Magic Mirror Using Kinect

By

Anika Binte Habib

Adnan Asad

Wasiq Bin Omar

A thesis to submitted in partial fulfillment of the requirements for the Bachelors In Science Degree.

Bachelor of Computer Science and Engineering

BRAC University, December 2015

Abstract

In this modern civilization, everyone is depended on technology and technological revolution has made our lives easier in every aspect. With the blessings of technologies, shopping is been brought into the clients palm and there is no doubt about the fact that online shopping has replaced the way of traditional shopping for daily goods and clothing. Now a day, everyone enjoys online shopping because it eliminates the hassles of billing queues and most importantly getting stuck in traffic jam which is really time consuming. Online shopping has made peoples works less in many ways like most of the online shopping offers free shipping, cash on delivery and different kinds of discounts. Therefore, online shopping has been accepted widely all over the world. However, there is a small drawback of online shopping which might loses peoples' attraction on online shopping and that is people cannot try on their desire clothes. Our goal is to build a real time virtual dress up system consists of multiple tasks including extraction of the users' different body parts from video stream, alignment of models, torso detecting, and resizing input dress images and dress up using Kinect. Extraction of the users' different body parts allow us to create augmented reality environment by separating the user area from the video stream and imposing it onto the virtual environment in the user interface. We also have used 3D locations for joint positioning, rotating and system so that user can command to open or close the app. By integrating all the techniques, our system will produce prominently credible and well fitted virtual dress up results.

Declaration

I hereby declare that this thesis is based on the results found by myself. I have gone through numerous reference papers from researchers of both national and foreign universities, and a full list is mentioned at the end of the paper. This thesis, neither in whole nor in part, has been previously submitted for any degree.

Signature of the Supervisor	Signature of the Author
______________________	______________________
Mr. Md. Zahangir Alom	Anika Binte Habib
Signature of the co supervisor	______________________
	Adnan Asad
______________________	______________________
Risul Karim	Wasiq Bin Omar

Certificate

This is to certify that the thesis on “Magic Mirror Using Kinect” is a bona-fide record done by Authors name for the fulfillment of the degree of Bachelor of Science in Computer Science and Engineering (BSCSE) from BRAC University.

The thesis has been prepared under my guidance and is a record of bona-fide work carried out successfully.

Supervised By

Md. Zahangir Alom

Lecturer III

Department of Computer Science & Engineering

BRAC University

Dhaka, Bangladesh

Acknowledgement

It is an immense pleasure for us to show our gratitude and heartfelt thanks to our respected thesis supervisor Md. Zahangir Alom and co-supervisor Risul Karim who agreed to supervise this thesis. Without their great guidance and dedication we would not have come this far and could not have accomplished our thesis paper. We would like to thank them for their patience and support throughout the process and helped us whenever we needed them over the past year.

At the outset, we would like to express our gratitude to our beloved and respected acting chairperson Dr. Md. Khalilur Rhaman for his support and blessing to accomplish this project.

We would also like to express our gratefulness to our family and friends for their endless moral support and inspiring to finish this project.

Lastly, we would like to thank BRAC University for giving us opportunity to complete our B.Sc degree in Computer Science and Engineering and constant support

Table of content

List Of Figure:

Chapter 1: Introduction

1.1 Goals:

Technology has brought shopping in customers palm and online shopping is highly appreciated all over the world. Most current online shopping systems suggest many types of catalogs and photos, size, colors and fabric information of their products. However, this information is not enough for customers to understand about their products better. The main flaw of online shopping system is that customers cannot try their clothes before purchasing them. After researching about online shopping, we found that customers felt riskier than any other traditional retail methods when they went attire shopping on the web because of the deficiency of the palpable knowledge. Sometimes they found that the clothes are not well fitted after purchasing them. So it is hard for them to choose the appropriate outfit for them. Our main goal of this project is to increase the time efficiency and improve the accessibility of clothes try on by creating virtual dress-up room environment.

1.2 Methodology and Approaches:

This task has two main alignments which are the user and the cloth models with accurate position, scale, rotation and ordering. The first step of these problems is to detect users and their body parts. In several literatures, many approaches are proposed for body part detection, skeletal tracking and posture estimation. The problem can be brilliantly managed by the means of simple software like visual studio. In our proposed system we have used Microsoft Kincet sensor to create a tag free real time augmented reality dressing room application.

Our approaches can be summarized as follows:

1. To develop 2D models for clothes and human body with precise mechanical control
2. To develop a user-friendly application for users to try out different types of clothes for visualization using Kinect.

This report will mainly focus on the first task to develop 2D models for cloth and the calculation of interactive with the human body.

1.2.1 Approach for cloth-fitting

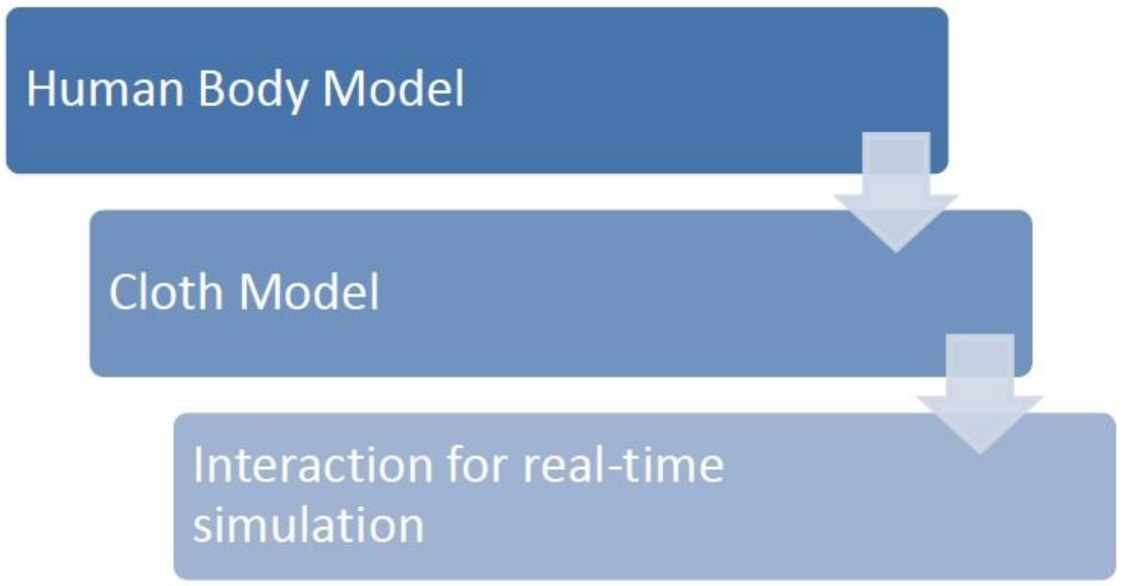

Fig 1: Model Approach

Firstly, we will detect a 3D human model according to the user's dimensions (the body shape, height, width, length of limbs etc.) from the data captured by Kinect. The whole human model will always follow the motion of the skeleton model captured by the Kinect. That is, how the user moves will be reflected by the skeleton model, and our human model will do the same movements.

Secondly, a cloth model will be created according to some mechanics such as frictional forces, gravity, elasticity etc. The cloth model is used to build various types of clothes for dressing on.

In real-time, the interaction takes place between the human model and cloth model. The clothes will be fitted on the human model, and will move in the same way as the human model moves. Hence, a realistic simulation on fitting is done by the interaction between our two models.

Our main methodology for this proposed system is to create a virtual dress up system by using Microsoft Kinect SDK due to its robust and practical skeletal tracking algorithm.

An application with user interface is developed in order to test practicality and performance. The user interface allows the user to choose a t-shirt, pants and caps by making a hand gesture towards to it. It also allows users to show or close the window by using their voice.

1.3 Challenges:

The main aim of this thesis is to create a Virtual Dressing Room that realistically reflects the 2D appearance of the cloth model and the adaption of specific bodies of different persons depending on their body measurements. The main challenge here is to fit the 2D clothes perfectly on the users' bodies. Microsoft Kinect, an innovative technology which provides a new way of interaction between humans and the computer. 2D models of the cloth is overlaid with the color image from the camera obtain the function of a virtual mirror. From the depth image of the Microsoft Kinect, the skeleton is extracted and the position and the orientation of the cloths are adapted in regard to joint position and human body measurement. To achieve a realistic simulation of the cloth, we used the Microsoft Kinect SDK to implement application programming interfaces for developers including a skeletal body tracker method.

1.4 Related Works on the problem:

The current method of online shopping does not ensure the perfect size of the clothing. The major setback of the online shopping is that the products are being returned due to wrong idea of the clothing. To solve this problem, several approaches are proposed for body part detection, skeletal tracking and posture estimation.

> Kjaersia et al. [1] propose a tag-based approach which on requires manual labeling of body parts in order to create an augmented reality of the customer wearing a cloth simulation.
>
> Use of shape descriptors such as Histograms of Oriented Gradients (HOG) can also be an option for detection of the user and body posture estimation.
>
> Onishi et al. [2] propose a HOG-based approach to estimate human postures as discrete states. However, use of a limited number of skeletal posture states may not be convenient for a continuous body tracking system.
>
> Microsoft Kinect has become the state of the art depth image sensor in the market after its launch in 2010. There is currently a quite intensive study to implement application

programming interfaces for developers including a skeletal body tracking method. OpenNI [3] organization develops an open-source, cross platform framework to process the data from a Kinect sensor. Microsoft Research has also released the Kinect SDK – which is freely available for non-commercial use – as a beta version with a robust real time skeletal body tracker.

Chapter 2: Literature Review

2.1 Detecting Human Body:

“A REAL TIME VIRTUAL DRESSING ROOM APPLICATION USING OPENCV”

By Aswini Vijay Araghavan, Indhumathi T.A, Jasothiri Chopra A.R.N and Kezia Miracline R

[4] For detecting users from video input, this paper has focused on face detection to determine whether human faces appear in a given image and where the faces are located at. In order to make further face recognition system more robust and easy to design, face alignment are performed to justify the scales and orientations of these patches. Besides serving as the pre-processing for face recognition, face detection could be used for region-of-interest detection, retargeting, video and image classification, etc. For such purpose VIOLA JONES algorithm is used.

“Histograms of Oriented Gradients for Human Detection”

By Navneet Dalal and Bill Triggs

[5] In this proposed paper, they studied on issue of feature sets for human detection, showing that locally normalized Histogram of Oriented Gradient (HOG) descriptors provide excellent performance relative to other existing feature sets including wavelets. They implemented HOG and systematically study the effects of the various choices on detector performance.

2.1.1 Skin Color Segmentation:

“Face Segmentation Using Skin Color Map in Videophone Application”

By Douglas Chai and King N. Ngan

[6] The main objective of this research is to design a system that can find a person’s face from a given image data. This problem is commonly referred to as face location, face extraction or face

segmentation. Regardless of which terminology, they all share the same objective. However, the problem usually deals with finding the position and contour of a person's face since its location is unknown but given the knowledge of its existence. If not then there is also need to discriminate between "image containing faces" and "image not containing faces". This is known as face detection. Nevertheless, this paper focuses on face segmentation

2.2 User Extraction

"Tailoring Measurement and virtual Try-On"
By Honey Gandhi, Moulik Jain and Ayesha Butalia

[7] This paper has given a general idea of user extraction. Extraction of user allows us to create an augmented reality environment by isolating the user area from the image and superimposing it onto a virtual environment in the user interface. The camera provides the image. When the device is working, image is segmented in order to separate background from the user. The Background is removed by blending the RGBA image with the segmented image for each pixel by setting the alpha channel to zero if the pixel does not lie on the user.

"A Virtual Dressing Room based on Depth Data"

By Philipp Presle

[8] Author of this paper has given idea of optical tracking techniques which are calculating the position and orientation of a point in space by using passive reflectors (e.g. spheres) that are placed all over the human body. They are emitted with light (usually infrared light) and this light is then captured from a camera. Using multiple cameras, the three dimensional data can be obtained by implicating their captured view and the orientation and location in relation to each other camera using triangulation. Before tracking starts, a calibration is necessary as well. The developed application uses an avatar which is moving in a virtual scene according to captured positions. An optical tracking system is used that is based on passive reflective markers that are placed on the required points all over the body. An emitter is sending out the infrared light which is sent back by the reflectors. This light is then captured by multiple cameras and output to a computer, which is analyzing the data. By using triangulation on the captured positions the particular 3D locations of the markers can be extracted. Before the capturing of a person can start, additional calibration steps have to be taken into account. Also, reflectors have to be

positioned on the appropriate locations of the body based on the placement rules. In a last step, the movement is assigned in real-time to an avatar which is then able to move in a virtual environment.

2.3 Joint Position and skeleton driven deformation:

“Real-Time Human Pose Recognition in Parts from Single Depth Images”

By Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp and Mark Finocchio

[9] In this research paper it is proposed that, for positions of 3D skeletal joint, they used tracking algorithm to self-initialize and recover from failure. A simple option is to accumulate the global 3D centers of probability mass for each part, using the known calibrated depth. However, outlying pixels severely degrade the quality of such a global estimate. Instead we employ a local mode-finding approach based on mean shift with a weighted Gaussian kernel.

“Automatic Modeling of Virtual Humans and Body Clothing” By Nadia Magnenat – Thalmann, Hyewon Seo, Frederic Cordier

[10] The main focus of this paper is the dynamic shape of the body. They focused on the method for systematically deforming the skin shape during the animation. The skeleton-driven deformation, a classicalmethod for the basic skin deformation is perhaps the most widely used technique in 3D character animation. The concept of Joint-dependent Local Deformation (JLD) operators to smoothly deform the kin surface and this technique has been given various names such as Sub-Space Deformation (SSD), linear blend skinning, or smooth skinning. This method works first by assigning a set of joints with weights to each vertex in the character.

2.4 For Model Positioning and rotating:

“A Real Time Virtual Dressing Room Application using Kinect”

By Furkan Isıkdogan and Gokcehan Kara

[11] In this proposed system, they focused on model positioning and rotating. The skeletal tracker returns the 3D coordinates 20 body joints in terms of pixel locations for x and y coordinates and meters for the depth. They also illustrated the 9 of the 20 joints which are used to locate the parts of the 2D cloth model. One may notice flickers and vibrations on the joints due to the frame based recognition approach. This problem is partially solved by adjusting the smoothing parameters of the skeleton engine of the Kinect SDK.

“Magic Mirror: A Virtual Dressing Room”

By Young In Yeo

[12] The magic mirror system consists of template models and several databases for body, garment, motion and background. The system uses the example-based interpolation approach for avatar creation. This method modifies 3D template models for each gender into reconstructed model best matches the extracted silhouette from captured images. For the modification process, this system searches body DB where a collection of range scans of real human body is structured and statistically processed with PCA (principal component analysis), which is the effective mathematical procedure that transforms a number of correlated variables-vertices on body shape- into a smaller number of principal.

Chapter 3: System description

<u>3.1 Development Environment Microsoft Visual Studio</u>

3.1.1 Introduction

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs for Microsoft Windows superfamily of operating systems, as well as web sites, web applications and web services. Visual Studio uses Microsoft software development platforms such as Windows API, Windows Forms, Windows Presentation Foundation, Windows Store and Microsoft Silverlight. It can produce both native code and managed code. Visual Studio includes a code editor supporting IntelliSense as well as code refactoring. The integrated debugger works both as a source-level debugger and a machine-level debugger. Other built-in tools include a forms designer for building GUI applications, web designer, class designer, and database schema designer. It accepts plug-ins that enhance the functionality at almost every level including adding support for source-control systems (like Subversion and Visual SourceSafe) and adding new toolsets like editors and visual designers for domain-specific languages or toolsets for other aspects of the software development lifecycle (like the Team Foundation Server client: Team Explorer). Visual Studio supports different programming languages and allows the code editor and debugger to support (to varying degrees) nearly any programming language, provided a language-specific service exists. Built-in languages include C/C++[5] (via Visual C++), VB.NET (via Visual Basic .NET), C# (via Visual C#), and F# (as of Visual Studio 2010[6]). Support for other languages such as M, Python, and Ruby among others is available via language services installed separately. It also supports XML/XSLT, HTML/XHTML, JavaScript and CSS. Individual language-specific versions of Visual Studio also exist which provide more limited language services to the user: Microsoft Visual Basic, Visual J#, Visual C#, and Visual C++. Microsoft provides "Express" editions of its Visual Studio at no cost. Commercial versions of Visual Studio along with select past versions are available for free to students via Microsoft's DreamSpark program.

3.1.2 Architecture

Visual Studio does not support any programming language, solution or tool intrinsically, instead it allows the plugging of functionality coded as a VSPackage. When installed, the functionality is available as a Service. The IDE provides three services: SVsSolution, which provides the ability to enumerate projects and solutions; SVsUIShell, which provides windowing and UI functionality (including tabs, toolbars and tool windows); and SVsShell, which deals with registration of VSPackages. In addition, the IDE is also responsible for coordinating and enabling communication between services. All editors, designers, project types and other tools are implemented as VSPackages. Visual Studio uses COM to access the VSPackages. The Visual Studio SDK also includes the Managed Package Framework (MPF), which is a set of managed wrappers around the COM-interfaces that allow the Packages to be written in any CLI compliant language. However, MPF does not provide all the functionality exposed by the Visual Studio COM interfaces.The services can then be consumed for creation of other packages, which add functionality to the Visual Studio IDE. Support for programming languages is added by using a specific VSPackage called a Language Service. A language service defines various interfaces which the VSPackage implementation can implement to add support for various functionalities. Functionalities that can be added this way include syntax coloring, statement completion; brace matching, parameter information tooltips, member lists and error markers for background compilation. If the interface is implemented, the functionality will be available for the language. Language services are to be implemented on a per-language basis.

The implementations can reuse code from the parser or the compiler for the language. Language services can be implemented either in native code or managed code. For native code, either the native COM interfaces or the Babel Framework (part of Visual Studio SDK) can be used. For managed code, the MPF includes wrappers for writing managed language services. Visual Studio does not include any source control support built in but it defines two alternative ways for source control systems to integrate with the IDE. A Source Control VSPackage can provide its own customized user interface. In contrast, a source control plugin using the MSSCCI (Microsoft Source Code Control Interface) provides a set of functions that are used to implement various source control functionality, with a standard Visual Studio user interface. MSSCCI was first used to integrate Visual SourceSafe with Visual Studio 6.0 but was later opened up via the Visual Studio SDK. Visual Studio .NET 2002 used MSSCCI 1.1, and Visual Studio .NET 2003 used MSSCCI 1.2. Visual Studio 2005, 2008 and 2010 use MSSCCI Version 1.3, which adds support for rename and delete propagation as well as asynchronous opening. Visual Studio supports running multiple instances of the environment (each with its own set of VSPackages). The instances use different registry hives (see MSDN's definition of the term "registry hive" in the sense used here) to store their configuration state and are differentiated by their AppId (Application ID). The instances are launched by an AppId-specific .exe that selects the AppId, sets the root hive and launches the IDE. VSPackages registered for one AppId are integrated with other VSPackages for that AppId. The various product editions of Visual Studio are created using the different AppIds. The Visual Studio Express edition products are installed with their own AppIds, but the Standard, Professional and Team Suite products share the same AppId. Consequently, one can install the Express editions side-by-side with other editions, unlike the other editions which update the same installation. The professional edition includes a superset of the VSPackages in the standard edition and the team suite includes a superset of the VSPackages in both other editions.

3.1.3 Features:

Code editor

Like any other IDE, it includes a code editor that supports syntax highlighting and code completion using IntelliSense for not only variables, functions and methods but also language constructs like loops and queries. IntelliSense is supported for the included languages, as well as for XML and for Cascading Style Sheets and JavaScript when developing web sites and web applications.Autocomplete suggestions are popped up in a modeless list box, overlaid on top of the code editor. In Visual Studio 2008 onwards, it can be made temporarily semi-transparent to see the code obstructed by it.The code editor is used for all supported languages.

The Visual Studio code editor also supports setting bookmarks in code for quick navigation. Other navigational aids include collapsing code blocks and incremental search, in addition to normal text search and regex search. The code editor also includes a multi-item clipboard and a task list. The code editor supports code snippets, which are saved templates for repetitive code and can be inserted into code and customized for the project being worked on. A management tool for code snippets is built in as well. These tools are surfaced as floating windows which can be set to automatically hide when unused or docked to the side of the screen. The Visual Studio code editor also supports code refactoring including parameter reordering, variable and method renaming, interface extraction and encapsulation of class members inside properties, among others. Visual Studio features background compilation (also called incremental compilation). As code is being written, Visual Studio compiles it in the background in order to provide feedback about syntax and compilation errors, which are flagged with a red wavy underline. Warnings are marked with a green underline. Background compilation does not generate executable code, since it requires a different compiler than the one used to generate executable code. Background compilation was initially introduced with Microsoft Visual Basic but has now been expanded for all included languages.

Debugger:

Visual Studio includes a debugger that works both as a source-level debugger and as a machine-level debugger. It works with both managed code as well as native code and can be used for debugging applications written in any language supported by Visual Studio. In addition, it can also attach to running processes and monitor and debug those processes. If source code for the running process is available, it displays the code as it is being run. If source code is not available, it can show the disassembly. The Visual Studio debugger can also create memory dumps as well as load them later for debugging. Multi-threaded programs are also supported. The debugger can be configured to be launched when an application running outside the Visual Studio environment crashes. The debugger allows setting breakpoints (which allow execution to be stopped temporarily at a certain position) and watches (which monitor the values of variables as the execution progresses). Breakpoints can be conditional, meaning they get triggered when the condition is met. Code can be stepped over, i.e., run one line (of source code) at a time. It can either step into functions to debug inside it, or step over it, i.e., the execution of the function body isn't available for manual inspection. The debugger supports Edit and Continue, i.e., it allows code to be edited as it is being debugged (32 bit only; not supported in 64 bit). When debugging, if the mouse pointer hovers over any variable, its current value is displayed in a tooltip ("data tooltips"), where it can also be modified if desired. During coding, the Visual Studio debugger lets certain functions be invoked manually from the immediate tool window. The parameters to the method are supplied at the immediate window.

3.2 Introduction to Kinect Sensor:

3.2.1 General component

The components of Kinect for Windows are mainly the following:

1. Kinect hardware: Including the Kinect sensor and the USB hub, through which the sensor is connected to the computer;

2. Microsoft Kinect drivers: Windows 7 drivers for the Kinect sensor;

3. NUI API: core of the Kinect for the set of Windows API, supports fundamental image and device management features like access to the Kinect sensors that are connected to the computer, access to image and depth data streams from the Kinect image sensors and delivery of a processed version of image and depth data to support skeletal tracking.

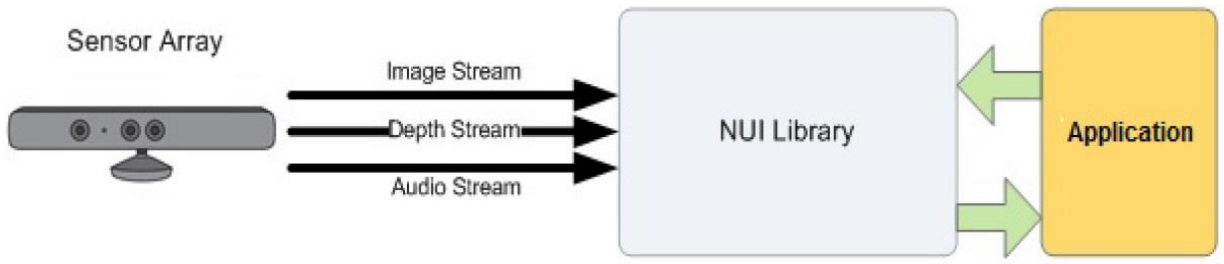

Fig 2: Hardware and software interaction with an application

Kinect sensor mainly provides 3 streams: Image stream, depth stream and audio stream, with detected range from 1.2 to 3.5 meters. At this stage, the first two streams would be utilized for development of human model, cloth simulation and GUI.

[16] The middle camera is a 640×480 pixels @ 30 Hz RGB camera, providing image stream which is delivered as a succession of still-image frames for the application. The quality level of color image determines how quickly data is transferred from the Kinect sensor array to the PC which is easy for us to optimize the program on different platform. The available color formats determine whether the image data that is returned to our application code is encoded as RGB.

The middle camera is a 640×480 pixels @ 30 Hz RGB camera, providing image stream which is delivered as a succession of still-image frames for the application. The quality level of color image determines how quickly data is transferred from the Kinect sensor array to the PC which is easy for us to optimize the program on different platform. The available color formats determine whether the image data that is returned to our application code is encoded as RGB.

The leftmost one is the IR light source with corresponding 640×480 pixels @ 30 Hz IR depth-finding camera with standard CMOS sensor on the right, which mainly provide the depth data stream. This stream provides frames in which the high 13 bits of each pixel give the distance, in millimeters, to the nearest object at that particular x and y coordinate in the depth sensor's field of view.

3.2.2 Nui skeleton API

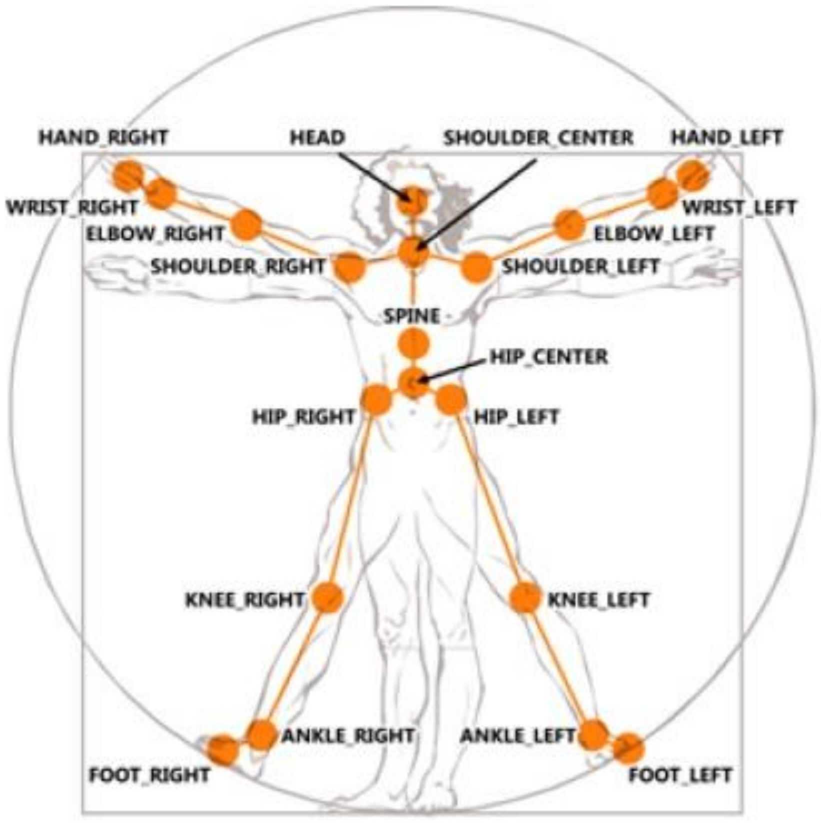

Fig 3: Skeleton joint positions relative to the human body

Among NUI API, NUI Skeleton API provides information about the location of users standing in front of the Kinect sensor array, with detailed position and orientation information. Those data are provided to application code as a set of 20 point, namely skeleton position. This skeleton represents a user’s current position and pose. Our applications can therefore utilize the skeleton data for measurement of different dimension of users’ part and control for GUI. Skeleton data are retrieved as aforementioned image retrieval method: calling a frame retrieval method and passing a buffer while our application can then use an event model by hooking an event to an event handler in order to capture the frame when a new frame of skeleton data is ready.

Chapter 4: Design

In this chapter I will take a look at the design aspects of the Virtual Dressing Room. An introduction of the provided features will be given. Subsequently, a basic setup for the application will be presented. Thereafter the program flow will be outlined, an overview of garment modeling and adaption will be given, the design of the user interface and at last the debug functions will be explained.

4.1 features:

As already stated in the introduction, trying on clothes in stores can be a time-consuming task. The goal of this thesis is to introduce a virtual fitting room that allows to take a look at several pieces of garment in a short period of time. The appearance of garment is depending on several factors: First of all, a realistic 2D model of the cloth is essential. This includes an exact mesh modeled down to the last detail also resembling the correct proportions of a human body. Second, a realistic look of the texture is essential. Third, to achieve a realistic behavior of the garment, physic simulations will be involved in the application. If trying on t-shirt for instance this will add relevant and realistic characteristics to the cloth letting it flutter as the user moves in front of the Kinect. Another feature that is contributing to realism of the Virtual Dressing Room is the adaption of the cloth to various sizes of the body. Depending on the measurements taken during the initialization step, the cloth colliders and therefore the particular parts of the cloth itself will adapt in respect to the body measurements of the user.

4.2 Program Flow

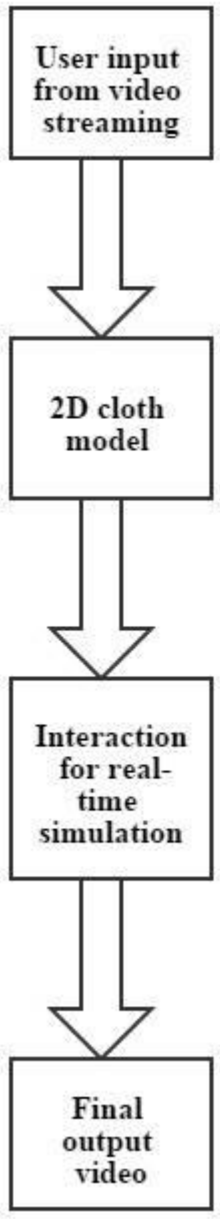
User input from video streaming
2D cloth model
Interaction for real-time simulation
Final output video

Fig 4: Steps of the process

This section will take a closer look on the program flow of the application from the start of the application up to the processing steps of the garment.

First of all, after the start of the program, the standard Unity dialog appears. This dialog allows a setup of the screen resolution, an option for full screen as well as selections for the desired graphics quality. Furthermore, a second tab is presenting various settings to the user in order to set/reset different key strokes for debugging purposes. This includes the display of the debug output (e.g. frames per second, currently active users, etc.), the rendering of the colliders that are in general responsible for the movement of the interactive cloth and an option to draw the skeleton of the user. A key to quit the application and shut down all the allocated resources is provided as well. These settings shouldn't be presented to the user, since they are just basic debug settings and are intended for administration purposes only.

Shortly after the program is started the Virtual Dressing Room is ready to accept users. The users can then try on garment and can interact with the virtual dress up by utilizing the Kinect in, combination with a screen, a so-called virtual mirror. These steps are described now.

The next part treats the general tracking process. After the program is fully started and the mirrored camera stream is appearing on the screen, it is ready to register users. If a user then enters the Kinect's field of view, the person is prompted to move the body into calibration pose. After the pose has been executed and measuring is finished, the registration is complete. It isnow possible to access the skeleton parameters with the help of OpenNI. If problems during the calibration pose are occurring, the process starts from the beginning. Subsequently, the user is now able to interact with the Virtual Dressing Room and chose between different garments. In general, the user can dress different shirts. The user has to select the desired garment that he wants to change by placing his hand over the icon. Details on the user interface will be explained in a later section. If the user has chosen a piece of cloth, the cloth gets attached to the colliders. The first positions of the colliders are based upon the initial joint positions of the human body model since the garment mesh depends on this model. The colliders are then moved to the according joint positions of the user in real-time. Furthermore the particular interactive cloth settings and additional informations are parsed from an XML file. This file is storing details on the cloth positions, price information and physic settings for every piece of cloth that is available for dress-up in the Virtual Dressing Room. Besides it is also possible for the user to enable background separation. This means, that the user can dress-up the garment in different surroundings. If the person in front of the mirror likes to take a look at a cloth to wear during the night, the background of the scene can be changed accordingly. At the end of the whole dress-up process - after a user has left the viewpoint of the Kinect device - the system resets. Furthermore this means that the application is rolling back to the first step and is waiting for a new user.

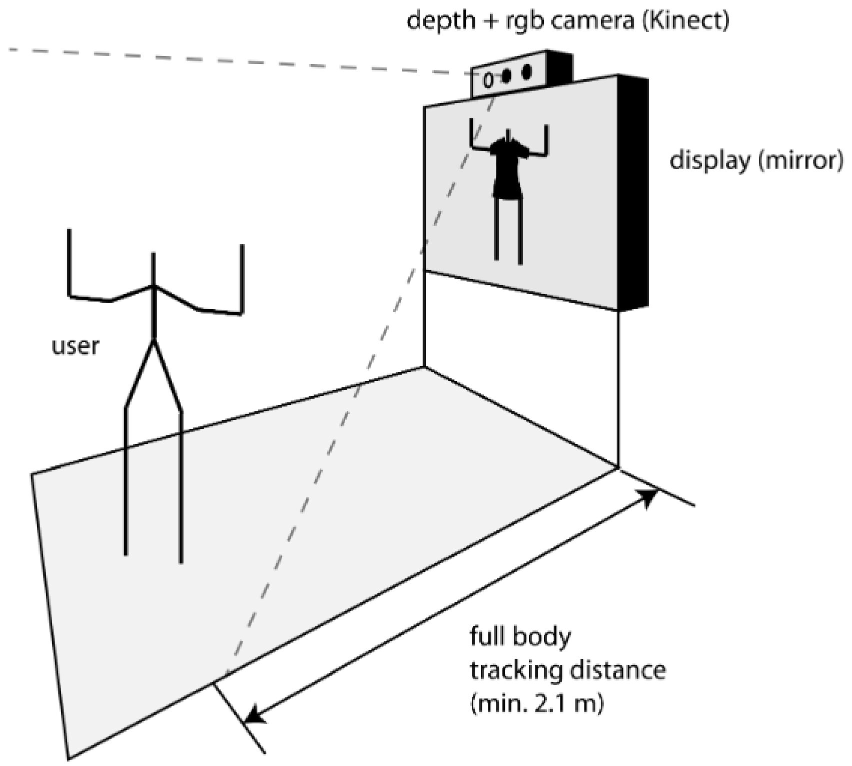

Fig 5: An intended basic setup of the fitting room. It consists of the Kinect device, a display as well as a computer (not illustrated here). The person in the front of the Kinect is interacting with the Virtual Dressing Room, the display showing the superimposed t-shirt selected by the user.

Chapter 5: Implementation and Interaction between human and computer

5.1 User Extraction

Extraction of user allows us to create an augmented reality environment by isolating the user area from the video stream and superimposing it onto a virtual environment in the user interface. Furthermore, it is here a useful way to determine the region of interest that is also used for skin detection which is explained in Subsection 5.2. The Kinect SDK provides the depth image and the user ID. When the device is working, depth image is segmented in order to separate background from the user. The background is removed by blending the RGBA image with the segmented depth image for each pixel by setting the alpha channel to zero if the pixel does not lie on the user.

5.2 Color Stream

Since the model is superimposed on the top layer, the user always stays behind the model which restricts some possible actions of the user such as folding arms or holding hands in front of the t-shirt. In order to solve that issue skin colored areas are detected and brought to the front layer.

Color data available in different resolutions and formats are provided through the color stream. The color image's format determines whether color data are encoded as RGB, YUV or Bayer. The RGB format represents the color image as 32 bit, linear X8R8G8B8 formatted color bitmap. A color image in RGB format is updated at up to 30 frames per-seconds at 640*480 resolutions and at 12 frames per second in high definition 1280*960 resolutions. The YUV format represents the color image as 16 bit, gamma corrected linear UYVY formatted color bitmap, where the gamma correction in YUV space is equivalent to standard RGB gamma in RGB space. According to the 16bit pixel representation, the YUV format uses less memory to hold bitmap data and allocates less buffer memory. The color image in YUV format is available only at the 640*480 resolution and only at 15 fps.

The Bayer format includes more green pixels values than blue or red and that makes it closer to the physiology of human eye. The format represents the color image as 32 bit, linear X8R8G8B8 formatted color bitmap in standard RGB color space. Color image in Bayer format is updated at

30 frames per seconds at 640*480 resolutions and at 12 frames per second in high definition 1280*960 resolutions.

HSV and YCbCr color spaces are commonly used for skin color segmentation. In this work we preferred YCbCr color space and the RGB images are converted into YCbCr color space by using following equations:

$$Y = 0.299R + 0.587G + 0.114B$$
$$C_b = 128 - 0.169R - 0.332G + 0.5B$$
$$C_r = 128 + 0.5R - 0.419G - 0.081B$$

Chai and Ngan [14] report that the most representative color ranges of human skin on YCbCr color space. A threshold is applied to the color components of the image within the following ranges:

$$77 < C_b < 127$$
$$133 < C_r < 173$$
$$Y < 70$$

Since we have the extracted user image as a region of interest the threshold is applied only on the pixels that lie on the user. Thus, the areas on the background which may resemble with the skin color are not processed.

5.2.1 Color frame class

A color image is represented and implemented by the ColorFrame class. The class contains information about a color image format, image dimensions and image data. The image data are represented as a byte array. The color image is stored in ARGB format, it means, the image pixels are stored as a sequence of four bytes in order blue, green, red and alpha channel.

TheColorFrame class provides an interface for a basic manipulation with pixel data such as getting and setting pixel color at given and coordinates and flipping the image. The class also provides a method Clone() for creation of its copy.

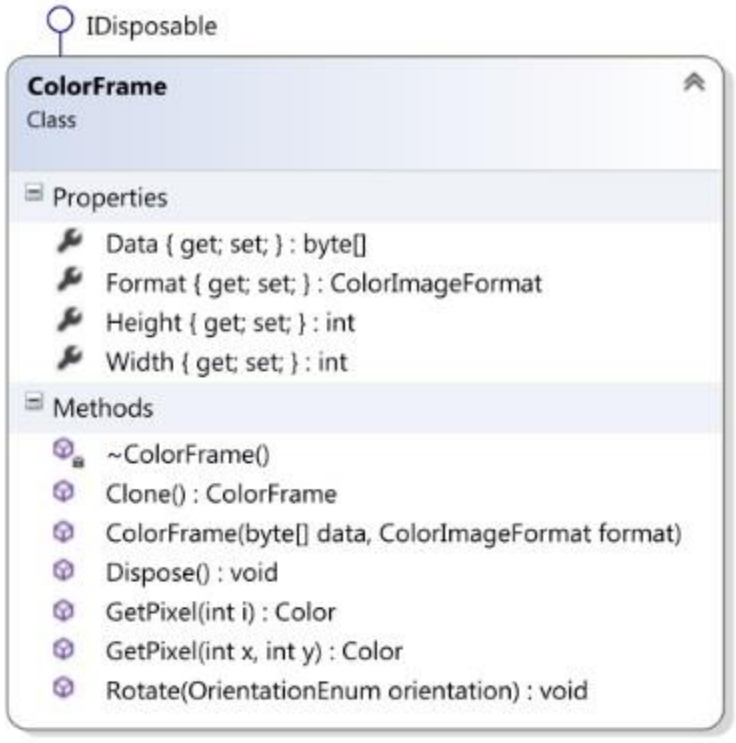
IDisposable
ColorFrame
Class
Properties
Data { get; set; } : byte[]
Format { get; set; } : ColorImageFormat
Height { get; set; } : int
Width { get; set; } : int
Methods
~ColorFrame()
Clone() : ColorFrame
ColorFrame(byte[] data, ColorImageFormat format)
Dispose() : void
GetPixel(int i) : Color
GetPixel(int x, int y) : Color
Rotate(OrientationEnum orientation) : void

Fig 6: A class diagram describing the color frame data structure.

5.3 Tracking

The tracking algorithm of Kinect SDK [1] is created using a large training data set with over 500k frames distributed over many video arrays. The data set have been generated synthetically by rendering depth images of common poses of actions which are frequently performed in video games such as dancing, running or kicking, along with their labeled ground truth pairs. In order eliminate same poses in the data set; a small offset of 5cm is used to create a subset of poses that all poses are at least as distant as the offset to each other. Pixel depths are then normalized before training so that they become translation independent.

5.3.1 Skeletal tracking

The crucial functionality provided by the Kinect for Windows SDK is the Skeletal Tracking. The skeletal tracking allows the Kinect to recognize people and follow their actions. It can recognize up to six users in the field of view of the sensor, and of these, up to two users can be tracked as the skeleton consisted of 20 joints that represent locations of the key parts of the user's body. The joints locations are actually coordinates relative to the sensor and values of X, Y, Z coordinates are in meters.

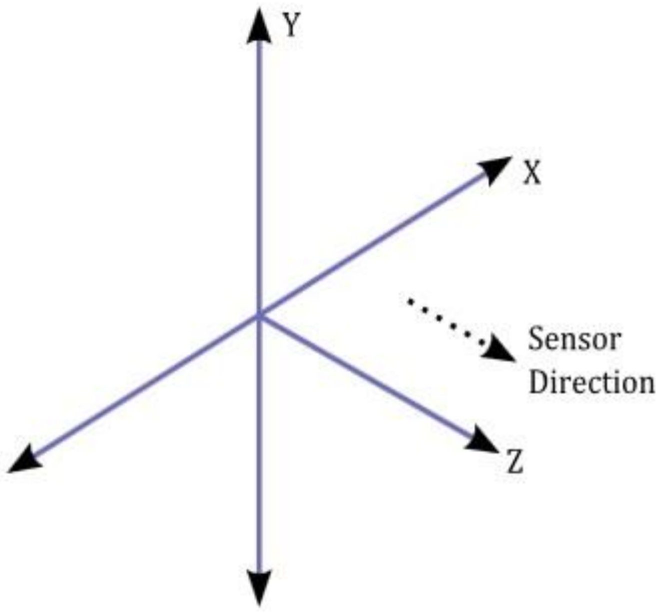

Fig 7: Illustration of the skeleton space

The tracking algorithm is designed to recognize users facing the sensor and in the standing or sitting pose. The tracking sideways poses is challenging as part of the user is not visible for the sensor. The users are recognized when they are in front of the sensor and their head and upper body is visible for the sensor. No specific pose or calibration action needs to be taken for a user to be tracked. The skeletal tracking can be used in both range modes of the depth camera. By using the default range mode, users are tracked in the distance between 0.8 and 4.0 meters away, but a practical range is between 1.2 to 3.5 meters due to a limited field of view. In case of near range mode, the user can be tracked between 0.4 and 3.0 meters away, but it has a practical range of 0.8 to 2.5 meters. The tracking algorithm provides two modes of tracking. The default mode is designed for tracking all twenty skeletal joints of the user in a standing pose. The seated mode is intended for tracking the user in a seated pose. The seated mode tracks only ten joints of upper body. Each of these modes uses different pipeline for the tracking. The default mode detects the user based on the distance of the subject from the background. The seated mode uses movement to detect the user and distinguish him or her from the background, such as a couch or a chair. The seated mode uses more resources than the default mode and yields a lower throughput on the same scene. However, the seated mode provides the best way to recognize a skeleton when the depth camera is in near range mode. In practice, only one tracking mode can be used at a time so it is not possible to track one user in seated mode and the other one in default mode using one sensor.

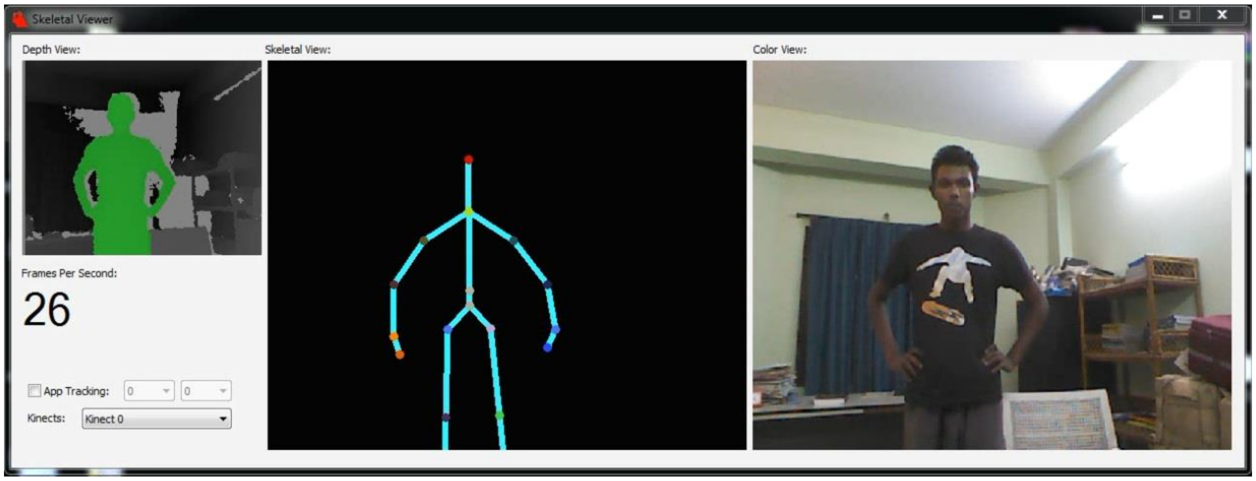

Fig 8: Skeleton Tracking

5.3.2 Model Positioning and Rotation

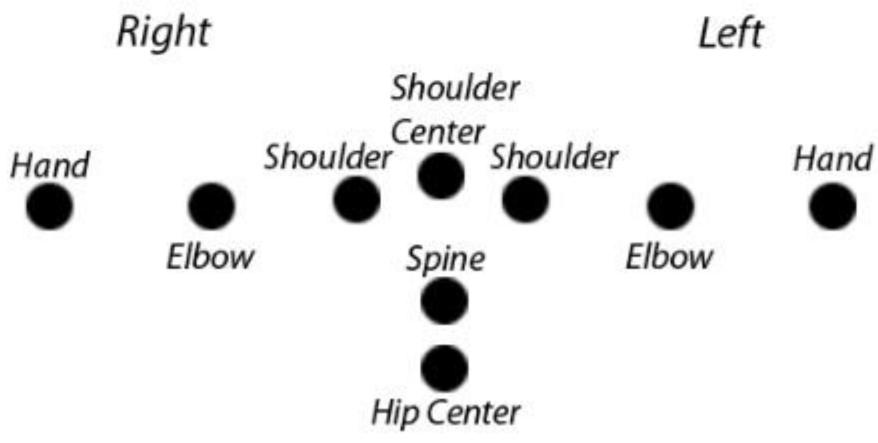

Fig 9: Body joints are used for positioning of the model

The skeletal tracker returns the 3D coordinates 20 body joints in terms of pixel locations for x and y coordinates and meters for the depth. Figure.. illustrates the 9 of the 20 joints which are used to locate the parts of the 2D cloth model. One may notice flickers and vibrations on the joints due to the frame based recognition approach. This problem is partially solved by adjusting the smoothing parameters of the skeleton engine of the Kinect SDK.

One may notice flickers and vibrations on the joints due to the frame based recognition approach. This problem is partially solved by adjusting the smoothing parameters of the skeleton engine of the Kinect SDK.

The model positioning and rotating parts of the models are defined as the main axis of the bodies and shoulders.

```
image1.Width = pointR.X - pointL.X + 10 + (((pointR.X - pointL.X) * 115) / 100);
image1.Height = pointHL.Y - pointHR.Y + 100;
```

here the main body axis is defined as the line between the shoulder center and the hip center, the axes for arms are defined as the lines between the corresponding shoulders and elbows and atan2 is a function similar to the inverse tangent but only defined in the interval(0;pi). For each model part an anchor point is defined as the center of rotation which is set to the middle of the corresponding line of axis.

5.4 Model Scaling:

A naive approach for model scaling can be to employ the distance between the joints as a scaling factor. This approach can be convenient to accommodate the height and weight related variations of users. However the distance between the joints is not sufficiently accurate to scale the model to simulate the distance from the sensor. Hence, we tried to adopt a hybrid approach in order to perform shape and distance based scaling with enhanced accuracy. The depth based scaling variable DS is defined straightforwardly as,

$$DS = \frac{\mathbf{s}_{\mathbf{model}}}{z_{spine}}$$

Where smodel is a vector of default width and height of the model when the user is one meter distant from the sensor and zspine is the distance of the spine of the user to the sensor. The shape based length of an arm Lkarm is defined as the Euclidean distance between the corresponding shoulder and the elbow. The width Wbody and height Hbody of the body are also calculated with a similar fashion by using the distance between the shoulders and the distance between the shoulder center and the hip center correspondingly as follows:

$$\begin{aligned}
H^k_{arm} &= \sqrt{(x^k_{shoulder} - x^k_{elbow})^2 + (y^k_{shoulder} - y^k_{elbow})^2} \\
W^k_{arm} &= H^k_{arm} \times \alpha \\
H_{body} &= |x_{shouldercenter} - x_{hipcenter}| \\
W_{body} &= |x^{left}_{shoulder} - x^{right}_{shoulder}|
\end{aligned}$$

Here x and y are the 2D coordinates of the joints, is constant the width to length ratio of the arms and k = {left, right}.

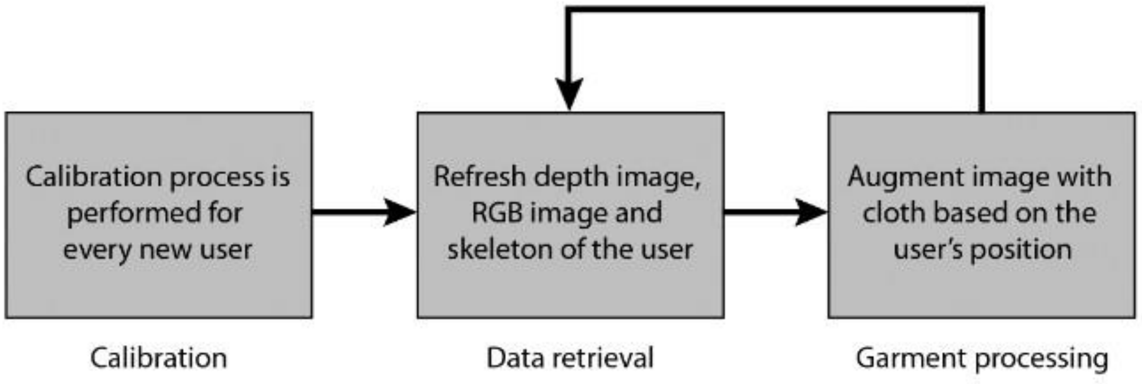
Calibration process is performed for every new user
Refresh depth image, RGB image and skeleton of the user
Augment image with cloth based on the user's position
Calibration
Data retrieval
Garment processing

Fig 10: An overview of the workflow for a single user is depicted in this illustration. Starting with the calibration step, up to the data retrieval process until finally reaching the algorithms dealing with garment processing (i.e. moving cloth, applying physics, etc.).

The last two steps are executed until the user leaves the tracking area. The system enters the calibration process again if a new user is detected.

5.5 User Interface

The user interface design was an essential part of the project since the interaction with the application is quite different to a 'normal' application that can be controlled with a computer mouse. Instead, other approaches to interact with the application have to be considered. The reason for this is not only the fact that a third dimension is added compared to the computer mouse, but moreover the Kinect allowing new ways of interaction that can involve all joints of a human body. This thesis takes a closer look on some different techniques that can be applied, for instance:

- Gesture recognition
- 3D interaction elements
- Voice Command

5.5.1 Gesture recognition

The first technique that I was implementing was a gesture recognition algorithm. The idea was based on a swipe gesture, furthermore distinguishing between a left and a right swipe. A simple swipe gesture would therefore change cloth, for example. The algorithm was based on a formula that is calculating the angle between the line defined by the 2 consecutive points (a, b) and the horizontal axis:

$$\alpha = atan(\frac{b_{t+1} - b_t}{a_{t+1} - a_t})$$

By comparing a set of calculations of the angle to a specific pattern (e.g. if most of the calculated angles of consecutive hand motions lie in between -20 to 20 degrees a swipe to the right side has been executed by the user), it becomes possible to identify gestures. Furthermore it may be necessary to limit the execution to a certain time in order to exclude wrong gestures. Though this method is quite useful for other purposes, it generated lots of errors during interaction for a user of the magic mirror application. The cause for that is the characteristic behavior performed by the users during the magic mirror of clothes. If a user was simply holding up his hands and taking a look at clothes from the right side and then turned his body and therefore the hands a bit to the left so that the user can take a look from the other side, the algorithm would have recognized a swipe gesture to the left. Since dealing with different parameters and adjusting the algorithm to ignore certain positions did not have the desired effect, other kinds of interaction had to be considered.

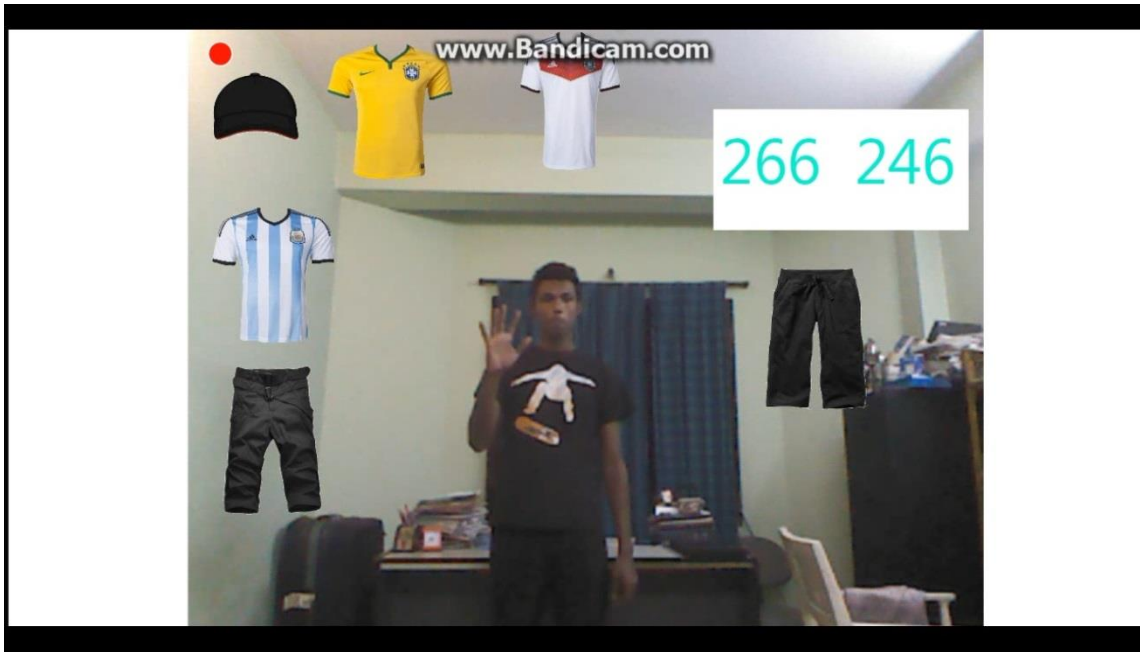

Fig 11: Hand movement for selecting the 2D model clothes

5.5.2 3D interaction elements:

The consideration was emphasized on 3D elements that are integrated into the Virtual Dressing Room in an augmented reality like manner. The basic idea of a concept was an interaction element that is always next to the customer. Moreover these elements should allow a person to quickly switch trousers, shirts and even furthermore change the actual background of the Kinect camera stream to try on clothes in different environments. More precisely this means that the operational elements are overlaid on the mirrored screen and are moving in respect to the position of the current user of the Kinect.

5.5.3 Voice Command:

With Kinect, voice recognition can be controlled. Kinect provides the means to access and manage an in-process speech recognition engine. The System.Speech.Recognition namespace contains Windows Desktop Speech technology types for implementing speech recognition. The Windows Desktop Speech Technology software offers a basic speech recognition infrastructure that digitizes acoustical signals, and recovers words and speech elements from audio input. Applications use the System.Speech.Recognition namespace to access and extend this basic speech recognition technology by defining algorithms for identifying and acting on specific phrases or word patterns, and by managing the runtime behavior of this speech infrastructure. SpeechRecognizer and SpeechRecognitionEngine objects generate events in response to audio input to the speech recognition engine. The AudioLevelUpdated, AudioSignalProblemOccurred, AudioStateChanged events are raised in response to changes in the incoming signal. The SpeechDetected event is raised when the speech recognition engine identifies incoming audio as speech. The speech recognition engine raises the SpeechRecognized event when it matches speech input to one of its loaded grammars, and raises the SpeechRecognitionRejected when speech input does not match any of its loaded grammars.

In this proposed system, voice command is used to open and close the app and dispatching time is 4 second.

5.6 Final result of the app:

In this section we will take a look at the tests and the results of the Virtual Dressing Room. In the first section we are going to provide a short presentation of the achieved results, illustrated with some screenshots.

5.6.1 Result and output:

During the testing stage special attention was paid to all of the implemented functions and how they behave on different participants. First of all the focus is set on the individual adaption of clothes since the dressing room should be usable for persons of varying measurements independent of their body and shoulder and arm length. 2D image is superimposed on the particular spots to simulate this behavior.

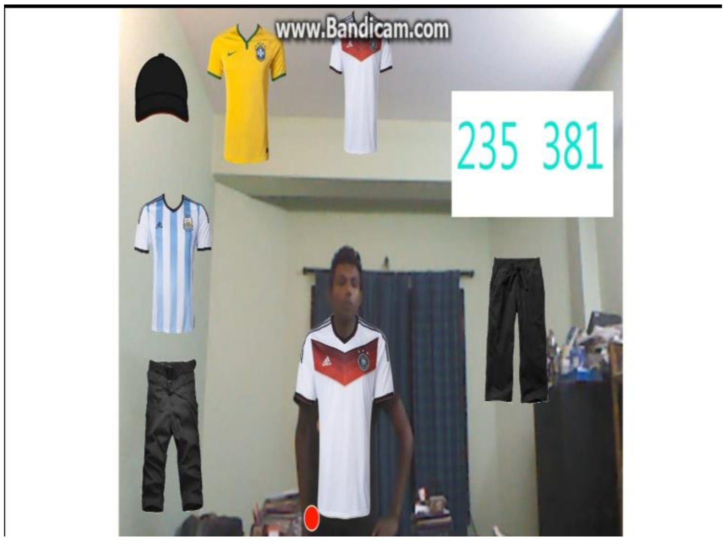

Fig 12: Superimposing cloths on 3D human body

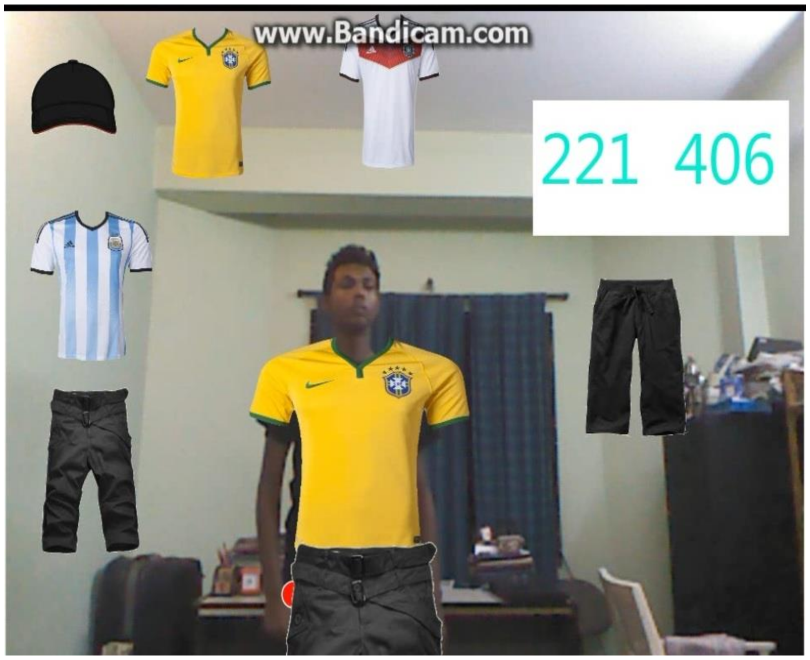

Fig 13: Superimposing cloths on 3D human body

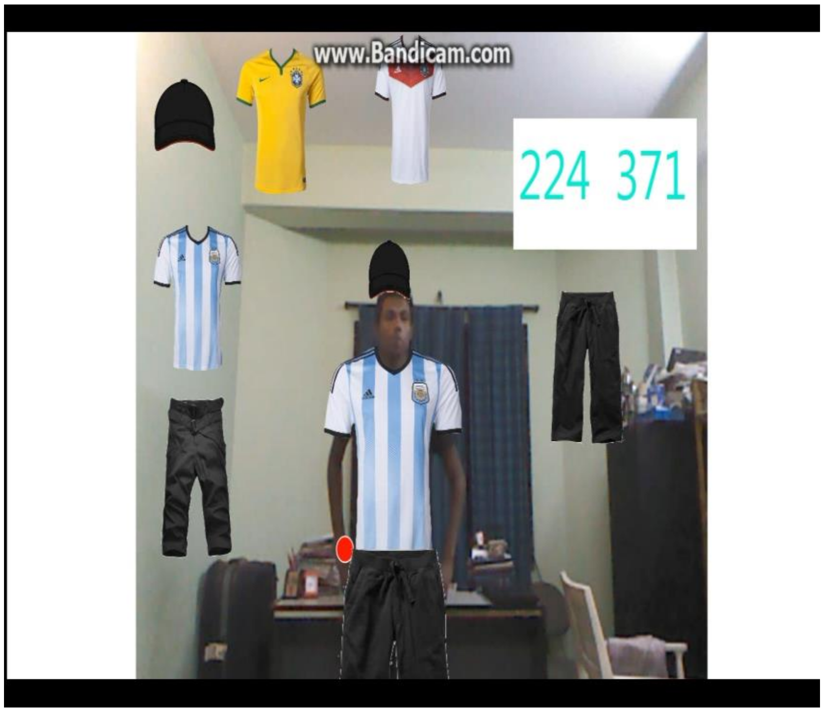

Fig 14: Superimposing cloths on 3D human body

5.6.2 Performance:

A performance measure is defined by employing the overlapping area of the ground truth model and the constructed model as the base criterion which is explicitly defined as,

$$P = \frac{A_c \cap A_g}{A_c \cup A_g}$$

where Ac is the area of the constructed model and Ag is the area of the ground truth model in terms of the number of pixels. A set of captures are obtained in order to test the performance of the system under different conditions.

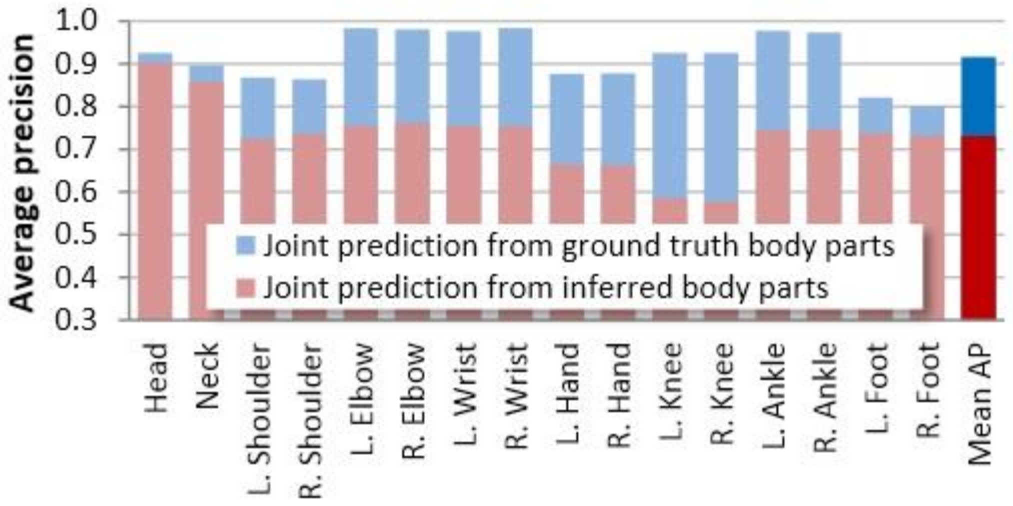

Fig 15: Joint prediction accuracy. We compare the actual performance of our system (red) with the best achievable result (blue) given the ground truth body part label

In Fig 15 we show average precision results on the synthetic test set, achieving 0.731 mAP. We compare an idealized setup that is given the ground truth body part labels to the real setup using inferred body parts. While we do pay a small penalty for using our intermediate body parts representation, for many joints the inferred results are both highly accurate and close to this upper bound. On the real test set, we have ground truth labels for head, shoulders and hands. An mAP of 0.984 is achieved on those parts given the ground truth body part labels, while 0.914 mAP is achieved using the inferred body parts. As expected, these numbers are considerably higher on this easier test set comparison with nearest neighbor. To highlight the need to treat pose recognition in parts and to calibrate the difficulty of our test set for the reader, we compare with two variants of exact nearest-neighbor whole-body matching in Fig.(15). The first, idealized, variant matches the ground truth test skeleton to a set of training exemplar skeletons with optimal rigid translational alignment in 3D world space. Of course, in practice one has no access to the test skeleton. As an example of a realizable system, the second variant uses chamfer matching [14] to compare the test image to the training exemplars. This is computed using depth edges and 12 orientation bins. To make the chamfer task easier, we throw out any cropped training or test images. We align images using the 3D center of mass, and found that further local rigid translation only reduced accuracy. Our algorithm, recognizing in parts, generalizes better than even the idealized skeleton matching until about 150k training images are reached. As noted above, our results may get even better with deeper trees, but already we robustly infer 3D body joint positions and cope naturally with cropping and translation. The speed of nearest neighbor chamfer matching is also drastically slower (2 fps) than our algorithm. While hierarchical matching [14] is faster, one would still need a massive exemplar set to achieve comparable accuracy. Comparison with [13], the authors of [13] provided their test data and results for direct comparison. Their algorithm uses body part proposals from [15] and further tracks the skeleton with kinematic and temporal information. Their data comes from a time-of-flight depth camera with very different noise characteristics to our structured light sensor. Without any changes to our training data or algorithm, Fig 15 shows considerably improved joint prediction average precision. Our approach can propose joint positions for multiple people in the image, since the per-pixel classifier generalizes well even without explicit training for this scenario.

Chapter 6: Conclusion

6.1 Conclusion:

In this work we introduce a virtual dressing room application which only requires having a front image for each product to superimpose it onto the user and the 2D graphics of the product seem to be relatively satisfactory and practical for many uses. We presented the methodology that we use to align the models with the user and we tested our procedure under different conditions. The experiments have resulted with acceptable performance rates for regular postures. After an introduction, the related work was presented real time tracking technologies up to an overview of comparable virtual dress-up system. Subsequently a closer look on the technologies and frameworks that were used for the implementation of the Virtual Dressing Room was taken. After this the different aspects of the design process up to the construction of the garment models were highlighted. This is followed by the implementation, describing the cloth colliders and the behavior of the garment, for instance. In the last section the tests were executed, also discussing the output, the appearance and the interaction with the virtual dressing room.

Overall, the presented Virtual Dressing Room seems to be a good solution for a quick, easy and accurate dress up of garment. The Microsoft Kinect offers the optimal technology for a successful implementation. Compared to other technologies like augmented reality markers or real time motion capturing techniques no expensive configurations and time-consuming buildups are required. From this point of view it is an optimal addition for a cloth store. Beyond that a simple setup of the system can also be assembled at home since the minimum requirements are a computer with a screen and a Kinect. This can also result in an additional feature for a web shop, for instance. It would allow a virtual dress up of clothes before people are buying it online, taking a closer look at the garment and even conveying the actual behavior of the real cloth. This demonstrates a huge advantage over the common web shopping experience.

6.2 Future Work

- There are many possible implementations regarding the model used for fitting. It is possible to apply a homographic transformation to the images rather than the simple scale-rotate technique in order to match multiple joints altogether although it would require more computation.
- Another alternative could be using many pictures at different angles so that it would be possible to create more realistic video streams. One could achieve a similar effect using 3D models and rendering them according to the current angle and positions. Second approach would also make it possible to implement a physics engine to go along with the model.
- Regarding future work, the Virtual Dressing Room can be enhanced in some points. Primarily taking a look at the RGB camera of the Kinect, an improvement concerning its resolution can be meaningful. The current resolution of 640x480 is quite low which results in a bad quality of the recorded scene. Since we are not really dependent on the camera of the Kinect, an additional external camera could be added that captures the scene. Of course, this would need a careful calibration, also implicating the position of the camera regarding the Kinect's depth sensor.
- Another improvement could be the implementation of a complete scanning procedure of the body that is executed prior to the tracking process. This would result in a more precise adaption of the cloth and can further include the depth/side measurements of a body.

Reference:

[1] K. Kjærside, K.J. Kortbek, H. Hedegaard, ”ARDressCode: Augmented Dressing Room with Tag-based Motion Tracking and Real-Time Clothes Simulation,” Proceedings of the Central European Multimedia and Virtual Reality Conference, 2005

[2] K. Onishi, T. Takiguchi, and Y. Ariki, ”3D Human Posture Estimation using the HOG Features from Monocular Image,” 19th International Conference on Pattern Recognition, 2008.

[3] OpenNI. http://www.openni.org/

[4] A. Vijayaraghavan, I. T.A, J. Chopra A.R.N, K. Miracline R, “A REAL TIME VIRTUAL DRESSING ROOM APPLICATION USING OPENCV” ANNA UNIVERSITY: CHENNAI 600 025 APRIL 2014

[5] N. Dalal, B. Triggs. “Histograms of Oriented Gradients for Human Detection” European Union Research projects ACEMEDIA and PASCAL.

[6] D. Chai, and K. N. Ngan, Face Segmentation using Skin-Color Map in Videophone Applications, IEEE Transactions on Circuits and Systems for Video Technology, vol. 9, no. 4, June 1999.

[7] H. Gandhi, M. Jain, A. Butalia. “Tailoring Measurement and Virtual Try-on”, International Journal of Advance Research in, Volume 1, Issue 5, October 2013. ISSN: 2321-7782 (Online).

[8] Philipp Presle “A Virtual Dressing Room based on Depth Data,” Vienna University of Technology, Klosterneuburg.

[9] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, ”Real-Time Human Pose Recognition in Parts from Single Depth Images,” Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2011.

[10] Nadia Magnenat-Thalmann, H. Seo, F. Cordier,"Automatic modeling of virtual humans and body clothing", Proc. 3-D Digital imaging and modeling, IEEE Computer Society Press 2003

[11] F. Isıkdŏgan and G. Kara. “A Real Time Virtual Dressing Room Application using Kinect”,

CMPE537 COMPUTER VISION COURSE PROJECT, JANUARY 2012.

[12] Y. In Yeo, “Magic Mirror: A Virtual Dressing Room”.

[13] V. Ganapathi, C. Plagemann, D. Koller, and S. Thrun. “Real time motion capture using a single time-of-flight camera”. In *Proc. CVPR*, 2010.

[14] D. Gavrila. “Pedestrian detection from a moving vehicle”. In *Proc. ECCV*, June 2000.

[15] C. Plagemann, V. Ganapathi, D. Koller, and S. Thrun. “Real-time identification and localization of body parts from depth images”. In *Proc. ICRA*, 2010.

[16] Kinect Quick Start Guide, http://support.xbox.com/en-GB/xbox-360/manuals-specs/manual-specs

[17] P. Ian Wilson and Dr. J. Fernandez “Facial feature detection using Haar classifiers,” JCSC 21, 4 (April 2006)

[19] Md. Z. Alom, F. Bhuiyan and H. Jong lee, “Implementation of Real Time Dress-up System based on Image Blending”, International Journal of Computer Applications (0975 – 8887) Volume 75– No.1, August 2013

[20] Kinect plugin. http://forum.unity3d.com/threads/67982-kinect-plugin/page10. Accessed: 2012-20-05

[21] Caecilia Charbonnier and Clementine Lo. Virtual mirror: A real-time motion capture application for virtual-try-on. Master’s thesis, MIRALab - University of Geneva, 2006.