Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

School of Computer Science and Statistics

Clothing Simulation Database

For Training Machine Learning Cloth Deformation Models

Cormac Madden

April 2022

A Report submitted in partial fulfilment

of the requirements for the degree of

BA(Mod) Computer Science & Business

Abstract

A key component of successful animation of the human form is realistic cloth simulation which can be achieved by increasingly diverse approaches. This project focussed on the creation of a dataset for training cloth prediction models, which in turn provides a useful resource for developers and researchers involved in the animation sector. A comprehensive literature survey exploring the different aspects of a cloth simulation workflow highlighted the benefits and drawbacks of a range of approaches to accomplish each stage. The open source Skinned Multi-Person Linear (SMPL) model was used to create a realistic and highly adaptable digital human form. Scripts were written using Blender Python API to generate the models and skeletal animations were sourced from Mixamo. Retargeting of the skeletal animations was achieved using a combination of the Rokoko plugin for Blender and a script written to select and assign the bones to each skeleton. Scripts were also created to automate the export of the mesh and skeletal animation from Blender to Houdini. In Houdini, scripts were written to iterate over each animation and perform the cloth simulation. A Houdini Digital Asset was then created with the nodes required for simulating the draping of cloth over a human mesh. The garment was generated using a series of nodes, and simulated using the Vellum Drape and Vellum Solver nodes. The final output is a simulation of a garment on a person moving through the animation. Scripts were written to export the geometry from Houdini using a custom data format. The exported array is accompanied with documentation providing an explanation of the array structure. The significance of this project lies in the creation of a resource for researchers and developers involved in cloth simulation and demonstrates the effectiveness of using scripts to adapt and automate the process.

Acknowledgements

Firstly, I would like to extend a very sincere thank you to my supervisor Dr. Rachel McDonnell for her constant support and guidance throughout this project. Her knowledge, interest and passion in the subject were inspiring and I have really enjoyed working on this project with her. In addition to fulfilling a personal goal of working in the area of computer graphics, the project has given me an appreciation of problem solving and learning through a range of resources, and I am particularly grateful to Dr. McDonnell for sharing her insights and expertise. Throughout all my academic endeavours, my family have always been there for me with their support and constant encouragement, and so I also extend my appreciation and thanks to them.

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Context and research motivation

Creating realistic digital humans is a highly creative and labour intensive process that has been evolving rapidly over the past decade. The applications for digital humans have also grown from their widespread use in the entertainment industry, including video gaming and film production, to their increasing application in the on-line fashion world with virtual-try-on apps. A key component of the process of creating a digital human in most applications involves the animation of clothing that when of high quality, has the potential to transform the model into a very realistic representation of the moving human body. There are several challenges to creating high quality virtual clothing, not least of which is the complex nature of cloth dynamics, and in some applications the need to accommodate a wide range of movements and different body shapes and sizes. Finally, the animation of clothing needs to be both cost and time effective for scalable projects involving crowds and real-time applications such as try-on apps.

The traditional and most common approach to animating cloth is to use what is known as **Physics-Based Simulations (PBS)**. It produces realistic animations, however it requires manually editing the garment shape in 2D, placing it on the digital character and adjusting parameters to achieve the desired results. The system then simulates many factors such as gravity, stretching, stiffness, collision etc. on millions of triangles, which is computationally very expensive. It is therefore not scalable and limited to high budget applications and in situations where fast turnaround is not critical to the project.

Data-Driven Models on the other hand, learn from training data, and given limited inputs can predict various aspects of cloth simulation. This approach to cloth animation is significantly faster than PBS, and in many cases thousands of times faster. For applications that require real-time cloth animations such as video games or ‘virtual-try-on’ apps, performing an accurate PBS is often too computationally expensive and too time-consuming. This is where predicting cloth animation from a pre-computed data model can be highly beneficial.

Research motivation

Given the potential benefits of the data driven models over physics-based simulation, the development of datasets to train these models is of critical importance to the ongoing evolution of cloth simulation and the delivery of realistic simulations. However, there is limited accessibility to publicly available datasets which makes creating a cloth prediction data model very challenging. Although there are many publications in the computer graphics literature on the different approaches to creating a cloth prediction data model, many do not publish the dataset they used to create and train the model. The purpose of this project is to create a publicly available dataset that can be used to train and potentially benchmark various machine learning data models.

1.2 Project goal and objectives

The overall **goal** of this project is to develop a dataset for training and benchmarking new and existing clothing deformation models, thus providing a useful resource for developers and researchers across a range of industry and academic sectors.

To achieve this goal, three **objectives** were specified as follows:

1. To **explore** the existing state-of-the-art approaches to cloth simulation with a focus on prediction data models.
2. To **create a dataset** for training clothing prediction data models that is comprehensive, realistic and inclusive for all body types.
3. To **write scripts** to automate the process of generating additional data.

1.3 Overview of Project Report

This report comprises four chapters. Having introduced the context and motivation for this research project and stated the overall goal and specific objectives here in Chapter 1, a comprehensive literature review outlining the development of human simulation in computer graphics and an exploration of the use of both physics-based models and machine learning datasets in clothing simulation is presented in Chapter 2. The details of the project implementation and outcomes including the final dataset and licensing considerations are described in Chapter 3. The final chapter 4, comprises an evaluation of project implementation, including the project limitations and the potential for future work. An overview of approaches to cloth simulation is presented in Figure 1. The numbers in the red dots represent the corresponding sections in the literature review and analysis chapter.

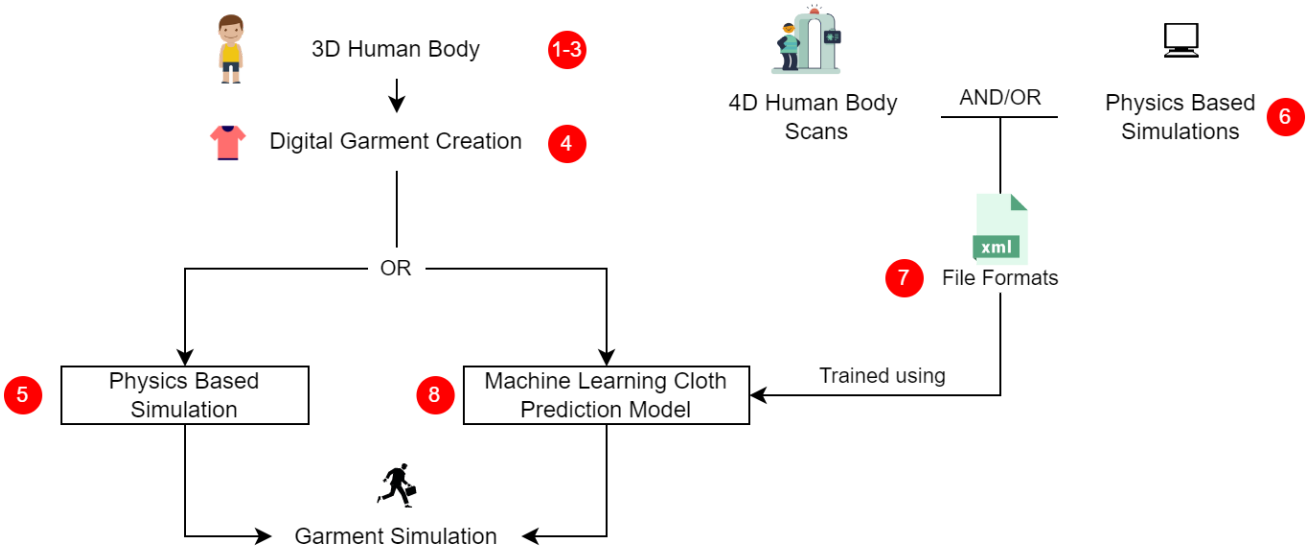

Figure 1. Overview of approaches to cloth simulation with relevant sections of the report from 2.1 to 2.8 highlighted numerically in red dots.

2 Literature review and analysis of the cloth animation process

This review outlines the evolution of human body representation in computer graphics and provides a comparison of virtual garment design methods. Techniques used to simulate garments in computer graphics are reviewed followed by an analysis of the software packages with which simulations can be performed. The published machine learning cloth prediction models and the manner in which they are trained are also compared. From this literature review process and analysis of the published data, conclusions are drawn that inform the implementation of this project.

2.1 The evolution of generative human models

When trying to predict the clothing deformations, it is important to consider the human they fall on. The shape, pose, and motion of the human will all impact on how the cloth should deform. To train a machine learning model, a wide variation of 3D humans are needed.
It would not be feasible to 3D scan a sufficient number of differently shaped humans, in enough poses for use in cloth model training. 3D modelling of humans manually would also be too time-consuming, and potentially inconsistent. The goal is to be able to generate any shaped human and pose, and to animate them in an infinite number of ways.

Early research regarding **generative human models** was closely interlinked with computer vision and human tracking. Researchers wanted to create a human model that could be generated using images and videos of humans. The first generative approach to creating a human model was introduced by Marr & Nishihara in 1978 as illustrated in Figure 2. They were the first to represent the body as a kinematic tree of primitives and joints. Each joint has a relative offset position and a rotation.

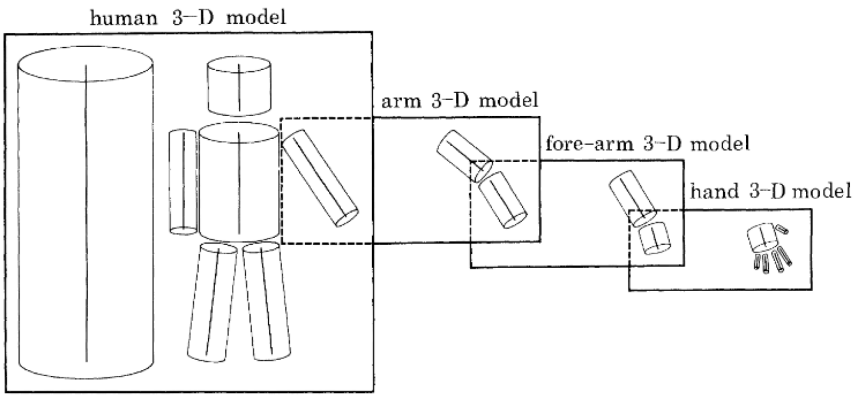

Figure 2. Representation of a Human as a Kinematic Tree (Marr and Nishihara, 1978)

The first case of representing the body using machine learning was in 2D. Models were created to determine the shape of a body from computer vision (Figure 3).

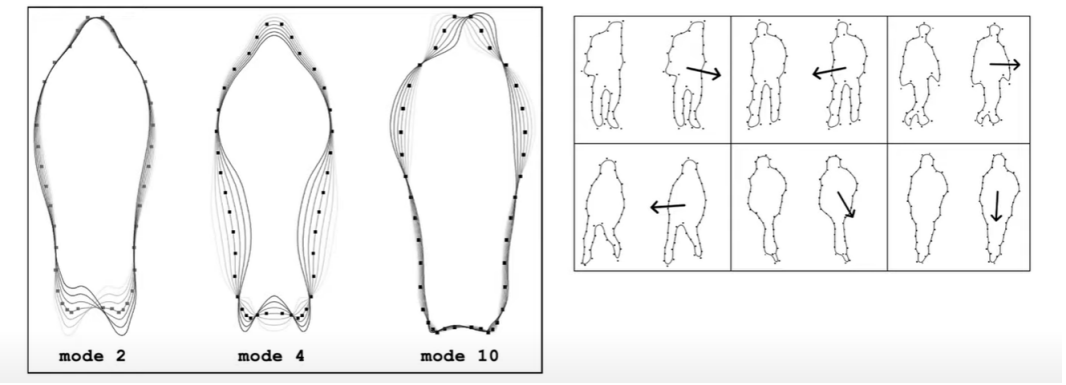

Figure 3. Machine-learned human model created to detect humans in video. (Baumberg and Hogg, 1994)

The first data-trained human 3D models were of faces in 1999 (Blanz and Vetter, 1999). A face model was developed first largely because they were easier to 3D scan (Figure 4), and are also a lot less complex than a full human body. This model was created using **Principal Component Analysis** (PCA) across a large dataset of facial scans. PCA is a popular dimensionality reduction technique used in statistics and Machine Learning applications. PCA condenses information from a large set of variables into fewer variables by eliminating some variables and creating correlations between others.

Figure 4. First data-trained face model and Cyberware Face scanner (Source: Cyberware)

2.1.1 CAESAR dataset

The next significant development in the field of human generative models was the CAESAR (*CAESAR Dataset*, no date) dataset created in 1999. This dataset by the US army comprises 4,000 full human body scans using the Cyberware scanner (Figure 5). They conducted these scans on a sample portion of the US population based on 1990 census data. This became known as the **CAESAR dataset**, and is still one of the few publicly available large datasets of 3D scanned humans.

Figure 5. Cyberware full body human scanner (Source: Cyberware)

2.1.2 The Allen model

In 2003, Allen et al., used the CAESAR dataset to create the first human generative model. They created a template mesh and registered the template mesh to several of the scans, creating a common topology or common frame of reference for each scan. This allowed them to use principal component analysis and build a shape space. With the shape space they could interpolate between each of the source meshes generating new human bodies. They also rigged these models with a skeleton so they could be reanimated but the bodies lacked realism when reposed, due to a lack of pose-dependent deformations (Figure 6). This concept is further elaborated upon in section 2.3.1 below in the context of the SMPL model. (Allen, Curless and Popović, 2003)

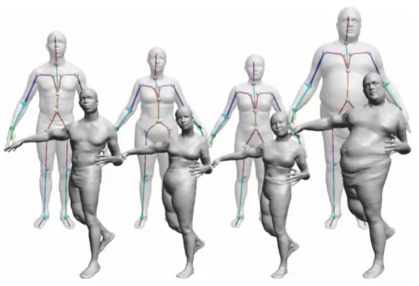

Figure 6. Example of strange deformations when posing different body shapes using the Allen model. (Black, Mahmood and Bolkart, 2021)

2.1.3 SCAPE 2005

The SCAPE (Shape Completion and Animation of People) model was a groundbreaking paper released in 2005. It was the first paper to perform retargeting of animated poses from one 3D scanned person on the static scans of several other people. SCAPE also made breakthroughs with regard to “Shape Completion”. Shape completion is where only part of the 3D person is scanned and the shape completion model predicts the geometry of the missing mesh (Figure 7). This paper was inspirational but its retargeting of body shape and pose had many issues. There was no underlying skeleton and so pose shapes were too closely interlinked with the shape of the body, often leading to oddly deformed meshes when the new body was given a new complex pose (Anguelov *et al.*, 2005). The limitations of these early models have been largely overcome in the following decade with the emergence of several newer approaches including the skinned multi person linear (SMPL) model.

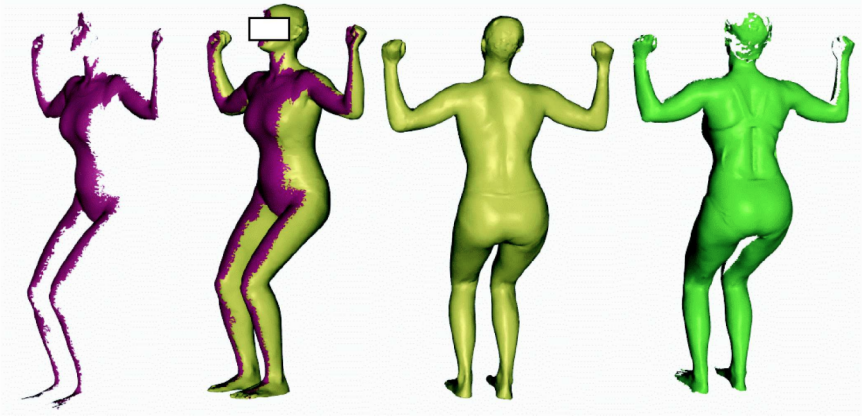

Figure 7. Example of Shape Completion using SCAPE (Anguelov et al., 2005)

2.2 Skinned Multi-Person Linear (SMPL) Model

The SMPL model published in 2015, was designed for human body capture from computer vision and was also trained on the CAESAR dataset. The SMPL model uses a vertex displacements on a template human mesh to create the final output mesh (Figure 8). The template or base mesh was created by a 3D artist and is the same mesh for male and female models. It is a vertex-based model that can accurately represent a wide variety of body shapes in natural human poses. The SMPL model has about 7,000 vertices and is segmented into parts that can rotate in a kinematic tree. (Loper *et al.*, 2015)

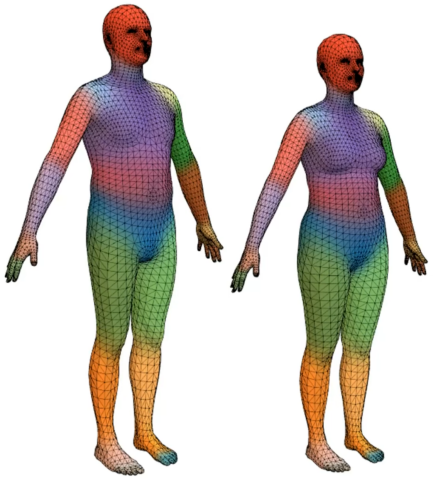

Figure 8. Template SMPL Mesh in A-Pose (Black, Mahmood and Bolkart, 2021)

2.2.1 SMPL skinning

SMPL is a “skinned” model which means that when it changes from one pose to another the skin moves as it would do on a regular human body. Creating realistic skinning is a common obstacle in computer graphics, and there are several approaches to tackling it.

Linear Blend Skinning is the most common form of skinning that is used to pose a model. The vertices in the final posed mesh are determined as a linear combination of transformations from potentially all the joints in the body but controlled by the blend weights. However, it has its well-known limitations such as volume collapsing and the candy wrapper effect as depicted in Figure 9.

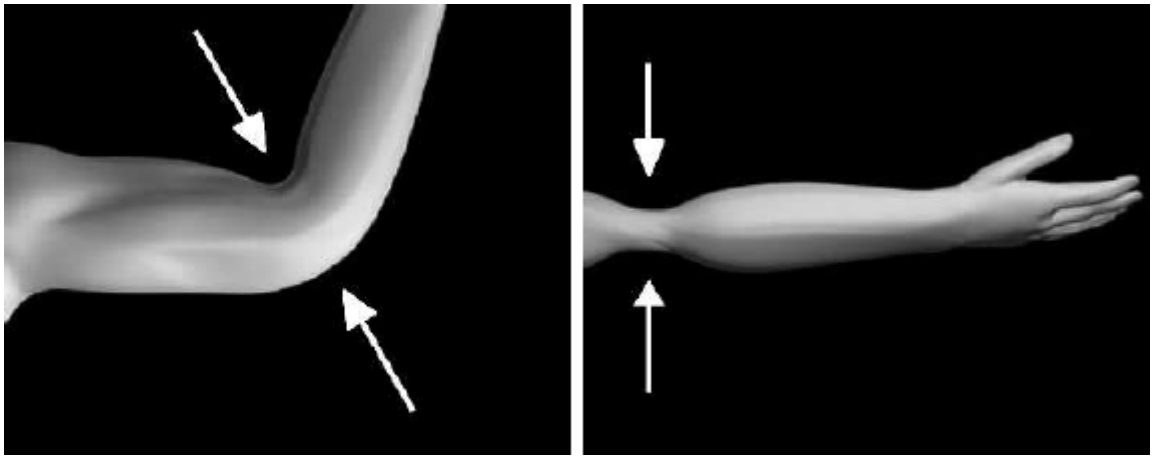

Figure 9. Example of the Collapse and Candy Wrapper effect associated with Linear Blend Skinning (Vasilakis and Fudos, 2011)

Other techniques such as **Dual Quaternion Blend Skinning** fix some of these issues but introduce new bulging issues at joints. The SMPL model learns blend shapes to correct for the limitations of standard skinning as outlined below.

2.2.2 SMPL Linear Blend Shapes

Linear Blend Shapes are the vertex displacements applied to the template mesh in the rest pose. They are determined by the principal components of the shape variation learned from performing PCA on the CAESAR dataset. So there is no specific definition of which body part each of the shape parameters changes. Here is a sample of the first few principled components (PC) illustrated in Figure 10:

PC 1: height / overall scale of the model

PC 2: weight

PC 3: torso height + shoulder width

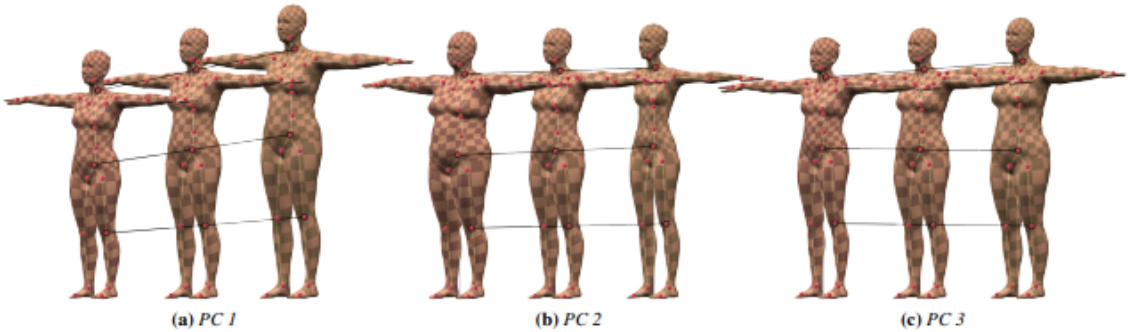

Figure 10. First Three Principal Components of the SMPL model (Loper et al., 2015)

2.2.3 Multiplicative vs Additive models

Unlike SCAPE (section 2.1.3), SMPL is an additive “factored model”. While SCAPE also factors deformations into shape and pose, SCAPE is a multiplicative model that multiplies the triangle deformations. With SCAPE a bigger person will have bigger pose-dependent deformations even though these deformations are not learned for different body shapes.

With additive models like SMPL each factor works independently and can be layered on top of one another as blend shapes. This allows various features such as breathing, dynamics and clothing to be layered on top of the model.

One of the more impressive blend shapes added to the SMPL model are the dynamic blend shapes that emulate soft-tissue dynamics from the body moving. It is learned by adapting techniques used in the Dyna model (Pons-Moll *et al.*, 2015). Dynamic-SMPL, or DMPL model, is trained from the same dataset of video scans at 60fps as the Dyna model. PCA is used to reduce the dimensionality, producing smooth transitions to each blend shape.

The different blend shapes for identity, pose, and soft-tissue dynamics are additively combined with a rest template before being transformed by blend skinning as illustrated in the formula below.

$$T(\theta, \beta) = T_{\mu} + B_s(\beta) + B_p(\theta)$$

T = Template Mesh

θ = Pose Parameters

β = Shape Parameters

B = Blend Shape Function

2.2.4 Building upon SMPL

Since it was first published, SMPL has been built upon several times adding new parameterized features such as facial expressions and complex hand gestures. In 2017, MANO was developed to map accurate hand poses onto the SMPL rig which became SMLP-H. MANO was learned from only 1,000 hand scans of 31 subjects (Romero, Tzionas and Black, 2017). In the same year FLAME, the face model was also released. It is learned from 33,000 facial scans including 3,800 from Caesar (Li *et al.*, 2017). The hand and face models were later combined to make SMPL-X in 2019 (Pavlakos *et al.*, 2019).

2.3 Other Human Models

Several other human models exist for the representation of humans in computer graphics although many lack the diverse and the realistic shape space of the SMPL model. A recent entrant into this space is GHUM and GHUML, a model developed by Google in 2020. GHUM consists of 10,168 vertices and the low-resolution GHUML(ite) of 3,194 vertices. It is learned using a VAE (Variational Auto-Encoder) machine learning model unlike SMPL which uses PCA. VAE differs from PCA in that PCA is essentially a linear transformation but auto-encoders are capable of modelling complex nonlinear functions. There are advantages and disadvantages to both approaches. Auto-Encoders enable GHUM to train on the full human body and learn all components such as dynamics and pose correctives in one consistent learning loop. PCA however is typically faster and computationally cheaper than autoencoders. GHUM also needs to be trained on a much bigger dataset of 4.6 thousand subjects across 60 thousand scans. GHUM's model is also not natively compatible with standard graphics pipelines unlike SMPL (Xu *et al.*, 2020). Other generative 3D human models include Metahuman, MakeHuman and RocketBox, each of which have their merits and drawbacks but are beyond the scope of this review.

2.4 Designing Garments in Computer Graphics

Prior to exploring garment simulation on the digital human, a brief review of the techniques used to generate garments is presented in this section.

2.4.1 Traditional Methods

Various 3D garment modelling methods have been developed since the 1980s, most of which are based on the traditional real-world techniques of going from 3D to 2D and back to 3D again. The process starts with a 3D conceptual idea in the designer's mind, from which 2D fabric patterns are generated by pattern design tools (3D–2D stage) such as OptiTex (Figure 11). These patterns are then assembled and sewn together to form clothing on a 3D human model (2D–3D stage). In such a process, expert knowledge on 2D pattern generation is required. Slight variations of this approach are typically how garments are generated in software packages such as Marvelous Designer and Houdini.

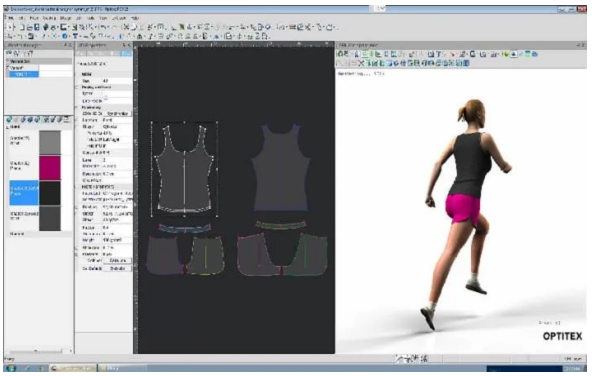

Figure 11. 2D Pattern design in OptiTex (OptiTex, no date)

2.4.2 Automatic Garment Generation

In the last decade researchers have developed new techniques for generating garments that require less human involvement. Machine learning models such as **Sketch2Mesh (Guillard *et al.*, 2021)** or the later publication (T. Y. Wang *et al.*, 2018) which proposes new garment generation techniques that takes a sketch and a human model as input, and outputs a 3D garment draped over the model. The most impressive aspect of this technique is how fold lines and creases drawn in the sketch will be represented in the garment (B. Wang *et al.*, 2018).

Figure 12. Example of generating garments from a sketch using the (T. Y. Wang et al., 2018) model

LiveCap (Habermann *et al.*, 2019), Tex2Shape (Alldieck, Pons-Moll, *et al.*, 2019), Octopus (Alldieck, Magnor, *et al.*, 2019), Pifu (Saito *et al.*, 2019), DeepCap (Habermann *et al.*, 2020) and ARCH use computer vision-based methods to extract the clothing of a person in a 3D scan. They first calculate the pose and shape of a SMPL model that would fit the scanned person. The difference between the vertices of the SMPL model and the original scanned mesh is then calculated, and from that the type of clothing is inferred. The difference between the SMPL model and the scan is used as a displacement added to the SMPL formula as shown below. This garment can then subsequently be used with SMPL models of different shapes and in different poses due to the additive nature of SMPL's model.

$$T(\theta, \beta) = T_{\mu} + B_s(\beta) + B_p(\theta) \textbf{ + D}$$

D = Clothing

The drawback to these techniques is that they do not separate the body from clothing, and you are limited to the typology of the SMPL model. A high resolution can be obtained from subdividing the SMPL mesh but the model still struggles with loosely fitted clothing like dresses or skirts.

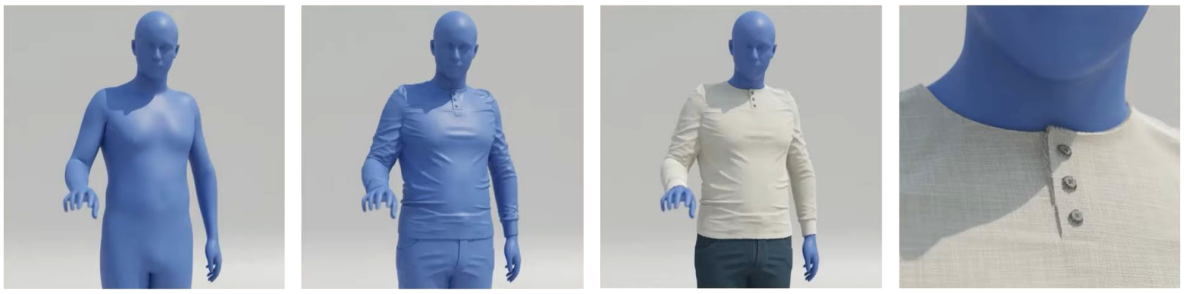

Figure 13. Clothes deformation on SMPL model using Tex2Shape (Microsoft, no date)

Multi-Garment Net (Bhatnagar *et al.*, 2019), **ClothCap (Pons-Moll *et al.*, 2017), BCNet (Jiang *et al.*, 2020)**, **SizerNet (Tiwari *et al.*, 2020), and DeepWrinkles (Lähner, Cremers and Tung, 2018)** use a similar computer vision-based approach however they reconstruct clothing as a layer separated from the body.

These methods all represent the garments as a mesh. A problem arises when you want the cloth prediction model to work for a new item of clothing that is a totally different shape and would require a new mesh, with new typology. This leads us to the speculative question about the future of cloth prediction models:

"Which is a more promising way to model garments, creating a general model for all garments or creating a separate model for each type of garment?"

A single mesh cannot represent all types of garments. If the machine learning model is based on a mesh representation, there needs to be a newly trained model for each garment. If the garment is represented by an implicit function it can be manipulated to take the shape of different garments using only one cloth prediction model. **SMPLicit** is an example of this (Figure 14). It enables the interpolation between different garments using a single clothing model, but it still has many drawbacks including difficulty in applying textures (Corona *et al.*, 2021).

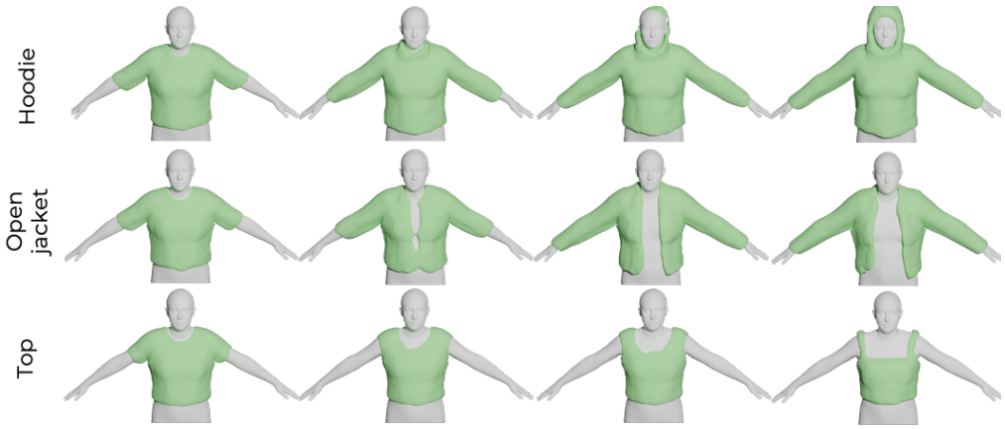

Figure 14. SMPLicit Garment Modelling of a top (Corona et al., 2021)

It is clear that the technology is evolving towards automatic garment generation which on a large scale is more cost effective and time efficient. The future is also likely to see implicit modelling techniques come to the fore of garment generation. In order to create implicit models, vast amounts of training data is required which can be generated by 3D scanning of cloth or by using physics-based simulations. As the more cost-effective and accessible approach and the one used to generate training data in this project, the section below describes the techniques used in physics-based simulations.

2.5 Physics-Based Cloth Simulations (PBS)

Despite advancements in 3D scanning, PBS is still the most practical solution for creating training data. It is an approach that is cheap, fast, versatile and infinitely scalable. In addition, it has the capacity to create simulations that are impressively realistic. Geometric and physical approaches to PBS are compared below.

2.5.1 Geometrical techniques

With geometrical techniques folds and creases were represented using geometrical equations. This resulted in very realistic looking cloth but was not very physically accurate.

The first technique used to simulate cloth was by Weil et. al. in 1986 and was based on the idea that hanging cloth usually followed catenary curves y = a*cosh(x/a), as shown in Figure 15. However, this implementation was limited to hanging cloth (Weil, 1986).

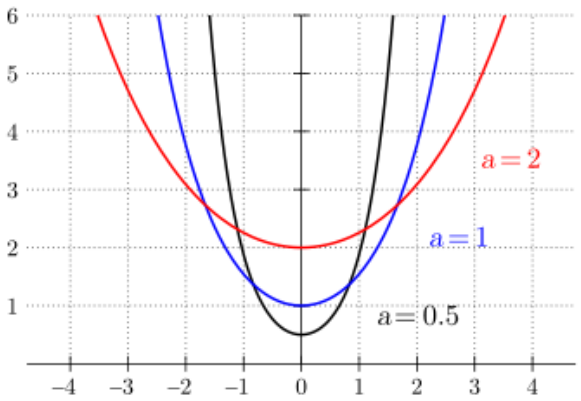

*Figure 15. Catenary curves following the equation y=a*cosh(x/a)*

The next technique proposed by (T and Y, 1990) was to represent a cloth sleeve represented as a hollow cylinder. The cylinder could bend by manipulating the length of the horizontal segments (Figure 16). The corners then needed to be smoothed by subdividing the number of faces (Ng, Grimsdale and Allen, 1995).

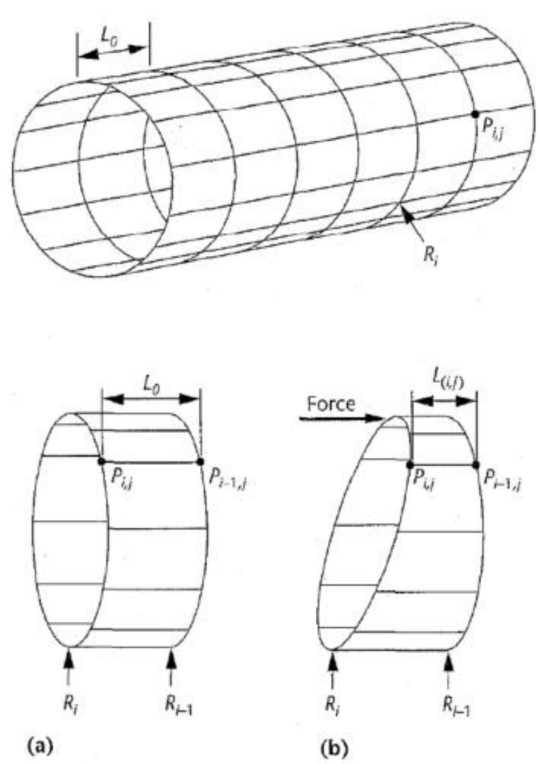

Figure 16. a) Cloth sleeve represented as a hollow cylinder, b) A section before and after bending (T and Y, 1990)

This technique was further developed by who proposed that the cloth should follow a sinusoidal function in areas where the cloth is slack. Despite the elegance and efficiency of these functions, they were not capable of performing dynamic collisions with other objects (Ng, Grimsdale and Allen, 1995).

2.5.2 Physical techniques

The most common technique used for realistic cloth simulations in computer graphics uses an energy model, where the energy of each vertex on a mesh is calculated based on a set of internal and external forces.

In 1995, Provot proposed the mass-spring model to simulate cloth as illustrated in Figure 17 (Provot, 1995). The model connected nearby vertices using springs and calculated internal forces such as stretch and shearing using Hooke's Law. Newton's equations of motion were used to calculate external forces such as gravity, air-drag and collisions. The energy at each point is typically calculated in relation to its 12 surrounding points.

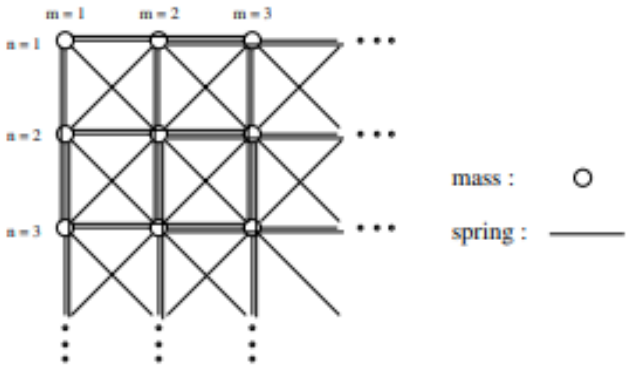

Figure 17. Structure of Mass-Spring Model (Provot, 1995)

One of the issues with the mass-spring method when it was first proposed was the time it took to calculate collisions. The mass-spring method calculated whether or not it had collided with another object for every step in time. In most applications this was a time step size of 0.02 seconds. Baraff and Witikin in 1998 proposed a solution with adaptive time stepping. In this case, if the simulator finds a collision at the larger step size it reduces the size until it finds the exact point of collision. The simulator then progressively tries to increase the step size again (Baraff and Witkin, 1998).

2.6 An Analysis of 3D Software Packages Used to Create a ML Dataset

The two most important requirements for generating the cloth simulations for this project are that the simulations must be as physically accurate as possible, and that the process of generating the cloth can be automated. With that in mind, several different software packages were considered and evaluated. Quantitatively determining the most physically accurate solution was not feasible given the variety of variables in each package, however a qualitative assessment was made of each package, comparing the features and drawbacks of each.

2.6.1 Marvelous Designer

Marvelous Designer is the industry standard for cloth simulations and is the most commonly used tool for creating realistic cloth simulation in movies and for 3D artwork. It has an impressive garment design tool-set, however it is a tightly controlled piece of software and doesn't facilitate scripting or automated workflows. This makes generating hundreds of simulations difficult and time-consuming. Marvellous Designer was used as the package of choice for the dataset used by TailorNet, (a cloth prediction model that is discussed below in section 2.8.6) but in their repository they commented on the painstaking hours that were spent manually inputting and simulating each garment for every pose, body shape and garment style.

2.6.2 Blender

Having prior experience with the open source Blender and its SMPL add-on made it an attractive choice to use for creating the dataset. A key feature of the software is that it has an inbuilt script editor and python console. However, its clothing simulation features were not as comprehensive and mature as the other packages. It struggled with stitching seams together and doesn't have the same draping features that Houdini, Marvelous and Optitex have.

2.6.3 Houdini

Houdini is a parametric based software developed by the Canadian company SideFX. It employs a node-based workflow so multiple iterations can be explored as a project is refined and different effects trialled. It also facilitates writing scripts that can be used to adjust the position of the garment depending on the height of the character and scripts for exporting the geometry in a custom array format. Additionally, there are excellent learning resources available online that demonstrate how to use the nodes required for cloth simulations and the fundamentals of script writing in Houdini.

SyFlex, NvCloth and Optitex were also considered and were used to create the training data in existing cloth prediction models which will be discussed in the next section.

2.6.4 Features comparison

The features of the six considered software packages are shown in Table 1 below.

Software	Scripting Toolset	Good Stitching	Pre-Animation Draping	Easy-to-use
Blender	✓	✗	✗	✓
Marvelous Designer	✗	✓	✓	✓
SyFlex	✓	✓	✗	✓
NvCloth	✓	✓	✗ *	✗
OptiTex	✗	✓	✓	✓
Houdini	✓	✓	✓	✓

Table 1. Comparison of Software Simulation Packages

Having considered the merits and drawbacks of these six software packages, Houdini was chosen as the cloth simulation software package for generating the training data for a machine learning model in this project because of its overall feature set and in particular the high quality simulations it generates and the tool set for automation. Once the data is generated it needs to be saved in a format that can be used to train a machine learning model. There are several format options for these files which are reviewed in the following section.

2.7 File Format Options for Machine Learning

A key objective of this project is to create the output in a format that is useful for training a ML model. When using ML, the file size, speed at which it can be read; and how well the computer can interpret the information is very important. Almost all 3D models can be divided into two categories: surface or solid type models, which are explained below. Each of these representations can then be stored in a variety of file formats.

1. **Surface 3D Models**: Boundary Representations/B-Reps (Meshes) or Implicit/Parametric surfaces.
2. **Solid 3D Models**: Constructive Solid Geometry or Point Cloud / Voxels

2.7.1 Surface 3D Models

B-Reps (Meshes)

Meshes are a form of B-Reps, and are the most commonly used representation for 3D models. A mesh is a geometric data structure that allows the representation of surface subdivisions by a set of polygons. A mesh is made up of vertices, connected by edges making faces of a polygonal shape. Figure 18 shows the typical data structure of the mesh of a cube.

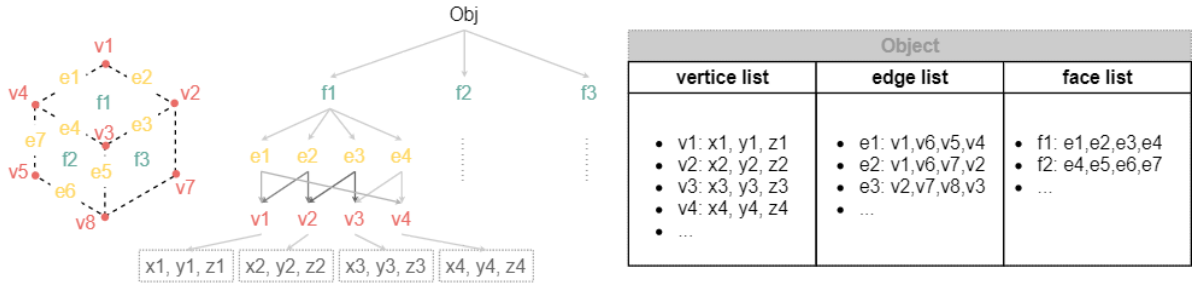

Figure 18. Typical data structure of the mesh of a cube (Poux, 2021)

Mesh representations, due to the amount of information they store, have relatively high memory requirements. Calculations and certain manipulations of the mesh can be quite computationally expensive. Convolutional Neural Networks (CNNs) struggle to interpret data such as 3D shape meshes or other graph structured data that is not represented by regular grids. A paper called FeaSTNet released in 2018 tries to address this problem by identifying features on a mesh. It is a big advancement in using meshes for machine learning but there are still limitations with regards to interpreting animated meshes.

Implicit Models

3D models can also be stored implicitly by using mathematical equations to determine the boundary of surfaces. An example of an implicit model is the SMPLicit (Corona *et al.*, 2021) garment model shown earlier. Implicit models are difficult to generate but are highly efficient for calculating operations such as geometry intersections, or the distance between geometry. Many new research papers are emerging focusing on what is called neural implicits. This is the process of using machine learning to generate implicit functions to represent surfaces, and in the case of creating a cloth deformation model it will drastically improve collision detection between the cloth and the body. (Schirmer *et al.*, 2021)

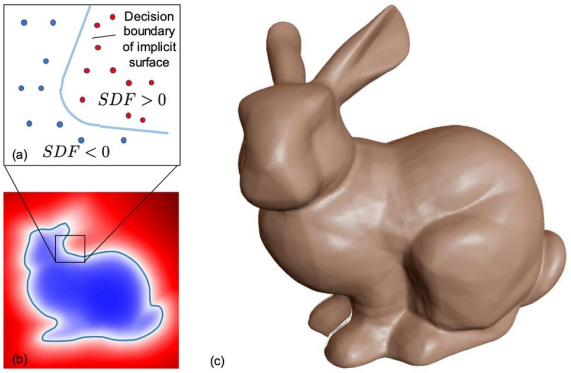

Figure 19. Example of Implicit model and calculating distances using Signed Distance Functions (Schirmer et al., 2021)

2.7.2 Solid 3D Models

Constructive solid geometry is typically created by adding and subtracting basic primitive shapes using boolean operations. However, this is not quite applicable to the clothing or human models we are trying to represent. **Point Cloud** is the simplest form of 3D model presentation. It is made up of a collection of 3D vertices, each with coordinates (x,y,z). This is a very common representation, as it is the one usually captured by 3D scanners. (Poux, 2021)

B-reps and Point cloud representations are used in this project and the justification of these choices are elaborated on in the implementation section in Chapter 3. The next section reviews the various machine learning models currently used in cloth deformation prediction.

2.8 Existing Machine Learning Cloth Prediction Models and Their Datasets

The merits and drawbacks of various ML models in addition to a review of how they were trained is presented in this section. This analysis informed the approach to project implementation so that the dataset created was compatible with existing models.

2.8.1 DRAPE 2012

DRAPE (DRessing Any PErson) was the first complete system for animating realistic clothing on synthetic bodies of any shape and pose without manual intervention. It adapts to different body shapes and poses without the redesign of clothing patterns; this effectively creates infinitely-sized clothing. This is not desirable when you would like to see the fit of a clothing of a fixed size, but the system was a big leap in the right direction (Guan *et al.*, 2012).

To train their model they used the SCAPE human model from 2005, and the OptiTex software package to generate and simulate their garments. DRAPE was trained on 24 Motion capture sequences resulting in 3500 different poses.

2.8.2 DeepWrinkles 2018

Deepwrinkles (Lähner, Cremers and Tung, 2018) is a machine-learning based cloth simulation model that combines low resolution mesh deformations with high resolution normal maps to create garments with realistic wrinkles.

A normal map is a 2D image texture that can be applied to mesh geometry to create the illusion of detail. It changes the surface slope (normal direction) of the mesh for each pixel on the texture. It is typically used to fake high-resolution details on low-resolution meshes. Normal maps are typically generated by calculating the difference between high resolution and low resolution meshes.

DeepWrinkles is trained on 4D scans of subjects in motion (3D scans, with time as the fourth dimension) which capture around 200k vertices at 60 fps. Each subject is dressed in a full-body suit with one piece of clothing that has colored boundaries.

DeepWrinkles estimates the pose and shape of a 3D scanned human using SMPL.
A low resolution normal map texture is then generated from the input pose and shape.
It also generates a high resolution normal map using the detail from the 3D human scan.

It uses the high and low resolution normal maps to train a model (eGAN, evolutionary generative adversarial networks) that can generate high resolution normal maps from low resolution ones at runtime.

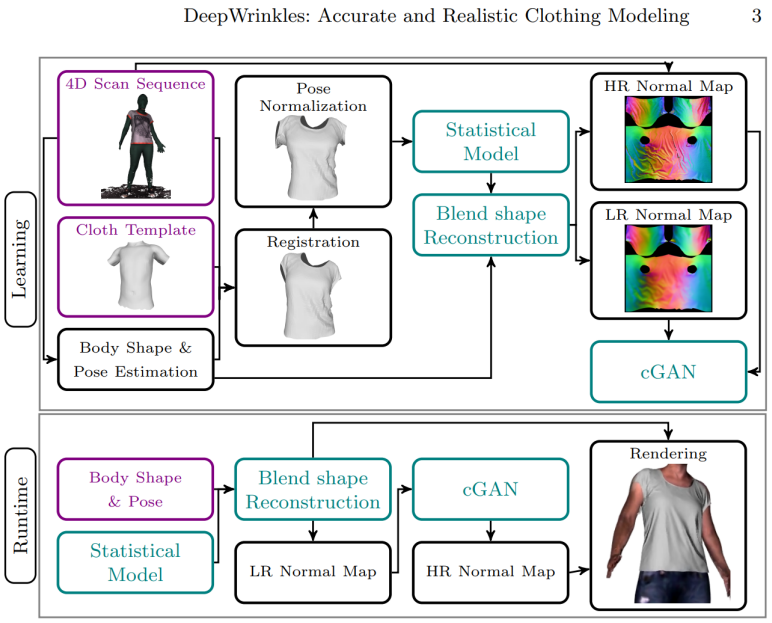

Figure 20. DeepWrinkles Architecture (Lähner, Cremers and Tung, 2018)

At runtime DeepWrinkles takes body shape and pose as input and outputs a low resolution mesh and a low resolution normal map, which is converted to high resolution and then applied to the mesh. An overview of the DeepWrinkles architect is illustrated in Figure 20.

2.8.3 SWISH 2021

SWISH (Lewin, 2021) uses a similar approach to DeepWrinkles of adding smaller details like wrinkles using normal maps. SWISH is used to generate wrinkles on the jersey's of American football players in the video game Madden NFL 21 and it was first presented at Siggraph 2021. This method is ideal for tightly fitting clothing like the jerseys in the video game.

SWISH however is trained with PBS whereas DeepWrinkles was trained with 4D scans. The SWISH model used Marvelous Designer to create its training data. SWISH used 700 simulations for each cloth asset and created a new model for each body shape. Between 6 and 48 hours per model was required to generate their cloth simulation data.

2.8.4 GarNet 2018

GarNet (Gundogdu *et al.*, 2019) is another machine learning cloth deformation model that focuses heavily on minimising intersections between the cloth and the human model. The system uses a two-stream architecture, the first stream takes, as input, the body represented by a 3D point cloud while the second, takes as input, the garment represented by a triangulated 3D mesh. When training the model, collisions are optimised using two terms. The interpenetration term penalises a garment vertex for being on the wrong side of the corresponding body point. The bending term penalises the vertices differing too far from the ground truth. GarNet's focus on avoiding collisions meant that the resultant deformations often lacked detail and appeared very smoothed out (Figure 21).

Figure 21. Example of smoothed garment of GarNet (Gundogdu et al., 2019)

GarNet trained its model using PBS in Nvidia's NvCloth. It comprises a pair of jeans, a t-shirt and a sweater worn by 600 bodies from the SMPL dataset in various poses. The training, validation and test sets are split into 500, 20 and 80 bodies, respectively. GarNet used the CMU dataset to animate their SMPL bodies into various poses.

2.8.5 Santesteban et al. 2019 Model

The model developed by Santesteban et al. is another machine learning cloth deformation model and it focuses on improving the smoothed out issues faced by GarNet. Their model separates global garment fit due to body shape, from local garment wrinkles due to both pose dynamics and body shape. This means it calculates high frequency deformations and low frequency deformations separately and on different areas of the cloth depending on where necessary (Figure 22). The result is a realistic looking and efficient cloth mesh (Santesteban, Otaduy and Casas, 2019).

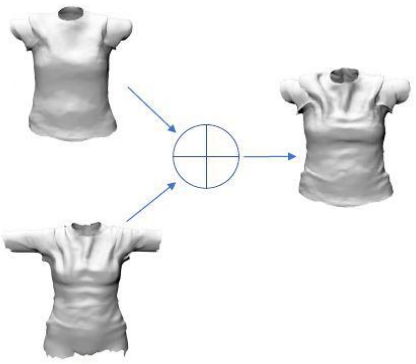

Figure 22. Combination of high and low frequency details (Santesteban, Otaduy and Casas, 2019)

2.8.6 TailorNet 2020

TailorNet (Patel, Liao and Pons-Moll, 2020) is the first first data-driven clothing model to separately account for pose, shape **and garment style**. It takes the same approach to wrinkles as that proposed by Santestban et al., and the resultant mesh looks very realistic (Figure 23). However it also has its limitations. It takes a submesh from the template SMPL model to create its garments and as with most of the other models it is limited to one surface.

TailorNet is trained using physics-based simulations in Marvelous Designer and uses the Multi-Garment Net dataset for its garments.

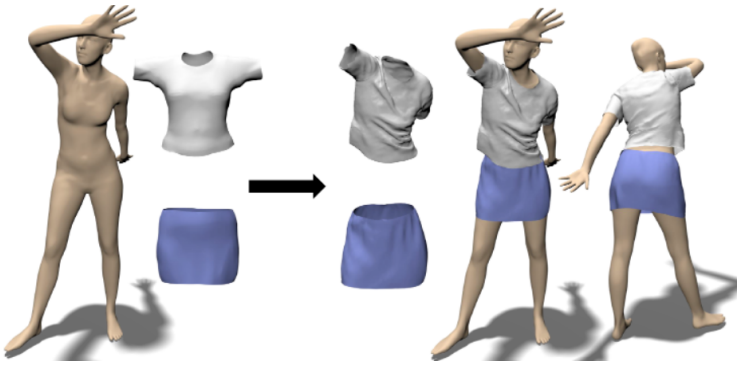

Figure 23. Example of TailorNet (Patel, Liao and Pons-Moll, 2020)

2.8.7 Comparative analysis of existing ML cloth prediction models

Prediction Models	Static/Dynamic	Trained using 3D Scans	Pose Variation	Shape Variation	Style Variation	Model Public	Dataset Public
Santestban et al.	**Dynamic**	✗	✓	✓	✗	✗	✗
Wang et al.	**Static**	✗	✗	✓	✓	✓	✓
DeepWrinkles	**Dynamic**	✓	✓	✓	✗	✗	✗
DRAPE	**Dynamic**	✗	✓	✓	✗	✗	✗
GarNet	**Static**	✗	✓	✓	✗	✗	✗
TailorNet	**Static**	✗	✓	✓	✓	✓	✓

Table 2. Comparison of Cloth Prediction Models

The comparative analysis of six ML cloth prediction models presented in Table 2 highlights the variety of approaches to the training of ML models. Consideration of these differences

should be given in the creation of the dataset for a particular prediction model to ensure compatibility.

2.9 Summary of literature review

The significance of cloth simulation in computer animation is evidenced by the growth in publications in this area and the data continually emerging tracking the evolution of techniques towards high quality simulations that are also cost effective and scalable. An analysis of the published data in relation to human generative models demonstrates that the SMPL model overcomes many of the issues associated with earlier models and provides diverse and dynamic blend shapes with smooth transitions in addition to other parameterized features available in the SMPL-X model. Creating garments for the digital human has also evolved significantly and despite the advantages of automated garment design, challenges remain, not least of which is the generation of large amounts of data required to train a ML model for cloth simulation. Physics-based methods provide a cost effective approach to generating this data and the most widely applied approach is based on an energy mass-spring model proposed initially by Provot et al., in 1995 and since refined. An analysis of the cloth simulation software packages available for the creation of a ML dataset, revealed Houdini as the package with the most comprehensive feature set, yielding high quality simulations and the potential for automation. The data output file format is a key factor in the training of a ML model with meshes commonly used, in addition to the simple format of Point Cloud representations. The literature on existing cloth prediction ML models reports on the variety of ways in which these models are trained including PBS and 3D or 4D scans. Prior to creating a database this analysis provides an understanding of the potential compatibility of the proposed dataset with existing ML models. A comprehensive review and analysis of the published literature facilitates an informed approach to developing a realistic, flexible and cost effective cloth prediction model.

3 Implementation and Outcomes

Following the analysis of the published literature and associated data, the project implementation was then initiated with the generation of the digital human body. This step was followed by re-targeting the animation to the body, the creation of the garment and its simulation on the body. Various approaches to achieving these objectives were trialled and the workflow that emerged from that process is detailed below in section 3.1 and outlined in Figure 24.

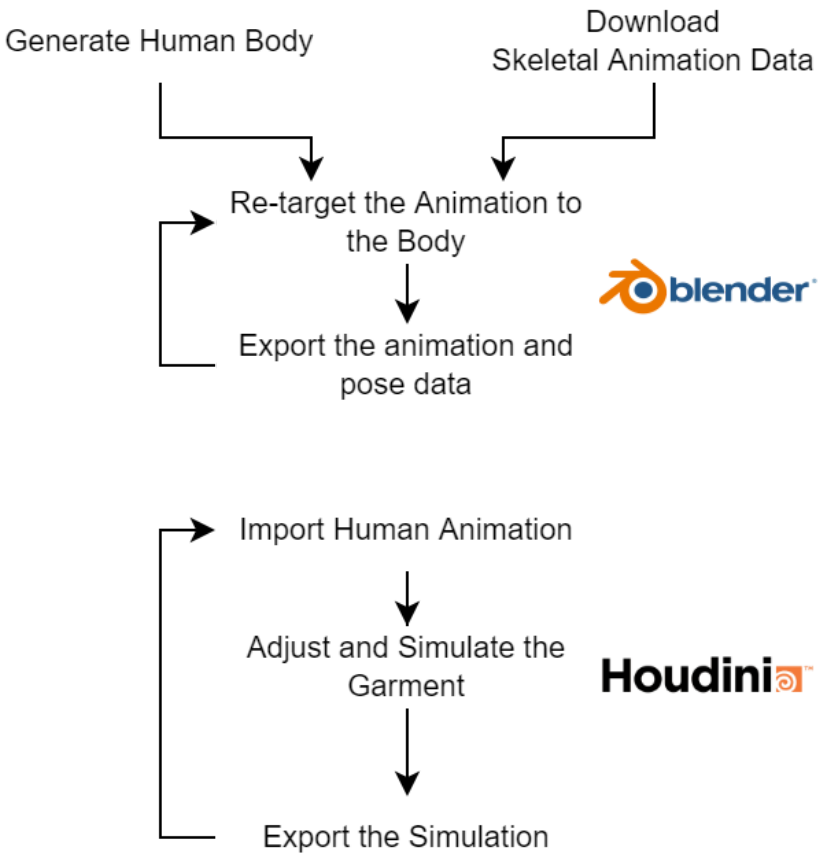

Figure 24. An overview of the workflow for the database generation process.

3.1 Detailed Description of the Database Generation Process

3.1.1 Model Generation

The SMPL Model was chosen as the base human for all of the simulations. It emerged as the natural choice for this project as it is public, commonly used as a basis for research in the field of computer graphics at the moment, and it is capable of representing humans in a diverse range of shapes and sizes.

The research group behind the SMPL model created a blender plugin that was used to generate the human meshes. The SMPL model is also publicly available on GitHub to use directly, however accessing and manipulating the pose information is not well supported.

To generate the model a script was written using the Blender Python API as illustrated in Table 3 below. The script first selects between a male, female or neutral body. It then generates a template human of that gender. It randomises the height, weight and shape parameters of the model, snaps the model to the ground plane and then assigns it a name.

```
def SMPLXsetup():
    if gender == 'male':
        bpy.data.window_managers["WinMan"].smplx_tool.smplx_gender = "male"
    elif gender == 'female':
        bpy.data.window_managers["WinMan"].smplx_tool.smplx_gender = "female"
    else:
        bpy.data.window_managers["WinMan"].smplx_tool.smplx_gender = "neutral"
    body = bpy.ops.scene.smplx_add_gender()
    bpy.ops.object.smplx_random_shape()
    bpy.ops.object.smplx_update_joint_locations()
    bpy.data.window_managers["WinMan"].smplx_tool.smplx_handpose = "flat"
    bpy.ops.object.smplx_set_handpose()
    height = random.uniform(min_height,max_height)
    weight = random.uniform(min_weight,max_weight)
    bpy.data.window_managers["WinMan"].smplx_tool.smplx_height = height
    bpy.data.window_managers["WinMan"].smplx_tool.smplx_weight = weight
    if gender != 'neutral':
        bpy.ops.object.smplx_measurements_to_shape()
    bpy.ops.object.smplx_snap_ground_plane()
    obj_object = bpy.context.selected_objects[0]
    obj_object.name = "SMPLX-mesh"
    myObj = bpy.context.active_object.parent
    bpy.ops.object.select_grouped(type='CHILDREN_RECURSIVE')
    myObj.name = "SMPLX-Armature"
```

Table 3. Blender Script to generate SMPL models

3.1.2 Skeletal Animation Generation

The skeletal animation is needed to put the human mesh in a particular pose. A wide variety of poses are needed to train the model and therefore a large dataset of skeletal animations was essential. The two most prevalent options were the Carnegie Mellon University (CMU) Motion Capture Database, a dataset of over 300 animations captured by the CMU in 2007. The second option and that chosen for this project was the use of over 3,000 high quality motion captured animations from the mixamo.com Adobe website.

In order to drape clothing over our model the animation needs to start in either a T-Pose or an A-Pose. These are often described as the canonical poses. The T-Pose is where the model has their arms outstretched in a T shape, and an A-Pose is more by their side in the shape of an A. The T-Pose was the default reset pose for the Mixamo skeletons and SMPL models so that pose was selected to be the canonical pose for this dataset (Figure 25).

In order to start each simulation with the model in a T-Pose a new keyframe needed to be added at the start of every animation. Twenty frames were designated for transitioning from T-Pose to the Mixamo animation. Unfortunately, this process needed to be done manually as the Blender API lacked the ability to access the keyframe information of skeletal animations. However once this step is done manually for one animation, it can then be retargeted to any shaped model.

Figure 25. Setup of Mixamo animation in Blender with a T-Pose keyframe added

3.1.3 Retargeting

Retargeting is the process of transferring skeletal animations from one skeleton to another. In this case the Mixamo skeletal animation needs to be transferred onto the skeleton of the SMPL mesh. This is done using another plugin for Blender called Rokoko. The Rokoko plugin is primarily intended for recording data from motion capture suits but it has also the ability to retarget skeletal animations.

This process required several steps such as selecting the skeletons to be retargeted and assigning which bones correspond to which on the two skeletons. To achieve this, a script was written that first selects the two relevant skeletons as illustrated below in Table 4. It then inputs which bones correspond to each other (some are linked automatically from standard naming conventions). It then runs the retargeting process and when finished snaps the new animated model to the ground plane.

```
def retargeting():
    #select source and target
    bpy.data.scenes["Scene"].rsl_retargeting_armature_source =
bpy.data.objects["Armature.004"]
    bpy.data.scenes["Scene"].rsl_retargeting_armature_target =
bpy.data.objects["SMPLX-Armature"]
    bpy.ops.rsl.build_bone_list()

    #assign bones
    bpy.data.scenes["Scene"].rsl_retargeting_bone_list[7].bone_name_target="left_shoulder"
    #Left Hand
    for x in range(10, 25):
        bpy.data.scenes["Scene"].rsl_retargeting_bone_list[x].bone_name_target = "null"

    #Right Hand
    for x in range(29, 44):
        bpy.data.scenes["Scene"].rsl_retargeting_bone_list[x].bone_name_target = "null"

    bpy.data.scenes["Scene"].rsl_retargeting_bone_list[44].bone_name_target = "left_hip"
    bpy.data.scenes["Scene"].rsl_retargeting_bone_list[45].bone_name_target = "left_knee"
    bpy.data.scenes["Scene"].rsl_retargeting_bone_list[48].bone_name_target = "right_hip"
    bpy.data.scenes["Scene"].rsl_retargeting_bone_list[47].bone_name_target = "null"
    bpy.ops.rsl.retarget_animation()
    bpy.ops.object.smplx_snap_ground_plane()
```

Table 4. Blender Script to Retarget Animations

3.1.4 Exporting Pose and Mesh Data from Blender

In order to run the cloth simulation in Houdini the human animation needs to be exported from Blender. The human mesh animation is exported as an Alembic file (.abc). This is a filetype designed by Sony that is intended for storing mesh animations.

The new skeletal animation was also exported in the BVH format as a .bvh file. This is a filetype that is designed to store motion capture data. It stores all the bones in a tree structure and then includes the positional and rotational information for each bone at each keyframe as a list of doubles. The skeletal animation is not necessary in Houdini however, it will be beneficial information for training the ML model.

To automate this step a script was written in Blender that selects and exports the human mesh as an Alembic file in a new folder (if one with the same name does not already exist) and also selects the skeleton and exports it as a BVH file to the same folder (Table 5). The number of animations that have already taken place are included as a parameter, and the gender global variable is checked so that the file can be named accordingly.

```
def exportResults(skeleton_number, version_number):

    version_number = str(version_number).zfill(4)

    if gender == 'neutral': gender_number = '0'
    elif gender == 'male': gender_number = '1'
    else: gender_number = '2'

    filename = skeleton_number + version_number + gender_number
    dir = 'D:\\Personal\\College\\Final Year Project\\Database\\' + filename
    bvh_file_loc = 'D:\\Personal\\College\\Final Year Project\\Database\\' +
filename + '\\' + filename + '.bvh'

    #export bvh
    bpy.context.scene.objects["SMPLX-Armature"].select_set(True)
    if os.path.isdir(dir):
        bpy.ops.export_anim.bvh(filepath=bvh_file_loc, check_existing=True,
filter_glob='*.bvh', frame_start=0, frame_end=60)
    else:
        os.mkdir(dir)
        bpy.ops.export_anim.bvh(filepath=bvh_file_loc, check_existing=True,
filter_glob='*.bvh', frame_start=0, frame_end=60)

    #export alembic
    bpy.ops.object.select_all(action='DESELECT')
    bpy.context.scene.objects["SMPLX-mesh"].select_set(True)
    abc_file_loc = 'D:\\Personal\\College\\Final Year Project\\Database\\' +
filename + '\\' + filename + '.abc'
    bpy.ops.wm.alembic_export(filepath=abc_file_loc,selected=True)
    print("exported alembic file")
```

Table 5. Blender Script to export mesh and skeletal animation

3.1.5 Importing to Houdini

The meshes from Blender need to be iteratively imported to Houdini in order to perform the simulation. To achieve this, a script was written for the “Python Source Editor” (Table 6). This is a window in Python where Python code and specific functions can be written, that can later be called using the Python Shell.

The script in the Source Editor searches the dataset folder. It iterates over every folder and looks for any human mesh animation files with no corresponding cloth simulation files. It creates a list of the human mesh animation files and subsequently runs the cloth simulation for each file.

```
def runLoop():
    folders = [join(database_loc,folder) for folder in listdir(database_loc) if
(isdir(join(database_loc,folder)) and (len(basename(folder)) == 8))]
    files =[]
    for folder in folders:
        for file in listdir(folder):
            if (isfile(join(folder,file)) and file.endswith('.'+str(ext))):
                  print(join(folder,file))
                  files.append(join(folder,file))

    importerNode = hou.node("/obj/cloth/alembic1")

    for file in files:
        importerNode.setParms({"fileName":file})
        exportCloth(file)
```

Table 6. Houdini Script to search for and simulate each animation

3.1.6 Creating a Houdini Digital Asset (HDA)

In Houdini the process of the cloth simulation is organised in a node network. The network of nodes can be packaged together into a single node and instances of that node-network can be created, called HDAs as illustrated in Figure 26. A user interface can be created for an asset by promoting parameters of the contained nodes up onto the asset node. This can be combined with scripting to create powerful user interfaces. More importantly these nodes can be packaged and exported as .hda files to be imported into any other Houdini project.

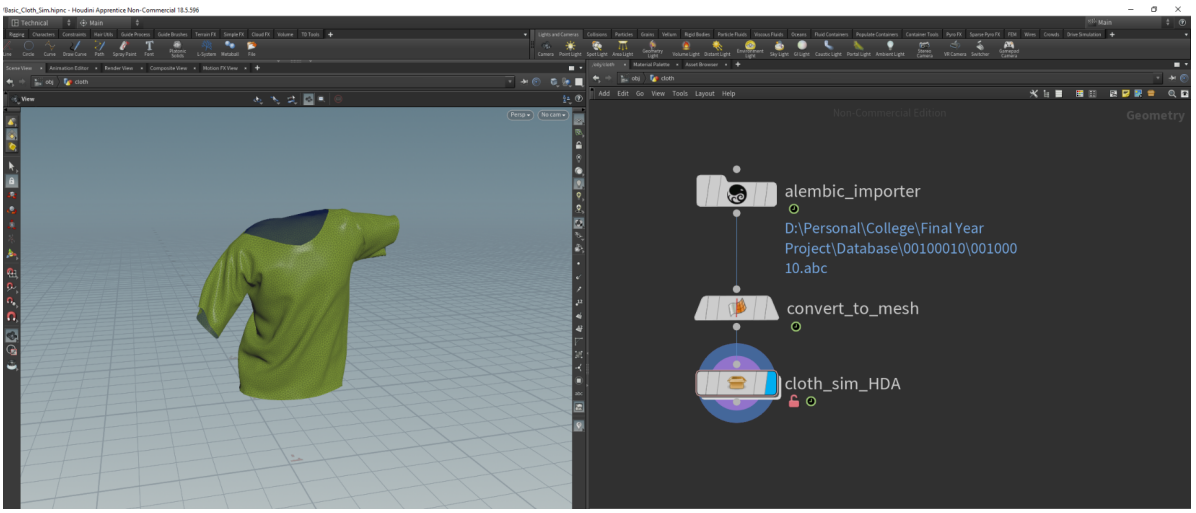

Figure 26. Houdini Node Setup with HDA selected

For this project a HDA was created containing all the nodes necessary to simulate the draping of clothing over a human mesh (Figure 27). Parameters such as details concerning the cloth simulation were promoted so that they could be controlled by the master node. The human mesh is used as an input to the HDA and the output is the simulated garment mesh.

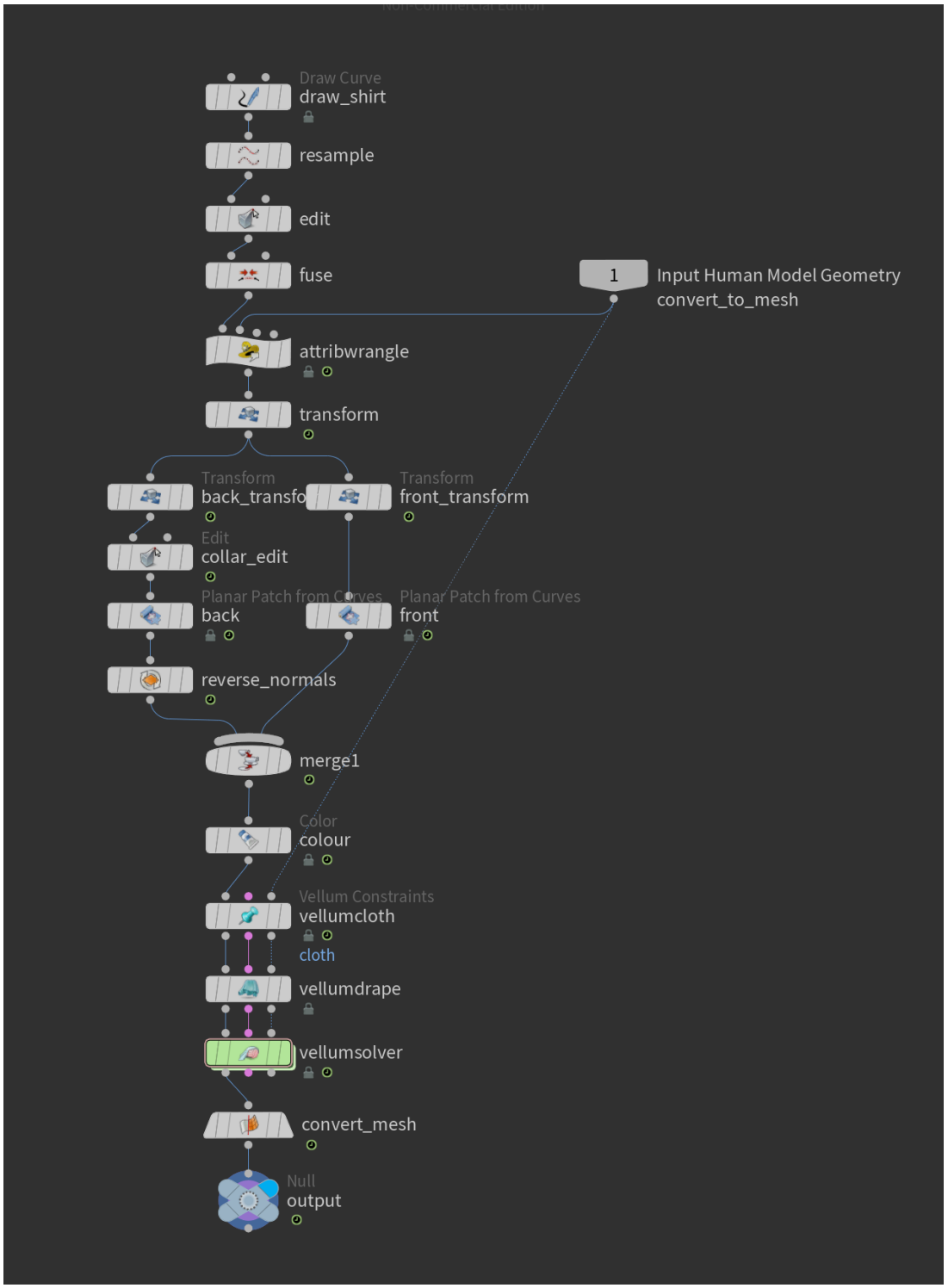

Figure 27. Cloth Simulation Sub-Network of the HDA

3.1.7 Generating Garments in Houdini

For experimental purposes a T-Shirt was chosen for the simulation. This process would work for any garment type provided the garment is initially set up in Houdini.

To generate the t-shirt garment for this project a rough t-shirt shape was manually drawn using the **Draw Curve node** in Houdini (Figure 28).

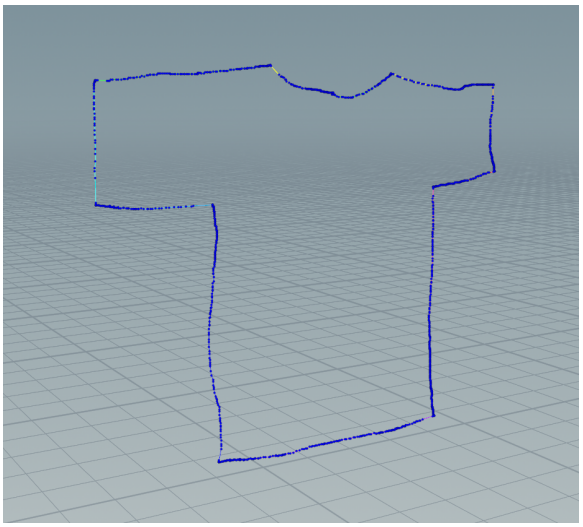

Figure 28. T-Shirt after Draw Curve Node

Each seam for the t-shirt was drawn separately so that they could later be stitched together. This can be seen by the different colored lines representing each seam. The shape is then simplified using a **resample node**. This node simplifies and reduces the number of points along a line (Figure 29).

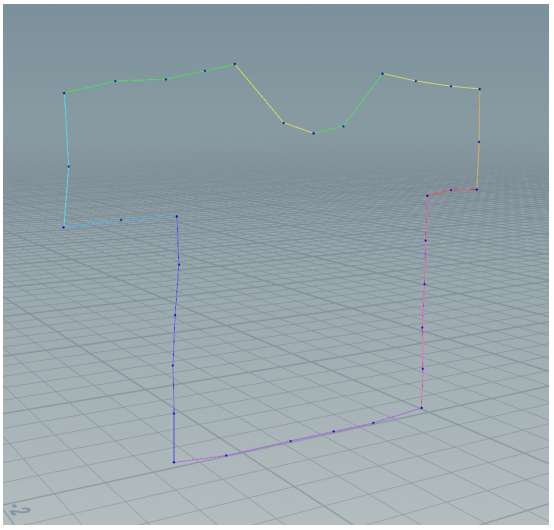

Figure 29. T-Shirt after the resample node

The shape is then converted to a plane using a **Planar Patch node**. This node generates a plane filled with triangular polygons. Triangular polygons lead to more organic deformations than rectangular polygons. Specifying the edge length of the triangular polygons for this node will determine the resolution of resultant mesh. The edge length of these polygons is promoted to be one of the global parameters so that it can be controlled without diving into the network of simulation nodes.

This plane is translated both in front and behind the human mesh, using a **Transform node**. This is to create the front and back of the t-shirt. The normals of the reverse plane are flipped using a **Reverse Normals node** (Figure 30).

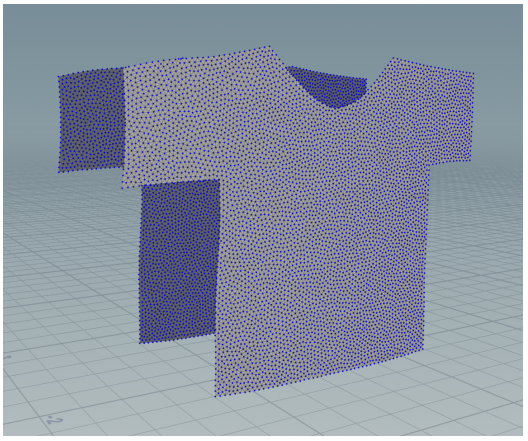

Figure 30. Triangulated front and Back Mesh

An **attribute Wrangle node** and VEX code is then used to extract the Y value of the highest point on the human input geometry mesh. This value is then used to transform the height of the garment relative to the height of the human. Finally, a colour is applied to the cloth using a colour node for visualisation purposes.

3.1.8 Simulating The Garment

The next stage of implementation required the use of Vellum nodes in Houdini to simulate the garment.

A **Vellum Constraints node** is used to specify the parameters for the cloth simulation. These parameters include: mass per vertex, vertex collision size, stretch stiffness, bend stiffness etc. The **Vellum Drape node** and the **Vellum Solver node** both simulate the clothing and have parameters such as the number of substeps, collision passes and forces such as gravity or wind.

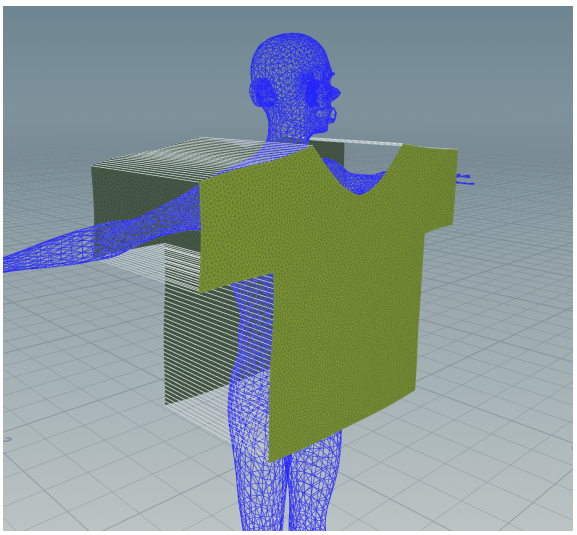

Figure 31. Example of stitching and draping in Houdini

The Vellum Drape differs from the Solver node in that it has the ability to stitch seams together. It typically comes first and handles the initial draping of the garment on the human. It freezes the human mesh animation at the first frame (which in this case is the T-Pose) and simulates the specified seams, stitching together and draping over the human for a set number of frames (Figures 31 and 32). After some experimentation, 20 frames was determined to be a sufficient amount of time for the cloth to come to rest on the body, without adding unnecessary time to the dataset generation.

A built-in Houdini function was applied that initially shrinks and then slowly expands the human mesh so that it is less likely for the cloth stitching to intersect with or miss the body.

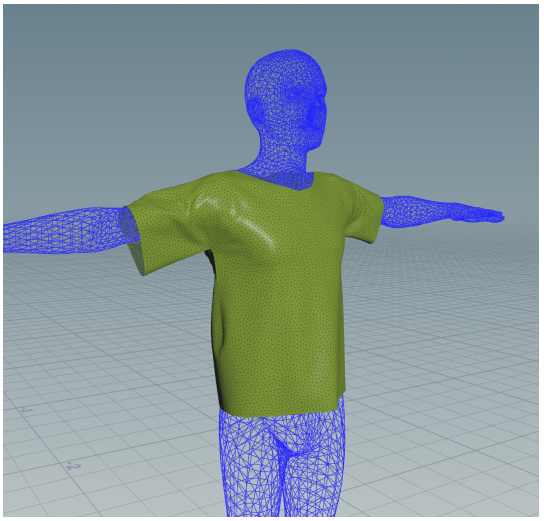

Figure 32. T-Shirt Mesh After Running the Vellum Drape Node

Once this simulation is completed the geometry of the garment resting in the T-Pose is saved to cache. That cached geometry is then used as the starting geometry for the Vellum Solver node to simulate over the animated human (Figure 33). The final output is an animated mesh where the cloth is simulated from T-Pose and then transitioning to and through the animation.

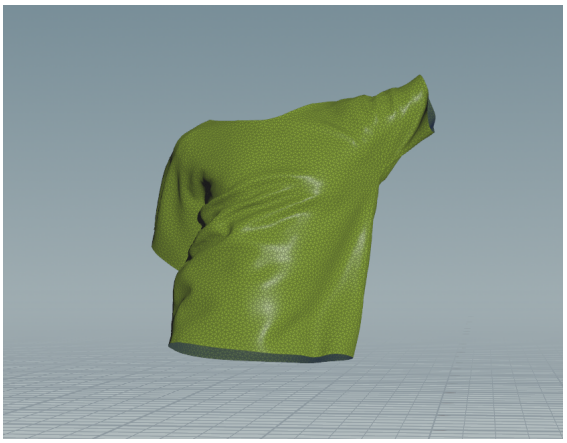

Figure 33. T-Shirt Mesh After Running the Vellum Solver Node

3.1.9 Exporting from Houdini

The final stage of implementation requires that the simulated cloth is exported and saved in the dataset. The Apprentice version of Houdini used in this project does not natively allow mesh geometry to be exported and therefore a script was written to instead export the geometry represented as a Point Cloud in an NPY file (Table 7).

The script iterates over each frame, and for each point on the cloth geometry at that frame, it takes the position and velocity attribute, and adds it to an array. It then exports the array as an NPY file to the same location as the Alembic geometry input file.

```
def exportCloth(filename):
    node = hou.node("/obj/cloth/cloth_sim1")
    frames = 59
    geo = node.geometry()
    num_points = len(geo.points())
    array = np.zeros((6*frames+3,num_points))
    for frame in range(frames):
        Pindex=0
        hou.setFrame(frame+1)
        for point in geo.points():
            posAtt = geo.findPointAttrib("P")
            velAtt = geo.findPointAttrib("v")
            attrib_count = posAtt.size()
            array[(frame*6):(frame*6)+3,Pindex] = point.attribValue(posAtt)[:]
            array[(frame*6)+3:(frame*6)+6,Pindex] = point.attribValue(velAtt)[:]
            Pindex = Pindex + 1

    #save to npy file
    np.save(filename[0:len(filename)-3] + "npy", array)
```

Table 7. Houdini Export Script

3.1.10 Exported Array Structure

A clear understanding of the exported structure is required to properly interpret and use the data. The structure of the generated array is illustrated in Figure 34 which displays the first frame of the animation. The vertices are stored in rows as indicated in the sample labelled 0 through 30 in Figure 34. The first three columns of the array store the Position x, y and z coordinate values. The next three columns store the velocity in the x, y and z direction. The following frame is stored in the next 6 columns and so on. For example on the 15th frame, the array coordinates for the second vertex's x velocity is:
[(frame * 6) + x, vertex number] = [(15*6) + 3 ,2] = [93, 2]
(Where x ranges from 0 through 5 depending on what position or velocity value is needed)

	P[x]	P[y]	P[z]	v[x]	v[y]	v[z]
0	-0.163358	0.911293	0.00639537	0.0569648	-0.166633	0.181818
1	-0.164978	0.912593	0.0161181	0.0128549	-0.16408	0.180226
2	-0.163104	0.91659	0.00146737	0.0677508	-0.168436	0.17923
3	-0.164406	0.918252	0.00941099	0.0437129	-0.166054	0.176179
4	-0.165064	0.922492	0.0154808	0.0356913	-0.165088	0.171611
5	-0.165587	0.927911	0.00949518	0.0408214	-0.166526	0.168448
6	-0.164588	0.924359	0.00430476	0.0476396	-0.17164	0.171756
7	-0.161458	0.910026	-0.0032773	0.061959	-0.170088	0.181712
8	-0.161729	0.915994	-0.00643973	0.0746512	-0.172098	0.178657
9	-0.16121	0.919381	-0.0119803	0.0715452	-0.173385	0.174734
10	-0.163125	0.926176	-0.00783849	0.0667298	-0.175409	0.168563
11	-0.163541	0.923092	-0.00217463	0.0659126	-0.174201	0.17527
12	-0.165982	0.932204	0.0146205	0.0439882	-0.167177	0.166837
13	-0.166291	0.93631	0.00724014	0.045982	-0.169945	0.158913
14	-0.165142	0.931088	0.000908867	0.053494	-0.173399	0.158915
15	-0.167187	0.941999	0.0132021	0.0329572	-0.167856	0.154729
16	-0.167067	0.945212	0.000557652	0.0423878	-0.173507	0.153897
17	-0.16748	0.944812	0.00759591	0.0341535	-0.172956	0.151716
18	-0.165982	0.938464	-0.00042783	0.049088	-0.172949	0.157428
19	-0.163553	0.93582	-0.0137806	0.0523996	-0.177863	0.159699
20	-0.165368	0.940951	-0.00807428	0.0487179	-0.176325	0.155926
21	-0.164435	0.933287	-0.00648412	0.0562441	-0.174315	0.162618
22	-0.16823	0.951859	0.0121449	0.0211662	-0.166218	0.144287
23	-0.169189	0.961639	0.0111133	0.00446677	-0.17257	0.141009
24	-0.168656	0.951054	0.00468689	0.0265592	-0.175323	0.144977
25	-0.168902	0.958584	0.00484776	0.0185931	-0.175474	0.14156
26	-0.1678	0.952471	-0.00223007	0.0328141	-0.175939	0.146959
27	-0.165716	0.949521	-0.0142474	0.0357825	-0.179458	0.146477
28	-0.166575	0.947613	-0.00706524	0.0399291	-0.176346	0.148882
29	-0.168729	0.95976	-0.00282017	0.022409	-0.176589	0.139618
30	-0.167206	0.961295	-0.0157965	0.0246155	-0.180566	0.136623

Figure 34. Example First Frame of Output Cloth Simulation Data

3.2 Outcomes

3.2.1 Steps Saved Through Automation

Significant resources produced from this project are the scripts written for Blender and Houdini which speed up the process of generating cloth simulations for machine learning models.

The script in Blender loops over the process of: (a) creating a random SMPL body shape; (b) retargeting; and (c) exporting the body and skeletal animation. From each skeletal animation the script can create an infinite number of animated SMPL body shapes for each, male, female and neutral body types. So once the skeletal animation is set up in Blender, a near infinite amount of mesh animations can be generated with a single click.

The script in Houdini (a) iterates over each folder (containing the animated SMPL mesh); (b) loads the geometry, (c) runs the simulation, and (d) exports the cloth simulation to each respective folder.

Table 8 below summarises the number of clicks that are no longer necessary, i.e. steps that would have to be performed for every simulation without the scripts.

	Clicks Automated	Text Boxes Automated
Blender		
Generating the SMPL model	**3**	
Retargeting	**20**	
Exporting .abc files	**6**	**2**
Exporting .bvh files	**4**	**1**
Deleting mesh	**3**	
Houdini		
Reloading the Geometry	**2**	**1**
Repositioning the Garment	**5**	
Running the simulation	**2**	
Exporting to NPY file	Requires Script	
Approximate Totals	**45**	**4**

Table 8. Steps Saved Through Automation

A total of approximately 45 clicks per simulation were automated and 4 text boxes per simulation were automated by scripts.

It is also worth considering the time spent waiting for the result of each click. Each step in Blender requires several seconds to load, and each cloth simulation in Houdini takes approximately 30 minutes. The time saved by these scripts is not only the time it would take to perform 45 clicks and fill out 4 text boxes, but also the waiting time between a simulation and making the next click.

In summary, the Blender script takes 20 minutes to generate the dataset of 500 humans with each human animation takes on average, 2.4 seconds to process.

The Houdini script took 14 hours and 20 minutes to process the 500 simulations with each simulation taking, on average 103.2 seconds to complete.

The dataset takes in total 14 hours and 40 minutes.
Without any of the automation steps, and assuming the HDA is already set up, creating a simulation took 3.5 minutes. This means it would approximately take approximately 29 hours to generate the same dataset manually.

3.2.2 The Final Dataset

The final dataset consists of 600 simulations, made up of 100 SMPL body shapes, in 6 animations (Figure 35) . The dataset was split into training, validation and test sets with 500, 20 and 80 simulations, respectively similar to the dataset of GarNet.

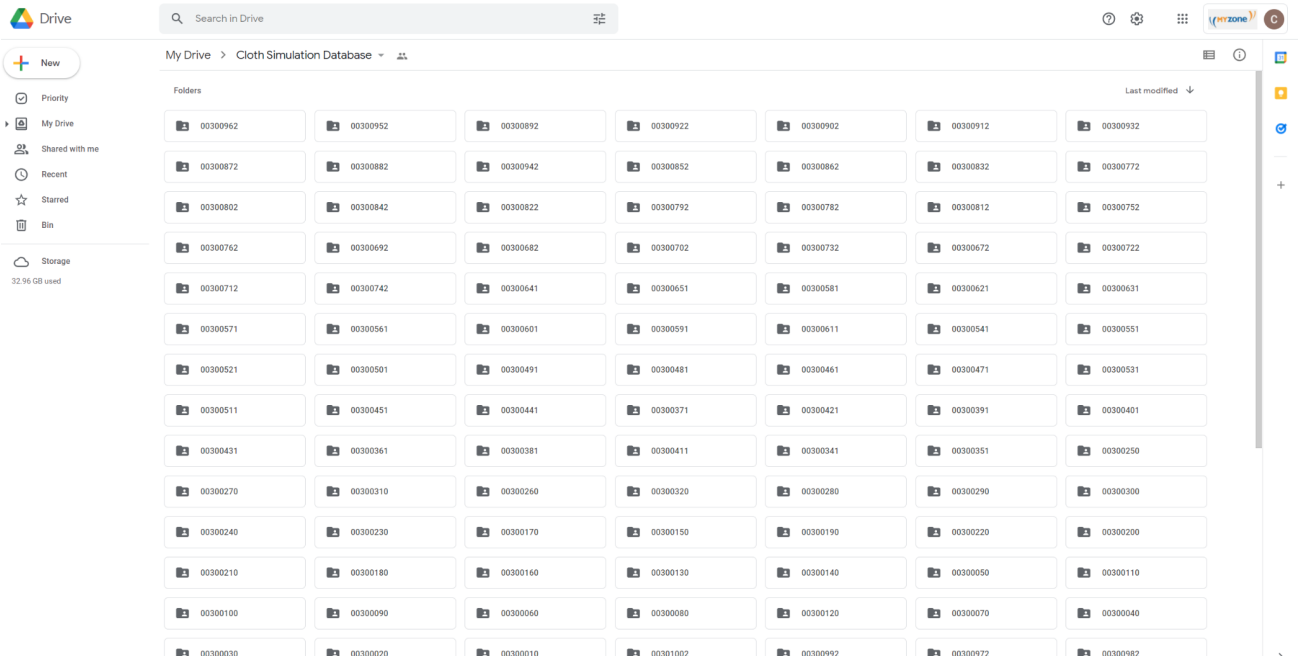

Figure 35. Folders of all simulations in the database

Each animation is saved to its own individual folder that uses the following **naming scheme**:

Skeletal Animation (3 digit) + index (4 digits) + Gender (1 digit)
Eg: 00110001.filetype

The number for the neutral, male and female gender are referred to as 0,1,2 respectively. Another number would need to be added if the dataset used more than one garment.

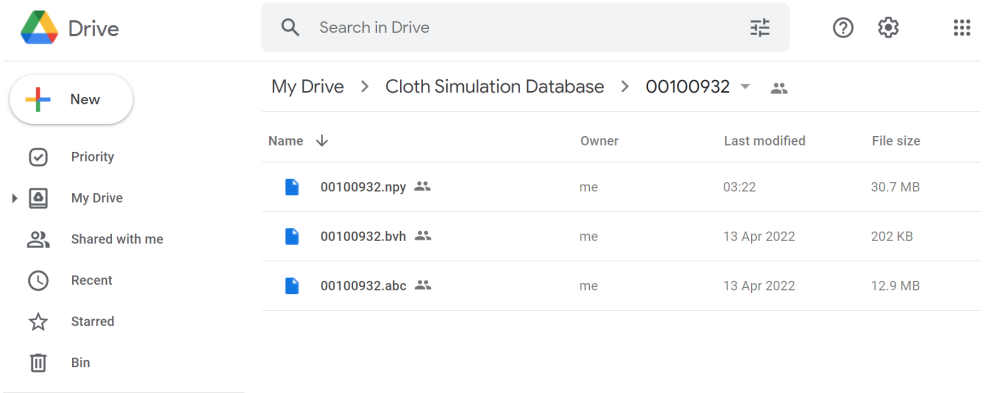

Figure 36. Contents of each simulation folder in the dataset.

So in the case illustrated in Figure 36, data for the first skeletal animation is shown, which is the 93rd dancing animation, and the human model is a female.

The script files for generating the data can be found on GitHib using this link: https://github.com/cormacmadden/Cloth-Sim-Database-Toolset

The documentation for the dataset includes a description of the structure of the data and the naming scheme. It also has a brief description of what the scripts are and how to use them.

Link to the Dataset Hosted on OneDrive (Check Github for most up to date links): https://drive.google.com/drive/folders/15Pm41YiPCGp4vHTGhWTkBAL64DMXvs4E?usp=sharing

3.2.3 Justification of File Formats

Careful consideration was given to the final file formats generated for this dataset.
In the case of the clothing, mesh data has been generated that could be stored as a mesh, a point cloud, or using a volumetric representation such as voxels. The Point Cache format just contains vertex positions, so to view the animated mesh in another application, you would also need to export the mesh itself in another format such as Alembic, GLTF, OBJ or FBX. However, despite these limitations Point Clouds are the format used for training the DeepWrinkles model and for storing the garments generated in Marvelous Designer to train TailorNet. TailorNet stored their garment point cloud files as .pc2 (Point Cache2) files. Point clouds were also used in the GarNet model to represent the human body and efficiently detect collisions.

Given the limitations of the Apprentice Version of Houdini outlined above, a script to generate a custom point cloud file was written and also included some other potentially useful data such as velocity. Mesh formats could then later be added with a single node once the version of Houdini is upgraded to the Personal Licence.

When creating a custom point cloud array the options of NPY, CSV or Text files were considered and NPY (.npy) format was chosen as it has the potential to be up to 100 times faster than CSV or Text files. This is due to the fact that it stores an array, in binary form, of only one variable type (in this case a double). This file format is intended to be used with Python and distributed as part of the main numpy package.

When deciding the **pose animation file format** The Biovision Hierarchical data (BVH) file format was originally developed by Biovision (a motion capture service company) to distribute mocap data to their customers. Later, it became a very popular format for storing mocap data.

Another file format that was considered to store the pose data was the Acclaim format. The Acclaim format is made up of two files, a skeleton file (.asf) and a motion file (.amc). This is done because typically the same skeleton will be used for many animations and so it makes sense to store the skeletal information separately as opposed to storing it in each motion file. BVH was chosen above other file formats such as C3D and Acclaim for several reasons.
The C3D format stores only the coordinates for each tracked marker and unlike BVH doesn't store the hierarchical relations between joints, i.e., their physical relations.
The Acclaim format of separate skeletal and motion files was not appropriate because a new skeleton is being generated for each new animation in this project and therefore an extra file was considered unnecessary.

3.2.4 Licensing

When publishing a dataset it is essential to ensure that all the content is free to distribute, and that credit is given where due. SMPL has a US patent (US10395411B2) and the SMPL-X Model is a derivative of that. Fortunately however, these models are available free for research purposes.

The SMPL-X **Body** licence is a subset of SMPL-X **Model** licence, which excludes the tools to create 3D bodies using the shape blendshapes of the SMPL-X Model. A SMPL-X Body contains a 3D mesh, a skeleton rig, pose blendshapes, and dynamic blendshapes. A SMPL-X Body can be shared in the form of OBJ file(s), animated FBX file(s), or as a set of mesh vertices, skeleton rig, pose blendshapes and dynamic blendshapes in other appropriate 3D file formats.

The SMPL-X Body is licensed under the Creative Commons Attribution 4.0 International Licence. Appropriate credit must be given, a link provided to the licence, and any changes that were made should be indicated. This should be done in a manner that does not suggest the licensor endorses you or your use of their model.

The dataset associated with this paper stores the SMPL-X Body mesh as an Alembic file which would be deemed appropriate and the credit has been given to the authors of the paper in the README.md file. The rest of the data is generated using Houdini which is free to use for any non-commercial purposes.

4 Evaluation of project implementation and future research opportunities

4.1 Project Limitations

One of the challenges faced at the start of the project was the insufficient experience of using various Python modules and attempting to run code from published research papers. A considerable amount of project time was also spent trying to use SMPL without the Blender add-on. Ultimately however, the add-on provided a workable solution for creating this dataset. Ideally the SMPL model would be generated using a Python script outside of Blender. This would provide the native pose parameters and the blend shapes of the model that would be useful in training a garment prediction model. This was experimented with, and despite being able to generate SMPL models in random poses, animating the meshes proved to be more complicated. The pose of the SMPL model is determined by an array of 24 values, one for each bone. However, it is difficult to manipulate these in an intuitive way. SMPL has no native solutions for exporting meshes with skeletons. Third parties created workarounds for exporting as an FBX with a skeleton instead of OBJ, but getting this to work proved difficult. The Blender add-on generated a skeleton with each mesh that was easy to manipulate. One advantage of using the Blender method is that the SMPL model could be easily swapped out for another character model. A game could then quickly create training data for their model by replacing the human model with the game character, and replacing the garment with the character's clothing.

Another limitation of the dataset generation process relates to the skinning of the SMPL model. SMPL's learned skinning model needs to be manually updated each frame in the Blender add-on and the mesh deformations are not saved. Essentially, a button needs to be clicked in order for the learned skinning model to be applied. A script was written to click this button and update the skinning every time a frame was changed, but this did not work for exporting the final mesh. One solution would be a script that would save the cache of the new geometry for each frame, and then export it all at once.

A further limitation of the Blender add-on is that it is not capable of applying dynamic blend shapes. The research team behind SMPL have committed to delivering future versions of the SMPL-X add-on that will have a way to export animations with corrective blend shapes such as skinning and soft-tissue dynamics.

While many of the challenges encountered were learning opportunities, the time constraints of the project meant that compromises had to be made. There is an opportunity however to extend the scope of this project to address some of the issues encountered and the prospect of applying a different approach could be exciting.

4.2 Future Work

This project provided an excellent platform to develop the skills required to generate a database for cloth simulations. From this acquired knowledge and experience, the potential to further refine and develop the process is more apparent. It would be interesting to experiment with a method that skipped the Blender stage of the process.

Another approach to the data generation process, would be to add the retargeting process to the Houdini Digital Asset instead of doing it in Blender, containing more stages within the same application. A retargeting solution in Houdini was discovered at a late stage in the project so it would be interesting to see it added to the HDA.

There are a number of approaches that could be trialled to improve the dataset itself. These include the use of: additional simulations; simulations with more substeps; the use of more garment types; and a wider range of skeletal animations. The new STAR model would also provide benefits as it offers an even wider range of body shapes than SMPL with a focus on muscle definition (Osman, Bolkart and Black, 2020).

From an industry perspective, ideally in the future, prediction models such as TailorNet (Patel, Liao and Pons-Moll, 2020) would be more widely accessible to 3D animators, whether through new features in existing software or through new specialised software packages. Cloth prediction models are still a highly specialised technique and require highly trained developers to implement - keeping the process beyond the reach of lower budget organisations.

It quickly became apparent working on this project that the topic of ML clothing predictions has an exciting future and is closely interlinked with many other rapidly developing areas of computer graphics research. From generative human models to automating garment design, machine learning has enabled these concepts to reach beyond what anyone would have expected a decade ago. Due to the progression in ML in recent years with techniques such as neural implicits yet to be taken advantage of, one should expect to see virtual try-on apps becoming more commonplace and clothing in video games becoming much more realistic.

4.3 Discussion of ethical issues

This project intends to assist the development of more efficient 3D clothing simulation which has a wide variety of uses and applications across the video-gaming, film and retail sectors. There are general ethical issues to consider when working as part of a team, within the animation industry, and more specifically ethical issues relating to cloth simulation.

Developing cloth simulation as part of an animation team will typically involve operating within a code of ethics which aims to foster a respectful inclusive place of work. Animation is a highly collaborative process and requires relationships that are built on trust and where mutual respect is valued. Honesty is paramount when issues arise as the overall project is impacted by all its parts. Maintaining a flexible approach so that workarounds can be used to solve problems is also core to this type of collaborative project.

Intellectual property (IP) needs to be respected and the work and IP rights of others considered at all stages of the process. It is vital for animators to ensure that public domain materials are in fact open source and that the image content created is not a minor adaptation of another's work. Non-disclosure agreements will be required on some projects and there is both a legal and ethical obligation to maintain that agreement.

In terms of cloth animation specifically, an ethical issue that was considered during this project was that of inclusion. Technology in the past has had issues with voice and facial recognition of people from varying ethnicities. In the case of this dataset, voice and skin colour is not relevant. Body shape and gender however, are. This project accounted for a wide variety of body shapes: tall, short, fat, skinny, male, female. However, despite SMPL having been learned from thousands of 3D body scans, SMPL is limited to the distribution of the training data which is based on a census of the US population in 1990. It is therefore limited and cannot represent a range of body shapes including those of babies, body builders or amputees. It would be of benefit to recreate or extend this dataset with other models such as SMIL for infants (Hesse *et al.*, 2018), SMAL for animals (Zuffi *et al.*, 2017), and STAR which is much better at dealing with larger muscle deformations than SMPL. Take for instance the use of cloth simulation for human avatars in a virtual or augmented reality work meeting. The cloth model should work for as many individuals as possible regardless of body abnormalities or whether they are using equipment such as a wheelchair or walking aid.

The ethical issues in relation to cloth simulation in the animation industry are diverse and in some cases will be identified by the organisation and an ethical framework established. In other instances the ethical issues will be project related and will require careful consideration of how best to ensure no offence, upset or harm is caused by the project implementation or by the project outcomes.

5 Conclusion

The three main objectives of: exploring available cloth simulation technologies; creating a training dataset; and automating the data generation process were all met. Challenges however were encountered at different stages of the implementation process as discussed above. This project generated a simulated cloth database of garments on animated SMPL models using Houdini. Scripts were written to automate and increase the speed of the process. Although the dataset created is not the optimal solution for training a cloth prediction model as it uses the Blender add-on which limits functionality, it does nonetheless achieve its objective and the code used to create it is a valuable starting point. The implementation process undertaken in this project demonstrates that the workflow is a viable way of generating training data.

6 References

Alldieck, T., Magnor, M., *et al.* (2019) “Learning to reconstruct people in clothing from a single RGB camera,” in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, pp. 1175–1186. doi: 10.1109/CVPR.2019.00127.

Alldieck, T., Pons-Moll, G., *et al.* (2019) “Tex2shape: detailed full human body geometry from a single image,” in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, IEEE, pp. 2293–2303. doi: 10.1109/ICCV.2019.00238.

Allen, B., Curless, B. and Popović, Z. (2003) “The space of human body shapes: Reconstruction and parameterization from range scans,” in *ACM SIGGRAPH 2003 Papers on - SIGGRAPH ’03*. *ACM SIGGRAPH 2003 Papers*, New York, New York, USA: ACM Press, p. 587. doi: 10.1145/1201775.882311.

Anguelov, D. *et al.* (2005) “SCAPE: Shape completion and animation of people,” in Marks, J. (ed.) *ACM SIGGRAPH 2005 Papers on - SIGGRAPH ’05*. *ACM SIGGRAPH 2005 Papers*, New York, New York, USA: ACM Press, p. 408. doi: 10.1145/1186822.1073207.

Baraff, D. and Witkin, A. (1998) “Large steps in cloth simulation,” in *Proceedings of the 25th annual conference on Computer graphics and interactive techniques - SIGGRAPH ’98*. *the 25th annual conference*, New York, New York, USA: ACM Press, pp. 43–54. doi: 10.1145/280814.280821.

Baumberg, A. and Hogg, D. (1994) “Learning flexible models from image sequences,” in Eklundh, J.-O. (ed.) *Computer vision — ECCV ’94: third european conference on computer vision stockholm, sweden, may 2–6, 1994 proceedings, volume I*. Berlin, Heidelberg: Springer Berlin Heidelberg (Lecture notes in computer science), pp. 297–308. doi: 10.1007/3-540-57956-7_34.

Bhatnagar, B. *et al.* (2019) “Multi-Garment Net: Learning to Dress 3D People From Images,” in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, IEEE, pp. 5419–5429. doi: 10.1109/ICCV.2019.00552.

Black, M. J., Mahmood, N. and Bolkart, T. (2021) *SMPL Made Simple*. Available at: https://smpl-made-simple.is.tue.mpg.de/ (Accessed: April 19, 2022).

Blanz, V. and Vetter, T. (1999) “A morphable model for the synthesis of 3D faces,” in *Proceedings of the 26th annual conference on Computer graphics and interactive techniques - SIGGRAPH ’99*. *the 26th annual conference*, New York, New York, USA: ACM Press, pp. 187–194. doi: 10.1145/311535.311556.

CAESAR Dataset (no date). Available at: https://www.sae.org/standardsdev/tsb/cooperative/caesar.htm (Accessed: April 19, 2022).

Corona, E. *et al.* (2021) “SMPLicit: Topology-aware Generative Model for Clothed People,” in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, pp. 11870–11880. doi: 10.1109/CVPR46437.2021.01170.

Guan, P. *et al.* (2012) “DRAPE,” *ACM Transactions on Graphics (TOG)*, 31(4), pp. 1–10. doi: 10.1145/2185520.2185531.

Guillard, B. *et al.* (2021) “Sketch2Mesh: Reconstructing and Editing 3D Shapes from Sketches,” in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, IEEE, pp. 13003–13012. doi: 10.1109/ICCV48922.2021.01278.

Gundogdu, E. *et al.* (2019) “GarNet: A Two-Stream Network for Fast and Accurate 3D Cloth Draping,” in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, IEEE, pp. 8738–8747. doi: 10.1109/ICCV.2019.00883.

Habermann, M. *et al.* (2019) “LiveCap,” *ACM transactions on graphics*, 38(2), pp. 1–17. doi: 10.1145/3311970.

Habermann, M. *et al.* (2020) “Deepcap: monocular human performance capture using weak supervision,” in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, pp. 5051–5062. doi: 10.1109/CVPR42600.2020.00510.

Hesse, N. *et al.* (2018) “Learning an Infant Body Model from RGB-D Data for Accurate Full Body Motion Analysis,” in Frangi, A. F. et al. (eds.) *Medical Image Computing and Computer Assisted Intervention – MICCAI 2018: 21st International Conference, Granada, Spain, September 16-20, 2018, Proceedings, Part I*. Cham: Springer International Publishing (Lecture notes in computer science), pp. 792–800. doi: 10.1007/978-3-030-00928-1_89.

Jiang, B. *et al.* (2020) “BCNet: Learning Body and Cloth Shape from a Single Image,” in Vedaldi, A. et al. (eds.) *Computer vision – ECCV 2020: 16th european conference, glasgow, UK, august 23–28, 2020, proceedings, part XX*. Cham: Springer International Publishing (Lecture notes in computer science), pp. 18–35. doi: 10.1007/978-3-030-58565-5_2.

Lähner, Z., Cremers, D. and Tung, T. (2018) “Deepwrinkles: accurate and realistic clothing modeling,” in Ferrari, V. et al. (eds.) *Computer Vision – ECCV 2018: 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part IV*. Cham: Springer International Publishing (Lecture notes in computer science), pp. 698–715. doi: 10.1007/978-3-030-01225-0_41.

Lewin, C. (2021) “Swish: neural network cloth simulation on madden NFL 21,” in *ACM SIGGRAPH 2021 Talks*. *SIGGRAPH ’21: Special Interest Group on Computer Graphics and Interactive Techniques Conference*, New York, NY, USA: ACM, pp. 1–2. doi: 10.1145/3450623.3464665.

Li, T. *et al.* (2017) “Learning a model of facial shape and expression from 4D scans,” *ACM transactions on graphics*, 36(6), pp. 1–17. doi: 10.1145/3130800.3130813.

Loper, M. *et al.* (2015) “SMPL,” *ACM transactions on graphics*, 34(6), pp. 1–16. doi: 10.1145/2816795.2818013.

Marr, D. and Nishihara, H. K. (1978) “Representation and recognition of the spatial organization of three-dimensional shapes.,” *Proceedings of the Royal Society of London. Series B, Containing Papers of A Biological Character. Royal Society (Great Britain)*, 200(1140), pp. 269–294. doi: 10.1098/rspb.1978.0020.

Microsoft (no date) *SMPL in Mixed Reality at Microsoft*, *YouTube*. Available at: https://www.youtube.com/watch?v=3ba1jlzuuPc (Accessed: April 19, 2022).

Ng, H. N., Grimsdale, R. L. and Allen, W. G. (1995) “A system for modelling and visualization of cloth material,” *Computers & graphics*, 19(3), pp. 423–430. doi: 10.1016/0097-8493(95)00012-2.

OptiTex (no date) *OptiTex*. Available at: https://optitex.com/ (Accessed: April 19, 2022).

Osman, A. A. A., Bolkart, T. and Black, M. J. (2020) “STAR: sparse trained articulated human body regressor,” in Vedaldi, A. et al. (eds.) *Computer vision – ECCV 2020: 16th european conference, glasgow, UK, august 23–28, 2020, proceedings, part VI*. Cham: Springer International Publishing (Lecture notes in computer science), pp. 598–613. doi: 10.1007/978-3-030-58539-6_36.

Patel, C., Liao, Z. and Pons-Moll, G. (2020) “Tailornet: predicting clothing in 3D as a function of human pose, shape and garment style,” in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, pp. 7363–7373. doi: 10.1109/CVPR42600.2020.00739.

Pavlakos, G. *et al.* (2019) “Expressive body capture: 3D hands, face, and body from a single image,” in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, pp. 10967–10977. doi: 10.1109/CVPR.2019.01123.

Pons-Moll, G. *et al.* (2015) “Dyna: a model of dynamic human shape in motion,” *ACM transactions on graphics*, 34(4), p. 120:1-120:14. doi: 10.1145/2766993.

Pons-Moll, G. *et al.* (2017) “ClothCap,” *ACM transactions on graphics*, 36(4), pp. 1–15. doi: 10.1145/3072959.3073711.

Poux, F. (2021) *How To Represent 3D Data*, *Medium*. Available at: https://towardsdatascience.com/how-to-represent-3d-data-66a0f6376afb (Accessed: April 19, 2022).

Provot, X. (ed.) (1995) “Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behaviour,” in *Proceedings of Graphics Interface ’95*. Quebec, Canada: Canadian Human-Computer Communications Society (GI ’95), pp. 147--154.

Romero, J., Tzionas, D. and Black, M. J. (2017) “Embodied hands,” *ACM transactions on graphics*, 36(6), pp. 1–17. doi: 10.1145/3130800.3130883.

Saito, S. *et al.* (2019) “PIFu: Pixel-Aligned Implicit Function for High-Resolution Clothed Human Digitization,” in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, IEEE, pp. 2304–2314. doi: 10.1109/ICCV.2019.00239.

Santesteban, I., Otaduy, M. A. and Casas, D. (2019) “Learning-based animation of clothing for virtual try-on,” *Computer Graphics Forum*, 38(2), pp. 355–366. doi: 10.1111/cgf.13643.

Schirmer, L. *et al.* (2021) “Neural networks for implicit representations of 3D scenes,” in *2021 34th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*. *2021 34th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, IEEE, pp. 17–24. doi: 10.1109/SIBGRAPI54419.2021.00012.

Tiwari, G. *et al.* (2020) “SIZER: A dataset and model for parsing 3D clothing and learning size

sensitive 3D clothing," in Vedaldi, A. et al. (eds.) *Computer vision – ECCV 2020: 16th european conference, glasgow, UK, august 23–28, 2020, proceedings, part III*. Cham: Springer International Publishing (Lecture notes in computer science), pp. 1–18. doi: 10.1007/978-3-030-58580-8_1.

T, N. and Y, N. (1990) "An expression method of cylindrical cloth objects-an expression of folds of a sleeve using computer graphics," *Trans. of Soc. Of Electronics, Information and Communication, Col. J73-D-II*, pp. 1095--1097.

Vasilakis, A. A. and Fudos, I. (2011) "GPU rigid skinning based on a refined skeletonization method," *Computer animation and virtual worlds*, 22(1), pp. 27–46. doi: 10.1002/cav.382.

Wang, B. *et al.* (2018) "Toward Characteristic-Preserving Image-Based Virtual Try-On Network," in Ferrari, V. et al. (eds.) *Computer Vision – ECCV 2018: 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XIII*. Cham: Springer International Publishing (Lecture notes in computer science), pp. 607–623. doi: 10.1007/978-3-030-01261-8_36.

Wang, T. Y. *et al.* (2018) "Learning a shared shape space for multimodal garment design," *ACM transactions on graphics*, 37(6), pp. 1–13. doi: 10.1145/3272127.3275074.

Weil, J. (1986) "The synthesis of cloth objects," *ACM SIGGRAPH Computer Graphics*, 20(4), pp. 49–54. doi: 10.1145/15886.15891.

Xu, H. *et al.* (2020) "GHUM & GHUML: generative 3D human shape and articulated pose models," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, pp. 6183–6192. doi: 10.1109/CVPR42600.2020.00622.

Zuffi, S. *et al.* (2017) "3D menagerie: modeling the 3D shape and pose of animals," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, pp. 5524–5532. doi: 10.1109/CVPR.2017.586.