Customized 3D Clothes Modeling for Virtual Try-On System Based on Multiple Kinects

Shiyi Huang

Thesis submitted to the

Faculty of Graduate and Postdoctoral Studies

In partial fulfillment of the requirements for the degree

Master of Applied Science in Electrical and Computer Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering

School of Electrical Engineering and Computer Science

University of Ottawa

Ottawa, Ontario, Canada

Abstract

Most existing 3D virtual try-on systems put clothes designed in one environment on a human captured in another environment, which cause the mismatching color and brightness problem. And also typical 3D clothes modeling starts with manually designed 2D patterns, deforms them to fit on a human, and applies stitching to sew those patterns together. Such work usually relies on labor work.

In this thesis, we describe an approach to reconstruct clothes and human that both are from the same space. With multiple Kinects, it models the 3D clothes directly out of a dressed mannequin without the need of the predefined 2D clothes patterns, and fits them to a human user. Our approach has several advantages: (i) a simple hardware setting consisted of multiple Kinects to capture a human model; (ii) 3D clothes modeling directly out of captured human figure. To the best of our knowledge, our work is the first one which separates clothes out of captured human figure; (iii) resizing of clothes adapting to any sized human user; (iv) a novel idea of virtual try-on where clothes and human are captured in the same location.

Acknowledgements

Deepest and sincerest appreciation given to my supervisor Prof. WonSook Lee, without her support and encouragement in the road of doing research, it is impossible to conquer all kinds of problems existed in the project.

Many thanks to all of my colleagues at Computer Graphics (CG++) group. Chao, Zhongrui, Jing, Yongji and Iman help me a lot not only in research but also in my daily life in Ottawa.

Thanks my nice roommates Ali, Carson, and Charles for richening my life and providing me homelike living environment.

Forever love and best wishes to my parents and my sister. Without their encouragements, patience and support, I cannot calm myself down to overcome difficulties in life and to pursue the higher degree. The thesis is dedicated to them.

Table of Contents

List of Figures

List of Tables

Abbreviations

2D	Two Dimensional
3D	Three Dimensional
IR	Infrared Radiation
ICP	Iterative Closest Points
CCD	Charged-Coupled Device
PSD	Positive Sensitive Device
ToF	Time-of-Flight
RBF	Radial Basis Function
FFD	Free-Form Deformation
RGB	Red Green Blue
E-Commerce	Electronic Commerce
HD	High-Definition
VGA	Video Graphics Array
MC	Marching Cubes

Chapter 1. Introduction

Online shopping is popular with customers and retailers nowadays. Among the products sold and bought online, clothes stand out for their characteristics: illumination, color, and size affect the customer's decision. In off-line in person shopping, it is not convenient for customers to try on the clothes until they find something fitting to their size, style, and color. Virtual try-on for clothes is the technique that can visualize the human dressed in selected clothes virtually without need to undress and dress themselves physically. Basically, it consists of two objects: human and clothes. And the main objective is to fit the clothes to a human's body.

Based on the data type of human and clothes, we define that virtual try-on system can be divided into 2D and 3D system. And we define 2D system to be 2D clothes and 2D/3D human while 3D system to be 3D clothes and 2D/3D human. Substantial progress has been achieved in this area during last decades.

Hauswiesner et al. [5]'s system was through image-based rendering approach, which was considered as a 2D virtual try-on system. Ten cameras were used to record videos of one model-user wearing clothes, and they extracted those clothes from the recorded video frames to create the clothes database. It then transferred the appearance of the recorded clothes to the user's body through matching user's poses from input and recorded video frames. Finally, the dressing result was displayed in a TV mounted in the front wall.

The EON Interactive Mirror [7] [8] was considered as a 3D virtual try-on system, which used one Kinect to capture depth data and track user's skeleton, one High-

Definition (HD) camera for VGA videos capturing of the user, and one vertical TV screen for displaying the virtual try-on result. Several professional artists used 3ds Max to model clothes based on actual catalogue images in the offline stage. While online, it detected one nearest person and tracked motion to merge 3D clothes with the HD videos of the user. Gesture-based interaction was deployed for customer to choose clothes or accessories.

Zhang et al. [12]'s system also adopted Kinect to capture data, which was also considered as a 3D virtual try-on system. One customized 3D static human model was generated from a template model based on the anthropometry method, which took measurements based on partial information extracted from Kinect, such as human body contour and skeleton joints as input. Animation of the 3D static human model was implemented based on the recorded customer's motion data captured by Kinect. Several 2D garment patterns were mapped onto the 3D human model to generated 3D clothes.

Other systems such as Fitnect [9], and the demo video about Kinect for Windows Retail Clothes [10] described the similar idea as the EON Interactive Mirror.

Those existing approaches for virtual try-on have several limitations. A large garment database was required to solve the mismatching problem (A missing pose in the database will cause the mismatch between garment and user's poser.) in [5]. 3D clothes were resized uniformly based on the customer's shoulder height in [7], which would cause problem when user is far from the standard portion. Specific format of 2D clothes patterns was required for the 3D clothes generation in [12].

Our work is inspired by the existing limitations in current virtual try-on systems,

such as the requirements of manually designed 2D clothes patterns. And we want to use the advance of 3D scanner, which can provide more accurate depth data and more types of data in low price to do the 3D clothes modeling without a designer and/or 2D patterns.

In this thesis, a new idea of virtual try-on is introduced. A mannequin is nicely dressed and a customer wants to check the clothes on him/her but without actually putting on the clothes. We just utilize a 3D scanner to capture the customer's body data (or the customer's previously stored data can be used in the shop), and then we transfer the mannequin's clothes to the customer virtually. It is important that clothes and human should be in the same space in reality, which can have the same illumination condition to achieve more realistic try-on result. While most existed 3D virtual try-on systems put clothes and human into different space, our system emphasizes on it by reconstructing clothes and human that both are from the same space. Although some illumination correction algorithms can be used to solve the mismatching brightness problem, capturing clothes and human that both are from the same place can produce far more realistic result.

Our long term goal is to design a 3D virtual try-on system that can be divided into two steps: offline and online. Several rooms that have the same capturing environment are placed at different locations in different cities. We capture clothes and customers' data, model them into virtual 3D clothes and 3D human in the offline stage. Reconstructed results are uploaded into the online system. Customer can visualize themselves dressed in selected clothes by loading their 3D body model and selecting the concerned clothes.

Our current work is part of the whole system to focus on 3D clothes modeling and customization.

1.1 Overview of the System

Our approach is based on multiple newly released Kinects, which has higher RGB sensor resolution and better depth sensor accuracy compared with the old Kinect, and it is cheaper compared with laser scanner like Cyberware [51] so that it can be used in practice. Four Kinects are used in the experiment to capture 360° views of the dressed mannequin with a shirt and a trouser. No special background is used. Clothes are extracted from the dressed mannequin using Kinect Function and color segmentation method. Then four views of clothes point clouds are aligned based on 3D registration technique. Next, surface reconstruction method is used to generate the clothes mesh, and then through mesh post-processing and texture mapping, the well-reconstructed and textured clothes can be obtained. Clothes resizing is achieved by using the same transformation that deforms the naked mannequin to have the user's body size and shape.

The system can be divided into two parts mainly, which are shown below:

- Data Acquisition
- Data Processing

Our 3D clothes modeling and customization pipeline is displayed in Figure 1.1.

In the data acquisition part, first, we implement camera calibration and stereo camera calibration for the two built-in Kinect cameras. And then we use KinectFusion

to scan the naked mannequin and one user dressed in tight clothes to obtain their respective 3D models. Next, four views of the naked mannequin are captured before the mannequin put on clothes. Finally, four views of the dressed mannequin are captured.

In the data processing part, we process the captured data to generate the 3D clothes and resize it to fit the user's body size and shape.

We use Kinect Body Tracking Function to extract mannequin from the background in front and back views, while using Depth Thresholding in right and left views. Then to extract clothes from the dressed mannequin and separate them into one shirt and one trouser (shirt and trouser are in different colors), color segmentation method is utilized. After that, region filling approach is used to fill in several data missing regions. Four views of shirt and trouser point clouds are generated using the extracted shirt and trouser images, depth data, intrinsic parameters, and extrinsic parameters. We map four views of the naked mannequin to the 3D mannequin model to obtain the transformation between each view. Then by using the same transformations, four views' point clouds of the clothes are aligned. To create clothes mesh from the aligned point clouds, surface reconstruction technique is used. While the reconstructed mesh in the experiment is watertight, mesh post-processing is used to process it to output a much more realistic model shape. And then we apply texture mapping to the mesh to obtain the textured clothes model.

In the stage of resizing, we deform the naked mannequin model to have the user's body size and shape based on mesh deformation approach. By using the same

deformation function, clothes are resized and fitted to the user's body. Then surface correction is performed to solve the penetrating problem so that a more realistic try-on effect can be achieved.

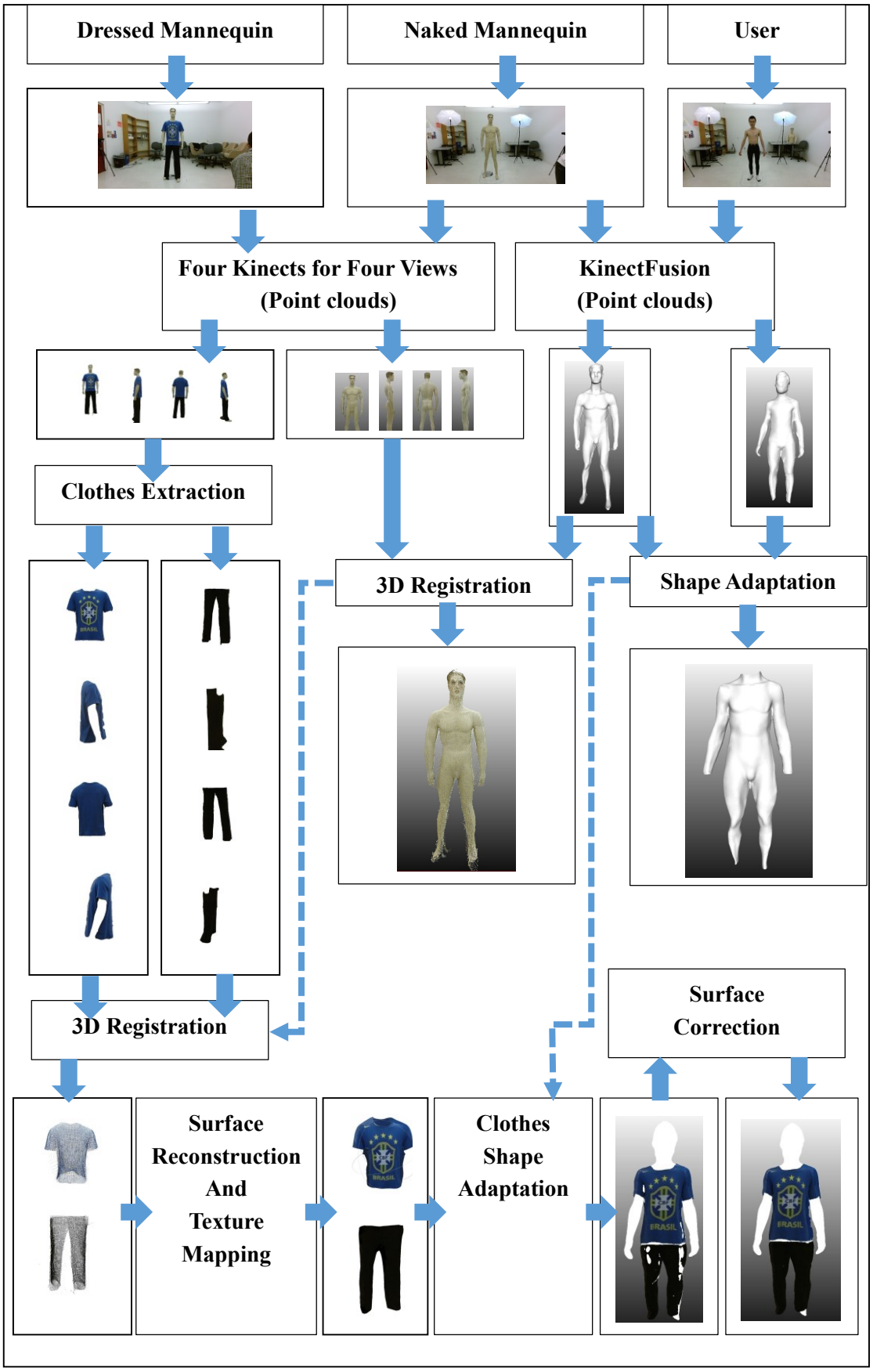


Figure 1.1 The pipeline of our 3D clothes modeling and customization system.

1.2 Organization of the Thesis

The thesis is organized as following:

- Chapter 2 is for the literature review, which covers related research work about the existing virtual try-on systems, 3D clothes modeling, 3D clothes resizing, 3D scanners, and related algorithms for processing 3D data are also introduced.
- Chapter 3 is the methodology part. Detailed descriptions of implementations are included in this chapter. Capturing environment setting, camera calibration and stereo camera calibration, data processing parts such as color segmentation, 3D registration, surface reconstruction, texture mapping, and resizing are introduced respectively.
- Chapter 4 is the result and validation part. The final results and evaluation are presented to demonstrate the efficiency, benefits and limitations of our proposed approach.
- Chapter 5 is the conclusion part. The main contributions of this thesis are described. Summary and potential future research work are also discussed.

Chapter 2. Literature Review

In this chapter, we introduce some related works to the virtual try-on system for clothes. And since our proposed 3D virtual try-on system is based on captured data, related works about data acquisition are also described. Then some common techniques for data processing such as color segmentation, calibration, 3D registration, surface reconstruction, texture mapping, and resizing are discussed later.

2.1 Virtual Try-On

Virtual try-on system for clothes is the technique for visualizing a human dressed in selected clothes without need to dress them physically [1]. And it has the potential application in online clothes shopping system [2] [3].

Human and clothes are two main concerned objects in the virtual try-on system. We define that virtual try-on system for clothes can be divided into two series: 2D and 3D, based on the data type of clothes. 2D clothes and 2D or 3D human are considered as 2D system. While for 3D system, 3D clothes and 2D or 3D human are included. We discuss 2D virtual try-on system in section 2.1.1, and 3D virtual try-on system in section 2.1.2.

2.1.1 Virtual Try-On in 2D

2D virtual try-on is the idea that represents clothes as the image frames or video frames, and to fit the clothes to human body. In the following, we discuss several latest works.

Araki et al. [4] proposed a 2D virtual try-on system that deformed clothes images to fit a user's motion from a web camera. It could be divided into two stages. In the offline stage, clothes images were triangulated and prepared for the deformation. In the online stage, the system captured user's video and recognized some markers. The clothes were resized and fitted to the human body based on the markers.

Hauswiesner et al. [5] constructed a virtual try-on system through image-based rendering. They used ten cameras to record videos of one model-user dressed in clothes. Then the garment database was created by extracting clothes from the recorded videos. In the fitting stage, the system transferred the appearance of the recorded clothes onto the user's body by matching the current user's pose from the input video frames with the model-user's pose from the recorded videos frames. The result was shown in a TV mounted in the front wall. A pipeline of this system is shown in Figure 2.1.

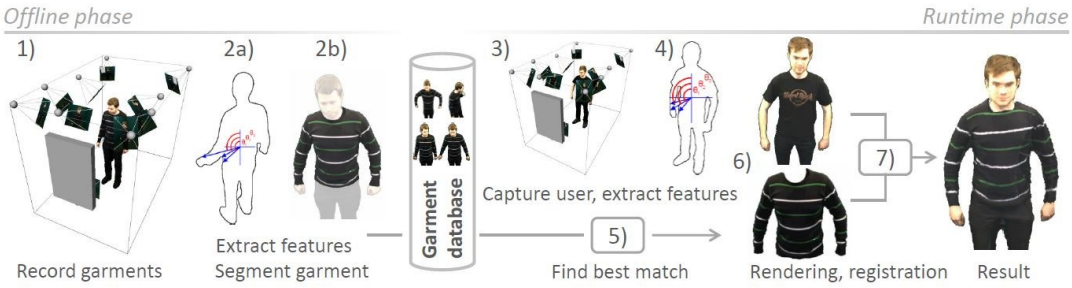


Figure 2.1 Overview of the virtual try-on system through image-based rendering (reprinted from [5])

A real time virtual dressing room using one Kinect was proposed in [6]. It used one Kinect for tracking user's skeleton. Then a 2D clothes was mapped onto the user for dressing.

One of the advantages of 2D virtual try-on system is that its implementation is not very complicated and has the potential to visualize dressing human in real time.

Moreover, its requirements for hardware device is not low. However, one obvious disadvantage of the 2D virtual try-on system is original clothes can be seen even after dressing selected clothes.

2.1.2 Virtual Try-On in 3D

With the advance of 3D sensing device, especially Kinect, the development of 3D virtual try-on system is unprecedented. Several 3D virtual try-on systems have been created over the past few years to focus on specific problems, such as customizing human model, fitting clothes to human body, and so forth. In the following, we discuss some latest works on 3D virtual try-on system.

Giovanni et al. [7] proposed to use one Kinect and one High-Definition (HD) camera to construct a virtual try-on system. It was named as EON Interactive Mirror [8], the setting is displayed in Figure 2.2. The Kinect was used for detecting one nearest user in the scenario. Then the 3D clothes from garment database was transferred into the HD video capturing of the user, and the final visualization result was displayed on the TV in real time. The User Interface (UI) for clothes selection and displaying virtual try-on effect is shown in Figure 2.3. They used the first-generation Kinect, which was released by *Microsoft* in 2010. But since the resolution of the built-in RGB camera in the Kinect was low, they utilized one HD camera to capture HD video frames of user instead. The system was divided into two stages: offline and online. Calibration between Kinect and HD camera, and the construction of 3D garment database were implemented offline. While human tracking and fitting clothes to the human body were

achieved online and in real time. Customers could use the built-in gesture-based interaction in Kinect SDK to choose clothes. Some artists built the 3D clothes manually using 3DS Max based on some real catalogue images. One of the drawbacks is that 3D clothes is resized uniformly based on the customer's shoulder height calculated from the tracked skeletal data. Most existing systems that use one Kinect to construct the virtual try-on system depicted the similar idea as the EON, such as the Fitnect [9], the demo video about Kinect for Windows Retail Clothes [10], and so forth. However, information of technical details about those systems is limited in public.

Figure 2.2 EON Interactive Mirror (reprinted from [7]) which uses one Kinect and one HD camera for capturing. The screen shows the preprocessed clothes collection.

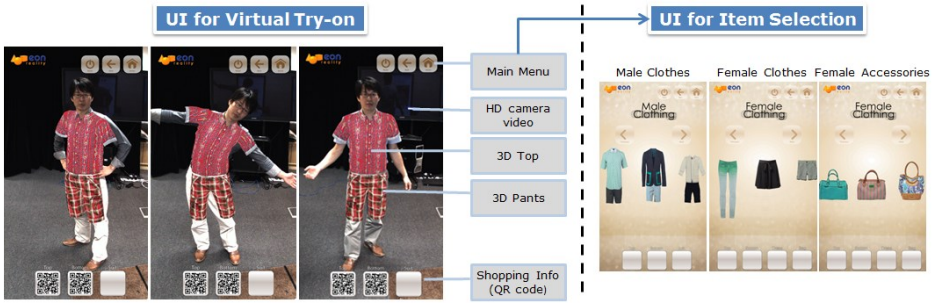


Figure 2.3 UI for virtual try-on and UI for item selection for the EON Interactive Mirror (reprinted from [7]).

In [11], a virtual try-on system using one Kinect was designed and three scenarios were displayed in the result as shown in Figure 2.4: avatar with 3D clothes; 3D clothes on user's image; 3D clothes on the avatar blended with the face image of user. They

used twelve measurements calculated from the Kinect captured data to customize the avatar. The avatar was scaled globally based on the height of the user. Then it was scaled locally based on other measurements, such as waist, knee heights etc. The avatar's body skin color was changed based on the skin color from captured image of user. For the 3D clothes modeling part, they utilized the available 3D clothes garment database, and technical details of 3D clothes modeling were not introduced.

(a) (b) (c)

Figure 2.4 Three scenarios of virtual try-on system using one Kinect: (a) avatar with 3D clothes (b) 3D clothes on user's image (c) 3D clothes blended with the face image of user (reprinted from [11])

The system proposed by Zhang et al. [12] was also Kinect-based. The workflows of the system is displayed in Figure 2.5. They used one Kinect to capture images of user and track user's skeletal data. 26 measurements calculated from the captured skeletal data were used to resize one template human model. Then the template model was refined based on the front and side contour images of the user. Several motion data of the user were also captured for animating the customized template model. For the 3D clothes generation, several predefined 2D patterns were created and prepositioned manually onto the model first. Then topological stitching was implemented to sew those patterns together to create the 3D clothes. An idea for automatic prepositioning was

proposed in the system, and it required special data format for the 2D clothes patterns, which consisted of geometrical, topological, and semantic information.

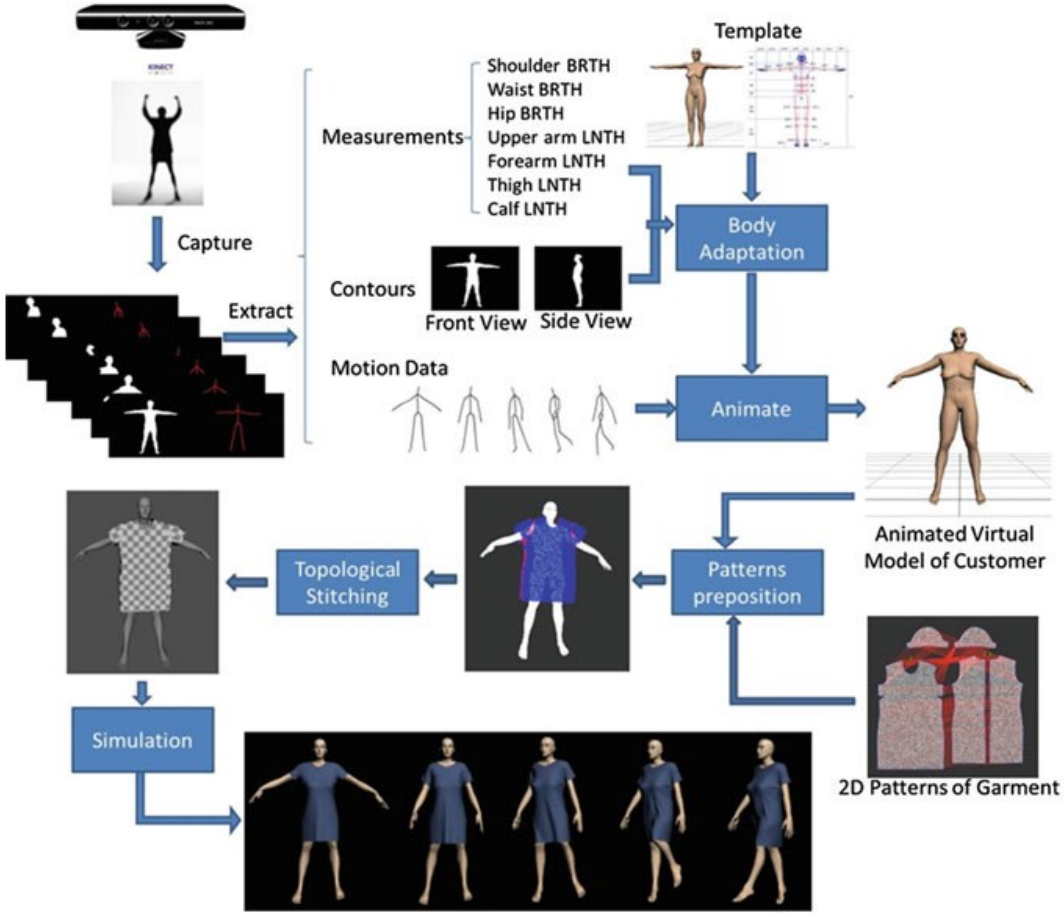


Figure 2.5 Workflow of the virtual try-on using one Kinect to capture images of user and track user's skeletal data (reprinted from [12])

2.2 Clothes Modeling for Virtual Try-On

3D clothes modeling is one part of the 3D virtual try-on system. Basically, it can be classified as three types; manually modeling, automatically 2D patterns to 3D clothes modeling, and scanner-based modeling. However, other methods are also proposed for 3D clothes modeling, such as [13], which is an automatic method to create 3D clothes based on 2D photo input where the body is created from three photo views and then clothes are extracted in 3D by color segmentation. However, as it was based on feature-

based human cloning technique, accuracy was lower than 3D scanner based method.

For the manually clothes modeling, it requires a user to use some kinds of 3D computer graphic software such as Blender [14], 3ds Max [15] or Maya [16] to generate the 3D clothes. Marvelous Designer [17] is a typical manually clothes modeling system. Several 2D clothes patterns are designed virtually. Then those patterns are sewed by using mouse to select corresponding points between two patterns. After that, 2D clothes patterns are deformed to fit a digital human model and textured to generate the 3D clothes. A lot of manual work is required to generate a high quality 3D clothes using this method.

For the automatically 2D patterns to 3D clothes modeling, it is much similar as the manual modeling, while the main difference is that 2D clothes patterns are based on actual images and the it is automatically modeled into 3D clothes. In [12], they proposed the automatically 3D clothes modeling based on several predefined 2D patterns. Several information should be included in the 2D pattern, such as geometrical information, topological information, and semantic information. In the geometrical part, 2D patterns were defined by 2D contours and triangulated into 2D mesh. In the topological part, linkages between the contours of patterns were specified by the seams. While in the semantic part, there were some tags for layer id, side id, etc. Several markers were used to specify with which skeleton of the template model the 2D patterns were linked, besides, initial 3D position placed on the template model for 2D patterns were recorded. The proposed method prevented manually pre-position 2D patterns onto customized human model, but it still required designers to position those patterns to the

template model. After the prepositioning step, 2D garments patterns were stitched together along the seams to generate the 3D clothes.

For the scanner-based modeling, it uses 3D scanner to scan the real clothes for 3D clothes reconstruction, which can provide higher accuracy and more realistic shape. The scanner-based modeling method is popular for human modeling [18], hair modeling [19] or other familiar objects reconstruction. However, to the best of our knowledge, we haven't seen any research about scanner-based 3D clothes modeling, which can be an area that we can have a contribution to. We break scanner-based 3D clothes modeling into two parts, which are data acquisition and data processing. 3D scanner is used to scan one dresses mannequin in the data acquisition step, the clothes are extracted, reconstructed and resized in the data processing step.

2.3 Clothes Resizing for Virtual Try-On

Fitting a 3D clothes to a 3D human is a crucial part in virtual try-on system. The fitting step directly affects the visualization result. For the automatically 2D patterns to 3D clothes, the clothes was sewed from several deformed 2D patterns. And information of the technical details about deforming 2D patterns to the 3D mannequin were not described in most papers, such as [11] [12].

Li et al. [20] proposed a method for fitting a given 3D clothes model onto human models of various body shapes and poses. It is noted that the fitting is achieved by deforming the clothes mesh to match the shapes of the human models by using several combinations of following steps: skeleton-driven volumetric deformation, garment-

human shape similarity matching and evaluation, the constraints of garment-human contact, and garment-human ease allowance [20]. The implementation of this approach was more complicated than the approach of using Radial Basis Function (RBF) for deformation.

Since fitting a 3D clothes to a human body is also the problem of resizing 3D model. We discuss it further in the section 2.5.6.

2.4 Data Acquisition

To model 3D objects from captured data, the step of data acquisition is required and the data captured affects the reconstructed result. Basically, there are two methods for capturing depth data: image-based and 3D scanner-based. It is noted that image-based method can provide high quality visualization result while 3D scanner-based responses with actual 3D depth data. In the following subsections, we discuss them in more details.

2.4.1 Structured Light

Structured light is the process of projecting a known pattern of pixels on to a scene for depth sensing. With the introduction of objects in the scene, those patterns captured from different views will show deformed projected patterns, which allow the vision system to calculate the range data and surface information of the objects [21]. Invisible structured light, such as infrared light and so forth, is used as the structured light, which avoids interfering with other computer visions tasks for which the projected patterns

will be confusing [21]. The structured light sensors usually consist of one or more cameras and one or more light sources [22]. A variety of techniques about structured light have been designed in the past few years. And a comparative survey on invisible structured light techniques can be found on [22], while the state of the art in structured light patterns for surface profilometry can be found on [23].

Valkenburg et al. [24] proposed a structured light system consists of one camera and one projector for accurate 3D measurement. A coded stripe pattern was projected on the scene and the camera captured its image. Then a corresponding stripe number and image location could be found for each visible point in the world, which were used to compute the depth data of the scene.

Scharstein et al. [25] proposed to use the structured light for high accuracy stereo depth data acquisition. A pair of cameras and one or more light sources that projected structured light patterns onto the scene were deployed. The usage of the structured light was for making it easier to solve the correspondence problem in stereo vision. A survey discussed several progresses in applying coded structured light to the stereo vision can be found on [26].

2.4.2 Stereo Vision

Stereo vision is the estimation of 3D information from 2D nearby images. By comparing information about one scene from two different viewpoints, 3D information can be calculated by examination of the relative positions of objects in the two planes [27]. Some primary problems to be solved in stereo vision are calibration,

correspondence, and reconstruction [28]. Cameras need to be calibrated for determining camera system's internal geometry and external geometry. Then correspondence between two locations in each camera projected from the same point in space should be computed. The displacement of the projected points in each image forms a disparity map. Based on the known camera geometry, range data can be obtained from the disparity map.

Stereo vision can be divided into two types based on with/without usage of light, such as structured light or laser, to probe the scene: active stereo vision and passive stereo vision [29]. The problem of finding stereo correspondence becomes easier with the introduction of structured light.

A lot of techniques of using stereo vision for range data acquisition have been researched and developed for decades. A relatively detailed survey about stereo vision can be found in [28]. Chen et al. [29] deployed passive and active stereo vision to detect the smooth surface of metal plates. Stereolabs [30] released a stereo camera for depth sensing based on the passive stereo vision technique.

2.4.3 Multi-view cameras

Multi-view camera vision is the technique that computes a 3D representation given several images of objects or scene [31]. The collection of images usually taken from known camera viewpoints [32]. Most existing approaches utilized the calibrated camera viewpoints system to calculate the transformation matrix between each viewpoint, and based on the photogrammetry measurement, objects' 3D shape can be obtained. A lot

of multi-view reconstruction algorithms have been proposed, but due to lack of suitable datasets, direct comparisons between those algorithms are quite hard. Seitz et al. [32] collected several calibrated multi-view image sets and corresponding ground truth 3D mesh models. They used those data to evaluate several algorithms based on accuracy and completeness.

Gu et al. [33] proposed a 3D reconstruction system for human body modeling based on six views. And they used two cameras to capture human body images for each view. A cylindrical object with patterns was used during camera calibration. The 3D human was generated based on super quadrics model fitting, which was a time-consuming step.

Bradley et al. [34] proposed to deploy sixteen inexpensive High-Definition (HD) consumer cameras for marker-less garment capturing and reconstruction. The initial mesh calculated from sixteen views usually contained holes. Two steps were applied to generate the final result, which were consistent cross-parameterization, compatible re-meshing and surface completion.

Infinite Reality [35] used 150 DSLR for data capturing. It took instant snapshots of the scenario to produce extremely accurate 3D model. Detailed technical descriptions are not available in public.

As the multi-view vision usually requires that several feature points can be detected on multiple images, in order to generate high quality result, more cameras are required, which will increase the cost of system. Moreover, without same lighting condition for each view, the color of concerned objects usually varies with the moving of viewpoint, which also makes the feature detection hard.

2.4.4 Laser Scanner

3D scanner is the device that adopts some kind of measuring techniques to calculate depth data in order to generate 3D model or scenario. It is usually divided into two types: contact and non-contact. And for non-contact, it can be further divided into two main categories: active and passive [36].

Laser scanner is the non-contact active 3D scanner. It emits laser to probe the object and detects the reflected laser for measuring the depth. The laser can be visible or invisible [37], such as infrared laser scanner, and so forth. Two main 3D laser scanner used in practice are the triangulation-based and the Time-of-Flight.

For the triangulation-based 3D laser scanner, laser is emitted to touch the object first. Next, a Charge-Coupled Device (CCD) or Position Sensitive Device (PSD) sensor is used for tracking the laser dots. A triangular can be formed based on laser emitter, laser dot and camera, which can be used to estimate the depth value [38].

For the Time-of-Flight (ToF) 3D laser scanner, it calculates the depth based on the known speed of the light, measuring the Time-of-Flight of the signal between the ToF sensor and the object in the scene [39]. In order to detect the phase shifts between the illumination and the reflection, the light source is pulsed or modulated by a continuous-wave source, typically a sinusoid or square wave [40]. The phase shirts between the illumination and the reflection are measured and translated into the distance between ToF sensor and the object.

It is noted that triangulation-based 3D scanner can provide quite high accuracy result, but the depth range is limited. The ToF 3D scanner is opposite [36].

2.4.5 Kinect

Kinect [41] is the 3D scanner released by *Microsoft*. So far, two versions have been released: Kinect V1 and Kinect V2. In the following subsections, we discuss their technical details respectively.

2.4.5.1 Kinect V1

Kinect V1 was released in 2010. And it received much attention due to its rapid human pose recognition system developed on top of 3D measurement. Besides, due to its in-expensive cost, reliability and speed of the measurement, it becomes the primary 3D measuring devices in indoor robotics, 3D scene reconstruction, and object recognition [42]. The architecture of Kinect V1 is displayed in Figure 2.6 and its specifications are listed in Table 1.

Kinect V1 has three major components: color sensor, Infrared (IR) emitter and IR depth sensor. The color sensor provides a 640 x 480 frame at the default frame rate. However, the resolution can be up to 1280 x 1024 at a lower frame rate. Kinect V1 uses structured light technique for estimating depth data. IR emitter projects the predefined pattern to probe the object, due to the introduction of the object, the pattern will be deformed, and then the Infrared Depth sensor captures the deformed pattern to compare it with the predefined reference pattern to calculate the depth value [44]. The depth frame is derived from the IR frame, and it is noted that a small offset existed in those two images. The offset is five pixels in the horizontal direction and four pixels in the vertical directions, respectively [42] [45].

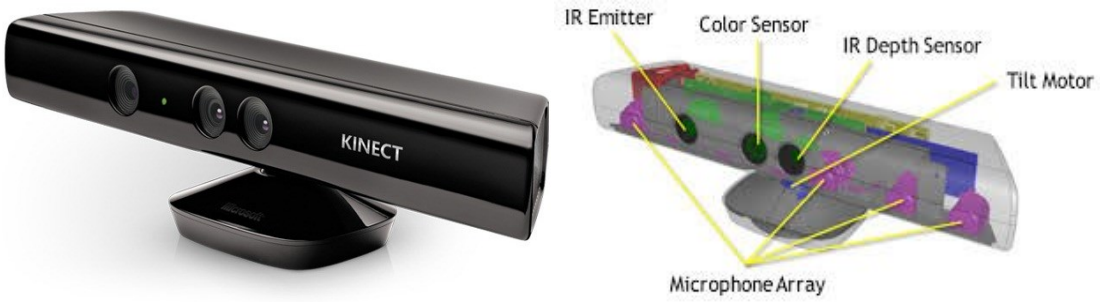


Figure 2.6 Kinect V1 (reprinted from [43])

Table 1 The specifications for Kinect V1

Kinect V1	Specifications
Color Camera Resolution & Frame Rate	640 x 480 @30fps
Depth Camera Resolution & Frame Rate	640 x 480 @30fps
Field of View	43° vertical by 57° horizontal
Depth Space Range in Default Mode	[0.8~4]meters
Depth Space Range in Near Mode	[0.5~3]meters
Skeleton Joints	20
Full Skeleton Tracked	2
USB Standard	2.0
Tilt Motor	Yes
Vertical Tilt Range	±27°

In order to get a good 3D model, high quality depth data are required. The accuracy of Kinect depth sensor affects the final result. The properties of the depth sensor, system's setup, and the condition of the objects' surface are among the elements that affect the accuracy of Kinect depth data [46].

2.4.5.2 Kinect V2

Kinect for Windows V2 [47] was released by *Microsoft* in July, 2014. And its structure is displayed in Figure 2.7. The specifications of Kinect V2 are listed in Table 2.

Compared with Kinect V1, Kinect V2 is equipped with one much higher resolution RGB camera up to 1920 x 1080, and the resolution of depth sensing is 512 x 424.

Different from the two modes for depth range in Kinect V1, one mode is utilized in Kinect V2, which extends the reliable depth range to 4.5 meters. Kinect V2 uses ToF technology for depth sensing, and the principle of ToF has been discussed in section 2.4.4. With higher depth fidelity and a significantly improved noise floor, the depth sensor has an improved 3D visualization, which is able to detect smaller objects and improve the stability of body tracking [47].

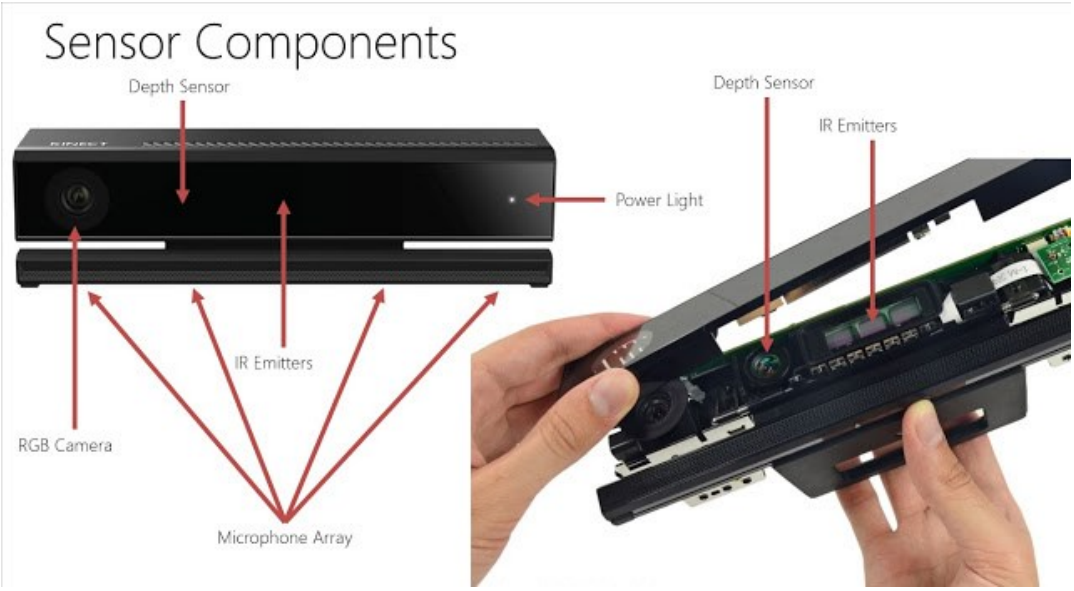


Figure 2.7 Kinect V2 (reprinted from [48])

Table 2 The specifications for Kinect V2

Kinect V2	Specifications
Color Camera Resolution & Frame Rate	1920 x 1080 @30fps
Depth Camera Resolution & Frame Rate	512 x 424 @30fps
Field of View	60° vertical by 70° horizontal
Depth Space Range	[0.5~4.5]meters
Skeleton Joints	25
Full Skeleton Tracked	6
USB Standard	3.0
Tilt Motor	No

In the experiment for the thesis, we use Kinect V2 as the main capturing device since the accuracy of depth data is better, the resolution of RGB image and depth image

is higher, and the available SDK [49] also makes it easier to construct a system that can be operated in Windows operating system. Moreover, the cost for each Kinect V2 sensor is much lower (hundred dollar) compared with other 3D scanners, for example the cost (tens of thousands dollar) of the Whole Body Color 3D Scanner from Cyberware.

2.4.5.3 KinectFusion

KinectFusion [53] is the technique developed by *Microsoft* for scanning real objects or environments to generate corresponding 3D model or scenario. It is included in the Kinect for Windows SDK [49]. And it only requires using one Kinect to scan the object. To generate the 3D model, it registers all valid depth frame into the global implicit surface model continuously using a coarse-to-fine interactive closest point algorithm. It is required to move the Kinect slowly to avoid data missing.

A scene captured in the experiment using KinectFusion is shown in Figure 2.8.

Figure 2.8 A scene captured using KinectFusion

Since KinectFusion captures depth data and incorporates it into the global model, the data quantity is usually around million or higher. The model displayed above has 6,585,549 vertices and 2,195,183 faces. Although the point clouds is well-aligned and

the reconstructed 3D shape is detailed, the texture information of the mesh is usually incorrect for some regions, which causes the problem of color flaw: texture from different views under different illumination conditions are mapped onto the same region repeatedly. The color flaw is displayed in Figure 2.9.

Figure 2.9 KinectFusion color flaw

In the experiment, we use KinectFusion to capture the aligned point clouds for the given object. And we use Poisson Surface Reconstruction algorithm to reconstruct 3D model out of it so that the data quantity can be decreased while ensures a relatively high quality model without texture.

2.5 Data Processing

The captured data from real world using 3D scanner usually contains noise or have some unconcerned components such as background objects etc. The raw data captured using Kinect usually consists of images and depth frames. And based on those data, point clouds can be obtained. The point clouds are the one of the representation forms of the 3D model, which consist of several organized or unorganized 3D points

represented by (x, y, z). Especially with the advent of the new low-cost 3D sensing device such as Kinect, many efforts in point clouds processing have been proposed and developed, and 3D perception gains more and more importance in robotics as well as other fields [54]. So based on the captured data, some data processing methods are required to get the concerned model. In this thesis, our intention is to generate 3D clothes and resize them, so some related and latest techniques about color segmentation, calibration, 3D registration, surface reconstruction, texture mapping and resizing are discussed in the following section.

2.5.1 Color Segmentation

In the captured data, either RGB images or depth data, environment objects, objects of interests and noise are all mixed. To extract concerned components out of the raw data, segmentation technique is required, which can be divided into 2D segmentation and 3D segmentation. As 2D segmentation (color segmentation) is far more mature and much easier to implement than 3D segmentation, we discuss the former in the following.

Color segmentation, which also can be called image segmentation, is the process of partitioning a digital image into multiple segments so that every pixel in the same segment shares certain characteristics, such as color, intensity or texture [55]. A variety of approaches have been proposed over the past few years, and many surveys or study analysis are available, such as [56] [57] [58] [59].

Lucchese et al. [56] divided those techniques for color segmentation into three categories: feature-space based, image-domain based, and physics based. Feature-space

based methods consider color as a constant property of the object, and then by mapping each pixel of the color image into a certain color space, different objects in the image can be represented as different clusters or clouds of points. Clustering and histogram thresholding methods were considered as feature-space based technique. Different than feature-based methods, image-domain based methods emphasize on spatial compactness. Split-and-merge, region growing, edge-based, and neural-network based classification were defined as image-domain based technique. For the physics based technique, it analyzed the light interaction with colored materials and introduced the model of physical interaction.

In the paper [57] [59] [60], color segmentation technique was divided into two broad groups: contour-based approaches and region-based approaches. Contour-based approaches usually starts with edge detection, followed by a linking process to exploit curvilinear continuity, while region-based approaches try to find partitions of the image pixels into different sets corresponding to coherent image properties such as brightness, color and texture [60]. A comprehensive survey for the contemporary trends in color segmentation technique can be found in [58].

Belongie et al. [61] proposed to the color- and texture-based image segmentation using Expectation-Maximization (EM) algorithm for content-based image retrieval. They adopted a ‘blobworld’ for image representation, which consists of dominant colors, mean textured descriptors, spatial centroid, and scatter matrix. Hue-Saturation-Value color space was used for mapping to the Cartesian coordinate, from which the pixel gets its color descriptor. And EM algorithm was used to divide pixels into several

groups.

An unsupervised color segmentation based on color and texture region proposed by Deng et al. [62] was broadly used for image and video segmentation, which was named as JSEG. It consists of two steps: color quantization and spatial segmentation. In the former step, colors in the image are quantized to several classes, which can be used to differentiate regions in the image. Then in the latter step, region growing method is used to segment the image based on the multi-class image. A system implementing the proposed algorithm is available for research purpose and can be found in [63].

In our experiment, we use JSEG which is region-based color segmentation method for extracting concerned objects from the images, since some properties such as color, texture and brightness are considered as the key elements.

2.5.2 Camera Calibration

Our initial representation of the 3D objects utilizes point clouds, which consists of three axis values. Depth value is originated from 3D scanner, while horizontal and vertical values are calculated based on the pin-hole camera model. In order to generate the point clouds and register point clouds with color information, calibration is required.

Calibration is the technique to find the transformation between different spaces. We use multiple Kinect cameras in our project. The Kinect has stereo cameras, where one camera is for images and the other depth camera for 3D points. Thus, for Kinect, we require two steps calibration, one focusing on intrinsic parameters calibration (so called as camera calibration) and the other focusing on extrinsic parameters calibration

(so called as stereo camera calibration). Camera calibration emphasizes on calibration for one camera, and from which parameters are calculated to link 2D space with 3D space. While for stereo camera calibration, extrinsic parameters are computed to find the transformation between different cameras.

Zhang's technique [86] is the widely used camera calibration method, which uses one special pattern consists of several detectable feature points to calculate the transformation between 2D and 3D space.

Stereo camera calibration is based on the triangulation principle. Several synchronized images from two cameras are used to calculate the transformation between two different spaces.

As calibration is usually a necessary step for 3D reconstruction, several open source libraries are available, such as OpenCV [89], and so forth.

Point clouds with color information can be acquired after implementing cameral calibration and stereo camera calibration.

2.5.3 3D Registration

Multiple 3D scanners can be used to capture whole views of the object. But since each view is with respect to the different coordinate space, alignment from multiple-view point clouds is a necessary and crucial step for a high quality 3D model generation.

3D registration is the technique that aligns multiple point clouds, which are with respect to different coordinate systems, to be one point clouds. Iterative Closest Point (ICP) is the widely used algorithm for 3D registration. And two variants have been

proposed, which are point-to-point and point-to-plane. Chen et al. [64]'s ICP algorithm is based on the point-to-point matching type, while Bergevin et al. [65] is based on the point-to-plane matching type. It is noted that the point-to-plane matching type usually performs better in practice compared with the point-to-point type [66] [67]. Iterative Closest Point (ICP) proposed by Besl et al. [68] algorithm is the point-to-plane type and widely used for aligning multiple views, which calculating the transformation matrix by finding the closest point in a target point clouds correspond to the point in source point clouds iteratively. The iteration is terminated when the change in mean-square error falls below a preset threshold that specifies the desired precision of the registration. The algorithm [68] is described in the following:

1. Finding the corresponding closest point in the target point clouds given one point in the source point clouds.
2. Calculating the transformation matrix, which consists of rotation and translation matrix, using the mean-square error function based on the source point and the best matching closest target point.
3. Applying the transformation matrix to the source point.
4. Iterating step 1-3 until the termination criterion is met.

Zhang [69] proposed a modified Kd-Tree algorithm for calculating closest points efficiently.

2.5.4 Surface Reconstruction

After the 3D registration, an aligned point clouds set is generated. Since point

clouds are scattered points, which are sampled points of the real object, surface reconstruction from scattered points is needed. Surface reconstruction is the technique that reconstructs the digital surface, which can be called mesh, based on the scanned and computed point clouds that contain a variety of defects, such as noise, outliers, data missing and misaligned data. A lot of surface reconstruction approaches have been created for emphasizing on different characteristics, such as the ability to deal with the point clouds defects, input data requirements, surface smoothness and reconstruction output type. Among them, Marching Cubes (MC) [70] is probably the most popular iso-surfacing algorithms [71].

The MC algorithm was first proposed by Lorensen and Cline [70], which is a fast and reasonably efficient approach for rendering constant density iso-surfaces that was originally meant for volumetric data. This algorithm proceeds through the scalar field, and takes eight neighbor locations at a time to form an imaginary cube. Then it determines the polygon needed to represent the part of the iso-surface which passes through this cube. Finally the individual polygons are fused into the desired surface. One drawback of the MC algorithm is that it suffers from artifacts at sharp angles, which can cause inconsistence and poor quality in the result [72]. Another disadvantage is that the method is not mathematically proven in terms of preserving the mathematical properties including topological properties of the original object [73]. A variety of algorithms built on the original marching cubes method have been designed, and a quite comprehensive survey about those techniques is introduced in [74].

Based on the surface smoothness classification method introduced in [75], surface

reconstruction methods can be divided into three types: local smoothness priors, global smoothness priors, and piece-wise smoothness priors. The characteristics of local smoothness priors method lies in it ensures the smoothness of the area where point clouds existed, while for the data missing area, it is poorly reconstructed. Moving Least Squares (MLS) methods, Multi-level Partition of Unity (MPU) methods and Parameterization-free Projection methods are all the examples of this type. In the contrast to the local smoothness priors methods, the global smoothness priors methods ensure the smoothness of entirety, which are more suitable for reconstructing watertight surface from the point clouds with data missing. Radial Basis Function (RBF) methods, Indication Function methods and Volumetric Segmentation methods are among this type. The piece-wise smoothness methods focus on explicitly recovering sharp features or boundary components, while ensuring smoothness away from these features, where smoothness may be prescribed locally or globally [75]. After the evaluation of the available surface reconstruction algorithms and the characteristics of the point clouds generated in the experiment, we use global smoothness priors methods for surface reconstruction. In the following, we discuss two surface reconstruction methods: Radial Basis Function and Poisson surface reconstruction. Both of them are global smoothness priors methods.

2.5.4.1 RBF for Surface Reconstruction

Radial Basis Function is a real-value function whose value depends on the distance from the origin, or alternatively on the distance from some other point called center. It

has the form

$$s(x) = p(x) + \sum_{i=1}^{N} w_i \emptyset(|x - x_i|) \tag{2.9}$$

The reconstruction output of RBF is in the implicit filed. A RBF-based surface reconstruction is introduced in J. C. Carr et al. [76]'s work. Let $\{(x_i, y_i, z_i)\}_{i=1}^{n}$ represent the scattered point clouds, using RBF for surface reconstruction lies in solving following equation

$$f(x, y, z) = 0 \tag{2.10}$$

In which the surface is implicitly represented by the function. A signed-distance function is constructed in order to avoid the trivial solution. Off-surface points are generated towards two sides by projecting original points along the surface normals such that

$$f(x_i, y_i, z_i) = 0,\ i = 1, \dots, n(\text{On-surface points}) \tag{2.11}$$

$$f(x_i, y_i, z_i) = d_i \neq 0, i = n + 1, \dots, N(\text{Off-surface points, outside}) \tag{2.12}$$

$$f(x_i, y_i, z_i) = -d_i \neq 0, i = N + 1, \dots, 2N - n(\text{Off-surface points, inside}) \tag{2.13}$$

And then the problem lies in finding a function $s(x)$ to approximate the signed-distance function $f(x)$. RBF is used to get all the coefficients inside the equation (2.9). A simple greedy algorithm for reducing RBFs centers can be divided into four steps [76], which are described in the following:

1. Choosing a subset from the interpolation points (x_i, y_i, z_i) and fitting a RBF to them.
2. Evaluating the residual $\varepsilon_i = f_i - s((x_i, y_i, z_i))$, at all points.
3. If max $\{|\varepsilon_i|\} <$ fitting accuracy then stop. Otherwise, appending new centers

where ε_i is large.

4. Re-fitting RBF and goto 2.

2.5.4.2 Poisson Surface Reconstruction

Poisson Surface Reconstruction [77] is widely used for surface reconstruction, which belongs to the global smoothness priors method and uses the Indicator Function. It considers all the points at once so that it is highly resilient to data noise. A 3D indicator function X (defined at 1 at points inside the model, and 0 at points outside) is computed from the oriented points. And the reconstructed surface is generated by extracting an appropriate iso-surface.

Let $\vec{V}$ be the vector field defined by the oriented points sampled from the surface of a model, so there is a relationship between X and $\vec{V}$, which is

$$\Delta X = \nabla . \nabla X = \nabla . \vec{V} \tag{2.14}$$

It is the standard Poisson problem. A reconstructed surface can be obtained by finding the best least-squares approximate solution to the equation (2.14).

It is noted that the reconstructed model using Poisson surface reconstruction algorithm is watertight, and the global smoothness is adjustable.

2.5.5 Texture Mapping

After the surface reconstruction, the result is usually the mesh without any texture information, which is not vivid and realistic. Texture mapping is the technique that applies texture to the 3D model. A texture map is used to contain all the texture

information, which can be represented by *u* and *v* coordinates. Thus, mapping texture onto mesh is the procedure of finding the corresponding *u* and *v* coordinates given one vertex of the 3D model. In the following, we discuss one of the texture mapping techniques.

TextureMontage proposed by Zhou et al. [78] is the texture mapping technique that is able to achieve the seamless texture mapping. Mesh and images are automatically partitioned by finding the corresponding feature regions. Then most charts will be parameterized over their corresponding image planes through the minimization of a distortion metric based on both geometric distortion and texture mismatch across patch boundaries and images. Finally, it used a surface texture inpainting technique to fill in the surface without corresponding texture patches.

2.5.6 Resizing

To fit the reconstructed 3D clothes to any human's body size and shape, the step of resizing is necessary. Resizing deals with the problem of deforming a model. A variety of techniques have been proposed for deforming the mesh over the past few years.

Igarashi et al. [79] proposed the as-rigid-as-possible shape manipulation. The method enables user to move and deform a 2D shape without manually establishing a skeleton or FFD domain beforehand. The 2D shape is represented by a triangle mesh and several vertices of the mesh can be chosen as the constrained handles by the user. By minimizing the distortion of each triangle, the system calculates the new positions of the remaining free vertices [79].

Li and Lu [80] proposed a method for customizing the 3D garment based on volumetric deformation. The proposed approach can transfer the clothes initially dressed on one reference human model onto a target human model. Three steps were implemented in order to achieve this goal: (i) a spatial mapping between the two human models is established by using Laplacian deformation method with cross-sections as shape constrains; (ii) the space around the clothes reference human model is tetrahedralized into five tetrahedral meshes; (iii) using constrained volumetric graph Laplacian deformation to fit the reference human model onto the target human model. The updated clothes models can be decoded from the deformed tetrahedral meshes and fitted onto the target human model finally [80].

In the following, we generally describe two main deformation techniques: Free-Form Deformation (FFD) and Radial Basis Function (RBF). Both are control-points based and widely used in the shape deformation. And we use RBF for resizing since it is able to deform a much more detailed 3D model compared with FFD.

2.5.6.1 Free Form Deformation

Free-form deformation (FFD) [81] for deformation is by warping the surrounding space. A source model is encapsulated in a cubic controllable box called lattice, which consists of 3D grid of control points. Moving the control points will deform the lattice, which also deforms the model inside the cubic.

The FFD method suffers from high computational cost, which is resulted from the computation of new point position, aliasing artifacts and imposing constraint on the

shape of lattice. Besides, FFD is not suitable for detailed 3D model deformation. Other variants of FFD are designed to improve the FFD, such as the Extended Free-Form Deformation [82], and Dirichlet Free Form Deformation [83].

2.5.6.2 RBF for Deformation

The RBF deformation is based on two data sets, which are control points and target points. Control points are usually selected from the surface of the concerned model, while the target points are on the target model. Any displacement of the control points can be used for RBF network training and calculating the weight. Then all the points of the concerned model are under the transformation of the same RBF so that a deformed model can be obtained. The technical details of RBF has been discussed in section 2.5.4.1.

The choice of control points and corresponding target points affects the final result. A bad selection of control points or incorrect selection for the target points can produce an unexpected and wrong deformed result. More features points should be included in the control points in order to generate a better result.

Two of the advantages of RBF for deformation are: (1) ensuring global smoothness (2) low memory consumption which only requires control points, Radial Basis Function and weight to represent the model.

RBF is widely used for deformation, such as for face deformation [84] or other complex models [85].

Chapter 3. Methodology

In this chapter, the proposed approach for 3D clothes modeling and customization is introduced in detail.

Two scans of a mannequin are processed, one naked and the other with clothes, and a tight-clothes user is also scanned. Clothes are extracted using color segmentation and the difference between naked mannequin and a user is used to calculate for extracting resizing parameters. Then the clothes from mannequin is transformed and resized to fit to the user.

We used KinectFusion, which is the technique that uses one Kinect to scan the real world to get the 3D model or scenario, to scan the naked mannequin and a user dressed in tight clothes to get their respective well-aligned point clouds. Then we applied Poisson surface reconstruction to the two point clouds to obtain their 3D models, respectively. Next, four Kinects were used to capture 360° views of the naked mannequin to generate four views' point clouds. Then Iterative Closest Point (ICP) was utilized to align four views' point clouds with the well-aligned point clouds of the naked mannequin. Based on the four-time usages of ICP, the space relation between every two Kinects could be known, which could be used to align multiple views of the clothes. Four Kinects were also utilized to capture four views' point clouds of the dressed mannequin. Clothes were extracted from the mannequin, then its' point clouds were aligned based on the relation calculated previously. Poisson surface reconstruction was used to reconstruct the initial 3D clothes mesh from the aligned point clouds. And then we applied the proposed mesh modification method and texture mapping technique to

it to obtain the well-reconstructed and textured 3D clothes. Next, Radial Basis Function (RBF) was implemented to deform the 3D model of the naked mannequin to have the user's body shape. Based on the same RBF, 3D clothes were resized and fitted the user's body. Finally, we applied the proposed surface correction approach to the resized clothes to pull several penetrated triangular faces out of the body.

3.1 Hardware Setting and Calibration

Our intention is to design a 3D virtual try-on system that is based on real captured data and also 3D scanner-based. This thesis is part of the whole system, which focuses on the 3D clothes modeling from real captured clothes data and its customization. We used one actual male mannequin to put on selected clothes for capturing clothes data. It had some benefits compared with capturing real dressed human, such as that mannequin could stand still while data capturing so that better data could be captured, fewer people involved in the data acquisition stage etc.

We used the newly released Kinect for Windows V2 as the capturing device since it has several advantages, such as low-cost price up to $199.00 [50], acceptable accuracy of depth data, available High Definition (HD) RGB sensor and powerful Software Development Kit (SDK) etc. Other 3D scanners like Whole Body Color 3D Scanner from *Cyberware* has better accuracy of depth data but the price is much higher up to $240,000 [52].

The Kinect for Windows V2 has three main components, which are RGB sensor, Time-of-Flight (ToF) depth sensor and microphone arrays. The horizontal field of view

(FOV) of depth sensor is 70° degrees, while the vertical FOV is 60°. To capture 360° views of the clothes, multiple Kinects could be used.

The KinectFusion [53] is the technique that uses one Kinect to scan the real world for the 3D model or scenario generation. The original model generated using KinectFusion usually has several drawbacks, such as huge data quantity, much noise and mis-registered color information etc. However, since it is able to register observed valid depth frame to the global model continuously and the distance between Kinect and object can be close, the well-aligned point clouds and quite detailed shape can be acquired.

We proposed to use four Kinects to capture four views' data of clothes. Since each Kinect got only one snapshot of the scenario, the data quantity was much smaller. And only four views' texture information would be mapped onto the 3D model of clothes, which made it easier for implementing texture mapping. The noise was also much less and easier to remove.

The comparison between using KinectFusion and Four Kinects for 3D model generation is listed in Table 3.

Combining the advantages of using KinectFusion with using Four Kinects for 3D model generation, we used the KinectFusion to capture aligned point clouds of the naked mannequin and the aligned point clouds of the user. And for clothes data capturing, four Kinects were used. Besides, to achieve a high 3D registration result from multiple point clouds of clothes, we used four Kinects to capture four views of the naked mannequin and aligned each view with the well-aligned point clouds of the naked

mannequin generated using KinectFusion to calculate the relation between every two Kinects.

The capturing environment setting and the calibration for each Kinect sensor are introduced in the following sections.

Table 3 Comparison between KinectFusion and Four Kinects for 3D model generation

Method	Texture Information	Noise	Point Data Quantity	Detailed Model
KinectFusion	Bad, not easy to do texture mapping	Too much noise inside model, not easy to remove	Million or higher	Much detailed
Four Kinects	Good, much easier to do texture mapping	Little noise, easy to remove	Around ten to hundred thousand	Smoother, not quite detailed

3.1.1 Multiple Kinects Setup

The depth range of Kinect for Windows V2 that ensures high quality operation is within [0.5~4.5] meters. The male mannequin has a height of 1.85 meters, chest of 1.01 meters, waist of 0.81 meters and hips of 0.91 meters. And it was supported by a chrome based. In order to capture whole body data of the mannequin, the distance between each Kinect and the mannequin was set to 2.10 meters. Besides, each Kinect was supported by the tripod, and the height of Kinect was set to 1.25 meters.

The size of the room we used for data capturing is 6m x 5m. The mannequin was placed at the center of the room, and kept still during data capturing. No special background was used. Two extra filament lightbulbs were added to illuminate the environment. The placement of four Kinects and capturing objects are displayed in Figure 3.1(Since it is difficult to take a normal image to display the capturing

environment due to the limitation of space, a panorama image is used instead).

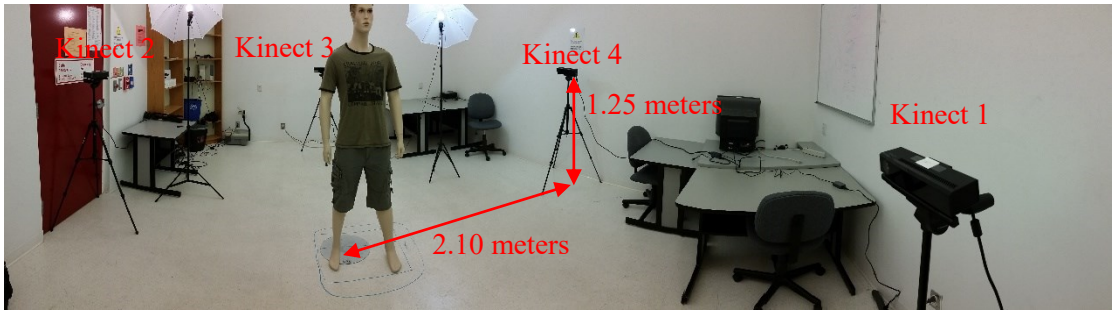


Figure 3.1 Capturing environment setting. The four Kinects are right angle to each other even though the panoramic picture does not show it due to the deformed space of panoramic views.

3.1.2 Camera Calibration

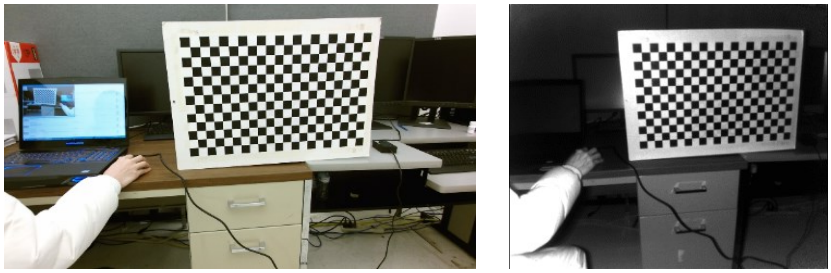
Kinect has two sensors: RGB sensor and depth sensor. From them, three kinds of images can be generated: RGB image, depth image and Infrared image. Depth image and Infrared image are generated from the same sensor. Each pixel value of a depth image is equal to depth data. In order to relate the 2D image with 3D world, we need to do camera calibration for each sensor.

In the camera calibration filed, Zhang's [86] method is used widely. It requires a simple pattern which has several feature points to be detected to calculate the intrinsic parameters. And it is easy to implement. A checkerboard consists of 12 x 20 squares was used as the pattern to do camera calibration in the experiment. The size of each square is 28mm x 28 mm.

By placing the checkerboard at the different position and orientation that should be visible to both sensors, 25 checkerboard images were captured, respectively. Since it was difficult to use the checkerboard's depth image to do camera calibration, we used the Infrared image instead. And in order to register point clouds with the color

information, stereo calibration between RGB sensor and depth sensor was implemented. Capturing of the RGB image and Infrared image were synchronized.

The resolution of color image is 1920 x 1080, while the Infrared image is 512 x 424. The captured RGB image and Infrared image are displayed in Figure 3.2.



(a) (b)

Figure 3.2 Captured checkerboard images of two Kinect built-in cameras (a) RGB camera image (b) Infrared camera image. The checkerboard image is used for calibration.

Camera calibration was implemented to calculate the intrinsic parameters, which consists of horizontal focal length, vertical focal length, horizontal principal point and vertical principal point. We used the Function *calibrateCamera* inside the OpenCV [88] to calculate the intrinsic parameters. The obtained intrinsic parameters for each Kinect sensor are listed in Table 4 and Table 5, respectively.

Table 4 Intrinsic Parameters of Kinect RGB Sensor

Intrinsic Parameters of RGB Sensor in pixels				
Sensor	f_{x_RGB}	f_{y_RGB}	O_{x_RGB}	O_{y_RGB}
Kinect 1	1073.42712	1068.65364	940.55748	531.24323
Kinect 2	1060.82164	1058.11634	1015.60097	534.45790
Kinect 3	1080.00537	1076.44882	928.65347	548.47715
Kinect 4	1123.59558	1118.93570	913.59859	540.25560

Table 5 Intrinsic Parameters of Kinect Infrared Sensor

Intrinsic Parameters of Infrared Sensor in pixels				
Sensor	f_{x_IR}	f_{y_IR}	O_{x_IR}	O_{y_IR}
Kinect 1	371.21951	369.969945	246.14200	199.23853
Kinect 2	365.39024	364.14578	268.24005	211.22079

Kinect 3	374.46472	373.23883	250.24772	198.72218
Kinect 4	386.51720	385.42395	248.92427	198.33406

3.1.3 Stereo Calibration

OpenCV [89] function *stereoCalibrate* was implemented to do the stereo calibration, from which the depth sensor space can be linked to the RGB sensor space by the homogeneous transformation matrix, which consisted of a normalized 3 x 3 rotation matrix and a 3 x 1 translation matrix.

Registering a point out of point clouds with color information lies in three steps, which are: (1) projecting one pixel in the 2D depth image into a 3D point in the 3D depth space using the intrinsic parameters of the depth sensor; (2) projecting the 3D point with respect to the 3D depth space to a 3D point with respect to the 3D RGB space using the extrinsic parameters between the depth sensor and the RGB sensor; (3) projecting the 3D point with respect to the 3D RGB space to a pixel in the 2D color image using the intrinsic parameters of the RGB sensor.

Let x and y represents the pixel coordinate of color and depth image. And the depth value for a given pixel is represented by $depth(x, y)$, then from the intrinsic parameters of the Infrared sensor, we have [87]

$$X_{IR} = \frac{(x - O_{x\_IR}) \times depth(x,y)}{f_{x\_IR}} \tag{3.1}$$

$$Y_{IR} = \frac{(y - O_{y\_IR}) \times depth(x,y)}{f_{y\_IR}} \tag{3.2}$$

$$Z_{IR} = depth(x, y) \tag{3.3}$$

From (3.1)-(3.3), a depth image is projected to 3D space. The coordinates are represented by (X_{IR}, Y_{IR}, Z_{IR}).

From stereo calibration, one rotation matrix $\boldsymbol{R}$ and one translation matrix $\boldsymbol{T}$ can be obtained, which are used to link Infrared sensor space $P_{IR}(X_{IR}, Y_{IR}, Z_{IR})$ with the RGB sensor space $P_{RGB}(X_{RGB}, Y_{RGB}, Z_{RGB})$.

$$P_{RGB}(X_{RGB}, Y_{RGB}, Z_{RGB}) = \boldsymbol{R} \times P_{IR} + \boldsymbol{T} \tag{3.4}$$

$$x = \frac{X_{RGB} \times f_{x\_RGB}}{Z_{RGB}} + O_{x\_RGB} \tag{3.5}$$

$$y = \frac{Y_{RGB} \times f_{y\_RGB}}{Z_{RGB}} + O_{y\_RGB} \tag{3.6}$$

From equations (3.5) and (3.6), a 3D point with respect to the 3D RGB space is projected to the 2D image, from which the point get its corresponding color information.

The extrinsic parameters for each Kinect sensor are listed in Table 6.

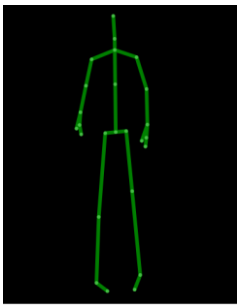
Table 6 Extrinsic Parameters of pairs of Kinect RGB and Infrared Sensor

Extrinsic Parameters of pair of RGB and Infrared Sensor		
Pairs	Rotation Matrix $\boldsymbol{R}$	Translation Matrix $\boldsymbol{T}$ (mm)
1	$\begin{bmatrix} 1.0000, & -0.0016, & -0.0021, \\ 0.0016, & 1.0000, & -0.0026, \\ 0.0021, & 0.0026, & 1.0000 \end{bmatrix}$	$\begin{bmatrix} 51.2367, \\ 0.0343, \\ -2.3366 \end{bmatrix}$
2	$\begin{bmatrix} 1.0000, & 0.0035, & -0.0074, \\ -0.0035, & 1.0000, & -0.0018, \\ 0.0074, & 0.0018, & 1.0000 \end{bmatrix}$	$\begin{bmatrix} 50.4508, \\ -0.3825, \\ 2.8473 \end{bmatrix}$
3	$\begin{bmatrix} 1.0000, & -0.0007, & 0.0070, \\ 0.0008, & 1.0000, & -0.0020, \\ -0.0070, & 0.0020, & 1.0000 \end{bmatrix}$	$\begin{bmatrix} 51.5690, \\ 0.8662, \\ -2.2158 \end{bmatrix}$
4	$\begin{bmatrix} 1.0000, & 0.0012, & 0.0075, \\ -0.0013, & 1.0000, & 0.0041, \\ -0.0075, & -0.0041, & 1.0000 \end{bmatrix}$	$\begin{bmatrix} 51.0942, \\ 0.0520, \\ -0.7606 \end{bmatrix}$

3.2 Clothes Extraction

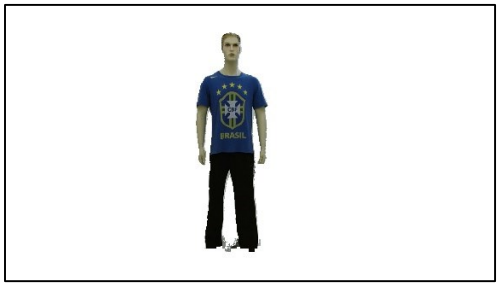
After the capturing environment setting and calibration for each Kinect, we

captured four views' data of the dressed mannequin using four Kinects. For each view, the original data consists of one color image and one depth image. Instead of dressing the mannequin with a single piece of clothing at a time, we dress the mannequin with one shirt and one trouser that have different color to save the data capturing time. To model 3D clothes from the captured data, extracting clothes out of the dressed mannequin is required. And since no special background was used in the experiment, it was not suitable to use background subtraction to extract clothes out of the dressed mannequin. Instead, in order to reduce the complexity of the clothes extraction, two steps are proposed: (1) extracting dressed mannequin out of the common background; (2) extracting clothes out of the dressed mannequin and separating into one shirt and one trouser.

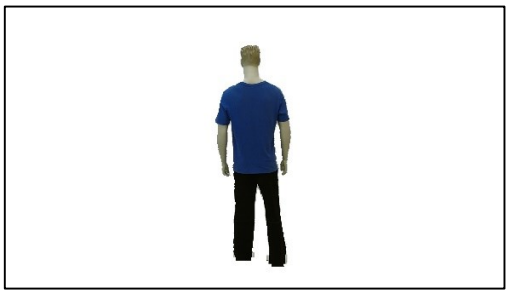


(a)

(b)



(c)

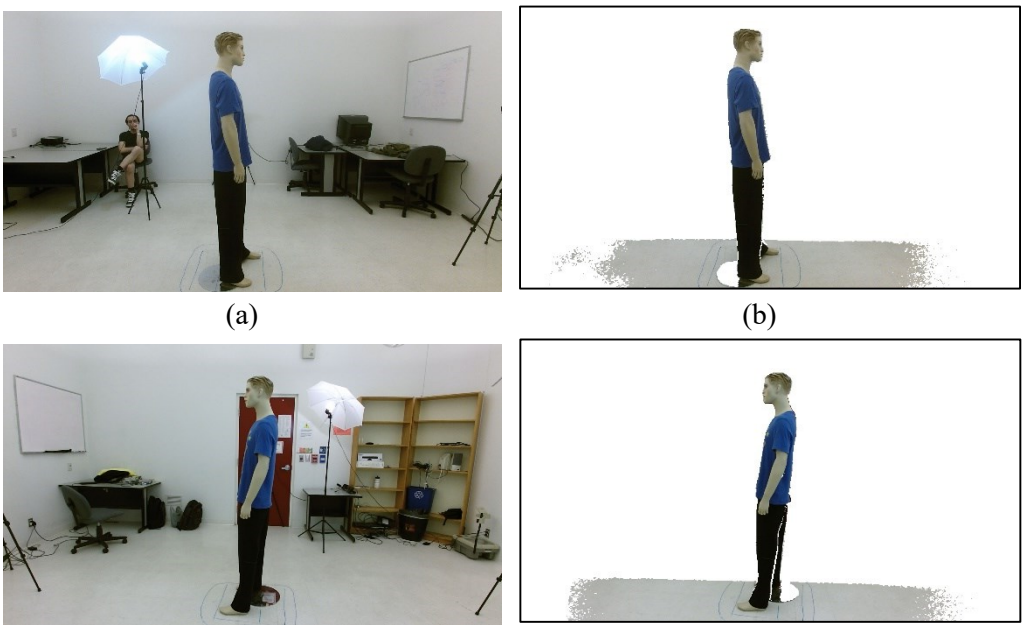


(d) (e)

Figure 3.3 (a) The Kinect skeleton of the mannequin in front view by using Kinect Body Tracking Function. (b) RGB image output of the front view Kinect. (c) The dressed mannequin extracted from the front view output by using Kinect Body Tracking Function. (d) RGB image output of the back view Kinect. (e) The dressed mannequin extracted from the back view output by using Kinect Body Tracking Function.

Kinect for Windows SDK provides function for body tracking, which is based on the tracked skeleton. The function was used to extract dressed mannequin out of the background in front and back view since it was able to track full skeleton of the mannequin in those two views. A full Kinect for Windows V2 skeleton consists of 25 joints. A Kinect skeleton captured from front view of the dressed mannequin is displayed in Figure 3.3 (a). The front and back view after using Kinect Body Tracking are displayed in Figure 3.3 (b) and (c), respectively.

However, for left and right view, the Kinect Body Tracking is usually not working since full skeleton of the mannequin cannot be obtained. But since the mannequin is the foreground object, which has smaller depth value than the background object, depth thresholding can be used to extract the dressed mannequin out of the background roughly. We utilized depth thresholding to extract the dressed mannequin out of the background in left and right view. The result is displayed in Figure 3.4.



(a) (b)

(c) (d)

Figure 3.4 Dressed mannequin extraction in left and right view: (a) RGB image output of the left view Kinect. (b) The dressed mannequin extracted from the left view output by using depth thresholding. (c) RGB image output of the right view Kinect. (d) The dressed mannequin extracted from the right view output by using depth thresholding.

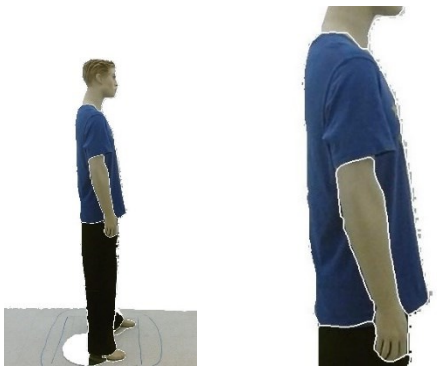
3.2.1 Color Segmentation

After extracting the dressed mannequin out of the background, we needed to extract clothes out of the dressed mannequin and separate it into one shirt and one trouser. We used color segmentation technique to solve this problem since it is much easier to do 2D segmentation compared with 3D segmentation. Color segmentation is the technique to partition one digital image into several homogeneous regions. Several methods of color segmentation have been designed over the past few years. We used one of them called JSEG [62] since the system [63] is available for research purpose and its performance is good. We used the default setting for some parameters, such as color quantization threshold, number of scales and region merge threshold etc. The segmented result is displayed in Figure 3.5.

After segmentation, every region is linked with a label, and every pixel has the same label fall into the same region. By finding the corresponding label value belongs to the shirt and the trouser, one shirt label set and one trouser label set can be obtained. Using the two sets, one shirt image and one trouser image were generated. The result is displayed in Figure 3.6 and Figure 3.7.

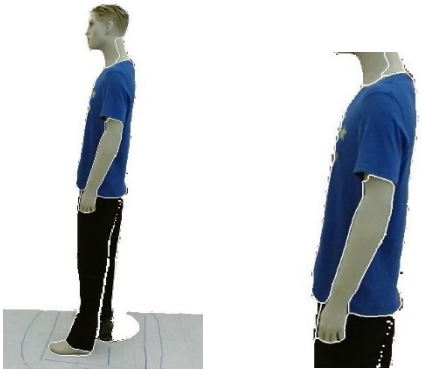
As you can see in Figure 3.8, the segmented result contains some noise. We manually select those regions and removed them. The new result is displayed in Figure 3.9 and Figure 3.10.

(a)



(b)

(c)



(d)

Figure 3.5 Four views of the dressed mannequin after applying the JSEG algorithm: (a) front view (b) left view (c) back view (d) right view

Figure 3.6 Four views of the extracted shirt after applying the JSEG algorithm: front view - left view - back view - right view (from left to right).

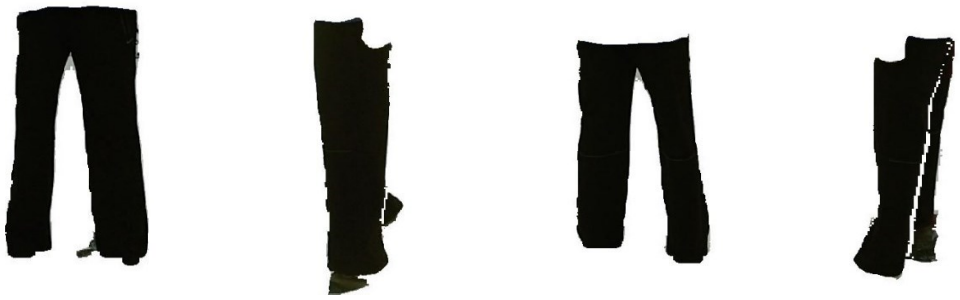


Figure 3.7 Four views of the extracted trouser after applying the JSEG algorithm: front view - left view - back view - right view (from left to right).

Figure 3.8 Front view of the extracted shirt and zoom of it. It shows some artifacts near the armpit area.

Figure 3.9 Four views of the extracted shirt after manually removing several artifacts: front view - left view - back view - right view (from left to right).

Figure 3.10 Four views of the extracted trouser after manually removing several artifacts: front view - left view - back view - right view (from left to right).

3.2.2 Region Filling

Due the block of left and right hands, a blank region (data missing region) was generated in left and right view. We used our proposed region filling technique based on neighborhood value to fill in the area. A 31 x 31 neighborhood that has a quite smooth transition of color value was manually selected first. Then, we manually selected the data missing region. For every pixel of the data missing area, it obtained the color value from the selected neighborhood randomly. The result is shown in Figure 3.11 and Figure 3.12.

(a) (b)

Figure 3.11 (a) Left view of the extracted shirt before and after region filling (b) Right view of the extracted shirt before and after region filling.

(a) (b)

Figure 3.12 (a) Left view of the extracted trouser before and after region filling (b) Right view of the extracted trouser before and after region filling.

3.3 3D Registration of Kinect Depth Data

We have five 3D data, four from four views of Kinect (four views of clothes data and four views of a naked mannequin) and one from Kinect Fusion (a naked mannequin) and we need to register them in one space.

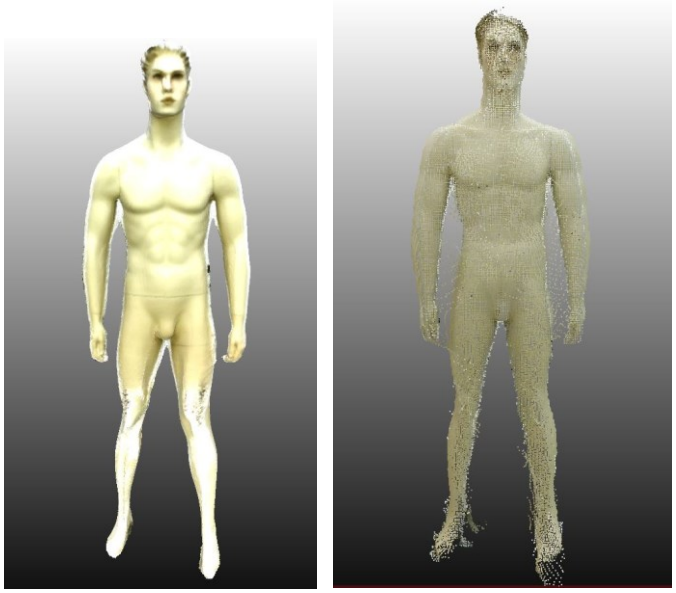
Based on the four types of data: (1) four segmented images of clothes (2) four views' depth data (3) intrinsic parameters of each Kinect RGB sensor and depth sensor (4) extrinsic parameters between RGB sensor and depth sensor, we could generate four views point clouds registered with color information. But since the four views point clouds were with respect to different coordinate systems, we used 3D registration technique ICP [68] to align them to create the 360° point clouds. The performance of ICP is related to the overlapping scale between two point clouds. Since the four views point clouds had small overlapping regions, we proposed a new approach to align multiple views of clothes, which was KinectFusion-based.

KinectFusion [53] is the technique that uses only one Kinect to scan real world to get the 3D model or scenario. And it utilizes ICP to continuously register valid depth

frame to the global model, which will generate a quite high quality well-aligned point clouds.

Our proposed approach to do 3D registration for clothes lied in four steps: (1) using KinectFusion to scan the naked mannequin to get the well-aligned point clouds (2) using four Kinects to capture four views of the naked mannequin (3) aligning each view of the naked mannequin with the KinectFusion clouds to obtain the relation between every two Kinects (4) aligning four view point clouds of clothes using the relation calculated in (3).

One 3D model of the naked mannequin is displayed in Figure 3.13 (a). And the alignment result from four views of the naked mannequin is displayed in Figure 3.13 (b).



(a) (b)

Figure 3.13 (a) The 3D model of the naked mannequin by using KinectFusion (b) A result of alignments of four sets of point clouds of the naked mannequin from four view Kinects.

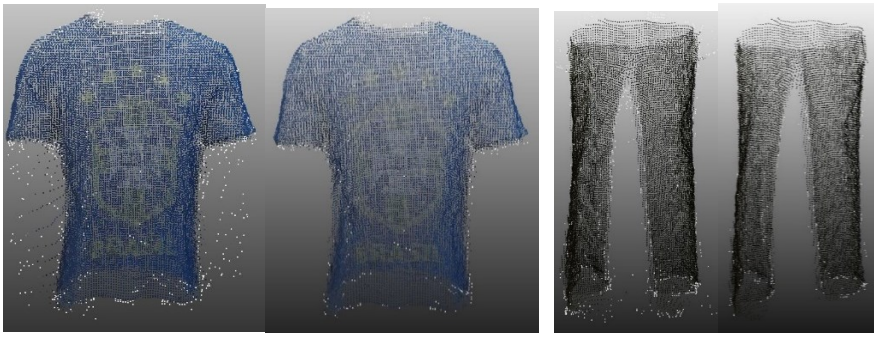
Then the transformation matrix for aligning each view to the naked mannequin model is listed in Table 7. The transformation matrix consists of one rotation matrix

and one translation matrix.

Based on the same transformation matrix, we aligned four views point clouds of shirt and trouser. Then we manually removed much noise in the aligned point clouds using one open source 3D model processing software named MeshLab [90]. The result is displayed in Figure 3.14.

Table 7 Transformation Matrix between one view Kinect to a KinectFusion data which is eventually used for aligning four views' point clouds.

External Parameters	
Kinect	Transformation Matrix to KinectFusion data
1	$\begin{bmatrix} 0.804483, & -0.154376, & -0.573563, & 0.986309 \\ -0.25353, & -0.962498, & -0.096545, & 0.263295 \\ -0.537149, & 0.223084, & -0.813452, & 0.933585 \\ 0, & 0, & 0, & 1 \end{bmatrix}$
2	$\begin{bmatrix} 0.608826, & -0.195429, & 0.768855, & -2.03587 \\ -0.0148175, & -0.971814, & -0.235284, & 0.662729 \\ 0.793165, & 0.131854, & -0.594561, & 0.192963 \\ 0, & 0, & 0, & 1 \end{bmatrix}$
3	$\begin{bmatrix} -0.819815, & -0.205271, & 0.534572, & -1.6118 \\ 0.220008, & -0.974799, & -0.0369131, & 0.219684 \\ 0.528677, & 0.0873479, & 0.844317, & -2.75818 \\ 0, & 0, & 0, & 1 \end{bmatrix}$
4	$\begin{bmatrix} -0.493953, & -0.167082, & -0.853284, & 1.48973 \\ -0.0492965, & -0.974403, & 0.219335, & -0.365104 \\ -0.86809, & 0.150405, & 0.473073, & -2.21439 \\ 0, & 0, & 0, & 1 \end{bmatrix}$



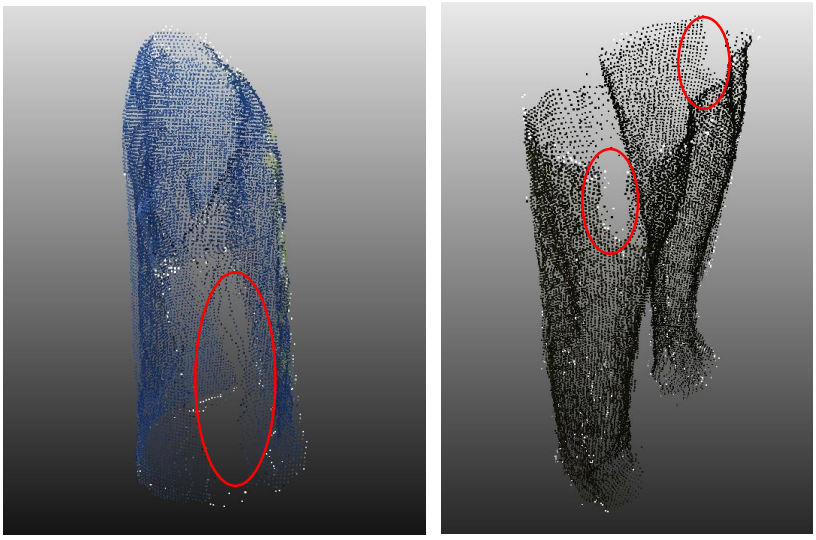
(a) (b)

Figure 3.14 (a) The point clouds of the shirt before and after manually removing some outliers (b) The point clouds of the trouser before and after manually removing some outliers.

3.4 Surface Reconstruction of Clothes

3.4.1 Poisson Surface Reconstruction

Based on the aligned point clouds of clothes, we used surface reconstruction technique to create 3D mesh out of them. The aligned shirt and trouser point clouds had several defects, such as several outliers and data missing etc. The data missing regions are displayed in Figure 3.15.

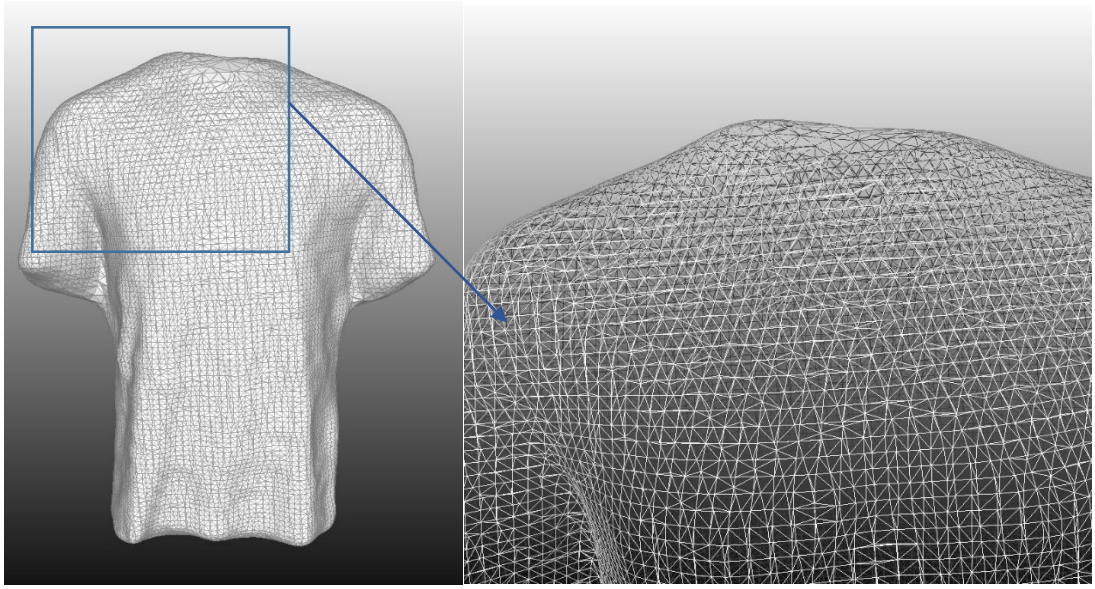


(a) (b)

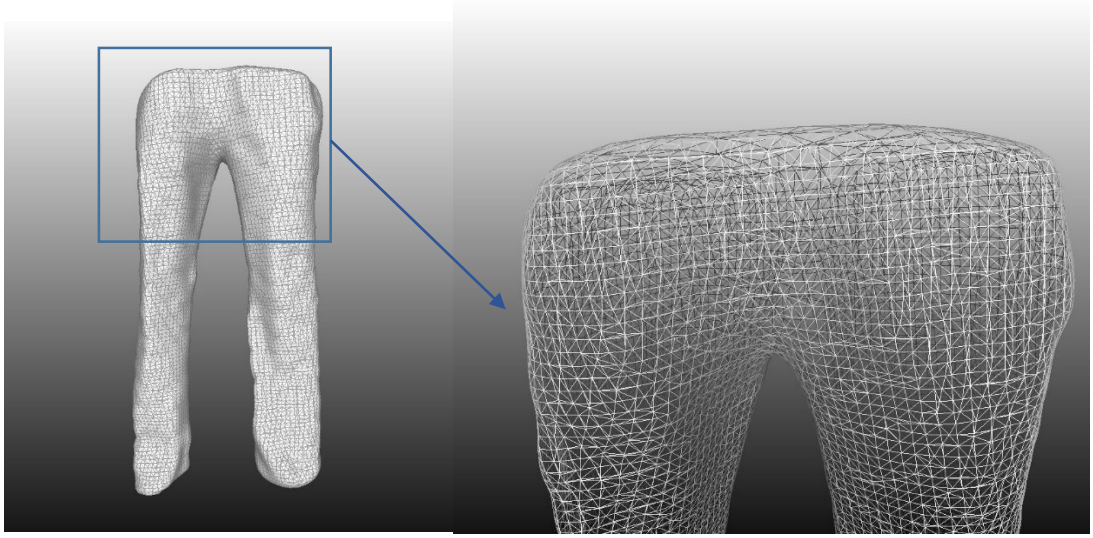
Figure 3.15 Multiple Kinect point clouds got registered in one space. (a) Missing points shown in the shirt point clouds (b) Missing points shown in the trouser point clouds.

Poisson surface reconstruction [77] was used in the experiment for surface reconstruction out of point clouds, since its performances for dealing with data missing and outliers are good. Besides, the implementation of the algorithm is not hard. Four kinds of parameters are used to determine the reconstruct model: Octree Depth, Solver Divide, Samples per Node and Surface offsetting. We set last three parameters as default values, which are 6 for Solver Divide, 1 for Samples per Node and 1 for Surface offsetting. The Octree Depth value has an impact on the smoothness of surface. A too

small value usually leads to the too smooth model. We set the Octree Depth Value to be 8 to reconstruct a high quality 3D mesh. The result is displayed in Figure 3.16.



(a)



(b)

Figure 3.16 Surface meshes created from point clouds using Poisson Surface Reconstruction. It produces the watertight surface, which requires post-processing. (a) Reconstructed shirt mesh (b) Reconstructed trouser mesh.

3.4.2 Mesh Post-processing

The model reconstructed using Poisson surface reconstruction is watertight, which makes the reconstructed shirt and trouser quite different from the real clothes. As shown

in Figure 3.17, in the captured images of clothes, regions (A, B, C, D, E, F, and G) are all open, but the corresponding regions (a, b, c, d, e, f, and g) in the reconstructed mesh are all closed.

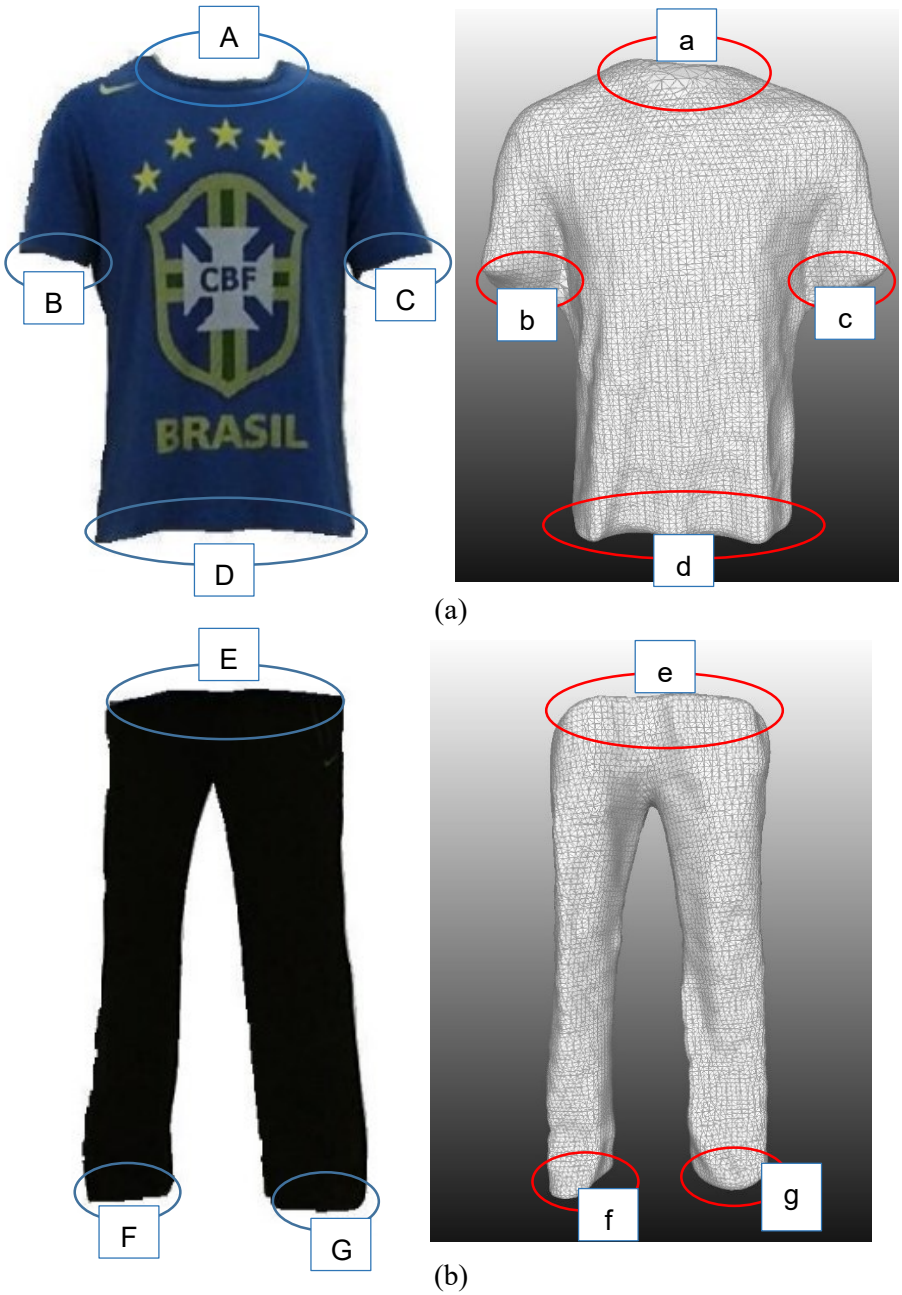


Figure 3.17 Comparison between the captured clothes images and the reconstructed clothes model from Poisson Surface Reconstruction: (a) Shirt (b) Trouser.

Several steps are required to process the watertight 3D model of clothes to match the clothes region in the image. Our proposed approach was divided into six steps:

1. Using Kd-Tree to do rough texture mapping for the clothes.

2. Selecting several points on the mesh manually and the point selections are in order.
3. Deleting triangular faces along the paths formed from the selected points.
4. Removing other unconcerned components.
5. Finding the boundary edges of the mesh left.
6. Creating several triangular faces to link boundary edges and the selected points.

It is quite difficult to accurately remove regions not belong to clothes if the model is without any texture information. Poisson Surface Reconstruction method is not able to transfer the color information of the aligned point clouds to the reconstructed model. And the relation between the reconstructed model and each view was lost. Since Kd-Tree is the structure that can be used for multi-dimensional searching, we used it to calculate the relation between the reconstructed model and four views point clouds. And based on the calculated relation, we applied color information from the point clouds to the reconstructed model. The method is introduced in the following and the result is displayed in Figure 3.18:

1. For current vertex in the reconstructed model, using Kd-Tree to find its nearest point from the four views point clouds and calculating the nearest distance.
2. Comparing the nearest distance with a preset threshold. If it is less, then applying the color information from the corresponding view to the vertex. Otherwise, none of color information will mapped on the vertex
3. Repeating step 1 and 2 until all vertices of the reconstructed model are processed.

Based on the rough texture mapping for the model of clothes, we manually selected several points to define the region that would be deleted. 247 points and 162 points were selected on the shirt model and trouser model, respectively. And the result is displayed in Figure 3.19. The selected points for shirt model and trouser model were put into four sets and three sets, respectively.

(a) (b)

Figure 3.18 Applying roughly texture mapping to the clothes mesh based on Kd-Tree: (a) Shirt mesh with texture information. (b) Trouser mesh with texture information

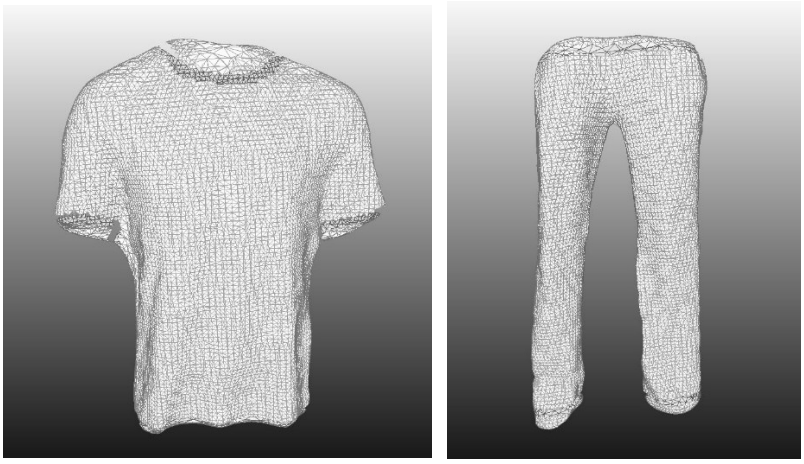
(a) (b)

Figure 3.19 (a) Selecting four sets of points on the shirt mesh to define four boundaries of the shirt (b) Selecting three sets of points on the trouser mesh to define three boundaries of the trouser.

Our proposed approach for deleting triangular faces along the paths formed from

the selected points can be divided into three steps and the result is displayed in Figure 3.20:

1. For every triangular face of the clothes model, calculating its centroid.
2. For each selected points set, creating several lines to link every two neighbor points so that one boundary defined by this set is formed. And for every line, calculating its center point.
3. For every centroid, calculating its distance to all center points. If it is less than a preset threshold, then deleting this triangular face.



(a) (b)

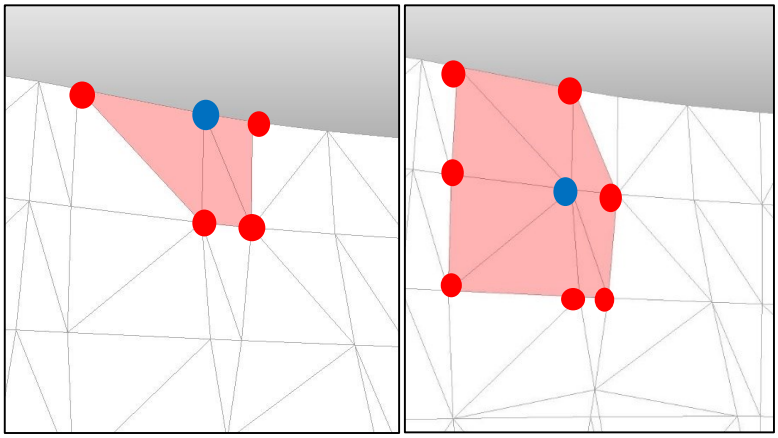
Figure 3.20 (a) Removing several shirt triangular faces based on the selected four sets of points (b) Removing several trouser triangular faces based on the selected three sets of points.

After deleting several triangular faces along the paths formed from the selected points, the shirt mesh was divided into five separated components and the trouser mesh was divided into four separated components. For each component, it was connected inside. We used MeshLab to delete those unconcerned components. Besides, in order to have the much smooth boundary edges, we manually deleted several triangular faces that were constructed from three edge vertices. And the result is displayed in Figure 3.21.

Then we searched the boundary edges of the mesh left so that the repair could be possible. The difference between one edge vertex and inside vertex is displayed in Figure 3.22.

(a) (b)

Figure 3.21 (a) Removing some components not belong to the shirt mesh (b) Removing some components not belong to the trouser mesh.



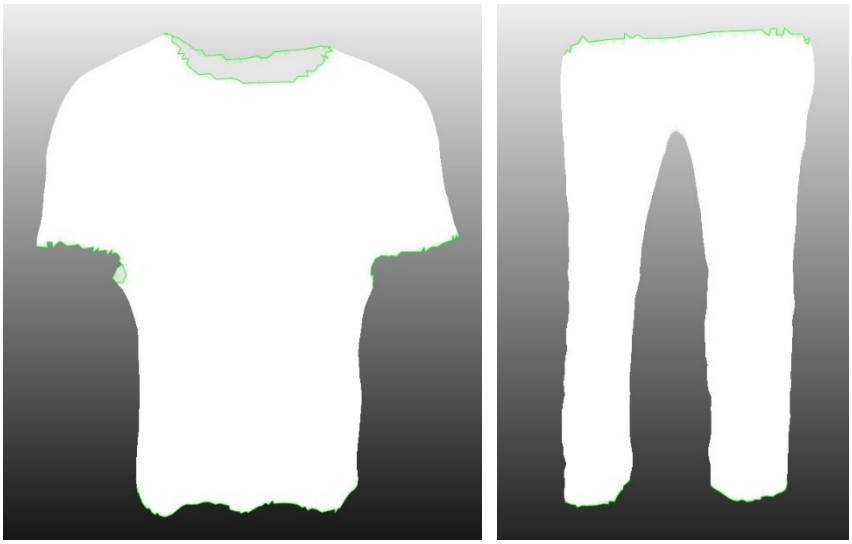
(a) (b)

Figure 3.22 Comparison between the edge vertex and the inside vertex: (a) Edge vertex shown in blue color is surrounded by several triangular faces that cannot form a closed path (b) Inside vertex shown in blue color is surrounded by several triangular faces that can form a closed path.

As you can see from Figure 3.22 (a), for the blue edge vertex, three triangular faces are constructed from it. And they cannot from a closed path. However, for the inside vertex, all triangular faces constructed from it can form a closed path. Our approach for

finding boundary edge was based on this idea and it can be divided into three steps and the boundary edges are displayed in Figure 3.23:

1. For current vertex, finding all triangular faces constructed from it
2. Based on the connectivity of the found faces, determining if current vertex is edge vertex or inside vertex
3. Repeating step 1 and 2 until all vertices are processed.



(a) (b)

Figure 3.23 (a) Four boundaries of the shirt mesh shown in green color (b) Three boundaries of the trouser mesh shown in green color.

The boundary vertices for shirt and trouser were put into four and three sets, respectively. Then a proposed sorting algorithm was applied to each set so that the boundary vertices in each set were in order. The proposed approach can be divided into four steps:

1. Choosing a starting edge vertex.
2. For current chosen vertex, finding all triangular faces constructed from it, and finding two other edge vertices from those faces.
3. Determining if the other two edge vertices have been sorted or not. Three cases

occur: first case, two vertices have not been sorted, then choosing one of them as the next sorting one; second case, only one vertex have not been sorted, then choosing it to the next sorting one; third case, two vertices have been sorted, then sorting for the set is done.

4. Repeating step 2 and 3.

The selected points and boundary edges are displayed in Figure 3.24 and Figure 3.25 with respect to shirt and trouser.

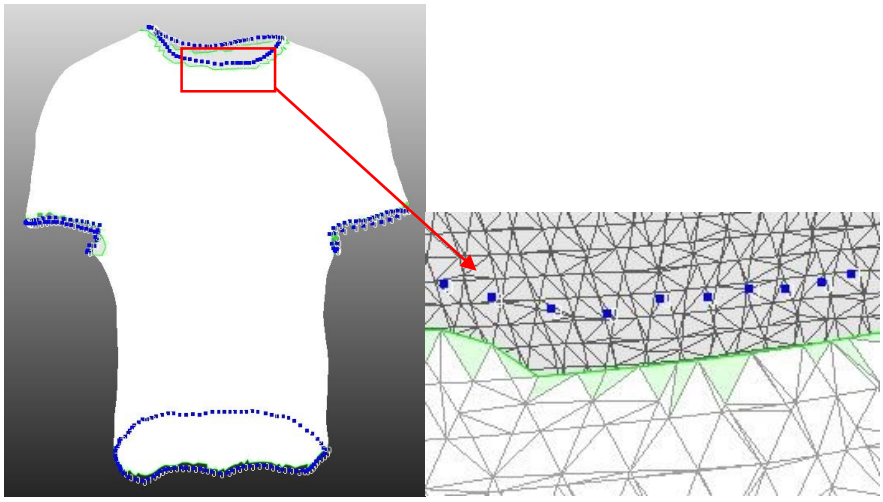


Figure 3.24 The selected four sets of points (in blue) and the four boundaries (in green) of the shirt mesh.

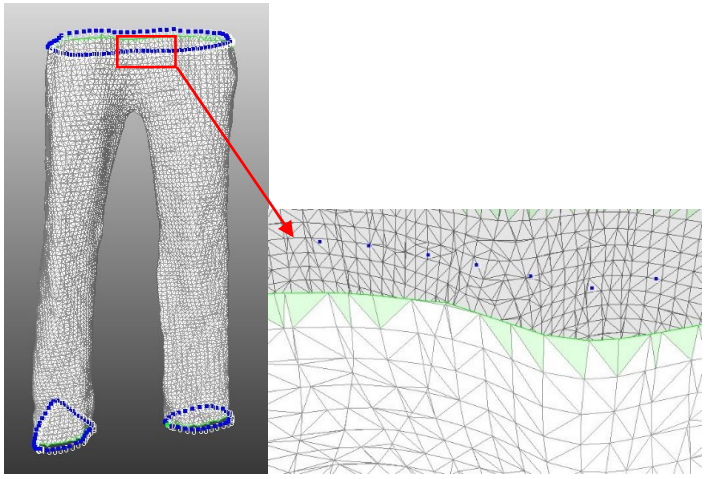


Figure 3.25 The selected three sets of points (in blue) and the three boundaries (in green) of the trouser mesh.

After sorting each set, several triangular faces that linked boundary edged with

selected points were created and attached to the mesh left. The proposed approach for adding new triangular faces is introduced in the following: Let $\mathrm{A} = \{a_1, a_2, \ldots, a_m\}$ represents the one set of the selected points, and $\mathrm{B} = \{b_1, b_2, \ldots, b_n\}$ represents the corresponding set of the boundary edges, usually $\mathrm{n} > \mathrm{m}$.

1. For a_i, finding one nearest boundary point from B and applying its value to e_i , a new triangular face formed by a_i, e_i and a_{i+1} is created.
2. Repeating step 1 until $\mathrm{i} = \mathrm{m}$.
3. For e_i and e_{i+1}, finding their index in B, let's say $e_i = b_j, e_{i+1} = b_k$. Several new triangular faces are formed by b_s, b_{s+1} and a_{i+1} $(\mathrm{s} = 1, \ldots, \mathrm{k} - 1)$.
4. Repeating step 3 until $\mathrm{i} = \mathrm{m}$.

The result is shown in Figure 3.26.

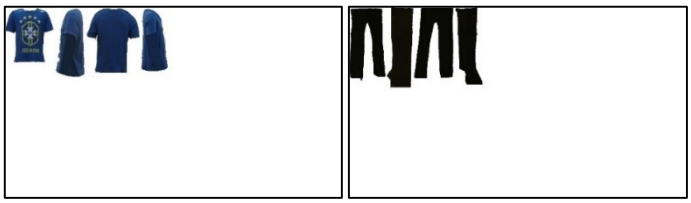
(a) (b)

Figure 3.26 (a) The shirt mesh after mesh post-processing (b) The trouser mesh after mesh post-processing.

3.5 Texture Mapping of Clothes

After the surface reconstruction and mesh modification, we obtained the mesh without texture information. We should apply texture mapping technique to get the more realistic shirt and trouser. In order to remove the effect of different illumination

condition for each view, we used Adobe Photoshop to adjust the exposure difference and color balance between those images. Then we integrate 4 views of images into one image to be the texture map, which is shown in Figure 3.27.

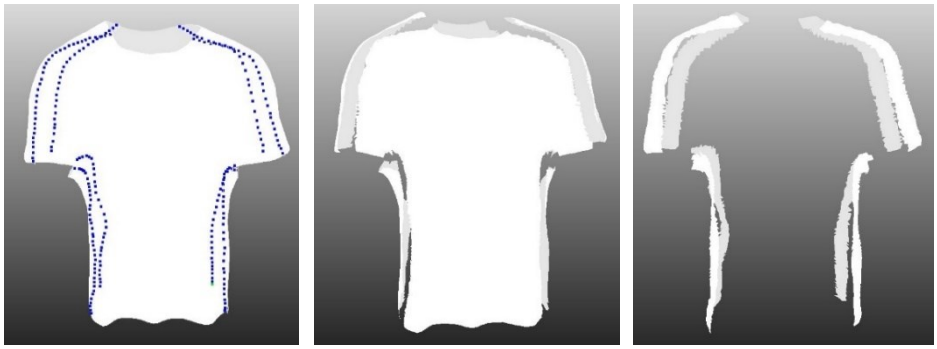


(a) (b)

Figure 3.27 (a) The shirt texture map (b) The trouser texture map.

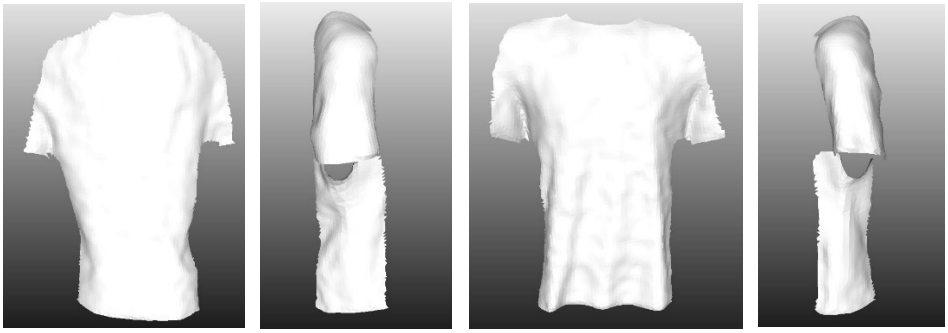
We segmented the mesh into four parts with respect to each view. Several points were manually selected on the mesh to define the intersection between every two views. 243 points were selected on the shirt mesh and 271 points were selected on the trouser mesh. Then we applied the algorithm used for deleting triangular faces along selected points to the clothes mesh. The algorithm has been introduced in section 3.4.2. The result is displayed in Figure 3.28, Figure 3.29, Figure 3.30 and Figure 3.31.

Shirt and trouser were segmented into four parts. Then for every part, we projected it from 3D space to 2D space to get its corresponding texture information. The idea has been introduced in section 3.1.3. The result is displayed in Figure 3.32 and Figure 3.33.



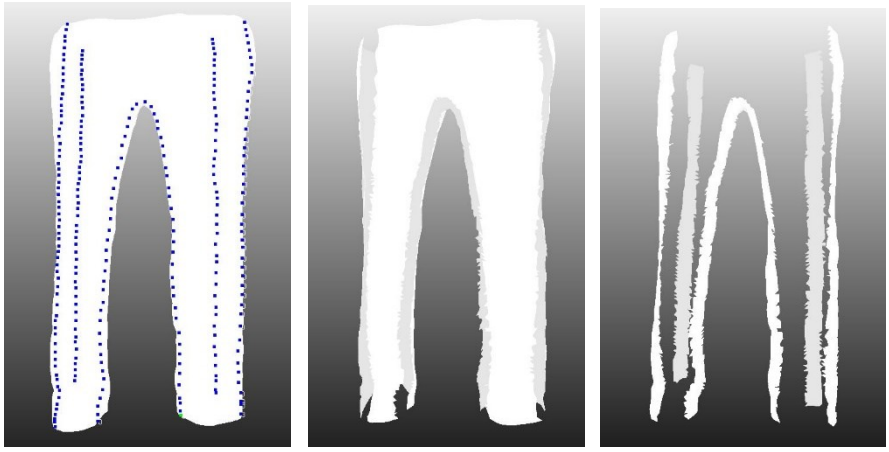
(a) (b) (c)

Figure 3.28 Shirt segmentation: (a) Selecting eight sets of points on the shirt mesh (b) Several parts after applying the 3D segmentation method (c) Eight strips formed from the eight sets of points.



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

Figure 3.29 Four main parts of shirt mesh after integrating eight strips to the corresponding views: (a) front part (b) left part (c) back part (d) right part.



(a) (b) (c)

Figure 3.30 Trouser segmentation: (a) Selecting five sets of points on the trouser mesh (b) Several parts after applying the 3D segmentation method (c) Five strips formed from the five sets of points.

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

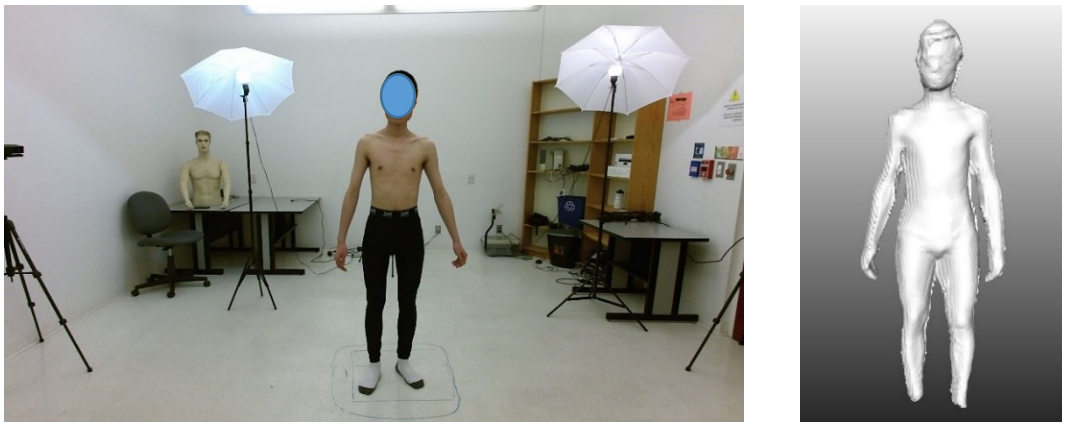
Figure 3.31 Four main parts of the trouser mesh after integrating five strips to the corresponding views: (a) front part (b) left part (c) back part (d) right part

Figure 3.32 The 3D textured model of the shirt.

Figure 3.33 The 3D textured model of the trouser.

3.6 Resizing of Clothes

We uses KinectFusion to scan one user dressed in tight clothes to obtain the point clouds. Then applying Poisson surface reconstruction to the user's point clouds to create one 3D model without any texture information. The male user dressed in tight trouser and his corresponding 3D model from KinectFusion are displayed in Figure 3.34.



(a) (b)

Figure 3.34 (a) One user dressed in the tight clothes (b) his 3D model from KinectFusion.

We used Radial Basis Function (RBF) for clothes customization. And the approach to resize the 3D clothes to fit user's body size can be divided into two steps: (1) deforming the naked mannequin to have the similar body shape of user using RBF (2) resizing the 3D clothes to fit the user's body size using the same RBF.

Using RBF for resizing requires two data groups: control points and target points. Several control points should be selected on the naked mannequin's body, and corresponding target points should be selected on the user's body.

Both 3D model of naked mannequin and user were reconstructed under two steps: (1) using KinectFusion to scan the body to get corresponding point clouds (2) using Poisson surface reconstruction to create mesh out of the point clouds. Then we manually removed head, neck, hands, feet of the naked mannequin and user since those parts were not related to the clothes customization. The result is displayed in Figure 3.35.

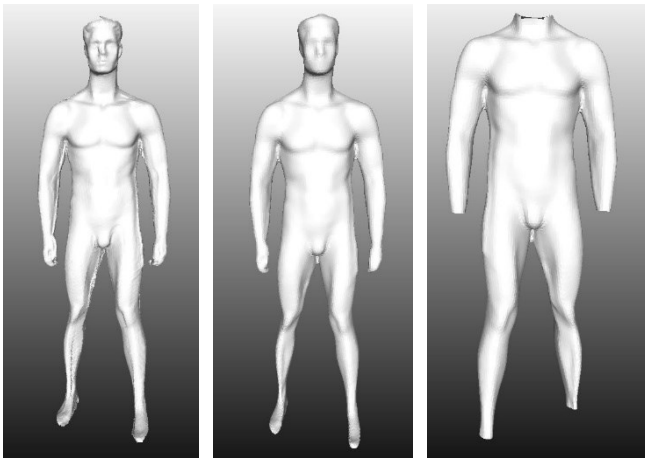
We aligned the naked mannequin with the user's model using ICP so that they were in the same space. The aligned result is displayed in Figure 3.36.

Then we used mouse to manually select control points on the naked mannequin and corresponding target points on the user's 3D model. 200 points were selected on the two models, respectively. Our selections of control points and target points are displayed in Figure 3.37 and Figure 3.38.

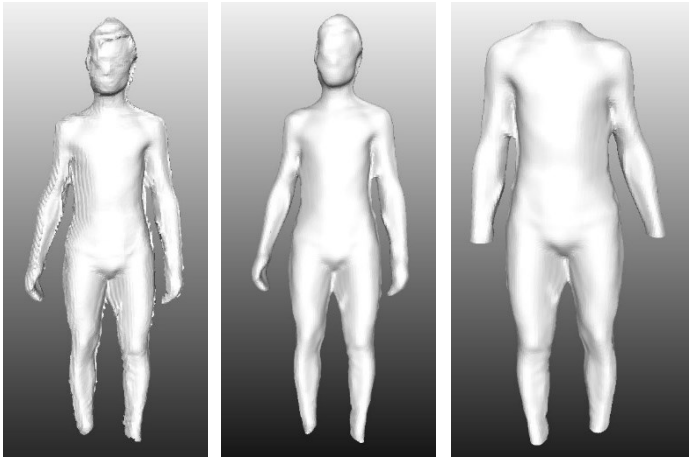
Usually, the selection of control points or target points should be feature points. But since it is difficult to find feature points on the body, our selection of points were to cover most parts of the body.

After selecting control points and target points, the naked mannequin can be deformed to have the similar shape of user's body. The result is displayed in Figure 3.39.

Then based on the same deformation functions, shirt and trouser can be resized to fit the user's body size. The result is displayed in Figure 3.40 (b).



(a)



(b)

Figure 3.35 (a) Left to Right: The raw 3D model of the naked mannequin using KinectFusion; Applying Poisson Surface Reconstruction algorithm to the point clouds of the naked mannequin from KinectFusion; Removing several parts of the 3D model of the naked mannequin, such as head, neck, hands, and feet.(b) Left to Right: The user's raw 3D model using KinectFusion; Applying Poisson Surface Reconstruction algorithm to the user's point clouds from KinectFusion; Removing several parts of the user's 3D model, such as head, neck, hands, and feet.

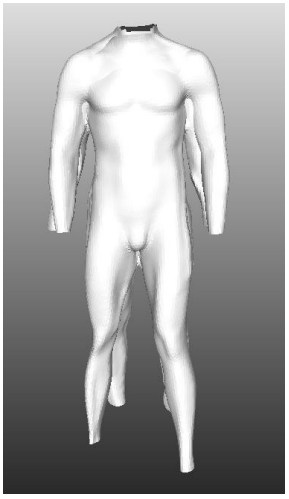


Figure 3.36 The alignment result of two 3D human models using ICP. One is the naked mannequin, and the other is the user.

As shown in Figure 3.40 (b), some vertices penetrating into the body, our approach to surface correction that pulls points outside of the body is described in the following.

1. For every body face, calculating its centroid.
2. For ever vertex in clothes, using Kd-Tree to find its nearest centroid, and computing the corresponding face normal (all face normals of the body are pointing outside). Then creating a vector pointing from the centroid to the vertex. Calculating the dot product between the vector and the normal. If the result is less than 0, current vertex is defined as the penetrated vertex, we projected it along the vector and set a distance value to the body so the new vertex coordinate was generated.

The result is displayed in Figure 3.40 (c).

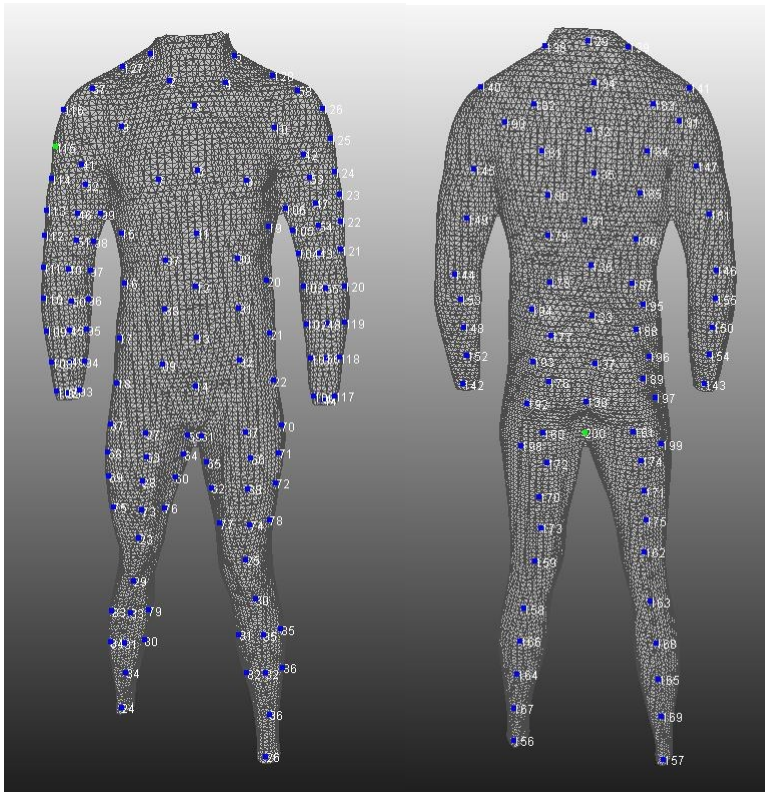


Figure 3.37 Selecting two hundred points on the naked mannequin as the control points for applying RBF for deformation.

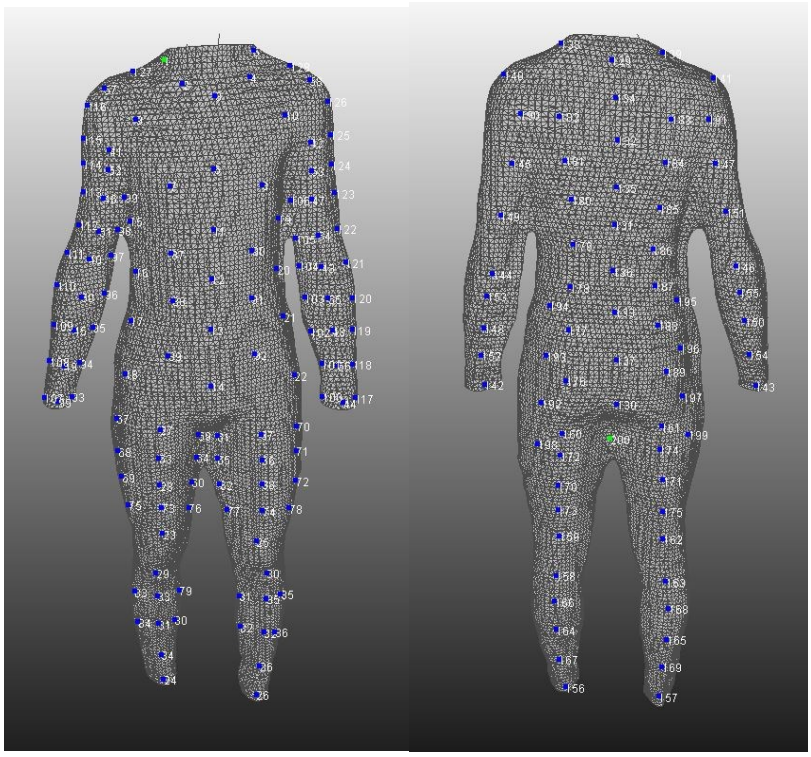
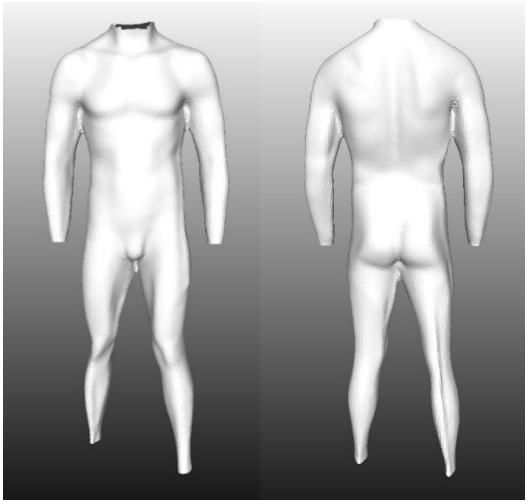
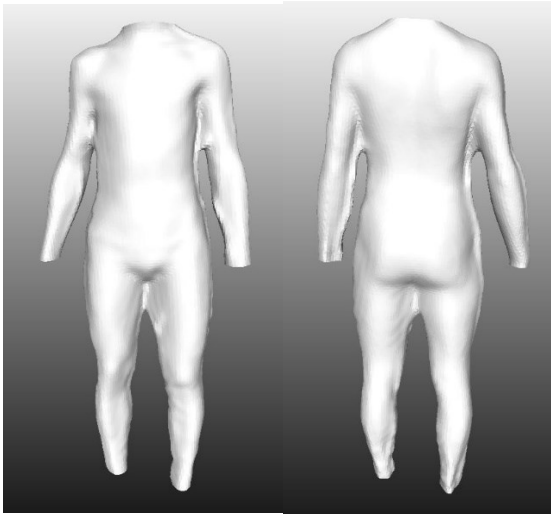


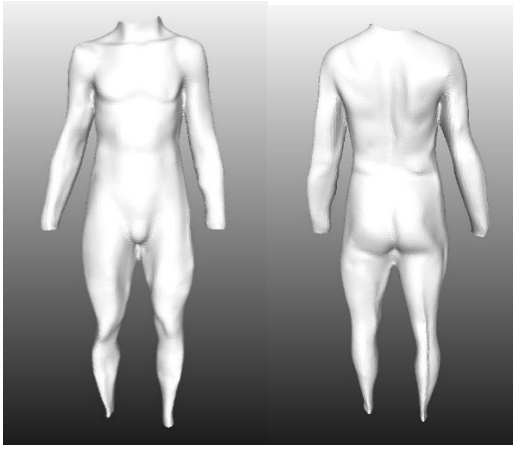
Figure 3.38 Selecting two hundred points on the user as the target points for applying RBF for deformation.



(a)



(b)



(c)

Figure 3.39 Deforming the naked mannequin to have the user's body size and shape (all models are displayed without head, neck, hands, and feet): (a) The un-deformed 3D model of the naked mannequin (b) The user's 3D model (c) The deformed 3D model of the naked mannequin.

(a)

(b)

(c)

Figure 3.40 Resizing the clothes to fit user's body size and shape: (a) the reconstructed 3D model of the naked mannequin with the 3D textured model of the clothes (b) the user's 3D model with the resized 3D textured model of the clothes before applying Surface Correction (c) the user's 3D model with the resized 3D textured model of the clothes after applying Surface Correction.

Chapter 4. Result and Validation

The chapter demonstrates the experiment result at first, and the validation is done to verify the feasibility, efficiency and stability of the proposed approach. Evaluation and discussion are introduced to describe the characteristics, advantage and limitation of the system.

4.1 Experiment Result

Our system is based on multiple views, in the following, we present the result with respect to front, back, left, and right view. And in the experiment, three types of shirts, three type of trousers and one skirt were used:

- Shirt 1: short sleeves blue shirt, which has several complicated patterns in the front surface such as stars, characters, and so on.
- Shirt 2: short sleeves dark green shirt, which also has complex patterns in the front surface.
- Shirt 3: long sleeves dark green shirt, which has two white and parallel lines in the front surface.
- Trouser 1: long black trouser.
- Trouser 2: short grey pants.
- Trouser 3: long dark blue jeans.
- Skirt: dark blue skirt.

Four groups are created based on the shirts, trousers, and skirt:

- Group 1: Shirt 1 and Trouser 1.

- Group 2: Shirt 2 and Trouser 2.
- Group 3: Shirt 3 and Trouser 3.
- Group 4: Shirt 1 and Skirt.

One actual male mannequin put on each group of clothes during data acquisition. So four groups of clothes were acquired and reconstructed in the experiment.

Figure 4.1, Figure 4.2, Figure 4.3 and Figure 4.4 demonstrate the procedure of extracting the dressed mannequin from the background, extracting clothes from the dressed mannequin and separating them into one shirt and one trouser, and region filling to the data missing area.

Figure 4.5 demonstrates the procedure to align four views' point clouds of the naked mannequin to the model of the naked mannequin captured using KinectFusion.

Figure 4.6 demonstrates the procedure to align four views' point clouds of the shirt and the trouser, and to apply Poisson Surface Reconstruction to create the watertight mesh.

Figure 4.7 demonstrates the procedure to apply Mesh Post-processing to the watertight mesh to create the more realistic clothes mesh.

Figure 4.8 demonstrates the procedure to apply texture mapping for the shirt mesh and the trouser mesh.

Figure 4.9 demonstrates the procedure to deform the naked mannequin to have the user's body size and shape using RBF.

Figure 4.10, Figure 4.11, Figure 4.12 and Figure 4.13 demonstrate the procedure to resize the four combinations of clothes to fit to the user.

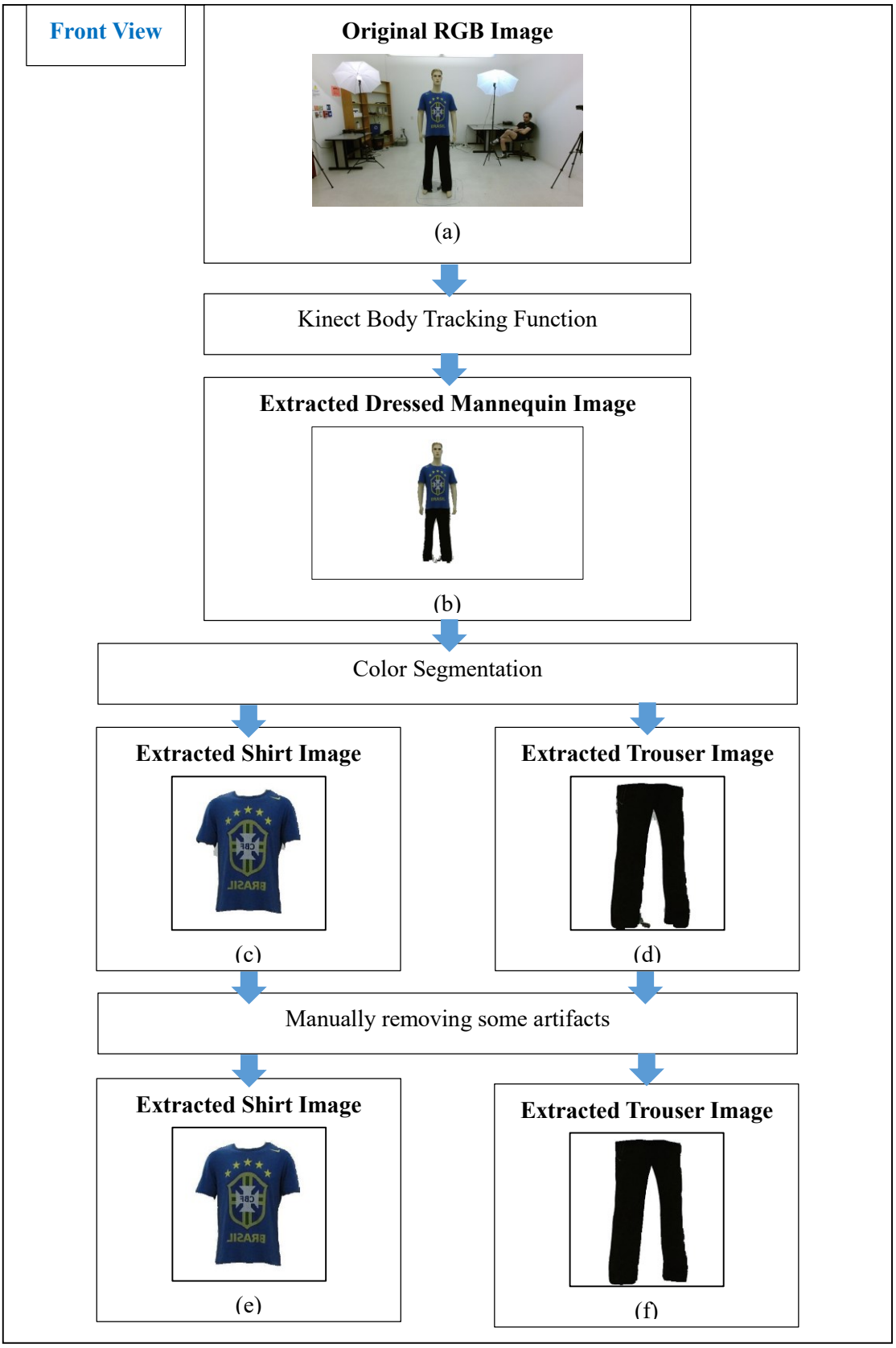


Figure 4.1 Front view clothes extraction: (a) color image from RGB sensor (b) extracted dressed mannequin image using Kinect Body Tracking (c) extracted shirt image using JSEG (d) extracted trouser image using JSEG (e) shirt image after manually removing some artifacts (f) trouser image after manually removing some artifacts.

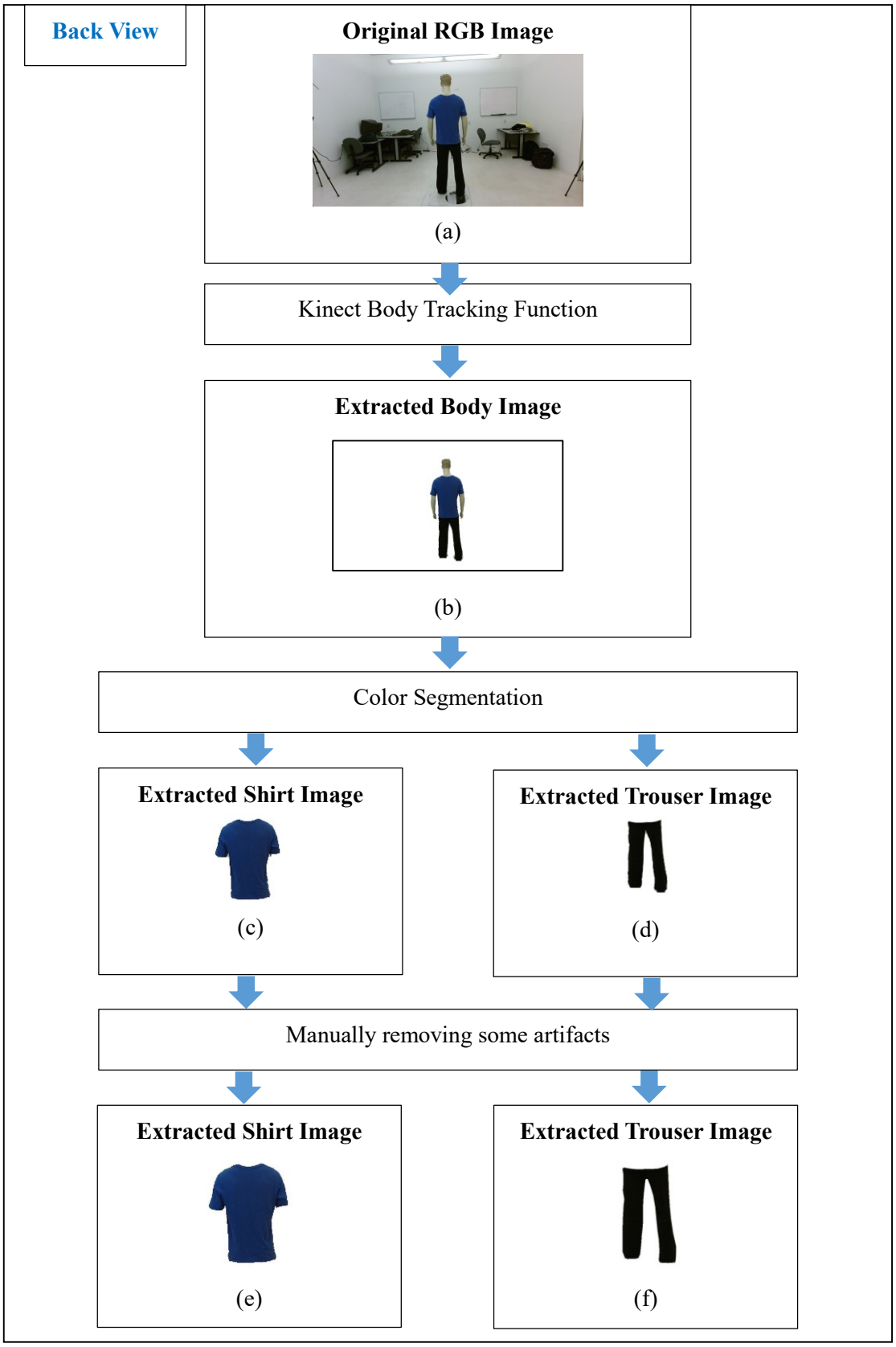


Figure 4.2 Back view clothes extraction: (a) color image from RGB sensor (b) extracted dressed mannequin image using Kinect Body Tracking (c) extracted shirt image using JSEG (d) extracted trouser image using JSEG (e) shirt image after manually removing some artifacts (f) trouser image after manually removing some artifacts

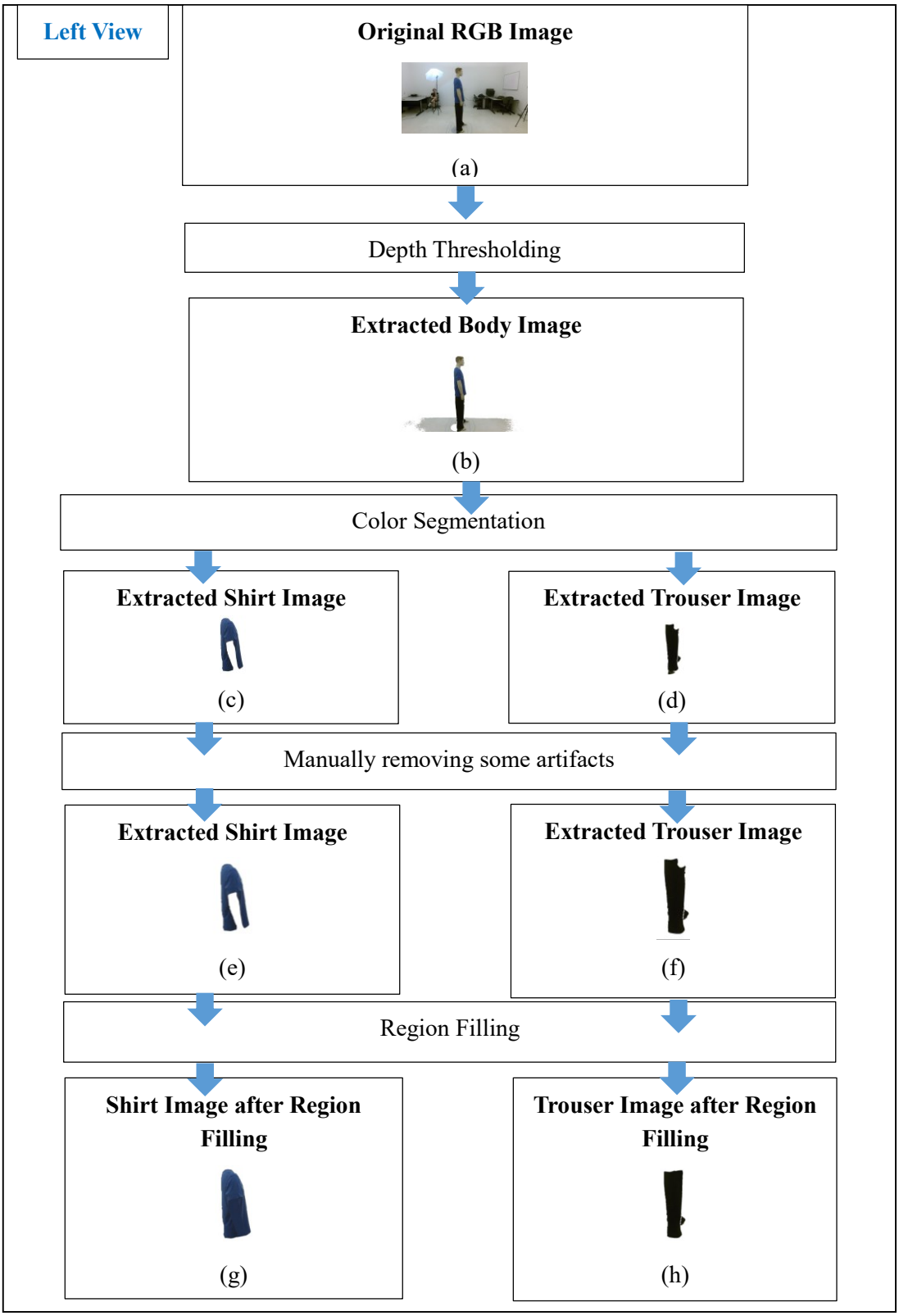


Figure 4.3 Left view clothes extraction: (a) color image from RGB sensor (b) extracted dressed mannequin image using Depth Thresholding (c) extracted shirt image using JSEG (d) extracted trouser image using JSEG (e) shirt image after manually removing some artifacts (f) trouser image after manually removing some artifacts (g) shirt image after region filling (h) trouser image after region filling.

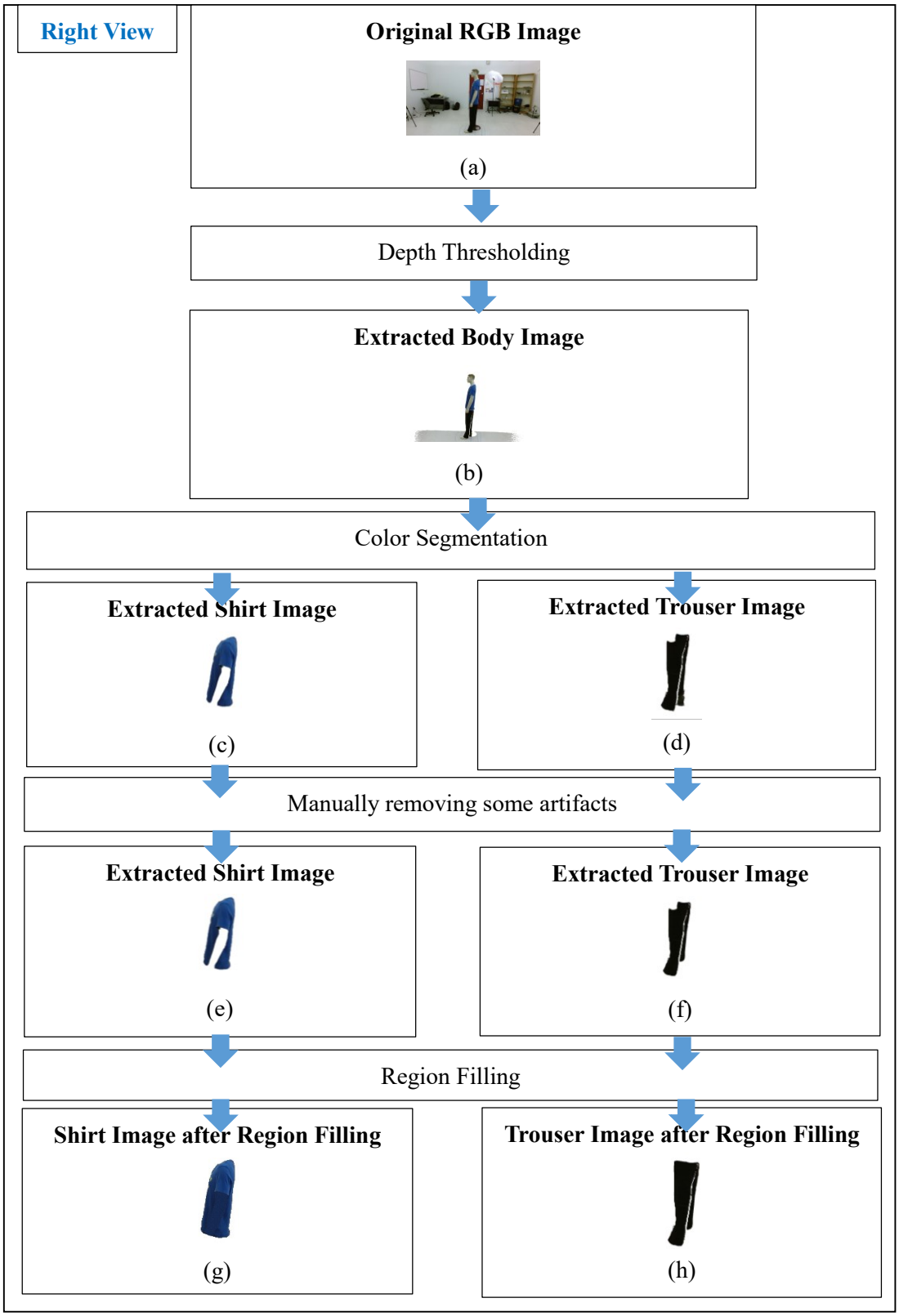


Figure 4.4 Right view clothes extraction: (a) color image from RGB sensor (b) extracted dressed mannequin image using Depth Thresholding (c) extracted shirt image using JSEG (d) extracted trouser image using JSEG (e) shirt image after manually removing some artifacts (f) trouser image after manually removing some artifacts (g) shirt image after region filling (h) trouser image after region filling

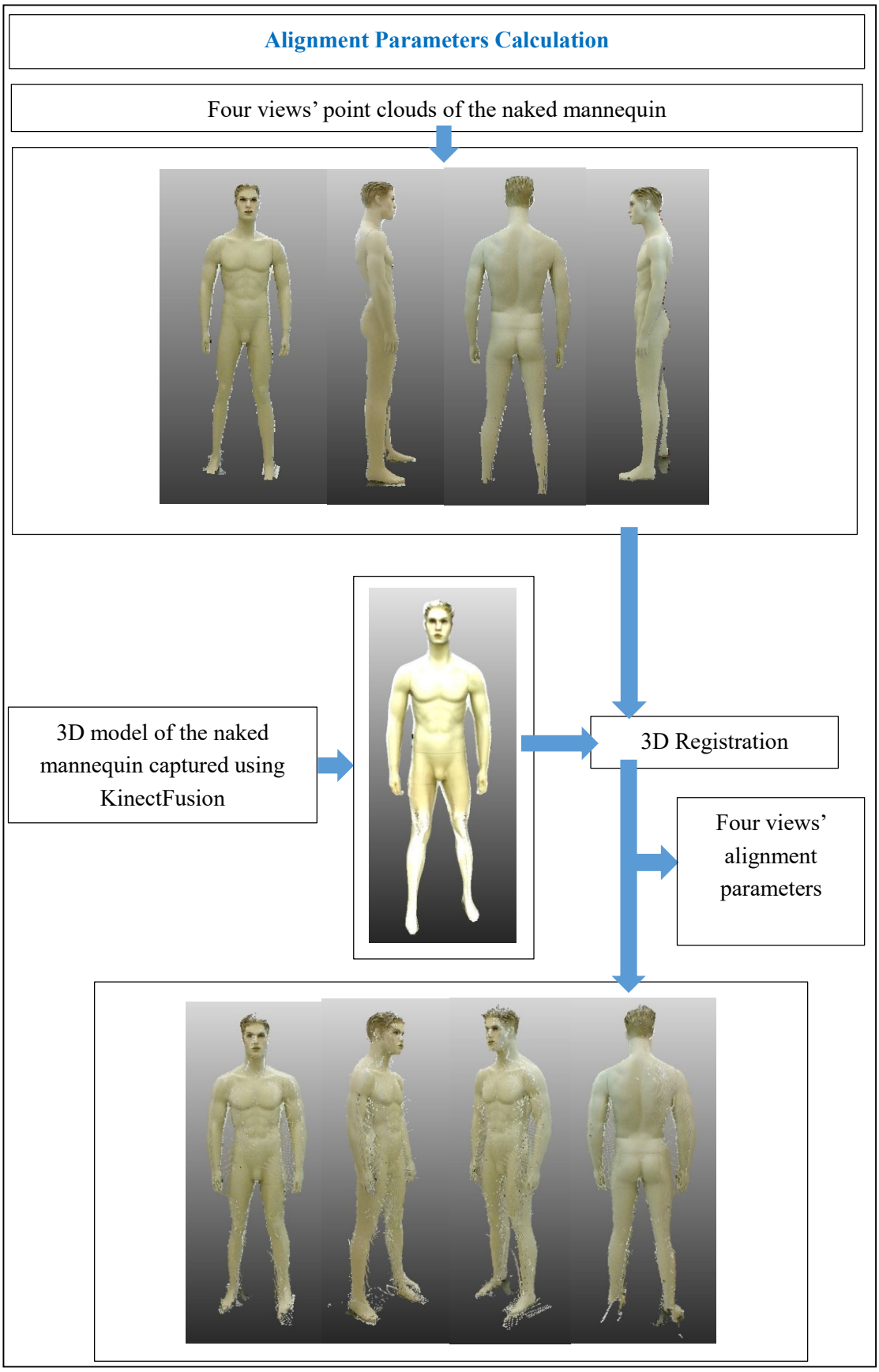


Figure 4.5 Four views of naked mannequin alignment (a) Four views point clouds of the naked mannequin (b) 3D model from KinectFusion (c) Aligned point clouds

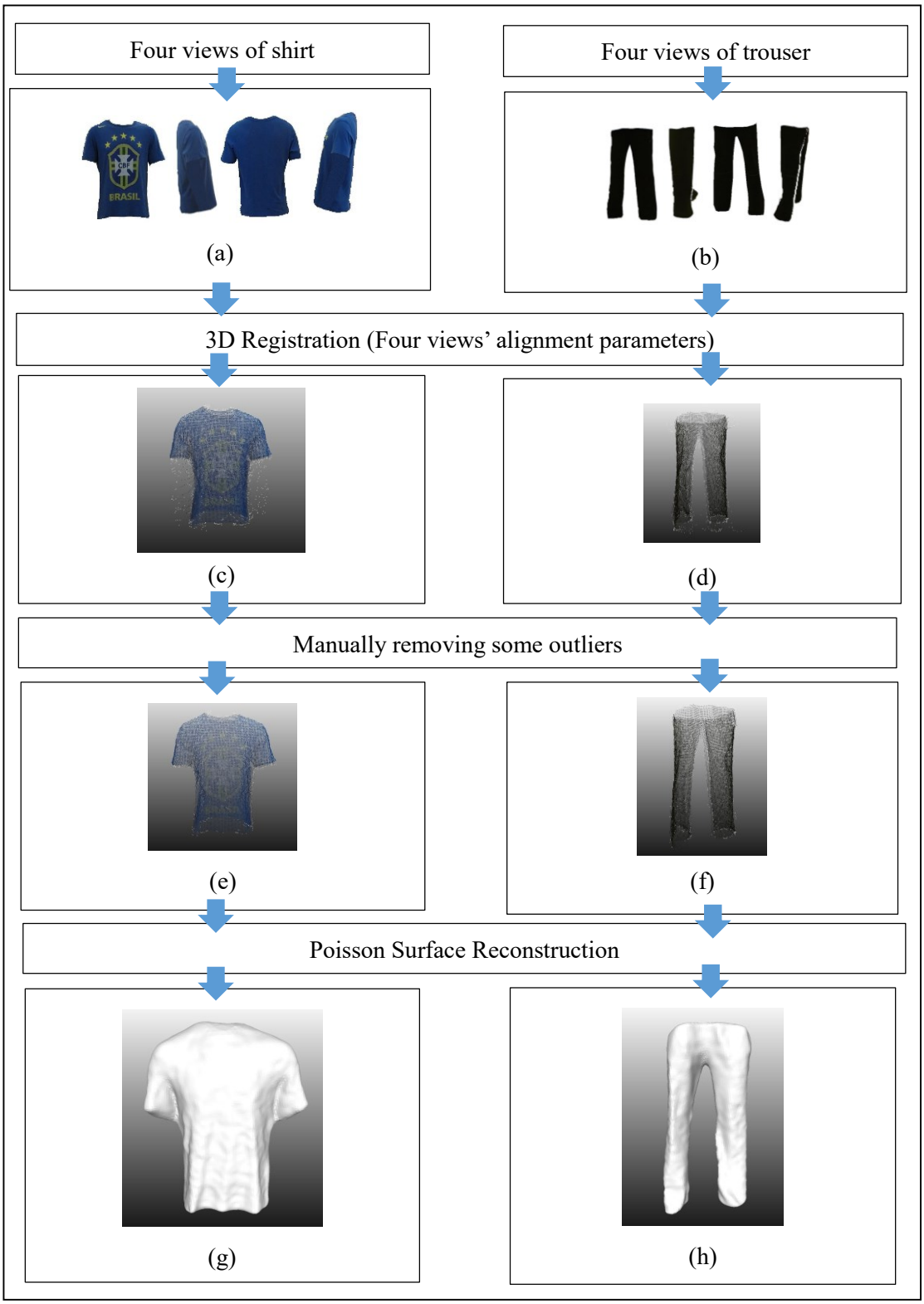


Figure 4.6 From four views point clouds to mesh: Shirt (a) four views of shirt (c) aligned point clouds (e) aligned point clouds after manually removing some outliers (g) after Poisson surface reconstruction. Trouser (b) four views of trouser (d) aligned point clouds (f) aligned point clouds after manually removing some outliers (h) after Poisson surface reconstruction

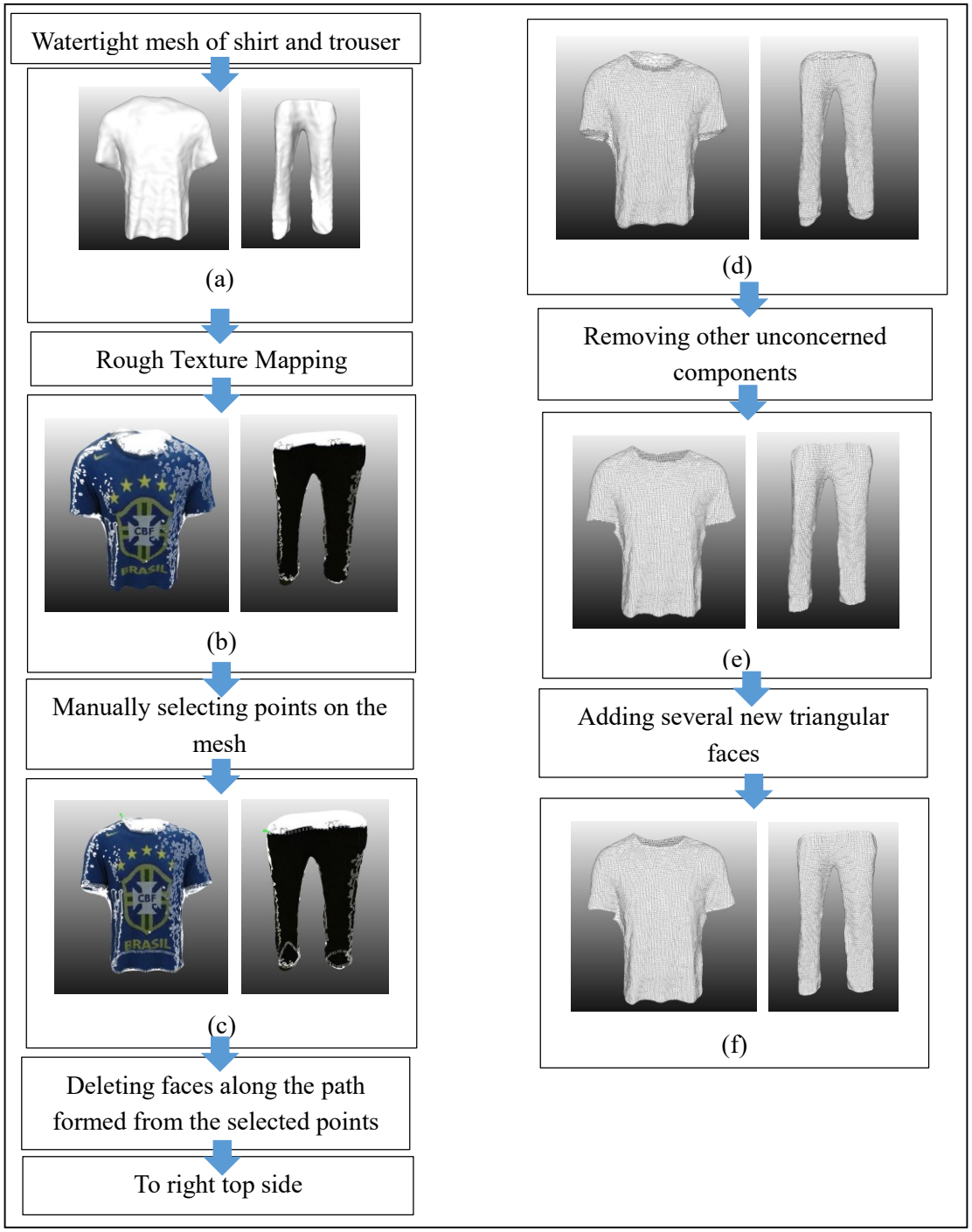


Figure 4.7 Mesh Post-processing: (a) original shirt and trouser mesh (b) rough texture mapping based on Kd-Tree (c) manually selecting several points on shirt and trouser mesh (d) deleting triangular faces along the path formed from selected points (e) removing other unconcerned components (f) adding new faces based on the selected points and boundary edges

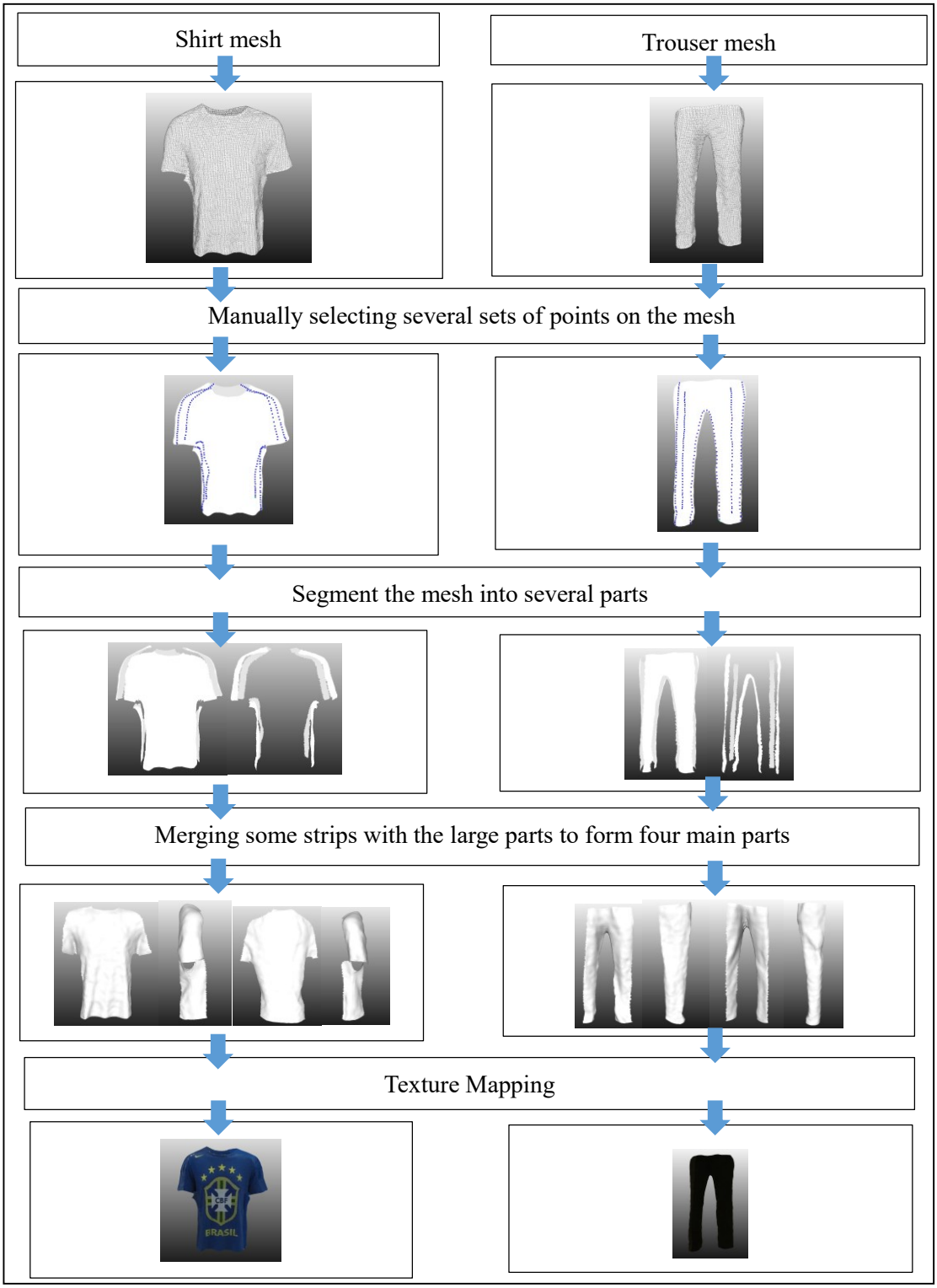


Figure 4.8 Texture mapping for the shirt and the trouser mesh

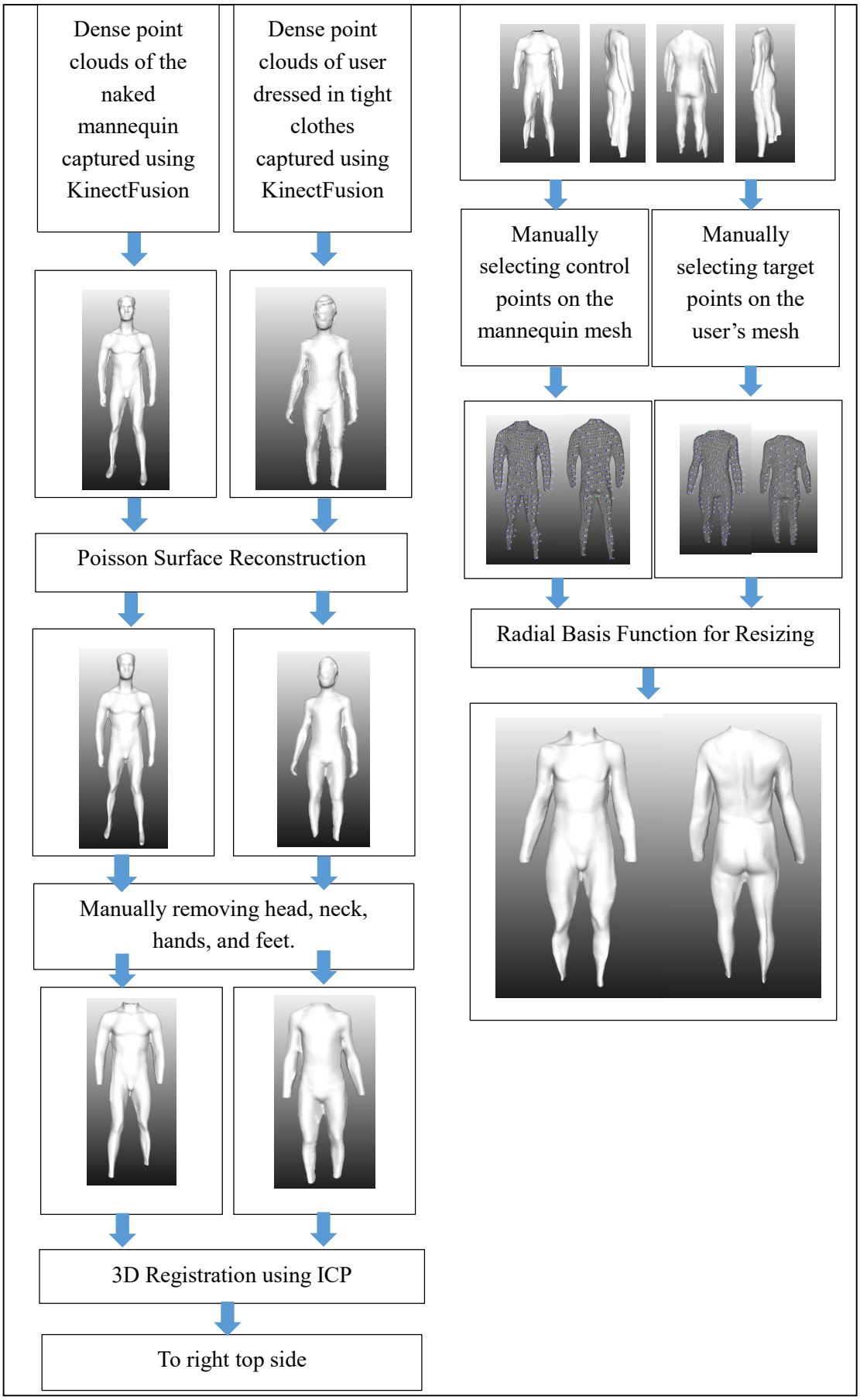


Figure 4.9 Deform the naked mannequin to have the user's body size and shape.

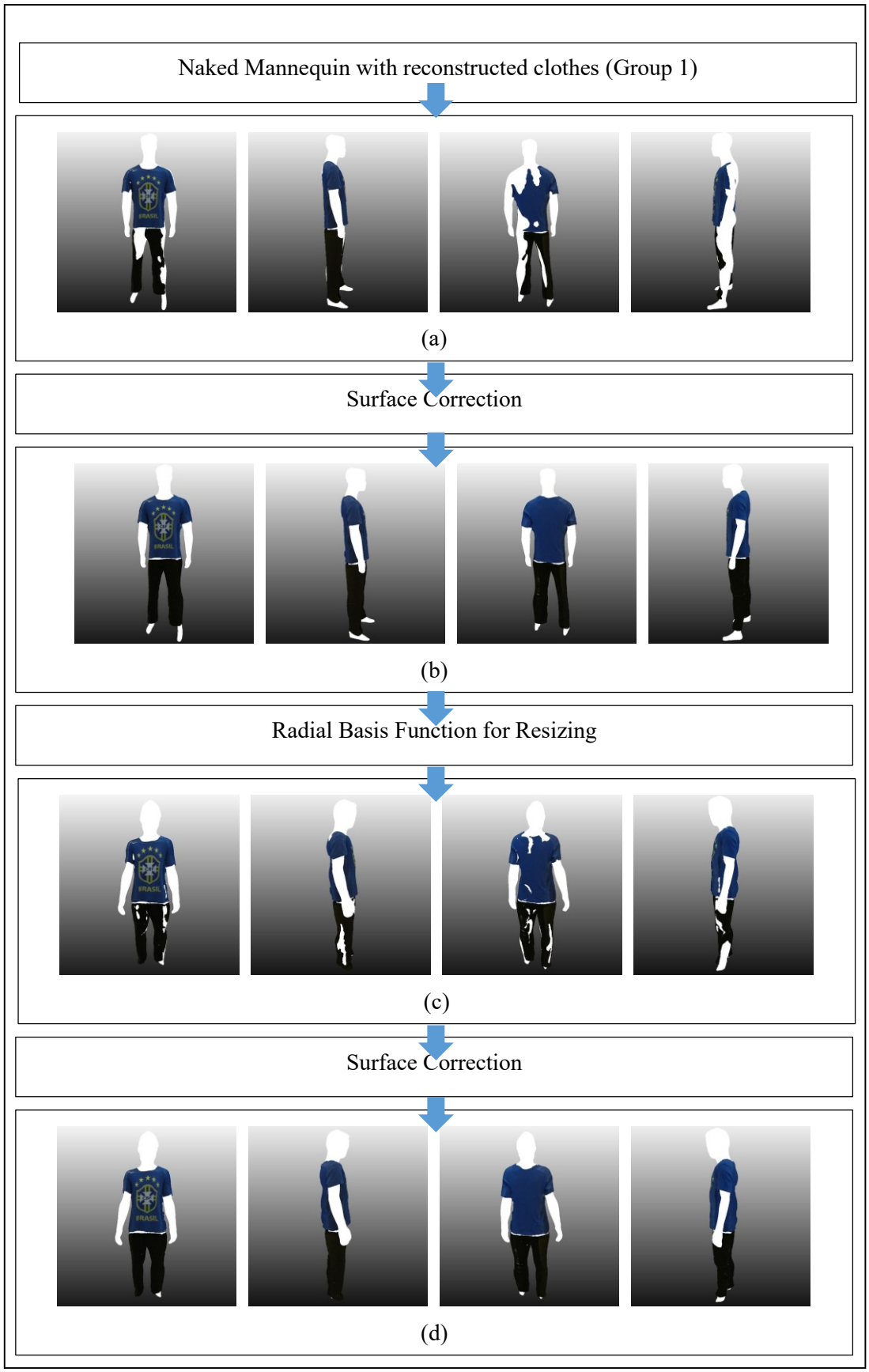


Figure 4.10 Clothes customization (Group 1) (a)-(b) naked mannequin with un-deformed clothes before and after Surface Correction, respectively; (c)-(d) user's 3D model with deformed clothes before and after Surface Correction, respectively.

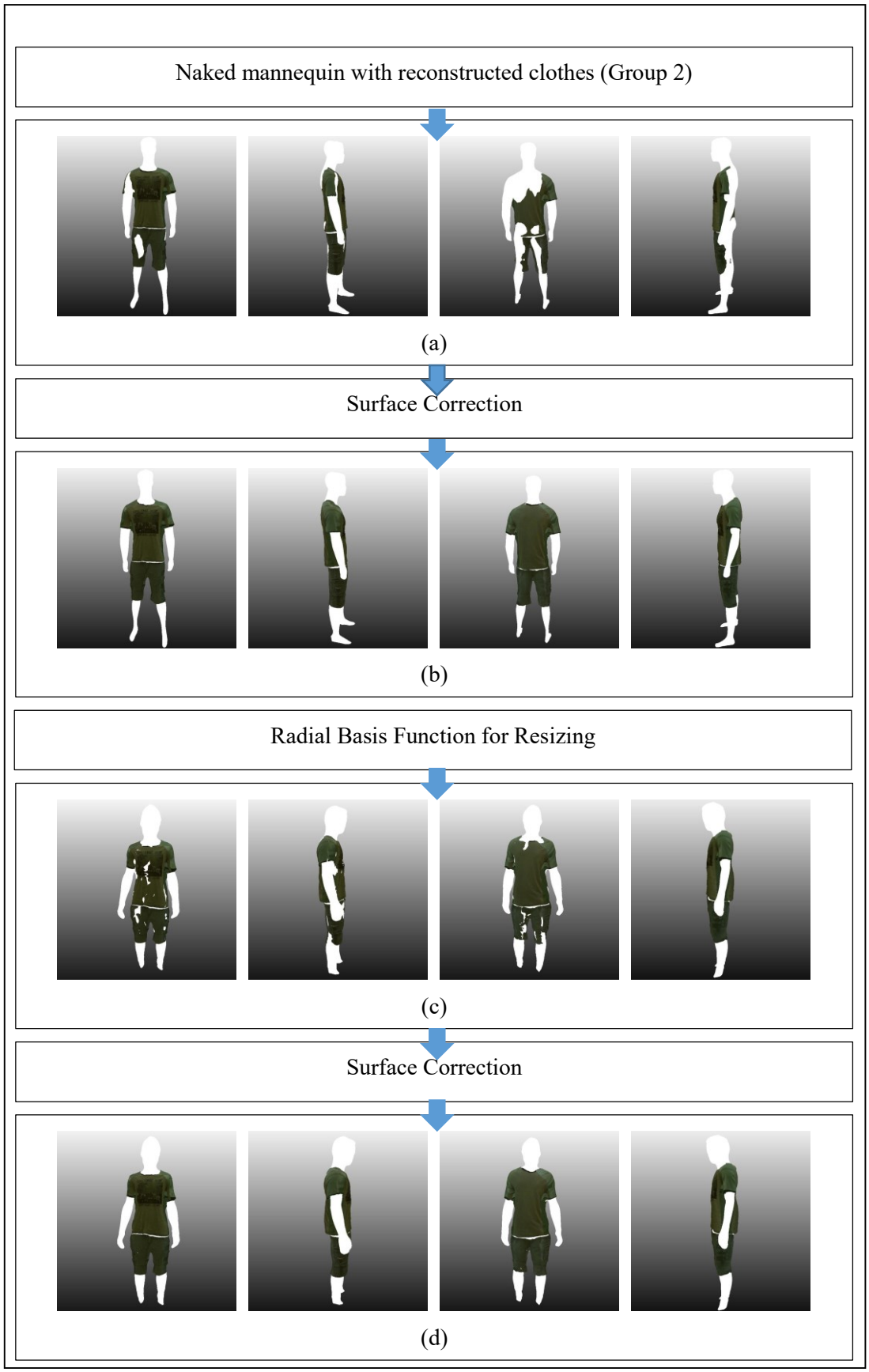


Figure 4.11 Clothes customization (Group 2) (a)-(b) naked mannequin with un-deformed clothes before and after Surface Correction, respectively; (c)-(d) user's 3D model with deformed clothes before and after Surface Correction, respectively.

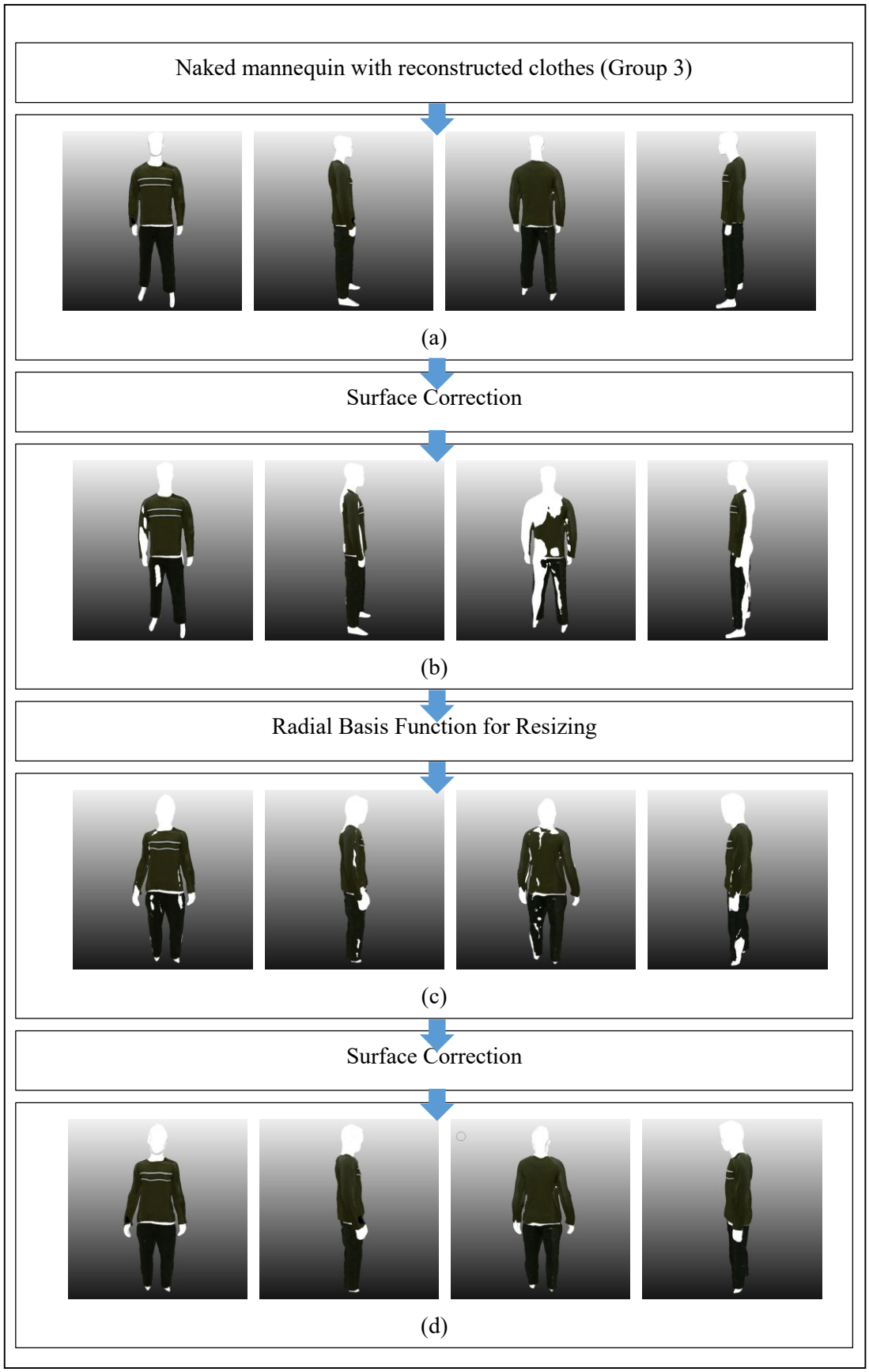


Figure 4.12 Clothes customization (Group 3) (a)-(b) naked mannequin with un-deformed clothes before and after Surface Correction, respectively; (c)-(d) user's 3D model with deformed clothes before and after Surface Correction, respectively.

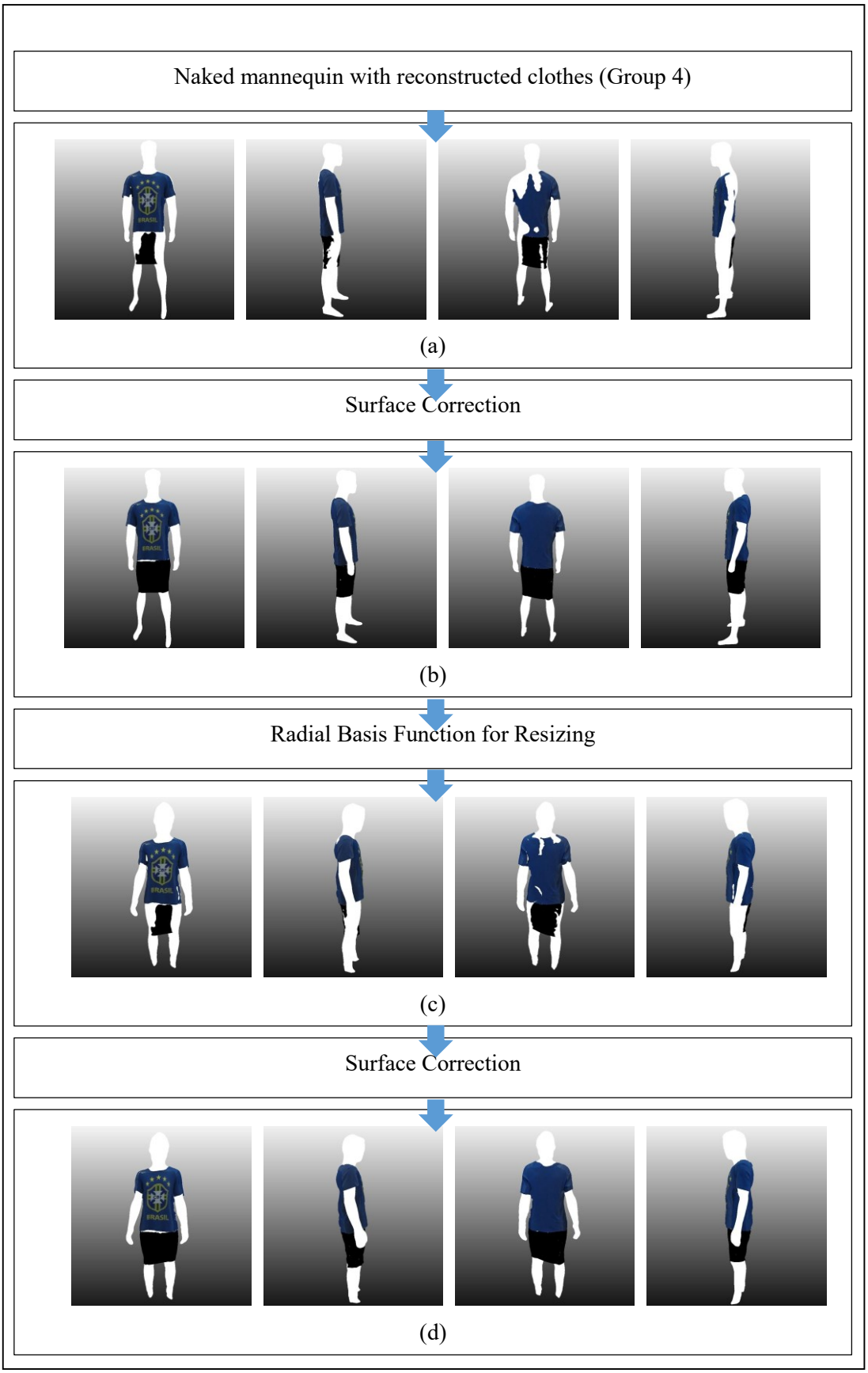


Figure 4.13 Clothes customization (Group 4) (a)-(b) naked mannequin with un-deformed clothes before and after Surface Correction, respectively; (c)-(d) user's 3D model with deformed clothes before and after Surface Correction, respectively.

4.2 System Performance and Validation

Our system consists of four Kinects V2, and one male mannequin. The data capturing and data processing systems are performed in a laptop. The configuration of the computer is displayed in Table 8.

Table 8 Computer Configuration

Computer Configuration	
Processor	Intel(R) Core(TM) i7-3840QM CPU @ 2.80GHz
Memory	16.0GB
Graphics	Intel(R) HD Graphics 4000
Operating System	Windows 8.1 Pro

The data capturing system is developed with Kinect for Windows SDK 2.0, OpenCV 2.49 (Open Source Computer Vision), OpenGL 4.4 (Open Graphics Library), utilizing C++ in Microsoft Visual Studio 2012. The data processing system can be divided into three subsystems. Color segmentation is implemented using JSEG system designed by Yining Deng. Mesh Post-processing is implemented in MATLAB 2013a. Other parts are implemented in Microsoft Visual Studio 2012 using C++.

For analysis of system performance, we evaluate the data units used in the experiment, such as the number of pixels of the 2D dressed mannequin and 2D clothes, the number of vertices and triangular faces in the mesh before and after Mesh Post-processing.

We evaluate the system based on the visual comparison between the captured RGB images and the corresponding reconstructed 3D models.

4.2.1 System Performance

Our clothes extraction is divided into two steps: (1) extracting dressed mannequin

from the background; (2) extracting clothes from the mannequin and separating them into one shirt and one trouser. Since the resolution of Kinect RGB sensor and Depth senor is different, the number of interested pixels in one image is greater than the number of corresponding area of point clouds.

Table 9 shows the number of pixels for dressed mannequin and for original image, since we use Kinect Human Body Tracking to extract dressed mannequin in front and back view, and use Depth Thresholding to extract dressed mannequin in left and right view, the pixels of extracted dressed mannequin in left and right view are greater than those in front and back view.

Table 10 and Table 11 demonstrate the difference before and after region filling for shirt and trouser image. For front and back view, it usually captures all valid data, the number of pixels in those two views are unchanged. But for left and right view, due to the obstruction of two hands, data missing can happen. The number of pixels after region filling is usually greater than the one before region filling.

Table 12 and Table 13 demonstrate the difference of original aligned point clouds, after manually deleting some vertices and after Poisson Surface Reconstruction with respect to shirt and trouser. The original aligned point clouds contain much noise near contour, especially at the data missing area, which can be easily selected and deleted using mouse. And since misaligned and overlapping exist in the aligned point clouds, after applying Poisson Surface Reconstruction, the number of vertices decreases.

Table 14 and Table 15 show the difference before and after mesh post-processing with respect to shirt and trouser. The number of vertices and faces is chosen as the factor

of comparison. Since the reconstructed mesh by using Poisson Surface Reconstruction is watertight, after mesh post-processing, the number of vertices and faces decreases.

Table 9 Comparison of number of interested pixels before and after dressed mannequin extraction.

Comparison between dressed mannequin pixels and original image pixels			
Type	View	Dressed Mannequin	Original Image
Mannequin dressed in Group 1 clothes	Front	157,752	2,073,600
	Back	152,397	2,073,600
	Left	325,796	2,073,600
	Right	304,750	2,073,600
Mannequin dressed in Group 2 clothes	Front	146,489	2,073,600
	Back	141,133	2,073,600
	Left	320,119	2,073,600
	Right	282,845	2,073,600
Mannequin dressed in Group 3 clothes	Front	164,941	2,073,600
	Back	156,052	2,073,600
	Left	306,653	2,073,600
	Right	299,784	2,073,600
Mannequin dressed in Group 4 clothes	Front	120,915	2,073,600
	Back	114,728	2,073,600
	Left	271,068	2,073,600
	Right	258,763	2,073,600

Table 10 Comparison of the number of pixels before and after region filling for shirt.

Comparison of the number of pixels before and after region filling for shirt			
Type	View	Before Region Filling	After Region Filling
Shirt 1	Front	65,529	65,529
	Back	75,535	75,535
	Left	44,521	53,818
	Right	39,268	46,448
Shirt 2	Front	63,367	63,367
	Back	71,689	71,689
	Left	38,463	46,627
	Right	38,390	45,224
Shirt 3	Front	79,659	79,659
	Back	86,967	86,967
	Left	48,921	48,921
	Right	44,203	44,203

Table 11 Comparison of the number of pixels before and after region filling for trouser and skirt.

Comparison of the number of pixels before and after region filling for trouser and skirt			
Type	View	Before Region Filling	After Region Filling
Trouser 1	Front	58,515	58,515
	Back	48,702	48,702
	Left	46,295	48,732
	Right	38,198	40,251
Trouser 2	Front	39,893	39,893
	Back	35,195	35,195
	Left	26,347	28,663
	Right	22,102	24,509
Trouser 3	Front	62,298	62,298
	Back	58,711	58,711
	Left	50,474	52,983
	Right	45,694	48,852
Skirt	Front	26,609	26,609
	Back	22,060	22,060
	Left	19,620	22,771
	Right	15,906	18,186

Table 12 Comparison of the number of vertices of aligned shirt point clouds, after manually deleting some outliers and after Poisson Surface Reconstruction.

Comparison of the number of vertices of raw aligned shirt point clouds, aligned shirt point clouds after manually deleting some outliers and after Poisson Surface Reconstruction			
Type	Aligned Point Clouds	After Manually Deleting Some Outliers	After Poisson Surface Reconstruction
Shirt 1	23,871	23,555	21,163
Shirt 2	23,297	22,699	21,580
Shirt 3	28,852	28,288	23,941

Table 13 Comparison of the number of vertices of aligned trouser and skirt point clouds after manually deleting some outliers and after Poisson Surface Reconstruction.

Comparison of the number of vertices of raw aligned trouser and skirt point clouds, after manually deleting some outliers and after Poisson Surface Reconstruction			
Type	Aligned Point Clouds	After Manually Deleting Some Outliers	After Poisson Surface Reconstruction
Trouser 1	21,769	20,787	15,473
Trouser 2	13,047	12,965	13,181
Trouser 3	23,774	23,232	16,325
Skirt	9,360	9,307	7,117

Table 14 Comparison of the number of vertices and faces of shirt mesh before and after Mesh Post-processing.

Comparison of the number of vertices and faces of shirt mesh before and after Mesh Post-processing		
Type	Before Mesh Post-processing (vertices/faces)	After Mesh Post-processing (vertices/faces)
Shirt 1	21,163 / 42,322	19,089 / 37,935
Shirt 2	21,580 / 43,156	19,595 / 39,006
Shirt 3	23,941 / 47,878	22,321 / 44,485

Table 15 Comparison of the number of vertices and faces of trouser and skirt mesh before and after Mesh Post-processing.

Comparison of the number of vertices and faces of trouser and skirt mesh before and after Mesh Post-processing		
Type	Before Mesh Post-processing (vertices/faces)	After Mesh Post-processing (vertices/faces)
Trouser 1	15,473 / 30,942	14,165 / 28,170
Trouser 2	13,181 / 26,358	11,895 / 23,669
Trouser 3	16,325 / 32,646	14,807 / 29,510
Skirt	7,117 / 14,230	6,026 / 11,802

4.2.2 Visual Validation of the Results

Four groups of clothes are captured and reconstructed. The evaluation is based on the visual comparison between the captured clothes images and the reconstructed

clothes.

As shown in Figure 4.10, Figure 4.11, Figure 4.12 and Figure 4.13, the clothes shape and texture in the captured images are passed to the 3D clothes model. The patterns on shirt are much similar to the raw patterns, and the virtual mannequin with the reconstructed clothes also shares a variety of similarities with the actual one. Clothes with different styles (such as pants, skirt, trouser, and short/long sleeves shirt) and various color and texture (such as blue, green, and black) are all successfully reconstructed using the proposed approach.

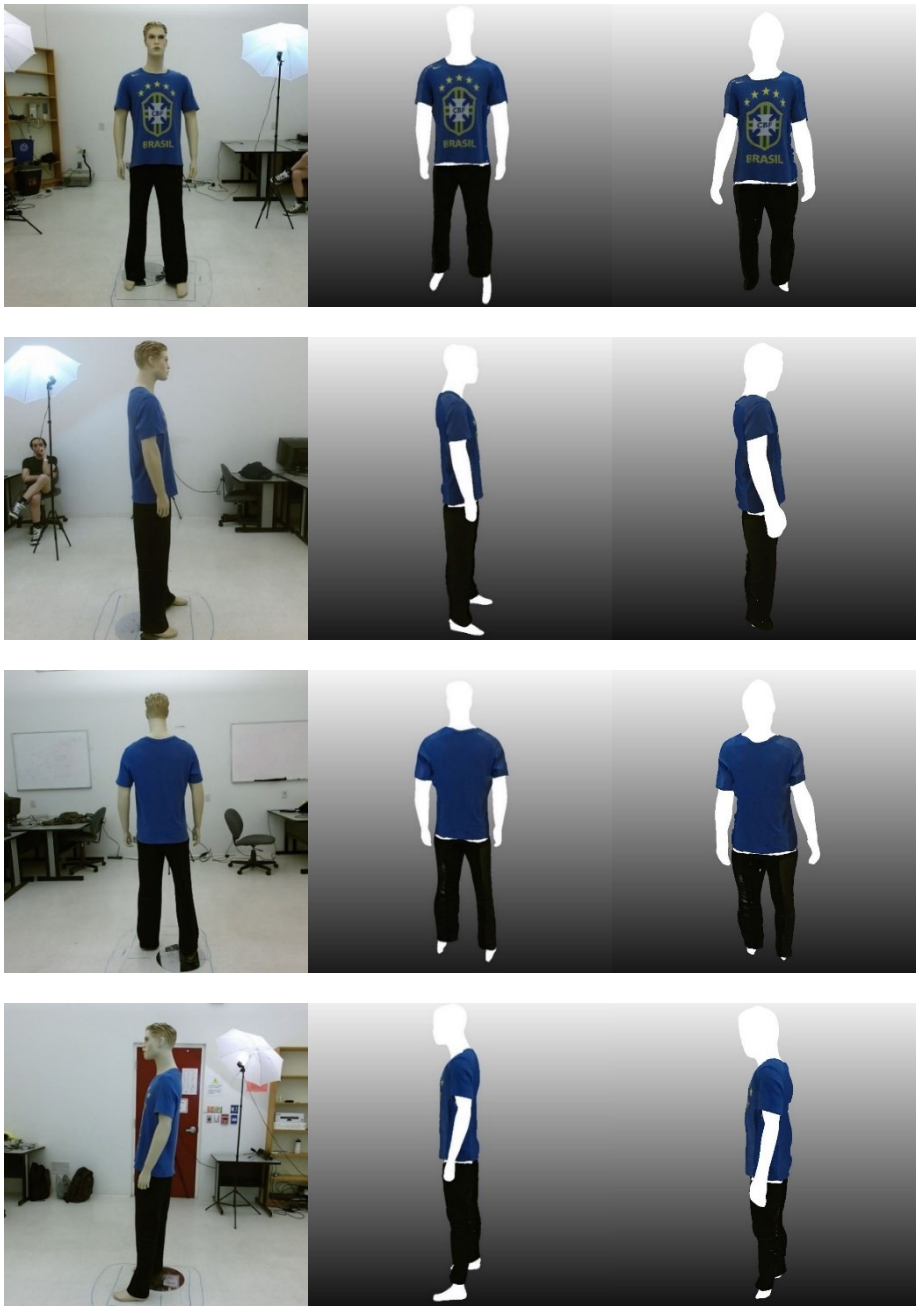


Figure 4.14 The comparisons of the captured clothes (Group 1) images and the reconstructed 3D clothes before and after resizing (Top to down: front view; left view; back view; right view).

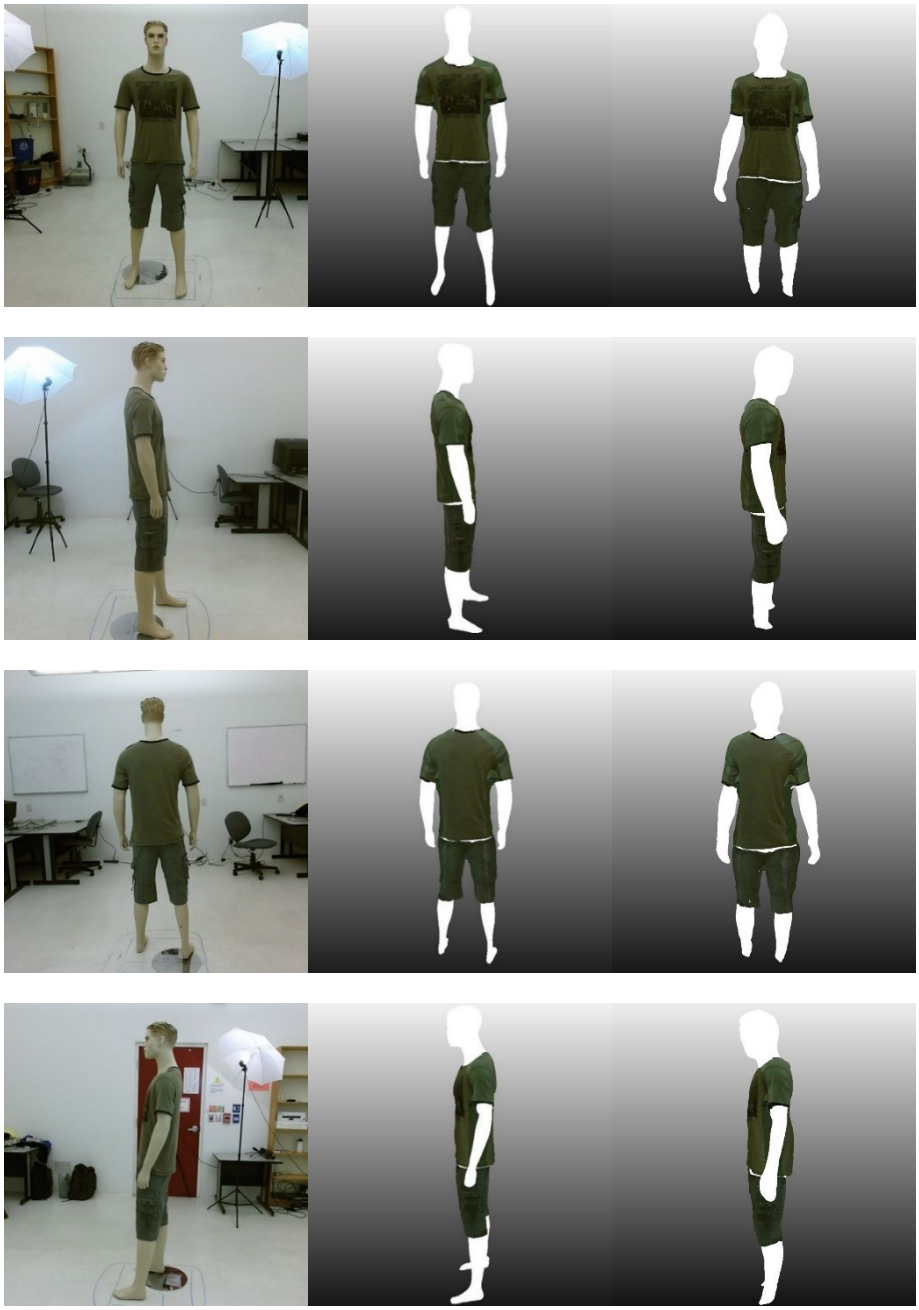


Figure 4.15 The comparisons of the captured clothes (Group 2) images and the reconstructed 3D clothes before and after resizing (Top to down: front view; left view; back view; right view).

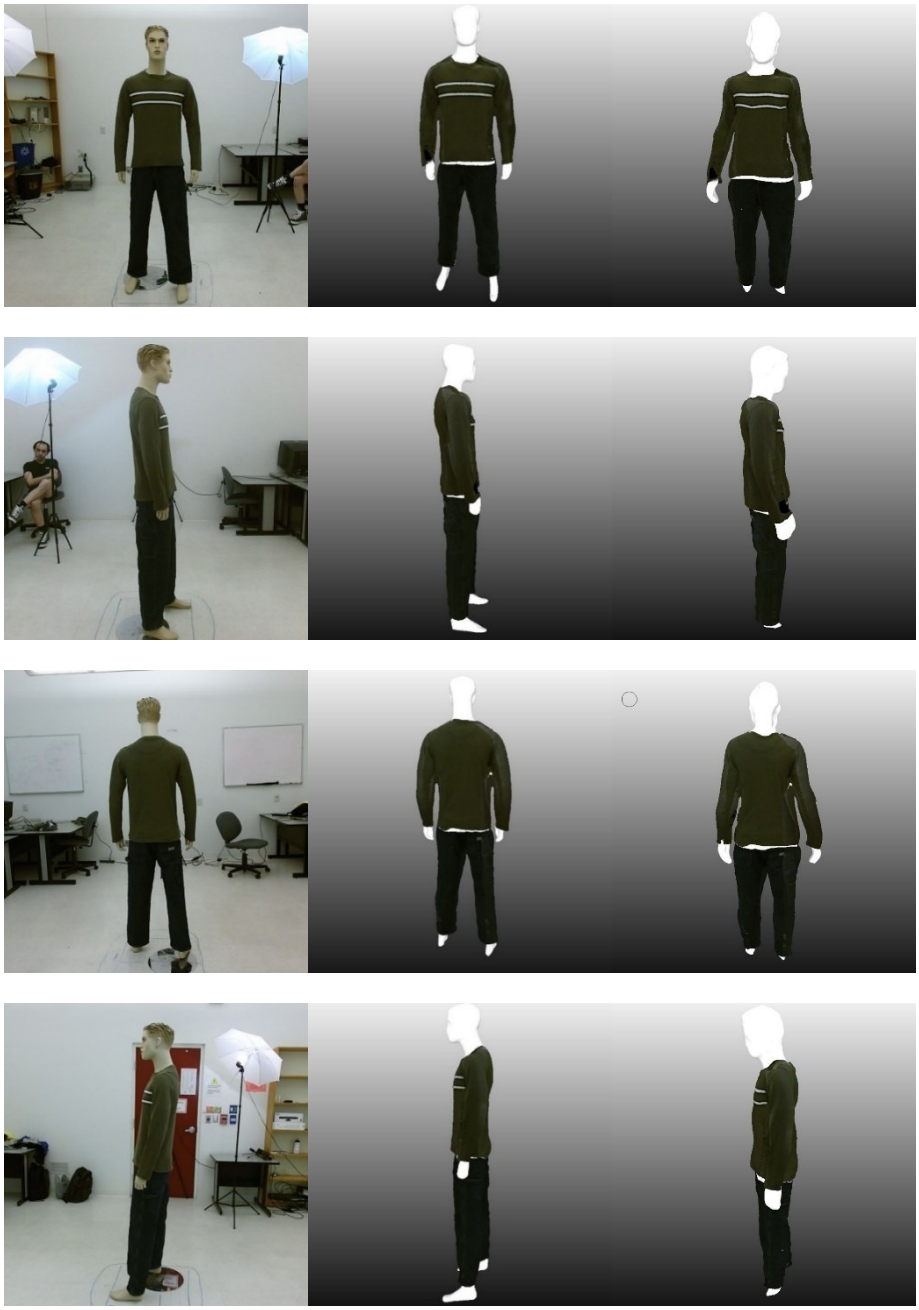


Figure 4.16 The comparisons of the captured clothes (Group 3) images and the reconstructed 3D clothes before and after resizing (Top to down: front view; left view; back view; right view).

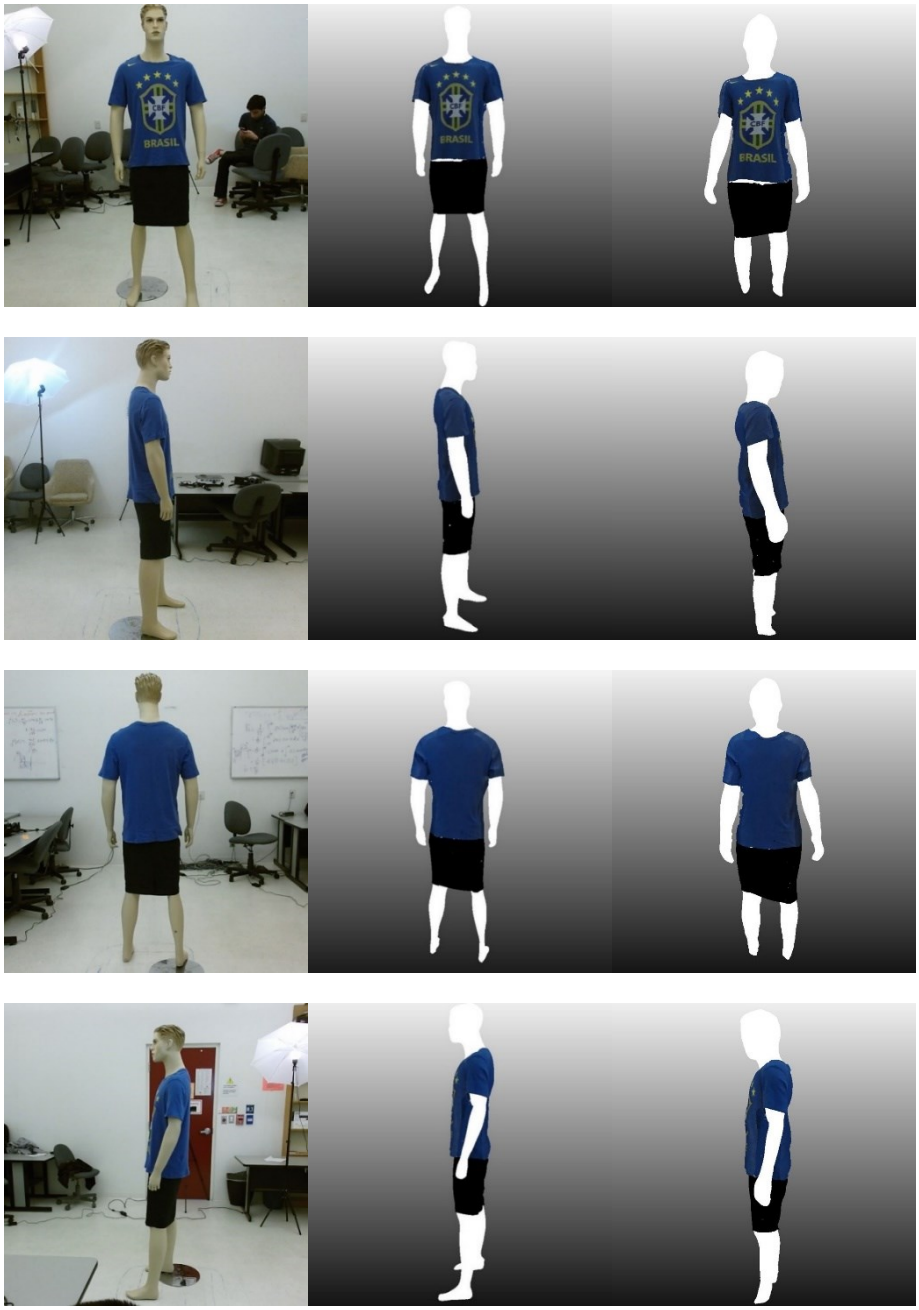


Figure 4.17 The comparisons of the captured clothes (Group 4) images and the reconstructed 3D clothes before and after resizing (Top to down: front view; left view; back view; right view).

4.3 Discussion

The result of the experiment proves that our approach for 3D clothes modeling and customization is effective and has the ability to deal with multiple colors and different clothes types. It takes about two hours for our users to model 3D clothes and fit to a

human user including the capturing time, while the typical 3D clothes modeling that starts with several predefined 2D patterns usually takes two days to reconstruct one clothes [7]. The reconstructed 3D clothes share many similarities as the real clothes, besides, our approach for clothes customization can provide a good result. But meanwhile, since the RBF we used for deformation is by manually selecting several control points and corresponding target points, which is not convenient, it can be improved. There is a space to extract feature points semi-automatically or atomically from the mesh for faster process.

Our approach for aligning multiple views of point clouds can provide a good alignment result, however at the same time, it is not so convenient since it requires using KinectFusion to scan the naked mannequin first, which can be redesigned and improved in the future.

Chapter 5. Conclusion

In this chapter, we summarize our system, our main contributions, limitations of our current system, and future work.

5.1 Summary

In this thesis, a novel approach for 3D clothes modeling and customization is proposed. The hardware setting consisted of four inexpensive Kinects is convenient. The Kinects are used to capture RGB and depth data of a mannequin dressed in one shirt and one trouser. Camera calibration and stereo calibration are implemented for generating point clouds and registering point clouds with color information. A color segmentation-based method called JSEG is used to extract shirt and trouser out of the dressed mannequin in any background. Region filling technique is used to solve the problem of data missing in four views' RGB images. And 3D registration technique ICP is used to align multiple point clouds, while Poisson Surface Reconstruction is performed later to reconstruct the clothes mesh from the clothes point clouds. Mesh post-processing is applied to the watertight clothes mesh to create a much more realistic clothes mesh. Texture mapping technique based on 3D segmentation is utilized to generate the textured clothes. Then RBF-based deformation technique is used for clothes size customization. Finally, an algorithm based on point-plane projection is used to solve the problem of some vertices penetrating into the human body.

5.2 Contribution

Existing virtual try-on systems target size problems while virtual try-on including online clothe shopping has both size and color mismatch problems. The main contributions of our system lie in four parts, which are:

1. A simple hardware setting consisted of four Kinects to capture a human model.
2. 3D clothes modeling out of captured human figure. Many existing systems capture human models in clothes, but our work is the first one, best of our knowledge, which separates clothes out of captured human model.
3. Resizing of clothes adapting to any sized human.
4. It is a first step for novel idea of virtual try-on where clothes and human are captured in the same location.

5.3 Future Work

Although our approach for 3D clothes modeling and customization can output the smooth and well-textured clothes and fit them into any size human model, several limitations still exist in the current system.

One obvious limitation is that Kinect raw data has missing points in either RGB image or depth data. We used inexpensive Kinects in order to make the system be practical. More Kinects can be adopted to compensate the problem of data missing and improve the performance in the future.

Another limitation lies in the method how 3D registration is performed. One 3D

model generated from KinectFusion is used as the medium to align multiple views. The accuracy of this 3D model affects the alignment result. New idea of 3D registration can be developed to improve the alignment effect.

In 3D clothes customization, the RBF for deformation based on manually selecting several control points and target points is troublesome. Potential algorithm can be developed to semi-automatically or automatically select points on the mesh.

In the experiment, we only test several shirts, trousers and a skirt with some simple color patterns, while, in reality, there are a lot of clothes with far more complex patterns. So in the future work, more clothes types with various patterns should be tested.

Our current clothes customization approach allows clothes to be resized into infinite size, while, in the market, only several size is available, such as XS, S, M, L, and XL. We consider clothes size classification into limited number of clothes size to make the system more matching to the reality.

References

[1]. Divivier, A., R. Trieb, Aea Ebert, H. Hagen, C. Gross, A. Fuhrmann, and V. Luckas. "Virtual try-on topics in realistic, individualized dressing in virtual reality." (2004).

[2]. Kim, Jiyeon, and Sandra Forsythe. "Adoption of virtual try-on technology for online apparel shopping." *Journal of Interactive Marketing* 22, no. 2 (2008): 45-59.

[3]. Cordier, Frédéric, W. Lee, H. Seo, and Nadia Magnenat-Thalmann. "Virtual-try-on on the web." *Laval Virtual* (2001).

[4]. Araki, Natsuha, and Yoichi Muraoka. "Follow-the-Trial-Fitter: Real-time dressing without undressing." In *Digital Information Management, 2008. ICDIM 2008. Third International Conference on*, pp. 33-38. IEEE, 2008.

[5]. Hauswiesner, Stefan, Matthias Straka, and Gerhard Reitmayr. "Virtual try-on through image-based rendering." *Visualization and Computer Graphics, IEEE Transactions on* 19, no. 9 (2013): 1552-1565.

[6]. Isıkdogan, Furkan, and Gökçehan Kara. "A real time virtual dressing room application using Kinect." (2013).

[7]. Giovanni, Stevie, Yeun Chul Choi, Jay Huang, Eng Tat Khoo, and KangKang Yin. "Virtual try-on using Kinect and HD camera." In *Motion in Games*, pp. 55-65. Springer Berlin Heidelberg, 2012.

[8]. EON Interactive Mirror, available online: http://www.eonreality.com/eon-interactive-mirror/, August 2015.

[9]. Fitnect, available online: http://www.fitnect.hu/, August 2015.

[10]. Kinect for Windows Retail Clothes Scenario Video, available online: https://www.youtube.com/watch?v=Mr71jrkzWq8, August 2015.

[11]. Yuan, Miaolong, Ishtiaq Rasool Khan, Farzan Farbiz, Susu Yao, Arthur Niswar, and Min-Hui Foo. "A mixed reality virtual clothes try-on system." *Multimedia, IEEE Transactions on* 15, no. 8 (2013): 1958-1968.

[12]. Zhang, Yuzhe, Jianmin Zheng, and Nadia Magnenat-Thalmann. "Cloth Simulation and Virtual Try-on with Kinect Based on Human Body Adaptation." In *Simulations, Serious Games and Their Applications*, pp. 31-50. Springer Singapore, 2014.

[13]. Furukawa, Takao, Jin Gu, WonSook Lee, and Nadia Magnenat-Thalmann. "3D clothes modeling from photo cloned human body." In *Virtual Worlds*, pp. 159-170. Springer Berlin Heidelberg, 2000.

[14]. Blender for 3D clothes modeling, available online: https://www.youtube.com/watch?v=kNy4d33WS3M, August 2015.

[15]. 3ds Max for 3D clothes modeling, available online: https://www.youtube.com/watch?v=UOY-nRhvWTc, August 2015.

[16]. Maya for 3D clothes modeling, available online: https://www.youtube.com/watch?v=U41bp50EEhY, August 2015.

[17]. Marvelous Designer, available online: http://www.marvelousdesigner.com/, August 2015.

[18]. Tong, Jing, Jin Zhou, Ligang Liu, Zhigeng Pan, and Hao Yan. "Scanning 3d

full human bodies using Kinects." *Visualization and Computer Graphics, IEEE Transactions on* 18, no. 4 (2012): 643-650.

[19]. Li, Zhongrui, Chao Sun, Won-Sook Lee, and Ig-Jae Kim. "Hair modeling using Kinect sensors and DSLR cameras." In *Ubiquitous Robots and Ambient Intelligence (URAI), 2014 11th International Conference on*, pp. 22-27. IEEE, 2014.

[20]. Li, Jituo, Juntao Ye, Yangsheng Wang, Li Bai, and Guodong Lu. "Fitting 3D garment models onto individual human models." *Computers & graphics* 34, no. 6 (2010): 742-755.

[21]. Structured light, available online: https://en.wikipedia.org/wiki/Structured_light, August 2015.

[22]. Fofi, David, Tadeusz Sliwa, and Yvon Voisin. "A comparative survey on invisible structured light." In *Electronic Imaging 2004*, pp. 90-98. International Society for Optics and Photonics, 2004.

[23]. Salvi, Joaquim, Sergio Fernandez, Tomislav Pribanic, and Xavier Llado. "A state of the art in structured light patterns for surface profilometry." *Pattern recognition* 43, no. 8 (2010): 2666-2680.

[24]. Valkenburg, Robert J., and Alan M. McIvor. "Accurate 3D measurement using a structured light system." *Image and Vision Computing* 16, no. 2 (1998): 99-110.

[25]. Scharstein, Daniel, and Richard Szeliski. "High-accuracy stereo depth maps using structured light." In *Computer Vision and Pattern Recognition, 2003.*

Proceedings. 2003 IEEE Computer Society Conference on, vol. 1, pp. I-195. IEEE, 2003.

[26]. Batlle, Joan, E. Mouaddib, and Joaquim Salvi. "Recent progress in coded structured light as a technique to solve the correspondence problem: a survey."*Pattern recognition* 31, no. 7 (1998): 963-982.

[27]. Computer stereo vision, available online: https://en.wikipedia.org/wiki/Computer_stereo_vision, August 2015.

[28]. Brown, Myron Z., Darius Burschka, and Gregory D. Hager. "Advances in computational stereo." *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 25, no. 8 (2003): 993-1008.

[29]. Chen, Chichyang, and Yuan F. Zheng. "Passive and active stereo vision for smooth surface detection of deformed plates." *Industrial Electronics, IEEE Transactions on* 42, no. 3 (1995): 300-306.

[30]. ZED, available online: https://www.stereolabs.com/, August 2015.

[31]. Multi-view Stereo & Structure from Motion, available online: http://cs.nyu.edu/~fergus/teaching/vision/11_12_multiview.pdf, August 2015.

[32]. Steven M. Seitz, Brian Curless, James Diebel, Daniel Scharstein, Richard Szeliski, "A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms", *Computer Vision and Pattern Recognition*, *2006 IEEE Computer Society Conference*, Volume 1, pages 519-528, June 2006.

[33]. Gu, Jin, Terry Chang, Ivan Mak, S. Gopalsamy, Helen C. Shen, and Matthew

Ming-Fai Yuen. "A 3D reconstruction system for human body modeling." In *Modelling and Motion Capture Techniques for Virtual Environments*, pp. 229-241. Springer Berlin Heidelberg, 1998.

[34]. Bradley, Derek, Tiberiu Popa, Alla Sheffer, Wolfgang Heidrich, and Tamy Boubekeur. "Markerless garment capture." In *ACM Transactions on Graphics (TOG)*, vol. 27, no. 3, p. 99. ACM, 2008.

[35]. Infinite Reality, available online: http://ir-ltd.net/, August 2015.

[36]. 3D scanner, available online: http://en.wikipedia.org/wiki/3D_scanner#Technology, August 2015.

[37]. Laser scanning, available online: https://en.wikipedia.org/wiki/Laser_scanning, August 2015.

[38]. Boehler, Wolfgang, and Andreas Marbs. "3D scanning instruments." In *Proceedings of the CIPA WG 6 International Workshop on Scanning for Cultural Heritage Recording, Ziti, Thessaloniki*, pp. 9-18. 2002.

[39]. Time-of-Flight camera, available online: https://en.wikipedia.org/wiki/Time-of-flight_camera, August 2015.

[40]. Li, Larry. "Time-of-Flight Camera–An Introduction." *Technical White Paper, May* (2014).

[41]. Kinect, available online: http://en.wikipedia.org/wiki/Kinect, August 2015.

[42]. Jan Smisek, Michal Jancosek, Tomas Pajdla, "3D with Kinect", *Consumer Depth Cameras for Computer Vision Advances in Computer Vision and Pattern Recognition 2013*, pages 3-25, 2013.

[43]. Kinect V1, available online: https://msdn.microsoft.com/en-us/library/jj131033.aspx, August 2015.

[44]. Rizwan Macknojia, Alberto Chavez-Aragon, Pierre Payeur, Robert Laganiere, "Experimental characterization of two generations of Kinect's depth sensors", *Robotic and Sensors Environments (ROSE) 2012 IEEE International Symposium*, pages 150-155, Nov 2012.

[45]. Technical Description of Kinect, available online: http://wiki.ros.org/kinect_calibration/technical, August 2015.

[46]. Kourosh Khoshelham, Sander Oude Elberink, "Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications", *Sensors 2012*, 12(2), pages 1437-1454, 2012.

[47]. Kinect for Windows V2 Features, available online: https://www.microsoft.com/enus/kinectforwindows/meetkinect/features.aspx, August 2015.

[48]. Kinect for Windows V2: Sensor and Data Sources Overview, available online: https://www.youtube.com/watch?v=qdchWQjddlw, August 2015.

[49]. Kinect for Windows SDK, available online: https://www.microsoft.com/en-us/download/details.aspx?id=44561, August 2015.

[50]. Kinect for Windows V2 sensor price, available online: https://www.microsoft.com/en-us/kinectforwindows/purchase/v2sensor.aspx, August 2015.

[51]. Cyberware, available online: http://cyberware.com/, August 2015.

[52]. Cyberware Domestic Product Price Information Sheet, available online: http://cyberware.com/pricing/domesticPriceList.html, August 2015.

[53]. Newcombe, Richard A., Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. "KinectFusion: Real-time dense surface mapping and tracking." In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, pp. 127-136. IEEE, 2011.

[54]. Rusu, Radu Bogdan, and Steve Cousins. "3D is here: Point cloud library (PCL)." In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 1-4. IEEE, 2011.

[55]. Image Segmentation, available online: http://en.wikipedia.org/wiki/Image_segmentation, August 2015.

[56]. Luccheseyz, L., and S. K. Mitray. "Color image segmentation: A state-of-the-art survey." *Proceedings of the Indian National Science Academy (INSA-A)* 67, no. 2 (2001): 207-221.

[57]. Sharma, Nikita, Mahendra Mishra, and Manish Shrivastava. "Colour image segmentation techniques and issues: an approach." *International Journal of Scientific & Technology Research* 1, no. 4 (2012): 9-12.

[58]. Vantaram, Sreenath Rao, and Eli Saber. "Survey of contemporary trends in color image segmentation." *Journal of Electronic Imaging* 21, no. 4 (2012): 040901-1.

[59]. Saini, Sujata, and Komal Arora. "A Study Analysis on the Different Image Segmentation Techniques."

[60]. Malik, Jitendra, Serge Belongie, Thomas Leung, and Jianbo Shi. "Contour and texture analysis for image segmentation." *International journal of computer vision* 43, no. 1 (2001): 7-27.

[61]. Belongie, Serge, Chad Carson, Hayit Greenspan, and Jitendra Malik. "Color- and texture-based image segmentation using EM and its application to content-based image retrieval." In *Computer Vision, 1998. Sixth International Conference on*, pp. 675-682. IEEE, 1998.

[62]. Deng, Yining, and B. S. Manjunath. "Unsupervised segmentation of color-texture regions in images and video." *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 23, no. 8 (2001): 800-810.

[63]. JSEG system, available online: http://old.vision.ece.ucsb.edu/segmentation/jseg/software/, August 2015.

[64]. Yang Chen, Gerard Medioni, “Object Modeling by Registration of Multiple Range Images”, *Image and Vision Computing*, pp. 145-155, 1992.

[65]. Bergevin, Robert, Marc Soucy, Hewe Gagnon, and Denis Laurendeau. "Towards a general multi-view registration technique." *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 18, no. 5 (1996): 540-547.

[66]. Low, Kok-Lim. "Linear least-squares optimization for point-to-plane ICP surface registration." *Chapel Hill, University of North Carolina* (2004).

[67]. Pomerleau, François, Francis Colas, Roland Siegwart, and Stéphane Magnenat.

"Comparing ICP variants on real-world data sets." *Autonomous Robots* 34, no. 3 (2013): 133-148.

[68]. Besl, Paul J., and Neil D. McKay. "Method for registration of 3-D shapes." In *Robotics-DL tentative*, pp. 586-606. International Society for Optics and Photonics, 1992.

[69]. Zhang, Zhengyou. "Iterative point matching for registration of free-form curves and surfaces." *International journal of computer vision* 13, no. 2 (1994): 119-152.

[70]. Lorensen, William E., and Harvey E. Cline. "Marching cubes: A high resolution 3D surface construction algorithm." In *ACM siggraph computer graphics*, vol. 21, no. 4, pp. 163-169. ACM, 1987.

[71]. Weber, Gunther H., Gerik Scheuermann, Hans Hagen, and Bernd Hamann. "Exploring scalar fields using critical isovalues." In *Visualization, 2002. VIS 2002. IEEE*, pp. 171-178. IEEE, 2002.

[72]. Marching Cubes, available online: http://dvizreview.blogspot.ca/2012/02/marching-cubes.html, August 2015.

[73]. Chen, Li. *Digital and Discrete Geometry: Theory and Algorithms*. Springer, 2015.

[74]. Newman, Timothy S., and Hong Yi. "A survey of the marching cubes algorithm." *Computers & Graphics* 30, no. 5 (2006): 854-879.

[75]. Berger, Matthew, Andrea Tagliasacchi, Lee Seversky, Pierre Alliez, Joshua Levine, Andrei Sharf, and Claudio Silva. "State of the art in surface

reconstruction from point clouds." In *EUROGRAPHICS star reports*, vol. 1, no. 1, pp. 161-185. 2014.

[76]. Carr, Jonathan C., Richard K. Beatson, Jon B. Cherrie, Tim J. Mitchell, W. Richard Fright, Bruce C. McCallum, and Tim R. Evans. "Reconstruction and representation of 3D objects with radial basis functions." In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pp. 67-76. ACM, 2001.

[77]. Kazhdan, Michael, Matthew Bolitho, and Hugues Hoppe. "Poisson surface reconstruction." In *Proceedings of the fourth Eurographics symposium on Geometry processing*, vol. 7. 2006.

[78]. Zhou, Kun, Xi Wang, Yiying Tong, Mathieu Desbrun, Baining Guo, and Heung-Yeung Shum. "TextureMontage." In *ACM Transactions on Graphics (TOG)*, vol. 24, no. 3, pp. 1148-1155. ACM, 2005.

[79]. Igarashi, Takeo, Tomer Moscovich, and John F. Hughes. "As-rigid-as-possible shape manipulation." *ACM transactions on Graphics (TOG)* 24, no. 3 (2005): 1134-1141.

[80]. Li, Jituo, and Guodong Lu. "Customizing 3D garments based on volumetric deformation." *Computers in Industry* 62, no. 7 (2011): 693-707.

[81]. Sederberg, Thomas W., and Scott R. Parry. "Free-form deformation of solid geometric models." In *ACM SIGGRAPH computer graphics*, vol. 20, no. 4, pp. 151-160. ACM, 1986.

[82]. Coquillart, Sabine, “Extended free-form deformation: a sculpturing tool for

3D geometric modeling", vol. 24, no. 4. ACM, 1990.

[83]. Farin, Gerald. "Surfaces over Dirichlet tessellations." *Computer aided geometric design* 7, no. 1 (1990): 281-292.

[84]. Noh, Jun-yong, Douglas Fidaleo, and Ulrich Neumann. "Animated deformations with radial basis functions." In *Proceedings of the ACM symposium on Virtual reality software and technology*, pp. 166-174. ACM, 2000.

[85]. Levi, Zohar, and Dan Levin. "Shape deformation via interior RBF." in *Visualization and Computer Graphics, IEEE Transactions on* 20, no. 7 (2014): 1062-1075.

[86]. Zhang, Zhengyou. "A flexible new technique for camera calibration." *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 22, no. 11 (2000): 1330-1334.

[87]. Macknojia, Rizwan, Alberto Chávez-Aragón, Pierre Payeur, and Robert Laganiere. "Calibration of a network of Kinect sensors for robotic inspection over a large workspace." In *Robot Vision (WORV), 2013 IEEE Workshop on*, pp. 184-190. IEEE, 2013.

[88]. Camera Calibration and 3D Reconstruction, available online: http://docs.opencv.org/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html, August 2015.

[89]. OpenCV, available online: http://opencv.org/, August 2015.

[90]. MeshLab, available online: http://meshlab.sourceforge.net/, August 2015.