

分类号 _____

密级 _____

兰州大学

研究生学位论文

论文题目（中文） 3D 虚拟人手的构建与实时交互

论文题目（外文） Construction of Virtual 3D Life-like Human

Hand for Real-time Human-Computer Interaction

研究 生 姓 名 _____ 任帅

学 科、专 业 _____ 信息与通信工程 · 信号与信息处理

研 究 方 向 _____ 计算机图形学 · 3D 虚拟仿真

学 位 级 别 _____ 硕士研究生

导师姓名、职称 _____ 万毅教授

论文工作起止年月 _____ 2014 年 3 月 至 2015 年 3 月

论 文 提 交 日 期 _____ 2015 年 3 月

论 文 答 辩 日 期 _____ 2015 年 5 月

学 位 授 予 日 期 _____

校址：甘肃省兰州市

原创性声明

本人郑重声明：本人所呈交的学位论文，是在导师的指导下独立进行研究所取得的成果。学位论文中凡引用他人已经发表或未发表的成果、数据、观点等，均已明确注明出处。除文中已经注明引用的内容外，不包含任何其他个人或集体已经发表或撰写过的科研成果。对本文的研究成果做出重要贡献的个人和集体，均已在文中以明确方式标明。

本声明的法律责任由本人承担。

论文作者签名: _____ 日期: _____

关于学位论文使用授权的声明

本人在导师指导下所完成的论文及相关的职务作品，知识产权归属兰州大学。本人完全了解兰州大学有关保存、使用学位论文的规定，同意学校保存或向国家有关部门或机构递交论文的纸质版和电子版，允许论文被查阅和借阅；本人授权兰州大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用任何复制手段保存和汇编本学位论文。本人离校后发表、使用学位论文或与该论文直接相关的学术论文或成果时，第一署名单位仍然为兰州大学。

本学位论文研究内容：

- 可以公开
 不宜公开，已在学位办公室办理保密申请，解密后适用本授权书。

(请在以上选项内选择其中一项打“√”)

论文作者签名: _____ 导师签名: _____

日期: _____ 日期: _____

3D 虚拟人手的构建与实时交互

摘要

虚拟人手在人机交互应用中扮演着非常重要的作用。通过虚拟人手我们能够在虚拟环境中得到更好的体验。但是人手是一个非常复杂的结构，模拟一个逼真的 3D 虚拟人手不容易，尤其是为其创建一个灵活的皮肤纹理。现有的虚拟人手的制作都是采用一般的 3D 模型，并不具有纹理信息。本文提出了一种新的方法可以自动的对 3D 虚拟人手进行纹理贴图，并能够将其实时的应用到人机交互当中，比如与虚拟环境进行实时交互。

本文的具体做法是首先根据拍摄得到的背面图像调整人手模型，并调整正面图像使其能够能够与背面图像的镜像相重合。之后需要对正面图像进行色彩传递使其可以和背面图像的颜色相似。通过选定裁剪框，我们使用正交投影建立映射函数，从而得到对应纹理坐标处的颜色值。如果直接使用两张图像进行纹理贴图则会在手指的边缘产生明显的接缝，因此我们需要使用平滑着色去除接缝。通过为人手模型添加骨骼结构，可以使其能够像真人手一样做出各种姿态，从而与虚拟环境进行实时交互。本文提出的方法非常有效，实验结果表明生成的个性化人手比传统的虚拟人手更加逼真，为我们在虚拟环境中带来更强的视觉冲击感。

关键词：人机交互，虚拟人手，纹理贴图，实时交互

Construction of Virtual 3D Life-like HumanHand for Real-time Human-Computer Interaction

Abstract

Virtual hand plays an important role in many human-computer interaction applications. We can get a better experience in a virtual environment through a virtual hand. However, hand is a very complex structure, modeling a life-like 3D virtual hand is not a trivial task, especially the creation of flexible skin textures. The existing virtual hand production are based on the general hand of the 3D models, without the texture information. In this paper, we propose a new method to execute texture mapping for the 3D virtual hand model automatically and apply it for human-computer interaction in real-time.

Specifically, We first adjust the palm image using the mirror of the dorsal image. Then color transfer is performed for the palm image to ensure its similar color as the dorsal image. By selecting the crop box, we can establish the mapping function using orthogonal projection, the colors can be gotten from the texture coordinates. Finally smooth shading is used to further adjust the direct texture mapping considering the virtual seams in fingers and thumb side. The proposed method is very efficient and experimental results show that the generated virtual hand model is more realistic than conventional virtual hands and can yield a stronger sense of immersion in the virtual world.

Keywords: human-computer interaction,virtual hand model,texture-mapping,real-time interactivity

目 录

摘要	i
Abstract	ii
第一章 绪论	1
1.1 研究背景与意义	1
1.2 研究发展现状	2
1.3 论文的主要工作和章节安排	3
第二章 手部解剖结构	5
2.1 骨骼结构	5
2.2 手指间运动的约束性	7
2.3 本章小结	8
第三章 虚拟人手的自动纹理贴图	9
3.1 3D 人手模型的调整	9
3.2 正面图的调整	10
3.3 色彩传递	12
3.3.1 RGB 颜色模型	13
3.3.2 $l\alpha\beta$ 颜色模型	14
3.3.3 基于 $l\alpha\beta$ 颜色空间的色彩传递	14
3.4 正反面顶点分离	16
3.4.1 深度排序算法	16
3.4.2 深度缓存算法	19
3.4.3 光线跟踪算法	21
3.4.4 顶点分离	22
3.5 纹理映射	24

3.5.1	投影变换	25
3.5.2	视口变换	28
3.5.3	一般纹理贴图	28
3.5.4	对手贴纹理	29
3.6	接缝移除	32
3.6.1	平面着色	32
3.6.2	平滑着色	33
3.6.3	基于平滑着色去除接缝	34
3.7	本章小结	36
第四章	虚拟手的实时交互	37
4.1	骨骼实现人机交互	37
4.2	实验结果与分析	40
4.3	本章小结	41
第五章	工作总结与展望	42
5.1	论文工作总结	42
5.2	研究展望	42
发表文章目录	44	
致谢	45	

第一章 绪论

1.1 研究背景与意义

虚拟现实作为一种新兴技术正快速的改变着人类对物体的认知过程和方式。其以沉浸性、交互性、构想性 (3I) 而成为目前非常活跃的一个技术领域。它为无人机交互的发展开创了一种全新的研究领域，并在航空航天、建筑设计、影视娱乐、医学和教育等方面都起着非常广泛的作用。它通过对用户视觉、听觉、触觉等感官的模拟，使用户仿佛置身于虚拟世界之中，并可以与虚拟环境进行主动的交互。而虚拟人手的构建与实时交互作为虚拟现实技术的一个子课题得到越来越多学者的关注。

科学研究表明，人手是人身体最重要的三大器官之一，我们在生活中进行的大量操作都是直接或间接通过人手完成的。它在我们的日常生活中扮演着非常重要的角色，并且它是人类在视觉方面最重要的组成部分。通过人手我们可以方便的感知物体、抓取物体从而让我们更加准确和快速的完成复杂的任务。模拟和控制虚拟人手是虚拟现实中一个非常重要的课题，在虚拟现实中虚拟人手扮演着与真实人手同样的角色（比如 [?]）。

通常我们都是使用键盘、鼠标或者手持式装置去与计算机进行交互，这些设备都比较笨重并且不稳定。而虚拟人手为我们提供了一个更加具有前景的替代方案，通过虚拟人手我们能够更加直观和方便的控制计算机。因此目前虚拟人手被广泛应用于手语教学、三维游戏，人机交互以及医学实验等领域。如果能够模拟出一个逼真的虚拟人手将会为我们带来更加强烈的视觉冲击，从而让我们能够在虚拟世界得到更好的体验。

虚拟人手的逼真性和准确性直接影响了仿真分析结果的可靠性以及用户的体验度。但是人手是一个非常复杂的结构，整个人手由 27 根骨骼组成，约占人身体骨骼的四分之一，并且由大量的血管、神经、肌腱以及韧带等组成，因此想要模拟出一个逼真人手变得非常的困难。随着计算机硬件水平的提高，在虚拟现实应用无处不在的今天，模拟出一个非常逼真的人手无论在理论研究还

是产业化应用方面都具有非常重大的意义。

1.2 研究发展现状

尽管人手在我们日常生活中随处可见，但是因为其巨大的复杂性，在计算机图形学应用中人手的模拟并未得到广泛的关注，尤其是在国内并未得到众多学者的研究。但是由于人手的重要性以及科技水平的不断提高，想必在不久的未来这种现状将会得到改善。

想要构建出一个逼近真实的虚拟人手，第一步需要做的是建模。目前的建模技术主要有 3D 扫描，三维软件建模以及基于图像等几种方法。3D 扫描仪以其高精度的优势而得到大量用户的关注，但是因受成本、空间的限制而只被一些企业所应用。基于图像的三维建模技术是通过拍摄一组采样图像并通过拼接融合等方法将二维图像恢复成景物的三维几何结构的技术，如 Apple 公司推出的 QuickTime VR 系统，以及 Autodesk 公司推出的 Autodesk 123D 软件，但是因为技术的限制通过这种方法生成的 3 维模型还不能精确的反映出物体的结构。而目前使用最多的是通过三维软件直接建模的方式，如目前被广泛使用的 3ds max, maya 以及 Auto CAD 等软件，通过这些软件用户可以直接使用点和线来构建出想要的三维模型，这种建模技术因其成本低，简单实用而得到广泛应用。

虚拟人手最重要的目的是为了让用户与计算机进行交互，这方面的研究依然在激烈的进行着。在 [?] 中，Jieun Lee 等人提出一种新的方法可以对虚拟人手进行逼真的变形，并通过碰撞检测与自碰撞检测技术实时的与物体进行交互。Paul G.Kry 等人则是采用 EigenSkin 的方法允许虚拟人手进行细微的非线性准静态变形 [?]。考虑到真实人手的运动学和生物力学特征，Mamy Pouliquen 等人在 [?] 提出了一种基于物理的人手模型从而能够方便的通过运动捕捉系统对虚拟人手进行操控。Matei Ciocarlie 等人则在 [?] 使用局部几何特性为手指与物体之间的接触建立了一种分析模型。考虑到接触力与物体的形变 [? ?]，虚拟人手被用来抓取弹性物体。在 3D 虚拟仿真技术中虚拟手套是最常用的工具，通过虚拟手套我们能够方便的与虚拟环境进行实时的交互 [? ? ? ?]。

虽然目前已经有一些虚拟人手的研究在进行，但是一个栩栩如生的虚拟人

手尚未充分开发并使用。如有些文献只是采用骨骼来去模拟虚拟手，更有一些实验简单的采用小球或者其他物体来代替人手。这很难让用户在虚拟世界中产生沉浸感，同时也降低了许多虚拟动作的准确度，比如抓取等。如果我们能够使用和用户看起来相同的个性化虚拟人手而不是一个通用的人手，这将会为我们在虚拟环境中漫游带来更好的体验。

普遍认为人手具有 27 个自由度之多，并且人手的构造非常复杂。手指之间的相关性，骨骼控制的复杂性以及皮肤易变形这些因素都严重的制约着一个逼真和精确的虚拟人手的构建。Irene Albrecht 等人 [?] 与 Taehyun Rhee 等人 [?] 均采用一张拍摄得到的人手图片去对通用人手进行变形从而得到个性化人手，但是他们并未实现纹理映射的自动化。纹理是物体最重要的特征之一，不同的人手通常具有完全不同的纹理，如果对虚拟人手进行纹理的渲染，我们将会很容易根据纹理去分辨不同的人手，并能够为我们在虚拟世界中带来更加强烈的视觉冲击。但是对不规则物体进行纹理映射同样面临着很多困难。我们通常需要计算三维物体的空间位置并且使用一个或多个映射函数将其映射到纹理空间中，然后使用纹理空间值从纹理中获得相应的值 [?]。另外如果直接采用拍摄得到的不同位置的纹理图对虚拟人手进行渲染时，由于光照以及拍摄角度等因素，将会在两幅图像的连接处产生明显的接缝，Kun Zhou 等人提出了一种纹理融合技术可以解决这种问题 [?]。

在虚拟人手与用户的交互方面，虚拟人手的实时交互性能并不高，大部分都仅仅是采用预定义的姿态或者是使用昂贵并且笨重的数据手套来完成。这样显然降低了虚拟手与虚拟环境的交互性。

1.3 论文的主要工作和章节安排

本论文提出了一种通过对一般人手进行自动纹理贴图从而生成个性化人手的方法并可以让虚拟人手实时的与虚拟环境进行交互。其具体步骤为：首先，拍摄真实人手的正面和背面图，并利用拍摄得到的背面图调整一般人手模型使其和背面图可以相重合。因为拍摄时可能由于拍摄角度以及手的晃动等问题造成拍摄得到的正面图并不能与背面图的镜像图完全重合，从而在直接进行纹理贴图时可能会贴到背景，因此我们需要根据背面图的镜像图调整正面图使他们

能够完全重合。其次，由于拍摄时的光照问题可能使得拍摄得到的正面图和背面图在颜色上有较大差异，这里我们采用 Reinhard 等人 [?] 提出的颜色转换方法对正面图进行处理使其可以和背面图颜色相近。再次，分别提取出 3D 手模型的正面点和背面点，并计算每个点对应到图像上的位置，从而对 3D 手模型进行贴图。并对两张图的接缝处进行平滑着色处理，使得接缝得到平滑过度。最后，我们为 3D 手模型添加骨骼，并对骨骼进行蒙皮处理，从而让虚拟人手能够与虚拟环境进行实时交互。

本文的组织结构如下：

第一章介绍了本论文的研究背景、意义以及目前国内外的研究现状。

第二章简要介绍了人手的解剖学结构以及真人手在运动上的限制，从而可以避免虚拟人手在与虚拟环境的交互过程中产生一些非自然的手势。。

第三章是本文的重中之重，其详细介绍了通过拍摄得到的手心图和手背图对 3D 手模型进行自动纹理贴图的方法。

第四章介绍了如何通过对 3D 手模型添加骨骼从而完成虚拟人手与虚拟环境的实时交互，对比了拍摄的真实人手图片以及具有同样姿态的虚拟人手，同时展示了虚拟人手与虚拟环境的交互。

第五章作为本文的最后一个章节，详细的总结了本文的研究工作，指出了本文的优势以及不足，并提出了未来需要做的改进以及研究方向。

第二章 手部解剖结构

人手作为人身体的三大器官之一，在人的日常生活中起着非常重要的作用。据了解，在哺乳类动物中，人类的手是独一无二的。比如人手的大拇指可以同其他手指对合，但是和人类生理结构最为接近的类人猿却做不到，因为他们的手指还不够柔韧 [?]。能够完美自如的运用自己的手指是人类科技和文化进步的关键所在。

为了模拟出逼真的人手模型，我们需要对人手的解剖结构进行深入的研究。同时为了避免在虚拟人手在与虚拟环境进行实时交互的过程中产生一些非自然的手势，我们还需要研究手指与手指之间在运动上的限制关系。

2.1 骨骼结构

人手是一个非常复杂的结构，它是由大量的骨骼、血管、神经、肌腱和韧带组成。因此人的手可以完成很多复杂和灵巧的动作，比如捏、握、抓等。人手是一个多关节系统，共包含 27 根骨骼 [?]，分别为 8 根腕骨，5 根掌骨和 14 根指骨组成。两只手加起来的骨骼数目约占人身体骨骼的四分之一。一个骨骼结构如图 1 所示。

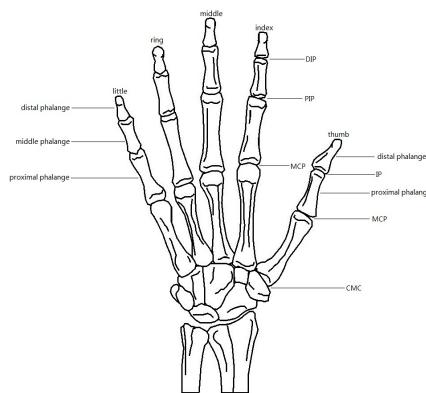


图 2.1: 手部骨骼结构

通常底部的 8 块腕骨在手的运动过程中只有微小的形状变化 [?]，因此在

对虚拟人手进行操控时，我们并不过度关注虚拟人手腕骨的运动。但是需要注意的是整个骨骼的坐标空间都是以腕骨骼为根节点进行的。掌骨的近侧端称为底，并与腕骨相连。而远侧端称为头，并与指骨相连。其中大拇指的掌骨较短较粗，但是比别的掌骨都更为灵活。指骨作用最大，我们通常进行的抓、捏等动作都是通过指骨来完成的。因此我们重点来研究指骨。

人手的食指、中指、无名指和小拇指都有三根指骨，分别是远指骨 (distal phalanges)、中指骨 (middle phalanges) 和近指骨 (proximal phalanges)[?]。每个指骨都是通过关节与别的指骨或掌骨进行相连，这些关节分别为远指骨关节 (distal interphalangeal(DIP))、近指骨关节 (proximal interphalangeal(PIP)) 和掌指关节 (metacarpophalangeal(MCP))。较为特殊的大拇指则只有两根指骨而没有中指骨，但它依然具有三个关节，这三个关节分别是普通关节 (IP)、掌指关节 (MCP) 和腕掌关节 (CMC)。

人手之所以那么灵活是因为其自由度很高。其中远指骨关节、近指骨关节以及大拇指的普通关节只在屈伸方向上具有一个自由度，而掌指关节不只在屈伸方向上有一个自由度，他们在内收和外展方向上也具有一个自由度，因此认为这些关节具有两个自由度。另外由于大拇指腕掌关节强大的运动能力，因此可以认为腕掌关节具有两个自由度。最后因为人手在空间中的运动我们认为它还具有另外六个自由度，因此人手总共具有 27 个自由度。图 2 为一个简略的人手自由度分布图：

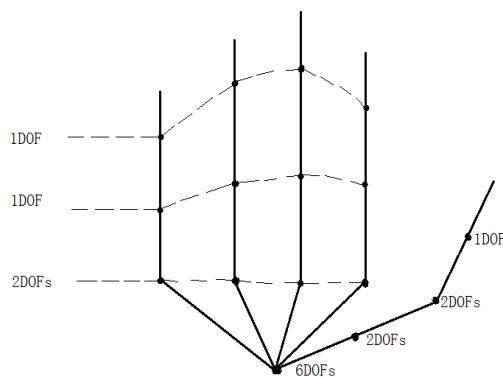


图 2.2: 人手自由度分布图

2.2 手指间运动的约束性

手指之间因为一些运动上的限制而不能做出任意的手势。比如当手指的运动超出一定的范围时，其能弯曲的角度必然会受到其他手指的限制。另外，手指并不能向背面过度弯曲。虽然有些非自然手势可以通过重复性的练习而做出来，但这种特殊情况在本章中并不加以考虑。我们主要研究人类手指运动的一般性情况。

另外虽然有一些手指之间的运动关系可以通过大量的研究来进行量化分析与定义，但有一些行为却很难从数学角度上进行分析，这些行为是人们约定俗成的一种观念。比如当我们由伸展开的手掌变成拳头时，通常会同时弯曲所有手指而不是逐个弯曲每根手指。再比如我们想用任意远指骨触碰手掌掌心时，在不弯曲相邻手指的前提下我们是难以达到这一要求的。虽然这些行为可能并不适合于所有人，但是它们却适用于绝大多数的人群。尽管无法量化这些行为，但我们仍然需要遵循这些行为。

首先是人手指关节之间的限制，通常我们认为这存在两种形式的限制。

第一种是单根手指上关节的限制 [?]，这种限制通常存在于远指关节与近指关节之间，当我们想活动远指关节时必然要用到与之相邻的近指关节，否则远指关节将很难单独运动。因此我们认为远指关节与近指关节存在一定的相关性。假设远指关节弯曲的角度为 θ_{DIP} ，而近指关节随之弯曲的角度为 θ_{PIP} ，这两个角度之间存在如下的线性关系：

$$\theta_{DIP} = \frac{2}{3}\theta_{PIP} \quad (2.1)$$

另一种是相邻手指关节的限制。这种限制通常存在于掌指关节之间。比如当我们弯曲中指的掌指关节时，超过一定范围后，若想继续弯曲，则必然会带动其相邻食指和无名指的掌指关节才能得到更大程度的弯曲。这些限制我们暂时还不能从公式上进行明确定义。

其次是人手指在未借助外力的情况下所能弯曲的角度范围，我们能够使用下面的不等式来量化他们 [?]：

$$\begin{aligned}
 \theta_{MCP_FE}^f &\in (-10^\circ, 90^\circ) \\
 \theta_{PIP_FE}^f &\in (-0^\circ, 110^\circ) \\
 \theta_{DIP_FE}^f &\in (-0^\circ, 60^\circ) \\
 \theta_{MCP_AA}^f &\in (-15^\circ, 15^\circ) \\
 \theta_{MCP_FE}^t &\in (-0^\circ, 90^\circ) \\
 \theta_{PIP_FE}^t &\in (-10^\circ, 80^\circ) \\
 \theta_{MCP_AA}^t &\in (-30^\circ, 30^\circ)
 \end{aligned} \tag{2.2}$$

其中 FE 代表屈和伸 (flexion/extension), 而 AA 代表内收和外展 (abduction/adduction), 下标 t 代表大拇指, 而 f 代表除大拇指外的其他手指。

另外对于中指来说通常具有很小的外展动作, 因此我们普遍认为:

$$\theta_{MCP_AA} = 0^\circ \tag{2.3}$$

手指之间的运动还具有很多约束性, 这里我们并不一一讨论, 遵循这些约束将使我们生成的虚拟人手更加逼真。

2.3 本章小结

本章主要介绍了人手的骨骼结构以及手指与手指之间的约束性, 通过对这些知识的介绍可以避免虚拟人手在与虚拟环境进行实时交互时产生非自然手势, 从而为我们将来模拟出逼真的虚拟人手打下坚实的基础。

第三章 虚拟人手的自动纹理贴图

物体的纹理通常是指物体的表面特征。纹理是物体最重要的特征之一，通常我们区分同一类物体是根据物体表面纹理来进行的。对物体进行纹理贴图则是指采用图像或者函数来改变物体的表面特征。在计算机图形学中当我们难以用可视化数据对物体表面进行精确描述时，就可以采用纹理贴图这种方式来方便的表达物体，并可在物体建模、运行速度和存储空间上节省大量的资源。本章详细介绍了对人手模型进行纹理贴图整个步骤。

3.1 3D 人手模型的调整

为了对 3D 人手模型进行纹理贴图，首先应该通过拍摄得到人手的正面和背面图。纹理图的质量将会严重影响最终的纹理映射，因此我们应该选择在光线较好的环境下进行拍摄。另外为了防止拍摄时的背景干扰最终的映射过程，因此我们以白纸作为背景。最终我们得到手的正面图和背面图如图 3.1 所示。

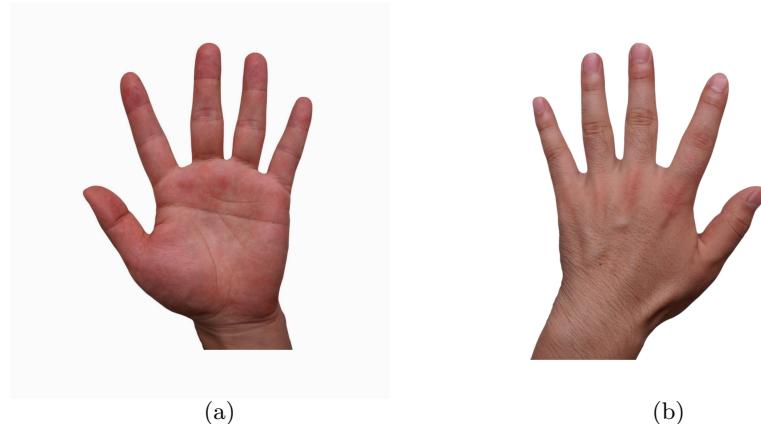


图 3.1: 拍摄得到的纹理图:(a) 正面图;(b) 背面图.

目前有大量的软件可以用来制作 3D 模型，比如 3DS max、Maya 以及 Autocad 等，这些软件已经非常的普及并且易于上手。而其中的 3DS max 更是以其强大的建模和渲染能力被广泛应用于三维动画、影视制作以及建筑设计中，得到大量研究者的青睐。但是由于其制作出来的很多模型文件格式的内部结构

并未得到公布，因此很不利于研究者进行二次开发和利用。不过这些未公布的文件格式可以通过格式转换功能来生成一些通用的文件格式，比如 3ds 和 obj 格式等。

这里我们采用 3DS max 软件直接对从 Internet 上已经共享的一般人手模型的 max 文件进行调整，该软件可以从让我们从多个不同的视角查看物体对象，并对其进行实时渲染和显示。如图 3.2 所示：

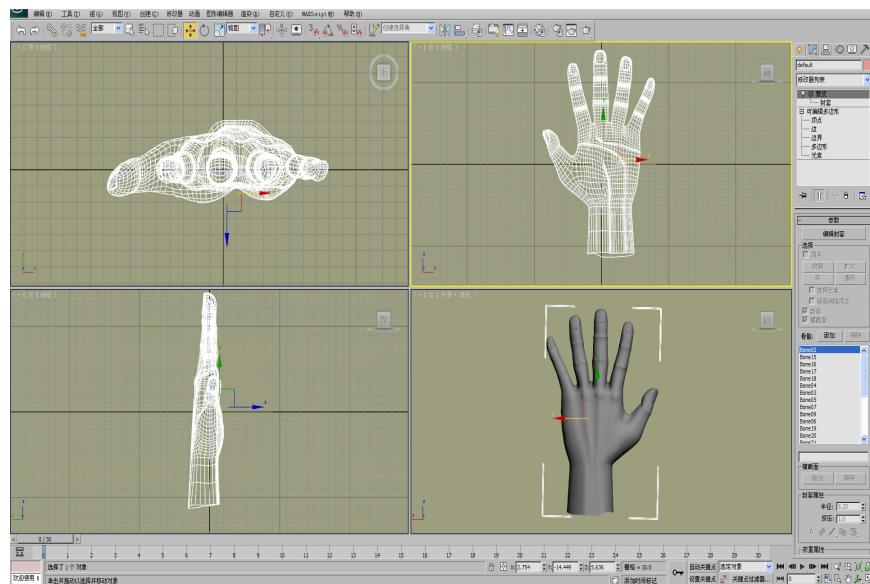


图 3.2: 3DS MAX 制作人手模型

在调整时需要以拍摄得到的背面纹理图作为模板。将调整后得到的个性化人手模型导出为 3ds 模型，便于进行二次开发。图 3.3(a) 和 3.3(b) 分别为调整前的人手模型以及调整后的人手模型。

3.2 正面图的调整

在上一小节中因为我们采用背面纹理图作为模板对一般人手模型进行调整，这样对模型背面采用人手背面图进行纹理贴图时并不会有什么问题。但是如果直接将拍摄得到人手正面图对模型正面进行纹理贴图时将会带来很大困扰。这是因为在拍摄过程中因为相机的抖动以及手指的运动，使得人手正面图并不能和人手背面图的镜像完全重合。图 3.4 是人手正面图和人手背面图的镜像放在一起后的结果。

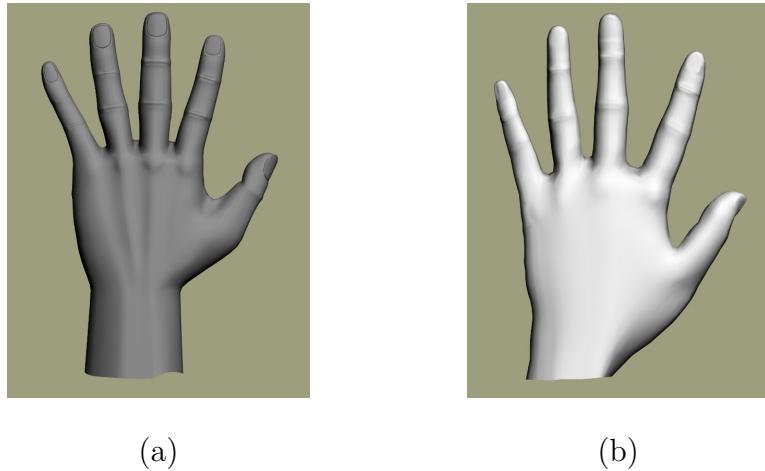


图 3.3: 人手模型: (a) 一般模型; (b) 个性化模型.

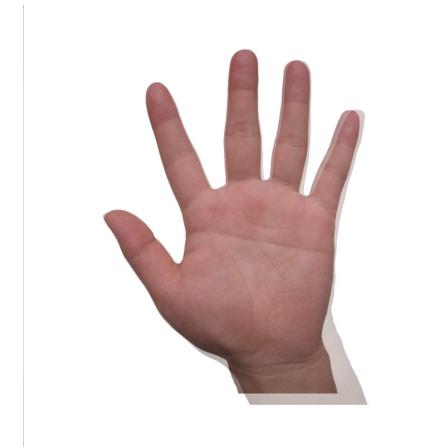


图 3.4: 人手正面图和人手背面图镜像后的重合

从图 3.4 中我们可以清晰的看出在小拇指指出以及手腕处两张图明显不能重合，因此为了便于后面的纹理映射我们需要对人手正面图进行调整使其可以和人手背面图的镜像重合。

调整的步骤如下：

- 1) 对人手背面图镜像处理。之后分别对处理后的背面图像以及人手正面图进行二值化处理，以便于下面对人手边缘进行检测。
- 2) 逐行扫描两幅图像，记录检测到的每根手指或者手掌的边缘坐标。这里以小拇指为例，如图 3.5 所示，

a 和 b 分别代表经过镜像处理后的背面人手小拇指的左右边缘，而 c 和 d 则分别代表正面人手小拇指的左右边缘， x 和 x' 代表边缘连接线上的任意位

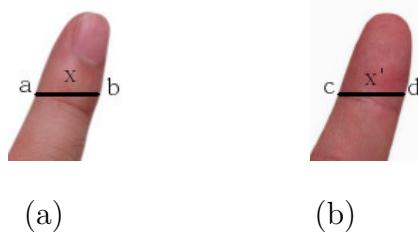


图 3.5: 小拇指的边缘检测: (a) 镜像后的小拇指背面; (b) 小拇指正面.

置。

3) 线性平移缩放线 cd 到线 ab 处。变换的公式为:

$$x = a + (x' - c) * \frac{b - a}{d - c} \quad (3.1)$$

将人手正面图 x' 位置处的像素值替换成由公式 3.1 得到的人手正面图 x 位置处的像素值。这里需要注意我们只是以小拇指作为例子，因此还需要将以上操作运用到检测到的每一个手指或者手掌的边缘。

图 3.6 分别展示了原始的手掌正面图、经过处理后的人手正面图以及将镜像后的背面手掌图与处理后的人手正面图叠加后的效果图。

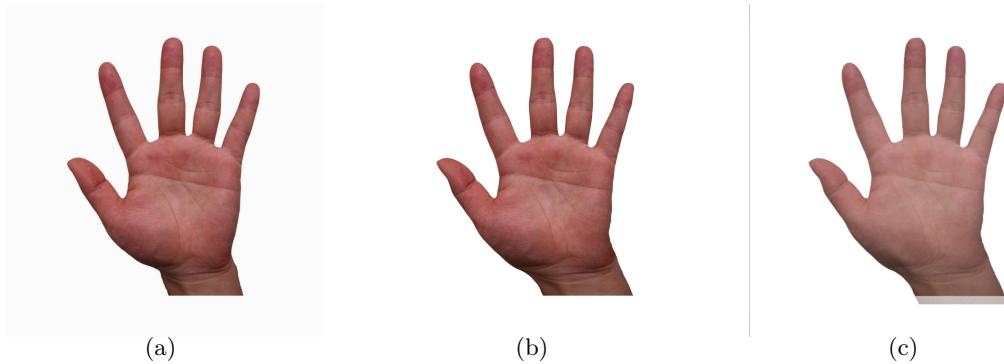


图 3.6: 人手正面图: (a) 原始图; (b) 处理图; (c) 叠加图.

从图 3.6(c) 我们可以看出经过处理后的人手正面图可以和镜像后的人手背面图完美重合。

3.3 色彩传递

直接拍摄得到的人手正反面图像因为光照和肤色等原因可能在色彩上有很大差异。如果直接进行纹理融合会使结果显得很不真实，因此我们需要对人手

正面图和背面图进行色彩传递。但在介绍算法前我们需要介绍两种将会用到的颜色模型。

3.3.1 RGB 颜色模型

RGB 颜色模型是人们使用最多的一种颜色模型。这种模型被广泛的应用于彩色监视器和显示器系统之中。它是采用红 (R)、绿 (G)、蓝 (B) 为三基色并通过一定程度的叠加来产生人类所能看到的各种各样的颜色。任意颜色的方程可以表示如下：

$$F = \alpha(R) + \beta(G) + \gamma(B), \quad 0 \leq \alpha, \beta, \gamma \leq 1 \quad (3.2)$$

其中 α 、 β 、 γ 被称为三色系数，用来对三原色进行混合。

我们可以用一个三维立方体来直观的对 RGB 颜色空间进行描述 [?]，如图 3.6 所示：

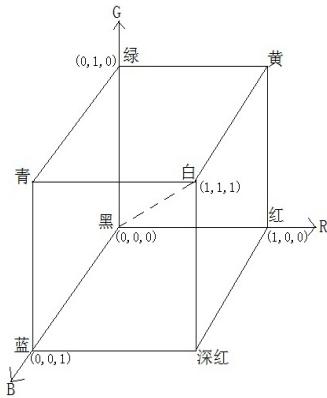


图 3.7: RGB 颜色模型

图中的原点表示为黑色 $(0,0,0)$ ，沿着正方形的主对角线依次产生灰度级变化，最终变为白色 $(1,1,1)$ 。另外别的各点也分别代表不同的颜色，如 $(0,1,1)$ 代表青色， $(1,0,1)$ 代表深红而 $(1,1,0)$ 代表黄色。对 R、G、B 进行不同比例的混合就可以产生任意需要的颜色。

3.3.2 $l\alpha\beta$ 颜色模型

Ruderman 等人 [?] 于 1998 年提出了这种新的模型，目前还未得到普及。这种颜色模型通过对自然界图片的颜色分布进行统计，使其能够更加符合人类的视觉感知系统。 $l\alpha\beta$ 颜色空间的中 l 通道表示亮度值坐标值、 α 表示黄蓝对立色坐标值而 β 则表示红绿对立色坐标值 [? ?]。这种颜色空间因为可以将颜色信息和亮度进行最大限度的分离，因此它极大的降低了各颜色通道之间的相关性，使其具有一定的相互独立性。

$l\alpha\beta$ 颜色模型可以使我们在不同的颜色通道中进行不同的运算从而能够避免通道的交叉问题。因此这种模型被广泛用于彩色图像处理和色彩传递等问题中。

3.3.3 基于 $l\alpha\beta$ 颜色空间的色彩传递

我们采用 Reinhard 等人 [?] 的方法通过将图像由 RGB 颜色空间变换到 $l\alpha\beta$ 颜色空间，并在 $l\alpha\beta$ 空间中采用目标图像的均值和方差去代替源图像的均值和方差 [?]，其具体步骤如下：

1) 首先将图像从 RGB 颜色空间转换到与设备无关的 XYZ 颜色空间中去，转换的公式如式 3.3 所示：

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.5141 & 0.3239 & 0.1604 \\ 0.2651 & 0.6702 & 0.0641 \\ 0.0241 & 0.1288 & 0.8444 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (3.3)$$

2) 将 XYZ 颜色空间转换到 LMS 锥细胞空间中去，转换的公式如式 3.4 所示：

$$\begin{bmatrix} L \\ M \\ S \end{bmatrix} = \begin{bmatrix} 0.3811 & 0.6890 & -0.0787 \\ -0.2298 & 1.1834 & 0.0464 \\ 0.0000 & 0.0000 & 1.0000 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad (3.4)$$

3) 将 3.3 式和 3.4 式合并得:

$$\begin{bmatrix} L \\ M \\ S \end{bmatrix} = \begin{bmatrix} 0.3811 & 0.5783 & 0.0402 \\ 0.1967 & 0.7244 & 0.0782 \\ 0.0241 & 0.1288 & 0.8444 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (3.5)$$

4) 由于图像在 LMS 颜色空间中的数据分布很不均匀, 因此我们需要对 L, M, S 分别取对数以改善数据相对于坐标轴的对称性, 其变换的方式如下:

首先使

$$L = \log L$$

$$M = \log M \quad (3.6)$$

$$S = \log S$$

之后旋转 LMS 颜色空间的坐标轴使其不相关。旋转的方式为:

$$\begin{bmatrix} l \\ \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{3}} & 0 & 0 \\ 0 & \frac{1}{\sqrt{6}} & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & -2 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} L \\ M \\ S \end{bmatrix} \quad (3.7)$$

5) 计算人手背面图像在 $l\alpha\beta$ 颜色空间的均值 $\bar{l}_s, \bar{\alpha}_s, \bar{\beta}_s$, 以及在该颜色空间中的方差 $\sigma_s^l, \sigma_s^\alpha, \sigma_s^\beta$.

6) 计算人手正面图像在 $l\alpha\beta$ 颜色空间的均值 $\bar{l}_t, \bar{\alpha}_t, \bar{\beta}_t$, 以及在该颜色空间中的方差 $\sigma_t^l, \sigma_t^\alpha, \sigma_t^\beta$.

7) 以人手背面图为基准, 计算经过变换后的人手正面图的 l', α', β' , 变换的方式为:

$$\begin{bmatrix} l' \\ \alpha' \\ \beta' \end{bmatrix} = \begin{bmatrix} \frac{\sigma_t^l}{\sigma_s^l} * (l_s - \bar{l}_s) + \bar{l}_t \\ \frac{\sigma_t^\alpha}{\sigma_s^\alpha} * (\alpha_s - \bar{\alpha}_s) + \bar{\alpha}_t \\ \frac{\sigma_t^\beta}{\sigma_s^\beta} * (\beta_s - \bar{\beta}_s) + \bar{\beta}_t \end{bmatrix} \quad (3.8)$$

8) 最后分别采用式 3.7 和式 3.5 的反变换将人手正面图由 $l\alpha\beta$ 颜色空间变換回 RGB 颜色空间中去。

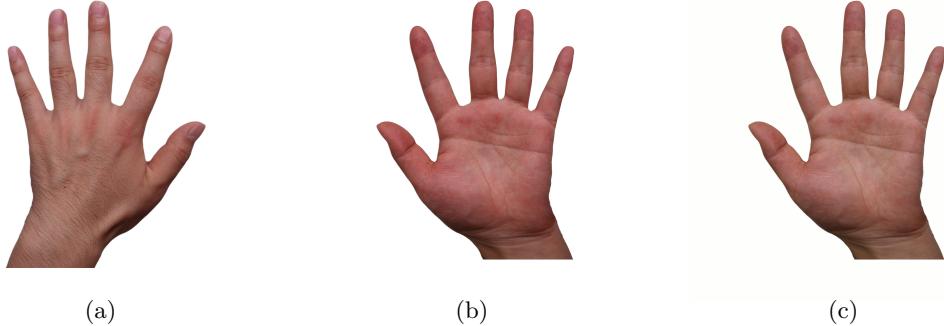


图 3.8: 色彩传递: (a) 原始的人手背面图; (b) 原始的人手正面图; (c) 经过色彩传递处理后的人手正面图.

最终的效果图如图 3.8 所示:

将图 3.8(b) 和 3.8(c) 进行对比我们发现色彩传递能够很好的使得处理后的人手正面图和人手背面图颜色非常接近。

3.4 正反面顶点分离

3D 模型通常是由许多点组成, 点与点之间进行连线组成多边形, 并经过纹理贴图等渲染技术生成最终的模型。因此当我们采用人手正反面图像对 3D 手模型进行纹理贴图时需要将模型的正反面顶点分别提取出来。顶点的提取涉及到计算机图形学中经典的隐藏面消隐技术, 这种技术的产生通常是因为我们在三维场景中并不能看到不透明三维物体的全部表面, 物体之间因为相互遮挡而使得部分不可见。如果我们还是将全部的多边形进行绘制将会浪费很多计算机资源, 并使得显示的三维物体不真实。因此在确定视点之后, 我们需要将没有贡献的场景部分删除, 而将可见的场景发送至绘制管线。图 3.9 展示了消隐算法的功能。

这里我们先介绍几种比较常用的隐藏面消隐算法, 之后详细介绍我们所使用的正反面顶点分离方法。

3.4.1 深度排序算法

这种算法又被称为画家算法。通常画家在作画时总是先在图纸上涂上背景色, 然后画上较远的景物, 最后画上较近的景物。通常近的场景会遮挡较远的

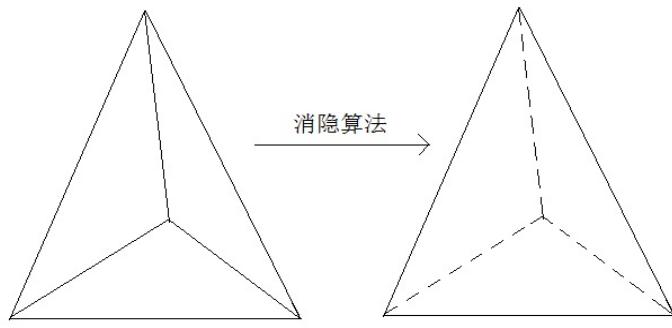


图 3.9: 消隐算法实例

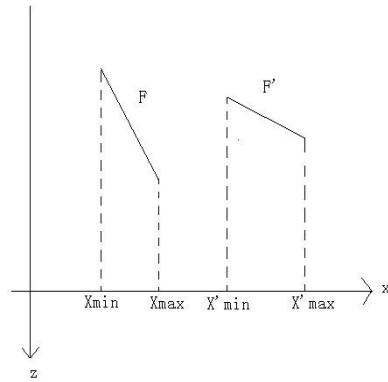
场景，从而将其覆盖。

采用同样的方法，我们首先对所有多边形按照深度进行排序，并定义深度大的场景优先级低，而深度小的场景优先级高。之后从优先级低的多边形开始逐个进行扫描转换，直到最高优先级的多边形为止。

该算法看似简单，但是需要对不同情况下的多边形以深度为条件进行排序。如果在深度上没有重叠才进行扫描转换。如果有重叠，则需要通过一定的比较来判定是否依然可以进行扫描交换，还是需要对表面进行交换来继续判定。下面以 F 作为需要绘制的面，而以 F' 作为与 F 在深度上重叠而需要与之进行比较的面，其中视点位于 z 轴正方向无限远处。比较准则如下所示 [?]：

(1) 两表面在 xy 坐标平面上投影的包围矩形无重叠。

如图 3.10 所示：

图 3.10: 两表面在深度上有重叠但在 x 方向无重叠

从图 3.10 我们可以看出 F 与 F' 在深度上有重叠，但是他们在 x 方向上无重叠，因此认为两表面无重叠。如果他们在 x 方向上有重叠则需要判断他们在 y 方向上是否有重叠。总而言之，只要他们在任一方向上无重叠就认为两表面无重叠。

(2) 相对于观察者而言，面 F 完全位于面 F' 之后。

如图 3.11 所示：

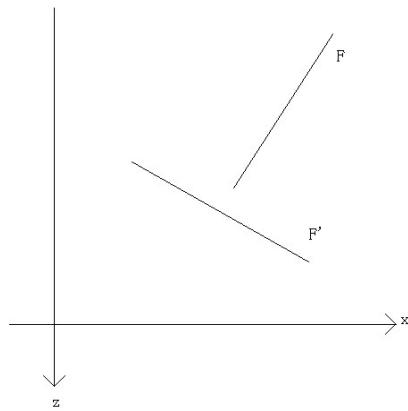


图 3.11: F 完全位于 F' 之后

(3) 相对于观察者而言，面 F' 完全位于面 F 之前。

如图 3.12 所示：

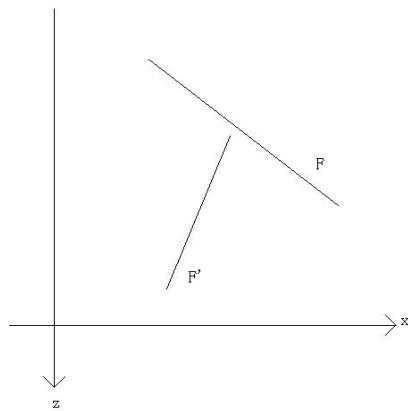


图 3.12: F' 完全位于 F 之前

上述三个条件只要任一条件为真则都不需要继续往下进行判断，将下一重叠表面代替 F' 继续从条件一开始判定即可。若是三个条件均不成立，则需要

判定条件 4。

(4) 两表面在观察平面上投影无重叠。

如图 3.13 所示：

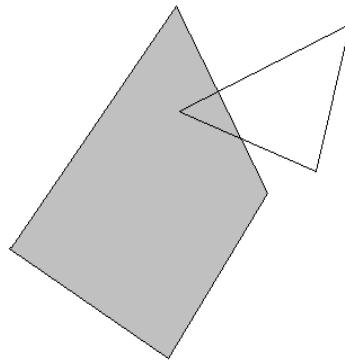


图 3.13: 两表面相交

上图中两个表面相交，因此不满足条件 4。

如果对于与 F 表面在深度上有重叠的所有表面进行以上测试后，只要有一个条件成立则可对 F 进行扫描转换。如果均不成立则需要将 F 与 F' 进行交换并再次执行上述的四个判定条件，直到找到满足条件的面为止。

深度排序算法因为需要对每个点都进行渲染，而忽略了场景中的遮挡因素。因此对于复杂的场景而言，深度排序算法可能会浪费很多的计算机资源。另外若是存在两个或者多个表面相互遮挡的情况时，利用上述判定准则还有可能出现死循环的现象。为了避免此算法的缺陷，科学家们在此基础上研究出了深度缓冲算法，我们将会在下一小节中详细介绍。

3.4.2 深度缓存算法

这种算法是目前最常用的一种消隐算法，其最早由 Edwin Catmull 提出。因其简单而易于实现，目前已成为 OpenGL 所使用的方法，并且被硬件化而存在于许多先进的图形工作站和计算机图形系统中，从而可以快速生成和实时显示需要的图形。

其具体的算法步骤如下：

1) 首先定义两个数组分别为深度缓存数组 ZB 与颜色属性数组 CB，并将

两个数组分别初始化使得 $ZB(i, j) = 1.0$, $CB(i, j) = backgroundColor$, 这里的 (i, j) 代表屏幕上的像素位置。

- 2) 处理场景中多边形 A , 计算其在点 (i, j) 处的深度值 z .
- 3) 若 $z < ZB(i, j)$, 则更新该位置的颜色使得 $ZB(i, j) = z$, $CB(i, j) = surfaceColor$.
- 4) 对场景中的所有多边形均执行 (2)、(3) 两步, 最终深度缓存数组 ZB 中存放的即是可见面的深度值, 而颜色属性数组 CB 存放的就是可见面。

如图 3.14 所示, 最终我们存放的即是 $F1$ 处的深度值以及颜色值 [?]。

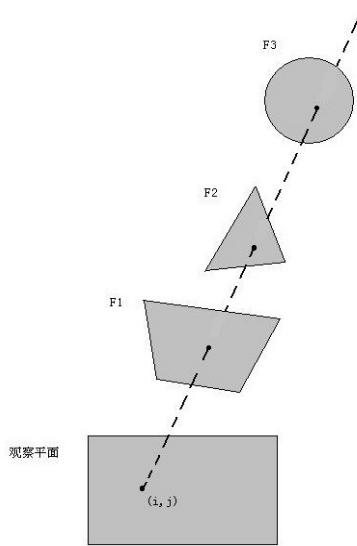


图 3.14: 通过观察面得到 $F1$ 的深度最小

深度缓存算法易于实现, 算法复杂度为 ON , 其中 N 为多边形表面采样点的数目。但是却是一种典型的以空间换取时间的算法。这种算法占用了大量的存储空间, 以目前常见的 $1280*1024$ 分辨率的系统而言, 需要 $2*1280*1024$ 个存储单元, 如果我们以 $float$ 型进行数据的存储, 则大约需要 1000 万个字节左右。为了解决此问题我们可以采用每次只存储一行的方式, 每计算出一行像素就进行输出, 然后再处理下一行 [?], 这种方式虽然能极大的减少存储空间, 但是因为它忽略垂直方向上点的相关性, 因此得到的结果并不十分理想。

另外在图 3.14 中, 当我们判断出 $F2$ 的深度值小于 $F3$ 的深度后, 我们以 $F2$ 的深度代替 $F3$ 的深度值, 并更新 (i, j) 处的像素值。但之后我们发现 $F1$ 的

深度值比 F1 还要小，这样对于 F2 我们就做了一些无用的计算。因此目前有些图形软件可以让我们通过选择测试深度范围的方式来排除一些不需要的计算。

OpenGL 支持深度缓存。在初始化显示模式前首先要指定将会使用深度缓存：

```
glutInitDisplayMode(GLUT_DEPTH|GLUT_RGB);
```

在对图像进行渲染前还需要启用深度测试：

```
glEnable(GL_DEPTH_TEST);
```

启用深度缓存后需要对深度缓存进行初始化：

```
glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
```

而关闭深度缓存为：

```
glDisable(GL_DEPTH_TEST);
```

3.4.3 光线跟踪算法

光线跟踪 (Raytracing) 算法可以看作深度缓存算法的一个变体，其最早由 Arthur Appel 在 1968 年提出。其基本思想是由观察平面上的所有像素都射出一条光线，计算出每个光线与场景中对象的所有交点，其中离像素点最近的非透明表面即为我们需要进行渲染的可见面。算法原理如图 3.15 所示：

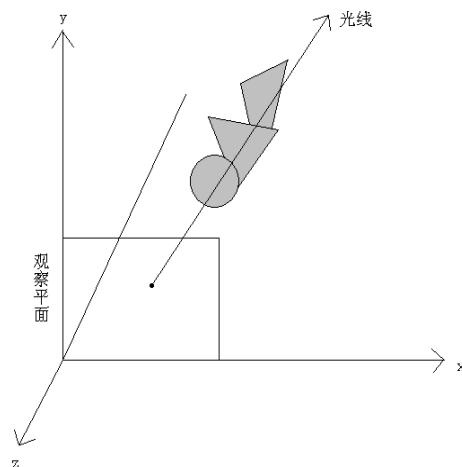


图 3.15: 光线跟踪算法

光线跟踪算法的基本步骤可简单描述如下 [?]:

- 1) 沿着视点和观察平面上的所有像素点出发作一入射线。

- 2) 将每个入射线与场景中的所有对象求交。
- 3) 对每根入射线，若求得交点，则将交点按照深度大小进行排序，取出离观察平面最近的表面上的颜色。若无交点，则取出背景色。
- 4) 将取出的颜色取代像素的颜色即得到最终渲染的结果。

光线跟踪算法其作用并不仅仅是计算得到可见面。其更大的作用是沿着到达视点的光线的反方向进行跟踪，通过计算光线在物体上的反射和折射，从而计算出物体上的总光源。这为渲染三维场景的反射和折射效果提供了一种简单有效的方法。这里我们不再详细讨论。

计算物体可见面除了以上三种方法外还有区域排序算法、扫描线算法和 BSP 算法等。虽然这些算法的基本思路各不相同，但它们大多都是采用排序和连贯性的方法来提高算法的效率。我们不再一一介绍。

3.4.4 顶点分离

正反面顶点的提取需要利用隐藏面消隐技术。当人手正面朝向观察者时将背面顶点组成的多边形删除，而当人手背面朝向观察者时将正面顶点组成的多边形删除。这样即可分别提取正反面的顶点，从而便于以后纹理的映射。不过考虑到人手的 3D 模型可看做是一个凸多边形，并且对 3D 模型进行渲染时是以三角形面片为单元进行 [?]。针对这一特性我们能设计一种更简单的方案来提取前后点。其具体步骤如下：

- 1) 使人手背/正面朝向观察者，将所有顶点均标记为正。
- 2) 对任一三角形面片，判断剩余顶点是否位于三角形之内。如果在则进一步判断它的深度是否小于组成三角面片中任意顶点的深度。如果是则认为此顶点是属于非背/正面顶点，将其标记为负。
- 3) 对所有三角形面片均进行 2) 中的处理，最终得到的标记为正的顶点就是人手背/正面的顶点。

这里的键是第二步，我们需要判断一个顶点是否在一个三角形内，判断的方法也很简单，如图 3.16 所示：

计算出三角形三条边所在的直线方程，如计算出 AB 的直线方程 $ax + by + c = 0$ 后，将 d 的坐标 (x_d, y_d) 和 C 的坐标 (x_C, y_C) 分别带入 AB 的方程，如

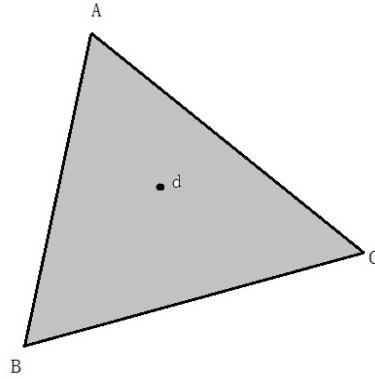


图 3.16: 判断顶点是否在三角形之内

果有 $(ax_d + by_d + c) * (ax_C + by_C + c) > 0$ 说明两者同号, 也即位于 AB 直线的同一侧。同样对 AC 、 BC 进行判断, 如果三条直线均与 d 点在同一侧则认为其在三角形内部。

因为需要对每个三角形都遍历所有顶点, 其算法复杂度为 $O(M * N)$, 其中 M 为三角形个数而 N 为顶点个数。因此我们可以简化判断的方法, 采用包围体的设计思想 [?], 对每个三角形都用一个四方形的包围框围着, 如图 3.17 所示:

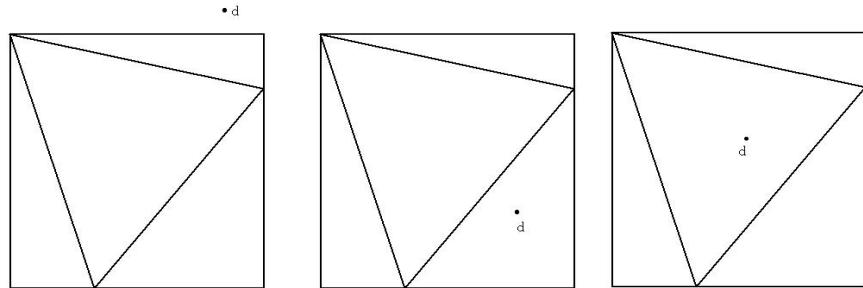


图 3.17: 包围盒

3.17(a) 中顶点未落入包围框内, 因此不对其进行处理。而在 3.17(b) 与 3.17(c) 中顶点落入包围框内, 因此还需要判断其是否在三角形内部。而判断一个顶点是否在包围框内只需比较将其坐标与包围框的最小与最大坐标比较即可。

另外还需要注意的是我们在步骤二中标记为负的点并不能直接作为人手正/背面上的顶点。比如对手背处理后那些被标记为负的点只能算是人手正面

点的一部分，因为在人手正面和人手背面的接缝处这些点既属于正面点，又属于背面点。因此需要把这些点也包含进去。我们采用的方式是忽略标记为负的点，将手在三维空间中旋转 180° 然后做和上面相同的处理即可。

最终提取出的正反面顶点如图 3.18 所示：

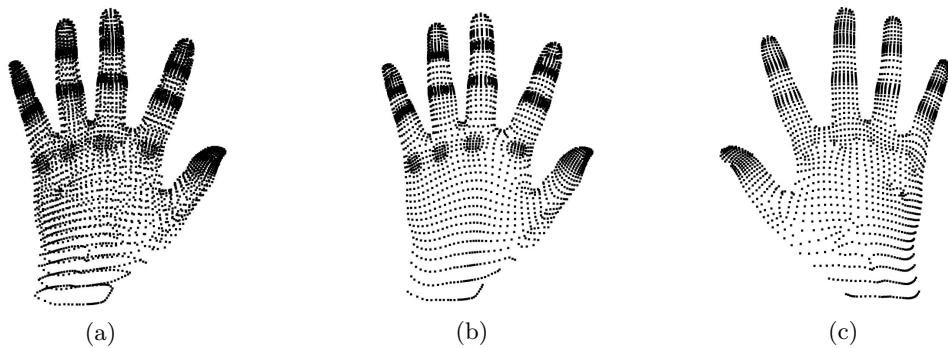


图 3.18: 顶点的渲染: (a) 所有顶点; (b) 背面顶点; (c) 正面顶点.

3.5 纹理映射

提取出每个点以后我们需要使用映射函数将其坐标值映射到纹理空间去，并通过得到的纹理空间值从纹理中得到相应的颜色值，这个过程就被称为纹理映射。而纹理映射的过程实际上与将三维空间中的物体投影到观察平面的过程是极其相似的。因此我们首先介绍物体的投影过程，然后将其直接应用到我们的纹理映射当中去。

通常我们在对一个三维对象进行显示时首先需要选择裁剪区域，裁剪区域的作用是限制三维对象的显示范围，只有处于裁剪区域内的对象才会进行投影并最终显示在我们的观察视口之中，而处于裁剪区域外的对象则不进行显示。其裁剪过程如图 3.19 所示：

我们的三维对象是面片 ABC ，但是经过裁剪区域裁剪后变成了 $ABde$ 。因为只有在裁剪区域的三维对象才会进行投影，因此裁剪区域又被称作视景体。裁剪操作不仅仅是裁剪掉无用的对象，因为我们把需要的三维场景放入到了一个标准的立方体中，这使得我们在以后的坐标变化上更加简单而有效率。

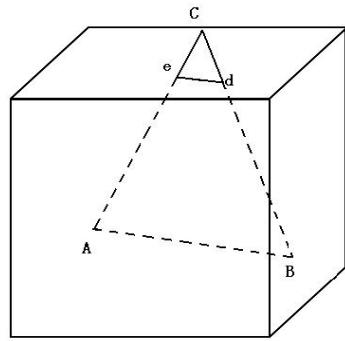


图 3.19: 裁剪对象

3.5.1 投影变换

经过裁剪操作后可以将我们感兴趣的场景投影到某个简单体或某种平面上去。投影变换实际上是一种坐标变换的过程。本文选择的投影体是一个单位立方体，这将使得我们的坐标变换操作更加简单和统一。甚至有一些文献把裁剪操作放到投影变换以后进行，因为在单位立方体中进行裁剪操作更加容易。

目前常用的投影的方式主要分为两种，分别是正交投影和透视投影。下面我们将一一介绍。

3.5.1.1 正交投影

正交投影是指投影线垂直于投影平面的一种投影方式，因此也被称为平行投影。这种投影方式因为可以精确的描绘出物体的长度和角度，因此被广泛应用于工程和建筑绘图之中。

正交投影的裁剪体通常用一个长方体包围盒来表示，而包围盒的上、下、左、右、前、后则分别用 t, b, l, r, f, n 表示。我们需要将包围盒变换成规范化观察体 (normalized view volume)，这就是我们上面所提到的单位立方体，其每一维的坐标都是从 -1 到 1。使用规范化观察体可以让我们更加有效的进行裁剪操作，尤其当我们采用硬件来实现裁剪操作时这种方式的优势更加明显，目前大部分软件包都是使用单位立方体作为规范化观察体。其变换过程如图 3.20 所示 [?]:

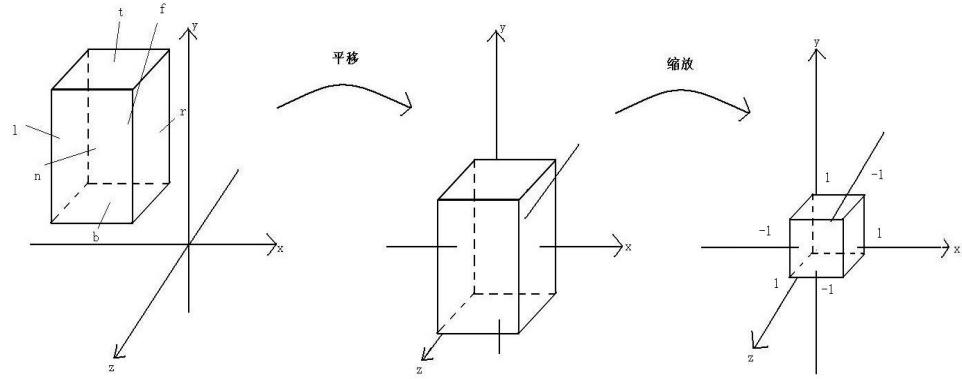


图 3.20: 正交投影转换过程

该正交变换可用下式来表示:

$$\begin{aligned}
 T &= \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & 0 \\ 0 & \frac{2}{t-b} & 0 & 0 \\ 0 & 0 & -\frac{2}{f-n} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -\frac{r+l}{2} \\ 0 & 1 & 0 & -\frac{t+b}{2} \\ 0 & 0 & 1 & -\frac{f+n}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
 &= \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & -\frac{2}{f-n} & -\frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.9)
 \end{aligned}$$

该矩阵是由平移矩阵和缩放矩阵相乘得到的。通过此矩阵即可完成视景体从世界坐标系到规范化观察体所在坐标系的转换。这时我们再进行裁剪操作，将需要绘制的几何体进行裁剪，并将单位立方体通过背景面消除等操作映射到最终的屏幕上，从而完成三维物体的绘制。

目前许多图形化软件已经包含了能够处理正交投影的函数，如在 OpenGL 中正交矩阵的生成函数是 *glOrtho* 以及 DirectX 中的 D3DXMatrix OrthoOffCenterLH。

3.5.1.2 透视投影

尽管正交投影较易生成并且可以保持物体原有的比例，但是却丧失了物体的真实感特征。比如两条无限长的铁轨，如果采用正交投影，则我们所看到的依然是两条平行的直线。但是实际上在无限远处我们所能看到的两条铁轨应该相交于一点，这一点就被称为灭点。而采用透视投影则可以达到这样的效果，因为它更加符合人类感知世界的过程：相同大小的物体距离观察平面越远则投影后就越小。

透视投影的裁剪体通常用一个棱台观察体来表示，而观察体的上、下、左、右、前、后则依然采用 t, b, l, r, f, n 表示。将棱台观察体转换为规范化观察体的过程如图 3.21 所示：

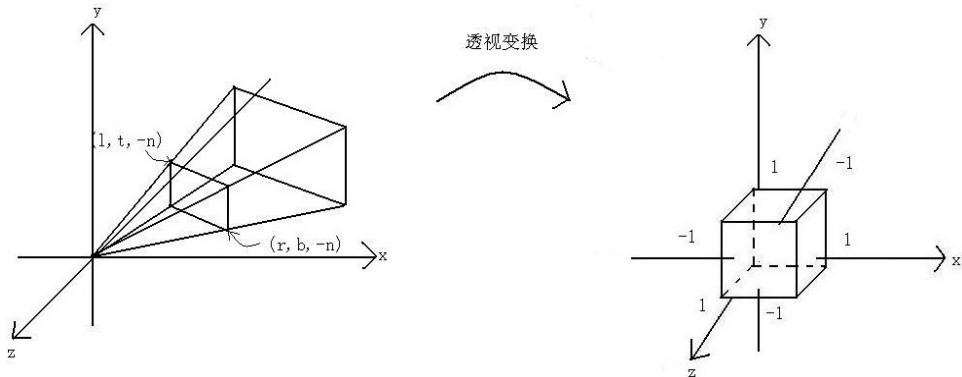


图 3.21: 透视投影转换过程

透视投影的变换通常是不可逆的，其变换可用下式来表示：

$$T = \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \quad (3.10)$$

这个矩阵就被称为透视矩阵，它也是由平移矩阵和缩放矩阵再加上透视变换得到的。通过此矩阵我们就可以将棱台观察体转换为规范化单位立方体，从而利于我们的裁剪操作。

OpenGL 中有两个函数可以实现透视投影，分别是 `glFrustum` 函数与

gluPerspective 函数，通常我们更多的使用第二个函数，因为它的参数更加直观。在使用这两个函数前需要先调用 *glMatrixMode(GL_PROJECTION)*。

3.5.2 视口变换

无论是正交投影还是透视投影都是通过平移和缩放等操作将一个平行六面体变成了一个单位立方体。比如一个长宽比为 2 的场景经过正交变换或透视变换后长宽比均变为了 1，其对 x 轴和 y 轴方向分别做了不同程度的拉伸，把任意的三维场景都变换到一个固定大小的空间中，这必然会造成物体的形变。为了解决此问题，我们可以通过视口变换再将正方体的近裁剪面（可看做观察平面）映射到长宽比为 2 的视口上，这样就可以将形变进行矫正。

其变换过程如图 3.22 所示：

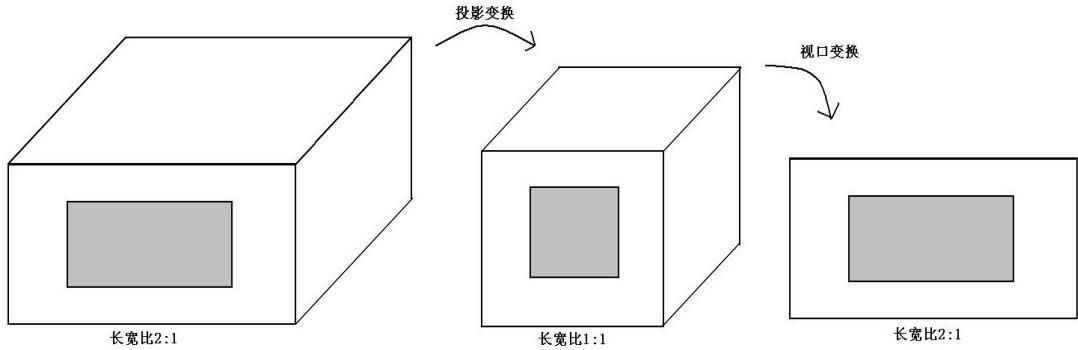


图 3.22: 视口变换

从图 3.22 我们也可看出视口变换实际上也是屏幕映射的过程，经过投影变换后的物体在单位立方体内的坐标依然是三维的，因此我们需要将其经过背影面消隐等过程提取出每个对应屏幕坐标位置处的颜色，从而将其在屏幕中显示出来。OpenGL 中处理视口变换的函数是 *glViewport*。

3.5.3 一般纹理贴图

纹理贴图实际上就是对物体表面属性进行物理建模的过程。为获得物体在纹理上的对应颜色值，首先需要通过一个或多个投影函数将物体在空间中的位置变换到纹理空间中去，然后使用纹理空间对应位置的颜色值来代替物体的表面颜色。这里我们采用一个例子来详细描述纹理映射的过程，具体过程如图

3.23 所示 [?]:

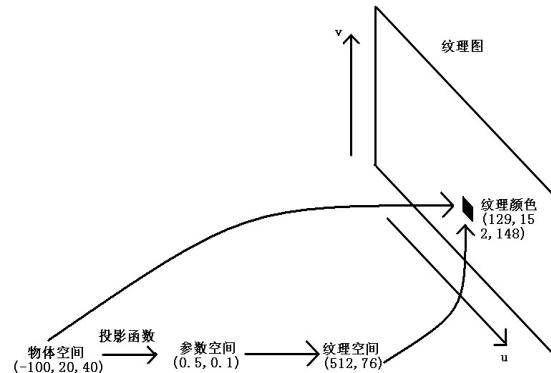


图 3.23: 纹理映射过程

假设物体在空间中的位置为 $(-100, 20, 40)$ ，这里我们使用正交投影函数将其投影到纹理空间中。通常为了便于管理，我们将纹理坐标值的范围规定为 0 到 1[?]，而不考虑纹理的重复与截取等特殊情况。因此对于二维纹理来说，纹理坐标按照逆时针方向依次是 $(0,0)$ 、 $(1,0)$ 、 $(1,1)$ 、 $(0,1)$ 。这里我们假设投影到纹理空间后的坐标为 $(0.5, 0.1)$ ，并假设纹理图像的分辨率为 1024×768 。因此可得真实的坐标值为 $(512, 76)$ ，在此位置处找到纹理图像中的颜色值为 $(129, 152, 148)$ 。这个颜色值即可用来代替物体的表面颜色。

3.5.4 对手贴纹理

对手贴纹理的过程实际上就是对 3D 人手模型进行投影的过程，只不过这里我们要对投影的位置做出较高的要求，而不仅是将其投影到任意可见的位置。我们希望投影到屏幕上的模型能够恰好与拍摄得到的人手正反面纹理图相重合。相对于人手的面而言其厚度比较小，因此这里我们采用正交投影就能达到目的。

上面我们已经讨论过纹理图的位置在 0 到 1 之间，而物体在单位立方体中的坐标为 -1 到 1，因此我们需要对 3.9 式做出调整，将其在 x 、 y 、 z 方向上均

平移 $1/2$ 个单位，并在三个方向上缩放为原来的 $1/2$ ，调整的方式见 3.11 式：

$$\begin{aligned}
 T' &= \begin{bmatrix} \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & -\frac{2}{f-n} & -\frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
 &= \begin{bmatrix} \frac{1}{r-l} & 0 & 0 & -\frac{l}{r-l} \\ 0 & \frac{1}{t-b} & 0 & -\frac{b}{t-b} \\ 0 & 0 & -\frac{1}{f-n} & -\frac{n}{f-n} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.11)
 \end{aligned}$$

通过对每个顶点右乘 3.11 式，即可得到每个顶点所对应的纹理坐标。因此我们的目的就是计算 t, b, l, r, f, n 这六个参数，考虑到前面我们已经将人手的正面和背面顶点分离开来了，因此我们可以不用计算 f 和 n 这两个参数。我们将会在下面给出计算 t, b, l, r 参数的方法。

首先给出背面人手的纹理图像以及背面人手的模型，如图 3.24 所示：

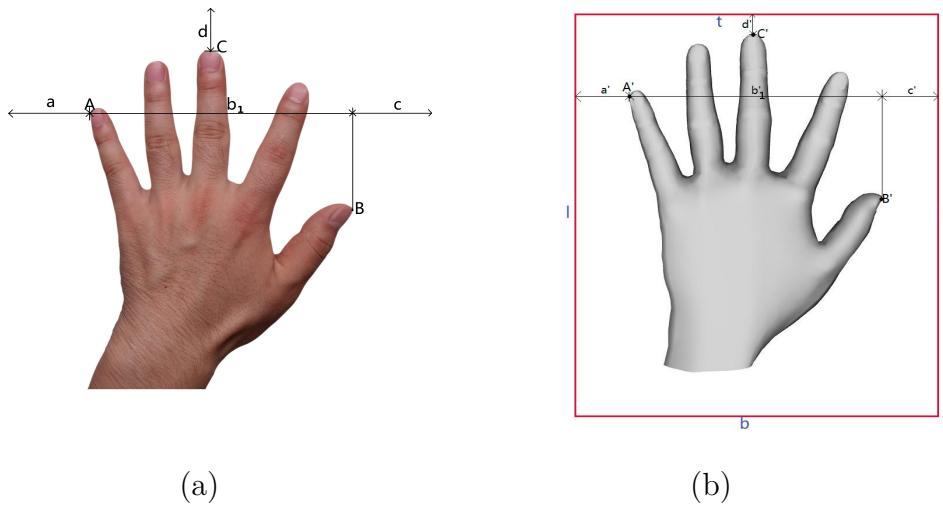


图 3.24：对两张图进行标记：(a) 标记纹理图 (b) 标记模型图。

图 3.24(a) 中 A, B, C 分别代表小拇指、大拇指和中指的最外边顶点， a, c, d 则代表 A, B, C 点与边缘的距离， b_1 代表 A 点到 B 点的横向距离。图 3.24(b)

中的标记点和图 3.24(a) 的标记点相似，只不过我们用 l, r, l, t 来代替上下左右四个边界。

从图 3.24 中我们可以看出 a, b_1, c, d, b'_1 是已知的，并且有：

$$\frac{b_1}{b'_1} = \frac{a}{a'} = \frac{c}{c'} = \frac{d}{d'} \quad (3.12)$$

从上式中我们即可求出 a', c', d' ，进而可以得到 l, r, t ：

$$\begin{aligned} l &= x_{A'} - a' \\ r &= x_{B'} + c' \\ t &= y_{C'} + d' \end{aligned} \quad (3.13)$$

其中 $x_{A'}$ 和 $x_{B'}$ 分别是 A' 和 B' 的横坐标， $y_{C'}$ 则是 C' 的纵坐标。

为了使投影后的 3D 人手模型能够和拍摄的人手纹理图像重合，我们需要满足式 3.14：

$$\frac{t - b}{r - l} = \frac{H}{W} \quad (3.14)$$

其中 W 和 H 分别代表纹理图像的宽度和高度，最终我们可以从上式中求得 b ：

$$b = t - \frac{H}{W} * (r - l) \quad (3.15)$$

图 3.25 是经过纹理贴图后的人手模型：

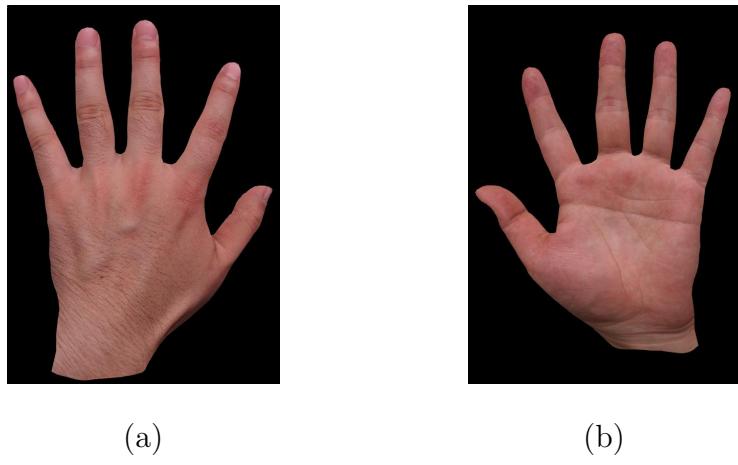


图 3.25：纹理贴图后的人手模型：(a) 模型背面；(b) 模型正面。

对比图 3.3(b) 和 3.25(a) 可以看出经过纹理贴图后的模型能够产生更加逼真的效果。

3.6 接缝移除

尽管前面我们已经对人手正面图像做了色彩传递，但是因为拍摄时在边缘处因为光照上有很大差异，因此如果直接通过两张图像对人手进行渲染，在边缘处依然会产生很严重的虚拟接缝。图 3.26 展示了手指处的接缝：



图 3.26：手指处的虚拟接缝

从图中我们可以看到中指出有明显的虚拟接缝。

为了去除接缝，我们采用一定的着色方式。在介绍具体方法前我们需要先介绍两种着色方式 [?]。

3.6.1 平面着色

平面着色的原理比较简单，通常当光源离面片很远时，面片上不同顶点的漫反射分量很小，可近似认为面片上的每个像素均具有同一种颜色。因此虽然我们在图形学渲染管线中为会一个面片的各个顶点都赋上一个颜色，但是采用平面着色我们只会选取其中一个顶点的颜色来作为整个面片的颜色。

采用这种着色方式并不会为我们带来真实的渲染效果，但是因为其快速的运算速度，因此被广泛用于速度大于细致度的场景之中。OpenGL 可以通过 `glShadeModel(GL_FLAT)` 来指定着色方式为平面着色。

图 3.27 给出了使用平面着色对球体进行渲染的效果图：

从图中我们可以清晰的看出整个球体呈现马赫带现象，但实际上我们所需要的球体应该是平滑的，而现在在每个面片都是同一种颜色，因此出现这种现象。为了解决此问题，研究者们又研究了一种平滑着色的方式，我们将在下一

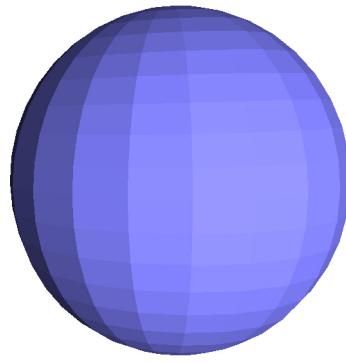


图 3.27: 平面着色渲染球体

小节中详细讨论。

3.6.2 平滑着色

这种着色方式并不是使同一面片具有一种颜色，而是通过对不同点的颜色进行线性插值，从而使整个面片显示出平滑的渐变色。平滑着色主要有两种方法，分别是 Gouraud 着色法和 Phong 着色法。第二种方法能够产生更加逼真的效果，但是它的运算复杂度和运算时间均大大高于第一种，因此在这里我们并不做讨论，我们重点讨论 Gouraud 着色法。

Gouraud 着色法又被称为高洛德着色法，它可以使面片中的任一个像素都具有不同的颜色值，这样就能够弱化相邻面片之间的颜色差异。我们以一三角形面片为例，如图 3.28 所示：

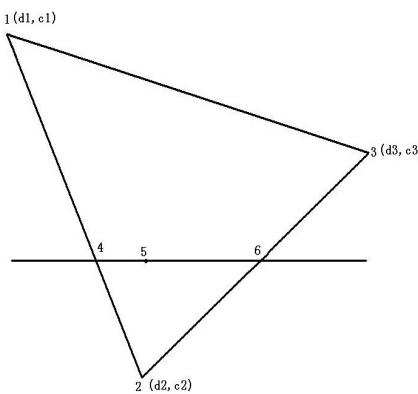


图 3.28: Gouraud 着色法

三角面片的三个顶点为 1,2,3, 其坐标分别为 $(d_1, c_1), (d_2, c_2), (d_3, c_3)$ ，这里

的 d 代表位置，而 c 代表颜色。为了求出 5 的颜色值我们需要首先求出同一扫描线上 4 和 6 的颜色值，其中 4 和 6 的颜色值计算公式如下：

$$c_4 = c_1 + (c_2 - c_1) \times \frac{d_4 - d_1}{d_2 - d_1} \quad (3.16)$$

$$c_6 = c_3 + (c_2 - c_3) \times \frac{d_6 - d_3}{d_2 - d_3} \quad (3.17)$$

从而可以得到 5 的颜色值：

$$c_5 = c_4 + (c_6 - c_4) \times \frac{d_5 - d_4}{d_6 - d_4} \quad (3.18)$$

另外也可以得出在同一扫描线上相邻像素的颜色差值为 $\frac{c_6 - c_4}{d_6 - d_4}$ ，这是一个常数，通过采用增量计算可以提高计算效率。

图 3.29 是采用平滑着色渲染的球体：

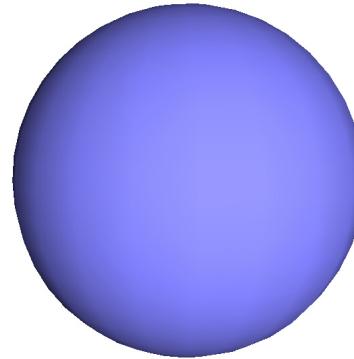


图 3.29: 平滑着色渲染球体

可以看出虽然平滑着色的处理算法比平面着色多了一些计算，但是经过 Gouraud 着色法处理后的球体更加平滑了，我们很难看出面片的边，取而代之的是面片间平滑过度的颜色。另外 OpenGL 也实现了这种着色方式，通过 `glShadeModel(GL_SMOOTH)` 即可调用 Gouraud 着色。

3.6.3 基于平滑着色去除接缝

虚拟接缝通常只存在与两张图片的拼接之处，因此我们只需针对拼接处的顶点采用平滑着色的方式即可。我们处理的方式大致思想为：

1) 对人手背面所有顶点依然采用纹理贴图的方式进行处理。而人手正面并不全部采用纹理贴图，而是提取出最外层顶点以及其最临近的次外层顶点，提取的结果如图 3.30 所示：

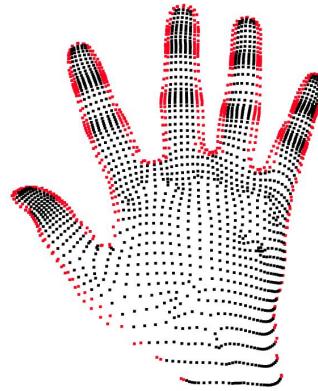


图 3.30：提取最外层和次外层顶点

2) 对最外层顶点置予背面人手接缝处的颜色，次外层顶点置予正面人手对应位置的颜色。而由这些顶点所组成的三角面片中像素的颜色则采用平滑着色的方式计算得到。

最终我们得到经平滑着色处理后的虚拟接缝如图 3.31：



图 3.31：去除虚拟接缝后的效果图

对比图 3.26 和图 3.31 可以看出，通过对人手模型的外轮廓采用平滑着色处理，能够较好的去除虚拟接缝。

3.7 本章小结

本章主要描述了为人手模型贴纹理的方法，为模型贴上纹理可以更真实的反应物体的本质特征。首先需要对拍摄得到的正面图像进行调整，使其可以和背面图像的镜像相重合。由于拍摄的原因，需要对正面图像进行色彩传递，从而使两张图像颜色接近。因为需要将正面顶点映射到正面图像而背面顶点映射到背面图像，因此需要分别提取出正反面顶点。为了使人手模型经过投影后能够和纹理图像完美重合，我们需要选择合适的裁剪框以及投影函数。直接采用两张图像进行纹理贴图会在交汇处产生明显的虚拟接缝，可以采用平滑着色的方式对其进行去除。产生逼真的人手模型之后我们需要让它像真实的人手一样能够变换出各种手势，骨骼动画可以帮助我们做到这一点。下一章我们将详细介绍通过骨骼动画实现人机交互的方法。

第四章 虚拟手的实时交互

通过纹理映射后的虚拟人手并不能直接与虚拟环境进行交互，为了使其能够像真是人手一样做出各种姿态，我们可以采用骨骼动画的方式为其添加骨骼从而实时的控制虚拟人手。

4.1 骨骼实现人机交互

当我们为 3D 人手模型贴上纹理之后，如果只是对整个人手进行旋转平移而不是像真实人手一样能够实现各种手势的话，对于我们的模型本身来说并没有太大的意义。真实人手之所以能够实现抓捏等动作主要是通过骨骼来控制的，而在计算机图形学中，为了让虚拟人手模型能够想真实人手一样，我们依然可以采取对虚拟人手赋予骨骼的方式来实现，不过这里的骨骼不再是真实的骨骼，而是虚拟骨骼。

目前在计算机图形学中为了实现复杂物体的运动主要的方式有两种，一种是帧动画，另一种是骨骼动画。帧动画是通过分解一个动作将其存放在很多帧中，然后对这些帧连续播放就可以形成一个完整的动作。对于每帧动画都需要存取所有的顶点，动作越流畅需要的帧数就越多，因此采用这种方式需要非常大的存储空间。骨骼动画则主要是通过存取骨骼的旋转和平移矩阵，并利用骨骼与骨骼之间的相对关系，从而使得绑定在骨骼上的顶点发生改变。这种方式相对帧动画方式来说虽然增加了一些计算量，但是存储空间得到极大的减少。因此常被用于游戏、动画与影视当中。图 4.1 描述了一个简易的帧动画和骨骼动画 [?]：

通过上图可以看出帧动画需要调整每一个顶点的位置，而骨骼动画则只需要调整骨骼即可，骨骼可以带动绑定其上的顶点运动到指定位置。

帧动画通常只能通过预先的定义来展示我们所需要的动作，但是我们在运行人手模型时则要使它能够与虚拟环境进行实时的交互，而不能采用预定的行为，骨骼模型可以满足我们实时的要求。综上所述，我们采用骨骼动画的方式来处理我们的人手模型。

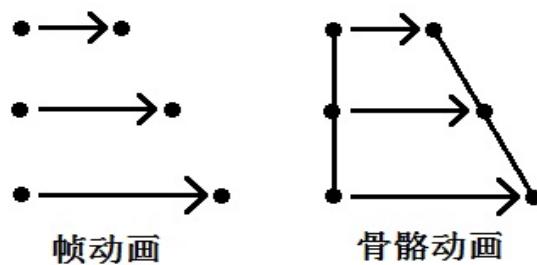


图 4.1: 帧动画和骨骼动画

3DS MAX 软件具有为 3D 模型添加骨骼的功能，通过为人手模型赋上骨骼，并采用蒙皮来将顶点绑定在骨骼上。绑定骨骼后的人手模型如图 3.33 所示：

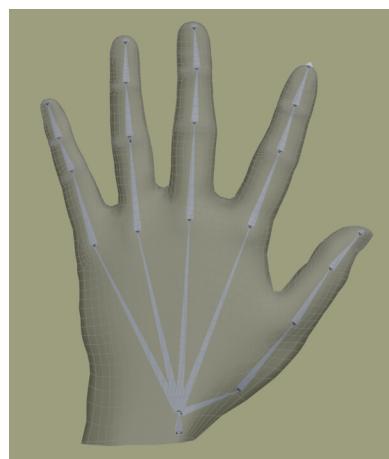


图 4.2: 骨骼绑定

从图中可以看出整个骨骼模型只有一个根骨骼，位于腕骨处。人手的运动都是以根骨骼的关节点为原点运行的，当移动根骨骼时整个人手模型都会运动。两根骨骼的连接处叫做节点，一根骨骼由两个节点控制，分别被称为父节点和子节点，其中父节点的运动控制着其下所有节点的运动。比如当食指的掌指关节运动时必然带动其下的近指关节和远指关节同时运动。对于人手模型来说一个节点只有一个父节点与其相连，因此能够简化我们的编程模块。

上面谈论了很多关于骨骼的概念，但是对于编程模型来说骨骼并不是真实存在的，可以将骨骼看成是坐标空间，而节点用来描述骨骼所在的位置，也就是坐标原点。因此使骨骼绕关节旋转就可以看成是坐标空间自身的旋转。每根骨骼都有一个相对于其父骨骼的相对矩阵，这个相对矩阵由旋转和平移操作组

成，根骨骼因为没有父骨骼因此其相对矩阵就等于绝对矩阵，也就是在世界坐标系中的位置。通过将骨骼的相对矩阵与其父骨骼的绝对矩阵相乘就可得到每根骨骼的绝对矩阵。而正是因为只需要存储骨骼的相对矩阵，因此骨骼动画非常节省内存。

另外因为 3ds 模型并不适合进行骨骼处理，因此我们选用 ms3d 模型，这种模型最初是为经典的 *Half – Life* 游戏而设计的，因其小巧灵活的特性而成为目前使用率很高的一种模型，使用这种模型能够让我们更方便的处理骨骼动画。

下面我们就介绍顶点跟随骨骼一起运动的方法。

首先将顶点绑定到骨骼之上，并定义当前骨骼的父骨骼所在空间的绝对矩阵为 $matrixGlobal_PAR$ ，当前骨骼的绝对矩阵为 $matrixGlobal_SON$ ，当前骨骼的相对矩阵为 $matrixLocal$ ，骨骼变换矩阵为 $matrixMotion$ 。 $matrixGlobal_SON$ 的计算方式为：

$$matrixGlobal_SON = matrixGlobal_PAR \times matrixLocal \quad (4.1)$$

当我们控制骨骼发生运动时，要想使得顶点跟随骨骼一起运动，则要先将顶点由世界空间变换到骨骼空间中，变换的方式是使顶点坐标右乘当前骨骼绝对矩阵的逆矩阵 $matrixGlobal_SON^{-1}$ 。而此时当前骨骼的相对矩阵为：

$$matrixLocal_NEW = matrixLocal \times matrixMotion \quad (4.2)$$

从而当前骨骼所在空间的绝对矩阵变为：

$$matrixGlobal_NEW = matrixGlobal_PAR \times matrixLocal_NEW \quad (4.3)$$

将顶点再右乘以 $matrixGlobal_NEW$ 即可将顶点由骨骼空间变换到世界空间中去，从而实现顶点跟随骨骼的移动。

这里需要注意的是对于手指这种特殊模型的运动，通常只会发生旋转而不会有平移，因此 $matrixMotion$ 只包含旋转矩阵即可。

如果我们将每个顶点只绑定到一个骨骼上面，对于非关节处并不会有影响。但是对于处于关节处的顶点在弯曲度较大的情况下会发生断裂，这会严重影响模拟的逼真度。为解决此问题，我们可以在对骨骼绑定顶点时给骨骼赋予不同

的权重，使得处于关节处顶点受到多个骨骼的拉扯作用 [?]，从而解决骨骼的断裂问题。图 4.2(a) 和 4.2(b) 分别显示了未对骨骼赋予权重与对骨骼赋予权重后的效果图。从图中我们可以清晰的看出处理前在掌指关节处有明显的断裂，

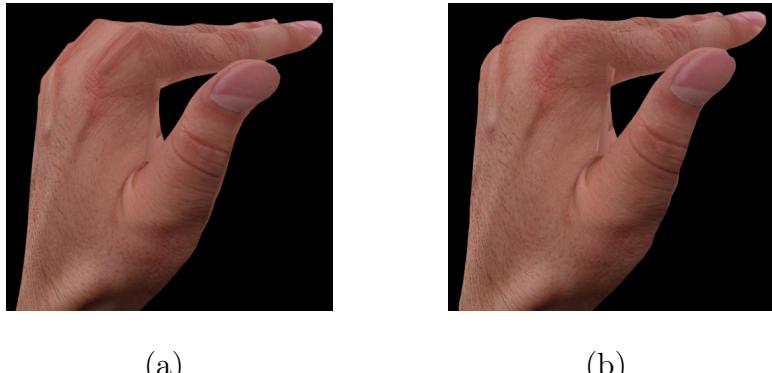


图 4.3: 为骨骼添加权重: (a) 添加前; (b) 添加后。

但是通过对骨骼添加权重信息，使得不同骨骼同时作用于顶点，非常好的解决了断裂问题。

使用骨骼带动顶点运动虽然改变了顶点位置，但是三角形面片索引和纹理索引并未发生改变，因此纹理能够跟随骨骼一起运动。但是对于影响光照信息的法线则需要重新计算，对顶点法线来说其变化的矩阵即为骨骼变化的矩阵 *matrixMotion*，只不过我们只需要考虑旋转变化而不用考虑平移变化。

通过以上这些步骤即可让我们生成一个逼真的虚拟人手，并可与虚拟环境进行实时交互。

4.2 实验结果与分析

虚拟人手的交互主要包括虚拟手在虚拟空间中的运动以及对真实人手动作的模拟 [?]。考虑到数据手套的贵重以及不方便穿戴性，我们使用键盘来控制我们的虚拟人手。考虑到 linux 平台的开源性，我们在 linux 操作系统下通过 C++ 语言和 OpenGL 图形库来实现。另外因为我们需要对拍摄得到的图像做一定的处理，因此还用了 OpenCV 图像库。用于实验的计算机配置了 AMD 六核 CPU，主频 3.5GHZ 以及 4G 内存，显卡型号为 ATI Radeon HD 7750。

我们首先拍摄了一组具有各种姿态的真实人手图像，然后控制我们的虚拟人手使其能够做出同样的姿态，其对比图如图 4.3 所示。另外，为了让我们的

手能够与虚拟环境进行交互，我们模拟了虚拟人手与布料以及虚拟人手与球的交互，其效果图如图 4.4 所示。

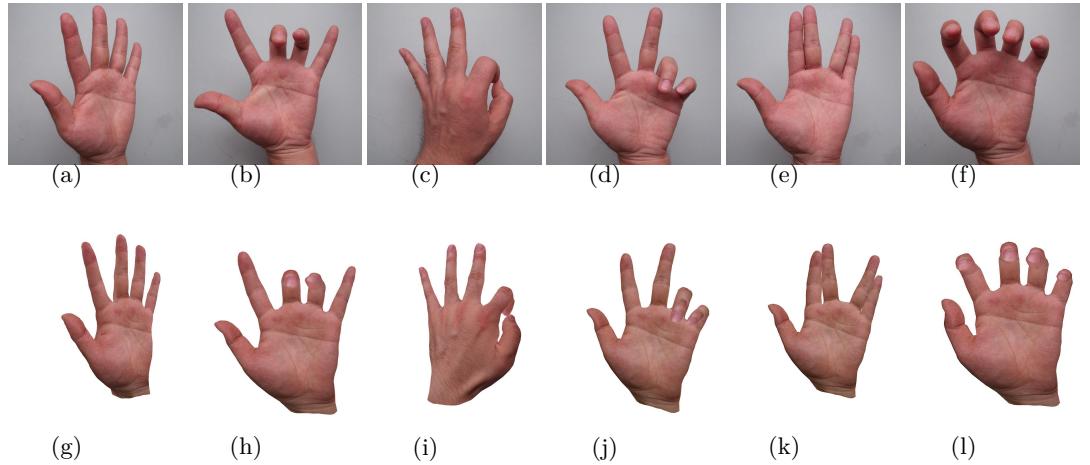


图 4.4: 真实人手与虚拟人手的对比: (a)-(f) 是真实人手的姿态; (g)-(l) 是具有同样姿态的虚拟人手.

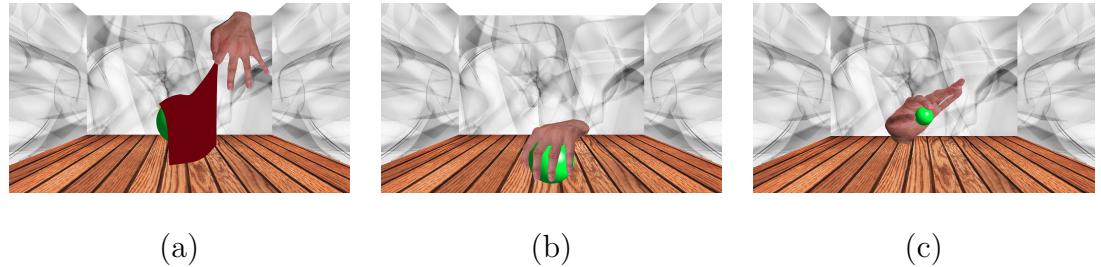


图 4.5: 虚拟人手与虚拟环境的实时交互: (a) 抓取布料; (b) 握住圆球; (c) 捏住圆球.

从图中可以看出经过纹理贴图后的虚拟人手极大的增强了模拟的逼真性，使我们能够在虚拟环境中获得更好的体验。因为虚拟人手的数据通常是轻量级的，因此能够让我们快速的与虚拟环境进行实时交互。

4.3 本章小结

为了使得经过纹理贴图后的虚拟人手能够与虚拟环境进行交互，本章介绍了一种骨骼动画的方法，通过为虚拟人手添加骨骼，并采用蒙皮等技术，可以让虚拟人手像真实人手一样做出各种姿态，能够与虚拟环境实时交互，从而为我们带来更好的体验。

第五章 工作总结与展望

5.1 论文工作总结

本文主要研究了虚拟人手的纹理贴图以及与虚拟环境的实时交互。虚拟人手在游戏、3D 动画、手语学习中都扮演着非常重要的作用。但是如果我们只是采用一般的人手模型并不能为我们带来很好的视觉体验，通过对人手进行纹理贴图则可以为我们生成逼真的个性化人手，从而使我们能够虚拟环境获得更好的体验。

我们首先描述了人手的骨骼结构以及手指与手指之间的约束性，通过对这些知识的介绍可以避免我们所生成的虚拟人手在与虚拟环境进行实时交互时产生非自然手势。纹理是物体重要的表面特征之一，为了生成个性化的虚拟人手，我们需要对人手进行纹理贴图，因此拍摄两张真实人手的纹理图像。通常我们需要先根据背面人手图像进行模型的调整，因为拍摄的原因正面人手图像并不能直接贴在模型的正面，因此需要对正面图像进行调整使其能够与背面图像的镜像重合。因为两张图像的色彩会有差异，因此需要对正面图像进行色彩传递使其可以和背面图像的色彩相似。为了使得两张图像贴到模型对应的位置，需要提取模型的正反面顶点。之后设定裁剪框，并采用正交投影从而为我们建立映射函数，根据映射函数即可找到每个顶点所对应的纹理坐标。为人手贴上纹理后会在两张纹理的连接处产生明显的虚拟接缝，利用平滑着色中的 Gouraud 着色法可以去除接缝。为了使得经过纹理贴图后的个性化人手能够进行抓捏等操作，我们可以采用骨骼动画的方法为 3D 人手模型赋上骨骼，利用蒙皮等技术使得虚拟人手能够像真实人手一样做出各种姿态，并能够与虚拟环境进行实时交互。

5.2 研究展望

由于时间和个人知识水平的限制，本文的研究工作仍有地方需要进一步的改进和提高：

(1) 在通过背面人手图像对模型进行调整时，我们是用手工的方式进行调节的。这部分工作其实可以采用基于径向基函数的方式对模型进行自动调节，目前这块工作已经开展了一部分。

(2) 真实人手在进行抓捏等操作时肌肉会有明显的变形，比如折叠等，而这些变化并不能反应在我们的虚拟人手上，下一步的工作需要将人手的生物特性考虑进去。

发表文章目录

- [1] 1. Shuai Ren, Hao Wu, Yi Wan. Construction of Virtual 3D Life-like Human Hand for Real-time Human-Computer Interaction. in Proceedings of the 7nd International Conference on Advanced Computational Intelligence, 2015.

致 谢

时光荏苒，岁月如梭，眨眼之间三年研究生生活即将结束。这三年是我人生当中最宝贵的三年，无论是在学习上还是在生活中，我都收获了很多。当然这些收获都是来自于教导过我的导师，帮助过我的同学以及一直爱我的父母，在这里我向你们表示最真诚的谢意。

首先我要感谢我的导师万毅教授，他渊博的科学知识、严谨的工作作风感染着信号所的每一名成员。是他为我指点了科研上的点点滴滴，让我在遇到困难时不放弃。而在生活中，他更是一位难得的严师慈父，是他教育我要开朗乐观，积极面对生活。他乐于助人的高尚品质无时无刻不感动着我。

同时我还要感谢信号所的每一位师兄师姐，师弟师妹，是他们为我营造了良好的学习氛围与生活环境，感谢他们给予我的每一个帮助。三年中我们一起努力一起奋斗，有欢笑，有泪水，正是因为你们的存在才让我在远离家乡的陌生城市里有家的感觉。

其次，我要感谢我的母校兰州大学，感谢哺育了我七年的成长摇篮。在这七年中，是您见证了我从青涩走向成熟，是您陪伴我走过了七年的风风雨雨，在我有能力时我一定回来报答您。

还要感谢本文所涉及的所有学者，没有你们研究成果的帮助和启发，我很难完成本论文的写作。

最后，我要感谢我的父母，感谢你们二十几年的养育之恩，你们是我最强大的精神支柱。尤其感谢我的母亲，是您撑起了整个家庭，在父亲病重时是您的悉心照料才使得他能早日康复。您对我的养育之恩我将用一生来回报。