毕业论文(设计)

论文（设计）题目：

高速旋转物体的残影幻象的定制化设计

姓　　名 陈胜杰

学　　号 201900130138

学　　院 计算机科学与技术学院

专　　业 人工智能

年　　级 2019 级

指导教师 赵海森

2023 年　5　月　10　日

山东大学本科毕业论文（设计）成绩评定表

学院：　　专业：　　年级：

<table>
<tr><td>学号</td><td></td><td rowspan="2">论文（设计）
总成绩</td><td>分数</td><td></td></tr>
<tr><td>姓名</td><td></td><td>等级</td><td></td></tr>
<tr><td colspan="2">论文（设计）题目</td><td colspan="3"></td></tr>
<tr><td rowspan="3">指
导
教
师
评
语</td><td colspan="4"></td></tr>
<tr><td>评定成绩（分数）</td><td colspan="2">×40%</td><td></td></tr>
<tr><td colspan="4">教师签名：　　年　月　日</td></tr>
<tr><td rowspan="3">答
辩
小
组
评
语</td><td colspan="4"></td></tr>
<tr><td>答辩成绩（分数）</td><td colspan="2">×60%</td><td></td></tr>
<tr><td colspan="4">组长签名：　　年　月　日</td></tr>
</table>

注：论文（设计）总成绩：分数=指导教师评定成绩（40%）＋答辩成绩（60%）。

等级：优秀（90 分以上），良好（80-89 分），中等（70-79），及格（60-69），不及格（60 分以下）。

目　录

高速旋转物体的残影幻象的定制化设计

摘　　要

旋转现象在我们的生活中频繁的出现，例如高速旋转的风扇、钻机等事物，而随着时代发展，旋转也成为了艺术的一部分。大多数关于旋转的艺术，例如陀螺、悠悠球等，通常是将关注的地方放在了物体在高速旋转的过程中实现动态平衡上，而未关注物体在高速旋转的情况下由于视觉暂留产生的成像效果。本文则是从这个方面进行研究设计。人眼独特的视觉暂留会让旋转物体出现深浅不一残影，而我们可以通过反向利用这种从物体旋转得到二维残影的现象，从图像制造三维模型，使制造出来的三维模型在旋转时，人眼看起来和给定图像高度相似。可以预见这会是很有潜力的时尚潮玩。而我们所做的工作就是研究高速旋转的物体和人眼看到的影像之间有什么关系，找出他们的转换规律，针对某些二维图像自动化生成相应的三维模型。

该项目的核心挑战是从图片到三维模型的生成算法，从代码中的数据到实际打印模型的自动化转换，并且需要考虑模型的可打印性。我们通过使用相关代码工具，模型设计工具，步进电机的控制来完成主要流程框架的搭建，实现从设计到实现到打印的流程。然后通过基础实验、阅读相关文献资料来完善我们的生成过程，完善模型的可打印性。

通过上述工作，完成了从图片到模型的生成流程，通过一些基础实验得出了相关模型与图像之间的映射规律，实现了简单的模型生成算法。

关键词：视觉暂留；旋转；模型生成；

ABSTRACT

The phenomenon of rotation frequently occurs in our daily lives, such as high-speed rotating fans, drilling rigs, and other things. With the development of the times, rotation has also become a part of art. Most art related to rotation, such as gyroscopes, yoyo balls, etc., usually focus on achieving dynamic balance of objects during high-speed rotation, without paying attention to the imaging effect caused by visual persistence of objects during high-speed rotation. This article focuses on this aspect of research and design. The unique visual persistence of the human eye can cause rotating objects to have varying shades of residue, and we can reverse this phenomenon of obtaining two-dimensional residue from object rotation to create a three-dimensional model from the image, so that when the three-dimensional model is rotated, the human eye looks highly similar to the given image. It can be foreseen that this will be a very promising fashion trend to play with. And what we do is to study the relationship between high-speed rotating objects and images seen by the human eye, identify their conversion laws, and automatically generate corresponding 3D models for certain 2D images.

The core challenge of this project is to generate algorithms from images to 3D models, automate the conversion from data in code to actual printed models, and consider the printability of the models. We complete the construction of the main process framework by using relevant code tools, model design tools, and stepper motor control, achieving the process from design to implementation to printing. Then, through basic experiments and reading relevant literature, we will improve our generation process and improve the printability of the model.

Through the above work, the generation process from images to models was completed, and the mapping rules between relevant models and images were obtained through some basic experiments, achieving a simple model generation algorithm.

Keywords: Visual persistence; Rotation; Model generation;

第1章 绪　论

1.1 选题背景和研究意义

自古代以来，旋转玩具就一直是吸引人们想象力的好玩的东西。世界各地独立发明的纺纱陀螺其实在古希腊文献中有所雏形。同样，虽然溜溜球被认为是在中国发明的，但也有许多历史参考，比如莫扎特的《费加罗的婚礼》,其中就有一段是通过旋转一个溜溜球来减轻压力。这些玩具有着悠久的传统，并且直到今天。对于我们来说，创造新的设计一直是一个反复试验的过程， 需要艺术家和爱好者的直觉和一丝不苟的耐心。[1]

由于 3D 制造技术的进步，面向制造的设计最近引起了计算机图形社区越来越多的兴趣。而我们的项目是从模型设计到三维模型的制造，因此跟 3D 制造业相关密切，下面介绍一下 3D 制造与设计方面的历史与重要人物。

提起 3D 制造，就不得不提 Sketchpad 的创始人 Ivan Sutherland，他有着“虚拟现实之父”、“计算机图形之父”等数个头衔，他是技术产业中最伟大的先驱者和思想者之一。Sketchpad 这个开创性的项目有助于创造第一个 3D 人工制品[2]。Sutherland 与他的合作者 David Evans 一起建立了犹他大学的首个计算机技术系[3]。1969 年，Sutherland 或 Evans 创立了第一家 3D 图像公司，简称为“Evans & Sutherland”[4]。

不久之后，Ed Catmull 开发了 RenderMan 渲染器以及在数字影像合成方面做出杰出贡献而三次获得奥斯卡科学与技术奖[5]。Ed Catmull 发明的纹理映射技术对于呈现双三次曲面和形式至关重要。

除此之外，Martin Newell 可能是创建 3D 可视化和建模的另一个关键人物。他对 3D 图形的算法实现有着许多独到的见解，比如渲染图形的阴影效果、反光的材料质地或是通过旋转来展现不清晰的图形表面。他用简单的三维形状制作了一个茶壶，被称为犹他茶壶。犹他茶壶已成为渲染的标志。几乎所有 3D 图形工具都包含这样一个模型，用于启动绘图过程[6]。

关于 3D 制造与设计的软件，在 1990 年代，光线追踪就在 Autodesk 3D 建模和建模产品中用于可视化，而将光线追踪应用于可视化就是在模拟人眼对自然

光线的捕捉过程。 Syntha Vision 图形程序是最早使用这种方法的[7]。Tom Hudson 在 80 年代建立了用于 3D 建模的 THUD 程序[8]， Autodesk 于 1990 年推出了第一个 3d Workshop[9]。 Hudson 和 Dan Silva 一起制作了第一个 Autodesk 3d Workshop，由五个组件组成：shaper、loftier、editor、material editor 和 key - 构图[10]。

而本课题的灵感来自美国计算机协会期刊 2017 年发布的一篇 Emily Whiting 的文章《Spin-It: Optimizing Moment of Inertia for Spinnable Objects》。这篇文章主要研究的是通过改变物体内部结构、质量分布来实现非对称物体的稳定平衡旋转。从该文章的内容中，我们发现了旋转艺术的另一方面体现。如图 1-1，这是来自文章《Spin-It: Optimizing Moment of Inertia for Spinnable Objects》的一张配图，虽然它本来是用来研究非对称物体平衡旋转的，但是我们可以从这张图中看到物体的旋转产生的残影现象，这也是旋转艺术的一方面体现。因此，我们的课题选择了关于这方面的旋转艺术的研究。

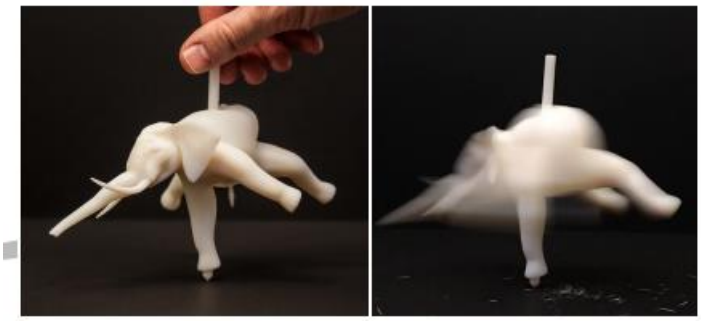


图 1-1 来自《Spin-it》文章中的物体旋转图

本文研究的方向是人眼视觉，有关残像的形成。这些都来源于人眼的视觉暂留效应，下文进行相关介绍。

1.2 视觉暂留

视觉暂留现象即视觉暂停现象又称“余晖效应”，1824 年由英国伦敦大学教授彼得·马克·罗杰特在他的研究报告《移动物体的视觉暂留现象》中最先提出[11]。

视觉暂留在生活中是很常见的一种现象。简单来说，物体上反射到人眼的光线已经消失了，但是人眼人脑还是能在一段时间内感受到视觉的残留。我们的眼睛能看到物体，是因为光在物体上反射到**视网膜**上产生作用，视网膜上的感光细

胞将光信号转换成电信号，电信号经过神经传输给大脑。**感光细胞**是通过**感光色素**来实现感光的，但是色素的形成以及消退是需要时间的，所以对物体的感光也是需要时间的，当失去光时，需要延续 0.1 到 0.4 秒的时间视觉神经对物体的影像才会消失，这就形成了视觉暂留现象。[12]

一开始，William Benjamin Carpenter 认为视觉暂留现象“相当于精神而不是视网膜现象”[13]。视觉暂留论认为人类对运动的感知（以大脑为中心）是视觉持续性（以眼睛为中心）的结果[14]。一个更合理的解释运动知觉的理论认为视觉暂留有两种截然不同表现：phi 现象和 beta 运动[15]。Joseph Anderson 和 Barbara Fisher 认为，phi 现象是电影中的建构主义方式[16]，而视觉的持久性则是现实主义的方法[17]。

关于视觉暂留的相关解释、理论与应用如上所述。很明显可以看出，对视觉暂留的应用大部分都是物体上光快速的消失，或者说物体的快速移动。而很少从旋转物体中应用到视觉暂留，尽管这是个很普遍的现象。关于视觉暂留的应用，因为平时生活中的旋转物体大多都是对称的、简单的，甚至是高速旋转下无法产生残影的，因此大部分人很难将视觉暂留与旋转的艺术联想到一起。也正是如此，在看到图 1-1 中非对称物体旋转产生的残像后才产生了通过高速旋转和视觉暂留效应产生残像的现象来实现对 3D 模型的设计，从视觉方面而不是经典力学方面来研究旋转的艺术。

1.3 论文组织结构

本文首先详细介绍该项目中完成的工作，然后介绍相关实验以及得出的规律，并且介绍通过这些规律得到的模型生成算法。

第 1 章 为绪论部分，简要介绍了模型旋转的研究背景与意义，以及相关重要概念视觉暂留。在本章的最后介绍了本文的组织结构。

第 2 章 详细介绍了实验平台以及算法框架的搭建，介绍了项目主要流程。

第 3 章 详细介绍了基础实验以及相关规律，简历数学模型对问题进行分析，通过基础实验对规律进行总结，然后提出了模型生成算法。

第 4 章 为结论部分，介绍了对本次毕业设计的总结与展望。

第2章 项目基础平台搭建

本章对本文提出的项目基础平台及其搭建做了详细介绍。首先整体框架和平台的介绍，详细介绍了每个部分的设计思想及用处。在算法框架搭建部分介绍了算法主要流程以及对项目的意义，模型设计打印部分介绍了将想法变为实际模型的流程，搭建旋转平台部分介绍了如何通过具体设备完成项目的实际操作。

2.1 主要框架及平台介绍

给出一个图像生成对应模型，使其在高速旋转的情况下人眼残像与给出图像相似。完成生成算法的设计前，需要相应的算法框架以及实验平台搭建来完成人眼拟真、模型转换、实际观察旋转掌握相关规律等事项，以此更好的回馈模型生成算法，这在本文被统称为**项目基础平台**。项目基础平台就像是一台 CD 机，不同的模型生成算法相当于卡碟。CD 可以放不同的卡碟，并且播放质量不仅与卡碟有关也与 CD 机有关。项目基础平台是为了让图像到模型的算法解耦，能轻松的完成各部分的更替而不影响其他部分，同时该平台的完善度也会影响最后生成算法的优秀程度。

算法框架负责完成主要目标**模型生成算法**。而其他的算法则用来辅助完成该目标，包括将三维网格转换成可打印模型的**转换算法**、模拟人眼看模型旋转图像的**人眼拟真算法**。

对模型的设计与打印方面是借助相关软件设计一些模型，设计出一些基础独特的模型并且打印出来，然后在旋转平台上进行旋转。这样来完成一些实验总结出一些有用的规律，反馈到模型生成算法上来优化目标。

最后本项目需要进行实物的旋转操作，因此需要搭建好用来旋转的平台。

2.2 算法框架搭建

本节介绍具体的算法框架搭建工程以及细节。

首先介绍相关代码工作的意义以及流程，本项目主要目标是得到一个好的**模型生成算法**，将图像输入得到一个三维可打印模型。次要辅助工作是对生成模型的验证工作，该部分主要就是**人眼拟真算法**，该算法将模型输入得到一个图像，

顾名思义得到的图像是模拟人眼在看到的模型在高速旋转下得到的残像，该算法拟真程度越好就越能看出我们生成算法的优秀程度。并且根据拟真算法的结果来调整优化生成算法。然后我们在代码中用到的模型是三维网格的形式存储，而三维网格的形式所存储的内容是离散的点，并不能直接用来打印出模型。因此，本文合理的设计出了将三维网格转换成可打印文件的**转换算法**。

主要流程如图 2-1。按照如图 2-1 的流程，程序运行完成之后能得到可打印的模型文件，以及旋转拟真的图像。

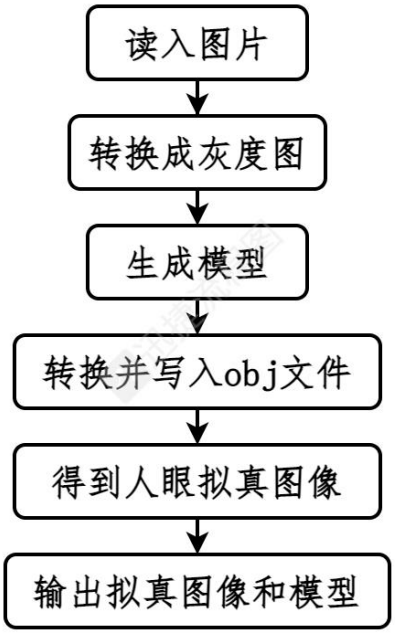


图 2-1 算法流程图

接下来对所使用的数据结构以及主要辅助算法进行详解。

2.2.1 数据结构

本文所提到的算法要对相关的数据进行存储以及一些运算，需要用到相应的数据结构来完成。主要使用的数据结构如下：

1. vector：动态数组，可动态申请内存，用来存储模型三维网格数据以及临时存储二维图像。vector<vector<vector<uchar>>> 三维动态数组用来存储三维网格数据，以及二维图像集合。
   vector<vector<uchar>>用来作为二维图像操作的临时存储。相较与其他数据结构，在本文算法中 vector 动态数组更适合相关操作。

2. Mat：Mat 是 opencv 中的一种矩阵数据类型，用来存储图像。Mat 类包含两个数据部分：矩阵头和指向存储所有像素值的矩阵的指针。矩阵头包含矩阵的大小尺寸、存储方法、存储地址等。在 opencv 中，对矩阵 Mat 的复制分为深复制和浅复制，与 c++中的深拷贝和浅拷贝类似，深复制创建了一个读取部分的矩阵头，而浅复制只是拥有自己矩阵头，与原像素矩阵共用一个矩阵数据。使用 Mat 来完成图像的读取、输出以及对图像的一些操作。
3. Matrix：Matrix 在 Eigen 中，所有的矩阵以及向量都是 Matrix 模板类的对象。主要使用 MatrixXi 和 MatrixXd，是 Matrix 的对象，用来完成矩阵方面的运算和操作。
4. FILE*：文件指针。每当打开一个文件的时候，系统会根据文件的情况自动创建一个 FILE 结构的变量，并填充其中的信息。而文件指针变量就能够提供一个对文件进行操作的入口。通过文件指针来完成对文件的打开关闭、写入删除等操作。实现将数据写入 Obj 文件中。

2.2.2 人眼拟真算法

人眼拟真算法是对主要目标完成辅助矫正的工作，通过人眼拟真算法可以在得到模型的同时就能得到近似的效果预览图。也就是说避免了将模型从文件中打印出来，然后再进行实际旋转，能节省大量的时间。因此该算法是很有必要的，并且拟真算法的优秀程度也是很重要的。

人眼拟真算法的主体思路就是将模型变成图像输出，而拟真则根据现实去模拟这个过程。现实中的人眼观察高速旋转物体残影，物体在旋转，人眼有视觉暂留效应，因此能看到残影。与此对应的，拟真算法就要将模型旋转起来，而计算机只能模拟离散的过程，就每隔一定的时间记录下模型的投影。这里粗略的将人眼看作是一个平面，将模型对应的投射到一个平面上来拟真投射到人眼中。

具体的思路，对于三维网格存储的模型，每个点都有相应的坐标。对于一个点 P0（x0 , y0 , z0），将其投射到一个平面上会得到一个二维坐标，先设其映

射点的坐标为 P1（x1 , y1）。旋转轴为 x 轴，投影平面为 xOy 平面，假设当前旋转角度为 α ，那么对于点 P0（x0 , y0 , z0），假设其旋转之后得到的点的坐标为 P2（x2 , y2 , z2）。那么 P1 和 P2 映射关系为：x1 = x2 , y1 = y2 。P0 与 P2 对应关系，因为旋转轴为 x 轴，因此 x 坐标不会发生改变：x0 = x2。关于 y 坐标和 z 坐标，首先点 P0 和点 P2 到旋转轴的距离不会发生改变，因此可以根据三角函数得到相应的坐标映射关系。因为旋转轴是 x 轴，所以旋转角度就是点到旋转轴的垂线与平面 xOy 的夹角。设 P0 一开始对应的角度为 β ，那么 P2 对应的旋转角度就是（ α + β ）。设 P0 到旋转轴的距离为 L，那么 P2 到旋转轴的距离也为 L。

通过分析可以得到以下式子：

$$y_0^2 + z_0^2 = L^2 = y_2^2 + z_2^2 \tag{2-1}$$

$$\tan\beta = z_0/y_0 \tag{2-2}$$

$$\tan（\beta + \alpha） = z_2/y_2 \tag{2-3}$$

通过上述式子就可以得到

$$y_2 = \sqrt{y_0^2 + z_0^2}\cos(\text{atan}\,(z_0/y_0) + \alpha) \tag{2-4}$$

$$z_2 = \sqrt{y_0^2 + z_0^2}\sin(\text{atan}\,(z_0/y_0) + \alpha) \tag{2-5}$$

又因为点 P1(x1 , y1)坐标中，x1 = x0 , y1 = y2。因此，通过上述式子就可以得到点 P0(x0 , y0 , z0)映射到平面上的点 P1 的准确坐标了。

对于三维网格模型中每个点都经过上述运算就能得到当前角度的二维映射图像。对于每个角度都能得到一个映射的二维图像，因此旋转一圈后就能得到一组图像，图像数量 n 与固定旋转角度 α 的关系为：n* α =360^o。得到 n 张图像后，考虑将其融合成一张人眼拟真图。简单的方法就是直接加权平均每一处的像素值。

2.2.3 转换算法

在代码当中，模型的存储一直都是通过三维网格的形式进行存储。但是三维网格的形式并不能用于打印模型，模型打印是需要特定的文件格式的，如 Obj、Stl 等文件格式。因此需要将三维网格转换成相应的文件输出才能完成后续的模型打印工作。

Obj 文件格式为关键字后面加上数据。例如，表示一个点就用关键字“v”，后面跟上点的坐标，如：“v 1 0 0”，就表示该模型中有一个坐标为（1,0,0）的点。表示一个面就用关键字“f”，后面跟上该面的顶点，如：“f 1 2 3 4”，就表示该模型中有一个由第 1、2、3、4 号点共同围起来的面。基本上靠点和面就可以组成一个模型了。

再来看看三维网格的数据，对应的坐标上如果有值就表示此处是模型所占据的空间。将三维网格数据转换成 Obj 文件格式，首先就要将离散的点转换成连续的点和面，简单来说，对每个点进行变换，将点变成正六面体即可。简单的将每个点展开成八个顶点和六个面然后写入 Obj 文件中是我们不乐于看到的，因为这样会有很多的点、面重复了，我们看到的模型只有模型的表面和上面的顶点，而模型内部展开的点和面是没有意义的，只会让 Obj 文件变得更大、算法效率更低下、读取模型花费时间更长。

我们所采取的方法是对每个点进行**检测变换**，就是对于六个方向分别进行检测，检测对应方向上是否存在点。也就是判断每个方向的面是否是模型的表面，如果是则添加一个面，如果该方向上存在一个相邻的点，则表示该方向处于模型内部，就不用添加一个面。

遍历模型上所有点，对于点 P(x,y,z)。有六个方向，就是三个左边分别加减 1。可以提前将方向矫正值存入数组，遍历方向时直接遍历方向数组，然后与坐标进行相加即可得到相应方向上的坐标，即$P' = P + fx[i]$，i 的值为 0-5 的整数。然后检测P'处是否有值，也就是是否处于模型内部，如果不在模型内部，说明点 P 在这个方向是面对外界的，是模型的表面，因此需要创建对应的面的信息写入到 Obj 文件中。反之则说明在模型内部，就不用创建对应的面来写入到 Obj 文件中。

2.3 步进电机搭建旋转平台

本节介绍步进电机以及通过代码控制步进电机搭建旋转平台。

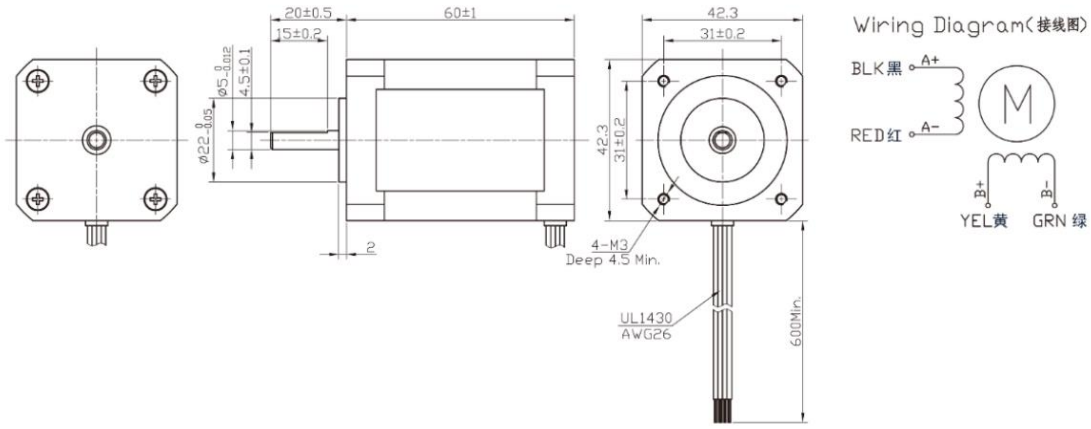
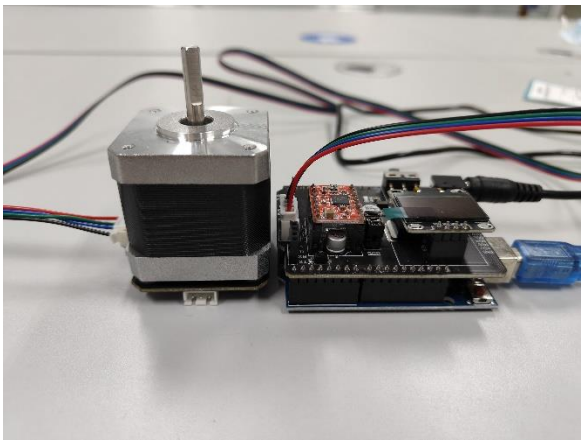
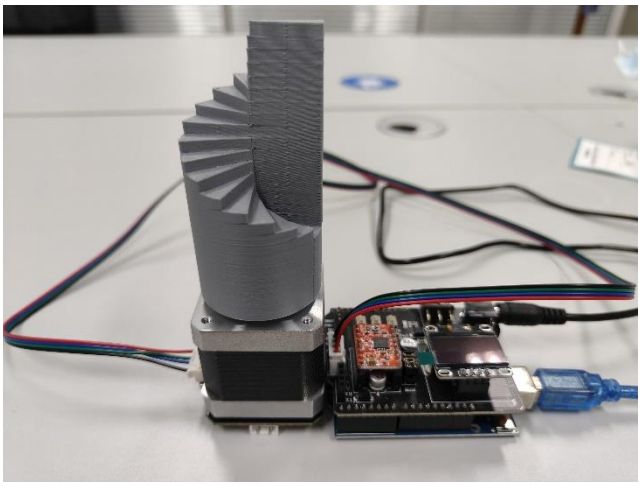


图 2-2 42 步进电机示意图





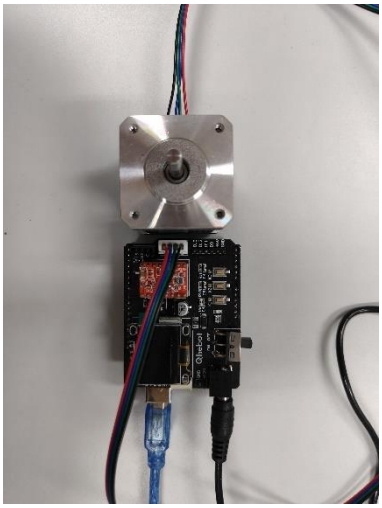


图 2-3 步进电机加 uno 板实物图

步进电机尺寸样式如图 2-2，实物如图 2-3。下面介绍步进电机驱动板。

步进电机驱动板接线端子共有 12 个，六个为脉冲和使能接线口，六个为两相电机接口，和电机直流供电端口。

中间有八个拨码开关，前三个是设置电流，通常情况下，电流设定为电机的额定电流。如果系统对发热的要求很高，可以适当减小电流以降低电机的发

热，但是电机的输出力矩会同时降低。不要求电机连续长时间运行，可以适当的增大运行电流来获得更大力矩，但是不能超过最大额定电流的 1.5 倍。第四个拨码开关是对电流的半/全流控制，是用来设置停止时的静态电流值， off 表示驱动器上电脉冲停止时，驱动器将输出给电机的电流切换为转动时的一半（半流）； on 表示驱动器上电脉冲停止时，驱动器保持与转动时同样的电流输出给电机（全流）。一般用途中应将 SW4 设成 off，使得电机和驱动器的发热减少，可靠性提高。后四个是用于设置电机每转所需脉冲个数，电机转速 = 指令脉冲频率÷每转脉冲；电机行程 = 指令脉冲数量÷每转脉冲。

关于电路接线，首先是控制信号接线，标准 R 系列驱动器信号接口为脉冲形式，R42 可以接收两种脉冲指令信号。上位控制器可以是 PLC、单片机、控制卡、控制器等脉冲信号发生装置。 R42 驱动器可接受的脉冲电平为：3.3V~24V（无需串电阻）。

其次是使能接线，默认光耦关闭时驱动器输出电流给电机；内部光耦导通时，驱动器将切断电机各相的电流使电机处于自由状态，此时步进脉冲不被响应。当电机处于报错状态时，ENA 输入可用于重启驱动器。首先应排除存在的故障，然后输入一个下降沿信号至 ENA 端，驱动器可重新励磁运转。 ENA 信号的电平逻辑可以设置为相反，逻辑与上述相反。

最后就是按照四线步进电机接线规则接线，并通入符合参数的直流电源。

步进电机通过 uno 板来控制，uno 板可通过连接电脑，从 Arduino IDE 中拷贝程序进来运行。通过代码控制上述介绍中做说的相应脉冲信号端口、使能信号端口、方向信号端口的开启与关闭，再通过 delayMicroseconds 函数控制旋转之间的间隔来实现对速度的控制。

第3章 相关实验以及规律总结

本章介绍完成的一些基础实验所掌握的相关规律。在上述框架和平台搭建完成之后，就需要完成对从图像到模型之间的映射关系的掌握。通过提出想法，设计相应模型，完成实际操作等一系列步骤来验证相关想法的正确与否。并且寻找新的规律。

3.1 相关定义

定义 1：亏格是代数几何和代数拓扑中最基本的概念之一，定义为：若曲面中最多可画出 n 条闭合曲线同时不将曲面分开，则称该曲面亏格为 n。连接的，可定向的表面的亏格是一个整数，它表示沿着不相交的闭合简单曲线的最大切割次数，而不会导致曲面断开。 或者，可以通过关于闭合表面的关系 $X = 2 – 2g$ 来定义欧拉特征 X，其中 g 是亏格。 对于 b 边界分量的曲面，方程式为 $X = 2 – 2g - b$。 按照外行人士的说法，它是一个物体的“孔”数量（“孔”在圆环孔的意义上解释;空心球体在这个意义上被认为是零孔）[18]。连接的，不可取向的封闭表面的不定向亏格，表示连接到球体的交叉数量的正整数。 或者，可以通过关系 $X = 2 – k$ 来定义关于欧拉特征 X 的闭合表面，其中 k 是不定向属性[19]。

定义 2：实体区域是指在物体高速旋转下始终能看到的区域。物体从任何角度去看都会得到二维影像，所有二维影像始终重合的地方在人眼看来就是实体的也就是完全不透明的区域。如图 3-1 中所示的不透明区域。

定义 3：残影区域是指在物体高速旋转下不一定始终能看到的区域。物体从不同角度观看得到的影像不一定相同，而不同的地方也就会在高速旋转下看起来若隐若现。如图 3-1 中的半透明的区域。

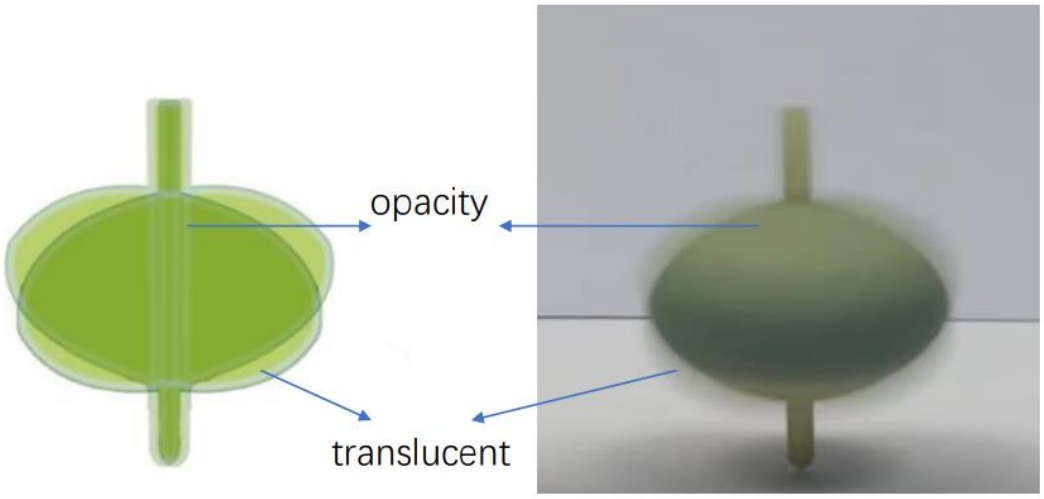


图 3-1 实体区域与残影区域示例图

3.2 对物体旋转的分析

本节介绍了对模型旋转的数学分析，尝试建立数学模型分析图像到模型的关系。

从简单的开始进行分析。从**亏格**的角度来对模型进行划分，从而进行逐步分析。对于实的闭曲面来说，亏格就是曲面上孔洞的个数。因此从亏格的个数上进行分析就能从简到难的逐步分析问题。

3.2.1 零亏格

零亏格表示闭曲面上没有空洞，比如实的正立方体、椭球、圆球或者是其他一些不那么规则的物体。对于零亏格的模型分析较为简单，零亏格的模型不存在外圈遮掩内圈的情况，该情况在后续 n 亏格的分析中会提到。

零亏格的情况如图 3-1 所演示，很明显的能看出来**实体区域**和**残影区域**。但是也很容易发现，残影区域只会在实体区域的外侧而不会出现在实体区域的内侧。残影区域的产生正是因为某些角度下的模型映射在该区域存在投影的同时某些角度下模型映射在该区域不存在投影，这样才会有若隐若现的半透明感。而如果存在残影区域在实体区域内部，就说明一定存在某个角度下模型映射存在孔洞，而这与零亏格的定义产生矛盾。

通过上述说明，零亏格的模型在高速旋转的情况下是不存在残影区域在实体区域内部的情况。而从模型到图像的这种规律对于到我们从图像到模型的数学模

型上，就表现为只有半透明区域在不透明区域外侧的图像才会生成零亏格模型。而这种图像都比较简单并且具有较小的意义。

3.2.2 N 亏格

介绍完了相对简单的零亏格模型与图像的对应关系，接下来分析相对复杂的 n 亏格模型与图像之间的对应关系。

相对来说 n 亏格和零亏格的区别就是 n 亏格的模型中有孔洞，因此会出现残影区域在实体区域内部的情况。而绝大多数灰度图都会有这种情况，比如说眼睛、嘴巴等位置。所以绝大多数图像生成的模型会是 n 亏格的。

对 n 亏格的模型进行分析。首先，模型旋转起来后，模型上每个点对应到图像上是从旋转轴到该点到旋转轴的距离上。如图 3-2，红色虚线是视觉平面的平行平面，两个长方形是简化的模型俯视图，并且代表模型旋转到不同的位置时的状态。可以看到在旋转到不同的角度时，外圈的部分在视觉平面上会转到内圈。在我们人眼所看到的平面上就是，每个部位会对其离中心最远位置到中心位置的部分的颜色深度提供贡献。

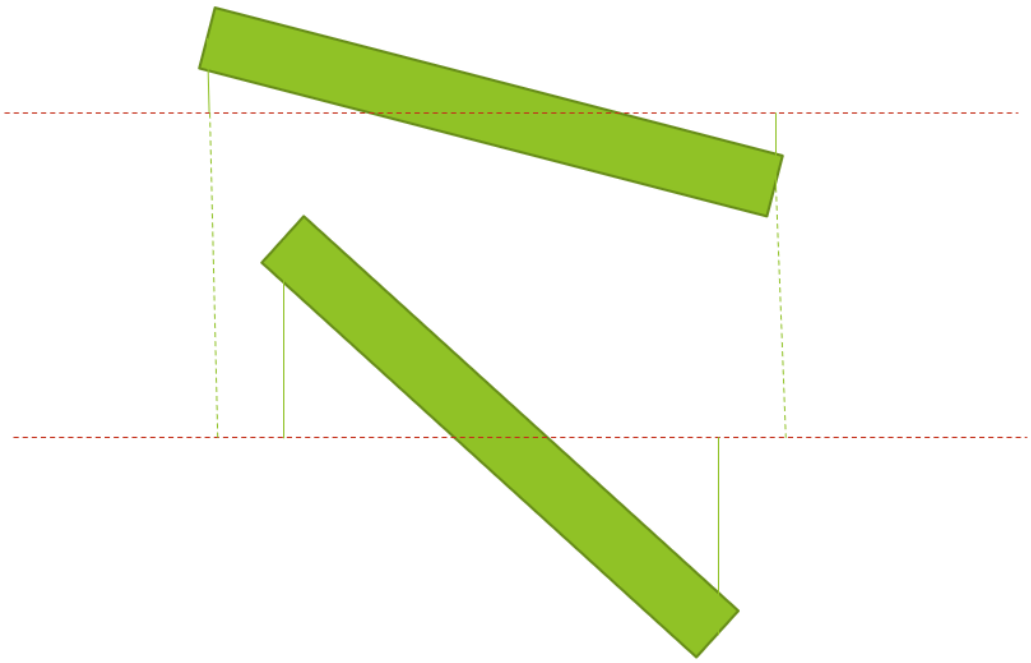


图 3-2 旋转俯视图

从模型对图像的贡献上分析，图像上一点是在同一截面上由模型上与旋转轴的距离大于等于图像上该点与旋转轴距离的所有点贡献而成。可以对此建立相应的数学模型进行分析。

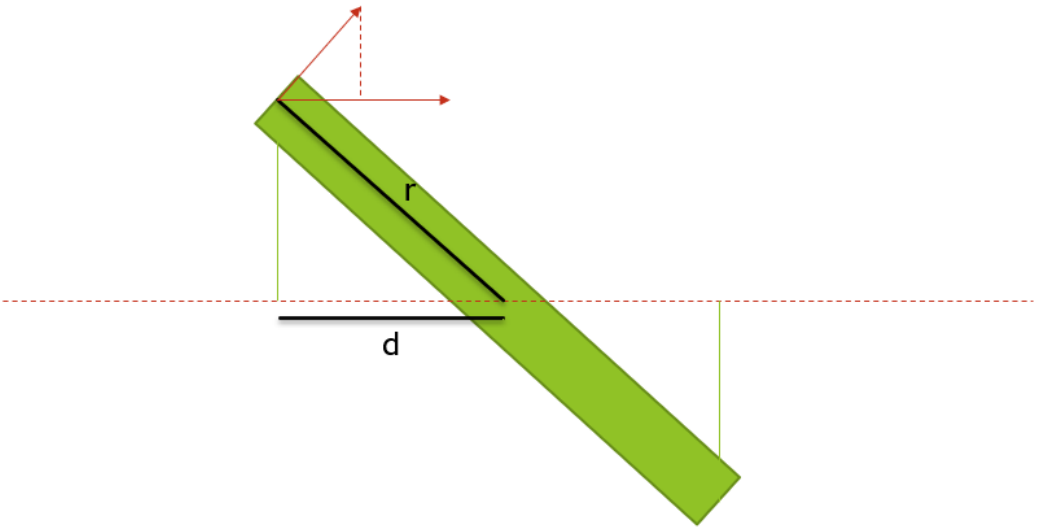


图 3-3 简单的旋转示意图

如图 3-3 所示的简单的模型旋转示意图。对于某个点，设其与旋转轴的距离为 r，模型旋转角速度为 α 。设该点横向速度为 v，可以计算出该点的横向速度。

$$v = \alpha \frac{\sqrt{r^2+b^2}}{r} \qquad (3\text{-}1)$$

而该点对与旋转轴距离 d 处的图像上的贡献是出现的频率，出现频率与横向速度成负相关。设出现频率为 p，则对于 p 的计算为：

$$p = \frac{1}{v} \qquad (3\text{-}2)$$

根据以上分析，可以计算出模型上该点对与旋转轴距离的 d 的图像上的贡献。而对于图像上的点来说，它的灰度值是所有能在该处投影的模型上的点的贡献的综合。因此这是个求积分的过程。设该点像素值为 t，那么 t 的计算如下：

$$t = \int_d^r \frac{x}{\alpha\sqrt{x^2+b^2}} \qquad (3\text{-}3)$$

以此可以推到每个像素点的值与模型之间的关系。但是该映射关系并不完美，因为并非每个点对二维图像上的像素值都有贡献。因为存在遮掩现象，即模型的某些地方掩盖住了其他的地方，导致从人眼看不到某些属于模型的部分。也就是说这些部分是不会提供贡献的。因此需要将这些地方排除，或者将提供贡献的地方单独拿出来。所以在上述公式中，添加一定的条件。对每个点在计算贡献前判断对应位置是否已经被覆盖。如果已经被覆盖就不进行贡献的计算。

有了从模型到图像的映射关系，但是从图像到模型的映射关系还需要再继

续摸索，并不能直接对该公式进行颠倒从而得出从二维图像到三维模型的映射关系。

3.3 相关实验及规律总结

本节介绍模型旋转实验带来的关于模型与图像之间的一些规律。

实验模型如图 3-4，这是一个比较典型的旋转实验，通过控制每一层的角度来控制每一层在旋转中出现的频率。

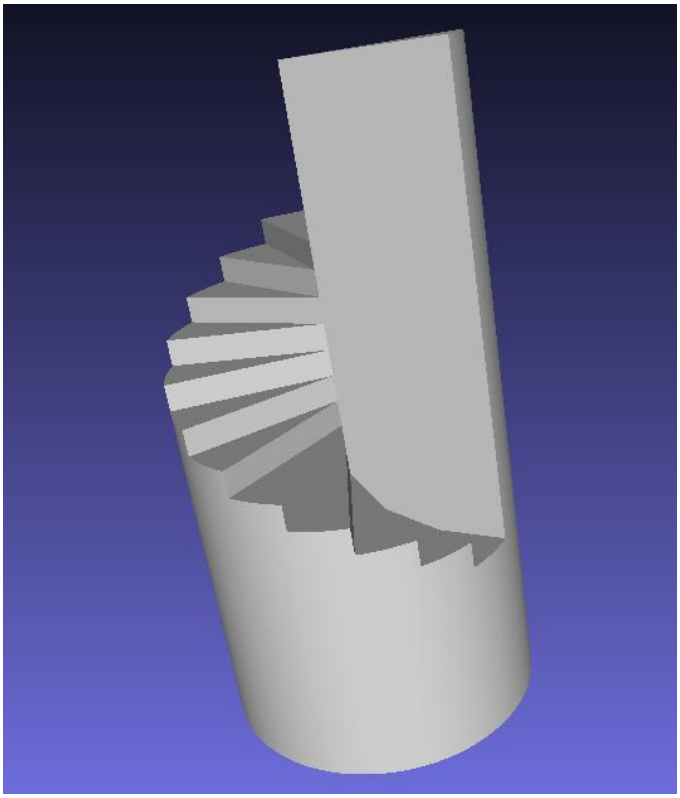


图 3-4 实验模型图

在实验之前，对该模型的旋转预测如图 3-5。灰度值是从上到下越来越深的，这是很符合常理的。然而实际上并非如此，在人眼观察下的图像比较难以描述并且不好记录。通过相机拍摄并不能得到人眼的效果，指挥出现某一个角度的模型图片，因此并不放相机拍摄的人眼观测图。本质上还是人眼是连续的接收图像信息，然后由人脑进行加工得到的人眼观测残影图。相机的拍摄包括视频的录制都不能反映出人眼观察的真实模样。

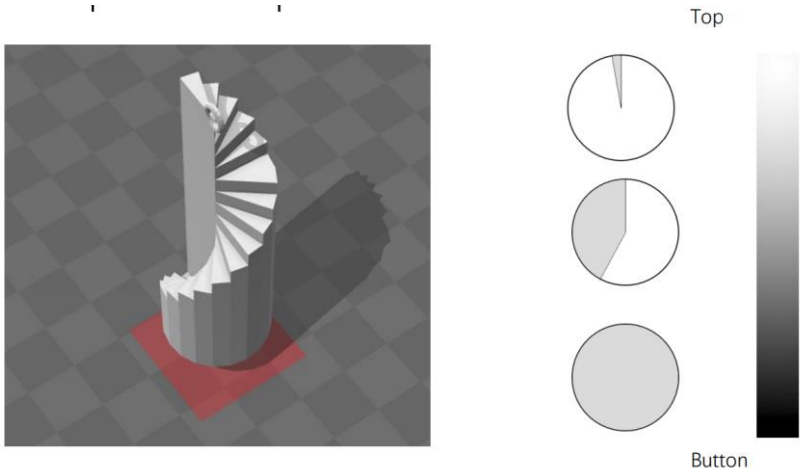


图 3-5 旋转预测图

在此描述实际旋转得到的图像。首先，人眼的分辨能力并没有那么突出，其实残影部分很难看出 256 个灰度值。只能清晰的分辨出实体区域与残影区域。对残影区域并不能做细致的区分。所以在**扇形角度小于 180 度**的模型上方区域，都是残影但是并不能很好的区分深度值。这里提及小于 180 度的上方区域是因为在**扇形角度大于 180 度**的下方区域有与预测不一样的地方，在下方区域每一层都有实体区域。如图 3-6 两条红色线之间就是实体区域，不管从哪个角度看都是存在模型的。因此可以得出一个新的规律，大于 180 度角的扇形在旋转时必定会出现一个实体区域，随着角度的越来越大，实体区域也就越来越大。

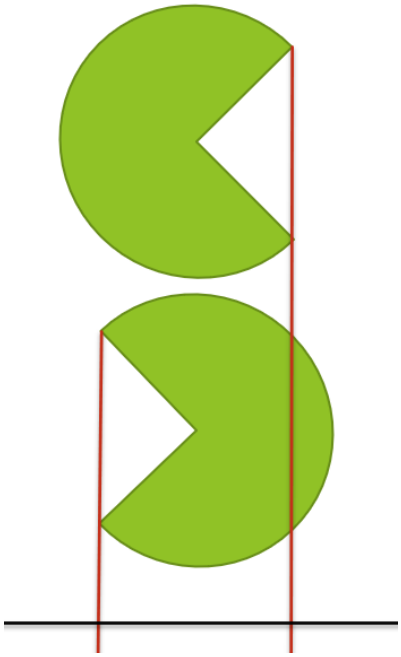


图 3-6 扇形实体区域演示图

3.4 模型生成算法

通过本章上述对问题的分析，以及规律的总结。最后给出得到的一些模型生成算法。

通过上述规律的探索，可以得知，不存在完全空洞的内部区域，存在外部实体区域的话，内部肯定是会有残影的。并且有从模型到图像的映射公式可以看出，横向速度越慢对残影的贡献越高。因此可以得到一种简单的模型生成方法，让模型正面看起来和给出的输入图像一样。当然，图像是有灰度的。而模型是没有灰度值的，是二值的存在模型或不存在模型。因此需要将图像转换成黑白图像，通过直方图均衡化让分布更均衡后再调整阈值分成二值图像[20]。然后让模型正面与该二值图像保持一致，再调整模型的厚度。先给定一个初始值，然后根据反馈来适当调整局部。如果原图像此处灰度值较低，则可以减小厚度值，反之亦然。

通过这种方法生成的模型在高速旋转之下与图像有一定程度的相似性。

另一种模型生成方法是假定我们的验证算法十分准确，也就是人眼拟真算法十分准确的情况下。从欺骗验证算法来使得验证算法对模型生成算法的结果产生很好的结果。目前，验证算法是均分成 24 个方向进行映射融合。也就是说如果模型在 24 个方向看都与图像相似，就可以欺骗验证算法，达到从算法角度很优秀的生成算法。但是目前通过该方法得到的生成算法存在一些问题。上述操作之后会使三维模型不连通，以及验证算法并不是那么优秀，结果也只是有一定程度的相似性。

第4章 结束语

4.1 本文总结

本节主要对全文进行总结。介绍具体的工作以及相关贡献，分析目前仍然存在的一些问题，探讨后续如何进行改进以及相关工作的展望。

4.1.1 主要工作及贡献

在关于旋转艺术上的研究普遍都倾向于从力学方向研究旋转物体自身的平衡性上的背景下，本文提出了另一方面的旋转艺术研究。利用人眼暂留效应，从不规则物体旋转会产生残像的现象出发，提出了从图像到三维模型的设计。让三维满足高速旋转下产生的残影与给定图像相似。主要工作如下：

1. 算法框架：本文设计并实现了一套针对从图像到三维模型在代码方面的流程框架。该算法框架主要由模型生成算法、人眼拟真算法和转换算法构成。模型生成算法是主要目标，其他部分均为更好的实现或验证该算法服务。人眼拟真算法是研究从模型到图像的算法，对给定模型模拟在高速旋转下人眼看起来的效果。转换算法是将三维网格数据转换为可打印的三维模型数据。
2. 实验平台：本文对具体的高速旋转模型搭建了线下的平台，通过 uno 板与步进电机完成了高速旋转平台的搭建。通过 andrinoIDE 编写程序完成对步进电机的控制，实现可控速的高速旋转。
3. 数学分析：本文对模型的高速旋转建立了可靠的数学模型，将模型映射到图像的过程转换成了数学公式，可通过该数学模型完成三维模型到二维图像的映射。并且通过有层次的分析探讨了亏格对于模型映射图像之间的关系。
4. 简单生成算法的提出：本文通过对该问题进行分析，以及通过实物实验掌握了一些规律，提出了两种模型生成算法。第一种是保持模型正面与图像高度相似，再通过具体灰度值的不同来设置不同的厚度。第二种是通过欺骗验证算法，将模型 24 个角度的画面均与图像相似。两种算法都有一定的效果也都有各自的缺点，这些在接下来的不足中详细介绍。

本文完成了相关的框架搭建工作，对本课题或本课题的延申课题做出了一定的贡献。首先，对于该课题的提出本身就是对于旋转艺术的一种贡献，从一个新颖的角度研究旋转的艺术。针对于我们的工作对本课题的贡献，主要是完成了一些基础项目平台的初步搭建，对该课题数学方面的分析，并且给出了一定的数学模型，提出了两种模型生成算法。

4.1.2 存在的不足

其实在本文对课题的工作中已经说明，本文对课题的相关工作已经完成的比较全面，整体比较完整。但是细分来说，存在一些不足。虽然整体项目工作比较完整，但是不够完善。最后的生成算法不够优秀，之间的人眼拟真算法，转换算法也有完善的空间。

对于模型生成算法，本文提出了两种模型生成算法。第一种是保持模型正面与图像高度相似，再通过具体灰度值的不同来设置不同的厚度。第二种是通过欺骗验证算法，将模型 24 个角度的画面均与图像相似。第一种算法是比较容易想到了简单的生成算法，也有一定的优秀程度。但是对于复杂图像不好处理，残影区域和实体区域的范围也存在与图像不匹配的情况。第二种生成算法理论上的效果应该很好，但是会受到验证算法的影响，毕竟是针对验证算法反向提出了生成算法，验证算法不够优秀必然会对该生成算法造成影响。另一方面，第二种生成算法会使生成的模型不够连通，会存在无法打印的情况。

关于其他部分，都属于有一定的完成度，可以保证流程的完整性，存在优化空间。人眼拟真算法很难拟真出人眼看到的残像，通过照片或者视频的拍摄也达不到人眼观测得到的残像图，因为人眼与人脑的机制比较复杂，并且相关专业知识不够丰富，所以人眼拟真算法存在很大的优化空间。转换算法得出了模型过于棱角分明，作为艺术品不够圆润。

4.1.3 总结

综上所述，本文提出了关于旋转艺术方面的新颖的课题。并且有一定的完成度，对后续的深入研究打定了基础。完成了基础项目平台的搭建，在后续工作 zhon 给可以沿用或对其进行优化。提出了相关分析的数学模型，对完善课题提供了一定的思路。提出了两种模型生成算法，可用于继续优化或者引出新的模型生成算

法。当模型生成算法达到一定的优秀程度后有可能成为一种艺术潮玩，在日常生活中随处可见。

4.2 工作展望

本文提出并完成了一种新的旋转艺术，但是仍然存在一些不足之处须在后续工作中进行完善。

在人眼拟真算法方面，本文由于缺乏相关专业知识，只能完成比较简单的算法。后续对人眼拟真算法的优化方面会多查阅相关资料，了解人眼人脑的运行机制，从物体本身的旋转怎样变成人眼所看到的图像的具体机制。只有这样才能更好的完善人眼拟真算法。补充所缺少的知识是一方面，另一方面需要进行更多的实验，通过实践来完善算法。与普通的照片不同，人眼残像实际上是需要物体不断运动的，高速旋转下的物体更是如此，因此人眼所看到的真实图像是有一定的模糊程度，不好描述也很难记录下来。这也是另一个需要攻克的难点。目前还没有很好的解决办法。

在转换算法方面，将三维网格转换成可打印格式，本文是通过将网格点拓展成六面体。这样会导致三维模型棱角太过分明，这也是一种艺术形式，但是会与某些图像相性不好。针对不同风格的图像，或者是不同风格的模型，打印出来的效果不应该都是这种方方正正的。这方面需要通过大量的实验来具体分析不同的情况，考虑如何对三维网格数据的边界做柔化。

在生成算法方面，本文提出了两种模型生成算法，但是都有缺陷。后续工作的重点或者是主要目标也是对模型生成算法的优化。对于提出新的模型生成算法或是对现有的模型生成算法进行优化，目前主要由两个方面的思路，一方面是继续通过数学分析的方法，将从三维模型到图像的关系逆推得到图像到三维模型的关系，如果该想法有一定的可靠性，那么能得到很好的模型生成算法。另一方面就是继续完善验证算法，也就是人眼拟真算法，然后尝试找出欺骗验证算法，让代码识别出来生成算法优秀程度很高。这样也能得到很好的模型生成算法。

对该工作未来的前景。如果该课题得到更进一步的完善，可以预见将会成为一种新的艺术作品，应用于时尚潮玩。在相对可靠的模型生成算法产生之后，打印出来的模型将成为很有意思的艺术品。每个人可以自定义自己喜欢的图像，生

成相应的模型，然后旋转就能看得到。而旋转是在每个方向看到的图像都是一样的，也就是说，从任何方向看都能看到你所定义的图像，这很有趣，也很有潜力成为人们办公桌上、床头柜旁的饰品，发展前景十分可观。

致　谢

在本文写作和本文算法的实现过程中，首先我要感谢我的导师赵海森老师。赵老师在本文的写作和算法实现过程中，给予我很多非常有益的建议和指导帮助我解决碰到的问题。

同时，感谢计算机学院的钟凡老师和吕琳老师，在图形学和计算机视觉上的教导让我在本课题的思路和算法实现上有了很多专业的想法，少走路很多弯路。

最后，我要感谢我的父母及家人，是他们在生活上无微不至的关怀和学习上始终如一的支持，使我完成顺利学业。我衷心的感谢他们，同时也祝愿他们幸福平安。

我衷心的祝愿所有帮助过我的人健康、平安、快乐！

参考文献

[1] Moritz Bächer,Bernd Bickel,Emily Whiting,Olga Sorkine-Hornung.Spin-it[J].Communications of the ACM,2017,60(8).
[2] 贾菡.计算机图形学之父——伊凡·苏泽兰特[J].程序员,2005,(11):9.
[3] Bentley George.David Evans (1925 to 2020).[J].The bone & joint journal,2021,103-B(7).
[4] Miscellaneous Electrical Machinery, Equipment and Supplies; Evans & Sutherland Computer Corp. Files SEC Form 10-Q, Quarterly Report [Sections 13 Or 15(D)] (Nov. 7, 2014)[J].Computer Weekly News,2014.
[5] Ed Catmull to Keynote at Upcoming Conference[J].Wireless News,2014.
[6] Blinn, J. F., & Newell, M. E. (1976). Texture and reflection in computer generated images. Communications of the ACM, 19(10), 542-547.
[7] Mittelman, P., & Goldstein, R. (1976). Operational Aspects of the Syntha Vision Process. SMPTE Journal, 85(8), 636-638.
[8] Zhang, H., Manocha, D., Hudson, T., & Hoff III, K. E. (1997, August). Visibility culling using hierarchical occlusion maps. In Proceedings of the 24th annual conference on Computer graphics and interactive techniques (pp. 77-88).
[9] Reinhart, C., & Breton, P. F. (2009). Experimental validation of Autodesk® 3ds Max® Design 2009 and DAYSIM 3.0. Leukos, 6(1), 7-35.
[10] Chemerys, H., Osadcha, K., Osadchyi, V., & Kruhlyk, V. (2019). Increase of the Level of Graphic Competence Future Bachelor in Computer Sciences in the Process of Studying 3D Modeling. In ICTERI Workshops (pp. 17-28).
[11] Roget, P. M. (1825). V. Explanation of an Optical Deception in the Appearance of the Spokes of a Wheel Seen through Vertical Apertures. Philosophical Transactions of the Royal Society of London, (115), 131-140.
[12] 王立文,王剑薇.视觉暂留[J].中国科技教育,2019,(02):52-53.
[13] Carpenter, W. B. (1862). Introduction to the Study of the Foraminifera: By William B. Carpenter, Assisted by William K. Parker and T. Rupert Jones. Publ. for the Ray Society (Vol. 22). Hardwicke.
[14] Wertheimer, M. (1938). Gestalt theory.
[15] Dimmick, F. L. (1920). An experimental study of visual movement and the phi phenomenon. The American Journal of Psychology, 31(4), 317-332.
[16] Siegel, R. L., Miller, K. D., Goding Sauer, A., Fedewa, S. A., Butterly, L. F., Anderson, J. C., ... & Jemal, A. (2020). Colorectal cancer statistics, 2020. CA: a cancer journal for clinicians, 70(3), 145-164.

[17] Stupp, R., Mason, W. P., Van Den Bent, M. J., Weller, M., Fisher, B., Taphoorn, M. J., ... & Mirimanoff, R. O. (2005). Radiotherapy plus concomitant and adjuvant temozolomide for glioblastoma. New England journal of medicine, 352(10), 987-996.
[18] 邓汉元, 黄元秋. 图的最大亏格、支配数和围长[J]. 高校应用数学学报, 2001, 16(1):15-20.
[19] 刘彦佩. 图的不可定向最大亏格[J]. 中国科学, 1979(S1):193-203.
[20] 李乐鹏,孙水发,夏冲,陈鹏 & 董方敏.(2014).直方图均衡技术综述. 计算机系统应用(03),1-8.

附录 1 代码及相关附件

```
#define _CRT_SECURE_NO_DEPRECATE

#include <igl/cotmatrix.h>
#include <Eigen/Dense>
#include <Eigen/Sparse>

#include <iostream>
#include <math.h>
#include <opencv2/opencv.hpp>
#include <opencv2/highgui.hpp>
#include <opencv2/highgui/highgui_c.h>

using namespace std;
using namespace cv;
using namespace Eigen;

// 1000 * 1000;

const int N = 1000;
const int LEFT = 490;
const int RIGHT = 510;

//Matrix<Matrix<uchar, 1, 70>, 70, 70> model;
uchar model[N][N][N];

//三维模型不同角度变到二维人眼视角
vector<vector<uchar> > Three2Two(double theta);
```

```
//将多角度二维影像合成
vector<vector<uchar> > Merge(vector<vector<vector<uchar>>> Alist);

//损失函数
void Loss(MatrixXd A , MatrixXd B);

//将模型写入文件
void Write(int l , int r);

//模型生成方法test0
void Test0(Mat input);

//模型生成方法test1
void Test1(Mat input);

//模型生成方法test2
void Test2(Mat input);

int main()
{
    Eigen::MatrixXd V(4, 2);
    V << 0, 0,
         1, 0,
         1, 1,
         0, 1;
    Eigen::MatrixXi F(2, 3);
    F << 0, 1, 2,
         0, 2, 3;
```

```
    Eigen::SparseMatrix<double> L;
    igl::cotmatrix(V, F, L);
    std::cout << "Hello, mesh: " << std::endl << L << endl << L * V
<< std::endl;
    string path = "D:\\Study\\Graduation
Design\\Project\\P1\\test\\1.jpeg";
    Mat input = imread(path);

    //imshow("input", input);
    //waitKey(0);
    //return 0;

    cvtColor(input, input, CV_BGR2GRAY);

    //1、完成三维数组到平面的映射

    //2、完成连续映射到人眼


    //imshow("input", input);
    //cout << (int)input.at<uchar>(3, 3) << endl;
    //waitKey(0);
    //return 0;

    int l = input.cols;
    int r = input.rows;

    for (int i = 0; i < r; ++i) {
        for (int j = 0; j < l; ++j) {
```

```
            if (j > l / 2)
                input.at<uchar>(i, j) = input.at<uchar>(i, l - j);
        }
    }

    //生成模型
    Test0(input);
    //Test1(input);

    cout << "生成完毕" << endl;

    Write(r, l);

    cout << "写入完毕" << endl;

    vector<vector<vector<uchar>>> list;
    for (double i = 0; i < 360; i += 10) {
        list.push_back(Three2Two(i));
    }

    cout << "转换完毕" << endl;

    vector<vector<uchar>> result;
    result = Merge(list);

    cout << "合并完毕" << endl;

    Mat output(1000, 1000, CV_8U, 1);
```

```
    for (int i = 0; i < 1000; ++i) {
        for (int j = 0; j < 1000; ++j) {
            output.at<uchar>(i, j) = result[i][j];
        }
    }

    //outFile.close();
    imshow("input", input);

    imshow("output", output);

    waitKey(0);

    return 0;
}

//将模型写入文件
void Write(int l , int r) {
    FILE* fp;
    if ((fp = fopen("model.obj", "w")) == NULL) {
        printf("cant open the file");
        exit(0);
    }

    //方向
    int fx[6][3] = { {0 , 0 , 1} , {0 , 0 , -1} , {0 , 1 , 0} ,
        {0 , -1 , 0} , {1 , 0 , 0} , {-1 , 0 , 0} };
```

```
    //写入点的序号
    int point = 1;

    for (int i = 2; i < l - 1; ++i) {
        for (int j = 2; j < r - 1; ++j) {
            for (int k = LEFT + 1; k < RIGHT; ++k) {
                //判断该点是否存在
                if (model[i - l / 2 + 500][j - r / 2 + 500][k] !=
255) {
                    int x = i - l / 2 + 500;
                    int y = j - r / 2 + 500;
                    int z = k;
                    for (int p = 0; p < 6; ++p) {
                        //按照六个方向判断，该方向上是否有相邻的点
                        //如果有相邻的点，那么该面就没有绘制的意义
                        if (model[x + fx[p][0]][y + fx[p][1]][z +
fx[p][2]] == 255
                            || z + fx[p][2] <= LEFT || z +
fx[p][2] >= RIGHT
                            ) {
                            double px[4][2] = { {-0.5 , -0.5} , {-
0.5 , 0.5} ,
                                {0.5 , 0.5} , {0.5 , -0.5} };
                            if (fx[p][0] != 0) {
                                for (int q = 0; q < 4; ++q) {
                                    fprintf(fp, "v ");
                                    fprintf(fp, "%lf ", (double)x -
500 + (double)fx[p][0] / 2.0);
                                    fprintf(fp, "%lf ", (double)y -
```

```
500 + px[q][0]);
                                    fprintf(fp, "%lf\n", (double)z -
500 + px[q][1]);
                                }
                            }
                            else if (fx[p][1] != 0) {
                                for (int q = 0; q < 4; ++q) {
                                    fprintf(fp, "v ");
                                    fprintf(fp, "%lf ", (double)x -
500 + px[q][0]);
                                    fprintf(fp, "%lf ", (double)y -
500 + (double)fx[p][1] / 2.0);
                                    fprintf(fp, "%lf\n", (double)z -
500 + px[q][1]);
                                }
                            }
                            else {
                                for (int q = 0; q < 4; ++q) {
                                    fprintf(fp, "v ");
                                    fprintf(fp, "%lf ", (double)x -
500 + px[q][0]);
                                    fprintf(fp, "%lf ", (double)y -
500 + px[q][1]);
                                    fprintf(fp, "%lf\n", (double)z -
500 + (double)fx[p][2] / 2.0);
                                }
                            }
                            fprintf(fp, "f %d %d %d %d\n" , point++ ,
point++ , point++ , point++);
```

```
                        }
                    }
                }
            }
        }
    }
    fclose(fp);
}

//三维模型不同角度变到二维人眼视角
vector<vector<uchar> > Three2Two(double theta) {
    vector<vector<uchar>> result(1000, vector<uchar>(1000, 0));

    for (int i = 0; i < 1000; ++i) {
        for (int j = 0; j < 1000; ++j) {
            for (int k = 0; k < 1000; ++k) {
                if (model[i][j][k]) {
                    double pp;
                    if (j == 500) {
                        if (k > 500) {
                            pp = (90 - theta) * 3.14159 / 180.0;
                        }
                        else {
                            pp = (270 - theta) * 3.14159 / 180.0;
                        }
                    }
                    else {
                        pp = atan((double)(k - 500) / (double)(j -
500)) - theta * 3.14159 / 180.0;
```

```
                    }

                    result[i][int((j - 500) * cos(pp) + 500)]
                    = (int)model[i][j][k];
                }

            }
        }
        //cout << i << endl;
    }

    return result;
}

//将多角度二维影像合成
vector<vector<uchar> > Merge(vector<vector<vector<uchar>>> Alist) {
    vector<vector<uchar>> result(1000, vector<uchar>(1000, 0));

    for (auto t : Alist) {
        for (int i = 0; i < 1000; ++i) {
            for (int j = 0; j < 1000; ++j) {
                result[i][j] += (int)t[i][j] / 36;
                //cout << (int)t[i][j] << ' ';
            }
        }
    }

    return result;
}
```

```
//模型生成方法test0
void Test0(Mat input) {
    int l = input.cols;
    int r = input.rows;

    for (int i = 0; i < r; ++i) {
        for (int j = 0; j < l; ++j) {
            for (int k = 0; k < 1000; ++k) {
                if (k > 490 && k < 510) {
                    model[i - r / 2 + 500][j - l / 2 + 500][k] =
                        (int)input.at<uchar>(i, j) > 128 ? 255 : 0;
                    if ((int)input.at<uchar>(i, j) == 0) {
                        model[i - r / 2 + 500][j - l / 2 + 500][k] =
0;
                    }
                }
                else {
                    //model[i - l / 2 + 500][j - r / 2 + 500][k] =
255;
                }
            }
        }
    }
}

//模型生成方法test1
void Test1(Mat input) {
    int l = input.cols;
```

```
    int r = input.rows;

    for (int i = 0; i < 1; ++i) {
        for (int j = 0; j < r; ++j) {
            for (int k = 0; k < 1000; ++k) {
                if (k > LEFT && k < RIGHT) {
                    model[i - 1 / 2 + 500][j - r / 2 + 500][k] =
                        (int)input.at<uchar>(i, j) > 128 ? 255 : 0;
                    if ((int)input.at<uchar>(i, j) == 0) {
                        model[i - 1 / 2 + 500][j - r / 2 + 500][k] =
0;
                    }
                }
            }
        }
    }


}

//损失函数
void Loss(MatrixXd A, MatrixXd B) {
    int loss = 0;
    for (int i = 0; i < 1000; ++i) {
        for (int j = 0; j < 1000; ++j) {
            loss += (A(i, j) - B(i, j)) * (A(i, j) - B(i, j));
        }
    }
}
```