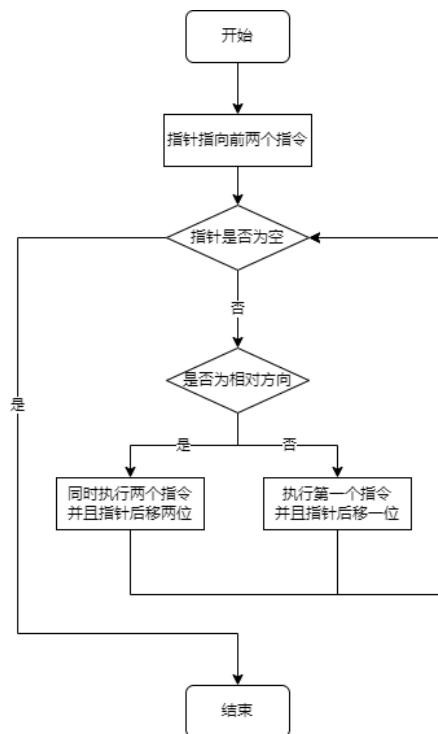


图 3-9: 并行优化算法流程图

该算法流程如下所示:

1. 新建两个指针指向还原指令中的前两个指令。
2. 判断指针是否为空。若为空, 结束该算法; 若不为空, 跳转至步骤 3。
3. 判断指针指向的两个指令是否控制相对方向的旋转, 即 U 与 D、L 与 R、F 与 B。如果是, 跳转至步骤 4, 如果不是, 跳转至步骤 5。

4. 同时执行两个指令并使指针后移两位。跳转至步骤 2。
5. 执行第一个指针指向的指令并使指针后移一位。跳转至步骤 2。

## 3.6 系统界面截图

### 3.6.1 初始界面

本系统的整体界面是由图 3-2 的整体设计图转化而来的。

本系统的界面一共包含三个部分：界面最上端是图片显示部分，位于界面中心位置是魔方颜色模拟部分，下端是系统操作部分。

系统开始运行后，其初始界面如下图 3-10 所示。点击 Recognize 按钮，本系统便开始进行魔方块颜色识别环节。在该环节，三个摄像头分别从左前、右前、右后方三个角度获取四张照片从而对各个魔方块进行颜色识别。

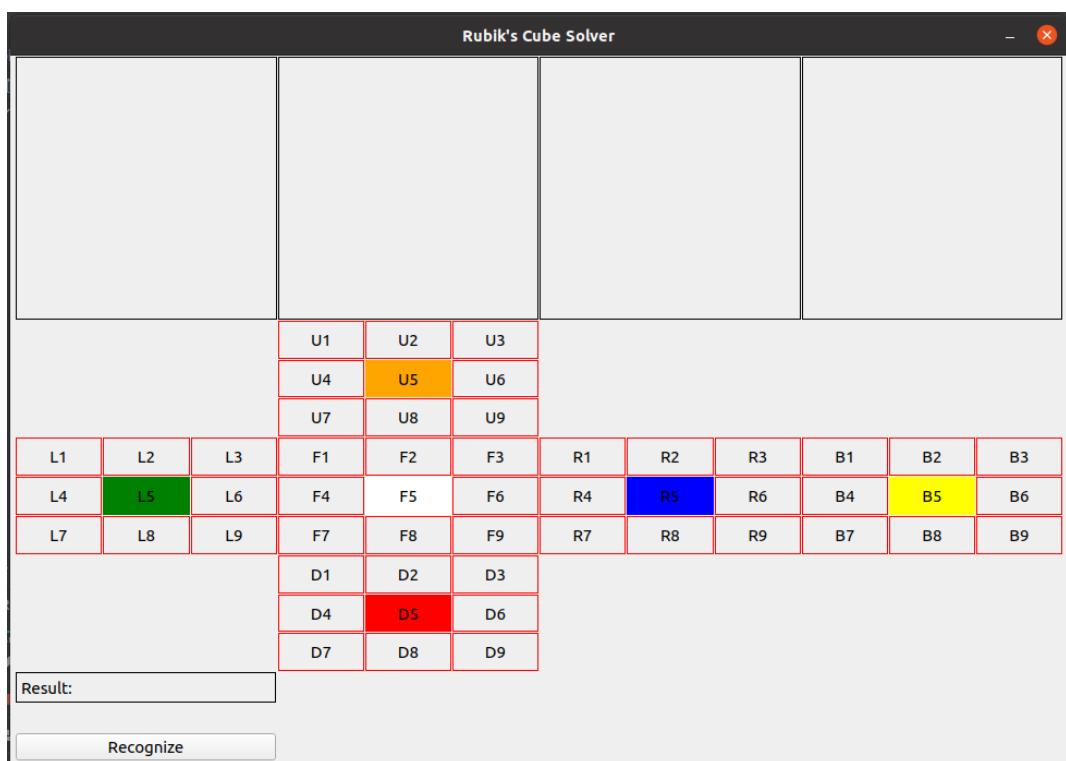


图 3-10: 系统初始界面

### 3.6.2 颜色矫正

对于颜色矫正部分,该界面如下图 3-11 所示。魔方块颜色识别完成后,系统界面上端会显示摄像头捕捉并标注候选框后的四张图像;系统界面中间部分会显示对应位置的颜色,例如 L1 位置的背景色是橙色,这代表着魔方左侧第一块的颜色为橙色;而当系统不能完全正确识别魔方块时,系统下端出现颜色修正以及可行性验证按钮。

本系统可对所有魔方块的颜色进行手工矫正,在颜色矫正栏输入魔方块位置与颜色即可对修改相应魔方块的识别颜色,例如在下图的文本框中输入 R1 与 Red 并点击 Correct 按钮,系统中间部分 R1 位置的魔方块背景色会被更改为红色,同时魔方块颜色序列中的相应位置也会被修改成红色,因此实现人工更正识别结果,可进一步增加系统的还原准确性。

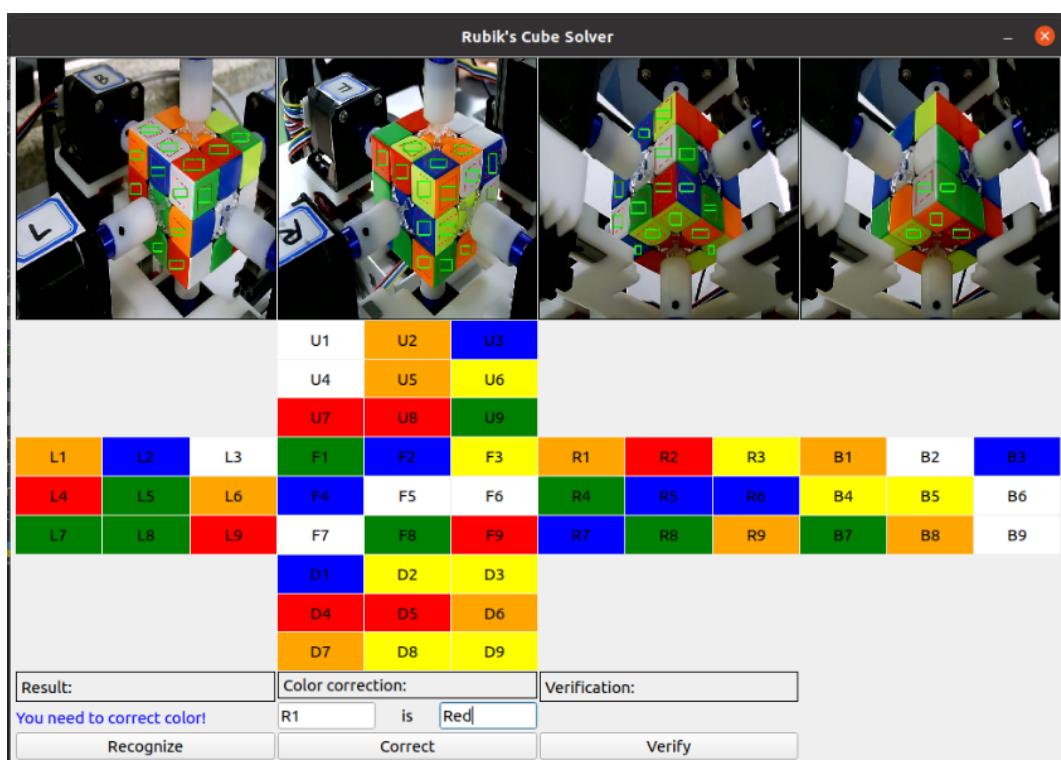


图 3-11: 颜色矫正界面

### 3.6.3 还原可行性验证

在上述的魔方块颜色矫正步骤之后, R1 位置的魔方块背景色被更改为红色, 如下图 3-12 所示。此时, 系统可再次对校正后的魔方块颜色进行还原算法的求解。

如果矫正后仍然不能得到可行的还原步骤, 那么证明并没有完全矫正正确, 仍然需要重复颜色矫正操作。

如果系统提示 Solved, 则证明已经得到可行的复原步骤, 下一步点击 Solve 按钮即可进行还原操作。

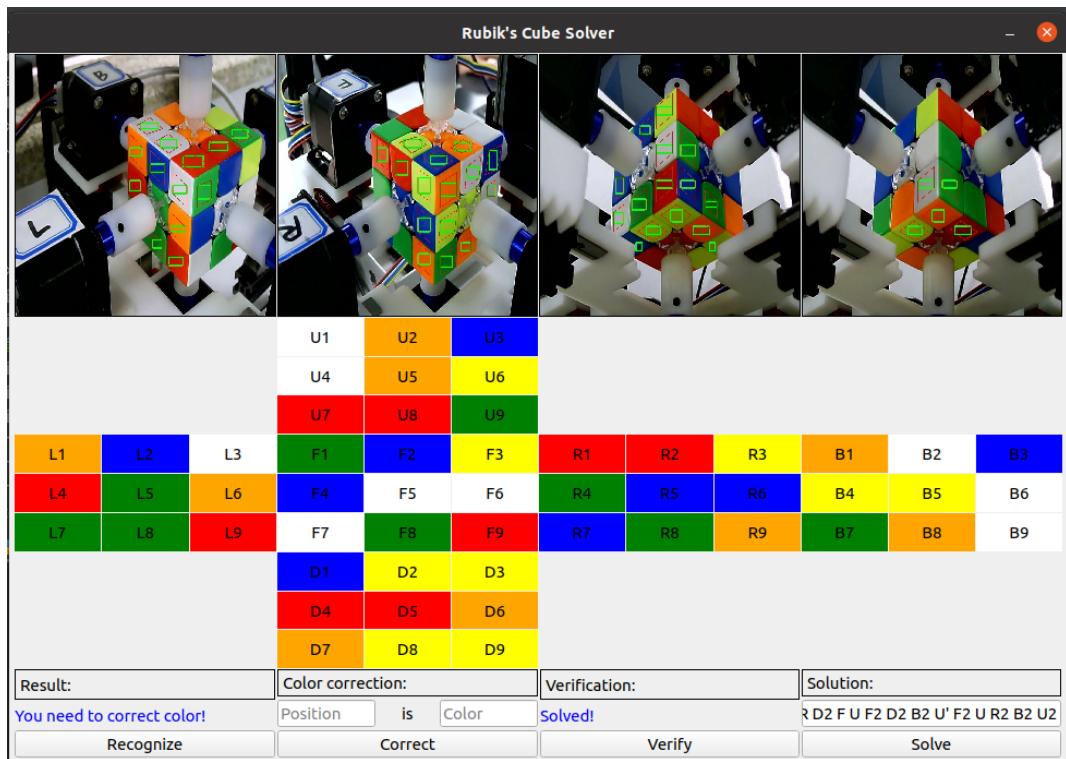


图 3-12: 还原可行性效果图

### 3.6.4 魔方块还原

在魔方块还原部分, 不管是识别完全正确还是魔方块颜色矫正完成后得到可行的复原步骤, 都需要执行魔方块还原操作。

在本系统得到魔方还原操作的序列后, 点击 Solve 按钮后即可发送电机还原旋转指令对打乱的魔方进行还原, 如下图 3-13 所示。

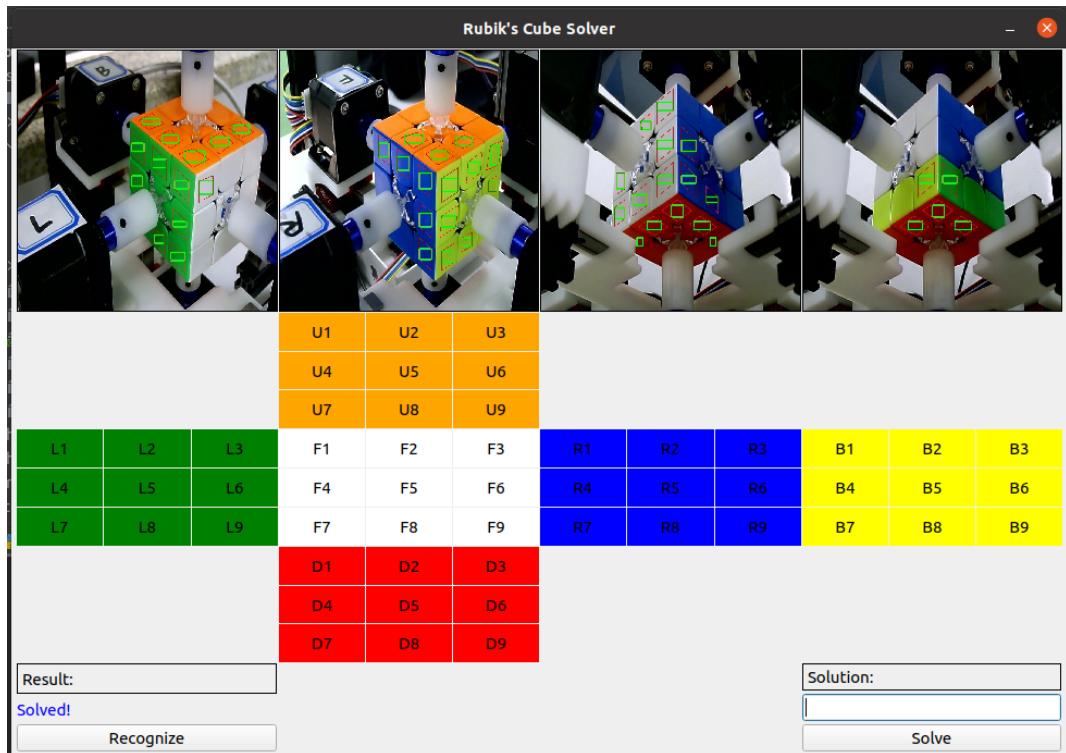


图 3-13: 魔方块还原结果界面

### 3.7 本章小结

本章首先简要介绍系统的开发平台和环境配置,接着列举系统目录规划表以及系统整体算法的流程图,再详细介绍本系统的界面设计,本系统使用的两阶段还原算法以及设计并行优化旋转操作策略。最后根据系统的不同功能截图展示本魔方复原系统的界面。

## 第四章 系统测试与分析

本章主要是对已完成的魔方复原系统进行实验、测试以及与现有的魔方复原系统进行对比分析。得到以下实验结果：

### 4.1 魔方块颜色识别实验

在本实验中，将魔方随机打乱 50 次，每次获取 48 个魔方块的颜色，得到实验结果如下所示：

表 4-1: 颜色识别实验结果

表 4-2: 颜色误识别次数表

颜色	识别正确率	颜色	识别正确率	原颜色	误识别颜色	误识别次数
白	100%	红	99.5%	红	白	2
橙	98.3%	蓝	100%	橙	红	5
绿	100%	黄	100%	橙	白	2

由上表 4-1 可知，在测试环境下，白色、蓝色、绿色、黄色的识别正确率均为 100%，但红色、橙色的识别正确率分别为 99.5%、98.3%。由表 4-2 可以看出，在该实验中，有两次将红色误识别为白色，有五次将橙色误识别成红色，有两次将橙色误识别为白色。仔细观察颜色识别前后的图片后发现，有一部分原因是测试环境中的部分灯源直接照射到魔方块上导致魔方块反光，摄像头捕捉到的魔方块为反光后的白色，因此造成误识别。还有另外一部分原因是橙色与红色的色调分量有些接近，导致无法区分二者颜色。但考虑到系统颜色误识别的次数比较少，也允许系统存在一些偶然误差，因此总体上来看系统在魔方块颜色识别上的表现还是令人满意的。

### 4.2 魔方复原算法对比分析

本文对几种应用于现有魔方系统的解魔方算法进行实验分析，其中包括层先法<sup>[39]</sup>、CFOP 法、角先法、Thistlethwaite 算法以及 Kociemba 算法。后两种算法已经

在本文 3.3.2 节与 3.3.3 节中详细介绍, 此处便不赘述, 下述是其他各个算法的简要介绍:

层先法的思路是以层为单位, 逐层复原。先是在第一层拼成交叉十字形, 接着通过旋转使第二层的四个棱块归位, 最后复原第三层。看似简单的思路每一层的旋转步骤却格外复杂, 转动序列也较长, 存在较多的冗余操作。

CFOP 法是对层先法的改进, 改进之处在于第一层到第二层、第二层到第三层的复原不需要进行转换, 不需要进行多余的操作, 但该方法并不能解决上述提到的冗余问题, 仍存在旋转步骤多等问题。

角先法的思路与层先法类似, 该方法并没有先将特定的某个面复原, 而是先复原互不干扰的几个角块, 最后根据特定的变换公式将整个魔方复原, 复原的过程中虽然会破坏复原好的角块, 但是并不会影响最后的复原结果。其复原的平均步骤数稍优于层先法, 但与 Kociemba 算法仍存在较大差距。

针对上述五种魔方复原算法, 本文对 10 组不同状态下的魔方进行复原步骤的计算与统计, 得到平均值如下图所示:

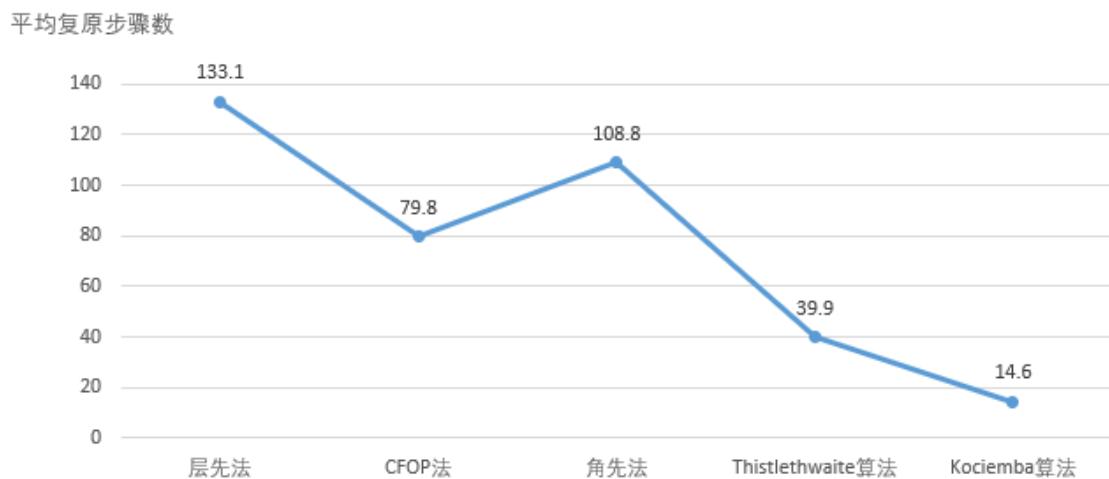


图 4-1: 还原可行性效果图

由图 4-1 可以看出, 在平均复原步骤数这个指标上, Kociemba 算法具有明显优势。此外, 在魔方复原系统的机械构造、整体框架相同的情况下, 平均复原步骤数越少, 其复原速度越快。因此, 本论文选取 Kociemba 算法作为本系统的魔方复原算法可极大程度地提升魔方复原的速度。

### 4.3 串并行复原实验

本文对系统中的串行复原与并行复原进行十次对比实验,得到以下实验数据:

表 4-3: 串并行复原实验数据表

实验编号	总步骤数	串行操作耗时(单位:秒)	并行操作耗时(单位:秒)
1	13	1.52	1.28
2	16	1.83	1.62
3	14	1.46	1.44
4	17	1.99	1.68
5	13	1.41	1.30
6	20	2.35	1.94
7	15	1.70	1.63
8	16	1.79	1.55
9	20	2.16	2.03
10	19	1.99	1.92

上表的每组数据中,当复原魔方的总步骤数相同时,并行操作消耗的时间总是比串行操作消耗的时间要少,将上述数据中每次实验的串行操作耗时与并行操作耗时除以复原步骤数即可得到复原单位步骤所消耗的时间,如下折线图所示。

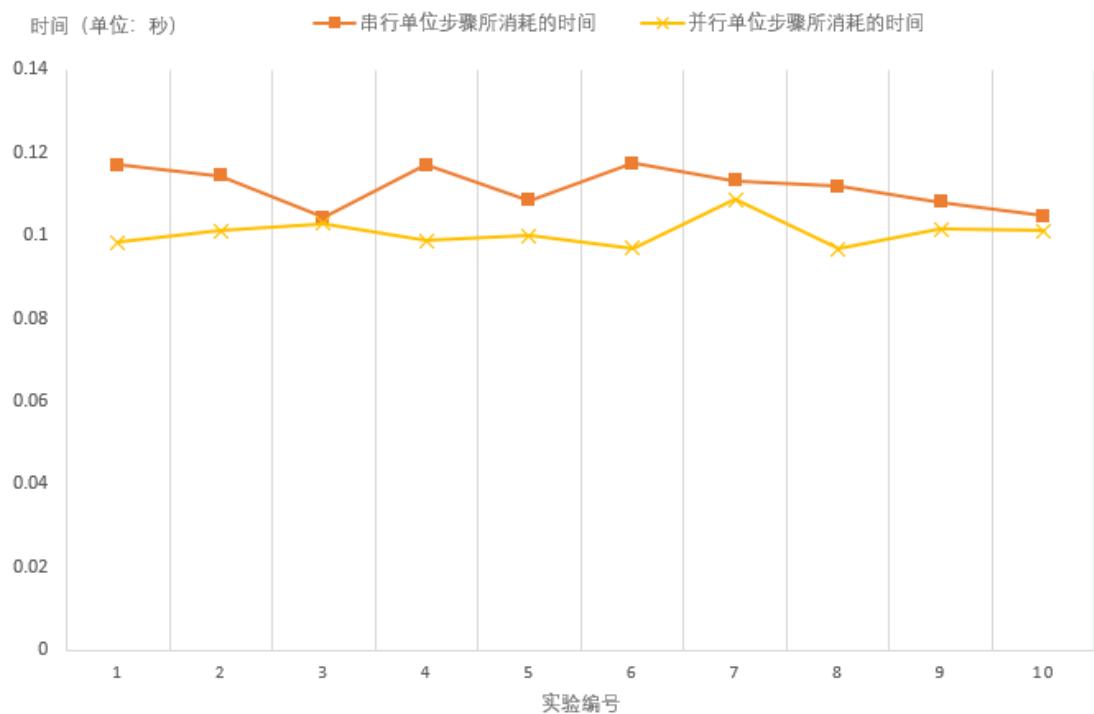


图 4-2: 单位步骤耗时图

由上表 4-3 与上图 4-2 的数据可以得出, 并行操作的复原速度总是优于串行操作, 因为在并行操作过程中总能将其复原序列中互不干扰的旋转操作并行化, 而单位旋转操作的耗时又相同, 因此整体旋转操作数较少的方法, 即并行操作, 所耗费的时间也相对较少。

#### 4.4 本章小结

本章从魔方块颜色识别、魔方复原算法与串并行复原三个方面对本系统进行对比实验。实验表明, 本系统的魔方块颜色识别算法的识别正确率优于其他系统, 并且魔方复原算法与并行还原优化算法均能在一定程度上有效提高魔方复原的速度, 实现魔方的快速还原。

## 第五章 总结

### 5.1 完成的工作

本文论述魔方求解算法的发展历程以及国内外魔方复原系统的研究现状，接着分别从硬件和软件层面详细阐述如何设计并实现了该魔方复原系统，并且在一定程度上具有教育意义和商业价值。本系统完成的工作如下所示：

- (1) 设计建模并搭建了魔方复原系统的框架，设计 PCB 电路板以达到系统高度集成化的目的。
- (2) 对上下位机进行编程，在系统完成的基础上优化了魔方复原的方式，并行还原的方式实现了魔方复原步骤的最少化。
- (3) 设计并开发了系统的可视化界面，增加人机交互功能，例如颜色矫正，还原可行性验证等。

### 5.2 存在的问题及下一步工作

本魔方复原系统结合已知最优的还原算法，并自主设计实现并行操作的优化算法，在复原操作方面可能已经达到国内外最新发展水平。但是，本系统在诸多方面仍然存在着明显不足，例如识别魔方块颜色的速度与准确率等，仍然会存在一些无法避免的错误，也正是因为这些错误，往往对系统的运行结果产生重大的影响。因此，在后续更进一步的研究中，本系统将会在以下几个方面着重进行其完善的工作：

- (1) 完善图像识别算法，使系统免于不同环境因素尤其是不同光照条件的影响。目前本算法的鲁棒性不够强，并不能完全准确识别所有不同环境情况下的魔方块颜色。与此同时，在测试系统功能期间也可以增加多种复杂环境下的情况，使其充分考虑环境因素。初步拟定设计环境适应性更强的图像识别算法。
- (2) 降低识别过程中的操作繁琐程度。就目前的系统而言，使用三个不同方位

的摄像头需要通过两次的底面  $180^\circ$  的旋转才能捕捉到所有魔方面的所有魔方块颜色，在识别的过程中仍存在着较高的操作繁琐程度。

(3) 设计更美观的可视化界面。目前，本系统的可视化界面仅仅是由数个简单的按钮和标签栏组成，并不是特别美观，也无操作提示，系统界面比较一般。在后续的工作中将融入其他风格的系统界面元素，使之更加美观。

## 参考文献

- [1] 哈金才, 李若雪, 哈瑞. 魔方的数学模型研究及其应用 [J]. 创新创业理论研究与实践, 2018, 1(19): 83–86.
- [2] Zassenhaus H. Rubik's cube: A toy, a galois tool, group theory for everybody[J]. Physica A: Statistical Mechanics and its Applications, 1982, 114(1-3): 629–637.
- [3] 张霞. 校本教材《基于 Arduino 的智能机器人设计》的开发和实践研究 [D]. 重庆: 西南大学, 2021.
- [4] 王一帆, 陈浩东, 陈文秀, 赵康莊, 赵萍. 基于机器视觉的智能魔方机器人研究综述 [J]. 机械设计, 2019, 36(03): 8–13.
- [5] Kunkle D , Cooperman G. Twenty-six moves suffice for Rubik's cube[C] // Proceedings of the 2007 international symposium on Symbolic and algebraic computation. 2007: 235–242.
- [6] Rokicki T, Kociemba H, Davidson M, et al. The diameter of the Rubik's cube group is twenty[J]. siam REVIEW, 2014, 56(4): 645–670.
- [7] Zeng D X, Li M, Wang J J, et al. Overview of Rubik's cube and reflections on its application in mechanism[J]. Chinese Journal of Mechanical Engineering, 2018, 31(1): 1–12.
- [8] Kociemba H. Two-Phase Algorithm Details[EB/OL]. 2015 [2022].  
<http://kociemba.org/math/imptwophase.htm>.
- [9] Zhang L, Tian X, Xia S. A scrambling algorithm of image encryption based on Rubik's cube rotation and logistic sequence[C] // 2011 International conference on multimedia and signal processing : Vol 1. 2011: 312–315.
- [10] 卢桂萍, 程开, 罗泽奇, 吴荣鑫, 陈永键, 康诗铄. 基于 Kociemba 算法的双臂解魔方机器人还原算法研究 [J]. 机电工程技术, 2021, 50(09): 100–103.

- [11] 高达. 基于 STM32 双臂魔方机器人的设计 [J]. 电子产品世界, 2018, 25(11): 51–53.
- [12] 郝崇清, 过仕安, 焦敏, 马海港, 于清超, 赵宇洋. 基于四臂协调控制的魔方还原系统设计与实现 [J]. 河北工业科技, 2019, 36(04): 278–286.
- [13] 吉长军, 张艳珠, 邓宣金, 杨志, 庞根蒂. 魔方机器人硬件系统设计 [J]. 电子制作, 2015(7): 81–81.
- [14] 卢仕, 张志文, 张寅, 万美琳. 基于 SoC FPGA 异构平台的魔方快速还原系统设计与实现 [J]. 计算机测量与控制, 2019, 27(06): 213–217.
- [15] 胡鑫. 基于 ARM9 的嵌入式魔方机器人系统设计 [D]. 广东: 华南理工大学, 2021.
- [16] 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(04): 29–32.
- [17] 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): 248–250.
- [18] 田田. 解魔方气动组合机械手功能结构设计与系统分析 [D]. 山东: 济南大学, 2015.
- [19] 田田, 徐林, 赵洪华, 冀向博. 解魔方四爪机械手结构设计与操作 [J]. 机器人技术与应用, 2014(05): 38–39.
- [20] 董海阳, 魏巍. 类人四轴解魔方机器人的设计 [J]. 电子技术与软件工程, 2013(08): 62.
- [21] 张雪娇. 智能魔方机器人的视觉感知与复原算法研究 [D]. 辽宁: 东北大学, 2011.
- [22] Korf R E. Finding optimal solutions to Rubik's Cube using pattern databases[C] // AAAI/IAAI. 1997: 700–705.
- [23] Agostinelli F, McAleer S, Shmakov A, et al. Solving the Rubik's cube with deep reinforcement learning and search[J]. Nature Machine Intelligence, 2019, 1(8): 356–363.

- [24] Joyner D. Adventures in group theory: Rubik's Cube, Merlin's machine, and other mathematical toys[M]. [S.l.] : JHU Press, 2008.
- [25] Knill O, Mäder R E. The rotation group of Rubik's cube[J]. ACM SIGSAM Bulletin, 1987, 21(3) : 33–43.
- [26] 夏斯权, 周亦敏, 杨一波, 黄松. 步进电机闭环控制系统的研究与应用.[J]. 机电工程, 2017, 34(12) : 1446–1450.
- [27] Fredriksen T. Applications of the closed-loop stepping motor[J]. IEEE Transactions on Automatic Control, 1968, 13(5) : 464–474.
- [28] 吴连港. 基于 Qt 的嵌入式水质检测系统界面软件设计 [J]. 农业装备与车辆工程, 2021, 59(11) : 140–142.
- [29] 乐万德, 任静, 刘舟洲, et al. Arduino 串口通信控制系统的研究 [J]. 电子设计工程, 2022, 30(04) : 69–73.
- [30] Zhang Y, Li W, Yang P. Shot Boundary Detection Based on HSV Color Model[C] // 2019 IEEE International Conference on Signal, Information and Data Processing (ICSIDP). 2019 : 1–4.
- [31] Su C H, Chiu H S, Hsieh T M. An efficient image retrieval based on HSV color space[C] // 2011 International Conference on Electrical and Control Engineering. 2011 : 5746–5749.
- [32] Fu Q, Zhang Y, Xu L, et al. A method of shot-boundary detection based on hsv space[C] // 2013 Ninth International Conference on Computational Intelligence and Security. 2013 : 219–223.
- [33] Aqthobilrobbany A, Handayani A N, Lestari D, et al. HSV Based Robot Boat Navigation System[C] // 2020 International Conference on Computer Engineering, Network, and Intelligent Multimedia (CENIM). 2020 : 269–273.
- [34] 王娟娟. 魔方机构的拓扑结构表示及其构态变换过程分析 [D]. 河北: 燕山大学, 2018.

- [35] 王士同. 基于反复加深的学习式启发式搜索算法 Learning-IDA 和 Learning-PIDA[J]. 计算机工程, 1995(S1): 109–112.
- [36] 刘宏义, 王金宝. IDA\* 和边缘搜索算法的研究 [J]. 科技信息, 2011(30): 270–272.
- [37] 王远敏. 深度优先搜索算法的应用研究 [J]. 网络安全技术与应用, 2022(03): 40–42.
- [38] Chakrabarti P P, Ghose S, Acharya, A, et al. Heuristic search in restricted memory[J]. Artificial intelligence, 1989, 41(2): 197–221.
- [39] 刘润川. 魔方按层求解算法设计 [J]. Computer Science and Application, 2019, 9: 406.

## 致谢

文至于此，意味着该告一段落了。

大学匆匆四年，数不尽的故事都已封存在记忆里，千言万语说不尽。遇到过困难，遇到过挫折，也遇到过帮助。迷茫过，失落过，也欣喜过。

一致恩师，首先感谢潘翔老师这段时间以来的点评与指导以及在我论文撰写的过程中不遗余力的帮助，感谢陈国定老师、褚衍清老师在智能车比赛期间给予的问候与关心，感谢王海贵老师、周超老师等其他计算机学院学工办的老师，也感谢所有传道授业解惑的老师们。

二致密友，感谢机器人队友陈兴迪、杨振鑫、丁晨俊，智能车队友单琦玮、陈泽众、王宣治，感谢备赛期间的奋斗与付出，正是我们不懈的努力才有如今的成就。感谢强哥一直以来的陪伴，在我需要倾诉的时候成为我无怨无悔的树洞。也感谢体育部的朋友们对给我带来的欢乐。

三致父母，感谢你们二十多年来对我的关心、照顾与支持。在外求学，是你们给了我坚实的后盾，为我创造优越的条件，让我在学习路上无所顾虑。

再次感谢帮助过我的所有人，但寥寥数语无法道尽感激之情，天下也没有不散的宴席，一段旅途的结束意味着另一段征程的开始。未来还很漫长，我必将乘风破浪、扬帆起航！

## 附录

附录 1 毕业设计文献综述

附录 2 毕业设计开题报告

附录 3 毕业设计外文翻译(中文译文与外文原文)