Deep Learning for 3D Human Digitisation

Jorge González Escribano, BAI

A Dissertation

Presented to the University of Dublin, Trinity College

in partial fulfilment of the requirements for the degree of

Master of Science in Computer Science (Intelligent Systems)

Supervisor: Aljosa Smolic, Olivier Riviere

August 2021

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

Jorge González Escribano

August 29, 2021

Deep Learning for 3D Human Digitisation

Jorge González Escribano, Master of Science in Computer Science

University of Dublin, Trinity College, 2021

Supervisor: Aljosa Smolic, Olivier Riviere

In this Dissertation, the process of 3D Human Digitisation from single photographs using Deep Learning techniques is studied. A review of the related work in this field is performed and an approach that improves the state of the art techniques is presented, obtaining a working pipeline that yields 3D models that can be directly animated and imported into video games, animations and other content creation projects.

Acknowledgments

I would like to thank my supervisors Professor Aljosa Smolic and Olivier Riviere, as well as Susana Ruano and Amar Arslaan, for their helpful comments and advice, which proved to be invaluable during the development and writing of this dissertation project.

Jorge González Escribano

University of Dublin, Trinity College
August 2021

Contents

List of Figures

Chapter 1

Introduction

In recent years, the latest advances in Artificial Intelligence have enabled new ways for animators, video game developers and content creators to build the assets that they require for their projects. With these new tools at their disposal, they can either create some assets to be directly used in their projects or generate a base model that an artist can use as a source of inspiration. Some examples of artificial intelligence techniques that have been used for this purpose are:

- **Generative Adversarial Networks (GANs) for image and texture generation**: this neural network architecture has gained great notoriety lately as they are capable of generating very high quality and realistic images. This type of neural networks are founded on having two different competing networks, one that learns how to generate realistic images and another one that learns how to spot fake generated images. This way, they form a feedback loop where the generating network continuously improves the quality of its generated output. They could be used, for example, to procedurally generate textures in video games [Fadaeddini et al. (2018), Amato (2017), Summerville et al. (2018)]; see figure 1.1, or to render photorealistic scenarios from sketches [Park et al. (2019)] for an artist to grab inspiration from. One such example of realistic image generation from a sketch is the GauGAN architecture, which is at the core of the tool Nvidia Canvas and achieves incredible results and it is mainly marketed towards artists to serve as a creative reference; see figure 1.3.

- **Deep Learning to build complex animations**: many research papers have been published in the last years that explore novel techniques for character rigging and animation. These would allow for complex and procedural animations to be used in video games [Starke et al. (2019)], making it more feasible for small development

Figure 1.1: Example of textures generated by a GAN that could be used inside a video game

studios with less resources to achieve a final quality closer to that of big animation and video game development companies; see figure 1.2.

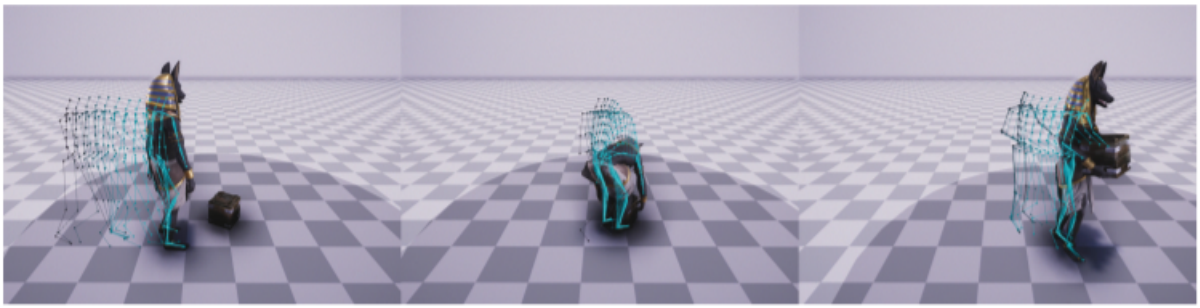

Figure 1.2: Example of a generated animation of a character picking up an object

- **Rendering photorealistic scenarios with Neural Rendering**: this is a very recent field in neural network research that merges artificial intelligence techniques with physical knowledge of the environment [Tewari et al. (2020)]. It explores new ways of creating and rendering photorealistic scenarios more efficiently than with traditional methods, as well as manipulating existing images and videos, maintaining physically correct lighting, perspective, reflections and other light phenomena. This is relevant to the field of content creation as it allows new sources of creative inspiration as well as new tools for animation post-processing and real-time photorealistic video game rendering. This technique is also used by the previously mentioned generative adversarial neural network GauGAN, which allows its generated images to be drawn in different styles representing a a variety of environments and weather conditions; see figure 1.3.

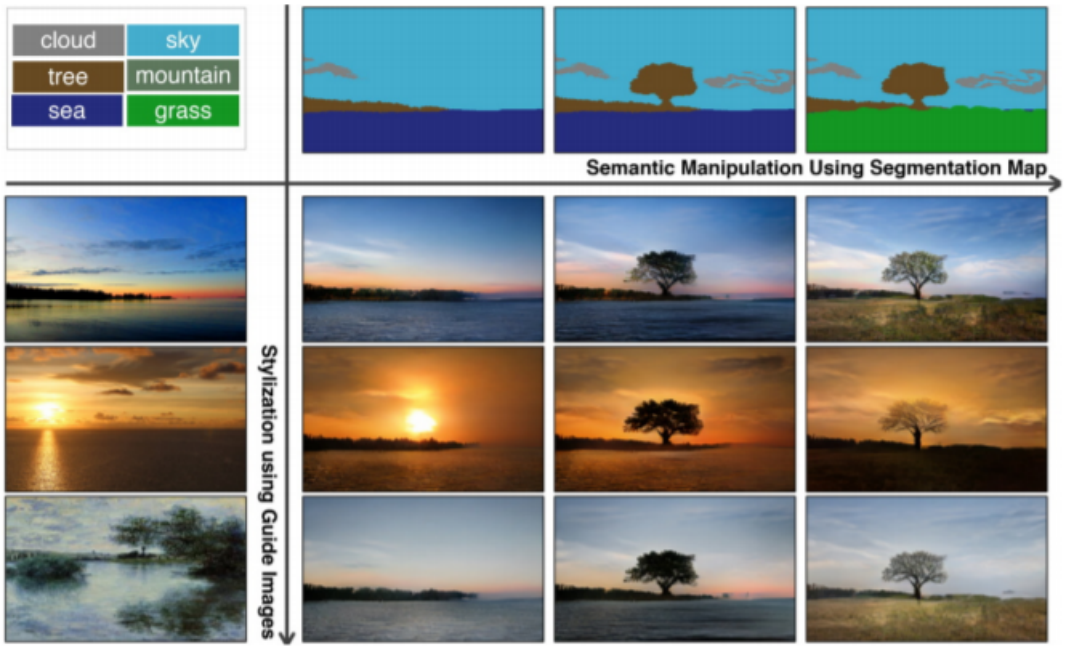

Figure 1.3: Example of the different rendering styles that can be applied to GauGAN

1.1 Main Goal

Following this field of research, this final dissertation project focuses on the creation of a tool that, by leveraging the capabilities of Deep Neural Networks, enables content creators, animators, video game developers and artists of all kinds to generate 3D human models from one single photograph. This would enable smaller and independent studios and creators with lower budgets to quickly prototype and create human model assets, as well as enable new possibilities for applications and video games.

A related research overview will be performed, which will allow to formulate the main goal of this project: improve the state of the art in 3D human model generation in some way making a direct contribution to this field of research and allow for the final result to be directly imported into video games and animations. During the development of the project, a greater scope was achieved as the final working pipeline is capable of producing a rigged 3D model with higher quality texture than current state of the art techniques. These rigged models all share the same bone structure, allowing an artist to create an animation once and use it in all 3D models; see figure 1.4.

As it will be seen in the following section, currently available state of the art tools and neural models are capable of addressing several of the individual goals presented in this project, but there is none capable of addressing them all at once and make their capabilities easily available to users without machine learning knowledge. It is because of

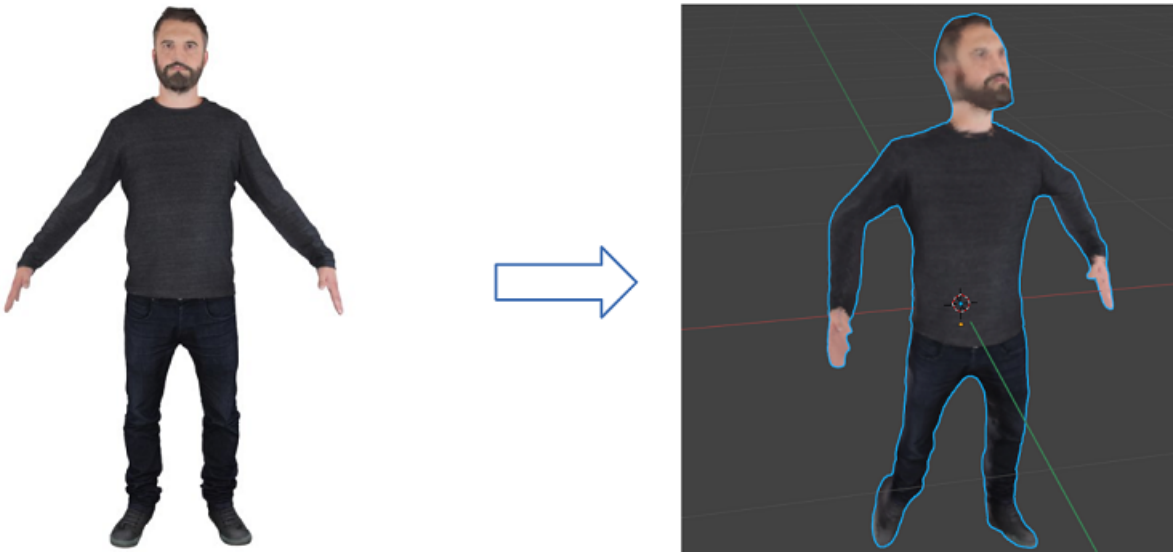

Figure 1.4: Example of what this project tries to achieve. Original input image (left) and 3D rigged model in Blender 3D model edition software (right)

this that this project is proposed, as it would be able to address this problem and make itself accessible to a growing audience of content creators.

Chapter 2

Related Work

In this section, an overview of the related work is performed, while making way towards presenting the state of the art in 3D human digitisation.

2.1 Image to 3D model

Rendering is the process of converting a 3D scene into a 2D image, by following a series of rules that attempt to mimic the way light would interact with the physical objects in such scene if it were to be real. There are multiple well studied techniques to achieve this, but the same can not be said for the inverse process, which has only become available recently thanks to the advances in artificial intelligence. Multiple neural models have been proposed that are able to predict the three-dimensional structure of inanimate objects from single photographs. These neural models focus on the digitisation of common objects such as chairs, cars and aeroplanes.

A good example of these neural models is the architecture "Soft Rasterizer" [Liu et al. (2019a)], which was capable of generating high quality meshes from object pictures; see figure 2.1. One of the properties of traditional rendering techniques is that they are not differentiable, which means that they can not be used in neural networks as they can not be optimized using gradient descent. This paper's main contribution is developing a differentiable rendering process, which outputs color probabilities instead of discrete pixels as traditional renders do. This way, it can be used in a neural network that can be trained to render 3D scenes and then can be inverted to predict 3D scenes from single images.

Reconstructing the 3D mesh from the output generated by a neural network can be a complicated task. Simpler reconstruction approaches, such as voxel based reconstruction, have their quality limited by the size of each voxel and obtain lower resolution meshes.

Figure 2.1: Example of an input image (left) and the output produced by Soft rasterizer (right)

On the other hand, more accurate approaches, such as ray casting based methods, have the drawback of being too computationally demanding. Later the same year of the publication of "Soft Rasterizer", the paper "Learning to Infer Implicit Surfaces without 3D Supervision" [Liu et al. (2019b)] was published, introducing a new way of representing the 3D reconstructed surface by making use of occupancy fields, which are functions that determine the probability of a point in space being inside or outside the solid 3D object, and thus being able to classify which points in space are part of the 3D model. Furthermore, this paper provided a way of applying this technique with unsupervised learning, obtaining higher fidelity reconstructions of the 3D objects depicted in images than the current state of the art; see figure 2.2.

Although these neural models are able to produce high quality geometries from single images, they are very general as they are expected to predict 3D meshes for a wide variety of objects depicted in the input images. For the purposes of this project, it would be interesting to review the state of the art in 3D model generation applied exclusively to human models, as being this specific will probably allow their predictions to be more close to the ground truth.

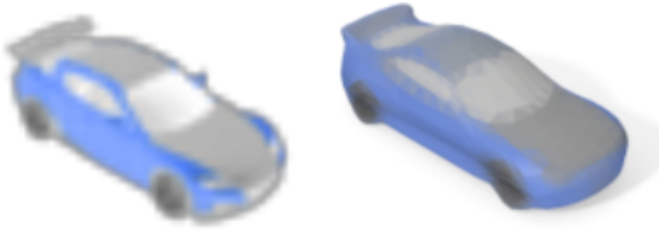

Figure 2.2: Example of an input image (left) and the output produced by the approach presented on the paper Learning To Infer Implicit Surfaces without 3D supervision Output (right)

2.2 Image to 3D model with humans

As stated, the previous models allowed to digitise non animate objects. When it comes to digitizing humans, many options specifically tuned to reconstruct 3D human models exist:

- The first example of human digitisation comes from the neural model "BodyNet" [Varol et al. (2018)], which exhibits an incredible performance when it comes to identifying human pictures and its body parts. This is great for tasks that require image segmentation and has many useful applications, as the information analyzed by this network could be used, for example, to identify the actions being performed by a human in an image. On the other hand, the 3D reconstructions performed by this model are not very precise and can not be used as artistic assets; see figure 2.3.

Figure 2.3: Example of an output produced by BodyNet

- The next year, the architecture "SiCloPe" [Natsume et al. (2019)] was published. This model focuses on digitising clothed people, putting emphasis on the digitization of their outfit. The resulting model more closely matches the figure of the human in the picture, including the shape of the attire and texture, that is reproduced with high fidelity. On the other hand, the person itself does not have such detailed geometry. This can be seen, for example, in the lack of facial features; see figure 2.4.

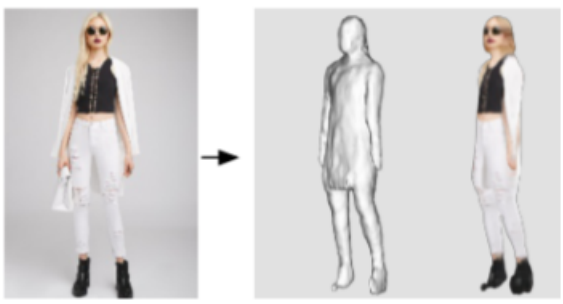

Figure 2.4: Example of an output produced by SiCloPe

- The next approach, called Robust 3D Self-Portraits in seconds [Li et al. (2020)], is able to provide the best results, as it combines a highly defined geometry with a precise texture. The only drawback to this approach is that it requires multiple photographs to be taken of the subject we want to digitise, instead of a single picture; see figure 2.5.

Figure 2.5: Example of an output produced by the approach presented in the paper Robust 3D Self Portraits in Seconds

2.3 State of the Art

This brings us to the state of the art in human digitisation with the papers PIFu [Saito et al. (2019)] and PIFuHD [Saito et al. (2020)], published by Facebook Research Lab. PIFu is the first of them, which introduced the best results in human digitisation while also including high quality textures, thanks to its novel approach called Pixel-Aligned Implicit Function. The next year, PIFuHD was published, which is able to improve the geometry quality even more, but lacks texture.

2.3.1 Pixel-Aligned Implicit Function

Before explaining how the neural model itself works, it is necessary to explain what is the Pixel-Aligned Implicit Function and why it is necessary in order to better understand it.

The Pixel-Aligned Implicit Function defines whether one point along a projected line through the three-dimensional geometry is inside or outside the object represented. This way, we can represent the full 3D geometry if we project a line through the model along the depth axis for every pixel in the image. This allows for a lower memory requirement in order to store the full 3D geometry in the neural model, as classical approaches that rely on space partitioning techniques such as voxels require every single point in space to be stored, meanwhile with PIFu it is only required to store the parameters of the function.

This drop in memory requirements is what allows the model to increase in resolution, yielding better results.

2.3.2 PIFu

Now that the PIFu function is explained, the inner workings of the neural model can be described. The neural model is divided into two sub-networks. On the first one, the input image is fed to an hourglass shaped image encoder which outputs a two dimensional array of PIFu functions. This output is called the occupancy space, and can be used to reconstruct the geometry using an algorithm such as marching cubes. The second sub-network is similar to the first one, but in this case the PIFu function is not used to predict whether one point in space is inside the model or not. Instead, it is used to predict the color of such point in space. This way, we end up with a 3D space in which the color of the model is defined at every point in the space. Now, the vertices recovered from the first sub-network can be used to query for the corresponding color predicted by the second sub-network at that coordinates. The final result is a high-fidelity reconstruction of the person in the input photograph; see figure 2.6.

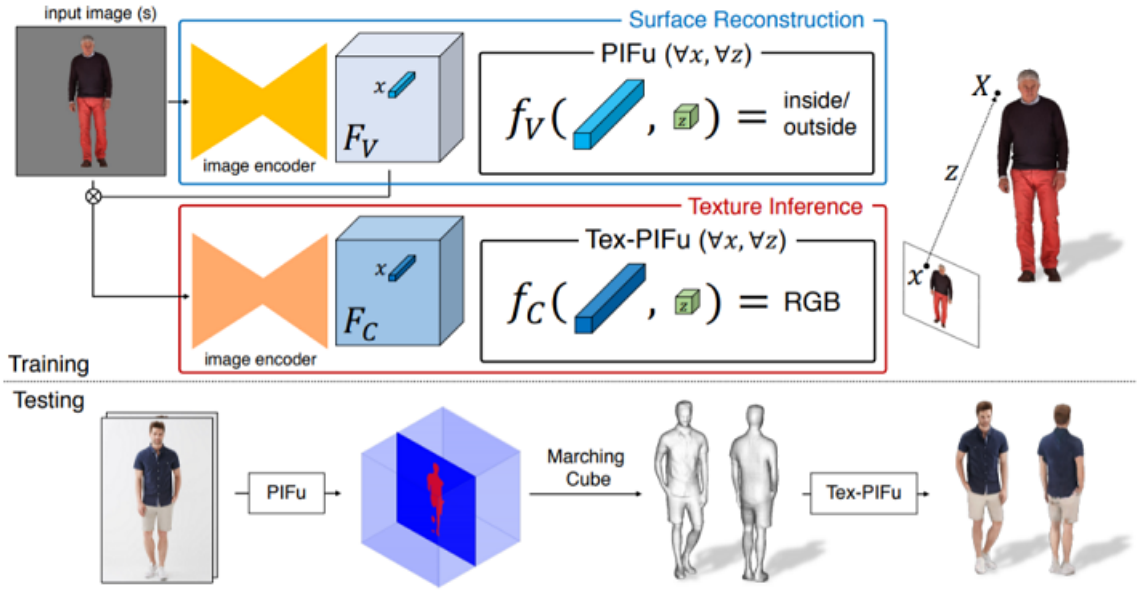

Figure 2.6: Overview of the archicture of the neural model PIFu

2.3.3 PIFuHD

The improved version of PIFu, called PIFuHD, has a very similar architecture to that of its predecessor. It also makes use of the Pixel-Aligned Implicit Function and has two

sub-networks. The first one, called Coarse PIFu, is identical to the first sub-network of the original PIFu architecture, but with more and bigger layers. The result is a slightly higher resolution mesh, but it is not a huge improvement. The memory requirements to store the network starts to become the main constraint to keep growing this sub-network, as it is a fully-connected network. The novelty comes from the second sub-network. First, the original photograph is fed to an image-to-image translation network that estimates the normal map of the texture of the photograph. Then, using the same technique, the normal map of the back-side of the person in the picture is predicted. Now, the resulting normal map is fed to the convolutional version of the first sub-network with higher resolution. This network being convolutional allows to drop the high memory requirement of the fully-connected network, with the disadvantage that when predicting the geometry, only the information in the neighbouring pixels is available. This is not a problem though, as the purpose of this sub-network is to add details to the main geometry, which was generated by the fully-connected network. Lastly, the base model and the detailed geometry are added to obtain a very high-quality mesh, with the drawback being that it lacks texture; see figure 2.7.

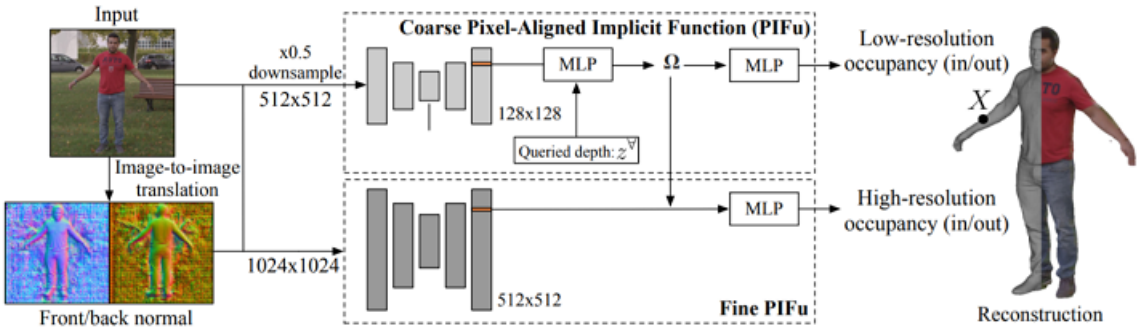

Figure 2.7: Overview of the archicture of the neural model PIFuHD

2.4 Model Rigging and Animation

When it comes to character rigging models, there has not been much research, with the most notable model being "RigNet" [Xu et al. (2020)]; see figure 2.8. This model identifies the articulations in 3D character models and generates an armature that adapts to these articulations, allowing them to be animated. The way it works is by means of a deep graph neural network, which receives the 3D model's mesh geometry as a graph for its input and predicts the probability of a group of vertices forming part of an articulation.

Then, depending on an user controllable parameter called "bandwidth", the algorithm clusters these possible articulations and outputs the resulting skeleton structure. This allows the end user to control the number of articulations in the model indirectly, but not in a precise manner, as the resulting articulation count for a given bandwidth value is also dependant on the geometry of the input model. Even if the correct value for a desired articulation count was found by means of trial and error, there is no guarantee for two human-like bipedal 3D models to be given the exact same bone structure.

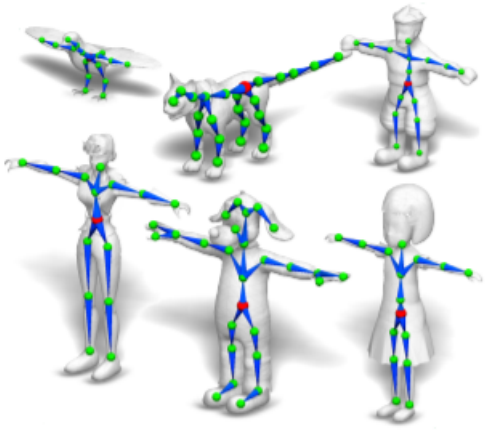

Figure 2.8: Example of RigNet neural model

With regard to animating the rigged models, there are multiple highly specific alternatives based on neural approaches. For example, the paper "Local Motion Phases for Learning Multi-Contact Character Movements" [Starke et al. (2020)] presents a neural framework based on deep learning that helps with animating 3D models performing actions which have multiple contact points with the environment they are interacting with, such as sports like basketball. It does this by using motion capture data as training input and differentiating the local points in the model that change in response to a specific contact. Then, the model allows to mix and blend different animations to generate more complex ones that react to a player input and the character's environment; see figure 2.9.

The paper "Neural Animation Layering for Synthesizing Martial Arts Movements" [Starke et al. (2021)], published one year after the previous paper was published, improves upon its previously proposed framework by producing a new neural layer that helps with controlling and generating the different animations, as well as adapting them to the interaction with a highly dynamic object. This is specially suitable, for example, for martial art character animations, in which the target of one interaction (the opposing player's target) will also be continuously moving and changing, making it necessary to adapt the character's animations to any possible state the opposing character may be in; see figure 2.10.

Figure 2.9: Example of basketball animations generated by the model presented in the paper "Local Motion Phases for Learning Multi-Contact Character Movements"

Lastly, the paper "Mode-Adaptive Neural Networks for Quadruped Motion Control" [Zhang et al. (2018)] present their novel architecture Mode-Adaptive Neural Networks, which enables artists to automatically generate movement animations for quadruped models. This architecture is composed by gated layers that try to predict the position and rotation of the character bones in the next frame depending on their position on the current frame and the user's input. Quadruped characters are way harder to animate than biped characters, so the results of this study can be of special value as they may save a great amount of time to animators; see figure 2.11.

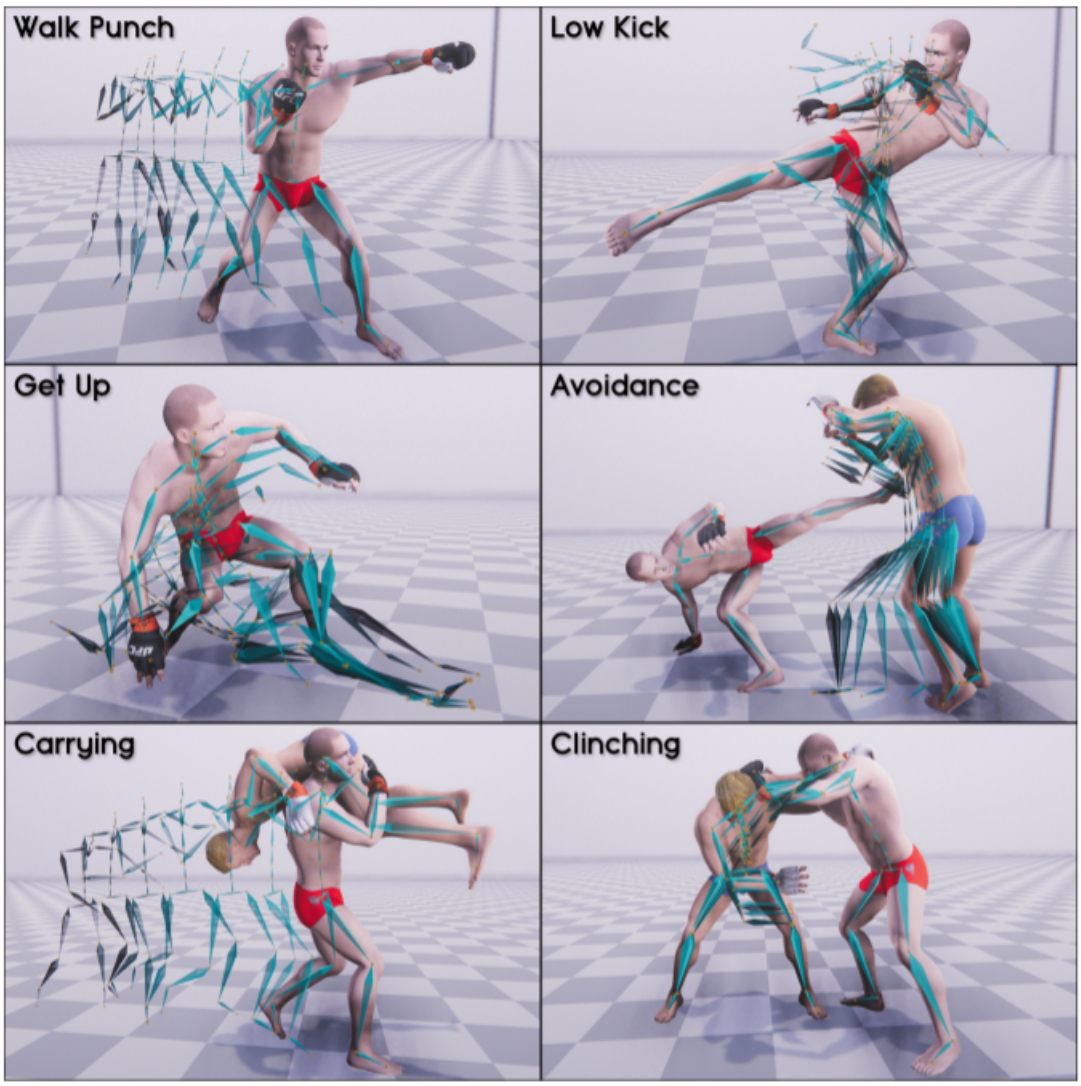

Figure 2.10: Example of martial art animations generated by the model presented in the paper "Neural Animation Layering for Synthesizing Martial Arts Movements"

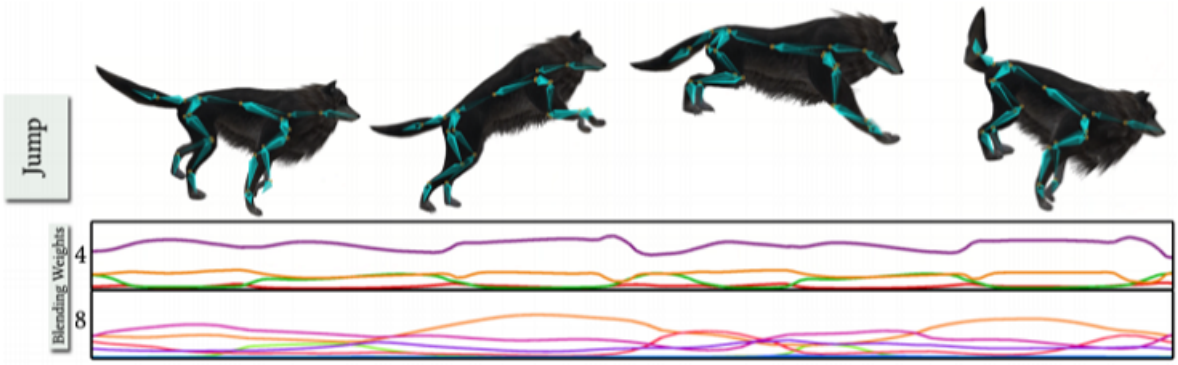

Figure 2.11: Example of a quadruped character animated with the neural model proposed in the paper "Mode-Adaptive Neural Networks for Quadruped Motion Control"

Chapter 3

Developed Project

After performing an investigation of all the related research, the pipeline in figure 3.1 was developed.

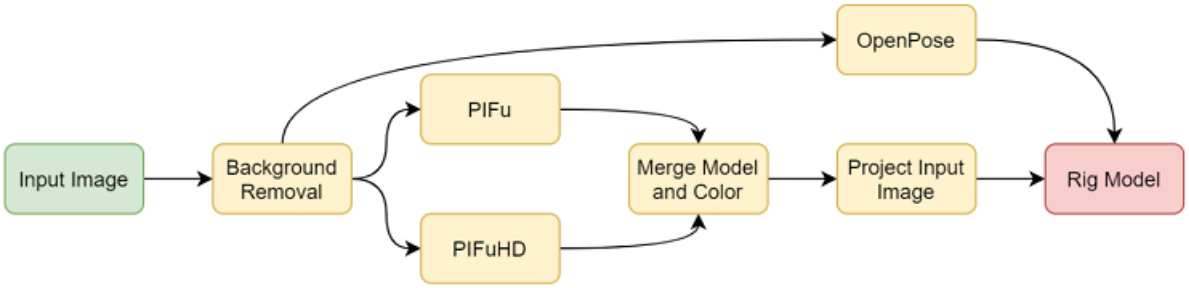

Figure 3.1: Overview of the project's pipeline

First, the background will be removed from the input image, as it is required by PIFu and will help PIFuHD generate a better result. Then, a 3D model is predicted by both PIFu and PIFuHD. The resulting models are merged to obtain a result with high quality geometry and texture. Then, the input image is projected onto the front side of the resulting merged model. Now, this model is rendered from different perspectives and its pose is estimated. The resulting poses will be used to reconstruct the 3D pose and the coordinates of its articulations will serve as starting and ending points for its reconstructed skeleton. This skeleton is imported into blender, along with the previously generated 3D model. Both the skeleton and the model are aligned and the latter is rigged using the Bone Weight algorithm. The output of this last stage will be a fully rigged 3D character that can be animated and imported into any video game development or animation software. The animations have still to be created by hand or by other means, but once they are made for one model, they can be copied and pasted over any other model generated this way as they all share the same skeleton structure.

Now in the following sections each of the steps composing the pipeline will be explained more in depth.

3.1 Model Generation

The first step in the development process was finding a way of merging the highly defined geometry of PIFuHD with the texture generated by PIFu.

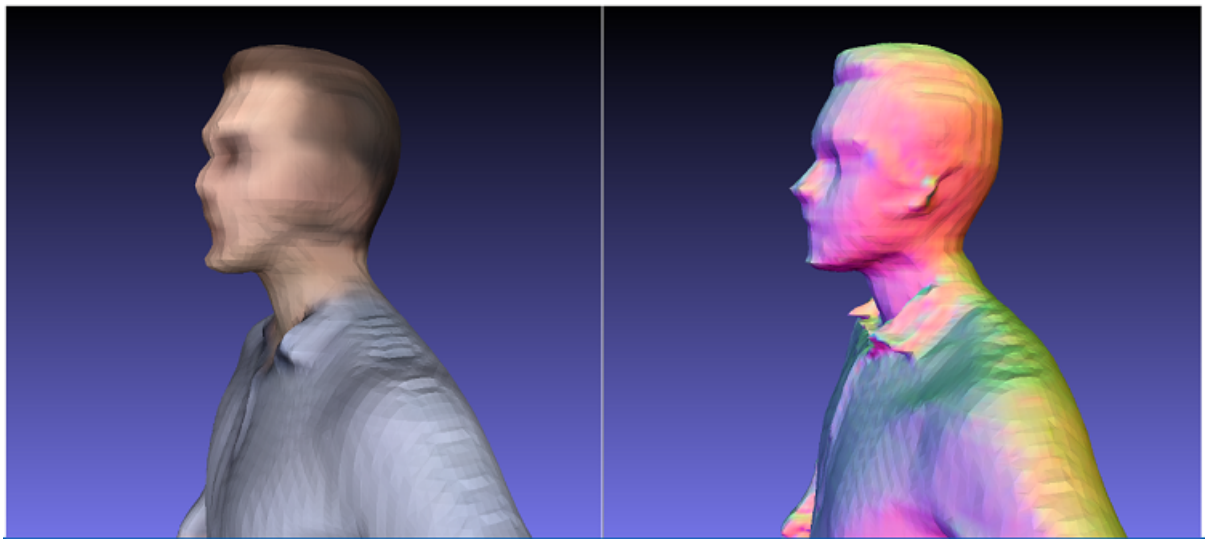

Figure 3.2: Comparison of the models produced by PIFu (left) and PIFuHD (right) of the same image

Figure 3.3: Another comparison of the models produced by PIFu (left) and PIFuHD (right) of the same image

The initial approach consisted of taking the vertices generated by the PIFuHD network and feeding its coordinates to the color sub-network of PIFu instead of the ones generated by it. This led to some strange colors in places where PIFu would not generate geometry, so another approach had to be taken. The selected approach consisted on generating both the PIFu and the PIFuHD 3D models and then for each vertex in the PIFuHD model assign the color of the closest vertex on the PIFu model, this way it was ensured that the coordinates that were queried for the texture would be valid. To achieve this, a kd-tree structure was used to store the vertex coordinates and perform nearest-neighbour finding operation more efficiently. This led to better results, but some of the resulting models had texturing artifacts on the sides; see figure 3.4. An attempt to use the average color of multiple neighbouring vertices was made, but no great quality improvement was found and the idea was dropped.

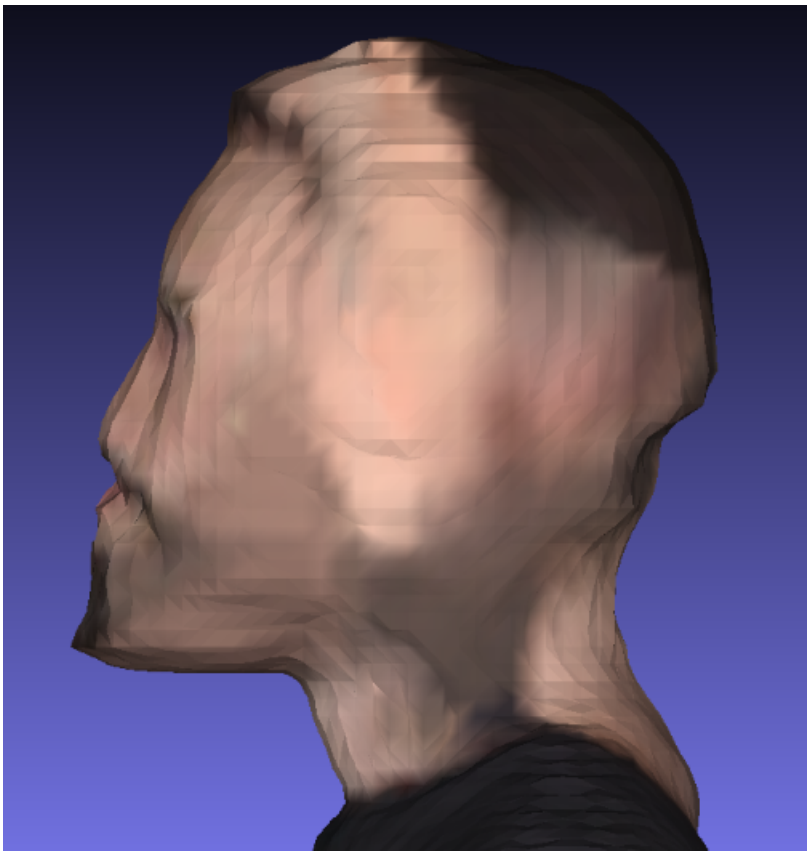

Figure 3.4: Texturing Artifact when merging PIFu texture and PIFuHD geometry

These artifacts were found to be generated due to the fact that both PIFu and PIFuHD predict the geometry of the models they generate to have very different depth as it can be seen in figure 3.5, leading to the closest vertex not belonging to the same body feature.

In order to fix this issue, a way of mapping PIFuHD vertices to the ones in PIFu

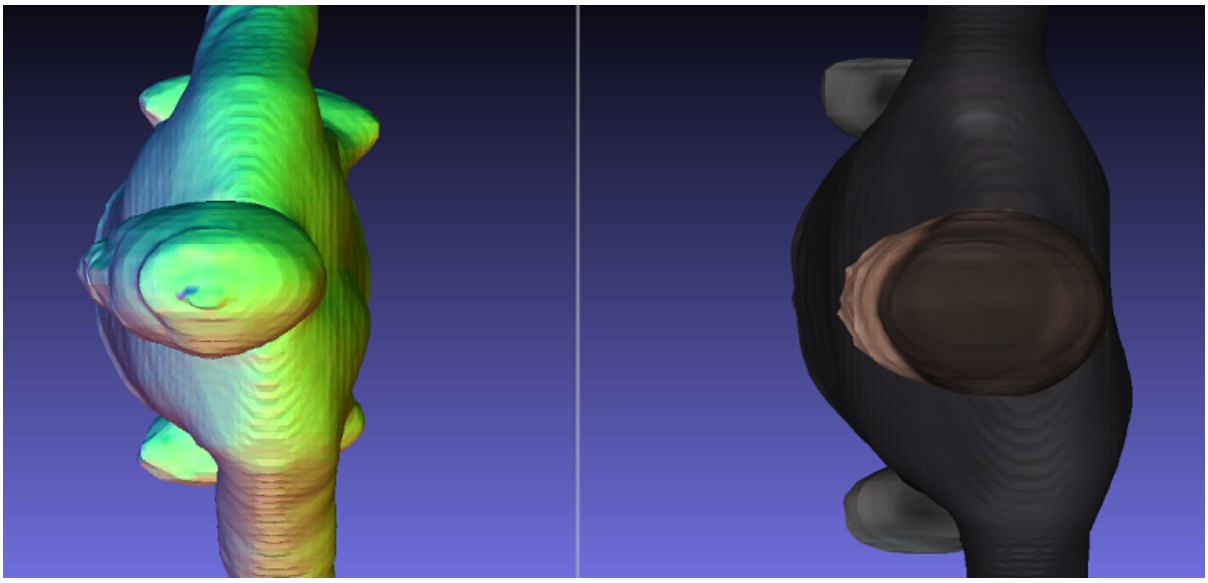

Figure 3.5: Top view of PIFu and PIFuHD models

belonging to the same feature is required. This was achieved by means of model registration, performed with the Python library "probreg" [Kenta-Tanaka et al. (2019)]. Multiple model registration algorithms were tried and their results compared to find the best candidate for this use case, and lastly the Support Vector Registration [Campbell and Petersson (2015)] algorithm was selected to perform non-rigid model registration. This is a computationally intensive task, and in order to be able to process these models with tens of thousands of vertices, the non-rigid registration is performed on a downsampled version of the original model. This lower quality model is enough to retain the main features of the human model, such as the head and the arms, but loses the geometry related with facial features, for example, which is not necessary for the current task.

The following pictures illustrate this process. In figure 3.6, PIFuHD model is drawn in green and the PIFu model is drawn in red. Here it can clearly be seen that the features in both models do not match with each other, specially in the head area. The first step is registering the PIFuHD geometry to that of PIFu with the goal of obtaining a series of parameters that will be used to map the vertices in one model to the other. Then, we apply this transformation to each of the vertices PIFuHD geometry. In the following picture the transformed geometry can be seen in blue along with PIFu's original geometry in red. As it can be seen in figure 3.7, they more closely match together.

Now, if the nearest neighbouring vertex to each of the remapped vertices is found then the texturing artifacts have been solved and a model that combines both high quality geometry and texturing has been achieved, effectively having improved the results over the current state of the art. Figure 3.8 illustrates this improvement. The pseudocode of

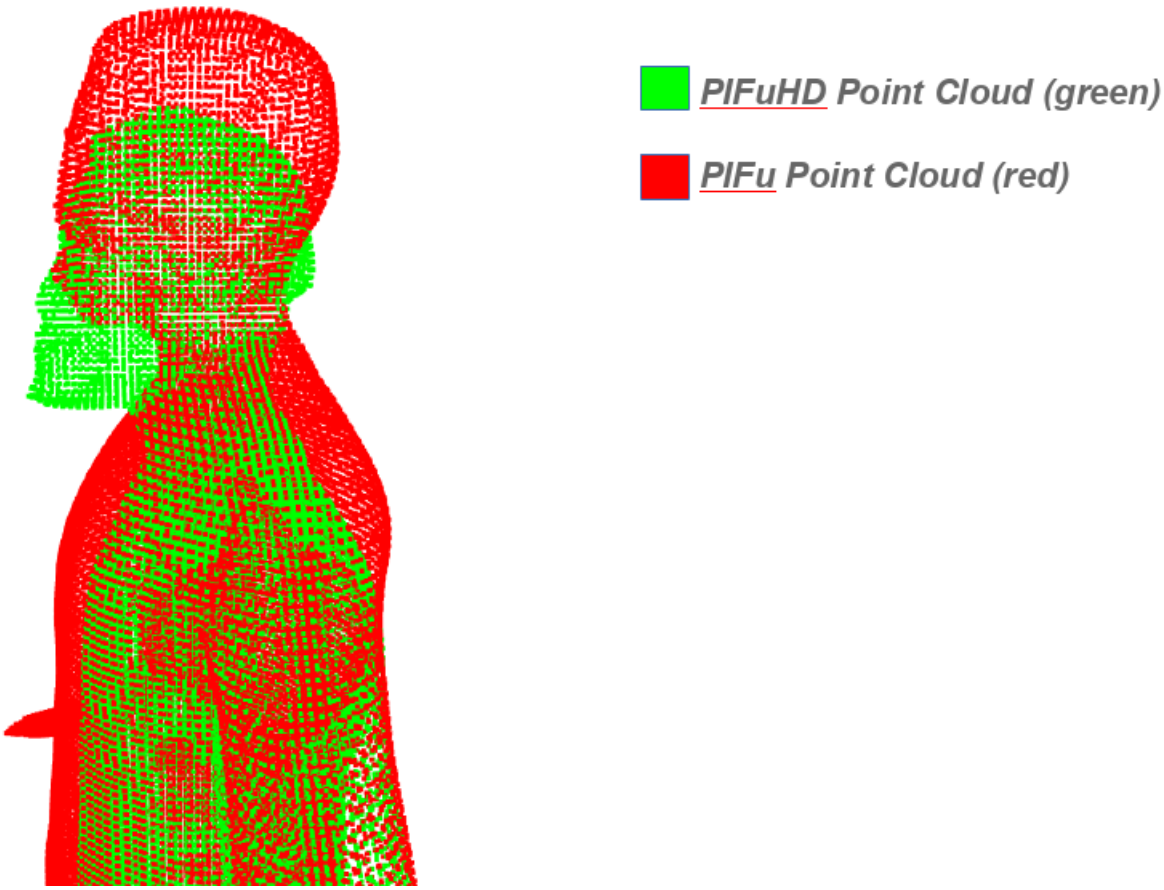

Figure 3.6: Visualization of PIFu and PIFuHD geometry

the algorithm used in this step can be seen in Algorithm 1.

Algorithm 1 Merging of PIFu texture with PIFuHD geometry

```
tree ← Calculate kd-tree from PIFu geometry vertices coordinates
P ← Calculate model registration from PIFuHD to PIFu
for v in PIFuHD geometry do
    v′ ← Apply P to v
    w ← Query tree for nearest neighbour to v′
    Transfer color from w to v
end for
```

The output model of this stage is encoded using Wavefront OBJ format. All the model's information is encoded in plain text, which means that the information contained in the file can be processed and manipulated using any text editor and text manipulation libraries found in every programming language. The use of this formatting enables faster development as less time has to be invested in writing code specific to 3D model file manipulation.

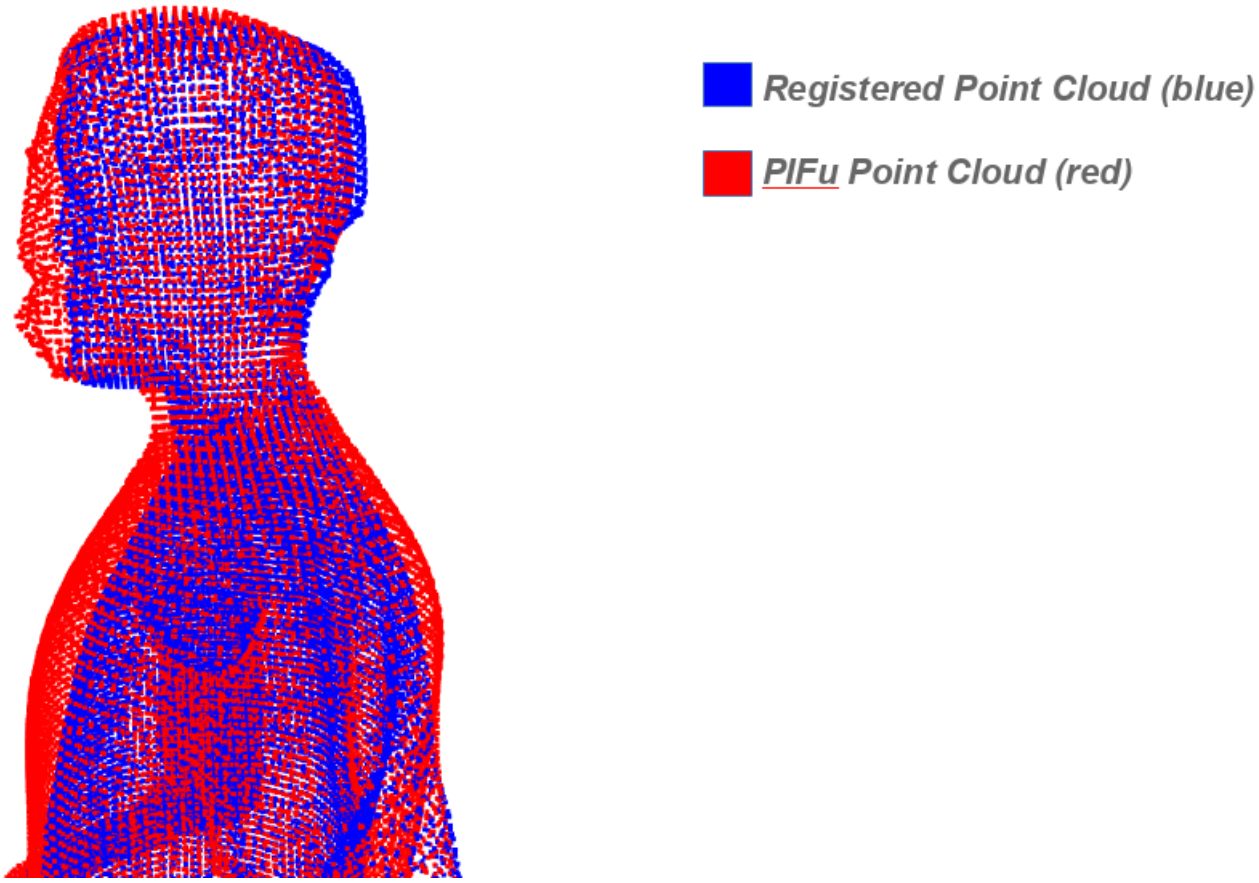

Figure 3.7: Visualization of PIFu and PIFuHD geometry after model registration

3.2 Texture Projection

After making this improvement, a way of further improving the texture quality was developed. The original photograph used as an input to the pipeline contains high quality information about the texture that currently is not being used, and the possibility of projecting it onto the improved model was explored. First, it would be necessary to match both the photograph's features with the model features. To do this, the affine transformation that correlates the most extreme points in both the image and the 3D geometry in the frontal plane was calculated and then applied to the image. Then, the color of each vertex in the model was blended with the color of its corresponding pixel in the image, by using a function dependent on the dot product between the vertex normal and the forward vector. This way, the color transition in the borders of the model would be smoothed, otherwise the boundary between the projection and the PIFu generated color would be too sharp and noticeable. Figures 3.9 and 3.9 show the results.

Currently this computation is performed sequentially on the CPU, but could be con-

Figure 3.8: Merged Model Comparison. Original PIFu model (left), merged models with texturing artifact (center) and merged model with registration (right)

verted to a GPU program with a framework such as CUDA, in order to perform this computation in parallel and greatly speed it up.

Figure 3.9: Visualization of the merged model without projection (left) and with projection (right)

After performing all these steps, the resulting model has already a noticeably better quality than the existing state of the art. In figure 3.11 an example is shown where the effects of these steps are the most meaningful. First, the face geometry can be seen to be much better defined after merging PIFu and PIFuHD models together, while still maintaining a base texture. Then, after projecting the original picture, the patterns in the texture shown on the attire worn by the model can be distinguished, as well as the texturing in the shoes and specially in the face.

Another example worth highlighting is the one shown in figure 3.12, where the 3D model generated originally by PIFu contained many flaws, such as glitched texture generation and weird geometry, such as a wobbly lower leg and floating geometry chunks, which would make this model unusable for most purposes. After running this model through the

Figure 3.10: Another example in which PIFu output (left) is compared with the merged and projected model (right) using an input picture with higher resolution

pipeline, the resulting 3D model was completely suitable for its use in video games and animations, in addition to he quality improvements in facial features after merging with PIFuHD geometry and projecting the input photograph. The texturing glitches caused by PIFu's color generation remained on the sides of the model, but were made less noticeable thanks to the texture projection step.

3.3 Model Rigging

Now that the texture quality has greatly improved over the base model, another way of improving the results was explored. The next step to make a 3D model that could be directly used in animations or video games would be rigging it and animating it, thus a research into the existing methods of automatically rigging was performed. Surprisingly, there are not many approaches making use of neural models that reliably rig and animate 3D models. This may be due to the wide range of different kinds of models that are used in this kind of projects, so it may be difficult to find an approach that can fit them all.

The state of the art in this field is led by "RigNet", a neural network based approach that calculates the armature and rigs articulated models. A look into the research paper reveals a very good performance and high quality results, that could probably be used to save an artist many hours of work. On the other hand, it has a major drawback that would make this approach unsuitable for the purposes of this project, as it yields unpredictable

Figure 3.11: Comparison between PIFu's generated model (left) and the resulting merged and projected model (right)

results on the structure of the armature predicted for each model. As it can be seen in figure 3.13, different human-like geometries yield different skeletons, as the neural model would calculate two or three bones in the arm, for example. This makes this approach inadequate as it would be better to have an homogeneous structure across every armature generated, so it would be easier to animate them.

The second researched approach consisted on leveraging the pose estimation capabilities of the OpenPose [Osokin (2018)] model. It is used to estimate the pose of humans in pictures with high quality results and the returned results always have the same structure, making them predictable and suitable to be automatically animated; see figure 3.14. It has the only problem that it only outputs the predicted skeleton structure as a JSON file instead of a rigged model, making it necessary to have a post processing step to merge the model with the estimated armature. Also, it is used to predict the pose of humans in pictures, so the 3D geometry can not be directly fed to the model.

First, a rendering program was written using the library ModernGL, a high level abstraction layer over OpenGL, that takes the 3D model as an input and outputs a series of renders of the model taken from multiple perspectives with an orthogonal camera. Then, these images are fed to OpenPose. Lastly, the resulting bone coordinates are collected and the transformation applied by the camera when rendering is undone, obtaining the coordinates of each bone's head and tail in the 3D space. An example can be seen in figure 3.15.

Now that the armature of the model is generated, before being able to animate it it is required for the model to be "rigged". Rigging is the process where each vertex of the 3D mesh is assigned to a bone to be moved with it. In more complex models, one single vertex can be assigned to multiple bones and its position calculated as a weighted average, so the animation works as expected in places such as the joints, where the mesh is expected to be flexible. Most of the existing approaches are algorithmical, with the only neural based alternative being "RigNet", which was deemed unsuitable for the current project. The first approach, called Pinoccio [Baran and Popović (2007)], is an algorithmic approach first published in 2007. The second option would be to use the "Bone Heat" algorithm used by Blender's automatic bone weighting function. There is not much information about this algorithm, but it is presumably based on the Pinoccio algorithm. Lastly, the algorithm "Bone Glow" [Wareham and Lasenby (2008)] exists, which claims to yield better results than the Bone Heat algorithm, specially in edge cases.

From all of them, the Blender approach was chosen as it was deemed the most convenient. It is one of the most used 3D modeling softwares and as such has a big community that could be used for support if required. It also has a Python 3 console that would give the possibility of applying some transformations to both the model and the skeleton if required.

The raw JSON output from OpenPose is processed and the information about the position and perspective of the render cameras is used to reconstruct the model's estimated skeleton in 3D space. Using a Blender Python script an armature is constructed using OpenPose pose information. The 3D model from the previous stage is imported into Blender. The current version of this tool does not keep the texture stored as color per vertex in Wavefront OBJ files, so as a workaround, the OBJ model is first imported into MeshLab [Cignoni et al. (2008)], another 3D model manipulation software, using its Python library "pymeshlab" [Muntoni and Cignoni (2021)], and then is exported as PLY (Standford Triangle Format) format. This converted file can now be imported into Blender and keep its color information. Then both are aligned vertically and horizontally by calculating the affine transform that maps the most extreme points of both objects and scaling down slightly the armature from its geometrical center. Then, its depth is

also aligned. Lastly, the Bone Weight algorithm is used to rig the model. There is a glitch with the latest versions of Blender where the algorithm may fail in models in which some vertices are too close together. The way to overcome this is scaling up both the armature and the model's mesh, applying the rigging algorithm and then scaling them back down. Then the resulting textured and rigged model is saved as a Blender scene file ready to be imported into any video game or modeling and animation software.

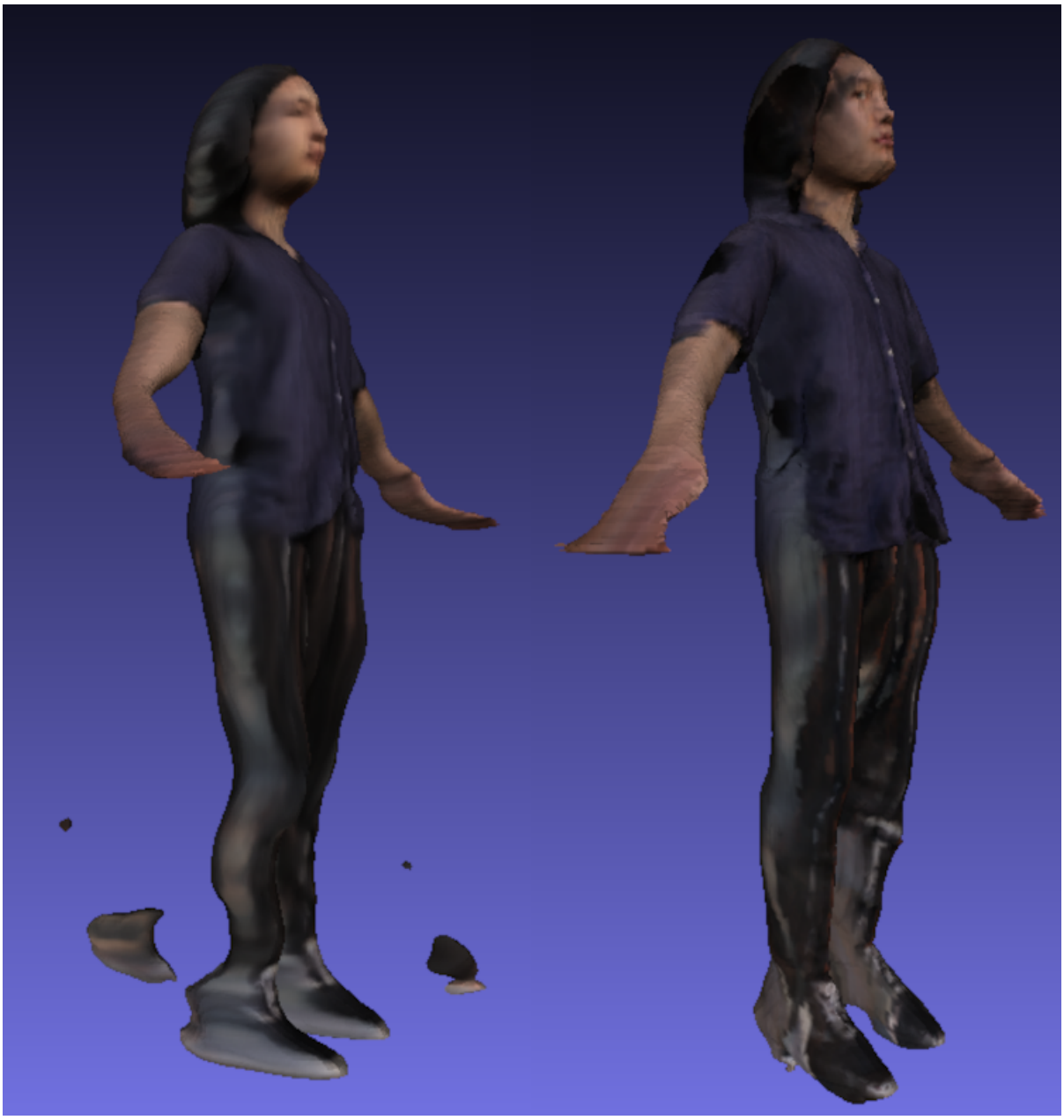

Figure 3.12: Comparison between another PIFu's generated model (left) and the resulting merged and projected model (right). The PIFu generated model can be seen showing many flaws, which were later fixed by the pipeline.

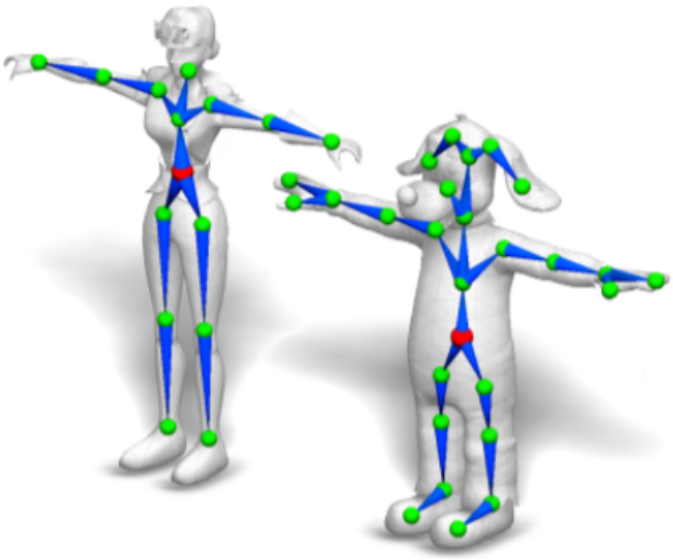

Figure 3.13: RigNet Output Example

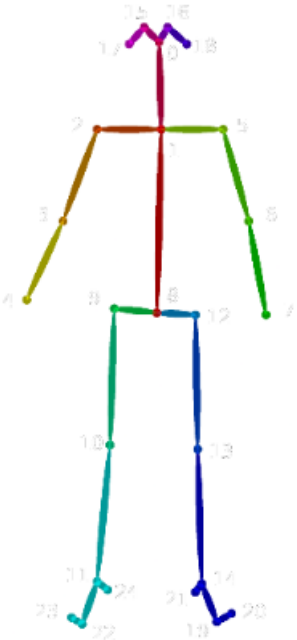

Figure 3.14: Structure of a pose estimated by OpenPose

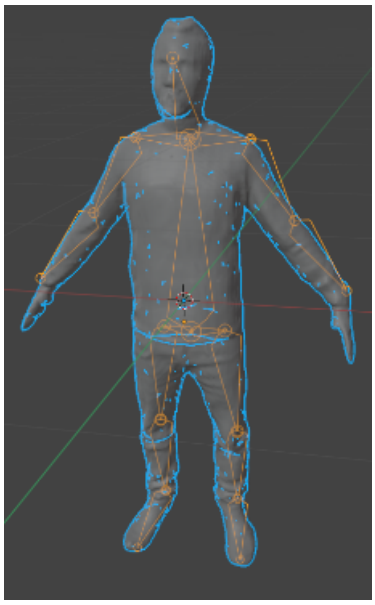

Figure 3.15: Generated model with skeleton

Chapter 4

Results

In the pictures seen in figure 4.1 some of the resulting models are shown as they are output from the pipeline. The original images are animated and these are some frames extracted from them. They can be seen as part of a "walking" animation which was created for one single model and copied to all the others, which is possible thanks to all of them sharing the same armature structure. It can be seen that the results are pretty good, with the geometry showing distinctive features, a good texture applied to them and being reasonably animated. In some of them some features can be clearly distinguished, such as the button of the shirt or the belt of the trousers.

Overall, the quality of the generated geometry is pretty good. The hands may be very simplified depending on the posture the model was holding them in the original image: if the palms of the hands were to be looking towards the camera, it would be possible for PIFuHD to generate the finger geometry, but as in these examples they were perpendicular to the camera, the geometry of the fingers is way too complex along the depth axis for a single PIFu ray to define.

There are specially important parts of the geometry, like the facial features and hair, that is generated with a high level of quality. On top of the highly defined facial features, the picture projection improves even more the quality of the 3D model when facing the camera. Figure 4.2 is an example were the generated 3D model shows a highly detailed hair geometry which shows volume around the face of the model, while still achieving a well defined texture even on the interior face of the hair geometry.

The generated geometry is not perfect though, as there are some generated artifacts that can be observed in these models. For example, in this model it can be seen that the geometry of the shoes is not well defined, being a mixture between a sports shoe and a high-heel. This may be because in the original photograph, the shoes can be seen tilted forward, either by the perspective of the camera or the pose of the model. In either

Figure 4.1: Example output

case, the neural model seems to have learned that a downward looking shoe is probably a high heel. Another example where this phenomenon can be seen is in figure 4.3, where geometric features such as the knees and the folding of the shorts are correctly generated and the texturing of the shorts is also well defined, but PIFuHD failed to correctly generate the 3D mesh of the shoes, which probably should be more similar to running shoes instead of high heels.

In these other examples seen in figure 4.4, it can be clearly seen that the high heels the models are wearing are more accurate. On the other hand, there is a small issue with the texture these models display on the back side. On the front side they can be seen wearing colorful and detailed clothing, but the coloring generated by PIFu seems to have lost these details. It is, as stated, a problem that only happens with models wearing an attire in which the occluded side has a complex pattern of coloring. This also happen with objects such as belts, that may fade when rotating from the front side of the model towards the back side. Other high frequency texturing features but that only occur on the front side of the attire, such as shirt buttons, do not suffer this problem. This is probably the most noticeable issue with the generated models, and possible workarounds are proposed on the "future work" section.

The model texturing is, in fact, the source of other issues in some edge cases, such as

in this one. As it was explained before, PIFuHD generates a normal map texture from the input picture in order to improve the quality of the generated geometry. The neural model tasked with the normal map texture generation seems to have associated a sharp transition between a bright color and black color as probably being a shadow casted by the clothing due to a folding. This is usually true and provides good results in features such as shirt necks and foldings, but in a shirt with a texture black lines forming white squares give rise to the following artifacts. As it can be seen in figure 4.5, the model shows a strange rugged texture.

As described during the PIFu architecture introduction, an image to image translation neural model is used to infer the back side normal map from the front side normal map. Because of this, any glitch on the front normal texture will leak into the back normal texture, as it is the case with this example. Due to the irregular front normal map caused by the checker pattern texture, the artifacts that propagate to the back side normal map end up causing more geometry issues that can be noted in the previously mentioned figure.

Another attire texturing pattern where this phenomenon has been observed is in black and white orizontal stripes, such as in figure 4.6. In this case though, the deformation caused by the normal map artifacts appear to be constrained to a specific part of the 3D model in both the front and the back face and does not propagate to other parts of the model. Other complex mesh features, such as the bow tie in the trousers, appear to have been generated with high quality and without noticeable artifacts.

The normal map artifact propagation to the back side issue can be fixed by estimating the back normal map from a projection of the PIFu model with its predicted texture applied. As the back side texture tends to lose these texture details, the resulting image would have a more uniform texture similar to that seen in the previous examples, leading to a more accurate geometry. On the other hand, this would only fix a small part of the issue, as the front side normal map prediction issues would prevail. The only possible solution would be training the image to image translation neural network with more examples, so it avoids generating these kinds of roughed geometry patterns.

Still, not every model dressed with a pattern that could potentially generate glitched normal map ends up being unusable. For example, in figure figure 4.7, the jumper worn by the model shows black and white squares, which cause the neural model to generate geometry on the normal map texture around the square seams. This can be better noticed on the PIFuHD model, as its texture corresponds to the 3D model's normal map. On the other hand, on the finished model, the geometry generated around this texture is barely noticeable.

Figure 4.2: Example with high quality generated hair geometry while also maintaining high fidelity texture

Figure 4.3: Example of a 3D model where the shoes were mistaken with high heels

Figure 4.4: Example of an output generated of images where the models were wearing an attire with higher detail

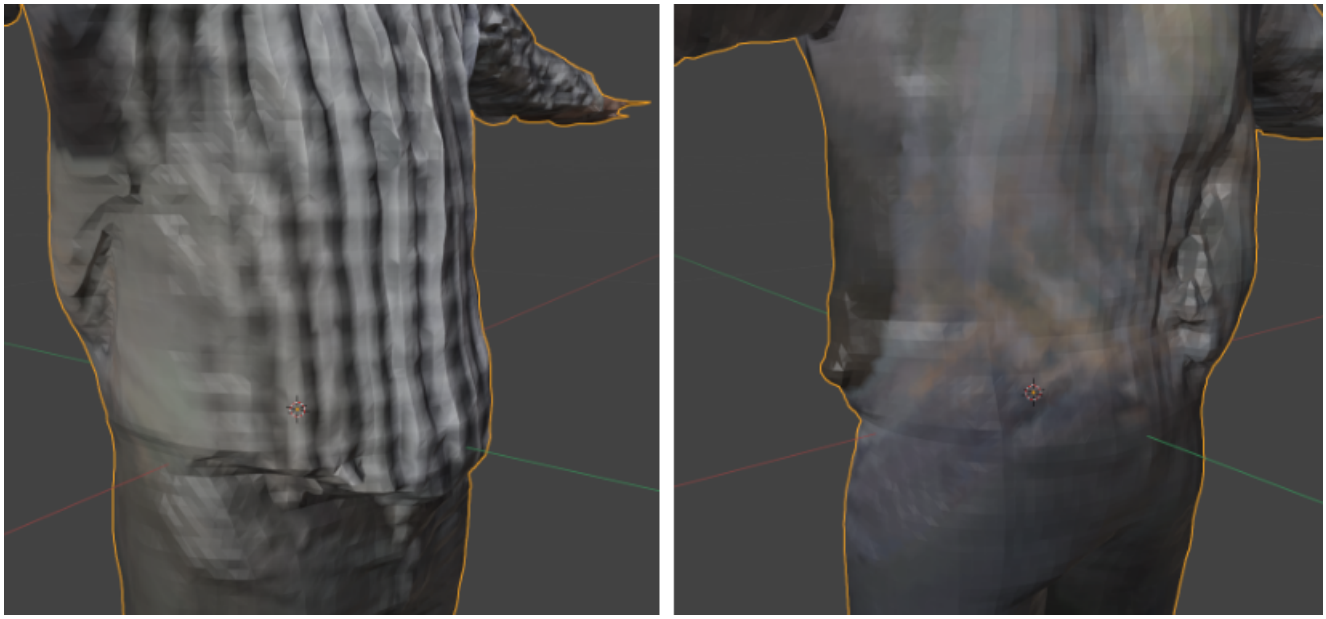

Figure 4.5: Example of geometry artifact due to normal mapping

Figure 4.6: Another example of geometry artifact due to normal mapping, where the rest of the model remains unaffected. Input image (left), resulting model front face (upper-right) and resulting model back face (bottom-right)

Figure 4.7: An example where a potentially glitch-causing clothes texture does not cause any disruptive geometry. Input texture (top), PIFuHD generated model (bottom-left) and final model (bottom-right)

Chapter 5

Conclusions & Further Work

Overall, I would say that the project was a success, as I was able to achieve every goal and was capable of going even further and developing more improvements for the digitisation pipeline that makes the models available for use as assets in video games and animations. Almost every model can be used at least as a reference for artists to create the final asset and many of them can be used directly with minor modifications or no modifications at all. It is worth to note that, even though the image projection step seems only like a minor improvement of the final quality of the model, I believe that it can have a huge impact on the final perceived result. In the environments that these models are aimed to be used in, that is video games and animations, it is expected for the human characters appearing in them to be mostly viewed from the front side. For example, when interacting with a human character in a video game, or watching someone talk in an animation, they will be mostly showing the front side as they will be looking at the camera or player character, that is, they will be showing the side in which the texture resolution is the highest. When the human character is not in the main view there will be times when the other sides with lesser texture quality will be shown the most, but as stated previously, they will not be on the main view and less attention will be paid to them, making the lower quality texture less noticeable.

5.1 Example Applications

The main application of this project would be creating human models for their use in animations and video games. For example:

- It could be used for users to create their own personal avatar in social style video games, where players could use avatars that more closely resemble themselves. This would be enabled by the 3D avatar being created with a single photograph that

could be directly taken in front of the mirror by the player, as other approaches that would require more cumbersome processes for the automatic 3D model to be generated, such as asking another person to capture a video around the player and using photogrammetry to generate a model with higher resolution, could make potential players lose their interest in the game due to the added work required in order to try it.

- It could also be used in small toy-like smartphone applications. For example, you could take a picture of a friend standing next to you, point the smartphone's camera to the table and have a small figurine of them dancing on the table through augmented reality. Again, the ease and convenience of creating the 3D model with just taking one single picture of the subject is what could enable these kinds of applications to be successful.

Computationally speaking, going through the complete pipeline is an expensive process, especially for the 3D geometry generation process as PIFuHD has a high memory requirement. Also, the computational requirement could be a barrier for it to be run in mobile devices. On the other hand, modern smartphones are beginning to incorporate artificial intelligence acceleration modules, such as the latest Pixel phone by Google, which will come with their tensor computing units. Also, there is the possibility of performing all the computationally intense operations in the cloud and returning the processed model to the user, as sending the input image and downloading the resulting model is expected to have a networking overhead of only around 20 Megabytes, most of which comes from the currently uncompressed model and could be improved by using compression algorithms, which is very little for most of current internet connections.

These applications were mainly targeted towards a non-technical market and providing entertainment value, but the potential of this project does not stop here, as it could also be shaped to be used as a professional 3D modeling tool. One single picture could be taken of a human model wearing the desired attire for an animation or video game, and the resulting 3D model would just be required to be slightly adjusted to remove any imperfection or add more detail. This would help an artist by reducing the time required to model the complete attire from the ground up and texturing it. Using the raw output model also has its applications as it could be directly used first as a prototype and then refined for the final product.

5.2 Short-term Improvements

There are still ideas being worked on that can improve the project in the short term. More precisely:

- As explained before, the generated models use the pose information generated by OpenPose. This means that posing information could be directly captured from a camera and animate the generated model in real time. Furthermore, creating the animations is currently the only step that would be required to be performed manually by an artist, but there is the possibility of animating these models by capturing someone performing the animations using a camera and porting them to the models armature, which should be a simple task as both would share the same estimated pose structure generated by OpenPose. This drops the requirement of having a green room full of cameras and an animation model wearing a motion capture suit, enabling smaller studios and content creators to capture realistic animations with just two smartphone cameras.

- For simplicity of development, the model created with the current pipeline uses a technique called "color per vertex" in order to store the texture of the models. This means that the color of the model is only defined at each vertex and that it is interpolated for displaying the pixels in between, meanwhile with a traditional texturing approach the color could be defined at every pixel if the texture used has high enough resolution. This means that currently the texture storage methodology acts as a bottleneck limiting the quality potential of the projected texture. As an example, the following model has been upsampled by subdividing the mesh, this way the model is defined by more vertices and the stored texture will have better quality; see figure 5.1.

As it can be seen in figure 5.1, the projected quality has greatly improved and is now limited only by the resolution of the image taken. Finer details can be seen, up to the point where the texture of the jeans worn by the model can be distinguished.

5.3 Long-term Improvements

There are also many ideas that can be used to further improve the quality of the generated models on the long term:

- The way PIFu chooses to generate the model's texture is by predicting the color of its vertices at every point in the full 3D space. I think that the texture could be better

generated by using a technique similar to what PIFuHD uses, that is generating the back side view by using an image to image translation neural network, but instead of predicting the normal mapping, it should predict the color texture. This could be achieved by using, as stated, an image to image translation network, but this problem could also be a candidate for a Generative Adversarial Network to solve. I believe that with this method a higher resolution view of the back of the model could be generated and then projected, similarly to what is currently done with the front view, yielding higher quality results.

- Also, the geometry generation could be improved, as there still exist some issues with the mesh generation. The only way to solve this is by using a bigger dataset on the training phase or by designing an architecture that is able to obtain better results.

- Lastly, an algorithmic approach has been used to rig and animate the models. An alternative could be explored that uses a neural approach, which may be able to achieve more flexible results than the current ones. It would be especially interesting to design a neural approach capable of rigging and animating models with a skeleton following the structure used by the OpenPose tool, as it is widely used and also it would be fairly easy to obtain a large enough dataset for it to be trained on.

Figure 5.1: Example of amodel with higher resolution texture

Bibliography

Amato, A. (2017). *Procedural Content Generation in the Game Industry*, pages 15–25. Springer International Publishing, Cham.

Baran, I. and Popović, J. (2007). Automatic rigging and animation of 3d characters. In *ACM SIGGRAPH 2007 Papers*, SIGGRAPH '07, page 72–es, New York, NY, USA. Association for Computing Machinery.

Campbell, D. and Petersson, L. (2015). An adaptive data representation for robust point-set registration and merging. *2015 IEEE International Conference on Computer Vision (ICCV)*.

Cignoni, P., Callieri, M., Corsini, M., Dellepiane, M., Ganovelli, F., and Ranzuglia, G. (2008). MeshLab: an Open-Source Mesh Processing Tool. In Scarano, V., Chiara, R. D., and Erra, U., editors, *Eurographics Italian Chapter Conference*. The Eurographics Association.

Fadaeddini, A., Majidi, B., and Eshghi, M. (2018). A case study of generative adversarial networks for procedural synthesis of original textures in video games. In *2018 2nd National and 1st International Digital Games Research Conference: Trends, Technologies, and Applications (DGRC)*, pages 118–122.

Kenta-Tanaka et al. (2019). probreg.

Li, Z., Yu, T., Pan, C., Zheng, Z., and Liu, Y. (2020). Robust 3d self-portraits in seconds. *CoRR*, abs/2004.02460.

Liu, S., Li, T., Chen, W., and Li, H. (2019a). Soft rasterizer: A differentiable renderer for image-based 3d reasoning.

Liu, S., Saito, S., Chen, W., and Li, H. (2019b). Learning to infer implicit surfaces without 3d supervision.

Muntoni, A. and Cignoni, P. (2021). PyMeshLab.

Natsume, R., Saito, S., Huang, Z., Chen, W., Ma, C., Li, H., and Morishima, S. (2019). Siclope: Silhouette-based clothed people. *CoRR*, abs/1901.00049.

Osokin, D. (2018). Real-time 2d multi-person pose estimation on cpu: Lightweight openpose.

Park, T., Liu, M., Wang, T., and Zhu, J. (2019). Semantic image synthesis with spatially-adaptive normalization. *CoRR*, abs/1903.07291.

Saito, S., Huang, Z., Natsume, R., Morishima, S., Kanazawa, A., and Li, H. (2019). Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization.

Saito, S., Simon, T., Saragih, J., and Joo, H. (2020). Pifuhd: Multi-level pixel-aligned implicit function for high-resolution 3d human digitization.

Starke, S., Zhang, H., Komura, T., and Saito, J. (2019). Neural state machine for character-scene interactions. *ACM Trans. Graph.*, 38(6).

Starke, S., Zhao, Y., Komura, T., and Zaman, K. (2020). Local motion phases for learning multi-contact character movements. *ACM Trans. Graph.*, 39(4).

Starke, S., Zhao, Y., Zinno, F., and Komura, T. (2021). Neural animation layering for synthesizing martial arts movements. *ACM Transactions on Graphics*, 40:1–16.

Summerville, A., Snodgrass, S., Guzdial, M., Holmgård, C., Hoover, A., Isaksen, A., Nealen, A., and Togelius, J. (2018). Procedural content generation via machine learning (pcgml). *IEEE Transactions on Games*, 10(3):257–270. Publisher Copyright: © 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. Copyright: Copyright 2020 Elsevier B.V., All rights reserved.

Tewari, A., Fried, O., Thies, J., Sitzmann, V., Lombardi, S., Sunkavalli, K., Martin-Brualla, R., Simon, T., Saragih, J. M., Nießner, M., Pandey, R., Fanello, S. R., Wetzstein, G., Zhu, J., Theobalt, C., Agrawala, M., Shechtman, E., Goldman, D. B., and Zollhöfer, M. (2020). State of the art on neural rendering. *CoRR*, abs/2004.03805.

Varol, G., Ceylan, D., Russell, B. C., Yang, J., Yumer, E., Laptev, I., and Schmid, C. (2018). Bodynet: Volumetric inference of 3d human body shapes. *CoRR*, abs/1804.04875.

Wareham, R. and Lasenby, J. (2008). Bone glow: An improved method for the assignment of weights for mesh deformation. In Perales, F. J. and Fisher, R. B., editors, *Articulated Motion and Deformable Objects*, pages 63–71, Berlin, Heidelberg. Springer Berlin Heidelberg.

Xu, Z., Zhou, Y., Kalogerakis, E., Landreth, C., and Singh, K. (2020). Rignet: Neural rigging for articulated characters.

Zhang, H., Starke, S., Komura, T., and Saito, J. (2018). Mode-adaptive neural networks for quadruped motion control. *ACM Trans. Graph.*, 37(4).