浙江大学

本科生毕业设计报告

设计名称 试衣机器人的参数化三维人体模型设计

姓名与学号 3100100423 马轲

专　　业 工业设计双学位

学　　院 国际设计研究院

A Dissertation Submitted to Zhejiang University for the Degree of Bachelor of Engineering

TITLE: The Parametric 3D Human Body Model Design of Fitting Robot

Author: Ma Ke

Major: Industrial Design Double Degree

College: International Design Institute

Submitted Date: May 31st, 2014

浙江大学本科生毕业论文（设计）诚信承诺书

1.本人郑重地承诺所呈交的毕业论文（设计），是在指导教师的指导下严格按照学校和学院有关规定完成的。

2.本人在毕业论文（设计）中引用他人的观点和参考资料均加以注释和说明。

3. 本人承诺在毕业论文（设计）选题和研究内容过程中没有抄袭他人研究成果和伪造相关数据等行为。

4. 在毕业论文（设计）中对侵犯任何方面知识产权的行为，由本人承担相应的法律责任。

毕业论文（设计）作者签名：

_____年_____月_____日

摘要

本设计是“试衣机器人与虚拟试衣系统”课题的子项目之一。本设计通过独创的方法进行了参数化的三维人体建模，并完成了三方面的工作：其一是编写交互界面引导用户输入尺寸参数，并得到三维人体模型形式的实时反馈；其二是能够将关键数据传递给试衣机器人的控制部分，引导机器人变形至特定形状；其三是能够导出可加工的三维模型文件。本设计中的参数化三维人体建模，基本思路在于先由尺寸参数驱动基准数据变形生成关键线条，再由关键线条按照保持形状的插值方法生成其他线条，最后对各线条进行均匀的三角化从而生成表面，进而得到三维人体模型。

关键词 参数化，人体建模，试衣机器人，尺寸驱动，形状保持插值，服装 CAD

Abstract

This design is one of the sub-projects of the topic Fitting Robot and Virtual Fitting System. This design achieves parametric 3D human body modelling with an original method, and also accomplishes three major tasks: The first is to program an interface that guides the user to enter the size parameters, and shows the user a real-time 3D human body model as feedback. The second is to pass the key data to the control module of the fitting robot, and make it deform to fit a specified figure. The third is to export a manufacturable 3D model file. The basic idea of the parametric 3D human body modelling in this design, is firstly to generate the key lines through size-driven deformation of the reference data, secondly to generate other lines using a shape-keeping interpolation method, and finally to uniformly triangulate the lines to generate the surfaces, so that the 3D human body model can be acquired.

Keywords Parametric, Human Body Modelling, Fitting Robot, Size-driven, Shape-keeping Interpolation, Garment CAD

目录

第1章 设计背景

1.1 绪论

随着社会经济的快速发展，网络购物已成为当前消费者的主要购物方式之一。与一些能够提供明确的性能、服务的商品不同，服装作为一类特殊的商品，其价值与消费者本身是息息相关的。对于同一件服装，不同消费者穿着后会体现出不同的效果，这也影响着消费者的消费决策。因此，消费者更倾向于在购买服装时就能看到自己穿着这些服装的效果，这对服装网购服务提出了更高的要求。

为了迎合这一需求，试衣系统应运而生，它主要包括两类解决方案：第一类全部利用三维计算机图像仿真来实现，使得用户能够得到完全符合自身尺寸的三维模型和实时、全角度的仿真结果，但这类技术成本太高，且仿真结果往往缺乏真实感；第二类则基于物理服装人台，通过将不同尺寸的服装穿着在不同尺寸的人台上进行拍照，从而展示给消费者，这保留了服装的质感，但物理人台种类有限未必能模仿所有消费者的体型。

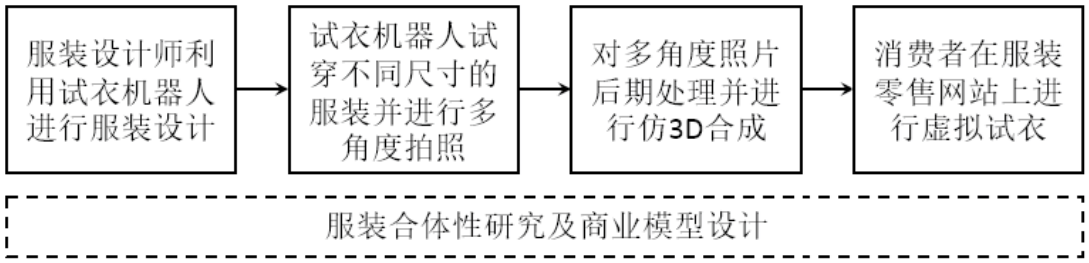

图 1-1 试衣服务体系流程图

本课题意在设计一个完整试衣服务体系，从而解决传统试衣系统解决方案的一些弊端，如图 1-1 所示。整个体系主要分为两个部分：试衣机器人和虚拟试衣系统。前者的主体为一个可变尺寸的机器人及其配套软件，厂商可以在设计服装时使用该机器人辅助裁剪，也可以在销售服装前将服装穿着在机器人上拍摄效果；后者的主体为拍摄装置和能进行效果展示的服装销售网站，厂商利用拍摄装置拍摄多角度的服装穿着效果，去除拍摄背景并以合成 3D 的效果在网站内展

示。除了系统开发的子项目之外，本课题还包括了服装合体性研究和商业模型设计的辅助性子项目，它们同样在课题中占据重要地位。

1.2 参数化三维人体模型

本设计“参数化三维人体模型”是“试衣机器人与虚拟试衣系统”课题的子项目之一，主要应用于试衣机器人及其配套软件中。

三维人体建模技术是计算机图形学领域的一个长期的研究热点。最早的虚拟三维人体模型是由三维线框来表示的，它结构简单、操作方便，但具有无法实现三维人体模型自动消隐和具有二义性的缺点。20 世纪 70 年代后实体模型逐渐发展，克服了上述缺陷，能够完整的表达人体模型的拓扑关系并生成具有真实感的图像，但结构复杂，应用受到限制。曲面建模作为当今最常用的人体建模方法，适用于表面光滑但难以用简单数学模型描述的复杂曲面。N.M. Thalmann 和 D. Thalmann（1987）最早使用多边形表面生成了虚拟人体。Forsey（1991）使用分层 B 样条技术进行三维人体建模。Douros 等（1999）则使用 B 样条曲面重建三维扫描人体。此外，A.H. Barr（1987）提出引入人体的物理特性，而非仅仅使用几何特性用于人体建模，使得三维人体的动画仿真效果更为真实。Chadwick 等（1989）提出使用多个层次进行人体建模，从而使人体模型在生理学的角度更为真实。Thalmann 等（1996）在此基础上提出了一种基于解剖学的分层建模算法。

本设计的参数化三维人体模型的构建是试衣机器人的核心技术之一，在整个课题中主要承担三方面的作用，如图 1-2：

其一，在试衣机器人控制软件中以三维人体模型的形式实时显示变形效果。由于试衣机器人的控制较为复杂，通过控制软件调整试衣机器人的尺寸是有一定延时的；使用几个控制杆来调整人体的各项参数也并不直观。从人机交互的角度考虑，用户希望能够在改变人体某项参数的同时，能够立即看到这种改变所带来的效果，因此，一个能够根据参数进行实时变化的三维人体模型是有必要的。用户可以通过控制杆来调节人体参数，并通过观察三维人体模型得到实时的反馈，在得到满意的调整结果后再另试衣机器人进行变形。

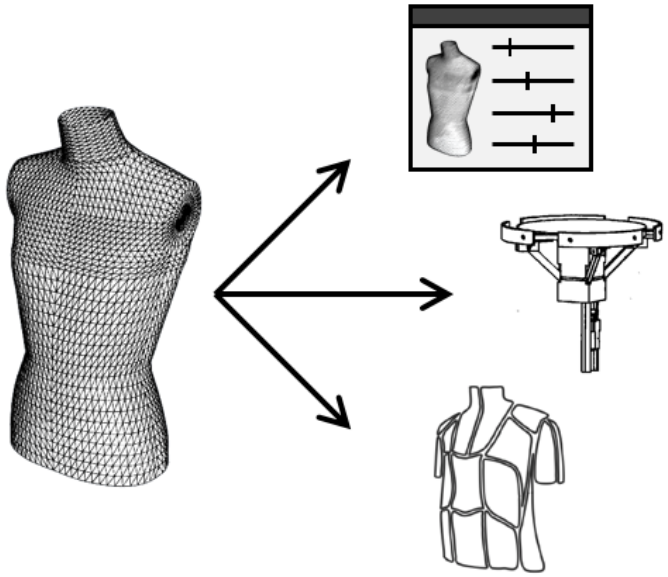

图 1-2 参数化三维人体模型的作用

其二，为试衣机器人的电动控制提供人体外型数据用于变形参考。在获取了满足一定尺寸参数的三维人体模型后，模型表面的每一个点的坐标都是可以求得的。在模型表面提取一些关键点，将这些点的位置传递给试衣机器人的电动控制部分，从而使试衣机器人能够更为精确地拟合该尺寸参数下的人体外形。

其三，为试衣机器人提供外壳的三维模型。试衣机器人需要用满足人体曲面的分块外壳将内部的机械结构包覆住，从而更好地拟合人体外形。这种外壳可以通过由参数化人体模型导出的可加工模型文件，利用一定的加工工艺进行获取，例如 3D 打印技术等。

1.3 本文章节安排

本文主体部分为层层递进的四个章节：第二章为方案构思，分析了本设计任务的一些主要工作，和这些工作的一些解决思路；第三章为设计细节，对设计方案进行了一些调整和深化，阐述了完成设计任务的具体方法和相关约定；第四章为方案实现，从实际制作的角度具体描述了设计任务的进行过程；第五章为设计展示，以图片的形式全面展示了设计任务中各个方面的成果。第六章则为整个设计的全面总结和对未来工作的展望。

第2章 方案构思

2.1 设计创新点及设计源点

传统的试衣系统解决方案往往把计算机图像仿真和物理服装人台视为相互对立的存在，前者是虚拟的、仿真的，后者则是现实的、物理的。但事实上，两者是可以相互结合的，实现优势互补。本课题“试衣机器人与虚拟试衣系统”正是对这种融合进行了探索，其中参数化三维人体模型则起到了将两种技术进行连接的桥梁作用。

图 2-1 三维图像仿真解决方案示意

大部分的计算机图像仿真解决方案，在得到了满足一定尺寸参数的人体模型后，会继续利用一些变形算法将服装贴合到人体模型表面，从而实现虚拟试衣效果，如图 2-1 所示。但这种方法由于种种技术问题的限制，很难达到令人满意的试衣效果，因为它很难表现出服装的材质、纹理和褶皱，因而也体现不出服装穿

着在身上是否合体。本课题在利用一定的技术获取参数化三维人体模型后，不仅将模型的数据进行显示，还将模型的数据用于试衣机器人的设计和控制过程；实际的试衣工作则由试衣机器人从物理上进行实现；对试衣结果进行多角度拍照，通过一定的图像处理操作，使结果成为伪 3D 合成图像。这种“虚拟—现实—虚拟”的过程，如图 2-2 所示，结合了两种传统解决方案的有点，既确保了真实的试衣效果，又减轻了服装厂商和消费者的负担，可以认为是课题的一大创新点。

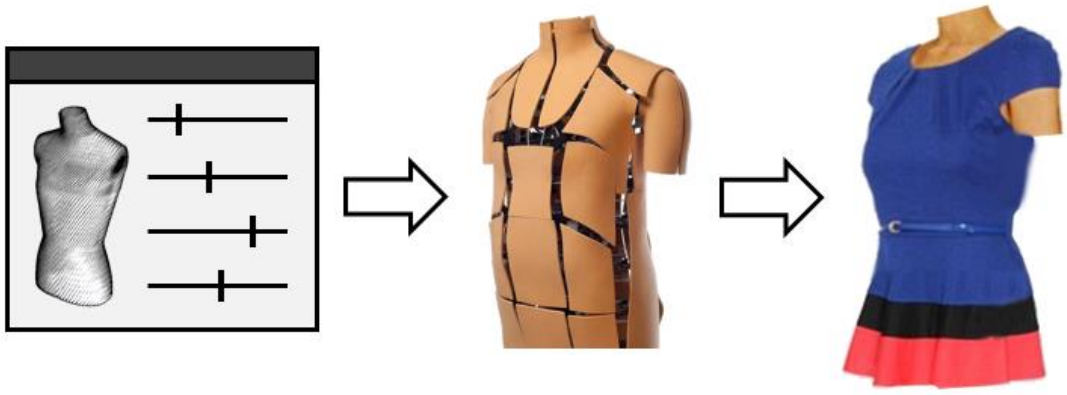

图 2-2 本课题的“虚拟—现实—虚拟”过程

单从参数化三维人体模型的角度来讲，本设计的创新点在于结合了多种人体模型的构建方法。本设计基本思路服从参数化曲面建模的方法，在人体的大量尺寸参数中，选取了一小部分对体型影响最大、最为关键的尺寸参数作为基准，其他参数则由这些基准参数通过固有的比例关系进行推演而得到。同时，本设计汲取了基于特征的人体曲面建模的一些基本思想，将三维人体模型按照不同的特点分成了两个部分，对两个部分分别进行模型构建。本设计中的三维人体模型同样使用小的三角面片来逼近曲面，这与多面体建模的基本原则相符。

2.2 设计概念草案

2.2.1 工作概览

根据前述的参数化三维人体模型在整个课题中的作用，本设计需要进行如下方面的研究：

（一）提取男半身人台的关键尺寸参数。这些参数将会在三维人体建模时使用，同时也将是试衣机器人控制程序的可调参数。所提取的参数应当是对人台体型有重要影响的参数，同时也应当是一些常用的、易于测量的参数。

（二）研究男半身人台的表面分块方法。优良的分块方式可以使试衣机器人对体型的拟合更为精确，同时还要考虑到设计的可行性，并与试衣机器人内部机械结构的设计紧密配合。

同时，本设计需要研发一个程序，它满足以下要求：

（一）引导用户输入和调整人台关键尺寸参数，并实时显示出对应的三维人体模型。用户的控制界面应当满足人机交互的基本需求，直接、美观、易用；三维人体模型应当准确满足关键尺寸参数的约束，同时允许用户进行多角度的查看。

（二）将关键尺寸、位置信息传递给试衣机器人的控制部分，使得机器人变形至相应的体型。找到对于试衣机器人的变形控制最重要的一些关键点，将关键尺寸连同关键点作为参考信息输入机器人控制部分，使得试衣机器人尽可能好的拟合指定体型，并与三维人体模型的变形同步。

（三）将三维人体模型保存为可加工的模型文件。试衣机器人的外壳需要符合人体的曲线，因而应以人体模型为基本形状进行加工。

（四）由于试衣机器人的控制部分是以网页为终端的网络控制，该程序应当能够嵌入到网页中，从而与机器人的控制部分进行协同工作。

2.2.2 建模方法

人体建模方法主要包括采用点线构造三维图形的线框建模、增加三维人体实心部分表达的实体建模、基于光滑曲面数学描述的曲面建模以及引入了人体物理信息和环境时间因素的基于物理的建模方法。而对于用于服装 CAD 的人体模型，则包括一些更为具体的要求。当前的服装 CAD 领域的人体建模技术主要有四种：基于特征的人体曲面建模、参数化曲面建模、多面体建模和以网格边界线为连续条件的三维人体建模。

基于特征的人体曲面建模首先将人体模型划分为数个基本结构特征，包括头、躯干、手、脚等，而每个部分的数据结构和造型方法可能并不相同。这种方法打破了对整个人体模型采用同一种曲面表达的禁锢，使得为不同的特征部位采用不同的曲面建模方法成为可能，因而这种方式相对更为灵活，也易于对某个部位的模型进行修改而不影响其他部位。这种方法通常先由测量得到的身体的特征点形成特征曲线，再与具有一定通用性的几何造型曲线共同形成网格，进而生成曲面。

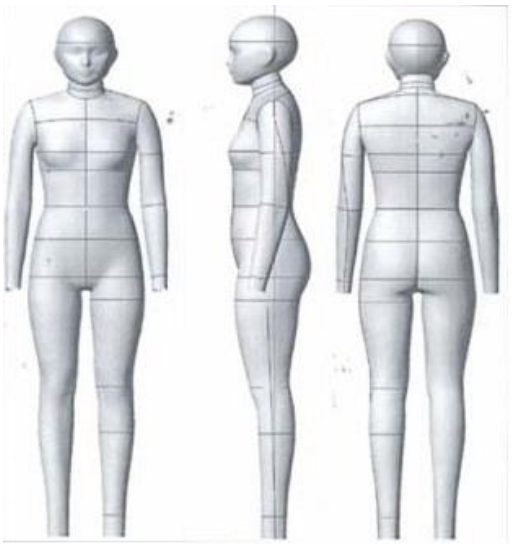

图 2-3 参数化曲面建模

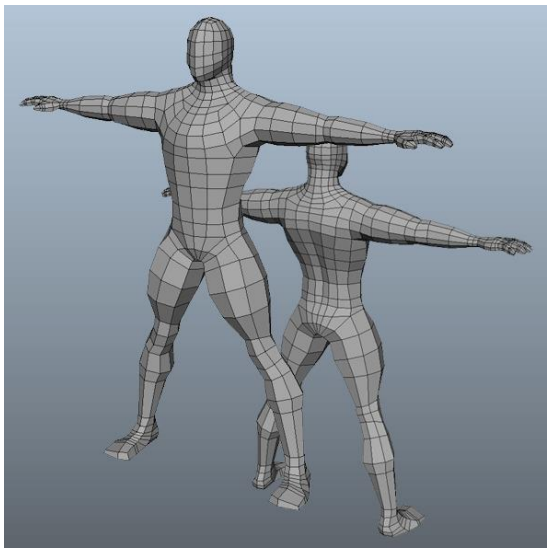

图 2-4 多面体建模

参数化曲面建模是利用人体的形状特征间的几何约束关系来进行人体建模的一种方法，它可以以一个统一的模型表达一组在形状上具有相似性的模型，如图 2-3。参数化建模着重考虑人体各个部分之间固有的比例关系，在由人机工程学中定义的诸多人体测量参数中挑选出最具决定性的数个参数，再由主要参数根据相关约束关系推演出其他造型参数，从而获得满足一定参数约束的人体模型。

多面体建模是以一种小平面逼近曲面的思想进行的人体建模方法，如图 2-4。它通过首先构造多面体，再对多面体的定点、棱线、表面进行局部修改以达到与目标实体外形近似的目的，而后由多面体的顶点自动产生自由曲面的控制点，从

而生成人体模型表面。在于当前的 CAD 系统大多由三角或四边形网格表面来表达曲面，这一类建模方法在 CAD 系统中尤为常用。

以网格边界线为连续条件的三维人体建模是基于曲线的建模和基于三角面片的建模的结合。它首先根据人体的特征构造出横向和纵向的 B 样条曲线，形成空间的曲线网格，再将这些曲线离散化，形成空间中的人体模型表面点云，最后利用这些点作为三角面片的定点拼合构成三维人体表面。

如前所述，本设计结合了多种建模方法的基本思路。以参数化曲面建模为基础，建立一个通用的模型来表征一系列模型，这些模型的区别在于一些关键尺寸的差异；利用基于特征的人体建模的思想，将三维人体模型分成了由截面环和截面线组成的两类；利用多面体建模的思想，用小的三角面片逼近曲面，形成模型的表面。

2.3 设计构思草图

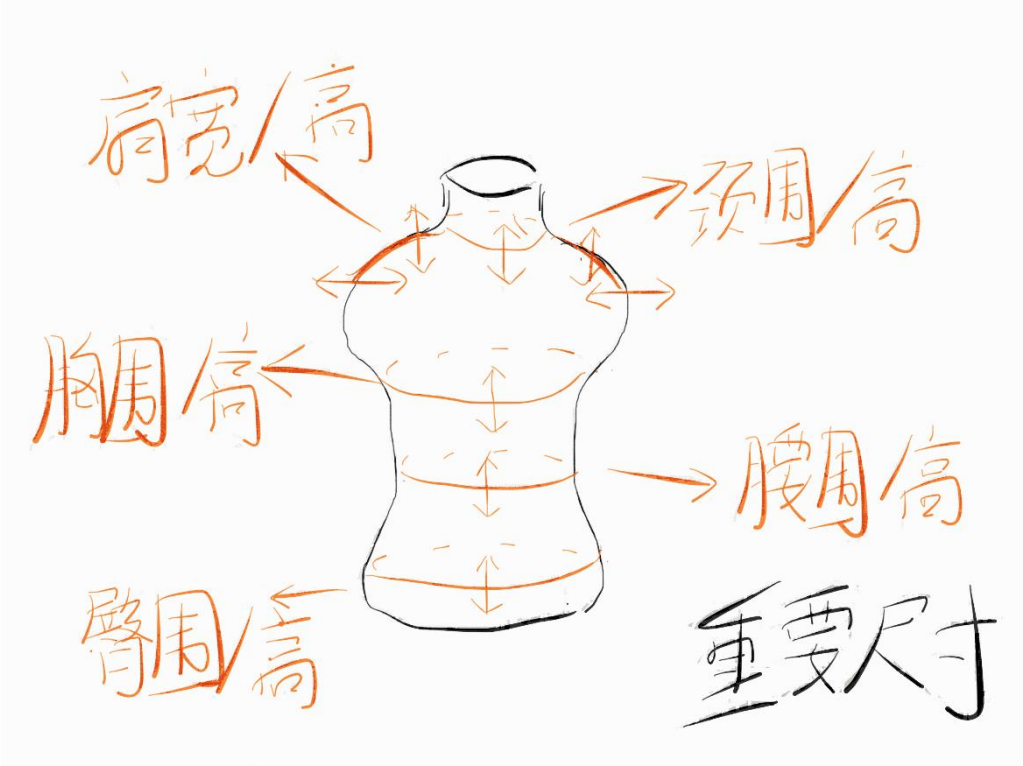

图 2-5 重要尺寸的构思草图

在设计的构思阶段，根据本设计的主要任务构思了一些解决思路。

对于三维人体模型，在此处特指男半身人台，选取了一些具有典型性的关键尺寸。关键尺寸主要选取了一些在选购服装时经常会测量的尺寸，包括胸围、腰围和臀围。除此之外，对于衬衫等颈口较小的服装，颈围和肩宽也是影响服装合体性的两个重要人体尺寸，虽然通常不会经常测量它们，但为了构建完整合理的人体模型也将其视为关键尺寸。上述的五个尺寸，其高度也会对人体模型产生巨大的影响，例如有些人腰部与臀部之间较长，有些人则较短。颈高、肩高、胸高、腰高、臀高这五个尺寸就决定了这几个部位之间的相对距离。因此，三维人体模型主要由这五组、十个参数决定形状，如图 2-5 所示。

图 2-6 表面分块的构思草图

为了使试衣机器人能够变形，需要对人体模型进行合理的分块，每一块都由内部机械结构进行控制，从而在整体上拟合人体的形状。对人体模型进行分块时既要考虑尽可能准确地拟合人体形状，又要考虑到分块数不能过多，以免机械结构过于复杂。因此，将整个人体模型分为 18 块，从而能够通过变化表现出前面

确定的重要尺寸的改变，如图 2-6。其中，颈部、腰部和臀部构成类似，均由 4 个分块（左前、左后、右前、右后）构成。由于它们均组成环状，可以通过开合来改变围度，通过上下移动来改变高度。胸部同样由 4 个分块（左前、左后、右前、右后）构成，运动方式也基本相同，但不同之处在于它需要与肩部相互约束。肩部由 2 个分块（左、右）构成，可以通过左右移动来改变肩宽，倾斜运动来改变肩高。值得注意的是分块与重要尺寸的位置关系。除了颈部分块的下围即是颈围外，其他围度均在分块的中部：肩宽是在肩部分块中间从左端到右端的长度，胸围是胸部分块中间位置处的周长，等等。这样的设计是为了一方面更准确地拟合人体外形，另一方面也赋予每个分块更为明确的意义。

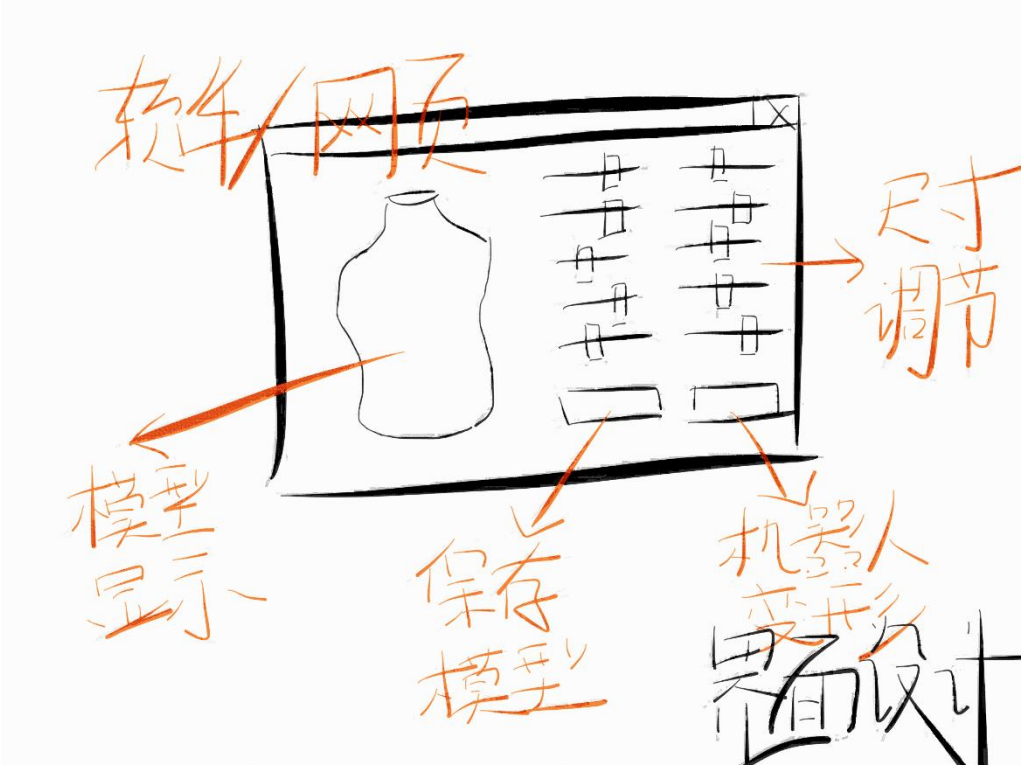

图 2-7 界面设计的构思草图

界面设计部分，首先要确定的是程序的形式：既可以是软件，方便本地使用；又可以是网页，可以嵌入到网站中，实现在线操作。整个程序的界面包括左右两个部分，左半部分用于显示，右半部份用于控制，如图 2-7。关于模型的显示，应当能够渲染出三维人体模型，并打上合适的光照，最好能够通过鼠标拖拽或是

其他形式来改变视角，从而全方位地观察三维人体模型。控制部分则包括对前述的十个关键尺寸的控制和一些其他功能。关键尺寸可供选择的输入方式包括文本框输入和滑动条拖拽等，最终选择滑动条是因为一方面操作方便、显示直观，另一方面也限制了输入数据的范围，可以防止无效的输入数据。功能按钮主要有两个，一是将当前的三维人体模型保存为可加工的模型文件，可用于加工试衣机器人的外壳；另一个是将相关的数据发送给试衣机器人的控制部分，从而使试衣机器人变形到当前的形态。

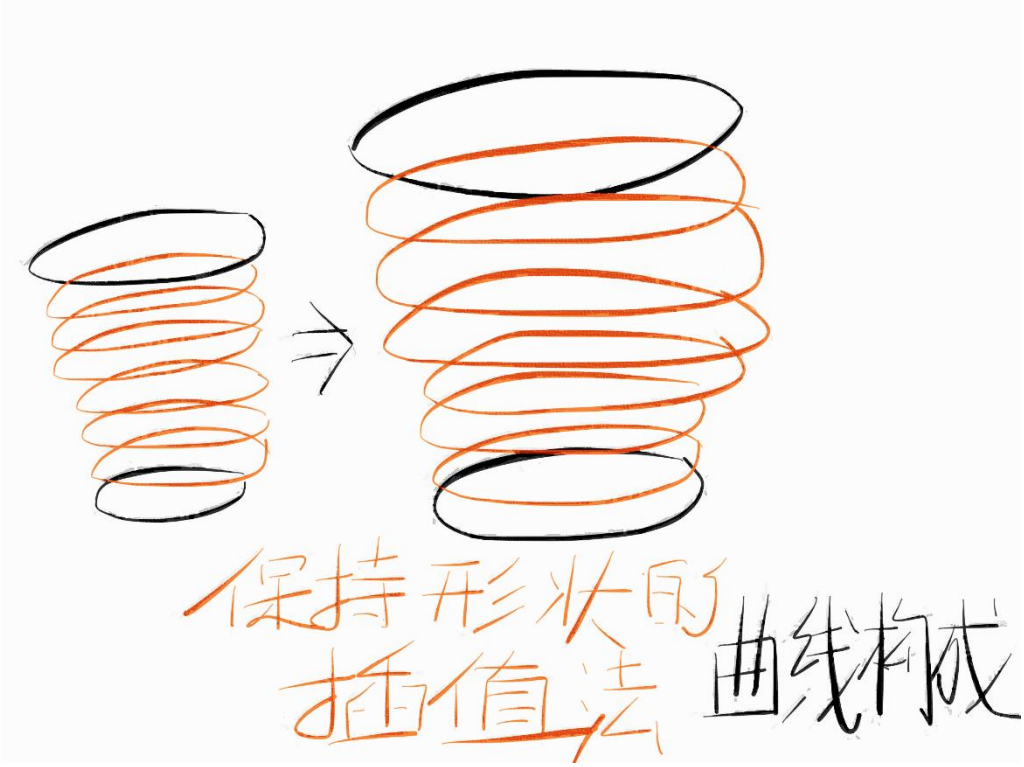

图 2-8 曲线构成的构思草图

如同利用三维建模软件一样，利用计算机程序进行三维人体建模时，也可以按照从点到线再到面的顺序进行。由于已经由输入的关键尺寸参数得到了一些关键截面线，下一步就是通过它们推导出其他的截面线。一个最为常规的方法是插值法，但此处无法使用线性插值，因为人体的每一个部位都不是基本几何形状（圆台）。因而，需要一种能够保持一定的外形比例关系的插值方法，来得到关键截面线之间的截面线，如图 2-8。

接着要由大量的紧密的截面线得到表面。设计采用三角化的方法来进行，如图 2-9，这是因为此类方法实现较为简单，同时也易于导出为可用于 3D 打印加工的模型文件。由相邻的截面线进行三角化，只需要将两条线进行离散化，再将得到的点进行一定的连接即可。由于要保持人体模型的左右对称性，三角化时也需要注意三角形的走向。

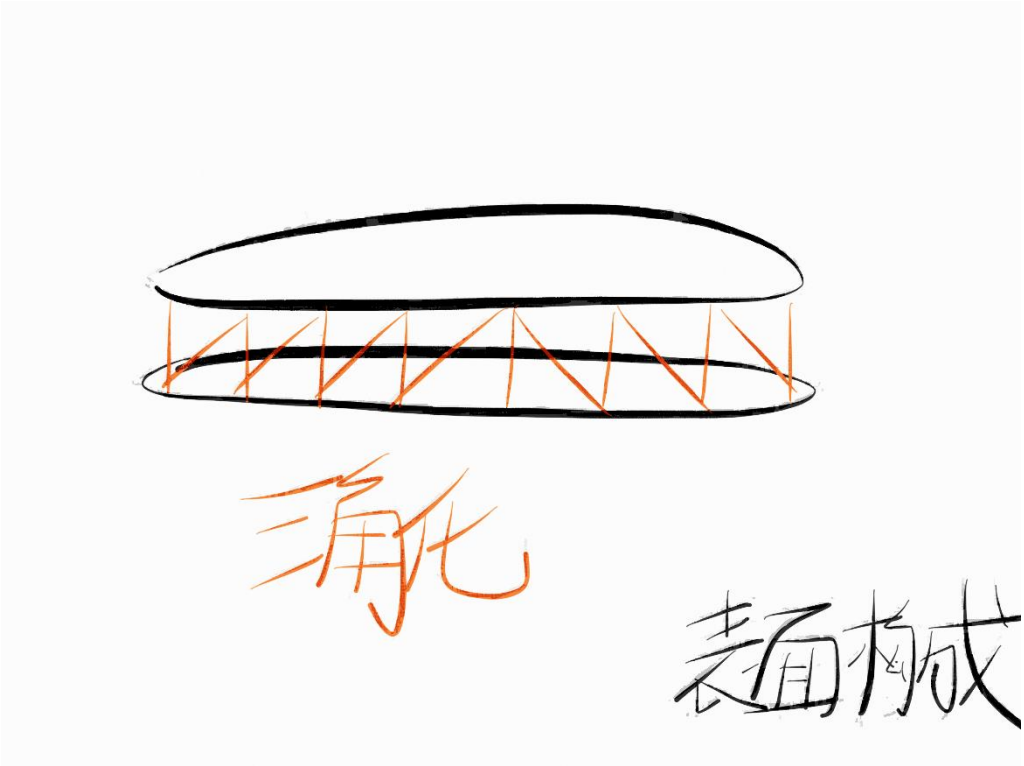

图 2-9 表面构成的构思草图

第3章 设计细节

3.1 设计方案深化

在设计的初期方案中，对关键参数的选择、人体模型的分块、用户界面的设计、三维人体模型的建模方法等方面进行了研究和探索。但在设计的进行过程中，由于一些技术条件的限制，和与课题中其他项目的相互协商，对设计方案进行了一些修改和深化。

在关键参数的选择方面，用于人体三维建模的参数取消了影响较小的肩高，保留九项关键尺寸参数；由于试衣机器人的机械结构设计工作量较大，无法在预定的时间内完成针对肩部和颈部的变形结构设计，因此将仅取胸围、胸高、腰围、腰高、臀围、臀高六项参数作为试衣机器人的可调参数。

在人体模型的分块方面，基于上述原因，腰部和臀部分块方式保持不变；而胸部和肩部进行合并，共分为 4 块；颈部不进行分块，保持环状。

在用户界面的设计方面，采用了既可以作为软件单独运行，又可以嵌入到网页中的 Java Applet 开发程序。同时，将改变模型观察视角的方式改为易于实现的滑动条方式，并对界面右侧的控制滑动杆和功能键进行了分类包装。

在三维人体模型的建模方法方面，沿用初步设计中的思路，并确定了保持形状的曲线插值方法的具体实现。

3.2 界面效果图

本设计中参数化三维人体模型的程序界面如图 3-1 所示。

界面的左侧显示三维人体模型。三维人体模型具有两种显示模式，一种是线框模式，另一种是渲染渲染，两种模式可以通过按钮快速切换。三维人体模型以一定的速度绕中轴线不断旋转，供用户察看多角度的效果。三维人体模型上以突出的线条标记出了关键截面线，供用户参考。

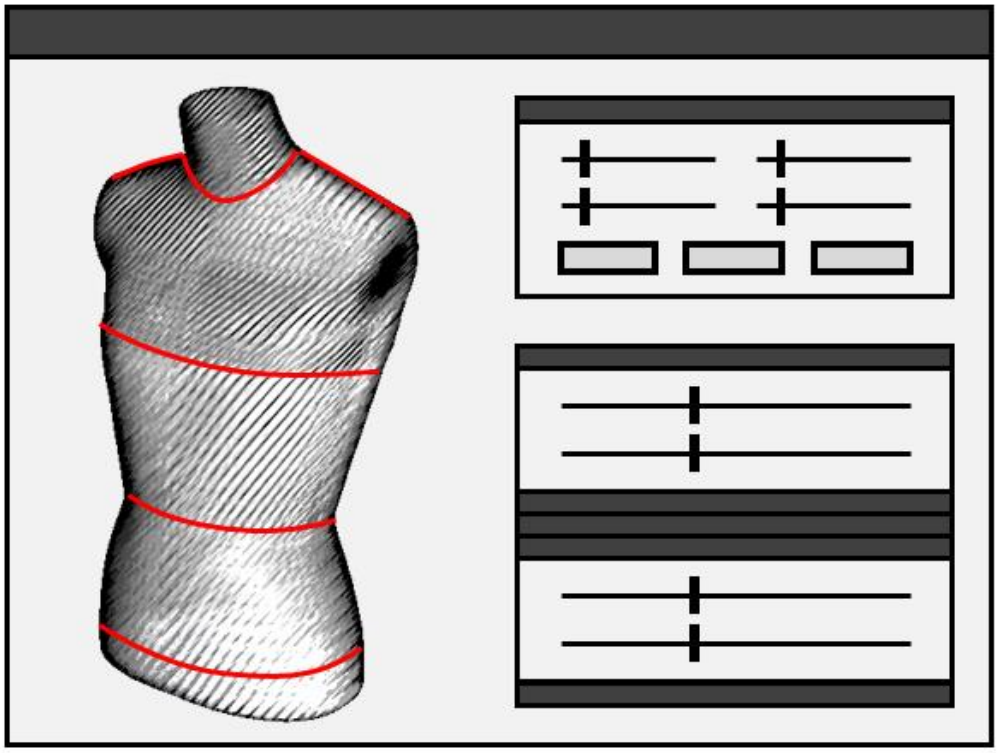

图 3-1 界面设计初步效果图

界面的右侧上方是功能区。主要功能包括模型显示方面的调整，具体有显示比例、旋转速度、显示模式等；此外包括模型导出和控制试衣机器人变形的功能按钮。

界面的右侧下方是调整区。根据部位将 9 个关键尺寸划分为 5 个分区，每个分区可以折叠。尺寸参数由滑动杆进行调整，调整结果可以实时显示在三维人体模型上。每个部位（除肩部）的两个参数，一个是围度参数，直接由具体的尺寸数值确定；另一个是高度参数，由相对偏移量确定，通过标准人体模型变形得到。

3.3 界面尺寸及初始值

根据惯例，程序的界面采用 4：3 的比例。为了能让大多数电脑完整显示程序界面，并保证一定的显示效果，程序的界面尺寸定为 640×480。

对于参数化三维人体模型而言，每个参数应当具有一个初始值。经研究发现，市面上最常见的男半身人台尺寸均为国标 96 码，即颈围 40.3 厘米，胸围 96 厘米，腰围 82 厘米，臀围 97 厘米，总肩宽 46 厘米。对于 4 个高度参数而言，均保持不变，即相对偏移量为 0。

3.4 使用方法

程序的使用方法非常简单。

程序提供两种入口：软件方式入口，双击程序目录下的批处理文件 launch.bat，即可利用 Java 小程序查看器打开程序；网页方式入口，双击程序目录下的网页 applet.html，即可通过网页浏览器打开网页，其中内嵌的 Java 控件将会运行。

用户在需要观看满足某种尺寸参数组合的人体模型时，只需要打开程序，拖动程序右下方的滑动杆进行调节，即可在左侧看到相应的三维人体模型；利用右上方一些滑动杆和按钮还可以改变观看的效果。在需要保存某个三维人体模型为可加工的模型文件时，可以点击保存模型的按钮，程序将把模型保存为 STL 文件。在需要改变试衣机器人的外形时，可以点击机器人变形的按钮，程序将通知试衣机器人变形至同步的形态。

3.5 色彩计划

由于本课题为偏工业级的应用，因此本设计的配色相对较为沉稳。

程序的底色选用白色。一方面白色为底色是计算机应用程序的惯例，另一方面考虑到此程序可以嵌入到网页中使用，白色同样也是大多数网页的底色，使用白色为底色在大多数情况下不会产生很突兀的效果。

三维人体模型的线框表示选用深灰色。这里没有选择黑色的原因在于，黑色太过醒目，与背景不能很好地融合。三维人体模型的渲染表示选用白色。由于光源的存在，模型表面具有阴影，即使使用白色也不会和背景颜色混在一起，而会产生一种较为柔和的效果。关键截面线的标注则采用醒目的红色。

程序的功能区和调整区选用蓝色系。蓝色受到人们广泛的喜爱，同时又具有现代感与工业感，与“控制”的内在含义相符。采用与黑白主色调不同的颜色，可以突出这部分的作用，同时又不至于过于突兀。

3.6 工具计划

本设计主要依赖于程序的实现，因此选用的编程语言、编程环境和类库非常重要。

本设计的编程语言采用 Java，如图 3-2。Java 是一种现代的面向对象的程序设计语言，由 Sun Microsystems 公司推出。Java 的好处在于它较为彻底的面向对象特性，同时包含了大量已经实现的数据结构和相关算法，大大提高了编程效率。此外，Java 具有良好的通用性、高效性、平台移植性和安全性。

图 3-2 Java 程序开发语言

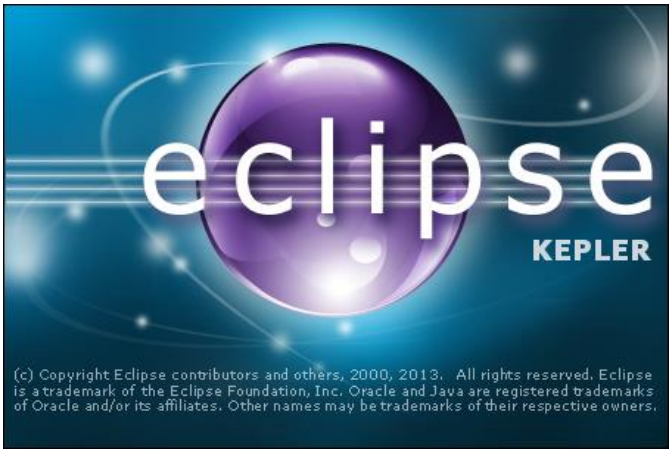

图 3-3 Eclipse 开发平台

基于既可以作为软件使用、又可以嵌入到网页中的考量，本设计所编写的程序为 Java Applet。Java Applet 是使用 Java 语言编写的小应用程序，它可以嵌入到网页中。当用户访问此类网页时，Java Applet 就自动下载到本地，再开始运行，从而保证了良好的运行速度。

集成开发环境（IDE）采用 Eclipse。Eclipse 是一个基于 Java 的开源开发平台，常被用作 Java 的集成开发环境。选用 Eclipse 的原因在于，它对 Java 语言的支持优良，提供了完善的自动补全、代码查错、运行调试等方面的功能。

主要使用的类库为 Processing。Processing 是一种开源的编程语言，主要用于艺术家和设计师进行电子艺术和视觉交互设计的创作，它也具有自己的 IDE，如图 3-4。从本质上讲，Processing 是对 Java 的一次再封装。本设计中牵涉到较为复杂的数学运算，需要定义多个类型，并使用 Java 中的一些数据结构，不适合使用 Processing 的 IDE 这样简单的开发工具进行编写，因此只是在程序中引用 Processing 作为类库。Processing 很好地处理了 Java 与 OpenGL 的关系，将复杂的 OpenGL 引用、调用过程包装成为数十条简单的绘图命令，极大地简化了程序开发过程中的模型绘制流程，这也就是本设计中使用 Processing 的原因。

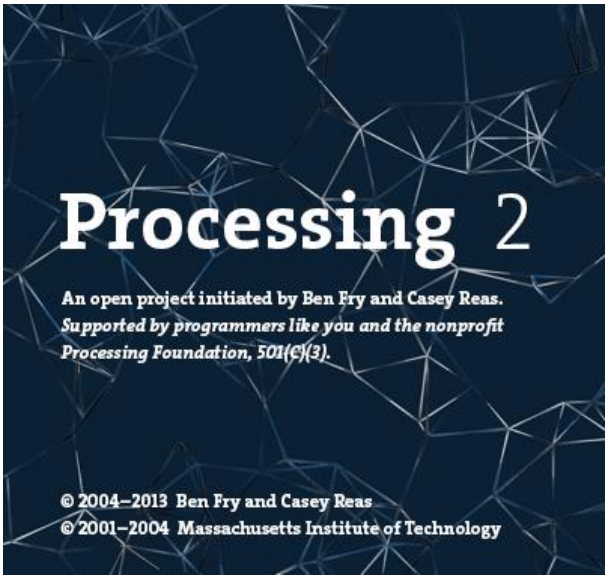

图 3-4 Processing 编程语言

在程序中绘制图形用户界面（GUI）的类库选用 ControlP5。ControlP5 实现了许多常用的控件，可以供用户以对象的形式方便地创建、设置和存取。

在程序中负责文件读写的类库选用 Princeton 大学的 Stdlib。Stdlib 对 Java 类库中的一些 IO 对象进行了重新封装，在内部完成了异常的捕获和处理，简化文件读写的编写。

3.7 技术计划

3.7.1 基准人体模型

为了使程序生成的三维人体模型具有人的体型，原始数据不能凭空生成，而是利用一个最初的模型进行变形而得到，这个最初的模型即为基准人体模型。本设计中所使用的基准人体模型如图 3-5 所示。

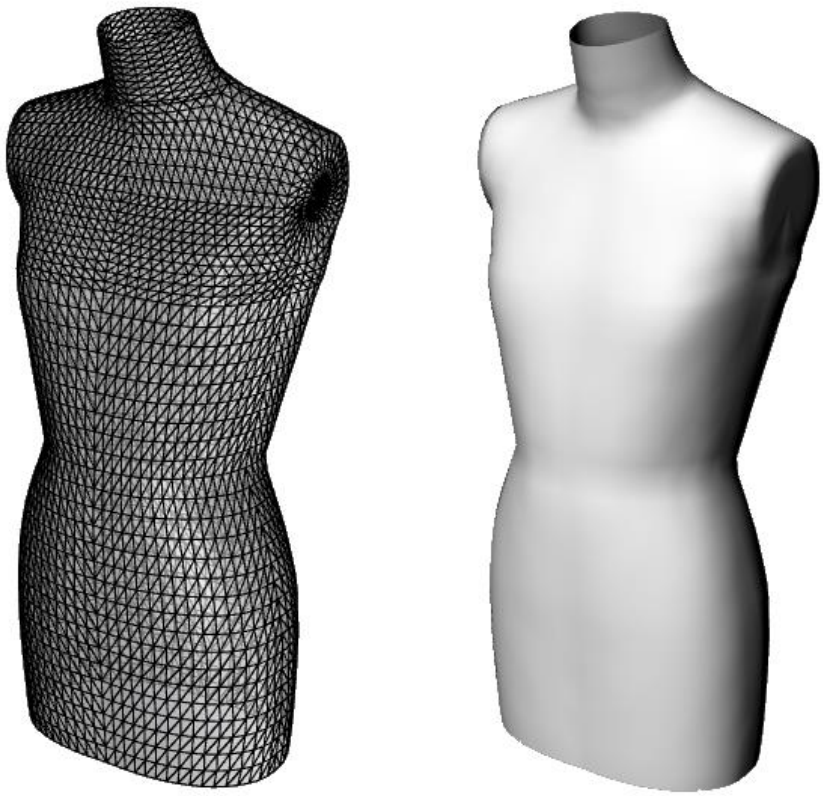

图 3-5 基准人体模型的线框图和渲染图

采用本基准模型的好处在于，它本身就是一个三角网格模型，与参数化三维人体建模的目标模型种类相同；模型的节点和网格非常整齐，为后面的截面线点坐标提取和截面线的插值提供了方便。

3.7.2 关键截面线

从基准网格模型中提取出前述的关键截面线，这些线均是网格模型上本身就有的线，它们通常具有一定的特点。

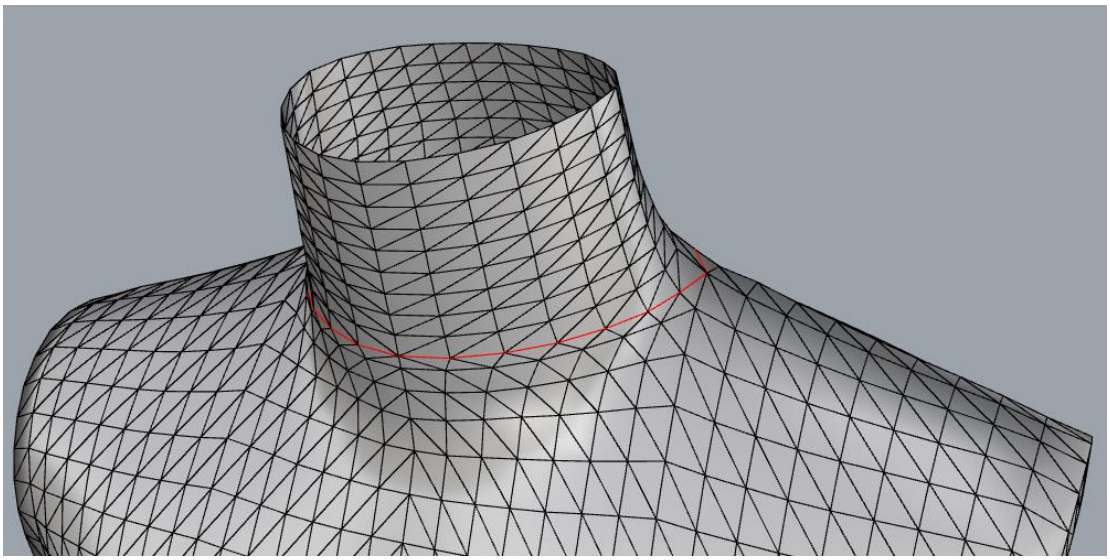

图 3-6 基准网格模型中的颈线

对于颈线，它是颈部最下方的线。在此处，上下网格的三角面片方向发生了变化，可以很快定位到这一条线，如图 3-6 所示。

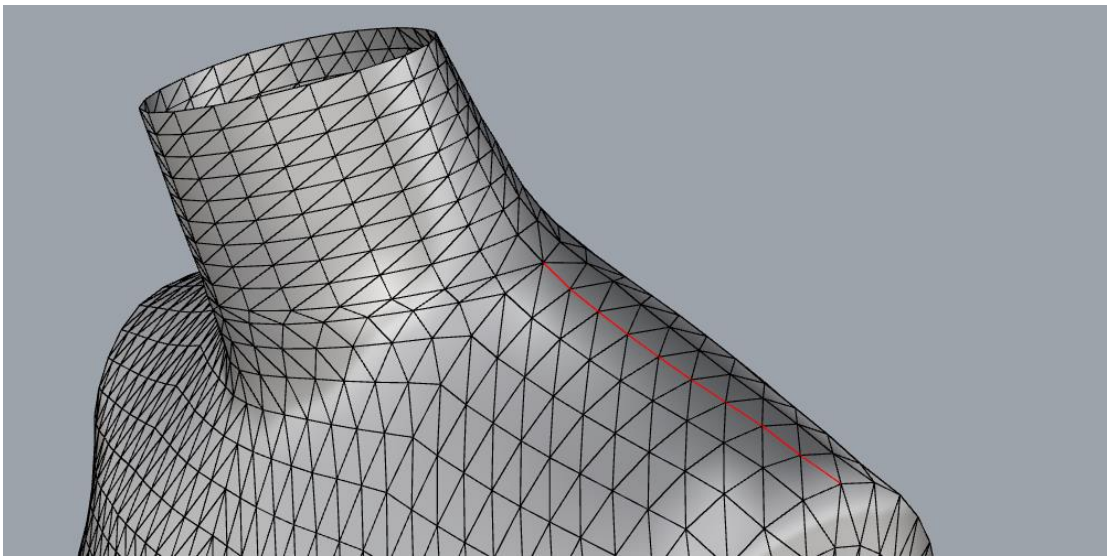

图 3-7 基准网格模型中的肩线

对于肩线，它是沿着肩部最高点的线。它的一端是颈线的左右各一个的拐点，另一端是臂环的最高点；在此处，前后网格的三角面片方向也发生了变化，如图 3-7 所示。但值得注意的是，肩宽并非两肩线的端点之间距离，而是两肩线与颈线后半部分的长度之和。

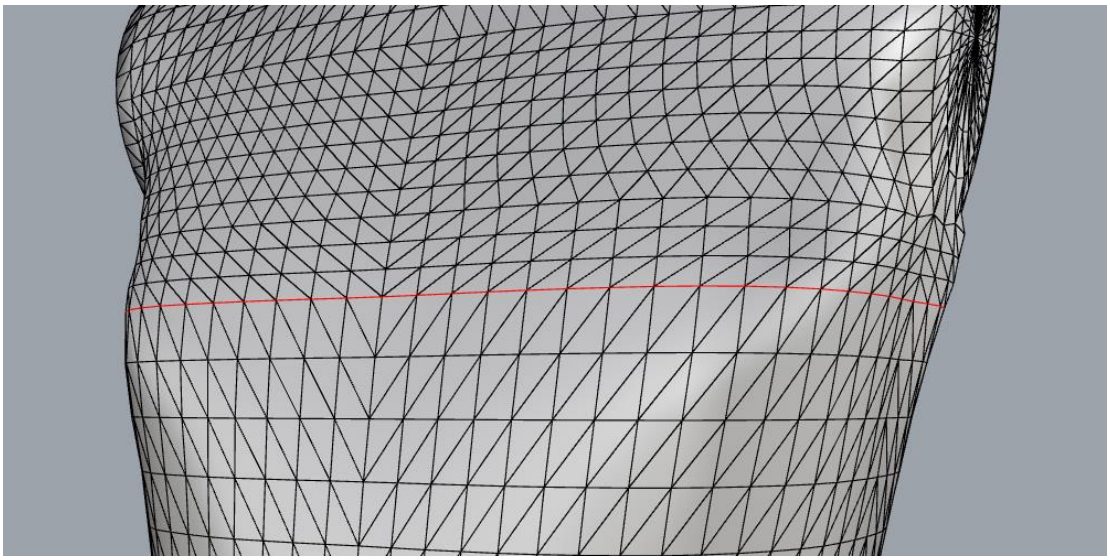

图 3-8 基准网格模型中的胸线

对于胸线，它是胸部沿着最突出处的线，也即胸部周长最大的线。在此处，上下网格的三角面片疏密程度有明显的不同，可用于定位胸线，如图 3-8 所示。

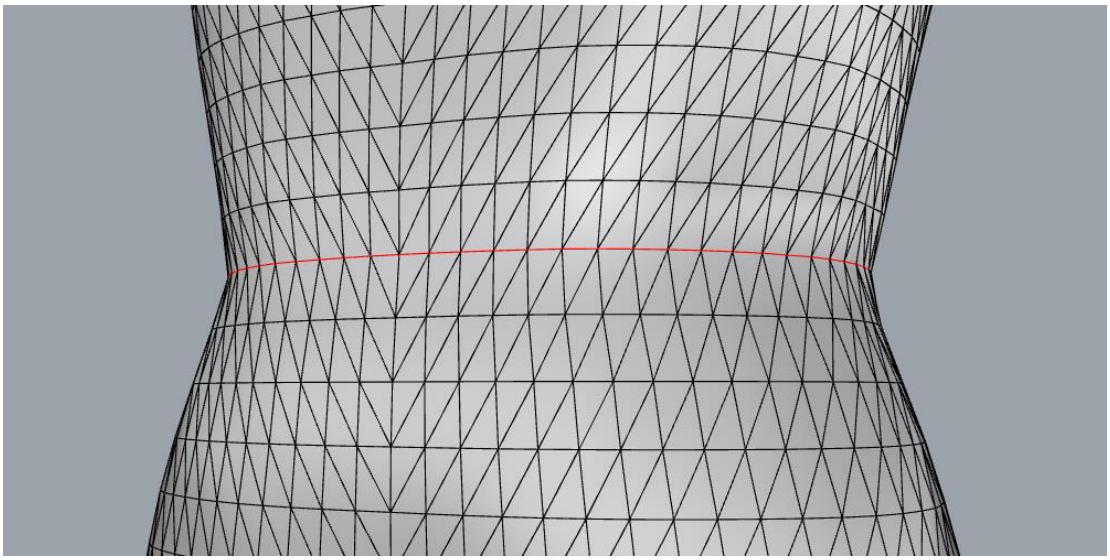

图 3-9 基准网格模型中的腰线

对于腰线，它是腰部沿着最凹陷处的线，也即腰部周长最小的线。在此处，网格表面能看出明显的凹陷，如图 3-9 所示。

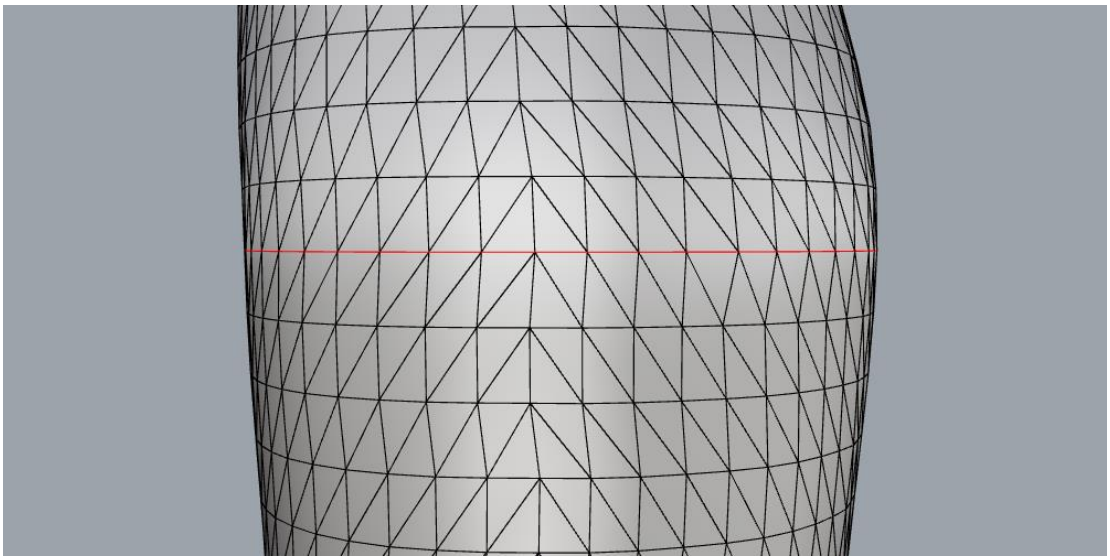

图 3-10 基准网格模型中的臀线

对于臀线，它是臀部沿着最突出处的线，也即臀部周长最大的线。在此处，网格表面能看出明显的突出，如图 3-10 所示。

除了前述的关键截面线之外，在三维人体建模时还需要三条辅助用的截面线，这里也一并说明。

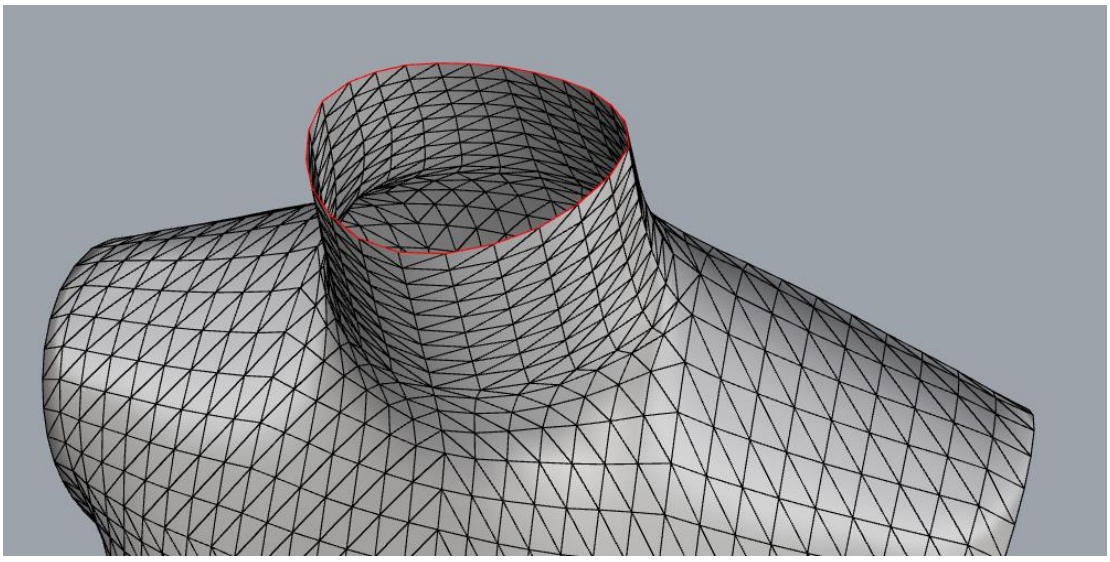

图 3-11 基准网格模型中的颈顶线

颈顶线，颈部最上方的线。它也是基准网格模型最上方的线，如图 3-11 所示。它在三维人体建模中用于颈部上沿的定位和定形，是不随参数而变化的截面线。

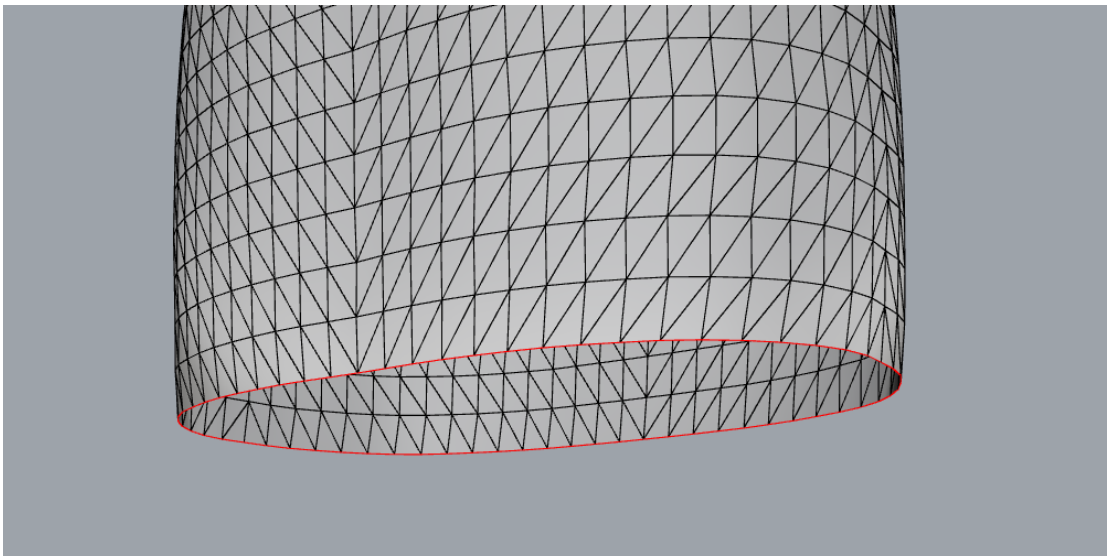

图 3-12 基准网格模型中的臀底线

臀底线，臀部最下方的线。它也是基准网格模型中最下方的线，如图 3-12 所示。它在三维人体建模中用于臀部下沿的定位和定形，是不随参数而变化的截面线。

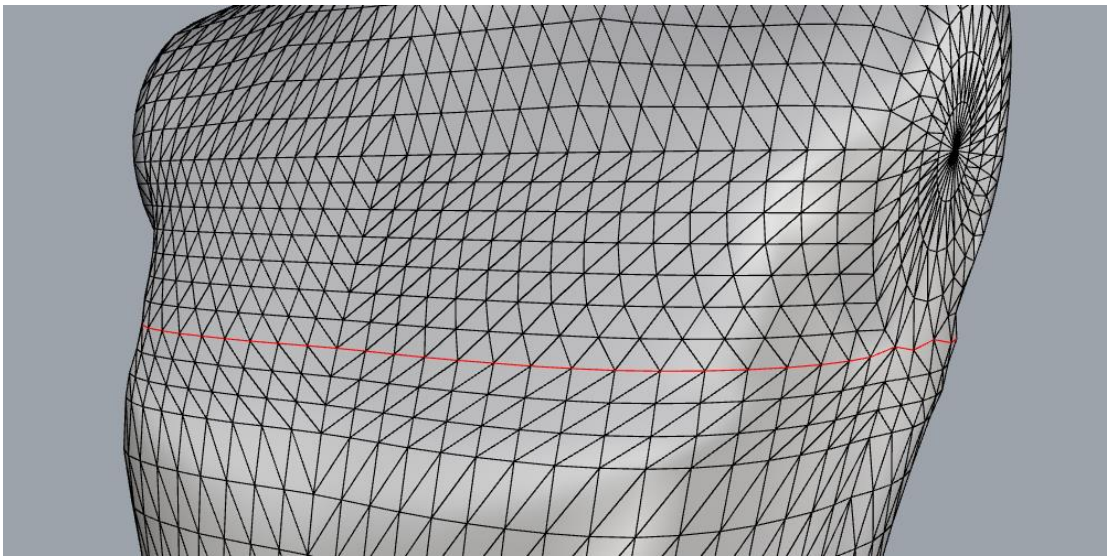

图 3-13 基准网格模型中的腋下线

腋下线，胸部经过腋窝的线。它经过了臂环最下方的点，比较容易定位，如图 3-13 所示。它在三维人体建模中是截面环（下方）和不成环的截面线（上方）的分界线，它会随着肩线和胸线的变化而变化。

3.7.3 截面线顶点坐标提取

有多种方法提取出基准网格模型（OBJ 文件格式）中的截面线，例如直接分析 OBJ 文件，但在这里仅介绍一种直观、简便的方法。

利用 Rhino 软件，打开基准网格模型文件。打开物件锁点，勾选顶点捕捉；利用多重直线工具，将需要提取的截面线逐顶点描一遍；再删去除这条截面线之外的所有对象，导出为 OBJ 格式；用文本编辑器打开 OBJ 文件，其中以 v 开头的行即为所需的按顺序排列的顶点数据，三个字段分别为 X、Y、Z 坐标值。这样就提取出了一条截面线上的顶点数据，如图 3-14 所示。

截面线.obj

```
# Rhino

g object_1
v -14.87174797058106 167.0913848876953 87.84934997558594
v 3.0000001061125660e-06 167.0913848876953 87.35987854003906
v 14.87175369262695 167.0913848876953 87.84934997558594
v 29.71737480163574 167.0913848876953 88.77980041503906
v 44.56918334960938 167.0913848876953 89.45307922363281
v 59.44096755981445 167.0913848876953 89.21923828125
v 74.28659057617188 167.0913848876953 88.18757629394531
v 89.13325500488281 167.0913848876953 87.17098999023438
v 103.7099838256836 167.0913848876953 84.21210479736328
v 117.9947738647461 167.0913848876953 80.08103942871094
v 130.9010162353516 167.0913848876953 73.02996826171875
v 142.4782257080078 167.0913848876953 64.06443786621094
v 152.4607086181641 167.0913848876953 53.32773971557617
v 160.5597381591797 167.0913848876953 41.02032089233398
v 166.5642852783203 167.0913848876953 27.47544097900391
v 170.3258972167969 167.0913848876953 13.09465789794922
v 171.873291015625 167.0913848876953 -1.690418004989624
v 175.3827667236328 169.6531524658203 -15.71033763885498
v 176.4126129150391 167.0913848876953 -31.00722885131836
v 178.2418670654297 169.6531524658203 -46.67941665649414
v 179.4219360351563 167.0913848876953 -62.42675399780273
```

图 3-14 提取的截面线顶点坐标数据

为了保证数据的统一性，需要进行一些约定。对于截面环，约定均从前面正中的顶点开始，逆时针的遍历截面环上的每一个顶点。对于截面线，约定从靠近中央的端点向两边遍历。

3.7.4 关键截面线参数化变形

在已知关键截面线的各个顶点坐标的情况下，需要它能在围度参数的约束下进行变形。大致过程如图 3-15 所示。

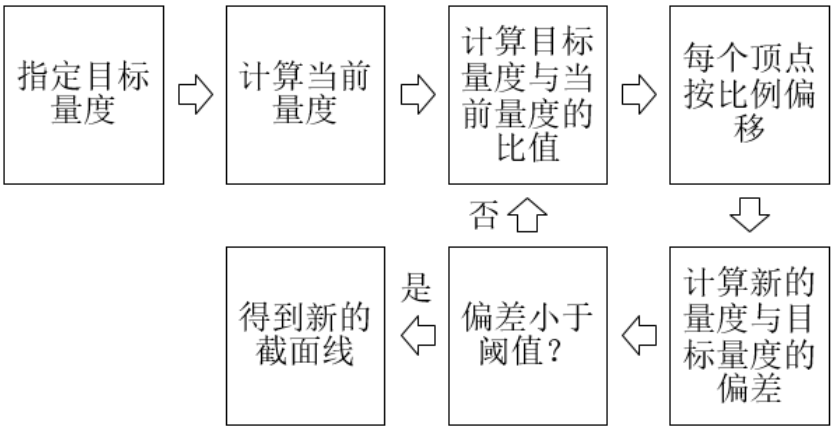

图 3-15 关键截面线参数化变形过程

对于封闭的截面环来说，变形采用了较易于实现的以几何中心为中心的比例缩放。具体为：当需要一个关键截面环变形到某个周长时，首先利用两点间距求和的方法求出当前截面环的周长；再计算目标周长与当前周长的比值；将截面环上的每个顶点以截面环的几何中心为中心按照这一比值进行偏移；计算新的截面环的周长，并求出与目标周长的偏差；迭代此过程，直到偏差小于一定阈值。利用这一方法可以精确地将截面环在目标周长的约束下进行保持总体形状的变形，适用于颈线、胸线、腰线和臀线。

还有一个特殊的截面环需要注意，即腋下线。腋下线的宽度，即最左点和最右点间的距离，是由肩线和胸线决定的，因而在变形时使用的量度为宽度。

对于开放的截面线来说，变形采用了以一个端点为中心的比例缩放，在变形时使用的量度为长度。利用这一方法可以在保持总体形状的同时变形到指定长度，适用于肩线。同时需要注意肩宽的定义。

3.7.5 中间截面线的插值

得到了关键截面线之后，就可以利用保持形状的插值方法来得到中间的其他截面线。在本设计中，每一条插值得到的截面线仅受到其上下的两条相邻的关键截面线的影响。

为了记录人体模型的大致形状，需要事先构建插值数据集。首先需要载入相邻的两条关键截面线的原始数据（即未变形数据），以及中间截面线的原始数据。这些截面线都具有相同的顶点数和排列顺序，它们直接提取自于基准网格模型。之后分别处理每一条中间截面线的每一个顶点。对于每一个顶点来说，记此顶点为 P，在第一条关键截面线上的对应顶点为 P_1，在第二条关键截面线上的对应顶点为 P_2：先计算 P 点与 P_1 点的 Z 坐标之差与 P_1、P_2 点的 Z 坐标之差的比值，作为高度比值 K_1，

$$K_1 = \frac{Z_P - Z_{P_1}}{Z_{P_2} - Z_{P_1}} \tag{3.1}$$

此后的计算均忽略各点的 Z 坐标，仅考虑投影平面上的情况，计算 P_1 点到原点的距离 d_1 和 P_2 点到原点的距离 d_2，

$$d_1 = \sqrt{{X_{P_1}}^2 + {Y_{P_1}}^2} \tag{3.2}$$

$$d_2 = \sqrt{{X_{P_2}}^2 + {Y_{P_2}}^2} \tag{3.3}$$

根据高度比值 K_1 进行线性插值得到 P 点到原点的估计距离 d_{est}，

$$d_{est} = K_1 \times d_2 + (1 - K_1) \times d_1 \tag{3.4}$$

用 P 点到原点的实际距离 d_{real} 除以估计距离 d_{est} 得到平面距离比值 K_2，

$$K_2 = \frac{d_{real}}{d_{est}} \tag{3.5}$$

最后，再计算 P 点的 X、Y 坐标与实际距离 d_{real} 的比值 K_X、K_Y，

$$K_X = \frac{X_P}{d_{real}} \tag{3.6}$$

$$K_Y = \frac{Y_P}{d_{real}} \tag{3.7}$$

对每一条中间截面线的每一个顶点进行上述处理后，每个点对应为四个比值数据，将用于后面的保持形状的曲线插值操作。

在进行曲线插值时，首先要载入相邻的两条关键截面线的点坐标数据，这里的关键截面线可以是变形过的。此后，对每一条中间截面线的每一个顶点进行分别的插值计算。同样地，记此顶点为 P，在第一条关键截面线上的对应顶点为 P_1，在第二条关键截面线上的对应顶点为 P_2：先利用高度比值 K_1 反求 P 点的 Z 坐标，

$$Z_P = K_1 \times \left(Z_{P_2} - Z_{P_1}\right) + Z_{P_1} \tag{3.8}$$

此后的计算均忽略各点的 Z 坐标，仅考虑投影平面上的情况，计算 P_1 点到原点的距离 d_1 和 P_2 点到原点的距离 d_2，如式(3.2)和(3.3)，根据高度比值 K_1 进行线性插值得到 P 点到原点的估计距离 d_{est}，如式(3.4)，再利用平面距离比值 K_2 反求 P 点到原点的校正距离 d_{adj}，

$$d_{adj} = K_2 \times d_{est} \tag{3.9}$$

最后，利用 X、Y 坐标比值 K_X、K_Y 反求 P 点的 X、Y 坐标，

$$X_P = K_X \times d_{adj} \tag{3.10}$$

$$Y_P = K_Y \times d_{adj} \tag{3.11}$$

对每一个中间截面线的每一个顶点进行上述计算求得之后，就构成了能够保持形状的插值中间截面线。

构建插值数据集和实际进行插值操作的整个流程见图 3-16 所示，两个过程存在着明显的对应关系，通过一些重要的比值数据进行联系。差值过程中所使用的各个量的几何关系如图 3-17 所示，其中蓝色实线表示关键截面线，黑色实线表示正在进行插值的中间截面线。

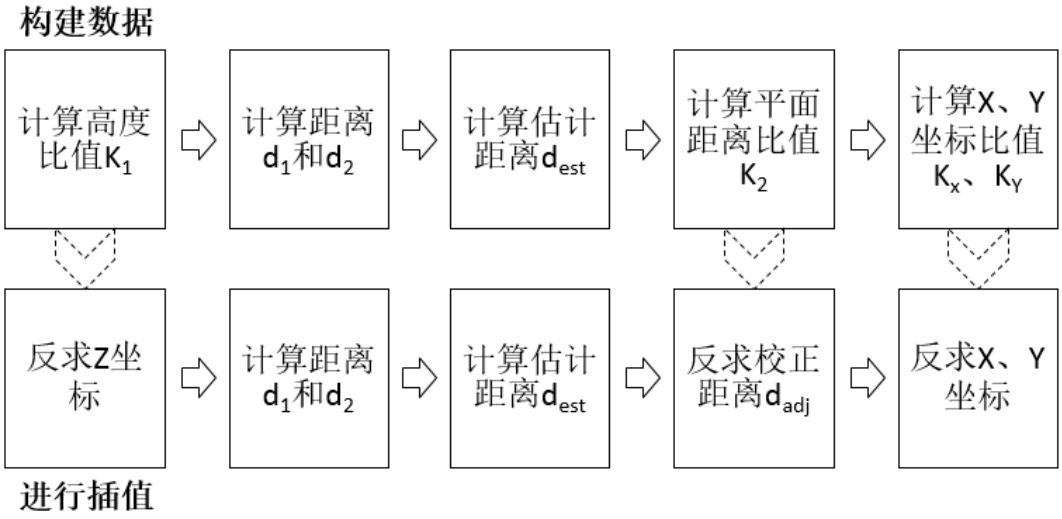

图 3-16 中间截面线的插值过程

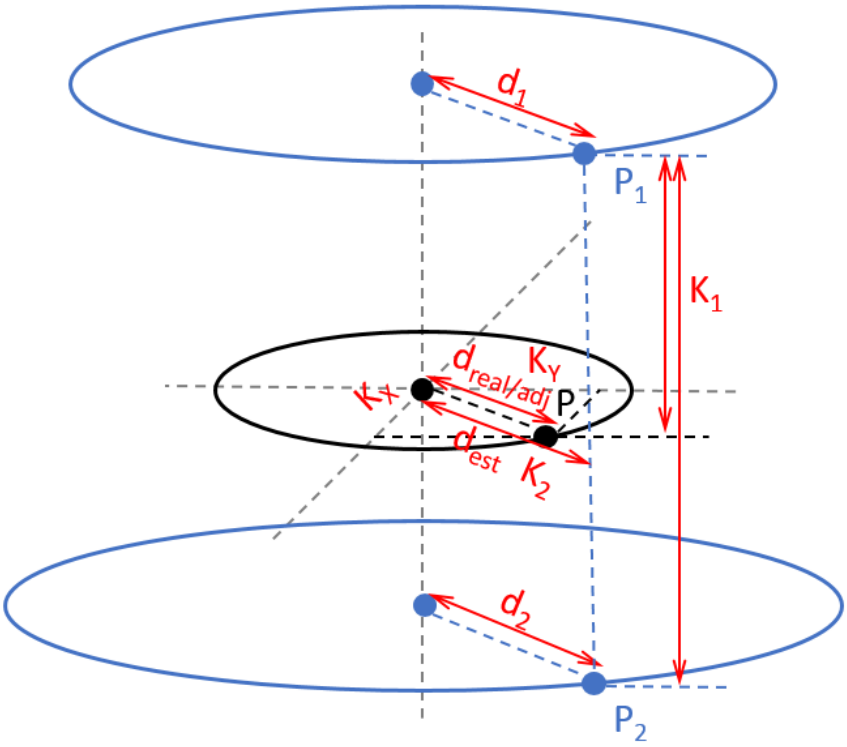

图 3-17 中间截面线插值的几何示意

对于颈顶线—颈线、腋下线—胸线、胸线—腰线、腰线—臀线、臀线—臀底线这几部分，由于各个截面线均为闭合的截面环，直接利用相邻的关键截面线进行插值曲线的生成即可。对于剩下的一部分，也就是颈线、肩线与腋下线之间的

部分，则有所不同，因为这一部分的中间截面线是开放的截面线：对于前面的部分，曲线插值时使用的上方截面线为左肩线—颈线的前半部分—右肩线组成的开放截面线，下方截面线则为腋下线的前半部分；后面的部分同理可得。

3.7.6 三角化形成表面

在得到关键截面线和中间截面线后，下一步是利用三角化形成表面。

由于相邻的截面线拥有相同的顶点数，而且顶点之间具有一定的对应关系，因而三角化比较简单。记相邻的两条截面线分别为截面线 A 和截面线 B，则具体流程如下：A 的第 i 个顶点、B 的第 i 个顶点和 A 的第 i+1 个顶点构成第 2*i 个三角面片；B 的第 i 个顶点、B 的第 i+1 个顶点和 A 的第 i+1 个顶点构成第 2*i+1 个三角面片；i 自增一位，继续进行。

由于要保持三维模型的左右对称性，不能使用同一种三角面片的构成顺序完成整个环的表面。可以以一种构成顺序先以正向完成半个环的表面，再以对称的顺序以逆向完成另外半个环的表面，如图 3-18 所示。

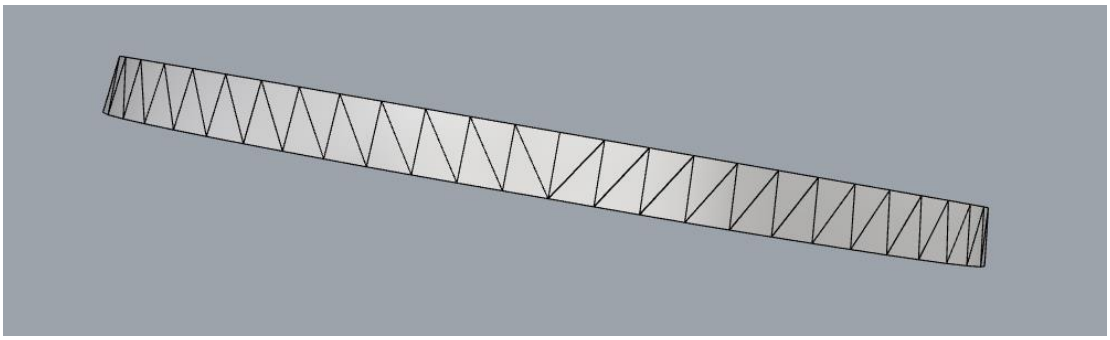

图 3-18 三角面片构成的左右对称表面

3.7.7 人体模型的生成流程

在进行初始化时，首先读入各个关键截面线的原始数据，以及预先处理好的中间截面线的插值数据集。之后，根据各个部分之间的约束关系，按照顺序进行人体模型的生成。各个关键截面线（不包括颈顶线、臀底线和腋下线）仅受关键尺寸参数的约束，通过变形可以立即得到。颈顶线和臀底线不会发生变化，也可以立即得到。在得到肩线和胸线之后，根据约束关系可以得到变形后的腋下线。

之后进行曲线插值操作，计算出全部的中间截面线。最后对每相邻的两条截面线进行三角化，形成表面。

在任意一个关键尺寸参数发生变化时，首先要对该参数直接和间接相关的关键截面线进行变形，再对发生了改变的关键截面线所在的部分进行重新曲线插值，最后再重新三角化。例如，改变了颈围这一参数，它直接与颈线相关，但同时与肩线和腋下线间接相关，因此要按顺序对这三条关键截面线进行变形；之后对颈顶线—颈线、颈/肩线—腋下线和腋下线—胸线三个部分进行重新曲线插值；最后再对这三个部分进行重新三角化。

3.8 可用性研究

合理的人体重要尺寸参数的选择，减轻了用户使用软件的额外负担。用户只需要知道一些购买服装时常见的尺寸参数就可以调整出符合自己体型的人体模型，而很少需要附加测量。

重新设计的人体模型分块方法，既配合了试衣机器人机械结构设计的需求，又能尽可能好得拟合人体外形。

本程序使用了应用非常广泛的 Java 语言编写，具有良好的可移植性，能够满足不同平台的用户的使用需求。简洁、直观的用户界面设计，确保了用户即便第一次使用本程序，也可以迅速上手。线框图和渲染图两种模型显示模式，一方面可以满足用户不同的观察需求，另一方面也照顾了计算机性能较差的用户。

通过优化的算法流程设计，使得每次尺寸参数变更时仅有受到其影响的部分进行重新运算，而不是直接重新生成模型，大大降低了运算负荷，使得三维人体模型变形效果的反馈具有实时性。

第4章 方案实现

4.1 试衣机器人外壳制作

根据上文，为配合试衣机器人的机械结构设计，试衣机器人的外壳最终定为13 个分块：颈部 1 块，肩—胸部 4 块，腰部 4 块，臀部 4 块。

同时，在外壳的加工工艺方面，因为外形较为复杂，原定使用 3D 打印制造。但因为外壳较大，所需要的加工时间比较长，最终放弃这一方案，而转为直接购买市面上的塑料外壳型的服装人台，再进行切割分块。所购买的服装人台为男性上半身人台，胸围为 94cm，腰围为 76cm，臀围为 94cm；PVC 材质，外壳厚度约 2.5mm，具有良好的切割性能。如图 4-1 所示。

图 4-1 塑料外壳型的男性上半身人台

在进行分块之前，首先需要对人台进行标记。根据一些人体测量的准则，首先用绿色胶带标记出胸线、腰线和臀线。之后用记号笔画出切割线。颈部直接以颈线为下边界进行划分；肩—胸部及腰部的划分边界线是胸线和腰线之间略偏上的一条线，它位于胸部的突起下方略微平坦处；腰部及臀部的划分边界线是腰

线和臀线之间略偏下的一条线，它位于臀部的突起上方略微平坦处；最后还要在前、后、左、右四个方位上竖直地划线，使肩—胸部、腰部和臀部均分为 4 块。此外，为防止切割分块后将各块混淆，应做好区分用的标记。人台的划线操作如图 4-2 所示。

图 4-2 人台的划线

人台的切割分块主要使用锯子和小刀配合进行。加工的过程如图 4-3 所示。

由于所使用的人台属于中等体型，为了试衣机器人能够变形到更小的体型，需要对每个分块进行进一步的处理，即磨除或削除每个分块的边界处的 1cm 的材料，使每个分块变得更小。

最后，将分块后的人台通过一定的连接方法固定到试衣机器人的机械结构上，即构成试衣机器人的外壳。安装的过程如图 4-4 所示。

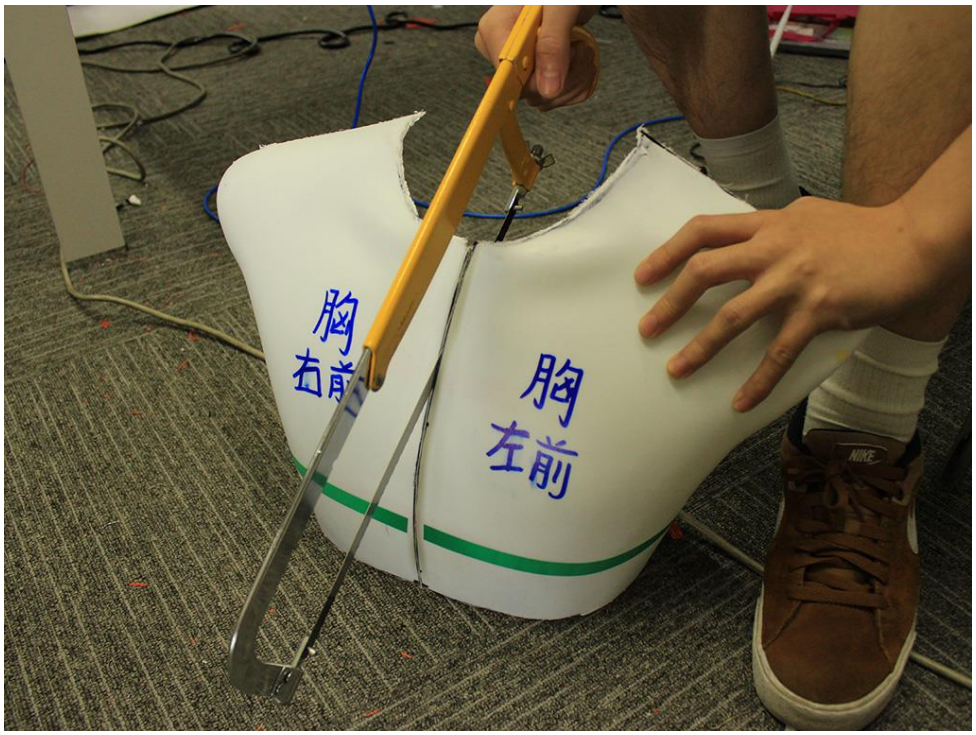

图 4-3 人台的切割

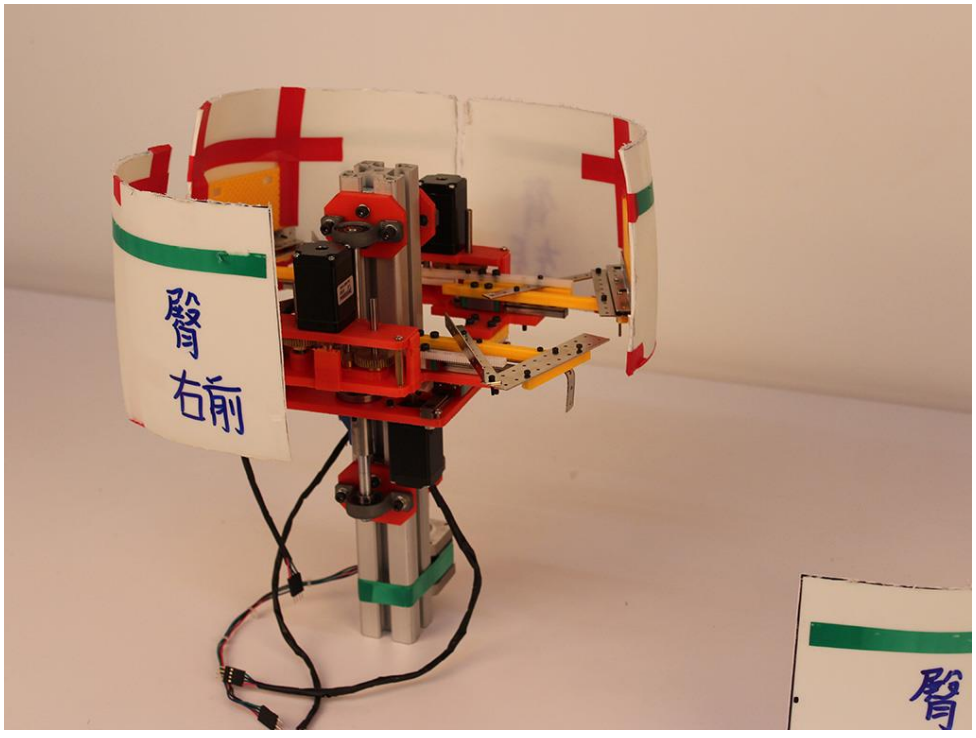

图 4-4 外壳的安装

4.2 三维人体模型生成程序

三维人体模型的生成程序严格按照面向对象程序设计的思路进行编写。程序中各类的关系如图 4-5 所示，其中实线箭头表示继承关系，虚线箭头表示包含关系，每类仅列出公开方法，右上方标有“s”记号的为静态方法。

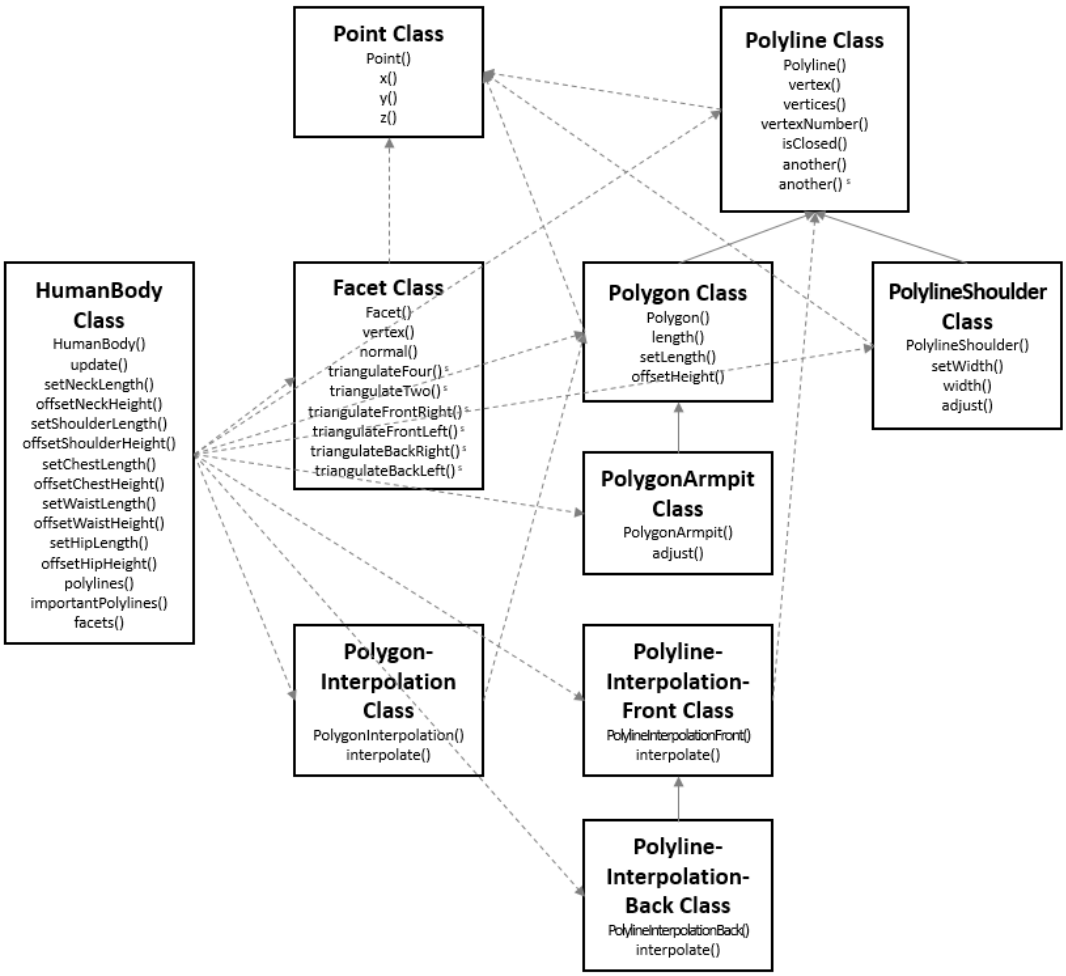

图 4-5 三维人体模型生成程序的类间结构

Point 类代表包含三维坐标 X、Y、Z 的点，每个坐标值均为单精度浮点型。此类为不可更改类。通过需要三个参数的构造器构造。通过三个访问器来访问三个字段 x、y、z。

Polyline 类代表一系列三维点按照一定顺序构成的多重直线，也就是前述的截面线，它可能是闭合的也可能是开放的。通过传入一个 Point 类型的数组来构造，可以指定是否闭合。可以通过访问器访问整个顶点数组，也可以单独访问某一个点。可以通过访问器访问多重直线包含的顶点数以及是否闭合。包括一个产生器方法，它能够产生与这个 Polyline 对象关于 YOZ 平面对称的另一个 Polyline 对象。还包括一个静态方法，用来产生输入数组中的每一个 Polyline 对象的关于 YOZ 平面的镜像，这是用来在已知人体模型某个部分的截面线右半部分（或左半部分）的情况下，求另一半部分时调用的，因为人体模型满足严格的左右对称。

PolylineShoulder 类继承于 Polyline 类，专门用来表示肩线，是一种特殊的多重直线。通过传入包含肩线各顶点坐标的原始数据文件来构造，事实上只记录右肩线的数据。可以通过访问器来访问肩宽，注意肩宽的定义为两条肩线及颈线后半部分的长度之和，这里计算了肩线的长度，但没有直接计算颈线后半部分的长度，而是使用一个系数乘以颈线最右点距中心的距离（即右肩线第一个点的 X 坐标），经实验得这个系数为 1.18。在程序中右肩线由两个参数唯一确定：一是起始点，即颈线的最右点；二是尺寸参数肩宽。因此两种情况会引发肩线的变化：一种是颈线的变化导致起始点的变化，需要调用调节方法进行调整；另一种是肩宽发生变化，需要调用设定肩宽方法进行设置。

Polygon 类继承于 Polyline 类，特指闭合的多重直线，也就是多边形或是前述的截面环。为了后续的运算，需要为多边形设定中心，这里其中心的 X、Y 坐标均为 0，Z 坐标为各顶点的 Z 坐标平均值。通过传入包含截面环各顶点坐标的原始数据文件来构造，也可以传入一个 Point 类型的数组来构造。可以通过访问器来访问周长。可以通过设定器来更改其周长，也就是前述的参数化变形，或是偏移其高度。

PolygonArmpit 类继承于 Polygon 类，专门用来表示腋下线，是一种特殊的多边形。其特殊之处在于，腋下线虽然是闭合的截面环，但它的形状取决于肩线和胸线的宽度（最左点到最右点的距离），而非取决于其周长。因此，在肩线或是胸线发生变化时，需要调用调节方法来调整腋下线。

PolygonInterpolation 类用来插值生成两个截面环之间的截面环集。它的构造需要传入两个关键截面环 Polygon 对象，以及它们之间的中间截面环的插值数据集文件。在初始化时，或是两个关键截面环中有任何一个发生变化时，应当调用插值方法进行重新插值生成中间截面环。

PolylineInterpolationFront 类专门用来插值生成颈/肩线与腋下线之间的正面部分的截面线集，实际上仅生成正面右侧的截面线集。它的构造需要传入颈线、右肩线和腋下线的 Polyline 对象，以及它们之间的中间截面线的插值数据集文件。同样在初始化时或发生变化时，需要调用插值方法重新插值。该类在插值时首先将颈线的右前侧的四分之一部分和右肩线合成为一个 Polyline 对象，再将腋下线的右前侧四分之一部分截取为一个 Polyline 对象，进而利用这两个对象进行插值。

PolylineInterpolationBack 类继承于 PolylineInterpolationFront 类，专门用来插值生成颈/肩线与腋下线之间的背面左侧部分的截面线集。它的使用与父类基本相同，但要注意传入的肩线为左肩线。

Facet 类代表三角面片，它具有三个顶点和法向量，均为 Point 类型。通过传入三个点的构造器进行构造，构造器会自动运算法向量。通过访问器来访问单个顶点或法向量。该类提供了多种静态方法，用来为不同部位的截面线进行三角化，传入参数均为多个截面线，返回值均为 Facet 类型的列表。在这些方法中，有两个用来处理截面环之间的三角化：其中一个三角面片更换两次方向，用于颈顶线—颈线部分的三角化；另一个更换四次方向，用于腋下线—胸线、胸线—腰线、腰线—臀线、臀线—臀底线部分的三角化。还有四个用来处理截面线之间的三角化，分别处理肩部的左前侧、右前侧、左后侧、右后侧。

HumanBody 类代表参数化的人体模型。它综合运用了上述各类，利用各成分之间相互约束的关系，按照一定的顺序进行初始化，得到一个三维人体模型。它的核心方法为更新方法，这个方法需要在对某些人体尺寸参数进行改变后及时调用，它会根据参数的变化对相关部分进行重新曲线插值和三角化，而对于无关部分则不做改变。它包括一系列的调整人体尺寸参数的设定器方法。该类提供多

种输出，可以以 Polyline 列表的形式输出人体模型中的所有截面线，也可以仅输出关键截面线，还可以输出 Facet 列表形式的所有三角面片。

4.3 三维人体模型显示程序

三维人体模型的显示程序主要通过引用的 Processing 类库来实现。它包含一个主程序类 Main 类，以及一个辅助的用于 STL 模型导出的 STLWriter 类。

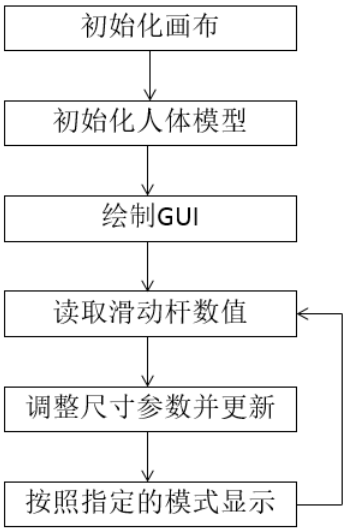

图 4-6 三维人体模型显示程序的工作流程

Main 类导入了 Processing 的 core 包用于构成程序的主体，也导入了 controlP5 包用于绘制 GUI。Main 类继承于 PApplet 类，从而以 Processing 程序的方式工作。核心方法包括 setup 方法和 draw 方法，前者仅在开始时运行一次，用于程序的初始化，后者则不断地重复运行。在 setup 方法中，需要初始化画布和 HumanBody 对象，并绘制出 GUI。GUI 由 ControlP5 对象生成；使用两个折叠面板 Accordion 对象分别包装功能区和调整区；每个 Accordion 包含若干个控件组 Group 对象；每个 Group 对象包含若干个滑动杆 Slider 对象或按钮 Button 对象。在 draw 方法中，首先读取调整区各个滑动杆的值，得到当前的各个尺寸参数；如果与记录的尺寸参数不同，则调用 HumanBody 对象的设定器进行更改，并调用更新方法调整模型；最后再根据设定的显示模式对模型进行显示。对于线框模

式，只需要设定为没有填充、描边为深灰色 1px，然后绘制全部三角面片即可；对于渲染模式，需要设定光源，并设定填充为深灰色、没有描边，再绘制三角面片。如果按下机器人变形按钮，会根据 API 生成控制链接，并打开链接进行联机控制。如果按下保存模型按钮，会调用 STLWriter 类的相关方法进行保存。主要工作流程如图 4-6 所示。

对于界面中有关尺寸参数调整的各个滑动杆，需要定最小值、最大值和默认值。本设计中参考国标男性上半身人台的尺寸进行设定，将相关尺寸参数的滑动杆的最小值定为最小的标准尺寸，最大值定为最大的标准尺寸，默认值则为 96 型号的标准尺寸。国标尺码表见表 4-1。

表 4-1 国标男上半身人台尺码表

国标男上半身人台尺码表： 单位：厘米

部位\型号	88	92	96	100	104	110
颈围	38	39	40.3	41.5	42.5	44
胸围	88	92	96	100	104	110
腰围	74	78	82	86	90	94
臀围	90	93	97	100	103	108
总肩宽	43.5	44.7	46	47.2	48.4	50

试衣机器人控制 API 由课题组中的其他成员负责，这里仅根据协定进行调用。在试衣机器人的实际制作中，胸部、腰部和臀部每个部位由 5 个步进电机进行控制，5 个步进电机由同一块 Arduino 控制板进行控制，每个 Arduino 控制板具有特定的 IP 地址，因此控制某个部位变形需要向 Arduino 控制板对应的 IP 地址发送指令驱动 5 个步进电机按照一定的方向旋转一定的步数。对试衣机器人的控制是基于物联网的，约定对于控制器的操作 URL 格式为：

http://[Arduino 访问 IP]/[指令类型]/[数据 1]/[数据 2]

控制主要包括两个部分，即特定部位的初始化和变形。对某个部位进行初始化的指令如下：

http://[Arduino 访问 IP]/initial/
[dir1][step1][dir2][step2]...[dir5][step5]/
[sign1][limit1][sign2][limit2]...[sign5][limit5]

其中，Arduino 访问 IP 表示控制该部位的 Arduino 控制板的 IP 地址；initial 表示初始化的指令类型；dir 和 step 代表电机驱动板的方向针脚和步进针脚与 Arduino 控制板的连接引脚，均为 2 位十进制数；sign 与 limit 代表电机初始化的方向和限位传感器输入引脚，前者为 1 位（0 或 1），后者为 2 位十进制数；前述数值位数不足则补前导零。对某个部位进行变形的指令如下：

http://[Arduino 访问 IP]/write/
[dir1][step1][dir2][step2]...[dir5][step5]/
[sign1][num1][sign2][num2]...[sign5][num5]

其中，Arduino 访问 IP 表示控制该部位的 Arduino 控制板的 IP 地址；write 表示变形的指令类型；dir 和 step 代表电机驱动板的方向针脚和步进针脚与 Arduino 控制板的连接引脚，均为 2 位十进制数；sign 与 num 代表电机运动的方向和步数，前者为 1 位（0 或 1），后者为 5 位十进制数；前述数值位数不足则补前导零。通过测量初始化时和最大行程时的试衣机器人尺寸，即可插值得到中间尺寸所对应的电机步数，进而可以控制机器人变形到指定的体型。至于在程序中对 URL 进行访问，存在两种解决方案：一是打开系统默认的浏览器访问 URL，需要使用 URI 和 Desktop 对象；二是直接后台利用 GET 方法访问 URL，需要使用 URL 和 HttpURLConnection 对象。

STLWriter 类仅包含一个静态方法，用于保存人体模型为 STL 文件。STL 文件仅保存模型的三角面片的顶点坐标和法向量数据，而不记录拓扑关系，非常符合本设计中人体模型的数据结构。需要按照 STL 文件格式的要求进行文件写入：前 80 个字节为文件头信息字节，可以全部填入空值占位；接下来 4 个字节为三角面片的个数；之后每 50 个字节记录一个三角面片，其中前 12 字节为表示法向量的 3 个浮点数，紧接着 36 字节为表示 3 个顶点的 9 个坐标的浮点数，最后 2 字节表示属性，同样可以填空值占位。值得注意的是，STL 文件中均使用小字节

序，即较低位的字节优先存储；而 Java 中默认使用大字节序。因此，不能直接使用 DataOutputStream 对象来写入文件，而要先使用一个 ByteBuffer 对象，设定其 ByteOrder 为 LITTLE_ENDIAN，将数据放进其中进行缓存，最后再将每个字节按顺序写入文件。

第5章 设计展示

5.1 设计结果展示

5.1.1 试衣机器人的人体模型分块

图 5-1 人体模型分块

图 5-2 试衣机器人外壳安装

5.1.2 参数化三维人体模型程序

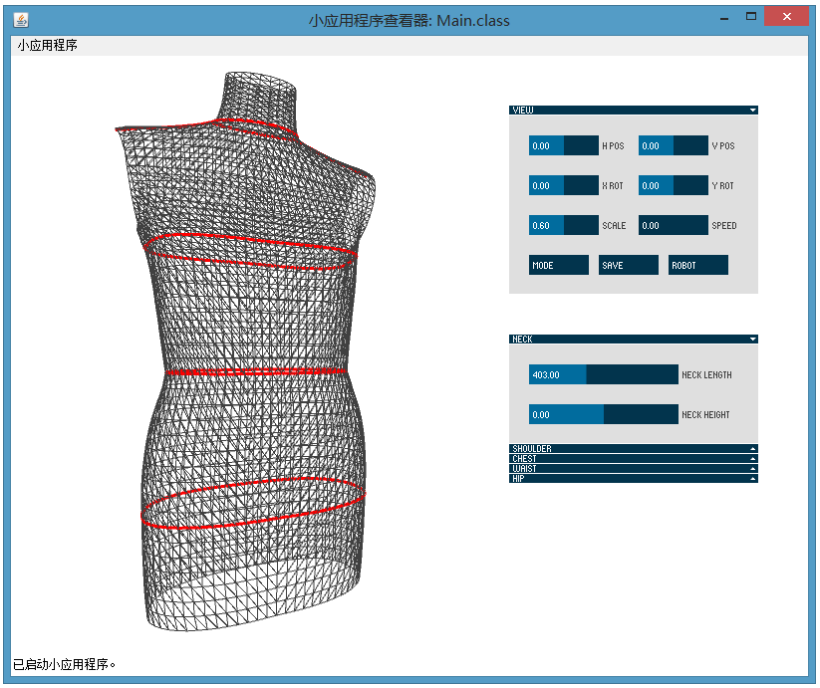

图 5-3 网格模式观察

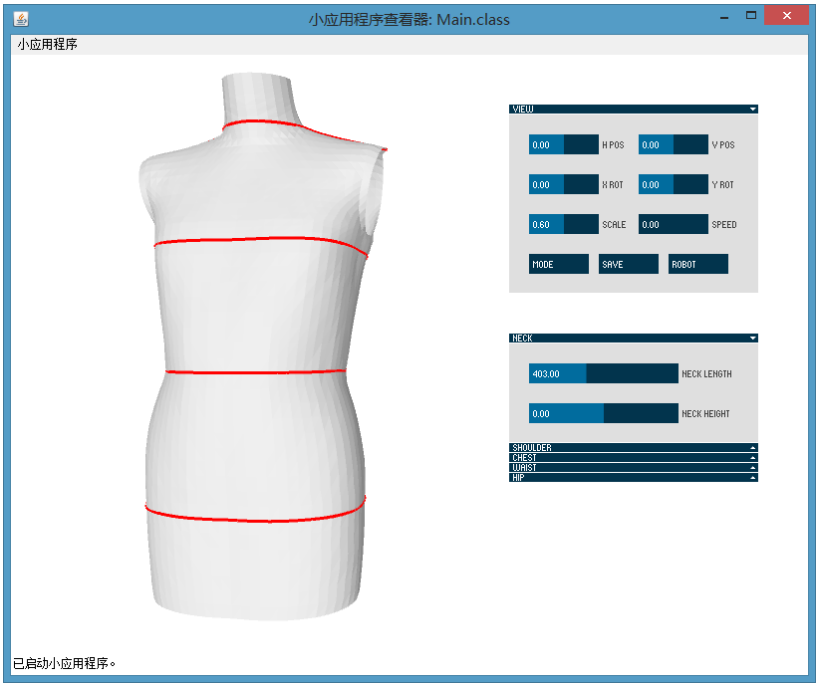

图 5-4 渲染模式观察

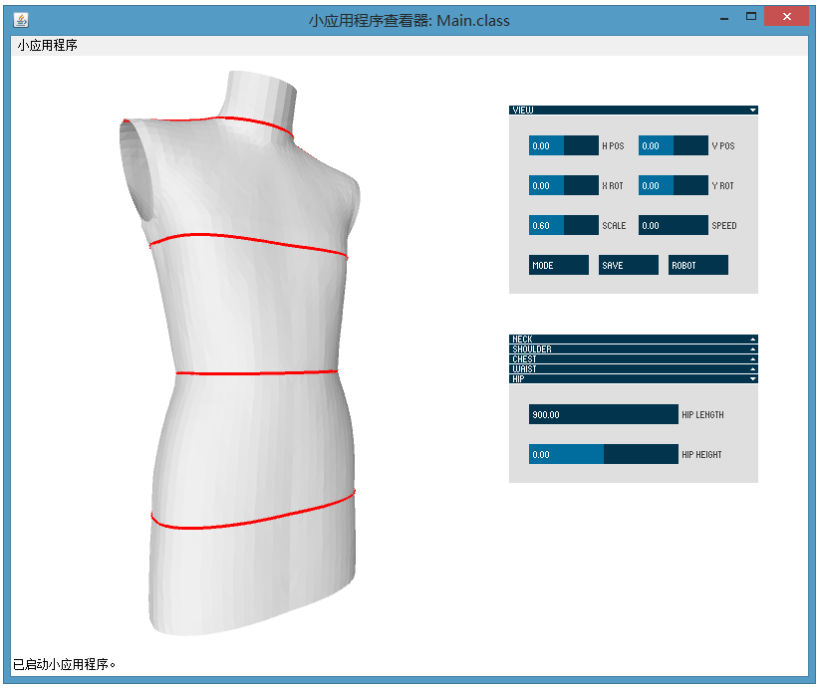

图 5-5 最瘦体型效果

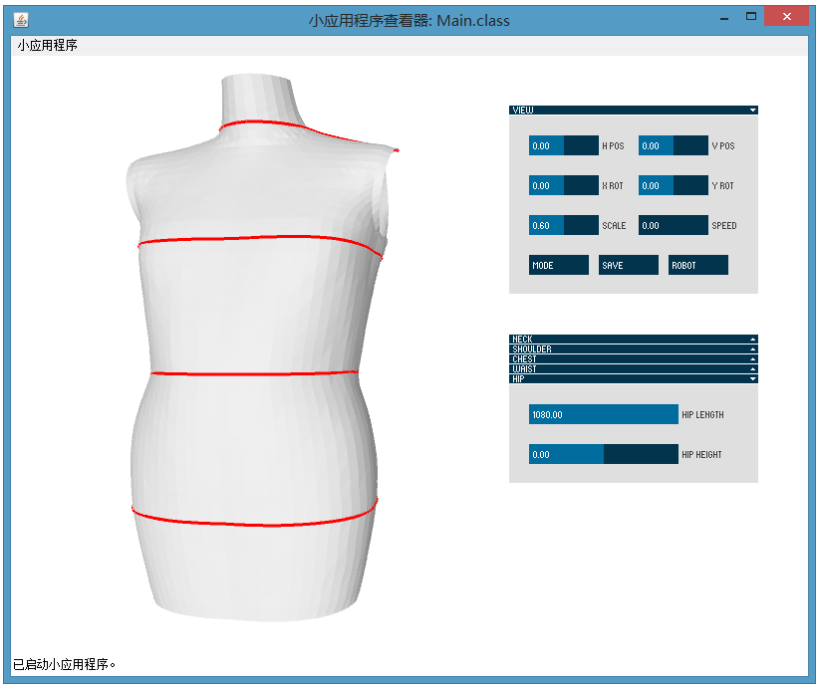

图 5-6 最胖体型效果

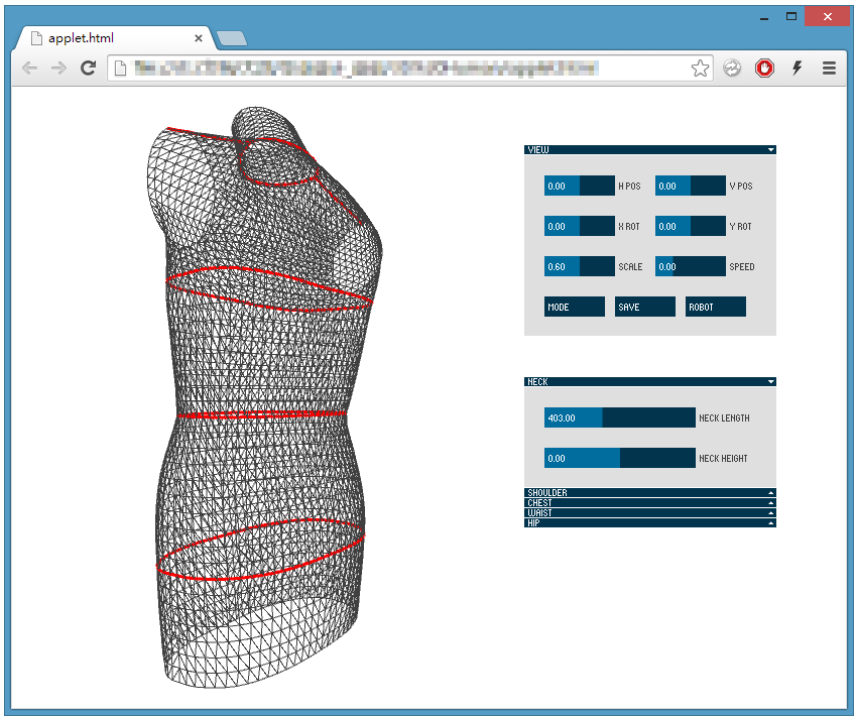

图 5-7 网页嵌入效果

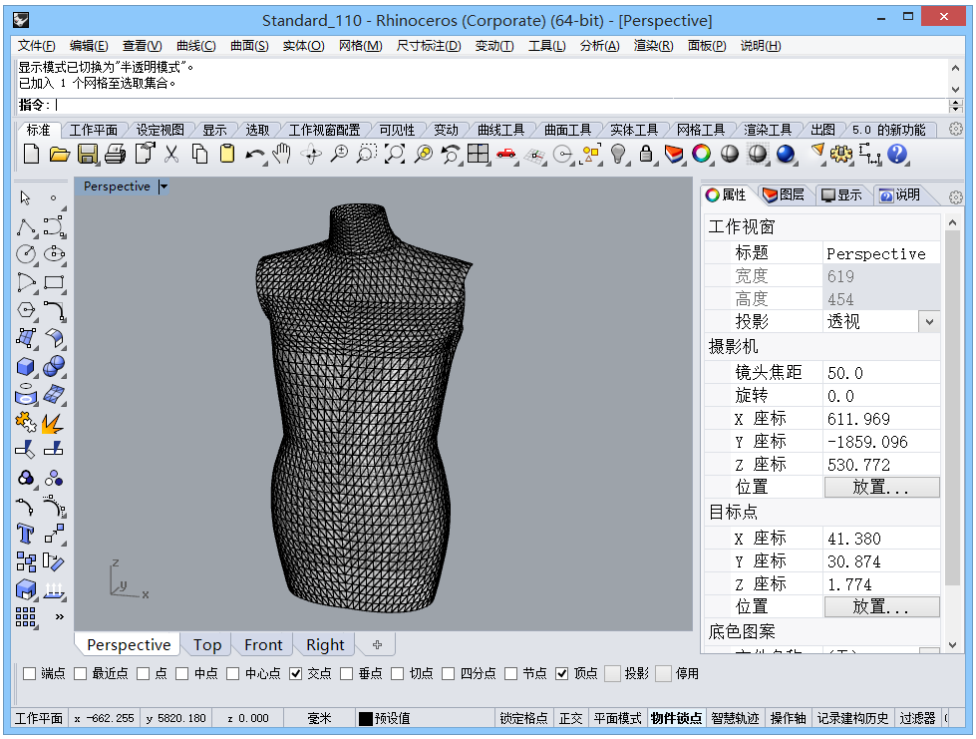

图 5-6 导出模型文件效果

图 5-7 控制机器人变形效果

5.2 展板设计

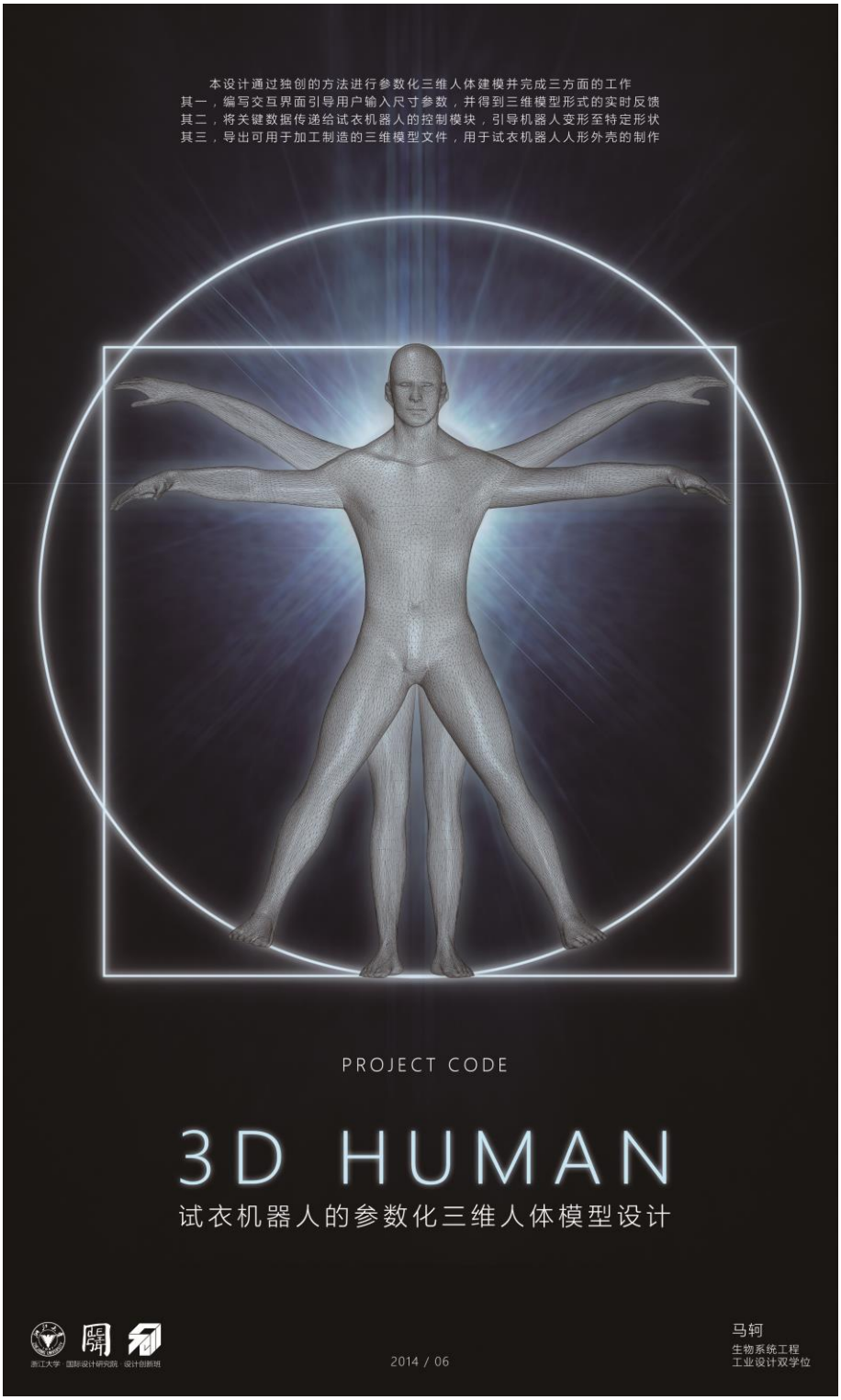

图 5-8 展板设计

第6章 设计总结

本设计作为“试衣机器人与虚拟试衣系统”课题的一部分，在通过一定的方法完成参数化的三维人体建模的基础上，主要承担三方面的工作：其一是编写交互界面引导用户输入尺寸参数，并得到三维人体模型形式的实时反馈；其二是能够将关键数据传递给试衣机器人的控制部分，引导机器人变形至特定形状；其三是能够导出可加工的三维模型文件。本设计基本完成了上述的设计任务。

本设计中的参数化三维人体建模，采用了与以往的方法有所不同的方法。其基本思路在于先由尺寸参数驱动基准数据变形生成关键线条，再由关键线条按照保持形状的插值方法生成其他线条，最后对各线条进行均匀的三角化从而生成表面。按照这种方法编写的模型生成程序能够产生满足要求的参数化三维人体模型。

当然，本设计仍然存在许多可以改进的地方。例如，当前的参数驱动的关键线条变形是以原点为中心的，但事实上人体的胖瘦变化并没有这么单纯，还需要深入的研究，利用复杂的公式进行拟合；当前的保持形状的曲线插值方法过于依赖原始数据，并且在相对于基准模型变形过大时会出现较为严重的失真。

在进行本设计的同时，笔者还参与了同一课题组中其他子项目的一些工作，包括机械设计、零件加工、程序编写等等，得到了全方面的锻炼。

参考文献

[1] 朱李丽, 邓中民, 李刚炎. 三维服装 CAD 中人体建模综述 [J]. 武汉科技学院学报, 2004, 01: 19-21.

[2] 秦可, 庄越挺, 吴飞. 服装 CAD 中三维人体模型的参数化研究 [J]. 计算机辅助设计与图形学学报, 2004, 07: 918-922.

[3] 闫迷军, 宁涛, 张兆璞. 服装 CAD 中三维人体尺寸信息提取技术的研究 [J]. 工程图学学报, 2005, 01: 26-29.

[4] 吴龙, 张欣, 任小玲, 李毅. 基于三维人体测量的参数化人台的研究 [J]. 西安工程科技学院学报, 2005, 04: 416-419.

[5] 王媚, 陆国栋, 张东亮. 服装 CAD 中三维人体建模技术的研究及应用 [J]. 工程图学学报, 2007, 01: 1-6.

[6] Wang C C L, Wang Y, Chang T K K, et al. Virtual human modeling from photographs for garment industry [J]. Computer-Aided Design, 2003, 35(6): 577-589.

[7] Kim S, Park C K. Parametric body model generation for garment drape simulation [J]. Fibers and Polymers, 2004, 5(1): 12-18.

[8] Wang C C L. Parameterization and parametric design of mannequins [J]. Computer-Aided Design, 2005, 37(1): 83-98.

[9] Baek S Y, Lee K. Parametric human body shape modeling framework for human-centered product design [J]. Computer-Aided Design, 2012, 44(1): 56-67.

[10] 王媚. 面向服装 CAD 的三维人体建模与变形技术研究及实现 [D]. 浙江大学, 2006.

[11] 崔树芹. 三维虚拟试衣系统中参数化人体建模技术的研究 [D]. 华中科技大学, 2006.

[12] 宋诗超. 基于 Kinect 的三维人体建模与测量的研究 [D]. 东华大学, 2013.

[13] Wikipedia. Virtual Dressing Room [EB/OL]. http://en.wikipedia.org/wiki/Virtual_dressing_room.

致谢

本论文是在导师张东亮教授的关怀和指导下完成的。从论文的选题、开题，到项目的准备、实施，再到结果的分析和论文的撰写，导师的关心与帮助都起到了不可磨灭的作用，在此谨向导师致以诚挚的感谢和崇高的敬意。导师正直的为人和严谨的治学态度，鼓舞和激励着我在学术道路上不断前行。

感谢浙江大学国际设计研究院的其他老师在三年中所传授的知识和给予的教导，使我的知识面得到了很大的拓展，也锻炼了我多方面的能力。特别要感谢黄敬华老师在学习和生活中给予的多方面帮助。

感谢浙江大学生物系统工程与食品科学学院生物系统工程系的老师们传授给我精巧的工程技术与严谨的工程思维。特别要感谢应义斌教授对于职业规划方面的咨询，和蒋焕煜教授在主修专业毕业设计中给予的指导和帮助。

感谢新加坡科技设计大学的陈陆捷副教授在交流期间给予的帮助和指导。陈老师通过科研训练带我初窥了计算机图形学的奥妙，并在留学申请方面给予了我很多帮助。

感谢在大学四年中陪伴我的同学们、朋友们以及同事们，是他们使我的大学生活变得多姿多态。尤其要感谢同一个课题组的周奥博、石磊、周家惠、王甜等同学，以及经常在实验室中共同学习的熊钊、李腾、魏凡丁、叶稚韵、刘子榆、甘茜珺等同学。还要感谢与我在同一个寝室共同生活、容忍我的缺点的室友李琛、张俊敏、马壮同学。

最后，还要感谢一直在精神和经济等方方面面支持我的家人，他们是我在学术的道路上越走越远的最大后盾。

马轲

二〇一四年五月于浙大紫金港

附录

本程序主要代码如下：

Point.java

```
public class Point {
    private final float x;
    private final float y;
    private final float z;

    public Point(float x, float y, float z) {
        this.x = x;
        this.y = y;
        this.z = z;
    }

    public float x() {
        return x;
    }
    public float y() {
        return y;
    }
    public float z() {
        return z;
    }
}
```

Polyline.java

```
public class Polyline {
    protected Point[] vertices = null;
    protected int n = 0;
    protected boolean closed = false;

    protected Polyline() {
    }

    public Polyline(Point[] vertices) {
        this.vertices = vertices;
        this.n = vertices.length;
    }

    public Polyline(Point[] vertices, boolean closed) {
        this.vertices = vertices;
        this.n = vertices.length;
        this.closed = closed;
    }

    public final Point[] vertices() {
```

```
        return vertices;
    }

    // 返回一个顶点
    public Point vertex(int index) {
        return vertices[index];
    }

    // 返回顶点数
    public int vertexNumber() {
        return n;
    }

    // 是否闭合
    public boolean isClosed() {
        return closed;
    }

    // 返回左肩线
    public Polyline another() {
        Point[] newVertices = new Point[n];
        for (int i = 0; i < n; i++) {
            newVertices[i] = new Point(-vertices[i].x(),
vertices[i].y(),
                    vertices[i].z());
        }
        return new Polyline(newVertices);
    }

    public static Polyline[] another(Polyline[] plys) {
        Polyline[] newPlys = new Polyline[plys.length];
        for (int i = 0; i < plys.length; i++)
            newPlys[i] = plys[i].another();
        return newPlys;
    }
}
```

PolylineShoulder.java

```
public class PolylineShoulder extends Polyline {
    private Point startVertex;
    private float width;

    // 假设为右肩
    public PolylineShoulder(String fileName) {
        In in = new In(fileName);
        n = Integer.parseInt(in.readLine());
        vertices = new Point[n];
        closed = false;
        for (int i = 0; i < n; i++) {
            String line = in.readLine();
            String[] fields = line.split(" ");
```

```
            float x = Float.parseFloat(fields[0]);
            float y = Float.parseFloat(fields[1]);
            float z = Float.parseFloat(fields[2]);
            vertices[i] = new Point(x, y, z);
        }
        in.close();
        startVertex = vertices[0];
        width = width(vertices);
    }

    private static Point[] polylineStretch(Point[] source, Point
startVertex,
            float k) {
        int n = source.length;
        Point[] newVertices = new Point[n];
        newVertices[0] = startVertex;
        for (int i = 1; i < n; i++) {
            float x = (source[i].x() - source[i - 1].x()) * k
                    + newVertices[i - 1].x();
            float y = source[i].y() - source[i - 1].y()
                    + newVertices[i - 1].y();
            float z = source[i].z() - source[i - 1].z()
                    + newVertices[i - 1].z();
            newVertices[i] = new Point(x, y, z);
        }
        return newVertices;
    }

    // 设定肩线起始点和肩宽，假设为右肩
    private void setStartVertextAndWidth(Point newStartVertex,
float distWidth) {
        Point[] newVertices = polylineStretch(vertices,
newStartVertex, 1f);
        float currentWidth = width(newVertices);
        do {
            newVertices = polylineStretch(newVertices,
newStartVertex,
                    distWidth / currentWidth);
            currentWidth = width(newVertices);
        } while (Math.abs(currentWidth - distWidth) > 0.01);
        vertices = newVertices;
        startVertex = newStartVertex;
        width = distWidth;
    }

    // 根据颈围线调整肩线
    public void adjust(Polygon neck) {
        setStartVertextAndWidth(neck.vertex(7), width);
    }

    // 设定肩宽
    public void setWidth(float newWidth) {
```

```
        setStartVertextAndWidth(startVertex, newWidth);
    }

    // 计算两点距离
    private static float dist(Point pt1, Point pt2) {
        return (float) Math.sqrt(Math.pow(pt1.x() - pt2.x(), 2)
                + Math.pow(pt1.y() - pt2.y(), 2)
                + Math.pow(pt1.z() - pt2.z(), 2));
    }

    // 计算长度
    private static float length(Point[] source) {
        float sum = 0;
        for (int i = 1; i < source.length; i++) {
            sum += dist(source[i - 1], source[i]);
        }
        return sum;
    }

    // 计算肩宽，假设右肩
    private static float width(Point[] vertices) {
        return 2 * (length(vertices) + 1.18f * vertices[0].x());
    }

    // 返回肩宽
    public float width() {
        return width;
    }
}
```

Polygon.java

```
public class Polygon extends Polyline {
    protected Point origin;

    public Polygon(Point[] vertices) {
        this.vertices = vertices;
        this.n = vertices.length;
        this.closed = true;
    }

    public Polygon(String fileName) {
        In in = new In(fileName);
        n = Integer.parseInt(in.readLine());
        vertices = new Point[n];
        closed = true;
        float zsum = 0;
        for (int i = 0; i < n; i++) {
            String line = in.readLine();
            String[] fields = line.split(" ");
            float x = Float.parseFloat(fields[0]);
            float y = Float.parseFloat(fields[1]);
```

```
            float z = Float.parseFloat(fields[2]);
            vertices[i] = new Point(x, y, z);
            zsum += z;
        }
        in.close();
        origin = new Point(0, 0, zsum / n);
    }

    // 计算两点间的距离
    protected static float dist(Point pt1, Point pt2) {
        return (float) Math.sqrt(Math.pow(pt1.x() - pt2.x(), 2)
                + Math.pow(pt1.y() - pt2.y(), 2)
                + Math.pow(pt1.z() - pt2.z(), 2));
    }

    // 计算多边形周长
    protected static float length(Point[] source) {
        float sum = 0;
        for (int i = 1; i < source.length; i++) {
            sum += dist(source[i - 1], source[i]);
        }
        sum += dist(source[source.length - 1], source[0]);
        return sum;
    }

    // 计算多边形周长
    public float length() {
        return length(vertices);
    }

    // 将某点以另一点为原点按某一比例偏移
    protected static Point pointOffset(Point source, Point origin,
float k) {
        float x = (source.x() - origin.x()) * k + origin.x();
        float y = (source.y() - origin.y()) * k + origin.y();
        float z = (source.z() - origin.z()) * k + origin.z();
        return new Point(x, y, z);
    }

    // 将多边形以某点为原点按某一比例偏移
    protected static Point[] polygonOffset(Point[] source, Point
origin, float k) {
        Point[] result = new Point[source.length];
        for (int i = 0; i < source.length; i++) {
            result[i] = pointOffset(source[i], origin, k);
        }
        return result;
    }

    // 改变多边形周长
    public void setLength(float distLength) {
```

```
        Point[] newVertices = vertices;
        float currentLength = length(newVertices);
        do {
            newVertices = polygonOffset(newVertices, origin,
distLength
                    / currentLength);
            currentLength = length(newVertices);
        } while (Math.abs(currentLength - distLength) > 0.01);
        vertices = newVertices;
    }

    // 改变多边形高度
    public void offsetHeight(float deltaHeight) {
        for (int i = 0; i < n; i++) {
            vertices[i] = new Point(vertices[i].x(),
vertices[i].y(),
                    vertices[i].z() - deltaHeight);
        }
    }
}
```

PolygonArmpit.java

```
public class PolygonArmpit extends Polygon {
    public PolygonArmpit(String fileName) {
        super(fileName);
    }

    // 计算多边形半宽
    private static float halfWidth(Point[] source) {
        return source[15].x();
    }

    // 改变多边形半宽
    private void setHalfWidth(float distHalfWidth) {
        Point[] newVertices = vertices;
        float currentHalfWidth = halfWidth(newVertices);
        do {
            newVertices = polygonOffset(newVertices, origin,
distHalfWidth
                    / currentHalfWidth);
            currentHalfWidth = halfWidth(newVertices);
        } while (Math.abs(currentHalfWidth - distHalfWidth) >
0.01);
        vertices = newVertices;
    }

    // 根据颈围线调整肩线
    public void adjust(Polyline shoulder, Polygon chest) {
        float k = (this.vertex(15).z() - chest.vertex(15).z())
                / (shoulder.vertex(9).z() - chest.vertex(15).z());
        float halfWidth = k * shoulder.vertex(9).x() + (1 - k)
```

```
                * chest.vertex(15).x();
        setHalfWidth(halfWidth);
    }
}
```

PolylineInterpolationFront.java

```
public class PolylineInterpolationFront {
    protected Polyline neck;
    protected Polyline shoulder;
    protected Polyline armpit;
    protected float[][] k1s;
    protected float[][] k2s;
    protected float[][] kxs;
    protected float[][] kys;
    protected int n;

    protected PolylineInterpolationFront() {
    }

    public PolylineInterpolationFront(Polyline neck, Polyline
shoulder,
            Polyline armpit, String fileName) {
        this.neck = neck;
        this.shoulder = shoulder;
        this.armpit = armpit;
        // 读取总多边形数
        In in = new In(fileName);
        n = Integer.parseInt(in.readLine());
        k1s = new float[n][];
        k2s = new float[n][];
        kxs = new float[n][];
        kys = new float[n][];
        // 分别处理每个多边形
        for (int i = 0; i < n; i++) {
            // 读取多边形顶点数
            int vn = Integer.parseInt(in.readLine());
            k1s[i] = new float[vn];
            k2s[i] = new float[vn];
            kxs[i] = new float[vn];
            kys[i] = new float[vn];
            // 分别处理每个顶点
            for (int j = 0; j < vn; j++) {
                // 读取z方向上的比例k1和xy平面上的比例k2
                String line = in.readLine();
                String[] fields = line.split(" ");
                k1s[i][j] = Float.parseFloat(fields[0]);
                k2s[i][j] = Float.parseFloat(fields[1]);
                // 读取xy平面上x和y的比例系数
                kxs[i][j] = Float.parseFloat(fields[2]);
                kys[i][j] = Float.parseFloat(fields[3]);
```

```
            }
        }
        in.close();
    }

    public Polyline[] interpolate() {
        Point origin = new Point(0, 0, 0);

        Point[] p1 = new Point[17];
        System.arraycopy(armpit.vertices(), 0, p1, 0, 17);
        Point[] p2 = new Point[17];
        System.arraycopy(neck.vertices(), 0, p2, 0, 7);
        System.arraycopy(shoulder.vertices(), 0, p2, 7, 10);

        Polyline[] plys = new Polyline[n];
        // 分别处理每个多边形
        for (int i = 0; i < n; i++) {
            int vn = k2s[i].length;
            Point[] pts = new Point[vn];
            // 分别处理每个顶点
            for (int j = 0; j < vn; j++) {
                // 读取z方向上的比例k1并计算z坐标
                float z = (p2[j].z() - p1[j].z()) * k1s[i][j] +
p1[j].z();
                // 计算xy平面上的投影距原点的距离
                float dist1 = distXY(p1[j], origin);
                float dist2 = distXY(p2[j], origin);
                float dist = k2s[i][j]
                        * (k1s[i][j] * dist2 + (1 - k1s[i][j]) *
dist1);
                // 计算x和y坐标
                float x = kxs[i][j] * dist;
                float y = kys[i][j] * dist;
                // 存入点数组
                pts[j] = new Point(x, y, z);
            }
            // 存入多边形数组
            plys[i] = new Polyline(pts);
        }

        return plys;
    }

    // 计算两点在xy平面上投影间的距离
    protected static float distXY(Point pt1, Point pt2) {
        return (float) Math.sqrt(Math.pow(pt1.x() - pt2.x(), 2)
                + Math.pow(pt1.y() - pt2.y(), 2));
    }
}
```

PolylineInterpolationBack.java

```
public class PolylineInterpolationBack extends
PolylineInterpolationFront {
    public PolylineInterpolationBack(Polyline neck, Polyline
shoulder,
            Polyline armpit, String fileName) {
        this.neck = neck;
        this.shoulder = shoulder;
        this.armpit = armpit;
        // 读取总多边形数
        In in = new In(fileName);
        n = Integer.parseInt(in.readLine());
        k1s = new float[n][];
        k2s = new float[n][];
        kxs = new float[n][];
        kys = new float[n][];
        // 分别处理每个多边形
        for (int i = 0; i < n; i++) {
            // 读取多边形顶点数
            int vn = Integer.parseInt(in.readLine());
            k1s[i] = new float[vn];
            k2s[i] = new float[vn];
            kxs[i] = new float[vn];
            kys[i] = new float[vn];
            // 分别处理每个顶点
            for (int j = 0; j < vn; j++) {
                // 读取Z方向上的比例k1和XY平面上的比例k2
                String line = in.readLine();
                String[] fields = line.split(" ");
                k1s[i][j] = Float.parseFloat(fields[0]);
                k2s[i][j] = Float.parseFloat(fields[1]);
                // 读取XY平面上x和y的比例系数
                kxs[i][j] = Float.parseFloat(fields[2]);
                kys[i][j] = Float.parseFloat(fields[3]);
            }
        }
        in.close();
    }

    public Polyline[] interpolate() {
        Point origin = new Point(0, 0, 0);

        Point[] p1 = new Point[15];
        System.arraycopy(armpit.vertices(), 32, p1, 0, 15);
        Point[] p2 = new Point[15];
        System.arraycopy(neck.vertices(), 12, p2, 0, 5);
        System.arraycopy(shoulder.vertices(), 0, p2, 5, 10);

        Polyline[] plys = new Polyline[n];
        // 分别处理每个多边形
```

```
        for (int i = 0; i < n; i++) {
            int vn = k2s[i].length;
            Point[] pts = new Point[vn];
            // 分别处理每个顶点
            for (int j = 0; j < vn; j++) {
                // 读取Z方向上的比例k1并计算z坐标
                float z = (p2[j].z() - p1[j].z()) * k1s[i][j] +
p1[j].z();
                // 计算XY平面上的投影距原点的距离
                float dist1 = distXY(p1[j], origin);
                float dist2 = distXY(p2[j], origin);
                float dist = k2s[i][j]
                        * (k1s[i][j] * dist2 + (1 - k1s[i][j]) *
dist1);
                // 计算x和y坐标
                float x = kxs[i][j] * dist;
                float y = kys[i][j] * dist;
                // 存入点数组
                pts[j] = new Point(x, y, z);
            }
            // 存入多边形数组
            plys[i] = new Polyline(pts);
        }

        return plys;
    }
}
```

PolygonInterpolation.java

```
public class PolygonInterpolation {
    private Polygon ply1;
    private Polygon ply2;
    private float[][] k1s;
    private float[][] k2s;
    private float[][] kxs;
    private float[][] kys;
    private int n;

    public PolygonInterpolation(Polygon ply1, Polygon ply2, String
ply2_ply1) {
        this.ply1 = ply1;
        this.ply2 = ply2;
        // 读取总多边形数
        In in = new In(ply2_ply1);
        n = Integer.parseInt(in.readLine());
        k1s = new float[n][];
        k2s = new float[n][];
        kxs = new float[n][];
        kys = new float[n][];
        // 分别处理每个多边形
```

```
        for (int i = 0; i < n; i++) {
            // 读取多边形顶点数
            int vn = Integer.parseInt(in.readLine());
            k1s[i] = new float[vn];
            k2s[i] = new float[vn];
            kxs[i] = new float[vn];
            kys[i] = new float[vn];
            // 分别处理每个顶点
            for (int j = 0; j < vn; j++) {
                // 读取Z方向上的比例k1和XY平面上的比例k2
                String line = in.readLine();
                String[] fields = line.split(" ");
                k1s[i][j] = Float.parseFloat(fields[0]);
                k2s[i][j] = Float.parseFloat(fields[1]);
                // 读取XY平面上x和y的比例系数
                kxs[i][j] = Float.parseFloat(fields[2]);
                kys[i][j] = Float.parseFloat(fields[3]);
            }
        }
        in.close();
    }

    public Polygon[] interpolate() {
        Point origin = new Point(0, 0, 0);
        Polygon[] plys = new Polygon[n];
        // 分别处理每个多边形
        for (int i = 0; i < n; i++) {
            int vn = k2s[i].length;
            Point[] pts = new Point[vn];
            // 分别处理每个顶点
            for (int j = 0; j < vn; j++) {
                // 读取Z方向上的比例k1并计算z坐标
                float z = (ply2.vertex(j).z() - ply1.vertex(j).z())
* k1s[i][j]
                        + ply1.vertex(j).z();
                // 计算XY平面上的投影距原点的距离
                float dist1 = distXY(ply1.vertex(j), origin);
                float dist2 = distXY(ply2.vertex(j), origin);
                float dist = k2s[i][j]
                        * (k1s[i][j] * dist2 + (1 - k1s[i][j]) *
dist1);
                // 计算x和y坐标
                float x = kxs[i][j] * dist;
                float y = kys[i][j] * dist;
                // 存入点数组
                pts[j] = new Point(x, y, z);
            }
            // 存入多边形数组
            plys[i] = new Polygon(pts);
        }
```

```
        return plys;
    }

    // 计算两点在XY平面上投影间的距离
    private static float distXY(Point pt1, Point pt2) {
        return (float) Math.sqrt(Math.pow(pt1.x() - pt2.x(), 2)
                + Math.pow(pt1.y() - pt2.y(), 2));
    }
}
```

Facet.java

```
import java.util.ArrayList;
import java.util.List;

public class Facet {
    private Point[] vertices;
    private Point normal;

    public Facet(Point a, Point b, Point c) {
        vertices = new Point[3];
        vertices[0] = a;
        vertices[1] = b;
        vertices[2] = c;

        float px = b.x() - a.x();
        float py = b.y() - a.y();
        float pz = b.z() - a.z();
        float qx = c.x() - a.x();
        float qy = c.y() - a.y();
        float qz = c.z() - a.z();

        float nx = py * qz - pz * qy;
        float ny = pz * qx - px * qz;
        float nz = px * qy - py * qx;

        float nmod = (float) Math.sqrt(nx * nx + ny * ny + nz *
nz);
        normal = new Point(nx / nmod, ny / nmod, nz / nmod);
    }

    public Point vertex(int index) {
        return vertices[index];
    }

    public Point normal() {
        return normal;
    }

    private static List<Facet> triangulateFour(Polygon ply1,
Polygon ply2) {
```

```
        int n = ply1.vertexNumber();
        ArrayList<Facet> tris = new ArrayList<Facet>(n * 2);
        for (int i = 0; i < n / 4; i++) {
            tris.add(new Facet(ply1.vertex(i), ply1.vertex(i + 1),
ply2
                    .vertex(i)));
            tris.add(new Facet(ply2.vertex(i), ply1.vertex(i + 1),
ply2
                    .vertex(i + 1)));
        }
        for (int i = n / 4; i < n / 2; i++) {
            tris.add(new Facet(ply1.vertex(i), ply1.vertex(i + 1),
ply2
                    .vertex(i + 1)));
            tris.add(new Facet(ply2.vertex(i), ply1.vertex(i), ply2
                    .vertex(i + 1)));
        }
        for (int i = n / 2; i < n * 3 / 4; i++) {
            tris.add(new Facet(ply1.vertex(i), ply1.vertex(i + 1),
ply2
                    .vertex(i)));
            tris.add(new Facet(ply2.vertex(i), ply1.vertex(i + 1),
ply2
                    .vertex(i + 1)));
        }
        for (int i = n * 3 / 4; i < n - 1; i++) {
            tris.add(new Facet(ply1.vertex(i), ply1.vertex(i + 1),
ply2
                    .vertex(i + 1)));
            tris.add(new Facet(ply2.vertex(i), ply1.vertex(i), ply2
                    .vertex(i + 1)));
        }
        tris.add(new Facet(ply1.vertex(n - 1), ply1.vertex(0),
ply2.vertex(0)));
        tris.add(new Facet(ply2.vertex(n - 1), ply1.vertex(n - 1),
ply2
                .vertex(0)));
        return tris;
    }

    public static List<Facet> triangulateFour(Polygon ply1,
            Polygon[] ply1_ply2, Polygon ply2) {
        int n = ply1.vertexNumber() * 2 * (ply1_ply2.length + 1);
        List<Facet> tris = new ArrayList<Facet>(n);
        tris.addAll(triangulateFour(ply1, ply1_ply2[0]));
        for (int i = 0; i < ply1_ply2.length - 1; i++)
            tris.addAll(triangulateFour(ply1_ply2[i], ply1_ply2[i +
1]));
        tris.addAll(triangulateFour(ply1_ply2[ply1_ply2.length -
1], ply2));
        return tris;
    }
```

```
    private static List<Facet> triangulateTwo(Polygon ply1,
Polygon ply2) {
        int n = ply1.vertexNumber();
        ArrayList<Facet> tris = new ArrayList<Facet>(n * 2);
        for (int i = 0; i < n / 2; i++) {
            tris.add(new Facet(ply1.vertex(i), ply1.vertex(i + 1),
ply2
                    .vertex(i)));
            tris.add(new Facet(ply2.vertex(i), ply1.vertex(i + 1),
ply2
                    .vertex(i + 1)));
        }
        for (int i = n / 2; i < n - 1; i++) {
            tris.add(new Facet(ply1.vertex(i), ply1.vertex(i + 1),
ply2
                    .vertex(i + 1)));
            tris.add(new Facet(ply2.vertex(i), ply1.vertex(i), ply2
                    .vertex(i + 1)));
        }
        tris.add(new Facet(ply1.vertex(n - 1), ply1.vertex(0),
ply2.vertex(0)));
        tris.add(new Facet(ply2.vertex(n - 1), ply1.vertex(n - 1),
ply2
                .vertex(0)));
        return tris;
    }

    public static List<Facet> triangulateTwo(Polygon ply1,
Polygon[] ply1_ply2,
            Polygon ply2) {
        int n = ply1.vertexNumber() * 2 * (ply1_ply2.length + 1);
        List<Facet> tris = new ArrayList<Facet>(n);
        tris.addAll(triangulateTwo(ply1, ply1_ply2[0]));
        for (int i = 0; i < ply1_ply2.length - 1; i++)
            tris.addAll(triangulateTwo(ply1_ply2[i], ply1_ply2[i +
1]));
        tris.addAll(triangulateTwo(ply1_ply2[ply1_ply2.length - 1],
ply2));
        return tris;
    }

    private static List<Facet> triangulateRightUpper(Polyline
ply1,
            Polyline ply2) {
        int n = ply1.vertexNumber();
        ArrayList<Facet> tris = new ArrayList<Facet>((n - 1) * 2);
        for (int i = 0; i < n - 1; i++) {
            tris.add(new Facet(ply1.vertex(i), ply1.vertex(i + 1),
ply2
                    .vertex(i + 1)));
            tris.add(new Facet(ply2.vertex(i), ply1.vertex(i), ply2
```

```
                .vertex(i + 1)));
        }
        return tris;
    }

    private static List<Facet> triangulateRightLower(Polyline
ply1,
            Polyline ply2) {
        int n = ply1.vertexNumber();
        ArrayList<Facet> tris = new ArrayList<Facet>((n - 1) * 2);
        for (int i = 0; i < n - 1; i++) {
            tris.add(new Facet(ply1.vertex(i), ply1.vertex(i + 1),
ply2
                    .vertex(i)));
            tris.add(new Facet(ply2.vertex(i), ply1.vertex(i + 1),
ply2
                    .vertex(i + 1)));
        }
        return tris;
    }

    public static List<Facet> triangulateFrontRight(Polygon neck,
            Polyline shoulder, Polyline[] ply1_ply2, Polygon
armpit) {
        Point[] pts1 = new Point[17];
        System.arraycopy(neck.vertices(), 0, pts1, 0, 7);
        System.arraycopy(shoulder.vertices(), 0, pts1, 7, 10);
        Polyline ply1 = new Polyline(pts1);

        Point[] pts2 = new Point[17];
        System.arraycopy(armpit.vertices(), 0, pts2, 0, 17);
        Polyline ply2 = new Polyline(pts2);

        int n = (ply1.vertexNumber() - 1) * 2 * (ply1_ply2.length +
1);
        List<Facet> tris = new ArrayList<Facet>(n);
        tris.addAll(triangulateRightUpper(ply1, ply1_ply2[0]));
        for (int i = 0; i < ply1_ply2.length / 2; i++)
            tris.addAll(triangulateRightUpper(ply1_ply2[i],
ply1_ply2[i + 1]));
        for (int i = ply1_ply2.length / 2; i < ply1_ply2.length -
1; i++)
            tris.addAll(triangulateRightLower(ply1_ply2[i],
ply1_ply2[i + 1]));

    tris.addAll(triangulateRightLower(ply1_ply2[ply1_ply2.length -
1], ply2));
        return tris;
    }

    private static List<Facet> triangulateLeftUpper(Polyline ply1,
Polyline ply2) {
```

```
        int n = ply1.vertexNumber();
        ArrayList<Facet> tris = new ArrayList<Facet>((n - 1) * 2);
        for (int i = 0; i < n - 1; i++) {
            tris.add(new Facet(ply1.vertex(i + 1), ply1.vertex(i),
ply2
                    .vertex(i + 1)));
            tris.add(new Facet(ply2.vertex(i + 1), ply1.vertex(i),
ply2
                    .vertex(i)));
        }
        return tris;
    }

    private static List<Facet> triangulateLeftLower(Polyline ply1,
Polyline ply2) {
        int n = ply1.vertexNumber();
        ArrayList<Facet> tris = new ArrayList<Facet>((n - 1) * 2);
        for (int i = 0; i < n - 1; i++) {
            tris.add(new Facet(ply1.vertex(i + 1), ply1.vertex(i),
ply2
                    .vertex(i)));
            tris.add(new Facet(ply2.vertex(i + 1), ply1.vertex(i +
1), ply2
                    .vertex(i)));
        }
        return tris;
    }

    public static List<Facet> triangulateFrontLeft(Polygon neck,
            Polyline shoulder, Polyline[] ply1_ply2, Polygon
armpit) {
        Point[] pts1 = new Point[17];
        pts1[0] = neck.vertex(0);
        arrayReverseCopy(neck.vertices(), neck.vertexNumber() - 1,
pts1, 1, 6);
        System.arraycopy(shoulder.vertices(), 0, pts1, 7, 10);
        Polyline ply1 = new Polyline(pts1);

        Point[] pts2 = new Point[17];
        pts2[0] = armpit.vertex(0);
        arrayReverseCopy(armpit.vertices(), armpit.vertexNumber() -
1, pts2, 1,
                16);
        Polyline ply2 = new Polyline(pts2);

        int n = (ply1.vertexNumber() - 1) * 2 * (ply1_ply2.length +
1);
        List<Facet> tris = new ArrayList<Facet>(n);
        tris.addAll(triangulateLeftUpper(ply1, ply1_ply2[0]));
        for (int i = 0; i < ply1_ply2.length / 2; i++)
            tris.addAll(triangulateLeftUpper(ply1_ply2[i],
ply1_ply2[i + 1]));
```

```
        for (int i = ply1_ply2.length / 2; i < ply1_ply2.length -
1; i++)
            tris.addAll(triangulateLeftLower(ply1_ply2[i],
ply1_ply2[i + 1]));
        tris.addAll(triangulateLeftLower(ply1_ply2[ply1_ply2.length
- 1], ply2));
        return tris;
    }

    private static void arrayReverseCopy(Object[] src, int srcPos,
            Object[] dest, int destPos, int length) {
        int m = srcPos;
        int n = destPos;
        for (int i = 0; i < length; i++, m--, n++)
            dest[n] = src[m];
    }

    public static List<Facet> triangulateBackRight(Polygon neck,
            Polyline shoulder, Polyline[] ply1_ply2, Polygon
armpit) {
        Point[] pts1 = new Point[15];
        System.arraycopy(neck.vertices(), 12, pts1, 0, 5);
        System.arraycopy(shoulder.vertices(), 0, pts1, 5, 10);
        Polyline ply1 = new Polyline(pts1);

        Point[] pts2 = new Point[15];
        System.arraycopy(armpit.vertices(), 32, pts2, 0, 15);
        Polyline ply2 = new Polyline(pts2);

        int n = (ply1.vertexNumber() - 1) * 2 * (ply1_ply2.length +
1);
        List<Facet> tris = new ArrayList<Facet>(n);
        tris.addAll(triangulateRightUpper(ply1, ply1_ply2[0]));
        for (int i = 0; i < ply1_ply2.length / 2; i++)
            tris.addAll(triangulateRightUpper(ply1_ply2[i],
ply1_ply2[i + 1]));
        for (int i = ply1_ply2.length / 2; i < ply1_ply2.length -
1; i++)
            tris.addAll(triangulateRightLower(ply1_ply2[i],
ply1_ply2[i + 1]));

    tris.addAll(triangulateRightLower(ply1_ply2[ply1_ply2.length -
1], ply2));
        return tris;
    }

    public static List<Facet> triangulateBackLeft(Polygon neck,
            Polyline shoulder, Polyline[] ply1_ply2, Polygon
armpit) {
        Point[] pts1 = new Point[15];
        arrayReverseCopy(neck.vertices(), 12, pts1, 0, 5);
        System.arraycopy(shoulder.vertices(), 0, pts1, 5, 10);
```

```
        Polyline ply1 = new Polyline(pts1);

        Point[] pts2 = new Point[15];
        arrayReverseCopy(armpit.vertices(), 32, pts2, 0, 15);
        Polyline ply2 = new Polyline(pts2);

        int n = (ply1.vertexNumber() - 1) * 2 * (ply1_ply2.length +
1);
        List<Facet> tris = new ArrayList<Facet>(n);
        tris.addAll(triangulateLeftUpper(ply1, ply1_ply2[0]));
        for (int i = 0; i < ply1_ply2.length / 2; i++)
            tris.addAll(triangulateLeftUpper(ply1_ply2[i],
ply1_ply2[i + 1]));
        for (int i = ply1_ply2.length / 2; i < ply1_ply2.length -
1; i++)
            tris.addAll(triangulateLeftLower(ply1_ply2[i],
ply1_ply2[i + 1]));
        tris.addAll(triangulateLeftLower(ply1_ply2[ply1_ply2.length
- 1], ply2));
        return tris;
    }
}
```

HumanBody.java

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

public class HumanBody {
    // 颈顶线
    private Polygon jaw;
    // 颈部
    private PolygonInterpolation jaw_neck_interpolation;
    private Polygon[] jaw_neck;
    private List<Facet> jaw_neck_surf;
    // 颈围线
    private Polygon neck;
    private float neck_height_offset;
    private boolean neck_changed;
    // 双肩线
    private Polyline left_shoulder;
    private PolylineShoulder right_shoulder;
    private boolean shoulder_changed;
    // 前肩部
    private PolylineInterpolationFront
front_neck_armpit_interpolation;
    private Polyline[] right_front_neck_armpit;
    private List<Facet> right_front_neck_armpit_surf;
    private Polyline[] left_front_neck_armpit;
    private List<Facet> left_front_neck_armpit_surf;
```

```
    // 后肩部
    private PolylineInterpolationBack
back_neck_armpit_interpolation;
    private Polyline[] right_back_neck_armpit;
    private List<Facet> right_back_neck_armpit_surf;
    private Polyline[] left_back_neck_armpit;
    private List<Facet> left_back_neck_armpit_surf;
    // 腋窝线
    private PolygonArmpit armpit;
    // 胸部
    private PolygonInterpolation armpit_chest_interpolation;
    private Polygon[] armpit_chest;
    private List<Facet> armpit_chest_surf;
    // 胸围线
    private Polygon chest;
    private float chest_height_offset;
    private boolean chest_changed;
    // 腰部
    private PolygonInterpolation chest_waist_interpolation;
    private Polygon[] chest_waist;
    private List<Facet> chest_waist_surf;
    // 腰围线
    private Polygon waist;
    private float waist_height_offset;
    private boolean waist_changed;
    // 腹部
    private PolygonInterpolation waist_hip_interpolation;
    private Polygon[] waist_hip;
    private List<Facet> waist_hip_surf;
    // 臀围线
    private Polygon hip;
    private float hip_height_offset;
    private boolean hip_changed;
    // 股部
    private PolygonInterpolation hip_leg_interpolation;
    private Polygon[] hip_leg;
    private List<Facet> hip_leg_surf;
    // 腿根线
    private Polygon leg;

    public HumanBody() {
        jaw = new Polygon("data/01.3hd");
        neck = new Polygon("data/02.3hd");
        right_shoulder = new PolylineShoulder("data/03.3hd");
        left_shoulder = right_shoulder.another();
        armpit = new PolygonArmpit("data/04.3hd");
        chest = new Polygon("data/05.3hd");
        waist = new Polygon("data/06.3hd");
        hip = new Polygon("data/07.3hd");
        leg = new Polygon("data/08.3hd");
```

```
        neck_height_offset = 0;
        chest_height_offset = 0;
        waist_height_offset = 0;
        hip_height_offset = 0;

        neck_changed = false;
        shoulder_changed = false;
        chest_changed = false;
        waist_changed = false;
        hip_changed = false;

        jaw_neck_interpolation = new PolygonInterpolation(neck,
jaw,
                "data/11.3hd");
        jaw_neck = jaw_neck_interpolation.interpolate();
        front_neck_armpit_interpolation = new
PolylineInterpolationFront(neck,
                right_shoulder, armpit, "data/12.3hd");
        right_front_neck_armpit =
front_neck_armpit_interpolation.interpolate();
        left_front_neck_armpit =
Polyline.another(right_front_neck_armpit);
        back_neck_armpit_interpolation = new
PolylineInterpolationBack(neck,
                left_shoulder, armpit, "data/13.3hd");
        right_back_neck_armpit =
back_neck_armpit_interpolation.interpolate();
        left_back_neck_armpit =
Polyline.another(right_back_neck_armpit);
        armpit_chest_interpolation = new
PolygonInterpolation(chest, armpit,
                "data/14.3hd");
        armpit_chest = armpit_chest_interpolation.interpolate();
        chest_waist_interpolation = new PolygonInterpolation(waist,
chest,
                "data/15.3hd");
        chest_waist = chest_waist_interpolation.interpolate();
        waist_hip_interpolation = new PolygonInterpolation(hip,
waist,
                "data/16.3hd");
        waist_hip = waist_hip_interpolation.interpolate();
        hip_leg_interpolation = new PolygonInterpolation(leg, hip,
                "data/17.3hd");
        hip_leg = hip_leg_interpolation.interpolate();

        jaw_neck_surf = Facet.triangulateTwo(jaw, jaw_neck, neck);
        right_front_neck_armpit_surf =
Facet.triangulateFrontRight(neck,
                right_shoulder, right_front_neck_armpit, armpit);
        left_front_neck_armpit_surf =
Facet.triangulateFrontLeft(neck,
                left_shoulder, left_front_neck_armpit, armpit);
```

```
        right_back_neck_armpit_surf =
Facet.triangulateBackRight(neck,
                left_shoulder, right_back_neck_armpit, armpit);
        left_back_neck_armpit_surf =
Facet.triangulateBackLeft(neck,
                right_shoulder, left_back_neck_armpit, armpit);
        armpit_chest_surf = Facet.triangulateFour(armpit,
armpit_chest, chest);
        chest_waist_surf = Facet.triangulateFour(chest,
chest_waist, waist);
        waist_hip_surf = Facet.triangulateFour(waist, waist_hip,
hip);
        hip_leg_surf = Facet.triangulateFour(hip, hip_leg, leg);
    }

    public void update() {
        if (neck_changed) {
            jaw_neck = jaw_neck_interpolation.interpolate();
            jaw_neck_surf = Facet.triangulateTwo(jaw, jaw_neck,
neck);
        }
        if (neck_changed || shoulder_changed) {
            left_shoulder = right_shoulder.another();
        }
        if (neck_changed || shoulder_changed || chest_changed) {
            armpit.adjust(right_shoulder, chest);
            right_front_neck_armpit =
front_neck_armpit_interpolation
                    .interpolate();
            left_front_neck_armpit =
Polyline.another(right_front_neck_armpit);
            right_front_neck_armpit_surf =
Facet.triangulateFrontRight(neck,
                    right_shoulder, right_front_neck_armpit, armpit);
            left_front_neck_armpit_surf =
Facet.triangulateFrontLeft(neck,
                    left_shoulder, left_front_neck_armpit, armpit);
            right_back_neck_armpit = back_neck_armpit_interpolation
                    .interpolate();
            left_back_neck_armpit =
Polyline.another(right_back_neck_armpit);
            right_back_neck_armpit_surf =
Facet.triangulateBackRight(neck,
                    left_shoulder, right_back_neck_armpit, armpit);
            left_back_neck_armpit_surf =
Facet.triangulateBackLeft(neck,
                    right_shoulder, left_back_neck_armpit, armpit);
            armpit_chest =
armpit_chest_interpolation.interpolate();
            armpit_chest_surf = Facet.triangulateFour(armpit,
armpit_chest,
                    chest);
```

```
        }
        if (chest_changed || waist_changed) {
            chest_waist = chest_waist_interpolation.interpolate();
            chest_waist_surf = Facet.triangulateFour(chest,
chest_waist, waist);
        }
        if (waist_changed || hip_changed) {
            waist_hip = waist_hip_interpolation.interpolate();
            waist_hip_surf = Facet.triangulateFour(waist,
waist_hip, hip);
        }
        if (hip_changed) {
            hip_leg = hip_leg_interpolation.interpolate();
            hip_leg_surf = Facet.triangulateFour(hip, hip_leg,
leg);
        }

        neck_changed = false;
        shoulder_changed = false;
        chest_changed = false;
        waist_changed = false;
        hip_changed = false;
    }

    public void setNeckLength(float length) {
        neck.setLength(length);
        neck_changed = true;
        right_shoulder.adjust(neck);
        shoulder_changed = true;
    }

    public void offsetNeckHeight(float dHeight) {
        neck.offsetHeight(dHeight - neck_height_offset);
        neck_height_offset = dHeight;
        neck_changed = true;
        right_shoulder.adjust(neck);
        shoulder_changed = true;
    }

    public float getNeckLength() {
        return neck.length();
    }

    public void setShoulderLength(float length) {
        right_shoulder.setWidth(length);
        shoulder_changed = true;
    }

    public float getShoulderLength() {
        return right_shoulder.width();
    }
```

```
    public void setChestLength(float length) {
        chest.setLength(length);
        chest_changed = true;
    }

    public void offsetChestHeight(float dHeight) {
        chest.offsetHeight(dHeight - chest_height_offset);
        chest_height_offset = dHeight;
        chest_changed = true;

    }

    public float getChestLength() {
        return chest.length();
    }

    public void setWaistLength(float length) {
        waist.setLength(length);
        waist_changed = true;
    }

    public void offsetWaistHeight(float dHeight) {
        waist.offsetHeight(dHeight - waist_height_offset);
        waist_height_offset = dHeight;
        waist_changed = true;
    }

    public float getWaistLength() {
        return waist.length();
    }

    public void setHipLength(float length) {
        hip.setLength(length);
        hip_changed = true;
    }

    public void offsetHipHeight(float dHeight) {
        hip.offsetHeight(dHeight - hip_height_offset);
        hip_height_offset = dHeight;
        hip_changed = true;
    }

    public float getHipLength() {
        return hip.length();
    }

    public List<Polyline> polylines() {
        ArrayList<Polyline> plys = new ArrayList<Polyline>(100);
        plys.add(jaw);
        Collections.addAll(plys, jaw_neck);
        plys.add(neck);
        plys.add(right_shoulder);
```

```
        plys.add(left_shoulder);
        Collections.addAll(plys, right_front_neck_armpit);
        Collections.addAll(plys, left_front_neck_armpit);
        Collections.addAll(plys, right_back_neck_armpit);
        Collections.addAll(plys, left_back_neck_armpit);
        plys.add(armpit);
        Collections.addAll(plys, armpit_chest);
        plys.add(chest);
        Collections.addAll(plys, chest_waist);
        plys.add(waist);
        Collections.addAll(plys, waist_hip);
        plys.add(hip);
        Collections.addAll(plys, hip_leg);
        plys.add(leg);
        return plys;
    }

    public List<Polyline> importantPolylines() {
        ArrayList<Polyline> plys = new ArrayList<Polyline>(10);
        plys.add(neck);
        plys.add(right_shoulder);
        plys.add(left_shoulder);
        plys.add(chest);
        plys.add(waist);
        plys.add(hip);
        return plys;
    }

    public List<Facet> facets() {
        ArrayList<Facet> surfs = new ArrayList<Facet>(6500);
        surfs.addAll(jaw_neck_surf);
        surfs.addAll(right_front_neck_armpit_surf);
        surfs.addAll(left_front_neck_armpit_surf);
        surfs.addAll(right_back_neck_armpit_surf);
        surfs.addAll(left_back_neck_armpit_surf);
        surfs.addAll(armpit_chest_surf);
        surfs.addAll(chest_waist_surf);
        surfs.addAll(waist_hip_surf);
        surfs.addAll(hip_leg_surf);
        return surfs;
    }
}
```

STLWriter.java

```
import java.io.DataOutputStream;
import java.io.BufferedOutputStream;
import java.io.FileOutputStream;
import java.nio.ByteBuffer;
import java.nio.ByteOrder;

public class STLWriter {
```

```
    public static void saveStlFile(HumanBody hb, String fileName)
{
        try {
            DataOutputStream out = new DataOutputStream(
                    new BufferedOutputStream(new
FileOutputStream(fileName)));

            ByteBuffer buffer = ByteBuffer.allocate(1000000);
            buffer.order(ByteOrder.LITTLE_ENDIAN);

            int facetNumber = hb.facets().size();
            int fileSize = facetNumber * 50 + 84;

            // 文件头 80B
            for (int i = 0; i < 20; i++)
                buffer.putInt(0);
            // 三角面片个数 4B
            buffer.putInt(facetNumber);

            // 三角面片信息 50B/each
            for (Facet f : hb.facets()) {
                // 法向量 12B
                buffer.putFloat(f.normal().x());
                buffer.putFloat(f.normal().y());
                buffer.putFloat(f.normal().z());
                // 顶点1 12B
                buffer.putFloat(f.vertex(0).x());
                buffer.putFloat(f.vertex(0).y());
                buffer.putFloat(f.vertex(0).z());
                // 顶点2 12B
                buffer.putFloat(f.vertex(1).x());
                buffer.putFloat(f.vertex(1).y());
                buffer.putFloat(f.vertex(1).z());
                // 顶点3 12B
                buffer.putFloat(f.vertex(2).x());
                buffer.putFloat(f.vertex(2).y());
                buffer.putFloat(f.vertex(2).z());
                // 属性 2B
                buffer.putChar((char) 0);
            }

            byte[] bytes = buffer.array();
            for (int i = 0; i < fileSize; i++)
                out.writeByte(bytes[i]);
            out.close();
            buffer.clear();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

```
}
```

Main.java

```
//import java.net.*;
import processing.core.*;
import controlP5.*;

@SuppressWarnings("serial")
public class Main extends PApplet {
    private HumanBody hb;
    private float ry;
    private ControlP5 cp5;

    private boolean surfaceMode;

    private float neck_length;
    private float neck_height;
    private float shoulder_length;
    private float chest_length;
    private float chest_height;
    private float waist_length;
    private float waist_height;
    private float hip_length;
    private float hip_height;

    public void setup() {
        size(800, 600, P3D);
        // smooth();

        hb = new HumanBody();
        ry = 0;

        gui();

        neck_length = Float.NaN;
        neck_height = Float.NaN;
        shoulder_length = Float.NaN;
        chest_length = Float.NaN;
        chest_height = Float.NaN;
        waist_length = Float.NaN;
        waist_height = Float.NaN;
        hip_length = Float.NaN;
        hip_height = Float.NaN;
    }

    public void draw() {
        if (cp5.getController("Neck Length").getValue() !=
neck_length) {
            neck_length = cp5.getController("Neck
Length").getValue();
            hb.setNeckLength(neck_length);
```

```
        }
        if (cp5.getController("Neck Height").getValue() !=
neck_height) {
            neck_height = cp5.getController("Neck
Height").getValue();
            hb.offsetNeckHeight(neck_height);
        }
        if (cp5.getController("Shoulder Length").getValue() !=
shoulder_length) {
            shoulder_length = cp5.getController("Shoulder
Length").getValue();
            hb.setShoulderLength(shoulder_length);
        }
        if (cp5.getController("Chest Length").getValue() !=
chest_length) {
            chest_length = cp5.getController("Chest
Length").getValue();
            hb.setChestLength(chest_length);
        }
        if (cp5.getController("Chest Height").getValue() !=
chest_height) {
            chest_height = cp5.getController("Chest
Height").getValue();
            hb.offsetChestHeight(chest_height);
        }
        if (cp5.getController("Waist Length").getValue() !=
waist_length) {
            waist_length = cp5.getController("Waist
Length").getValue();
            hb.setWaistLength(waist_length);
        }
        if (cp5.getController("Waist Height").getValue() !=
waist_height) {
            waist_height = cp5.getController("Waist
Height").getValue();
            hb.offsetWaistHeight(waist_height);
        }
        if (cp5.getController("Hip Length").getValue() !=
hip_length) {
            hip_length = cp5.getController("Hip
Length").getValue();
            hb.setHipLength(hip_length);
        }
        if (cp5.getController("Hip Height").getValue() !=
hip_height) {
            hip_height = cp5.getController("Hip
Height").getValue();
            hb.offsetHipHeight(hip_height);
        }
        hb.update();

        background(255);
```

```
        if (surfaceMode)
            lights();

        pushMatrix();
        translate(250 + cp5.getController("H Pos").getValue(), 280
+ cp5
                .getController("V Pos").getValue());
        rotateY(ry);
        rotateX(cp5.getController("X Rot").getValue() / 180 * PI);
        rotateZ(cp5.getController("Y Rot").getValue() / 180 * PI);
        scale(cp5.getController("Scale").getValue());
        drawHumanBodySurfaces(hb);
        // drawHumanBodyPolylines(hb);
        drawHumanBodyImportantPolylines(hb);
        popMatrix();

        if (surfaceMode)
            noLights();

        ry += cp5.getController("Speed").getValue();
    }

    @SuppressWarnings("deprecation")
    private void gui() {
        surfaceMode = false;

        cp5 = new ControlP5(this);

        Group gview =
cp5.addGroup("View").setBackgroundColor(color(0, 32))
                .setBackgroundHeight(180);

        cp5.addSlider("H Pos").setPosition(20, 20).setSize(70, 20)
                .setRange(-100,
100).setValue(0).setColorLabel(color(64))
                .moveTo(gview);
        cp5.addSlider("V Pos").setPosition(130, 20).setSize(70, 20)
                .setRange(-200,
200).setValue(0).setColorLabel(color(64))
                .moveTo(gview);
        cp5.addSlider("X Rot").setPosition(20, 60).setSize(70, 20)
                .setRange(-90,
90).setValue(0).setColorLabel(color(64))
                .moveTo(gview);
        cp5.addSlider("Y Rot").setPosition(130, 60).setSize(70, 20)
                .setRange(-90,
90).setValue(0).setColorLabel(color(64))
                .moveTo(gview);
        cp5.addSlider("Scale").setPosition(20, 100).setSize(70, 20)
                .setRange(0.2f,
1.0f).setValue(0.6f).setColorLabel(color(64))
```

```
                .moveTo(gview);
        cp5.addSlider("Speed").setPosition(130, 100).setSize(70,
20)
                .setRange(0,
0.02f).setValue(0.005f).setColorLabel(color(64))
                .moveTo(gview);
        cp5.addButton("Mode").setPosition(20, 140).setSize(60, 20)
                .plugTo(this, "switchViewMode").moveTo(gview);
        cp5.addButton("Save").setPosition(90, 140).setSize(60, 20)
                .plugTo(this, "saveHumanBody").moveTo(gview);
        cp5.addButton("Robot").setPosition(160, 140).setSize(60,
20)
                .plugTo(this, "controlRobot").moveTo(gview);

        Group gneck =
cp5.addGroup("Neck").setBackgroundColor(color(0, 32))
                .setBackgroundHeight(100);

        cp5.addSlider("Neck Length").setPosition(20,
20).setSize(150, 20)
                .setRange(380,
440).setValue(403).setColorLabel(color(64))
                .moveTo(gneck);
        cp5.addSlider("Neck Height").setPosition(20,
60).setSize(150, 20)
                .setRange(-20,
20).setValue(0).setColorLabel(color(64))
                .moveTo(gneck);

        Group gshoulder = cp5.addGroup("Shoulder")
                .setBackgroundColor(color(0,
32)).setBackgroundHeight(100);

        cp5.addSlider("Shoulder Length").setPosition(20,
20).setSize(150, 20)
                .setRange(435,
500).setValue(460).setColorLabel(color(64))
                .moveTo(gshoulder);

        Group gchest =
cp5.addGroup("Chest").setBackgroundColor(color(0, 32))
                .setBackgroundHeight(100);

        cp5.addSlider("Chest Length").setPosition(20,
20).setSize(150, 20)
                .setRange(880,
1100).setValue(960).setColorLabel(color(64))
                .moveTo(gchest);
        cp5.addSlider("Chest Height").setPosition(20,
60).setSize(150, 20)
                .setRange(-20,
20).setValue(0).setColorLabel(color(64))
```

```
                .moveTo(gchest);

        Group gwaist =
cp5.addGroup("Waist").setBackgroundColor(color(0, 32))
                .setBackgroundHeight(100);

        cp5.addSlider("Waist Length").setPosition(20,
20).setSize(150, 20)
                .setRange(740,
940).setValue(820).setColorLabel(color(64))
                .moveTo(gwaist);
        cp5.addSlider("Waist Height").setPosition(20,
60).setSize(150, 20)
                .setRange(-20,
20).setValue(0).setColorLabel(color(64))
                .moveTo(gwaist);

        Group ghip =
cp5.addGroup("Hip").setBackgroundColor(color(0, 32))
                .setBackgroundHeight(100);

        cp5.addSlider("Hip Length").setPosition(20,
20).setSize(150, 20)
                .setRange(900,
1080).setValue(970).setColorLabel(color(64))
                .moveTo(ghip);
        cp5.addSlider("Hip Height").setPosition(20,
60).setSize(150, 20)
                .setRange(-20,
20).setValue(0).setColorLabel(color(64))
                .moveTo(ghip);

        Accordion accordionView = cp5.addAccordion("View Control")
                .setPosition(500, 50).setWidth(250).addItem(gview);
        accordionView.setCollapseMode(ControlP5.SINGLE);
        accordionView.open(0);

        Accordion accordionSize = cp5.addAccordion("Size Control")
                .setPosition(500, 280).setWidth(250).addItem(gneck)
                .addItem(gshoulder).addItem(gchest).addItem(gwaist)
                .addItem(ghip);
        accordionSize.setCollapseMode(ControlP5.SINGLE);
        accordionSize.open(0);
    }

    public void switchViewMode(int theValue) {
        surfaceMode = !surfaceMode;
    }

    public void saveHumanBody(int theValue) {
        STLWriter.saveStlFile(hb, "model/ExportedModel.stl");
    }
```

```
    public void controlRobot(int theValue) {
        // 方式一：打开浏览器
        // try {
        // URI uri = URI.create("http://localhost");
        // java.awt.Desktop dp = java.awt.Desktop.getDesktop();
        // if(dp.isSupported(java.awt.Desktop.Action.BROWSE)) {
        // dp.browse(uri);
        // }
        // } catch(Exception e) {
        // }
        // 方式二：直接访问URL
        // try{
        // URL url = new URL("http://localhost");
        // HttpURLConnection conn =
(HttpURLConnection)url.openConnection();
        // conn.setRequestMethod("GET");
        // conn.connect();
        // } catch (Exception e) {
        // e.printStackTrace();
        // }
    }

    private void drawPolyline(Polyline p) {
        beginShape();
        for (int i = 0; i < p.vertexNumber(); i++)
            vertex(p.vertex(i).x(), p.vertex(i).z(),
p.vertex(i).y());
        if (p.isClosed())
            endShape(CLOSE);
        else
            endShape();
    }

    @SuppressWarnings("unused")
    private void drawHumanBodyPolylines(HumanBody hb) {
        noFill();
        strokeWeight(1);
        stroke(64);
        for (Polyline ply : hb.polylines())
            drawPolyline(ply);
    }

    private void drawHumanBodyImportantPolylines(HumanBody hb) {
        noFill();
        strokeWeight(3);
        stroke(255, 0, 0);
        for (Polyline ply : hb.importantPolylines())
            drawPolyline(ply);
    }

    private void drawFacet(Facet t) {
```

```
        beginShape();
        vertex(t.vertex(0).x(), t.vertex(0).z(), t.vertex(0).y());
        vertex(t.vertex(1).x(), t.vertex(1).z(), t.vertex(1).y());
        vertex(t.vertex(2).x(), t.vertex(2).z(), t.vertex(2).y());
        endShape(CLOSE);
    }

    private void drawHumanBodySurfaces(HumanBody hb) {
        if (surfaceMode) {
            fill(224);
            noStroke();
        } else {
            noFill();
            strokeWeight(1);
            stroke(64);
        }
        for (Facet surf : hb.facets())
            drawFacet(surf);
    }
}
```

本科生毕业论文（设计）任务书

一、题目：试衣机器人的参数化三维人体模型设计

二、指导教师对毕业论文（设计）的进度安排及任务要求：

前期调研（1 月～3 月）：查找资料，完成文献翻译和开题报告
初步实践（3 月～4 月）：三维人体模型研究，确定参数和建模方法
主体研发（4 月～5 月）：人体建模程序编写和可视化实现
后期完善（5 月中旬）：与试衣机器人在变形和控制方面进行配合
论文撰写（5 月末）：总结并完成毕业设计报告

起讫日期 2013 年 12 月 29 日 至 2014 年 5 月 31 日

指导教师（签名）职称

三、系或研究所审核意见：

负责人（签名）

年 月 日

毕 业 论 文（设计） 考 核

一、指导教师对毕业论文（设计）的评语：

本课题研究试衣机器人的参数化三维人体模型设计，与本专业方向相关，符合要求。选题难度较高，工作量较大。该同学在短时间内学会计算机图形学相关的知识并很好地完成课题的研究，体现了极强的研究能力。在毕业设计过程中，该同学认真负责、积极主动地完成本课题的研究工作，同时又能帮助项目组完成相关的辅助工作。课题采用参数化的方法实现三维人体模型的变形，并将关键数据传递给试衣机器人的控制部分引导机器人的变形。系统的界面友好，软件能稳定运行，功能达到了预期的效果。设计文档结构完整，格式规范，条理清楚，语句通顺。该设计达到本科生毕业设计的要求，建议安排参加答辩。

指导教师(签名）

年 月 日

二、答辩小组对毕业论文（设计）的答辩评语及总评成绩：

成绩比例	文献综述占（10%）	开题报告占（20%）	外文翻译占（10%）	毕业设计展览（20%）	毕业论文(设计）质量及答辩占（40%）	总评成绩
分值						

答辩小组负责人（签名）

年 月 日

浙江大学

本科生毕业设计
开题报告

学生姓名：马轲

学生学号：3100100423

指导教师：张东亮

年级与专业：设计创新班 10 级

所在学院：国际设计研究院

一、题目：试衣机器人的参数化三维人体模型设计

二、指导教师对国内外设计调研、设计概念、设计计划及创新点等具体要求：

对于国内外设计调研，要从试衣机器人与虚拟试衣系统的市场需求入手，分析其发展与应用前景，进而分类研究当今虚拟试衣系统的技术手段，最后落脚到三维人体建模方面。

对于设计概念，要在充分调研现有设计的基础上，针对问题进行创新，同时灵活运用第一专业所学习的工程知识，从创新性和可行性两个方面制定设计概念。

对于设计计划，应当充分理解每一阶段的任务，根据任务工作量规划项目时间表，严格按照计划进行，并需要特别重视后期调试与完善工作。

对于创新点，要在现有设计的基础上进行创新，重点突破现有设计中存在的一些问题，并充分考虑到可行性。

指导教师（签名）_____

年　　月　　日

毕业设计开题报告考核

答辩小组对开题报告成绩评定：

成绩比例	设计调研占（20%）	设计方法与支撑技术　占（10%）	设计比较占（10%）
分 值			

开题报告答辩小组负责人（签名）_____

年　　月　　日

目录

本科毕业设计　开题报告

1. 课题背景

本项目为“试衣机器人与虚拟试衣系统”课题的子项目之一。

随着社会经济的快速发展，网络购物已经成为当前消费者的主要购物方式之一。在网上销售的诸多商品中，许多商品的选购是不依赖于消费者自身的情况的。例如消费性电子产品、书籍、食品等等，它们能够提供明确的功能、服务，或是具有确定的品质、指标，不会因为不同的消费者而产生变化。与其相反的，服装作为一类特殊的商品，其价值则是与消费者紧密相关的。对于同一件服装，不同的消费者穿着之后会体现出不同的效果，这也直接影响了消费者的消费决策。因此，消费者往往不满足于看到网站上对服装的简单文字说明和图片展示，而更倾向于看到自身穿着这些服装的效果。

消费者对服装的选择主要基于两点考量。第一，服装的款式和颜色。消费者对服装款式颜色的考量，不仅包括消费者对某一件服装本身的喜爱与否，还包括这一件服装与自己已有的服装是否搭配。服装设计学科发展至今，服装的款式与颜色均具有很大的可变性，难以找到一个或一组能够涵盖所有服装的通用模式。第二，服装的大小和舒适性。其中舒适性又包含了服装本身设计、用料方面的舒适性，以及穿着在用户身上的合身性。每个消费者身材不同，在商店直销模式中，消费者可以通过亲自试穿来完成对服装大小、合身性的判断与选择。尽管服装大多分不同尺码进行销售，在网络服装销售的特定环境下，消费者很难准确地判断究竟哪一个尺码适合自己，只能通过历史的购买经验进行估计。

为迎合服装网购市场的这类需求，虚拟试衣系统应运而生。目前虚拟试衣系统主要有两大类解决方案。一类是利用三维计算机图像仿真来实现的。消费者通过拍摄设备或手动输入自己的相关参数，利用计算机生成人体三维模型，再将服装的三维模型覆盖在其上，生成可视的仿真结果供消费者观看。其优点为用户可以看到完全符合自身的三维模型，并且能够看到实时、全角度的仿真结果；而缺点则为为每件服装建立三维模型成本太高，且目前的技术得到的仿真结果往往缺乏真实感。另一类是物理模特的试衣系统。服装经销商事先使用不同尺寸的服装配合不同尺寸的模特，得到服装穿着效果的多角度照片，而消费者则可以选择最符合自己身材的模特，进而看到试穿特定服装的效果。其优点在于成本低，只需要进行拍照，且展示的效果较好，能保留服装的质感；缺点在于对经销商来说工作量稍大，消费者也无法看到自身试穿衣服的效果。

本课题意在设计一个完整的试衣服务体系，该体系可直接用于服装零售业，在为商家和消费者带来方便的同时，推动产业的发展并带来巨额收益，如图 1 所示。该体系的主要服务流程如下：服装厂商购买试衣机器人及其配套软件，用于服装设计、生产及销售的整个流程；通过在软件中调节人体的主要尺寸参数，能够直观地看到相应的虚拟人体模型，同时试衣机器人的相应尺寸也会发生变化；服装设计师可在设计过程中将服装穿着在可变尺寸的试衣机器人上，查看相应的穿着效果，并作出改善；服装生产后，将不同尺寸的服装穿着在试衣机器人上，调节试衣机器人尺寸参数以模拟不同身材的人穿着该服装的效果，并利用拍摄系统进行多角度的拍摄；利用图像处理软件对照片进行批量加工，删除拍摄背景而仅保留服装部分，再将服装图片覆盖到真实模特的图片上，从而模拟试穿效果；在服装厂商的服装零售网站上整合虚拟试衣的模块，该模块允许用户选择最接近自己身材的尺寸组合作为自己的虚拟模特，然后调用前述的不同尺寸服装穿着在不同尺寸机器人上的多角度图片，以合成 3D 的方式进行试穿效果展示；消费者进而可以直观地观察到衣服穿着在自己身上的近似效果，从而为购买决策提供参考。此外，本设计还包含了对于服装合体性、舒适性本身的研究，从人机工程学的角度为服装设计和服装购买决策提供参考；它还包括一个适用于本服务体系的完整的商业模型的设计。本课题将针对这一设计实现一个能够完成基本功能的产品原型，在保证基本设计思路与应用原则的基础上，对一些功能或模块加以简化，从而降低实现难度、提升项目可行性。

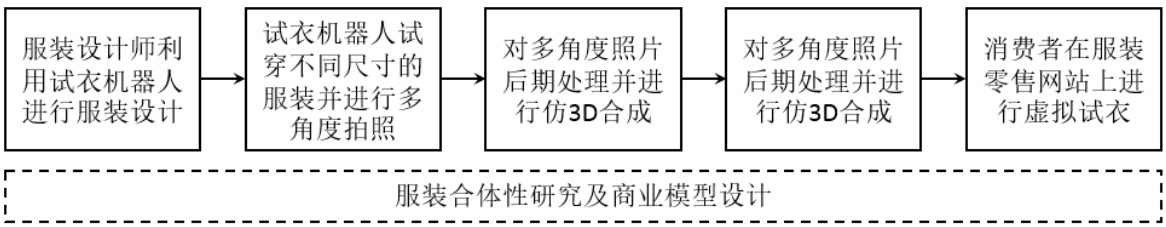

图 1. 试衣服务体系流程图

三维人体建模作为该课题的子项目之一，主要应用在试衣机器人的配套软件当中。如前文所述，该配套软件将实现如下功能：显示三维人体模型；提供数个可变的人体尺寸参数，供用户进行调节；在尺寸参数变化的同时，改变三维人体模型的形状并实时显示；与此同时，配合试衣机器人的控制系统对应调整试衣机器人的外形。因此，其中涉及到的三维人体建模将具有如下两个关键特征：不是静态的三维人体模型，而是具有一定的可变参数的动态人体模型；模型在外观上应与试衣机器人保持一致，且在对应参数进行调整时保持同步变化。三维人体建模是“试衣机器人与虚拟试衣系统”体系中有关人机交互的重要环节，它令用户在调整尺寸参数的时候可以直观地观察到调整效果，而不需要一边在计算机上调整参数、一边观看试衣机器人的实际变化效果。

2. 设计目的

本课题的目的在于开发一个可变尺寸的试衣机器人和虚拟试衣系统，以及相应的全套试衣服务体系，应用于服装的设计、生产及销售的整个流程中。本系统主要由两部分组成：第一部分是试衣机器人，通过计算机软件对人体特征尺寸参数进行调整，得到可视化的人体三维模型的反馈，同时通过控制系统相关联地改变试衣机器人的外形。第二部分是虚拟试衣系统，将穿着服装的试衣机器人放置在拍摄装置中，拍摄装置自动拍摄不同角度的照片，在对照片经过后期处理后，在服装销售网站上进行多角度的展示，供消费者参考。

这个体系主要包括下述设计任务：

（一）试衣机器人部分：

1）参数化人体模型设计：本任务重点在于合理地对人体模型进行分块，使得分块后的人体模型既能便于改变形状的控制过程，又使得整体与真实人体形状相似，不会出现严重的失真变形，如图 2。本任务还需要确定试衣机器人的可变尺寸参数，研究这些参数发生变化时会如何改变人体模型的形状，为后续任务奠定基础。

图 2. 人体模型分块示意

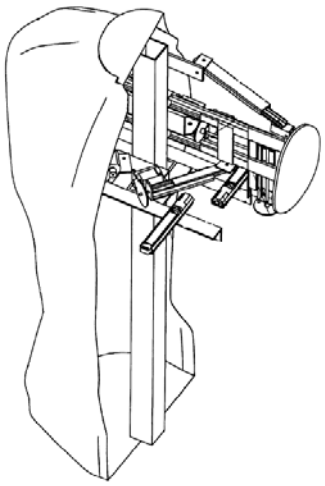

图 3. 试衣机器人机械结构示意

2）试衣机器人硬件设计：本任务重点在于制作试衣机器人的硬件部分，如图 3。这又包含两个主要方面：其一是外形设计，选择合适的材料制作人体模型的构造块，利用一定的结构对构造块进行支撑以构成表面，此外还要选择一种弹性材料覆盖在外面以增加平滑度；其二是内部机械结构设计，在机器人内部搭建机械结构，使得各个构造块可以进行独立或牵连运动，从而改变机器人的形状。

3）试衣机器人控制设计：本任务重点在于对上述机械结构引入电动控制装置，利用步进电机等器件使得机器人外形的变化可以被轻松调整和改变。其难点在于对各个构造块的控制并非相互独立的，而是有所关联的，如何正确的推导出各个构造块之间的相对运动关系也是本任务的关键之一。

4）试衣机器人软件设计：本任务重点在于开发试衣机器人配套的可视化控制软件。其功能包括指导用户调整人体模型的特征尺寸参数，实时显示变形的三维人体模型，联系控制系统使试衣机器人做出相应的调整等。本任务的难点在于需要人体模型根据某几个参数的变化而变化，同时还要试衣机器人的外形变化相同步。

（二）虚拟试衣系统部分：

1）拍摄装置设计：本任务重点在于设计一个能够对服装（及机器人）进行多角度自动拍摄的拍摄装置，如图 4。该拍摄装置应主要由旋转台、相机和相应的控制系统组成，穿着服装的机器人在旋转台上进行旋转，每隔一定的角度旋转台停止一次并控制相机进行拍照，随后重复此过程以获得多角度的照片。

图 4. 拍摄系统示意

图 5.服装图片处理示意

2）服装图片处理软件设计：本任务重点在于对拍摄的照片利用图像处理技术进行后期加工，去除拍摄背景仅保留服装部分，并将其覆盖在真实模特的图片上，如图 5。此外还需要利用 Flash 或 HTML5 的技术，将多角度的服装照片组合为仿 3D 的可旋转效果进行展示。

3）服装展示销售网站设计：本任务重点在于设计并开发一个服装零售网站。与其他同类网站不同之处在于，本网站应具有高度互动性，它允许消费者输入一些特征尺寸来产生自己的人体模型，并能看到服装穿着在该人体模型上的效果，从而为消费决策提供参考。

（三）相关研究部分：

1）服装合身性研究：本任务从两个方向对服装的合体性进行研究。其一是建立起一套服装合身性的评价标准，利用客观的数据而非主观的判断来评价服装的合身性。其二是通过对穿着在模特身上的服装的照片进行分析，获取有关合身性方面的信息并加以评价。

2）商业模式设计：本任务从商业的角度对试衣机器人与虚拟试衣系统进行分析与设计，重点包括市场分析、用户调查、应用前景、服务设计、品牌推广、盈利模式等方面。

其中本项目主要负责三维人体建模（试衣机器人软件设计）部分。具体研究要点包括：

1）三维人体模型的构建：通过彩色图片、深度图片、三维扫描等手段获取人体模型的原始数据，包括人体轮廓曲线或三位表面等，形成静态的三维人体模型为构建可变形的人体模型奠定基础。

2）人体特征尺寸的提取：首先确定人体的几个特征性的尺寸参数，再从静态的人体模型中通过三维计算手段对这些参数进行测量，形成准确、健壮的测量算法。

3）三维人体模型的变形：研究在特征尺寸参数变化的同时人体模型表面如何变形，并利用三维网格变形技术将观察结果在三维模型上实现出来，在变形后再利用参数测量算法验证变形结果，最终形成可变形的三维人体模型。

4）三维人体模型与试衣机器人变形的同步：将三维人体模型的变形结果和试衣机器人的物理变形结果相比较，并相互调整，使两者的变形结果较为相似。

5）计算机软件与控制系统的整合：在计算机三维人体模型显示软件中调用试衣机器人的控制 API，以实现同步变形。

3. 设计调研

3.1. 需求分析

试衣机器人与虚拟试衣系统具有广阔的市场前景，这种前景源自于快速发展的服装网购行业以及其中潜在的问题。

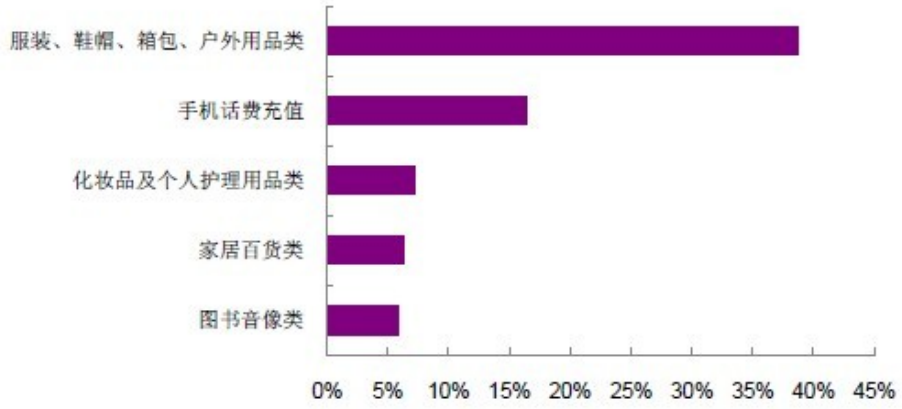

图 6. 中国网购用户最常购买的商品条形图

统计显示，服装类商品在 2012 年中国网购用户最常购买的商品中占 38.7%，位居榜首，见图 6。而事实上，2012 年服装网购的交易规模高达 3188.8 亿元，相比 2011 年增长率高达 56%，交易额占整个网购市场的 27%，同样也是位居榜首。虽然服装网购的增长率正逐年下降并趋于稳定，无法与发展初期的近 200%相比，但其交易规模正在迅速发展，其增长速度仍然超过了服装店面销售的增长速度，正在一步步占领服装零售市场。此外，当前服装店面销售的市场规模依然远远高于服装网购，这说明，服装网购有着广阔的潜在市场，在可预见的未来数年内还可以得到极大的发展。

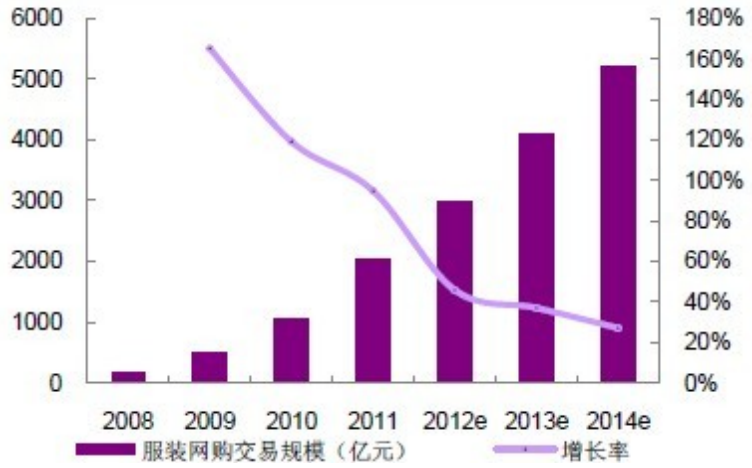

图 7. 服装网购交易规模柱状图

与此同时，一个不容忽视的事实是，服装电商的退货率基本为 10%至 20%，这是一个相对较高的数字。除了小部分是因为服装本身有质量问题或消费者对服装款式不满意，由于尺寸不合适引起的退货高达三分之二。即是说，由于服装网上销售无法提供试穿服务，消费者难以准确买到适合自己身材的服装，进而引发的退货比重高达总交易数的 6%至 12%，严重影响了服装厂商的利润和消费者的购物体验。虚拟试衣系统是这一问题的良好解决方案之一。

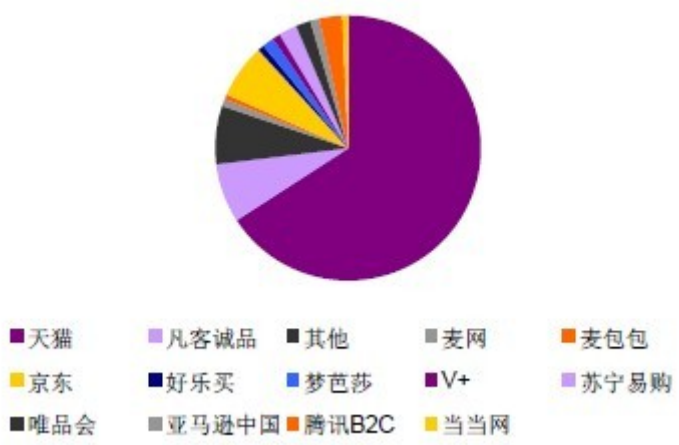

图 8. 2012 年 B2C 服装网购主要企业份额扇形图

服装网购与整个网购的发展历程类似，同样从初期的C2C模式逐步向B2C模式演进。一些传统的线下服装厂商发现了服装网购这一机遇，纷纷开设了官方的网络销售平台，这使得 2009 年至 2012 年短短三年间，B2C 模式的服装网购市场增长了近 5 倍。当前的 B2C 模式服装网购平台主要可以分为两类：允许大量商家入驻的平台型商城和专注于少数品牌发展的垂直型商城。虽然当前的服装网购仍然以平台型商城为主导，但垂直型商城更能迎合消费者对于服装商品的品牌和质量的需求，同时还能满足一些高端消费者的定制需要，在未来应当能够获得更大发展，见图 8。由于在垂直型商城中，零售商往往即是服装厂商本身，资金较为宽裕，仓储更加集中，非常适合引入试衣机器人与虚拟试衣系统以提升消费者的购物体验和降低退货率。这类垂直型商城可被认为是试衣机器人与虚拟试衣系统的理想客户。

“高端消费”是服装网购的另一个发展趋势。当前网购消费者主流群体正由 20-30 岁的年轻人向 30-40 岁的中龄人群发展，消费者普遍收入水平变高，对商品的需求从性价比向品牌、质量转变。这使得一些高端服装定制服务应运而生，而试衣机器人与虚拟试衣系统恰恰满足了网上服装定制的需要，因而必定能在这一领域有所作为。

3.2. 虚拟试衣技术分类分析

虚拟试衣系统起源于 2005 年，但从 2010 年后才得到广泛报道。这种系统现在已经可以从许多提供商中获取到，并且被越来越多的服装零售商用在它们的网店中。虚拟试衣系统可以根据它所解决的问题（大小、合身或样式），或它所使用的技术手段来进行分类。现在已经有许多不同种类的技术手段，其中使用最广泛的包括：尺寸建议服务、身体扫描、3D 解决方案、3D 顾客模型、利用 3D 仿真的试衣间、试穿人台/混合搭配、照片精度的虚拟试衣间、增强现实和真实模型。

尺寸建议系统能够基于一系列因素提供给顾客一个建议的服装尺寸。通常来讲，这类系统只能给出最低限度的合身信息，毕竟尺寸和合身仍存在很大区别。

图 9. 身体扫描技术

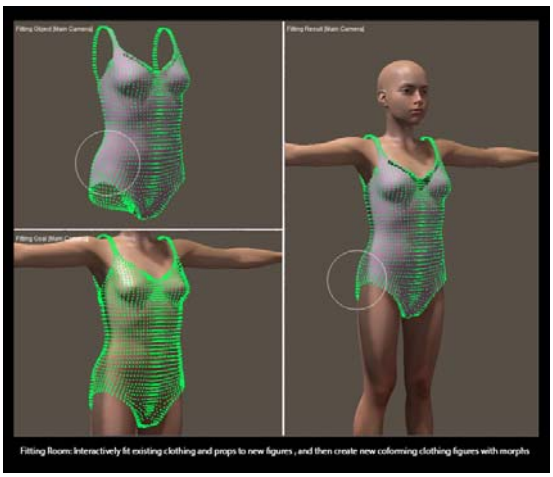

图 10. 3D 试衣间技术

身体扫描技术包含两类：一类使用较为简单的技术如摄像头、手机相机或微软 Kinect 设备，它们只要求顾客站在一个距设备固定的距离外即可使用；另一类使用更为复杂的技术包括激光、毫米波扫描或 Kinect 传感器阵列，它们往往体型巨大、价格昂贵，多由购物中心统一提供，见图 9。

3D 试衣间使用计算机生成的 3D 图像来创造一个接近于计算机游戏虚拟世界中所见到的体验，见图 10。这类解决方案利用顾客的人体测量和形状信息来生成虚拟人台。

3D 顾客模型允许顾客创造一个自己的 3D 版本，这可以通过扫描设备、身体测量或直接提供身体信息来实现。3D 模型可以进行一定的调整来改变身体形状。服装能够显示在 3D 模型上，且顾客可以上传面部照片来个性化 3D 模型。

3D 仿真试衣间结合了 3D 解决方案和照片精度试衣间的特点。使用照片和简单的身体测量，这种解决方案可以生成 3D 人台，它可以准确可视化穿着指定服装的顾客的形象。通常，这种系统会根据输入的测量数据建议一个合适的服装尺寸，但顾客也可以选择其他尺寸来估计其合身性。

在试穿人台技术中，服装被穿在真实人台上然后被拍照。这些服装随后被从图像中取出，顾客则可以拖拽不同的服装物件到同一个人台上，来观看它们的搭配效果，见图 11。

图 11. 试穿人台技术

图 12. 增强现实试衣技术

照片精度虚拟试衣技术是真实模型和试穿人台两种技术的融合。不同于将服装穿着在类似于顾客身体形状的模特身上进行拍照的技术，在这种技术中图片是使用可变性的机械人台得到的。电脑控制的人台可以快速变形，不同尺寸的服装被穿着在其上并进行拍照，随后连同测量数据存储在数据库中。

大多数增强现实解决方案通过将服装图片叠加在顾客图像之上来工作，见图 12。一些解决方案允许将服装叠加在顾客的实时视频上，这种视频可以由桌面摄像头或智能手机相机进行获取。

真实模型技术包括两个变种，其中一种已经广泛应用于许多网店。

3.3. 三维人体建模研究现状

本项目着重于对虚拟试衣系统中三维人体建模的研究。三维人体建模技术是计算机图形学领域的一个长期的研究热点。最早的虚拟三维人体模型是由三维线框来表示的，它结构简单、操作方便，但具有无法实现三维人体模型自动消隐和具有二义性的缺点。20 世纪 70 年代后实体模型逐渐发展，克服了上述缺陷，能够完整的表达人体模型的拓扑关系并生成具有真实感的图像，但结构复杂，应用受到限制。曲面建模作为当今最常用的人体建模方法，适用于表面光滑但难以用简单数学模型描述的复杂曲面。N.M. Thalmann 和 D. Thalmann（1987）最早使用多边形表面生成了虚拟人体。Forsey（1991）使用分层 B 样条技术进行三维人体建模。Douros 等（1999）则使用 B 样条曲面重建三维扫描人体。此外，A.H. Barr（1987）提出引入人体的物理特性，而非仅仅使用几何特性用于人体建模，使得三维人体的动画仿真效果更为真实。Chadwick 等（1989）提出使用多个层次进行人体建模，从而使人体模型在生理学的角度更为真实。Thalmann 等（1996）在此基础上提出了一种基于解剖学的分层建模算法。

4. 设计方法

4.1. 三维模型的表示方法

三维模型在计算机中的表示方式直接关乎到我们如何对其进行操作。当前主流的三维模型表示方法有以下三种：

体素。这是一种非常直观的表达三维模型的方法，它把三维空间均匀分成三维网格，从而创建所谓的体积图像，见图 13。图像中每个网格中的元素被称为体积像素，简称体素，也即是三维版本的像素。然而在三维世界中，数量的增长是指数级的，这也使得利用体素表达的三维模型往往占用过大的资源，难以处理。

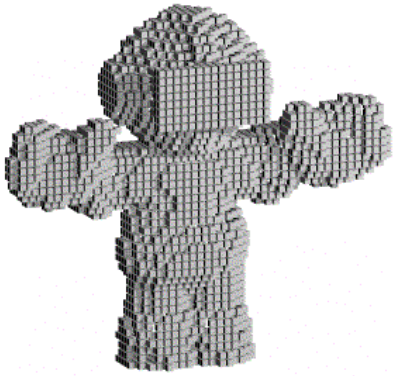

图 13. 体素模型

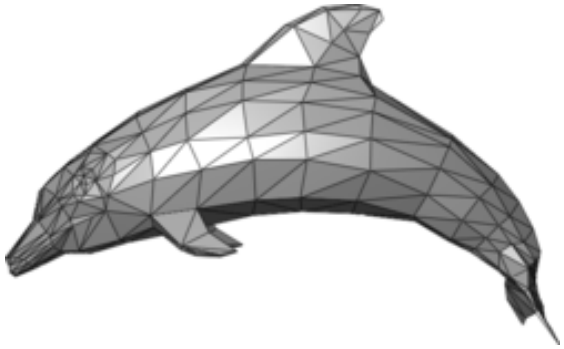

图 14. 网格模型

网格。这是一种非常紧凑的表达三维数据的方式，且被计算机游戏和三维动画所广泛采用，见图 14。它使用多边形，通常是三角形，来构建物体的表面。平坦的表面仅仅需要少数多边形，而复杂的区域则需要大量的多边形才能高精度地表达。这其实就是二维矢量图的三维版本。网格模型非常灵活高效，但却难以构建，因为它需要选择顶点位置和法线方向并确定表面的连接性。

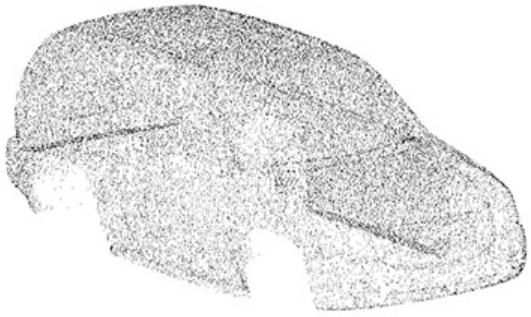

图 15. 点云模型

点云。这其实是没有相互连接的三维点的集合，见图 15。之所以称为“云”，是因为它在可视化后，点之间缺少连接使得它们好像漂浮在空间中。点云的最简单版本中，每个点可以只包含三维位置信息，但其实还可以包含其他属性，比如颜色和法线方向。点云可以在作为全局模型，也可以作为在表面重建创建网格模型前的中间步骤。

从操作便捷性、表达真实性等角度考虑，本项目主要采用网格模型来表达三维人体模型。但在处理过程中可能会用到点云等其他表达方式。

4.2. 人体模型的构建方法

人体建模方法主要包括采用点线构造三维图形的线框建模、增加三维人体实心部分表达的实体建模、基于光滑曲面数学描述的曲面建模以及引入了人体物理信息和环境时间因素的基于物理的建模方法。而对于用于服装 CAD 的人体模型，则包括一些更为具体的要求。当前的服装 CAD 领域的人体建模技术主要有四种：基于特征的人体曲面建模、参数化曲面建模、多面体建模和以网格边界线为连续条件的三维人体建模。

基于特征的人体曲面建模首先将人体模型划分为数个基本结构特征，包括头、躯干、手、脚等，而每个部分的数据结构和造型方法可能并不相同。这种方法打破了对整个人体模型采用同一种曲面表达的禁锢，使得为不同的特征部位采用不同的曲面建模方法成为可能，因而这种方式相对更为灵活，也易于对某个部位的模型进行修改而不影响其他部位。这种方法通常先由测量得到的身体的特征点形成特征曲线，再与具有一定通用性的几何造型曲线共同形成网格，进而生成曲面。

参数化曲面建模是利用人体的形状特征间的几何约束关系来进行人体建模的一种方法，它可以以一个统一的模型表达一组在形状上具有相似性的模型。参数化建模着重考虑人体各个部分之间固有的比例关系，在由人机工程学中定义的诸多人体测量参数中挑选出最具决定性的数个参数，再由主要参数根据相关约束关系推演出其他造型参数，从而获得满足一定参数约束的人体模型。

多面体建模是以一种小平面逼近曲面的思想进行的人体建模方法。它通过首先构造多面体，再对多面体的定点、棱线、表面进行局部修改以达到与目标实体外形近似的目的，而后由多面体的顶点自动产生自由曲面的控制点，从而生成人体模型表面。在于当前的 CAD 系统大多由三角或四边形网格表面来表达曲面，这一类建模方法在 CAD 系统中尤为常用。

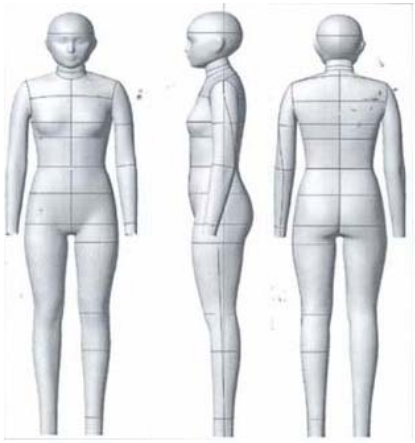

图 16. 参数化人体模型

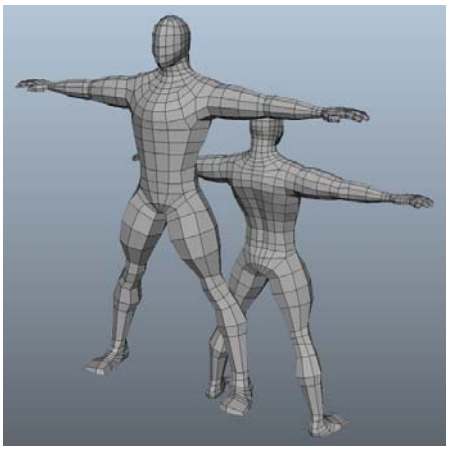

图 17. 多面体人体模型

以网格边界线为连续条件的三维人体建模是基于曲线的建模和基于三角面片的建模的结合。它首先根据人体的特征构造出横向和纵向的 B 样条曲线，形成空间的曲线网格，再将这些曲线离散化，形成空间中的人体模型表面点云，最后利用这些点作为三角面片的定点拼合构成三维人体表面。

综上所述，参数化曲面建模方法能够方便我们使用抽象的参数对人体模型进行调整，更为符合我们的设计需求，因而项目中将采用这一人体建模方法。

5. 预期效果

本课题预期能够开发出试衣机器人及虚拟试衣系统的原型。具体包含以下几个目标：

1）试衣机器人使用男性人台，能够对数个关键尺寸进行变化，并在程序中实时显示模型变形的效果；

2）拍摄系统可以在数分钟内对模型进行多角度拍摄，并依赖手工操作将拍摄图像中的服装部分进行编辑；

3）建立能够展示不同尺寸服装穿着在不同尺寸人台上的效果的网站，可用于服装的展示或在线销售；

4）形成较为完整、合理的商业模型。

本项目部分的具体目标包括：

1）编写一个能够显示可变性人体半身模型的程序，它允许用户改变模型的几个特征尺寸（如肩宽、胸围等），并实时显示模型变形的效果；

2）上述虚拟模型应与实体试衣机器人进行配合（包括可控参数一致、变形效果类似）；

3）虚拟人体模型程序中的控制界面应与试衣机器人控制程序进行整合，在改变虚拟模型的同时控制试衣机器人进行相应的变形。

6. 技术领域支持

6.1. 硬件设备

本课题的大部分任务是计算机程序的开发，计划使用一台装有 Microsoft Windows 8.1 的具有较强运算能力的 PC 完成。但除此之外，在人体模型的原始数据采集阶段还需要使用如下硬件设备：

Microsoft Kinect 传感器，见图 18。Kinect 是微软公司 2010 年推出的 XBOX360 游戏机的体感控制器，它允许人们不用触摸控制器即可进行游戏。它能够提供实时的彩色图像和深度图像，默认以 30 帧每秒的速度提供分辨率为 640×480 的彩色图像和 320×240 的深度图像数据。对于 Kinect for Windows 传感器而言，它的有效视距为 0.8 米到 4.0 米之间，最佳距离在 2.5 米左右，非常适合近距离的深度数据采集。利用这一特性，可以使用 Kinect 进行人体模型原始数据的采集。

图 18. Microsoft Kinect 传感器

旋转台。为了能够采集到人体模型全角度的数据，采集工作应将在旋转台上进行，使得 Kinect 能够获得多角度的深度数据流，再通过一定的配准操作，形成完整的人体模型表面点云，进而重建出三维人体模型。选择旋转台的时候主要考虑其转速和承重能力，转速不能过快，承重能力应在 60kg 以上。

6.2. 编程语言

本课题计划使用 C++编程语言。

C++是一种使用非常广泛的电脑程序设计语言。它是一种静态数据类型检查的，支持多范型的通用程序设计语言。C++支持过程化程序设计、数据抽象化、面向对象程序设计、泛型程序设计、基于原则设计等多种程序设计风格。很多开源类库包括 OpenNI、PCL 等均完整支持 C++，因此 C++非常适合用于本课题的相关任务，合理地运用各种类库也可以节省大量的开发时间。此外，C++程序的运行效率也比较高，适合三维运算等计算任务较为繁重的工作。

6.3. 集成开发环境

本课题计划使用 Microsoft Visual C++ 2010 作为集成开发环境。

Microsoft Visual C++（简称 Visual C++、MSVC、VC++或 VC）是微软公司的 C++开发工具，具有集成开发环境，可提供编辑 C 语言，C++以及 C++/CLI 等编程语言。VC++集成了便利的除错工具，同时集成了微软 Windows 视窗操作系统应用程序接口（Windows API）、三维动画 DirectX API 和 Microsoft .NET 框架。选择这款集成开发环境的原因在于它功能强大，比较适用于大规模项目的开发，同时也是 PCL 的推荐 IDE。选用 VC++ 2010 版本而非最新版本也是因为考虑到与 PCL 类库的兼容性。

6.4. 开发工具包、开源类库及 API 接口

本课题可能用到的开发工具包、开源类库及 API 接口包括：PCL、OpenNI、OpenGL、QT 等。

PCL（Point Cloud Library，点云库）是一个独立的大型的处理二维/三维图像和点云数据的开源工程，包含了许多先进算法，比如滤波、特征估计、表面重建、模型拟合和分割等。它可以与 OpenNI 紧密结合，从而直接从 Kinect 获取点云数据。此外 PCL 还包括 Kinfu，它是 Kinect Fusion 的开源实现，可以进行三维重建工作。

OpenNI（Open Natural Interaction，开放自然交互）是一个多语言、跨平台的框架，它定义了编写应用程序，并利用其自然交互的 API。OpenNI API 由一组可用来编写通用自然交互应用的接口组成。OpenNI 的主要目的是要形成一个标准的 API，来搭建视觉和音频传感器及视觉和音频感知中间件两方面之间通信的桥梁。其优点在于开源性，有着许多能实现各种不同功能的第三方插件。

OpenGL（Open Graphics Library，开放图形库）是一个定义了跨编程语言、跨平台的编程接口的规格，它主要用于三维图象（二维图像亦可）。OpenGL 是个专业的图形程序接口，是一个功能强大，调用方便的底层图形库。使用这一接口能够避免在项目中重新编写一些基本三维操作的实现，提高项目开发效率。

Qt 是一个跨平台的 C++应用程序开发框架。广泛用于开发 GUI 程序，这种情况下又被称为部件工具箱。也可用于开发非 GUI 程序，比如控制台工具和服务器。Qt 是自由且开放源代码的软件，使用标准的 C++和特殊的代码生成扩展以及一些宏。使用这个框架有利于我们在 C++环境下开发程序的图形用户界面。

7. 进度计划表

2014 年 3 月 1 日——2014 年 3 月 10 日：撰写项目开题报告；

2014 年 3 月 11 日——2014 年 3 月 20 日：三维人体模型的研究（可控参数的选择、模型表达方式的选择等）；

2014 年 3 月 21 日——2014 年 4 月 10 日：程序的编写和可视化（模型的导入、模型的变形和模型的显示）；

2014 年 4 月 11 日——2014 年 4 月 20 日：虚拟模型与实体模型的变形配合与相应控制程序的集成；

2014 年 4 月 21 日——2014 年 4 月 30 日：模型整体的调试；

2014 年 5 月 1 日后：准备展览与撰写设计报告。

本科毕业设计　设计比较

当前的虚拟试衣技术的竞争非常激烈，从许多服装大厂商到一些从事计算机图形学和机器视觉的研究机构再到对这一领域抱有浓厚兴趣的个人都在以不同的技术手段尝试解决这一问题，因此相关的设计可谓琳琅满目。这里仅挑选一些已经获得商业应用的、比较成熟的设计进行比较分析。

1. 虚拟试衣系统设计比较

1.1. 美国 Fitiquette 虚拟试衣间

Fitiquette 位于美国旧金山，专注于帮助在线服装销售商改善消费者的服装试穿体验，它希望通过它的服务彻底变革在线服装试穿的方式。

其基本操作流程是：先在网站给定的一系列体型各有不同的三维模特中挑选一个与自身体型最为接近的，之后再根据自己的实际特征尺寸，如三维、身高等，对三维模特进行进一步的微调，直到这一模型能够最大程度地贴合自身的体型特征。Fitiquette 会根据这一三维模特以一定的方式检索其数据库，推荐适合这一身材的服装，通常会列举出较多的不同尺寸的可能结果供消费者选择，消费者可以根据每一件服装的试穿效果进行抉择。当点击一个尺寸的服装，三位模特就将穿上这件服装，消费者就可以对这其进行观看，也可以进行旋转、缩放等操作，以看到全方位的试穿效果。

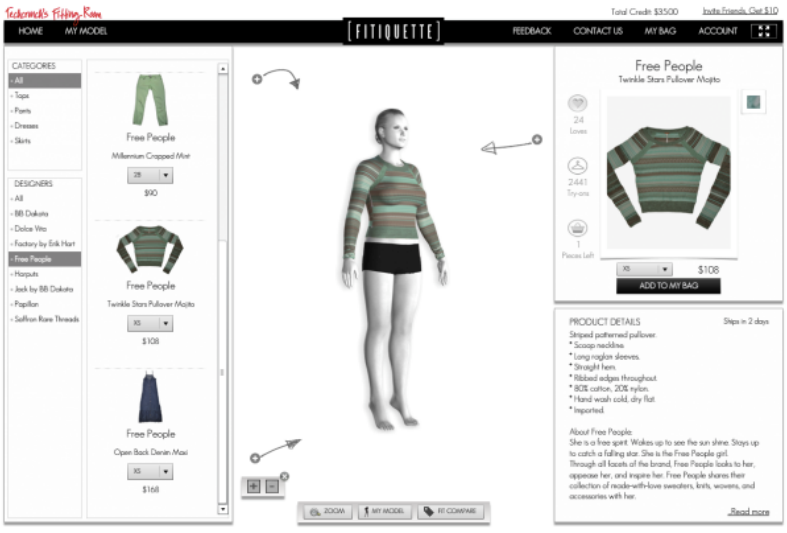

图 1. Fitiquette 虚拟试衣间

Fitiquette 所使用的虚拟试衣解决方案为全三维模拟，即模特和服装均为三维模型，如图 1。其服装的三维模型相对较为精细，能够很好地再现服装的纹理和褶皱效果。但这一做法也使这套技术的成本较高。

Fitiquette 服务在将来也可能能够为服装厂商优化生产业务提供参考。这项服务可以收集到消费者当前对流行服装的倾向数据，从而更好地指导服装厂商进行生产。服装厂商也可以提供服装的原型供消费者虚拟试穿，测试其款式是否受欢迎。

1.2. 瑞典 Clothes Horse 尺码推荐系统

Clothes Horse 尝试用“问答”的方式来解决消费者挑选服装尺码难的问题。这是一个基于数据挖掘技术的尺码推荐系统，它作为服装电商平台的插件而出现。

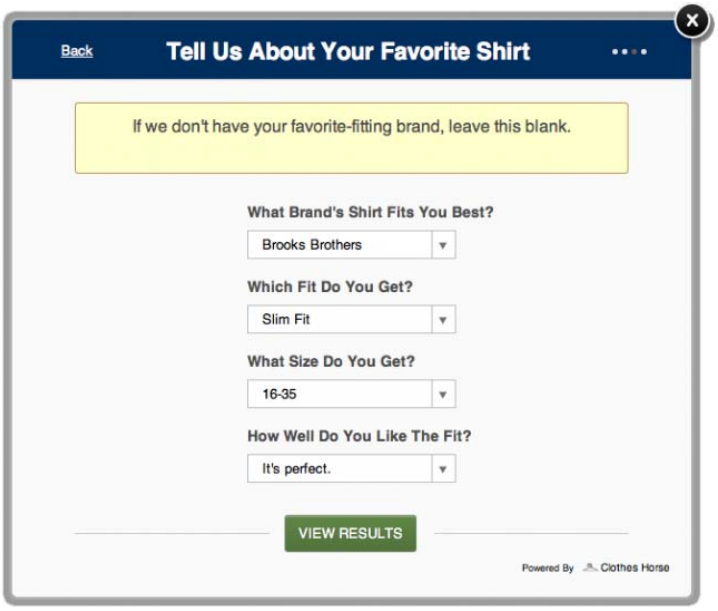

图 2. Clothes Horse 尺码推荐系统

消费者不确定某一服装的尺码是否合适自己的时候，可以方便地调用这一插件，通过回答数个简单的问题，就可以知道尺寸是否合适，如图 2。更重要的是，Clothes Horse 还会告诉消费者一些更为细节的信息，比如哪些部位合适、哪些部位较紧等等。对于向消费者提问的问题，不会涉及到任何需要实际测量才能获得的信息，而是包括“哪个品牌的衬衫最合身”、“购买的是哪种裁剪、哪种尺码”、“合身程度如何”这类问题。Clothes Horse 通过对后台庞大的数据库中的信息进行分析，给出相关的匹配结果。

Clothes Horse 的这一思路大大方便了消费者，因为基本不需要消费者进行额外的行动。但同时，它仅能给出一些描述性的建议，而不能生成直观的试衣效果。

1.3. 瑞典 Virtusize 可视化尺码比较插件

Virtusize 是一个为服装电商平台提供的插件，可以可视化地进行服装尺码比较。Virtusize 仅仅关注服装的尺码和合体性，而不考虑服装的款式设计等因素，其目标就是要成为服装电商的尺码标准平台。

图 3. Virtusize 可视化尺码比较插件

使用 Virtusize 插件的服装电商平台首先需要对所售衣服的详细尺寸数据进行采集，包括衣长、胸围、腰围、肩宽、袖长等等，将其存入后台数据库中。消费者在选购服装时，很有可能对服装的尺码选择没有把握，此时只需要自己测量一下自己所拥有的最合身的服装的相关数据，将其输入 Virtusize，它就能够以可视化的方式展现出两件服装的大小比较，如图 3。消费者因此可以判断哪个尺码才能最适合自己的身材。而当消费者注册了 Virtusize 的会员后，它可以记录消费者所拥有的服装尺码，消费者也就不需要再次测量并输入数据了。

Virtusize 的创始人们认为，衣服尺码的展示未必要使用三维模型，二维图形就可以有良好的效果，而三维模型则很有可能显示不出衣服的松紧。这种二维图形的展示方式，相比于其他基于三维模型的展示方式，无疑大大降低了成本；相比于问答式的尺码推荐系统，又显得更为直观。

2. 试衣机器人设计比较

2.1. 美国试衣机器人专利

由 Maarja Kruusmaa 等人持有的一个美国 2010 年的发明专利阐述了一种试衣机器人及相关服务体系的详情。

专利中描述了一种利用这一试衣机器人进行个性化裁剪的流程，如图 4。消费者首先进入一个三维身体扫描设备，由操作员或消费者自己操作设备获取三维图像，传给一台计算机进行一些处理。这些处理包括计算线性尺寸、表面积、形状和体积，还可能进行压缩以减小文件体积、进行面部模糊以保护消费者隐私或进行图像增强处理以使其更适宜观看。对应于消费者身材的数据通过网络传送到远端的另一台计算机，它可能位于服装厂商或裁缝店中。这台电脑连接着一个可调节的模特，即试衣机器人，它可以被精确的调节以代表消费者的身材，这种调节可能是半自动的或全自动的。服装设计师或裁缝利用这个模特进行个性化服装设计。试衣的过程通过相机进行记录，利用互联网传送到消费者或质量控制人员处以对个性化裁剪的质量进行监控。衣服和消费者的图像还可以进行组合以提供更为真实的试衣体验。

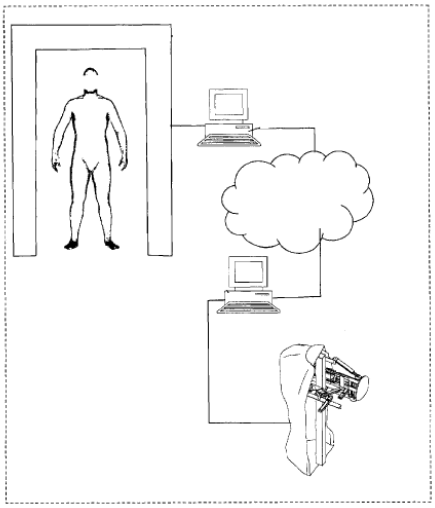

图 4. 个性化裁剪流程

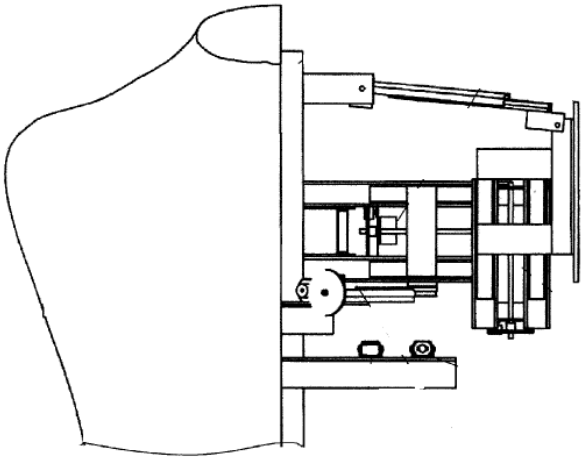

图 5. 试衣机器人机械结构

专利中描述的试衣机器人外表面覆盖物是完整的、连续的，它使用可拉伸且恢复能力强的材料制成。而机器人的变形是通过内部机械结构控制的支撑结构的变化来实现的。具体的机械结构比较复杂，如图 5，在此不做过多说明。

2.2. 爱沙尼亚 Fits.me 试衣机器人

Fits.me 认为，网络服装零售商要与实体服装店竞争，需要虚拟试衣技术的支持，因为网购者在购买之前无法试穿服装这一问题打消了很多人在网上购买服装的积极性。Fits.me 利用试衣机器人来模拟消费者的身材，通过其仿生学技术和科学算法，试衣机器人可以模拟出近一万种不同的体型。

Fits.me 的这款试衣机器人包含了 2000 多块不同的身体模块，这些身体模块间的组合与移动可以模拟人体的大多数身材特征，如图 6。通过试衣机器人的变形特点，首先在

服装生产商处采集各种服装穿着在各种体型机器人模特身上的照片，并存储在数据库中；当消费者需要网购服装时，在网站上输入自身的体型数据，Fits.me 会自动匹配最接近的试衣机器人体型，成为消费者对应的模特；随后消费者浏览服装时，则可以看到服装穿着在这一模特上的效果。

图 6. Fits.me 试衣机器人

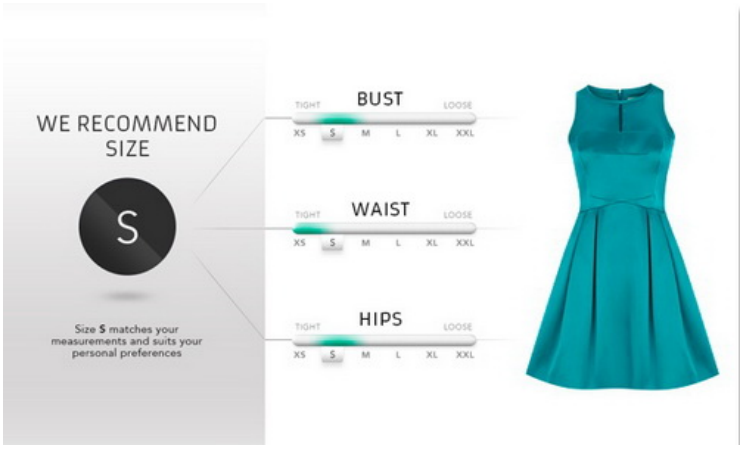

图 7. Fits.me 试衣顾问

Fits.me 目前提供两类服务：其一是虚拟试衣间服务，正如上文所述。这一服务相对较为复杂，需要用户输入较为详细的体型信息，同时也要求服装厂商为每款服装提供多组照片，这样才能获取比较理想的效果。但这种方法成本较高，对于服装厂商的负担较重。其二是试衣顾问服务，它让用户输入基本的体型数据，网站自动推荐合适的尺码，为用户选购服装提供参考，如图 7。这一服务较为方便快捷，但消费者无法观看到试穿服装的效果。

2.3. 中国 Fitsme.me 可调式三维试衣模特

Fitsme.me 是由北京旺商通商务咨询有限公司的姚建主要负责，并与清华医学院的人体仿真小组合作的项目。它把计算机图像学和人体仿真结合起来，为电子商务衣服销售商提供了网上试衣系统。

Fitsme.me 三维试衣间的实现机制如下：在消费者输入他们的身体尺寸后，能够看到服装的合适程度，就像在实际试衣间中一样。Fitsme.me 使用了可调式的服装模特模拟出了消费者的体型，并展示了消费者特定的体型之下针对某服装不同号码的试穿效果。这种试穿效果和客户实际试穿效果非常接近。

图 8. Fitsme.me 试衣机器人

Fitsme.me 的可调式三维试衣模特在设计中以国家颁布的 160/84A 号型标准为依据，实际测量人体各部位数据比对模型；进行验证调整，分析总结有效数据，体现我国人体体形特征，使模型覆盖面更大，准确性更高。它具有较高的标准值，比例均称，线条优美；采用断腰式设计，可变形范围为身高 150 厘米至 175 厘米和大部分正常体型；在一定程度上表现出肩胛骨、斜方肌、腹直肌、臀大肌等人体主要肌肉群。

3. 设计比较分析

从上面的设计比较中不难看出，当今主流的虚拟试衣系统之间的差别非常大，从文字式的到二维图形的再到三维模型的都有，说明了当今在虚拟试衣领域的几个主要发展方向和广阔的发展前景。这几种技术解决方案可以说各有各的长处与短处，但总的来说，消费者和服装厂商所付出的越多（包括消费者输入的数据和服装厂商需要预先采集的数据），服装试穿的展示效果越好，同时成本也就越大。因此，能够更好地调和这一对矛盾，从中找出最佳平衡点的方案将会带来巨大的影响。

而主流的试衣机器人之间也存在着差别，并没有形成统一的模式，比如表面覆盖层是分片处理还是采用柔性材料。不难看出，当前的试衣机器人解决方案还存在这研发、生产成本较高这一个非常关键的问题，在保证模拟效果的条件下压缩成本则成了研究的核心。

本课题的创新之处在于：首先，整个试衣机器人与虚拟试衣系统整合了当今的多钟虚拟试衣技术解决方案，起到了扬长避短的作用，在控制一定成本的条件下能够得到较好的试穿效果；其次，本系统非常注重人机交互的设计，这不仅包括服装设计师与试衣机器人控制软件之间的交互，也包括消费者与服装电商平台间的交互，以可视化的方法

优化用户体验，力求将用户的学习成本降至最低；此外，本系统在试衣机器人本身也做出了一些改进，例如在分块外壳的基础上包覆柔性材料以增加表面平滑度；最后，本课题还对服装合体性和商业模式进行了分析，从理论的层面为试衣机器人与虚拟试衣系统提供支持。

参考文献

[1] 朱德海. 点云库 PCL 学习教程 [M]. 北京: 北京航空航天大学出版社, 2012.

[2] 余涛. Kinect 应用开发实战：用最自然的方式与机器对话 [M]. 北京: 机械工业出版社, 2012.

[3] Kramer J, Burrus N, Herrera D C, et al. Hacking the Kinect [M]. New York City: Apress, 2012.

[4] 朱李丽, 邓中民, 李刚炎. 三维服装CAD中人体建模综述 [J]. 武汉科技学院学报, 2004, 01: 19-21.

[5] 秦可, 庄越挺, 吴飞. 服装CAD中三维人体模型的参数化研究 [J]. 计算机辅助设计与图形学学报, 2004, 07: 918-922.

[6] 闫迷军, 宁涛, 张兆璞. 服装 CAD 中三维人体尺寸信息提取技术的研究 [J]. 工程图学学报, 2005, 01: 26-29.

[7] 吴龙, 张欣, 任小玲, 李毅. 基于三维人体测量的参数化人台的研究 [J]. 西安工程科技学院学报, 2005, 04: 416-419.

[8] 王媚, 陆国栋, 张东亮. 服装CAD中三维人体建模技术的研究及应用 [J]. 工程图学学报, 2007, 01: 1-6.

[9] Wang C C L, Wang Y, Chang T K K, et al. Virtual human modeling from photographs for garment industry [J]. Computer-Aided Design, 2003, 35(6): 577-589.

[10] Kim S, Park C K. Parametric body model generation for garment drape simulation [J]. Fibers and Polymers, 2004, 5(1): 12-18.

[11] Wang C C L. Parameterization and parametric design of mannequins [J]. Computer-Aided Design, 2005, 37(1): 83-98.

[12] Baek S Y, Lee K. Parametric human body shape modeling framework for human-centered product design [J]. Computer-Aided Design, 2012, 44(1): 56-67.

[13] 王媚. 面向服装 CAD 的三维人体建模与变形技术研究及实现 [D]. 浙江大学, 2006.

[14] 崔树芹. 三维虚拟试衣系统中参数化人体建模技术的研究 [D]. 华中科技大学, 2006.

[15] 宋诗超. 基于 Kinect 的三维人体建模与测量的研究 [D]. 东华大学, 2013.

[16] 中商情报网. 2013 年中国服装网购市场现状分析 [EB/OL]. http://www.askci.com/news/201310/28/2811321134242.shtml, 2013-10-28.

[17] Wikipedia. Virtual Dressing Room [EB/OL]. http://en.wikipedia.org/wiki/Virtual_dressing_room.

纤维与聚合物 2004，第 5 卷，第 1 册，12-18

用于服装悬垂仿真的参数化人体模型的生成

Sungmin Kim[1] 和 Chang Kyu Park*

纺织工程系，建国大学，首尔 143-702，韩国
[1] 应用化学工程系，国立全南大学，光州 500-757，韩国
（收稿 2003 年 10 月 8 日；修回 2004 年 1 月 7 日；接收 2004 年 1 月 13 日）

摘要：一个参数化人体模型生成系统被开发出了。利用这一系统的多种数学和几何算法，一个三维扫描人体可被转换为一个可变大小的人体模型。一旦参数化人体模型生成后，它的大小和形状都可以通过提供合适的人体测量数据来随时改变。为了促进后续的用于服装悬垂仿真的图案排布过程，本研究中开发了一种边界盒生成算法。模型也可以转换为一系列参数化表面，它们可被用于三维服装图案设计系统。

关键词：参数化人体模型，参数化表面模型，悬垂仿真，自动化图案排布，三维图案设计

前言

近来计算机辅助设计（CAD）和制造（CAM）技术的应用已经在许多工业领域成为了一种明显的趋势[1]。对于服装制造业来说，已经通过利用二维 CAD 系统实现了很大程度的自动化[2,3]。然而，最近一种根本性的创新发生了，使得这种二维 CAD 系统稳步演变为三维 CAD 系统[4-7]。许多研究者从事于三维服装 CAD 系统的开发，包括基于织物行为力学分析的织物悬垂仿真系统，以及由三维人体模型的三维直接图案生成[8-13]。开发一种集成三维 CAD 系统所需要的最重要的特征之一是设计一个有功能的人体模型。通常，一个简单的三维对象建模技术已被用于生成人体模型。然而，现在直接从三维人体扫描数据为人体建模是可能的，因为由于非接触式测量技术的进步，全身扫描仪变得能够以一个可接受的成本得到了[14-17]。然而这种扫描的人体模型存在一些问题，每当需要一个不同尺寸的人体模型时，就必须要扫描一个新的模型，尽管每[1]个人体的形状并没有太大差异。另一个问题是它不适用于需要大量碰撞处理的服装悬垂仿真，因为这种模型的数据大小通常非常巨大。第三个问题是由于扫描的人体模型是由一系列的三角形元素构成的，无法轻易地将包括服装图案在内的任意形状绘制到上面。

在本研究中，开发了一个参数化的人体模型生成系统来克服这些困难。这种参数化人体模型相比静态人体模型来说具有许多优势。一是模型的形状和尺寸可以轻松改变，只要需要的新模型的整体形状与原模型的没有太大差异，就不需要进行再次扫描。二是模型的数据大小极大地缩减了，使它变得适用于服装悬垂仿真。三是由于人体模型的表面可以转变为一系列参数化表面，任意形状均可以轻松地被绘制到上面用于三维服装图案的生成。还有一个优势在于围绕人体模型可以定义出最优边界盒，它们可用于服装悬垂仿真系统的自动化空间图案排布功能。边界盒的使用对于服装悬垂仿真具有惊人的效果，因为围绕人体模型的图案块空间排布是最为耗时和依赖人工的步骤。如上所述，本研究中开发的参数化人体模型可以应用于包括三维服装 CAD 系统在内的许多领域。

3D 人体数据的准备

由于三维全身扫描仪变得可以以一个可负担的成本获得，近来基于三维定量人体数据的研究也变得越来越普遍。这类三维扫描人体模型被用于服装悬垂仿真领域。然而，利用扫描的原始人体模型存在着一些问题。首先，因为模型的数据大小较大，并不是十分有效率。其次，每当需要新模型的时候，必须扫描一个新的模型。因此，一个可变大小的模型是必须的，它最初由三维人体数据生成，进而可以简便地改变形状和大小。在本研究中，基于利用通常的全身扫描仪得到的三维人体数据生成了一个参

*通讯作者：cezar@konkuk.ac.kr

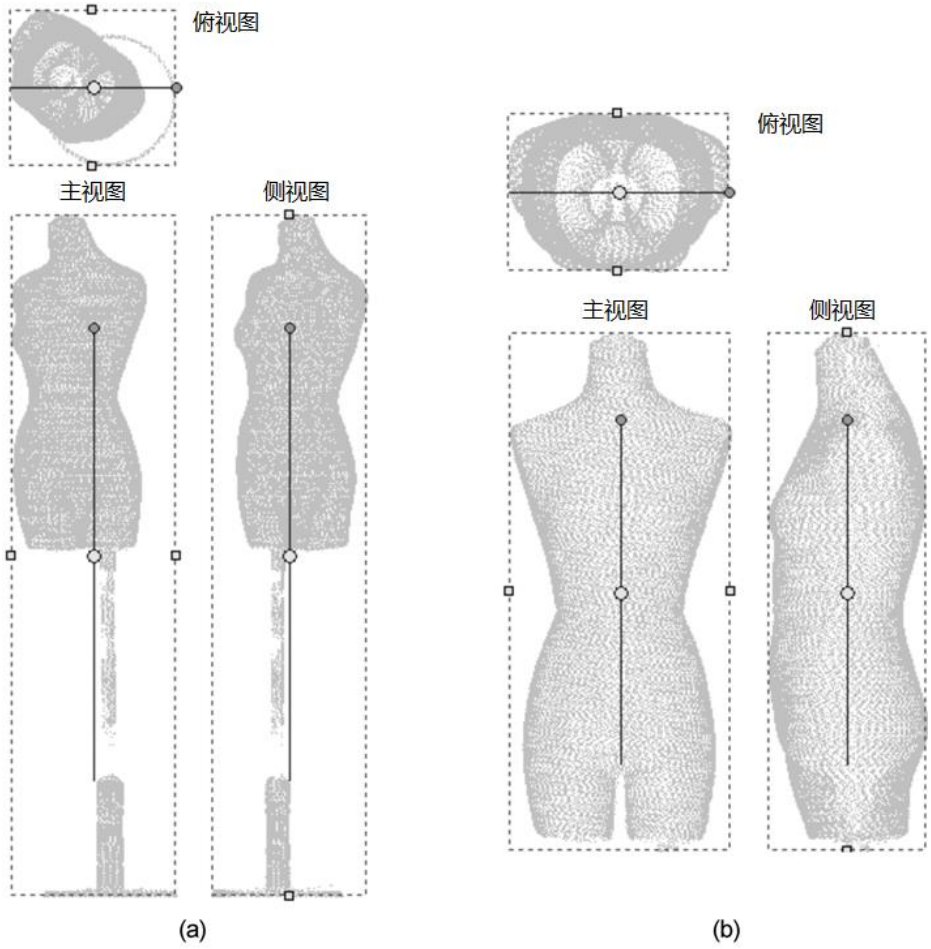

图 1. 三维人体数据操作的例子：(a)初始的人体模型和(b)对齐并裁剪过的人体模型。

数化的人体模型。扫描仪的类型在本研究中并不重要，只要它能够支持“*.iv”数据格式，这是三维对象最有名的数据类型之一。

当扫描人台而不是真实人体的时候，一些无用的部分，比如支架，可能也会被包含在内，如图 1(a)所示。此外，扫描的模型可能由于某些原因并没有恰好对齐主轴。在本研究中，用户可以通过轴控制重新对齐模型，并使用一个直观的界面系统轻松地裁剪掉模型的无用部分，如图 1(b)所示。

参数化人体模型的形成

人体截面的提取

参数化人体模型是利用一些特定的三维人体数据的横截面形状来生成的。利用这种方法，模型的形状和大小可以通过改变每个横截面的形状或改变每个横截面的垂直位置来更改。在本研究中，位于颈部顶端、颈部底端、肩部、臂孔、腋窝、胸点、腰部、腹部、臀部和股部的横截面被用于模型生成。每个横截面的位置和水平、垂直倾角可以利用一个简单的界面系统来定义，如图 2 所示。每个横截面的初始位置由对应的统计性人体测量数据得到。

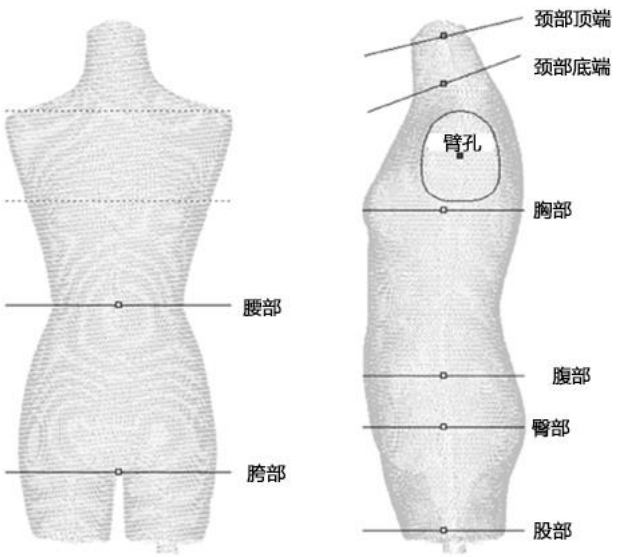

图 2. 人体横截面的定义。

一旦所有横截面的位置和倾角被定义后，每个横截面的形状可从人体模型中提取出来。目标截面的平面方程可被表述为公式(1)，其中截面的法向量和高度分别是 v 和 h。

$$v_x x + v_y y + v_z z = h \quad (1)$$

由于人体模型是由三角形元素构成的，每个截面可被定义为平面和这些元素每个边的一系列截交点。假设一个元素一个边的两个端点是 $p_1(x_1,y_1,z_1)$和 $p_2(x_2,y_2,z_2)$，仅当这两个端点满足公式(2)时这个边与平面才有一个截交点。

$$\left(v_x x_1 + v_y y_1 + v_z z_1 - h\right) \times \left(v_x x_2 + v_y y_2 + v_z z_2 - h\right) < 0 \quad (2)$$

在这种情况下，兴趣点 I 的坐标可由公式(3)得到。

$$p = \overrightarrow{p_1 p_2} = (x_2 - x_1, y_2 - y_1, z_2 - z_1)$$

$$k = \frac{v \cdot p_1 - h}{v \cdot p}$$

$$I_x = kp_x + x_1, I_y = kp_y + y_1, I_z = kp_z + z_1 \quad (3)$$

利用这种方法提取横截面的一些例子如图 3 所示。

横截面的圆弧拟合

如图 3 所示，每个横截面都是任意地分散的。每个横截面的形状都可以大致看作闭合环，每个点的位置可以由两个参数进行描述，一个是这个点到截面中心的距离 r，另一个是 x 轴和截面中心与该点之间连线的夹角 θ。一旦将所有点关于它们的 θ 值排序后，θ 和 r 之间的关系可以利用如公式(4)所示的傅里叶级数展开进行近似。利用这种方法，每个截面的形状可以通过单个逆时针方向旋转的连续曲线进行拟合，并且点 $P(\theta)$的位置可利用公式(4)得到。

$$a_n \cong \frac{1}{n\pi}\left[-\sum_{s=1}^{m} j_s \sin\frac{n\pi}{L}\theta_s - \frac{L}{n\pi}\sum_{s=1}^{m} j'_s \cos\frac{n\pi}{L}\theta_s\right]$$

$$b_n \cong \frac{1}{n\pi}\left[\sum_{s=1}^{m} j_s \cos\frac{n\pi}{L}\theta_s - \frac{L}{n\pi}\sum_{s=1}^{m} j'_s \sin\frac{n\pi}{L}\theta_s\right]$$

$$j_i = \frac{r_{i+1} - r_{i-1}}{2}, j'_i = \frac{j_{i+1} - j_{i-1}}{2}$$

$$r(\theta) = r_0 + \sum_{s=1}^{m}(a_i \cos i\theta + b_i \sin i\theta)$$

$$P(\theta) = (r(\theta)\cos\theta, r(\theta)\sin\theta) \quad (4)$$

其中，n 是傅里叶级数的项数，m 是点数，j 是主要跳跃，j'是次要（差分）跳跃。

对于一个人体模型来说有多种横截面形状，如图 3 所示。然而，大多数服装的形状是凸的，以至于服装通常能够覆盖住身体的凹面，除了一些需要精确拟合的特殊服装。因此，期望的模型的最终形状取决于每个横截面的曲线拟合方法。利用这种方法，使用有名的格雷厄姆凸包生成方法进行凸曲线拟合，圆弧拟合的结果如图 4 所示。

人体模型的变形

在本研究中，八个主要横截面和一些辅助横截面被用于人体模型的定义，且模型的形状和大小可通过考虑最终所需的几何形状改变每个截面的大小和位置来更改。每个身体部分都由扫掠这些横截面来生成。同样地，臂孔曲线的形状和几个参数被用于定义袖子，肩部角度以及肘部角度可被自由控制。一些主要界标自动生成于各自的横截面上。可控人

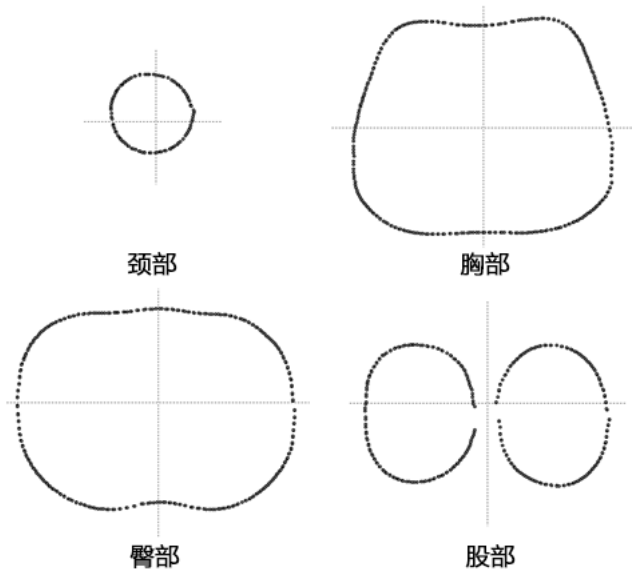

图 3. 提取的横截面形状的例子。

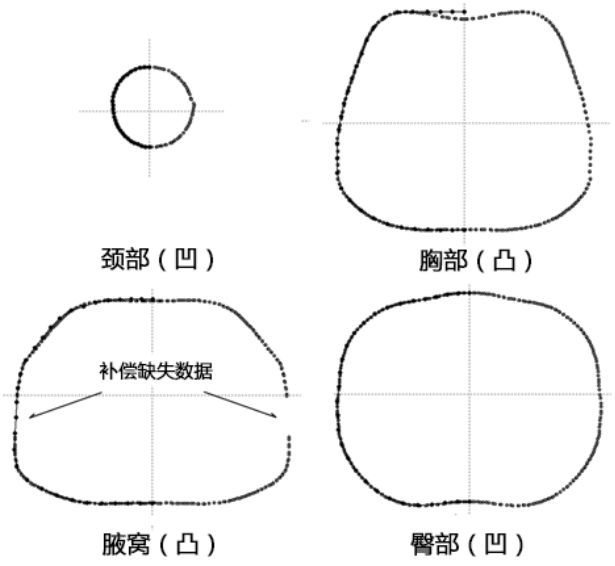

图 4. 选取的横截面圆弧拟合的例子。

表 1. 参数化人体模型的可控参数

围长/长度	高度	袖子
颈部	颈部	长度
肩部	肩部	肩部角度
胸部	腋窝	肘部角度
腰部	胸部	袖口细化比
腹部	腹部	
臀部	臀部	
股部	胯部	
腿长	股部	

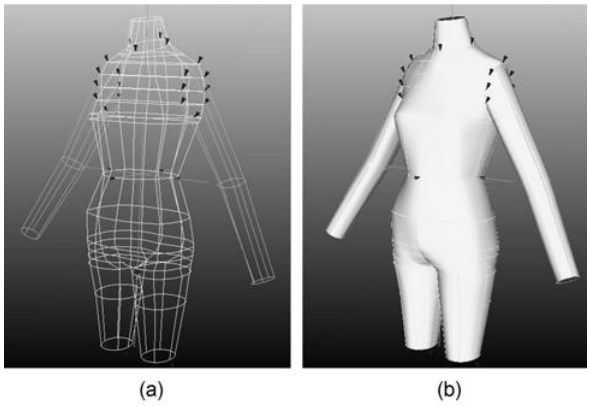

图 5. 参数化人体模型的例子：(a)结构框架和(b)贴面模型。

体测量参数列在表 1 中，参数化人体的结构框架和贴面模型如图 5 所示。

参数化表面的生成

近来，服装 CAD 系统的主流正从传统二维系统迁移至三维系统。为了从三维人体直接设计服装图案、要求有着多样身体信息的人体模型具有一个特别的特征。任意形状包括服装图案可以被轻松地绘制到它的表面。至于如图 5 所示的模型，这是不可能的，因为它的表面是由相互独立的三角形元素构成的。因此，在本研究中实现了一个参数化表面生成算法来创造一个有功能的表面模型，它可被用于服装悬垂仿真以及三维图案设计。在研究中使用的参数化表面是贝塞尔表面，它上面的任何点都可以用两个参数 u 和 v 轻松地引用，两个参数的范围均为 0 到 1。由于身体的每个部分都可以被大致近似为圆柱，参数 u 可以沿着它的圆周方向定义，而参数 v 则沿着它的轴向定义。由于每个横截面都是用连续曲线拟合的，参数化表面的控制点可被轻松地定义。参数化表面模型的一个例子如图 6(a)所示，图

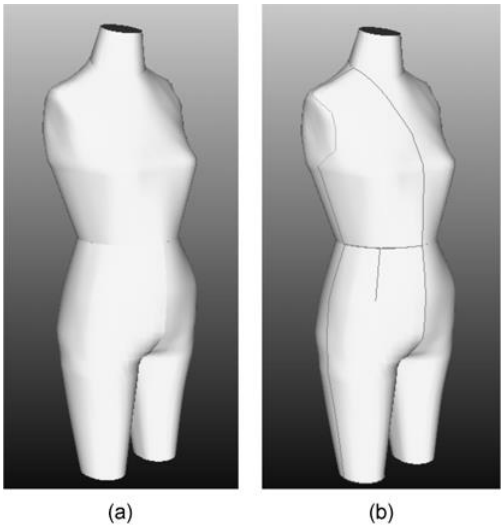

图 6. 参数化表面模型的例子：(a)参数化表面模型和(b)紧身胸衣和宽松长裤的图案绘制例子。

案绘制的一个例子如图 6(b)所示。如图 6 所示，这样的参数化表面人体模型可用于三维图案设计过程，这是我们的另一个研究主题。

边界盒的定义

本研究的主要目标是开发一个可变形的参数化人体模型以用于三维服装图案设计和服装悬垂仿真。特别是，服装悬垂仿真中的最困难的问题之一就是围绕身体的大量图案块的合适的空间排布。然而，对于用户来说仅仅使用二维输入设备排布这些图案是一个非常困难且费时的工作。因此，这个过程的自动化对于开发一个高效的服装悬垂仿真系统是不可避免的。在本研究中，开发了一个算法来生成圆柱卷，它们紧紧地覆盖住每个身体部分，称为包围盒。一旦为每个身体部分定义了所有的包围盒，它们可以被展成二维平面，因为它们均为可展面。接着用户可以轻松地关于这些盒排布图案块，这些图

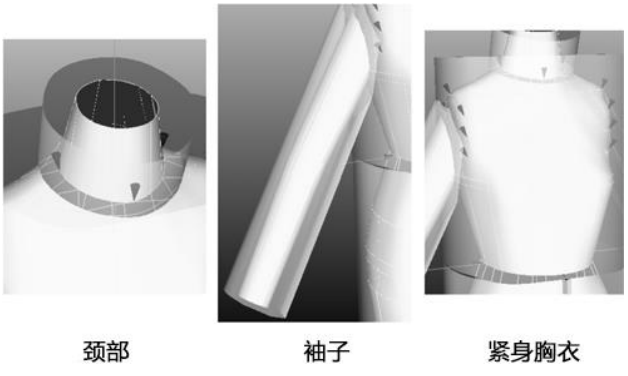

图 7. 不同身体部分的包围盒的例子。

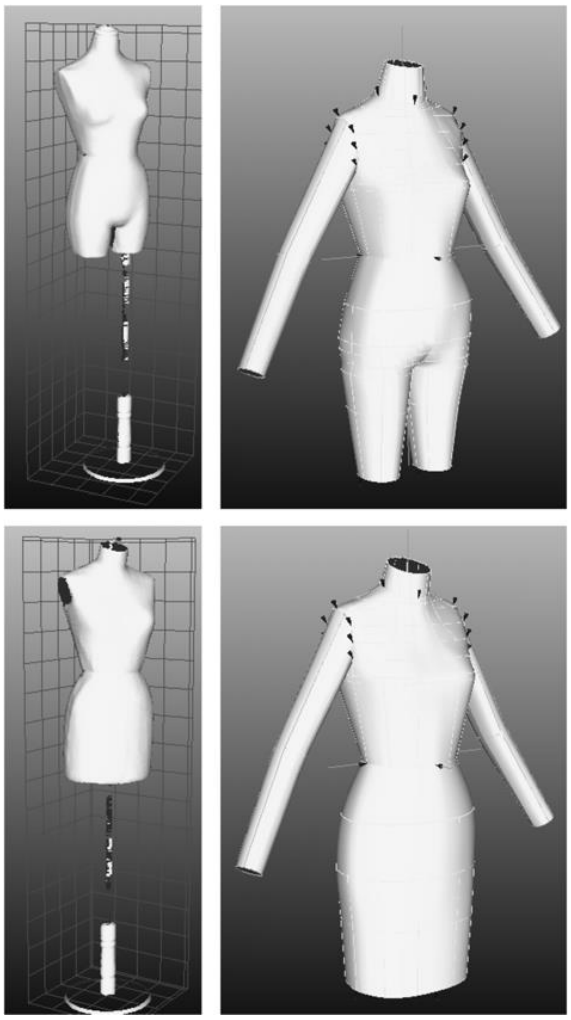

图 8. 原始扫描人体和重建的参数化模型的例子。

案块将通过包围盒的反向映射在空间中排布为原来的三维形态。利用这种方法，大量的图案块可被自动地、合适地排布，而不会对下层的人体模型有任何干扰，这样一来悬垂方针的整体耗时可被极大地减少。为每个身体部分定义的包围盒的例子如图 7 所示。

结果和讨论

三维扫描人体模型和它们对应的参数化人体模型的例子如图 8 所示。由于参数化人体的形状和大小可以在保持原模型整体形状的同时进行改变，服装制造厂商可以通过扫描并转换尽可能多的标准人体模型来保留自己的模型数据库。

如图 9 所示，一个参数化模型可以具有多种形状，并且这些变化可以由一个通用奔腾 4 1.3GHz 个人电脑系统通过快速计算算法进行实时确认。因此，某个参数对人体模型的形状和大小的效果可被快速分析。

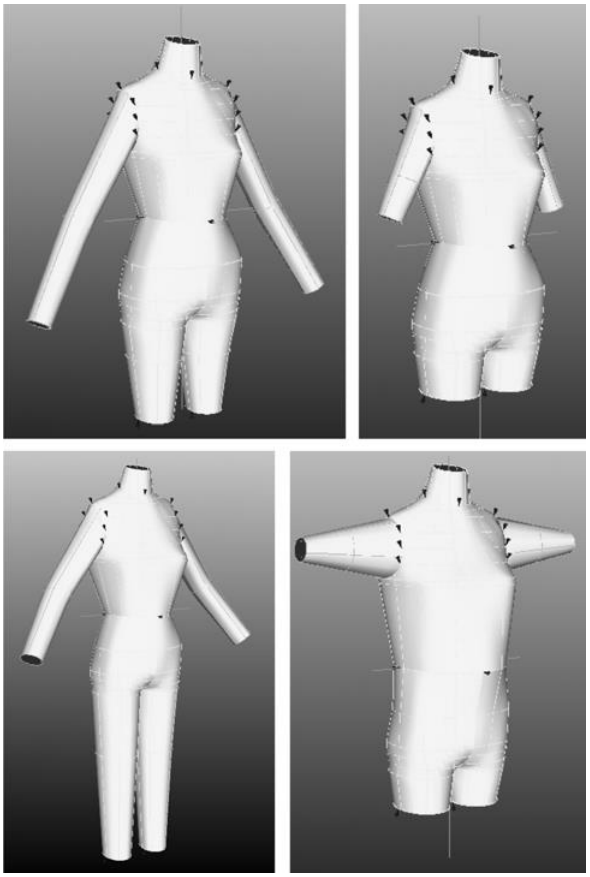

图 9. 参数化人体模型的变形。

包围盒可被自动生成，如图 10(a)所示。用户可以轻松地关于展开的包围盒排布大量的图案块，如图 10(b)所示，并且图案块将通过包围盒的反向映射被围绕身体排布，如图 10(c)所示。包围盒的形状和大小也会根据下层人体模型的变化而实时变化，改变可以立即被确认。

结论

一个参数化人体模型生成系统被开发出了。为此，由三维人体扫描数据利用多种数学和几何算法生成了一个可变大小的人体模型。人体模型的形状和大小可通过一个直观的用户界面进行轻松地、立即地更改，因而一旦参数化人体模型形成后，任何类型的人体模型都可以被轻松地生成。此外人体模型的表面可以被转换为一系列参数化表面。任意形状都可以被轻松地绘制在上面，因此它可被用于三维图案设计系统。最终，为人体模型定义了紧密拟合的包围盒，它将被用在服装悬垂仿真系统的自动化图案块空间排布中。参数化人体模型在服装 CAD 系统中具有多种应用领域，并且如果进一步的研究可以改善这个有功能的人体模型的话，它将对形成一个真正实用的系统很有帮助。

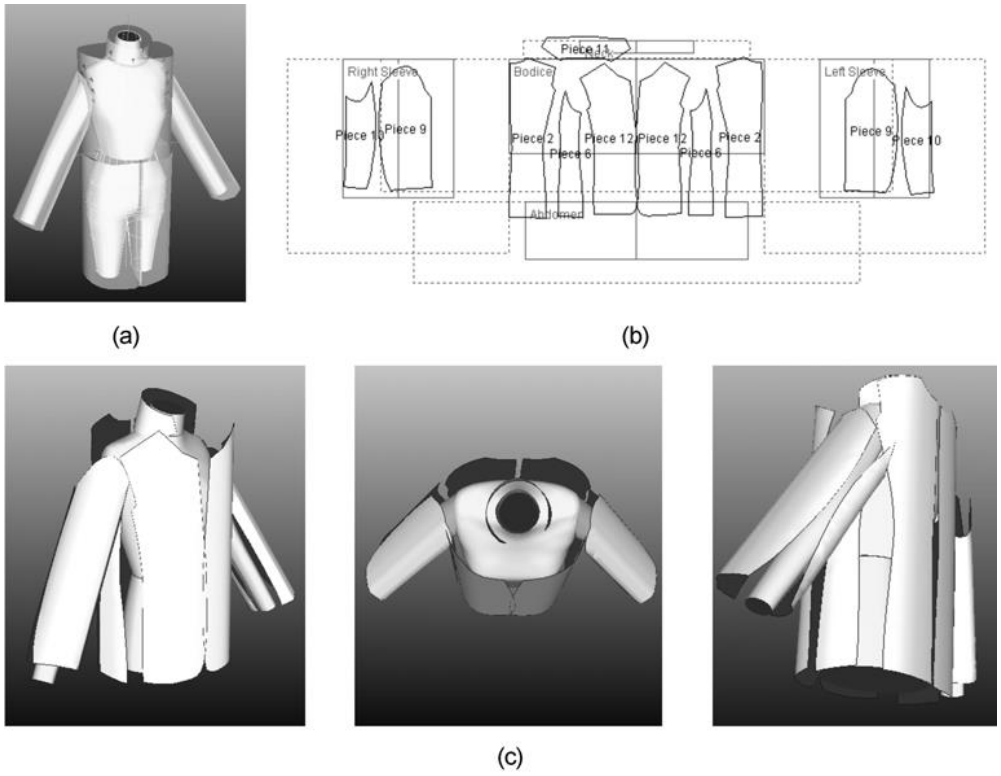

图 10. 利用包围盒进行空间图案排布的示意图：(a)围绕每个身体部分定义的包围盒，(b)关于展开的包围盒的图案块排布，和(c)展开的包围盒的反向映射用于空间图案排布。

参考文献

1. K. Lee, “Principles of CAD/CAM/CAE Systems”, Addison-Wesley, Boston, 1999.
2. B. J. Collier and J. R. Collier, *Cloth. Text. Res. J.*, **8**, 7 (1990).
3. W. Aldrich, “CAD in Clothing and Textiles”, Oxford BSP Professional Books, London, 1992.
4. H. Okabe, H. Imaoka, T. Tomiha, and H. Niwaya, *Comput. Graphics (Proc. SIGGRAPH 92)*, **26**, 105 (1992).
5. B. K. Hinds, J. McCartney, C. Hadden, and J. Diamond, *Int. J. Cloth. Sci. and Tech.*, **4**, 6 (1992).
6. Z. Stjepanovic, *Int. J. Cloth. Sci. and Tech.*, **7**, 81 (1995).
7. J. P. Turner, *Int. J. Cloth. Sci. and Tech.*, **6**, 28 (1994).
8. P. Volino and N. M. Thalmann, *VSMM '97 Proc., Int. Conf.*, Geneva, 109 (1997).
9. K. H. Roediger, *Proc. Comput. Graph. Int.*, Hanover, 396 (1998).
10. J. McCartney, B. K. Hinds, B. L. Seow, and D. Gong, *J. Mater. Process Tech.*, **107**, 31 (2000).
11. F. L. Heisey, P. Brown, and R. F. Johnson, *Textile Res. J.*, **60**, 690 (1990).
12. F. L. Heisey, P. Brown, and R. F. Johnson, *Textile Res. J.*, **60**, 731 (1990).
13. J. McCartney, B. K. Hinds, and B. L. Seow, *Comput. Aided Design*, **31**, 249 (1999).
14. S. Paquette, *IEEE Comput. Graphics and Appl.*, **16**, 11 (1996).
15. H. A. M. Daanen and G. J. V. Water, *Displays*, **19**, 111 (1999).
16. F. Arlt and A. Marach, *Int. J. Ind. Ergonom.*, **22**, 333 (1998).
17. C. K. Au and M. M. F. Yuen, *Comput. Aided Design*, **31**, 751 (1999).

Fibers and Polymers 2004, Vol.5, No.1, 12-18

Parametric Body Model Generation for Garment Drape Simulation

Sungmin Kim[1] and Chang Kyu Park*

Department of Textile Engineering, Konkuk University, Seoul 143-702, Korea
[1]*Faculty of Applied Chemical Engineering, Chonnam National University, Gwangju 500-757, Korea*
(Received October 8, 2003; Revised January 7, 2004; Accepted January 13, 2004)

Abstract: A parametric body model generation system has been developed. Using various mathematic and geometric algorithms of this system, a three-dimensionally scanned human body can be converted into a resizable body model. Once a parametric body model is formed, its size and shape can be modified instantaneously by providing appropriate anthropometric data. To facilitate the subsequent pattern arrangement process for garment drape simulation, a bounding box generation algorithm has been developed in this study. Also the model can be converted into a set of parametric surfaces that it can also be used for three-dimensional garment pattern design system.

Keywords: Parametric body model, Parametric surface model, Drape simulation, Automatic pattern arrangement, Three-dimensional pattern design

Introduction

The applications of computer aided design (CAD) and manufacturing (CAM) technology has become an obvious trend in many fields of industry recently [1]. For the garment manufacturing industry, a significant amount of automation has been achieved mainly by using two-dimensional CAD systems [2,3]. However, a radical innovation is taking place recently that such two-dimensional CAD systems are evolving steadily into three-dimensional ones [4-7]. Many researchers have worked on the development of three-dimensional garment CAD systems including the fabric drape simulation system based on the mechanical analysis of fabric behavior, and three-dimensional direct pattern generation from three-dimensional human body models [8-13]. One of the most important features needed for the development of an integrated three-dimensional CAD system is to design a functional human body model. Usually, a simple three-dimensional object modeling technique has been applied to generate a human body model. However, it is possible nowadays to model a human body directly from the three-dimensional body scan data because whole body scanners become affordable at an acceptable cost thanks to the advances in non-contact measurement technology [14-17]. However there are some problems in such a scanned body model that a new model must be scanned whenever a different sized human body model is required, although the shape of each human body does not differ largely. Another problem is that it is unsuitable for the garment drape simulation where a large amount of collision treatment is required, because the data size of such a model is usually very large. The third problem is that as the scanned body model is composed of a series of triangular elements, it is impossible to draw arbitrary shapes including garment patterns on it easily.

In this study, a parametric body model generation system has been developed to overcome such difficulties. The parametric body model has many advantages over a static body model. One is that the shape and size of the model can be changed easily that no more scanning is necessary as long as the overall shape of a newly required model does not differ largely from that of the original one. Another is that the data size of the model is significantly reduced so that it became suitable for the use in the garment drape simulation. The third is that as the surface of body model may be turned into a set of parametric surfaces, arbitrary shapes can be drawn easily on it for the generation of three-dimensional garment patterns. One more advantage is that the optimum bounding boxes can be defined around the body model that can be used for the automatic spatial pattern arrangement function of the garment drape simulation system. The use of bounding boxes has a striking effect for the garment drape simulation because the spatial arrangement of the pattern pieces around the body model is one of the most time consuming and human dependent procedures. As described above, the parametric body model developed in this study can be applied for many fields including the three-dimensional garment CAD systems.

Preparation of 3D Body Data

As the three-dimensional whole body scanners become available at an affordable cost, researches based on the three-dimensional quantitative human body data also have become more and more popular recently. The three-dimensionally scanned body models are used in the field of garment drape simulation. However, there are some problems in using the scanned raw body model. Firstly it is not so efficient because of the large data size of the models. Secondly, a new body model must be scanned whenever a new model is needed. Therefore, a resizable model is required, which is initially generated from the three-dimensional body data and further

*Corresponding author: cezar@konkuk.ac.kr

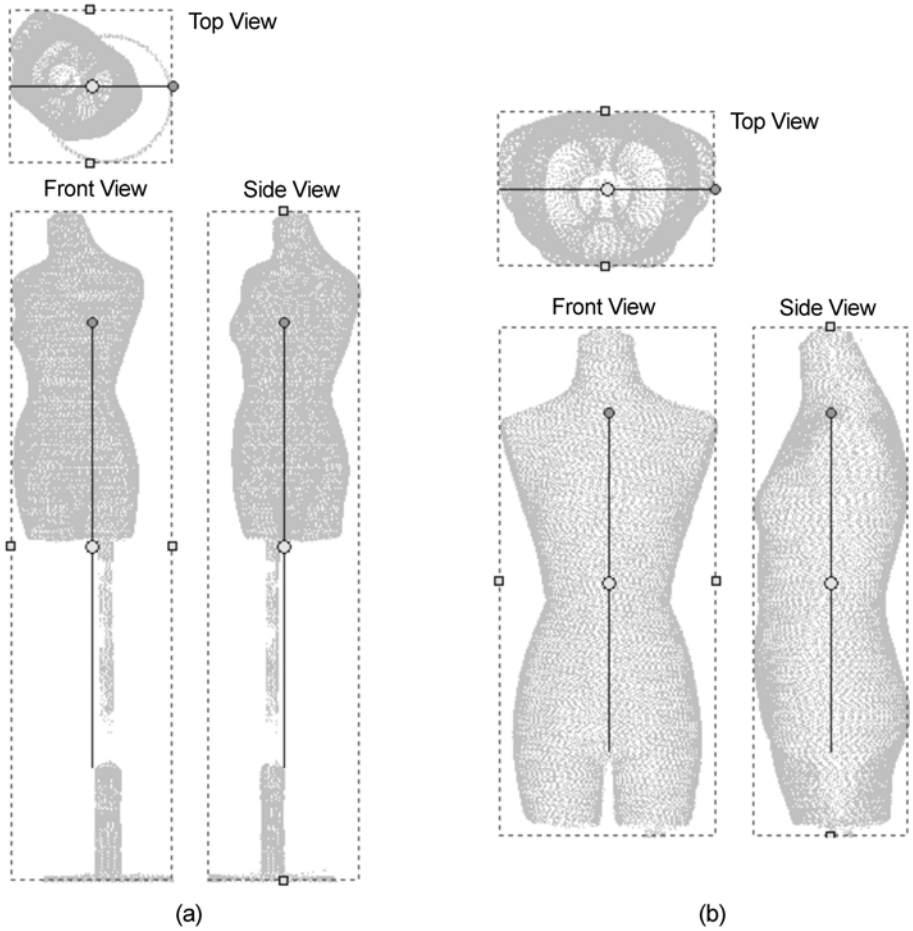

Figure 1. Example of three-dimensional body data manipulation: (a) Initial body model and (b) Aligned and cropped body model.

can easily be reshaped and resized. In this study, a parametric body model was generated based on the three-dimensional body data obtained using general whole body scanners. The type of scanner does not matter in this study as long as the scanner supports the '*.iv' data format which is one of the most famous data types for three-dimensional objects.

When scanning a mannequin instead of a real human body, some useless parts like a stand may also be included as shown in Figure 1(a). Also the scanned model may not be properly aligned along principle axes for some reasons. In this study, user can realign the model through axis control and crop the useless portion of the model easily using an intuitive interface system as shown in Figure 1(b).

Formation of Parametric Body Model

Extraction of Body Cross Sections

A parametric body model is generated using the shapes of some specific cross sections of the three-dimensional body data. With this method, the shape and size of the model can be changed either by reshaping each cross section or by changing the vertical location of each cross section. In this study, the cross sections at neck top, neck bottom, shoulder, armhole, armpit, bust point, waist, abdomen, hip, and thigh are used for model generation. The location and horizontal

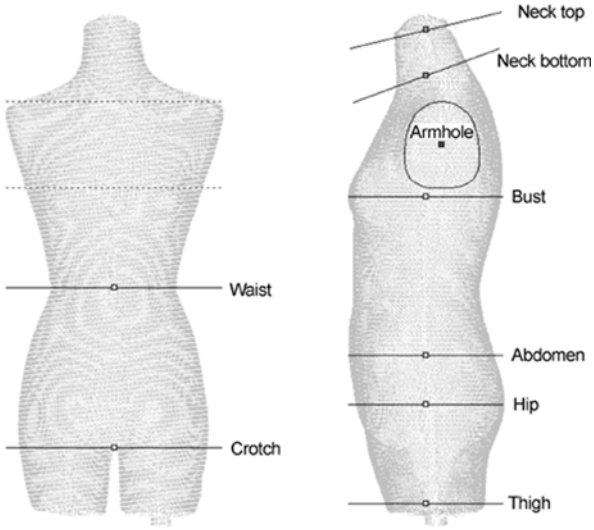

Figure 2. Definition of body cross section.

inclination of each cross section can be defined using a simple interface system as shown in Figure 2. The initial location of each cross section is obtained from respective statistical anthropometric data.

Once all the locations and inclinations of cross sections are defined, the shape of each section can be extracted from the body model. The plane equation for target section can be described as equation (1) when the normal vector and height of the section is v and h, respectively.

$$v_x x + v_y y + v_z z = h \tag{1}$$

As the body model is composed of triangular elements, each section can be defined as the set of intersecting points between the plane and each side of elements. Assuming that the two end points of a side of an element to be $p_1(x_1, y_1, z_1)$ and $p_2(x_2, y_2, z_2)$, the side has an intersecting point on the plane if the two points satisfy equation (2).

$$(v_x x_1 + v_y y_1 + v_z z_1 - h) \times (v_x x_2 + v_y y_2 + v_z z_2 - h) < 0 \tag{2}$$

In this case, the coordinates of the intersecting point I can be obtained using equation (3).

$$p = \overrightarrow{p_1 p_2} = (x_2 - x_1, y_2 - y_1, z_2 - z_1),\ k = \frac{v \cdot p_1 - h}{v \cdot p}$$
$$I_x = kp_x + x_1, I_y = kp_y + y_1, I_z = kp_z + z_1, \tag{3}$$

Some examples of cross sections extracted using this method are shown in Figure 3.

Circular Fitting of Cross Sections

As shown in Figure 3, each cross section has arbitrarily dispersed. The shape of each cross section can roughly be considered as a closed loop and the position of each point can be described with two parameters, one is the distance r between the point and section center and the other is the angle θ between the x-axis and the line connecting the section center and the point. Once all the points are sorted with respect to their θs, the relationship between θ and r can be approximated using Fourier series expansion as shown in equation (4). With this method, the shape of each section can be fitted through a single continuous curve rotating in counterclockwise direction and the position of a point $P(\theta)$ can be obtained using equation (4).

$$a_n \cong \frac{1}{n\pi}\left[-\sum_{s=1}^{m} j_s \sin\frac{n\pi}{L}\theta_s - \frac{L}{n\pi}\sum_{s=1}^{m} j_s' \cos\frac{n\pi}{L}\theta_s \right]$$
$$b_n \cong \frac{1}{n\pi}\left[\sum_{s=1}^{m} j_s \cos\frac{n\pi}{L}\theta_s - \frac{L}{n\pi}\sum_{s=1}^{m} j_s' \sin\frac{n\pi}{L}\theta_s \right]$$
$$j_i = \frac{r_{i+1} - r_{i-1}}{2}, \quad j_i' = \frac{j_{i+1} - j_{i-1}}{2} \tag{4}$$
$$r(\theta) = r_0 + \sum_{s=1}^{m} (a_i \cos i\theta + b_i \sin i\theta),$$
$$P(\theta) = (r(\theta)\cos\theta, r(\theta)\cos\theta)$$

where, n is the number of terms in Fourier series, m is the number of points, j is the primary jump, and j' is the secondary (differential) jump.

There is a variety of cross sectional shapes for a body model as shown in Figure 3. However, the shapes of most garments are convex so that the garment may generally cover the concavities of the body, except for some special garments where the exact fitting is required. Therefore, the final shape of desired model depends on the curve fitting method for each cross section. With this method, convex

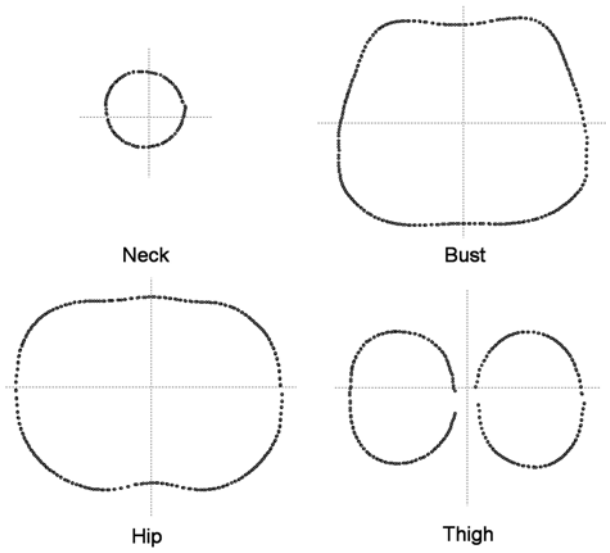

Figure 3. Examples of the shapes of extracted cross sections.

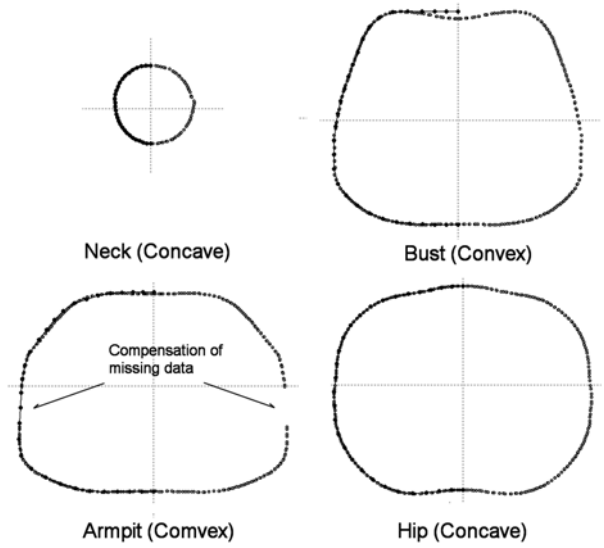

Figure 4. Examples of the circular fitting for selected cross sections.

Table 1. Controllable parameters for a parametric body model

Girth/Length	Height	Sleeve
Neck	Neck	Length
Shoulder	Shoulder	Shoulder Angle
Bust	Armpit	Elbow Angle
Waist	Bust	Cuffs Attenuation Ratio
Abdomen	Abdomen	
Hip	Hip	
Thigh	Crotch	
Leg Length	Thigh	

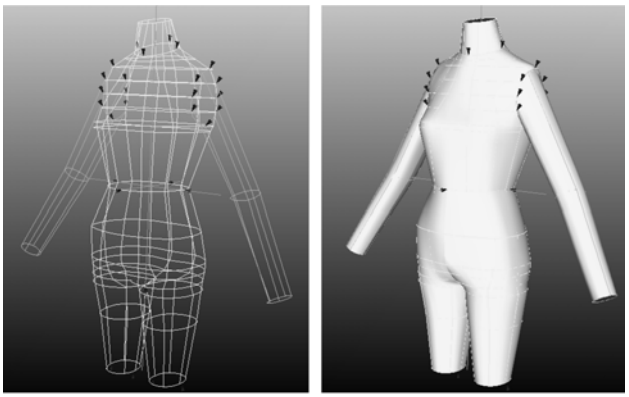

Figure 5. An example of parametric body model: (a) Structural frame and (b) Surfaced model.

curve fitting was performed using the well-known Graham convex hull generation method and the results of circular fitting are shown in Figure 4.

Deformation of Body Model

In this study, eight primary cross sections and a number of auxiliary cross sections were used for the definition of a body model and the shape and size of the model can be changed by resizing and relocating each section considering the final desired geometry. Each body part was generated by sweeping those cross sections. Also the shape of armhole curve and several parameters were used to define sleeves of which the shoulder angle as well as the elbow angle can be controlled freely. Some major landmarks are automatically generated on respective cross section. Controllable body measurement parameters are listed in Table 1 and the structural frame and surfaced model of a parametric body is as shown in Figure 5.

Parametric Surface Generation

Recently, the main trend of garment CAD system is migrating into three-dimensional system from conventional two-dimensional system. To design garment patterns directly from three-dimensional body, a special feature is required for a body model having various body informations. Arbitrary shapes

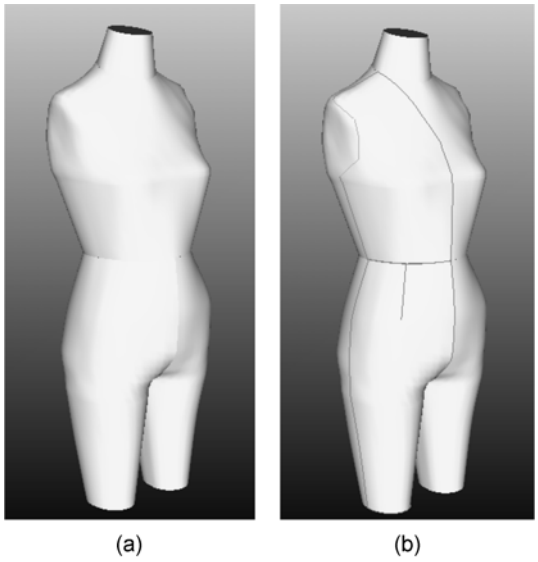

Figure 6. An example of parametric surface model: (a) Parametric surface model and (b) Pattern drawing example for bodice and slacks.

including garment patterns can be drawn on its surface easily. As for the model shown in Figure 5, it is impossible because its surface is composed of mutually independent triangular elements. Therefore, a parametric surface generation algorithm was implemented in this study to make a functional surface model, which can be used for garment drape simulation as well as three-dimensional pattern design. The parametric surface used in this study is the Bézier surface on which any point can be referred easily using two parameters u and v, each of which ranges from 0 to 1. As each part of body can be roughly approximated as a cylinder, parameter u can be defined along its circumferential direction while parameter v along its axial direction. As each cross section is fitted with a continuous curve, control points for parametric surface can be defined easily. An example of the parametric surface model is shown in Figure 6(a) and an example of pattern drawing is shown in Figure 6(b). As shown in Figure 6, such a parametric surface body model can be used for three-dimensional pattern design process, which is another one of our research topics.

Bounding Box Definition

The primary goal of this study is the development of a deformable parametric body model for three-dimensional garment pattern design and garment drape simulation. Especially, one of the most difficult problems in garment drape simulation is the appropriate spatial arrangement of numerous pattern pieces around the body. However, it is a very difficult and time-consuming job for the user to arrange those pieces using only two-dimensional input devices. Therefore, the automation of this process is inevitable for the

development of an efficient garment drape simulation system. In this study, an algorithm was developed to generate virtual cylindrical volumes, which tightly cover each body part called the bounding boxes. Once all the bounding boxes are defined for each body part, they can be flattened onto two-dimensional plane as all of them are developable surfaces. Then the user can easily arrange pattern pieces with respect to those boxes and the pieces will be arranged spatially through the inverse mapping of bounding boxes into their original three-dimensional forms. Using this method, numerous pattern pieces can be arranged automatically and appropriately without any interference with the underlying body model that the overall drape simulation time can be dramatically reduced. Examples of bounding boxes defined for each body part are as shown in Figure 7.

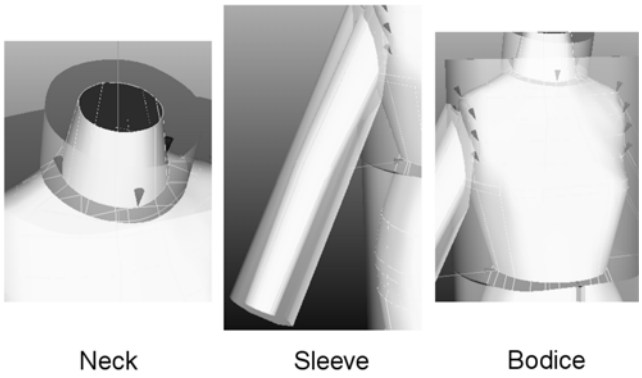

Figure 7. Examples of bounding boxes for various body parts.

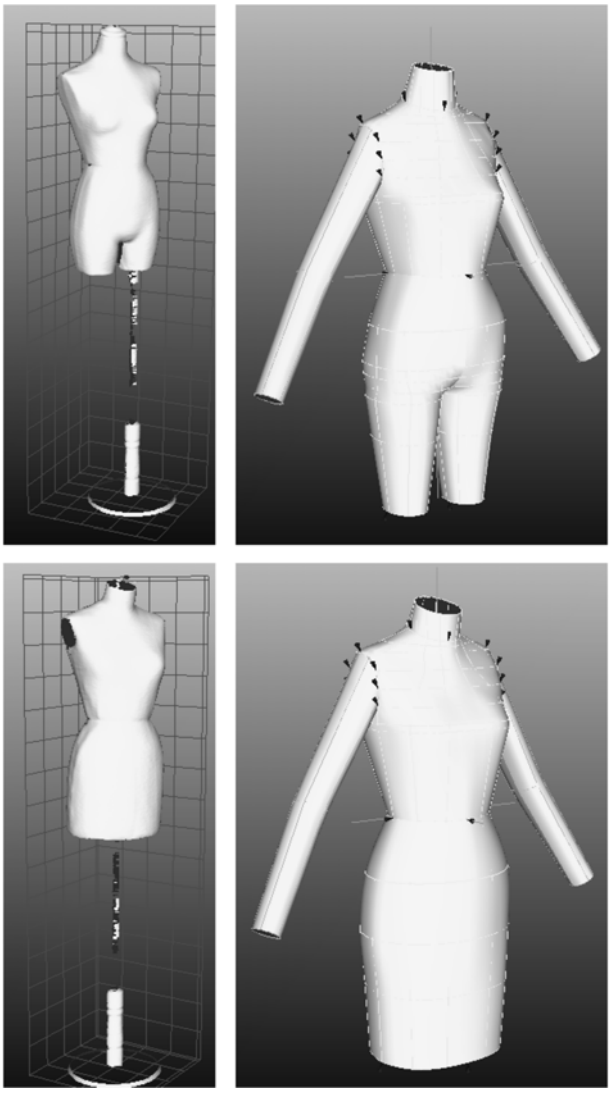

Figure 8. Example of originally scanned body and reconstructed parametric model.

Results and Discussion

Examples of three-dimensionally scanned body models and their corresponding parametric body models are shown in Figure 8. As the shape and size of a parametric body can be changed easily while maintaining the overall shape of its original model, a garment manufacturing company can keep its own model database obtained from scanning and converting

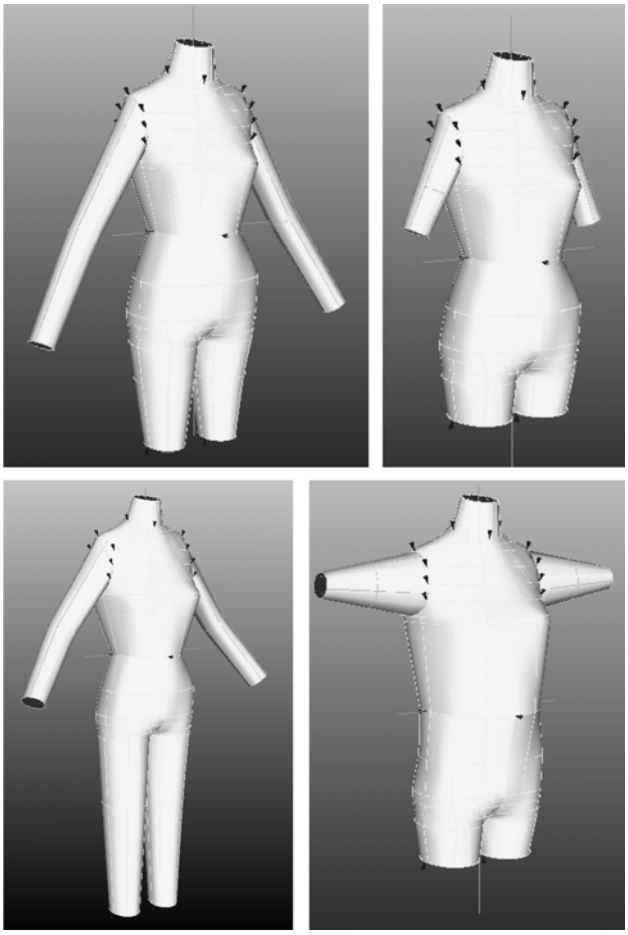

Figure 9. Deformation of a parametric body model.

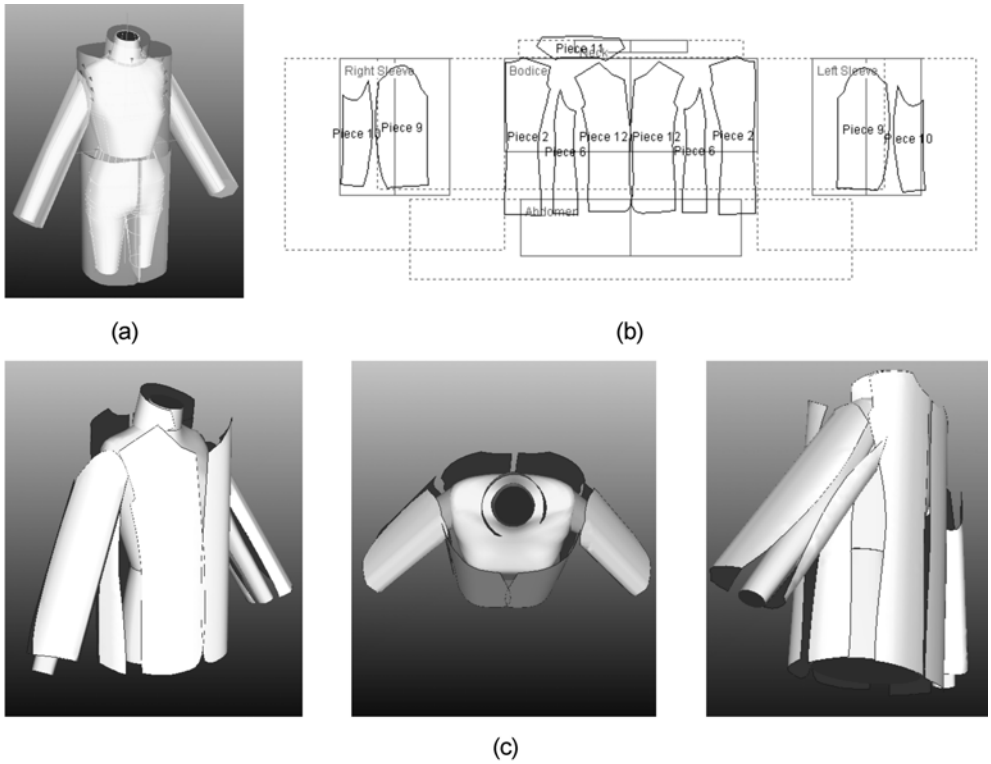

Figure 10. Schematic diagram of spatial pattern arrangement using bounding boxes: (a) Bounding boxes defined around each body part, (b) Arrangement of pattern piece with respect to flattened bounding boxes, and (c) Inverse mapping of flattened bounding boxes for spatial pattern arrangement.

as many standard body models as possible.

As shown in Figure 9, a parametric model can have a variety of shape and the changes can be verified in real time on a general Pentium 4 1.3 GHz personal computer system through a fast calculation algorithm. Therefore, the effect of a certain parameter on the shape and size of the body model can be analyzed very quickly.

Bounding boxes can be automatically generated as shown in Figure 10(a). User can arrange numerous pattern pieces easily with respect to the flattened bounding boxes as shown in Figure 10(b) and the pattern pieces will be arranged around the body through the inverse mapping of bounding boxes as shown in Figure 10(c). The shape and size of bounding boxes are also changed according to the underlying body model in real time that changes can be verified instantaneously.

Conclusions

A parametric body model generation system has been developed. For this, a resizable body model was generated from the three-dimensional body scan data using various mathematic and geometric algorithms. The size and shape of body model can be changed easily and instantaneously using an intuitive user interface that any kinds of body models can be generated easily once a parametric body model is formed. Furthermore the surface of body model can be converted into a set of parametric surfaces. Arbitrary shapes can be drawn easily on it and therefore it can be used for three-dimensional pattern design system. Finally, tight fitting bounding boxes can be defined for a body model, which will be used in the automatic spatial arrangement of pattern pieces in the garment drape simulation system. The parametric body model has various application fields in the three-dimensional garment CAD system and it would be helpful to generate a truly practical system if such a functional body model could be refined by further researches.

References

1. K. Lee, “Principles of CAD/CAM/CAE Systems”, Addison-Wesley, Boston, 1999.
2. B. J. Collier and J. R. Collier, *Cloth. Text. Res. J.*, **8**, 7 (1990).
3. W. Aldrich, “CAD in Clothing and Textiles”, Oxford BSP Professional Books, London, 1992.
4. H. Okabe, H. Imaoka, T. Tomiha, and H. Niwaya, *Comput. Graphics* (*Proc. SIGGRAPH 92*), **26**, 105 (1992).
5. B. K. Hinds, J. McCartney, C. Hadden, and J. Diamond, *Int. J. Cloth. Sci. and Tech*., **4**, 6 (1992).
6. Z. Stjepanovic, *Int. J. Cloth. Sci. and Tech*., **7**, 81 (1995).

7. J. P. Turner, *Int. J. Cloth. Sci. and Tech*., **6**, 28 (1994).
8. P. Volino and N. M. Thalmann, *VSMM '97 Proc*., *Int. Conf.*, Geneva, 109 (1997).
9. K. H. Roediger, *Proc. Comput. Graph. Int.*, Hanover, 396 (1998).
10. J. McCartney, B. K. Hinds, B. L. Seow, and D. Gong, *J. Mater. Process Tech.*, **107**, 31 (2000).
11. F. L. Heisey, P. Brown, and R. F. Johnson, *Textile Res. J*., **60**, 690 (1990).
12. F. L. Heisey, P. Brown, and R. F. Johnson, *Textile Res. J*., **60**, 731 (1990).
13. J. McCartney, B. K. Hinds, and B. L. Seow, *Comput. Aided Design*, **31**, 249 (1999).
14. S. Paquette, *IEEE Comput. Graphics and Appl*., **16**, 11 (1996).
15. H. A. M. Daanen and G. J. V. Water, *Displays*, **19**, 111 (1999).
16. F. Arlt and A. Marach, *Int. J. Ind. Ergonom*., **22**, 333 (1998).
17. C. K. Au and M. M. F. Yuen, *Comput. Aided Design*, **31**, 751 (1999).