VIDEO ANIMATION OF PEOPLE FROM RGBD DATA

Cathrine J. Thomsen

AALBORG UNIVERSITY
DENMARK

Master's Thesis
VGIS10
16gr1048
Aalborg University
June 2016

Vision Graphics and Interactive Systems
Aalborg University
http://www.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Video Animation of People from RGBD Data

Theme:
Computer Vision

Project Period:
Fall semester 2015 and spring semester 2016

Project Group:
16gr1048

Participant(s):
Cathrine Juel Thomsen

Supervisor(s):
Thomas B. Moeslund, Aalborg University
Adrian Hilton, University of Surrey

Copies: 2

Page Numbers: 91

Date of Completion:
June 2, 2016

Abstract:

This work is an investigation in a low cost solution for performing video animation using a Kinect v2 for Windows, where skeleton, depth and colour data are acquired for three different characters.
Segmentation of colour and depth frames was based on establishing person's range in the depth frame using the skeleton information, and then train a plane of the floor and exclude points close to it.
Transitioning between motions were based on minimizing the L2 distance between all feasible transitioning frames, where a source and target frame would be found. Intermediate frames were made to create seamless transitions, where new poses were found by moving pixels in the direction of the optical flow between the transitioning frames.
An interactive animation was made, where the motions of three different characters can be controlled by a user.
The realism of the suggested animation was verified through a user study to have a higher rate of preference and perceived realism compared to no animation and animation using alpha blending. The results from the user study also showed that there is still room for improvement, since the number of intermediate frames should be determined adaptively according to the similarity measure and the speed of motion before and after transitioning.

Contents

Preface

This report outlines the topic of video animation of people exploiting methods for creating seamless transitions between motions using captured sequences by a consumer camera. This project is a long master's thesis on the master's programme Vision, Graphics and Interactive Systems at Aalborg University. The thesis has been made in cooperation with the University of Surrey in the United Kingdom, which was visited by the author from November 2015 to June 2016.

The author would like to thank Prof Adrian Hilton from the University of Surrey and Prof Thomas B. Moeslund from Aalborg University for supervising throughout the past year. Furthermore, the author would like to thank Dr Marco Volino and Mr Leonardo Ribeiro for being willing to take part of the captured datasets.

The code is implemented in Python 2.7.6 and uses the following modules:

- MatPlotLib 1.3.1
- NumPy 1.8.2
- OpenCV 2.4.10
- SciPy 0.15.1

Aalborg University, June 2, 2016

Cathrine Juel Thomsen
cjth11@student.aau.dk

1. *Introduction*

Animating people using a maker-based motion capture have been widely used and is perhaps most known for the Gollum character in *The Lord of the Rings*, where the actor wears a tight suit with reflective markers as shown in Figure 1.1. Tracking each of these markers in a multiple view studio, when the actor performs different motions, will then be used for controlling the joints of an animated character performing the same motions.

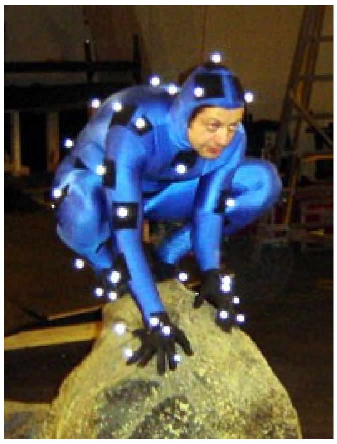


Figure 1.1: Andy Serkis performing as Gollum in *The Lord of the Rings* (Serkis).

Not only does the marker-based approach require a significant setup time, but it also, in the case of the Gollum character, takes the actor out of his natural environments and lacks surface details such as the dynamics of the hair and clothing.

Instead, a marker-less animation have been introduced, where different motion sequences are captured in a multiple view studio as in (Starck and Hilton, 2007) shown in Figure 1.2 using HD cameras. Using a multiple view stereo approach, 3D meshes are then reconstructed independently for each frame, which means both shape and appearance of the captured person are preserved as shown in Figure 1.3.

Capturing various motions of a person, thus having a library of different motions, new animations can be created by combining and transitioning between related motions in the library. Using a motion graph, a user have the possibility to control the different movements a character should perform. This motion graph is an animation synthesis that controls feasible transitioning points between chosen motions, where all states in the graph consists of small clips in a video library of that particular motion. For each motion, the end and start points of the video is manually labelled. An example is given in Figure 1.4, where all possible transitions between each motion is defined. Thus in order from first performing the ’walk’-motion and then the ’run’-motion, the character has to go through ’walk to run’.

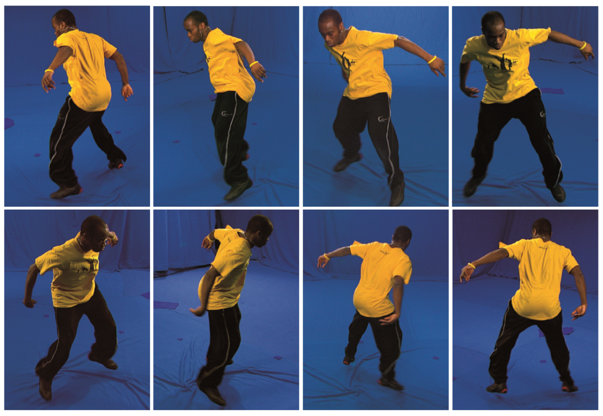


Figure 1.2: 8 cameras multiple view giving a 360 degrees of coverage evenly spaced with an interval of 45 degrees (Starck and Hilton, 2007).

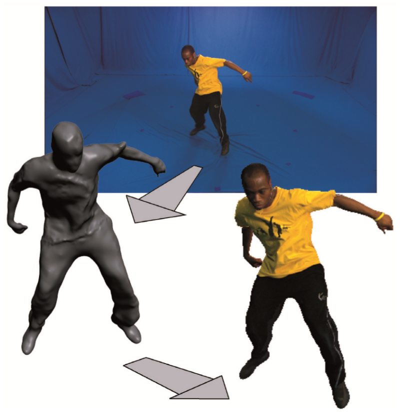


Figure 1.3: Reconstruction of a character based on the multiple view from Figure 1.2 (Starck and Hilton, 2007).

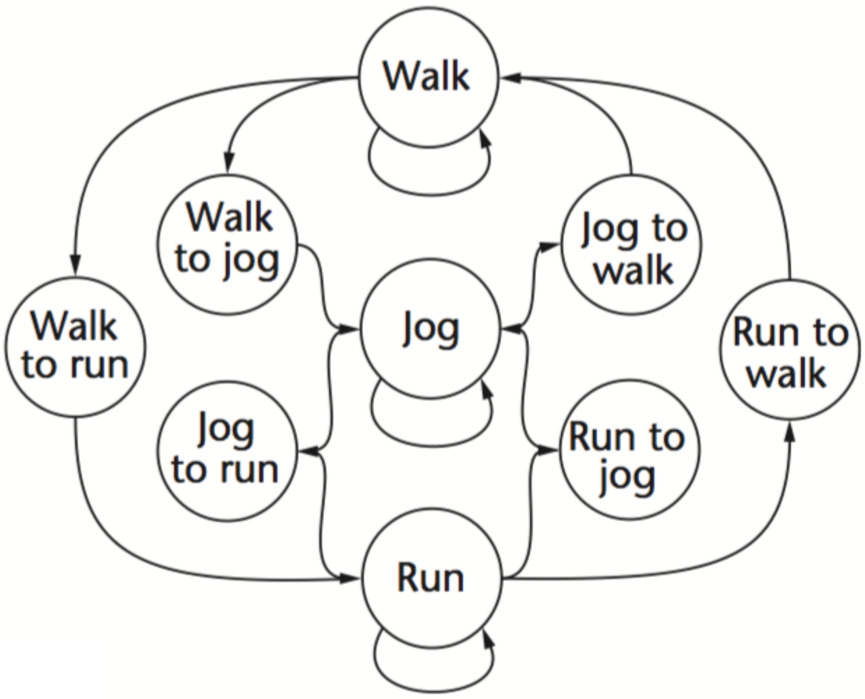


Figure 1.4: Example of a motion graph with walk, jog and run motions (Starck and Hilton, 2007).

In order to establish the best transition between two motions, a similarity measure between the feasible transitioning frames are made. According to a study of different similarity measures in (Huang et al., 2010), the shape histogram, illustrated in Figure 1.5, was a similarity measure that proved to give the best performance between different people and motions also used in (Budd et al., 2013). The shape histogram is subdivides a spherical coordinate system into radial and angular bins, where each bin represents the mesh in that specific area. The similarity measure between two meshes compares all rotations of the mesh around the centroid for which the L2 distance is minimum, thus being rotation invariant.

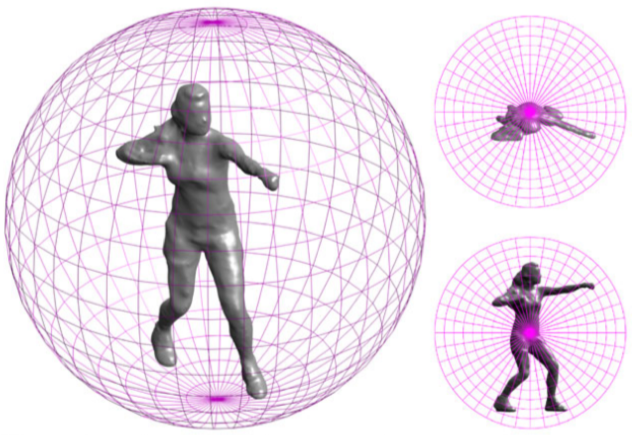


Figure 1.5: Illustration of the shape histogram similarity measure (Huang et al., 2010).

Having a reference mesh it is possible to compare how well the reconstruction of the surface and texture of a character is done as in (Casas et al., 2013). But whether the animations are made in a realistic way depends on how it is perceived. This means it is hard to establish a measure for the realism of the animated videos without testing how people perceive them, which is why the approach done in (Casas et al., 2014), where a user study was conducted to test the realism of their videos, would be an appropriate way for testing an animation.

1.1 Problem Statement

Recent work within reconstruction and modelling of people using multiple cameras can result in a realistic yet expensive video animation. The aim of this project is to investigate in a low-cost solution by using consumer video and depth cameras instead, i.e. a second generation Kinect for Windows, and use captured sequences to make an interactive animation of a person, which will build on previous work within 4D animation. The problem statement is therefore:

How should an interactive animation with seamless transitions which uses captured sequences of a person be constructed?

1.2 Kinect Overview

The first generation of Kinect, shown in Figure 1.6, was released in 2010 together with the 1.0 SDK in 2011 allowing developers to write applications for the Kinect. The second generation, shown in Figure 1.7, was released as a stand alone in July 2014 together with the upgraded

2.0 SDK. Before that, the sensor was only distributed together with all Xbox One consoles.

Figure 1.6: First generation Kinect.

Figure 1.7: Second generation Kinect.

The Kinect has video and audio modalities, which provides the four different data streams as listed below, and Table 1.1.

- Colour stream
- Infrared stream
- Depth stream
- Audio stream

The sensor overview between the two generations for the different modalities are summarized in

Sensor overview

Feature	Kinect v1	Kinect v2
RGB resolution	640 x 480	1920 x 1080
RGB format	.jpg	.jpg
IR and depth resolution	320 x 240	512 x 424
IR and depth format	.png	.png
Frame rate	30 Hz	30 Hz
Skeletal tracking	20 joints	25 joints
Skeltal format	.xml	.xml
FOV	Horizontal 57 degrees, Vertical 43 degrees	Horizontal 70 degrees, Vertical 60 degrees
Audio	4 microphone array	4 microphone array
Hardware prerequisites	USB 2.0, Windows 7	USB 3.0, Windows 8

Table 1.1: Overview of the two Kinect sensors (Microsoft, a).

It is possible with the first generation of Kinect to receive a higher resolution of the RGB camera of 1280 x 960, but this is at the expense of decreasing the frame rate. The second generation sensor can also capture RGB and IR/Depth at the same time, which is not possible for the first generation.

For each Kinect generation an Xbox and a Windows version exists, which are identical sensors. The only difference is that the Xbox Kinect could not connect to Windows until an adapter was released later in 2014. In 2015, Microsoft announced that they stopped producing the Kinect for Windows, meaning if one were to buy the second generation Kinect today and connect it to Windows, the Kinect One sensor must be bought together with an adapter (Microsoft, 2015).

Using the four data streams, the SDK can provide tools and additional functionalities for building Windows applications. A short list of these is listed below:

- Skeletal tracking
- Lean tracking
- Gesture tracking
- Face tracking
- Speech recognition

One of the featured functionalities using the SDK based on the input from the data streams is the skeletal tracking, where 25 body joints distributed as shown in Figure 1.8 are tracked for a person facing the sensor with no occlusions (Microsoft, b).

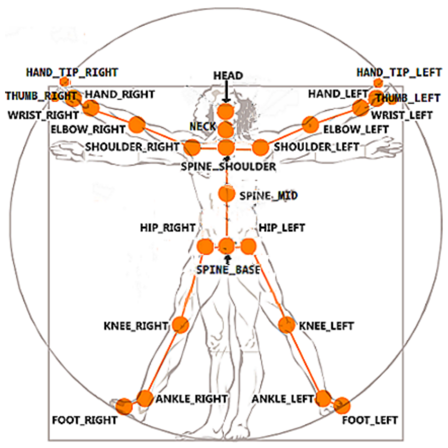


Figure 1.8: Overview of the 25 tracked joints in the skeleton (Microsoft, b).

The skeletal tracking is based on the depth frames as shown in Figure 1.9. It classifies body segments from the depth data, and hereafter place the joint points at the places with highest probability according to a pretrained model based on multiple persons of various size and different clothing (Zhang, 2012).

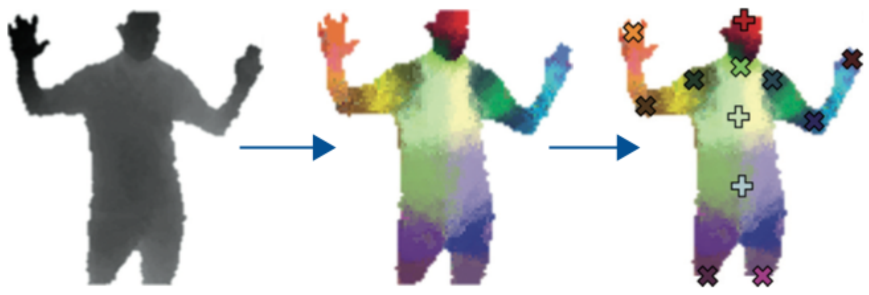


Figure 1.9: Extraction of the skeletal joints (Zhang, 2012).

Since the second generation Kinect for Windows is used in this project, this this referred to as Kinect throughout the report unless otherwise specified.

2. *Project Overview*

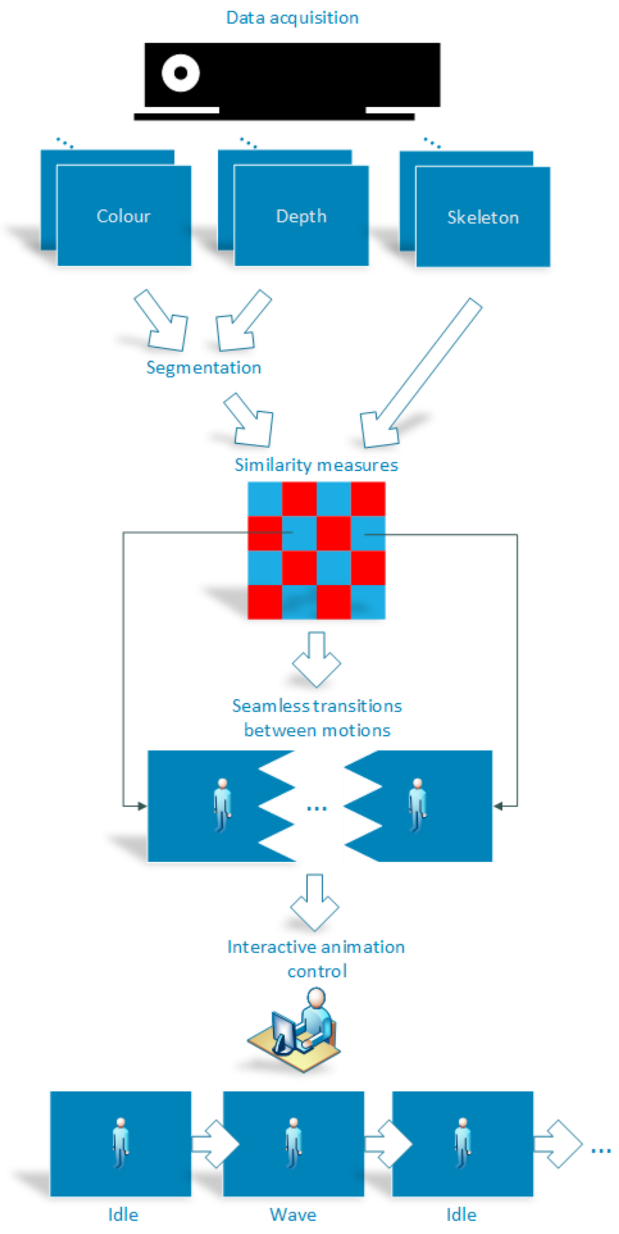


Figure 2.1: Illustration of the overall system design.

The overall system design for this project is illustrated in Figure 2.1, which shows the different stages that are shortly described below.

Data acquisition - Colour, depth and skeleton data is captured from the Windows Kinect v2 sensor.

Segmentation - The background is removed in the colour and depth frames so only the person remains.

Similarity measures - Similarity measures are found among all captured frames using colour, depth and skeleton data to find similar frames to transition between.

Seamless transitions between motions - Animations between transitioning frames are created to make seamless transitions.

Interactive animation control - All above steps are implemented in a program, where a user would be able to control the motions that a character should perform and thereafter play the video.

2.1 Requirements

The requirements for the captured data are summarized in Table 2.1.

Data requirements

Description	**Specification**	**Reference**
RGB frame resolution	1920 x 1080	Table 1.1
Depth frame resolution	512 x 424	Table 1.1
RGB frame format	Must be .jpg	Table 1.1
Depth frame format	Must be .png	Table 1.1
Skeleton format	Must be .xml	Table 1.1
Scene composition	The scene must only contain one person facing the camera	section 1.2
Scene composition	The scene must contain the entire person without occlusions	section 1.2

Table 2.1: Requirements for the input data to the system.

As described in chapter 1, the evaluation of the smoothness and realism of an animation is determined by how it is perceived, which is why the proposed animation throughout this work will be evaluated by a user study, which was also performed in (Casas et al., 2014).

2.2 Datasets

Datasets are captured using a c# - program, *KinectV2Recorder* (Bahnsen), where a screen shot of the interface is shown in Figure 2.2.

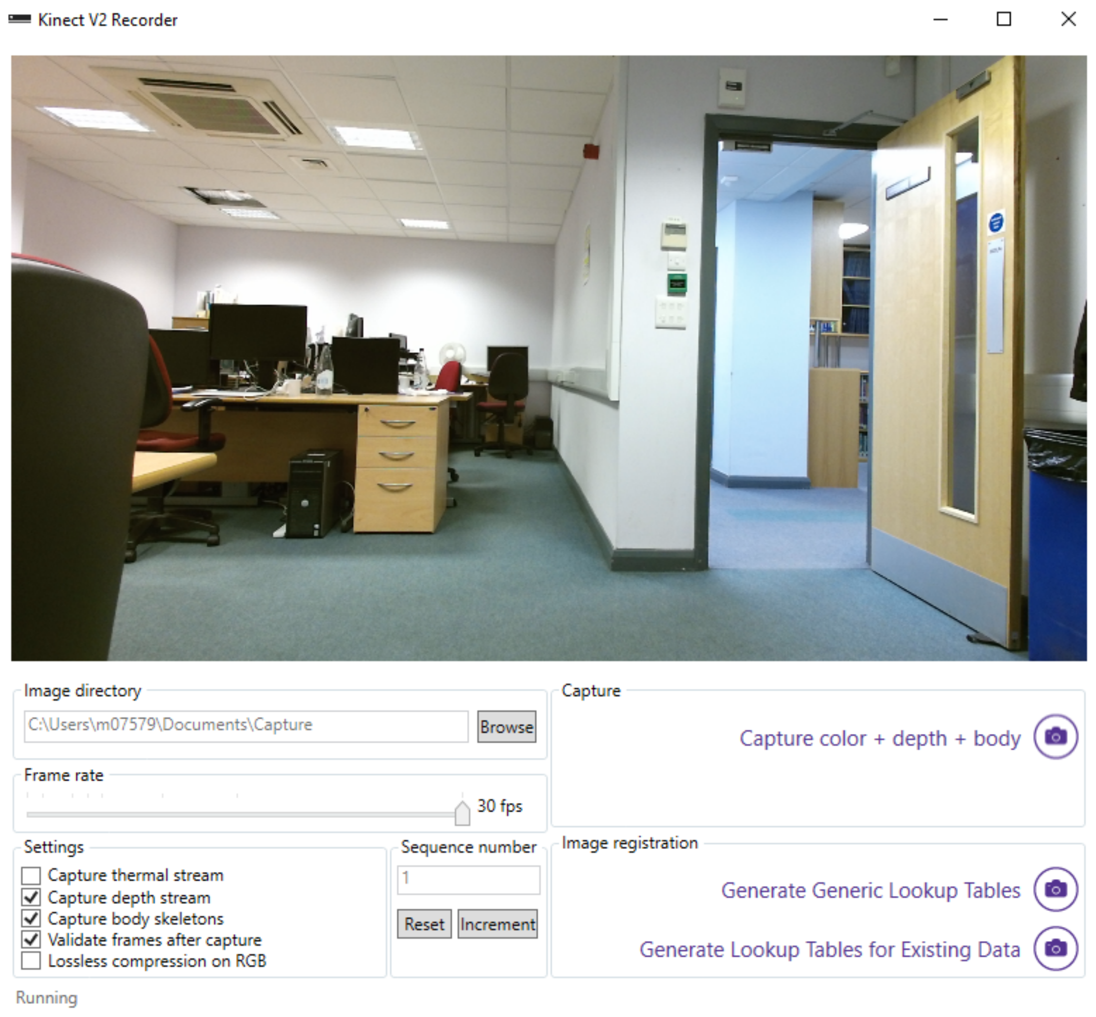

Kinect V2 Recorder

Image directory: C:\Users\m07579\Documents\Capture

Frame rate: 30 fps

Settings:
- [ ] Capture thermal stream
- [✓] Capture depth stream
- [✓] Capture body skeletons
- [✓] Validate frames after capture
- [ ] Lossless compression on RGB

Sequence number: 1

Running

Browse
Reset
Increment
Capture
Capture color + depth + body
Image registration
Generate Generic Lookup Tables
Generate Lookup Tables for Existing Data

Figure 2.2: Screen shot of the program used for capturing data.

The program saves he placement of the skeletal joints, the depth frame and the colour frame captured for each time instance in a directory specified by the user. Further, the lookup tables are also saved in order to perform an off-line mapping between colour and depth frames.

Four datasets of a person performing different motions are used in this project. The first dataset, which is used for evaluation throughout the report, is a person alternately waving with left and right hand. The other three datasets are used for testing, and their motions are further described in subsection 2.2.1. Frame 100 for each dataset is shown in Figure 2.3 to Figure 2.6.

2.2.1 Division of Motions

In previous work, (Huang et al., 2010), each motion sequence is captured in one video at a time including videos where a person transition from one motion to another. Since the captured datasets in this work are each a full video of a person performing different motions, the motion sequences an their transitions needs to be divided and labelled, which is done

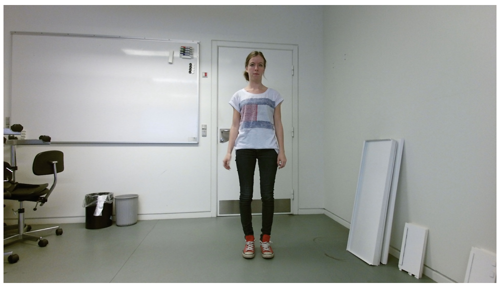


Figure 2.3: Frame 100 of the evaluation dataset.

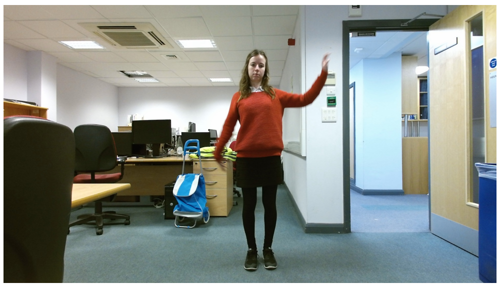


Figure 2.4: Frame 100 of the test dataset for the Cathrine character.

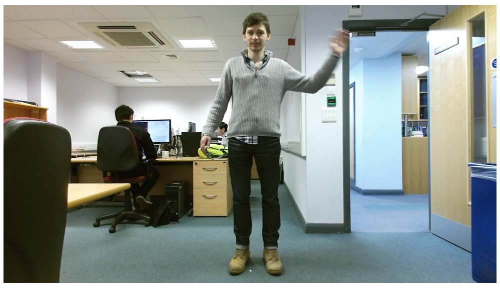


Figure 2.5: Frame 100 of the test dataset for the Marco character.

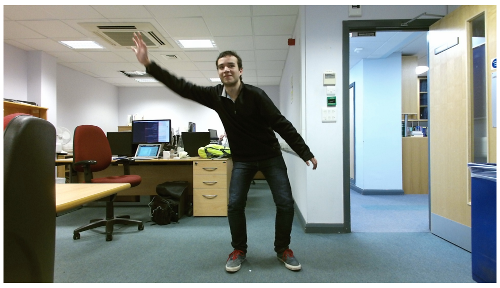


Figure 2.6: Frame 100 of the test dataset for the Leonardo character.

manually by running through the video and select where each sequence starts and stops. The motions are divided in a way so they do not overlap, i.e. $motion_i \cap motion_{i+1} = \emptyset$.

The divisions for the test datasets are illustrated by a time line for each video in Figure 2.7 to Figure 2.9.

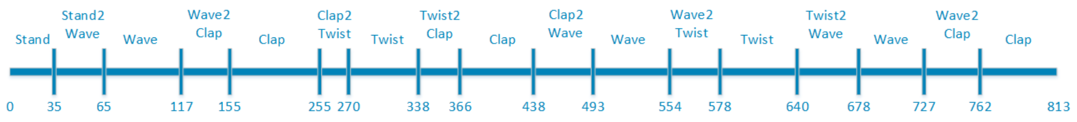


Figure 2.7: Division of motions for the Cathrine character.

In order to control the transition between each motion, a motion graph as in (Casas et al., 2012) is made, which keeps track of the current state and the possible transitions. The motion graphs for each character are illustrated in Figure 2.10 to Figure 2.12.

Other than the Cathrine character, which can directly transition between all motions, the two other characters need to return to idle between each motion, which the motion graph assures.

The information about the motions for each character is gathered in a meta data file, which contains:

- A list of possible motions as strings, e.g. motions = ["Wave", "Clap", "Twist"]
- A list of divisions in the video, e.g. divisions = [35, 65, 117, ...]
- A database of motions, where each motion is described by an end and start frame.
- A motion graph, where the current and next possible motions are controlled by if loops.

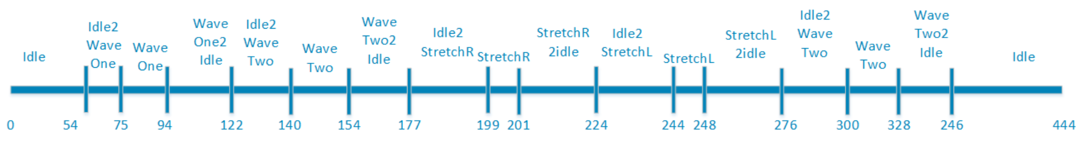


Figure 2.8: Division of motions for the Marco character.

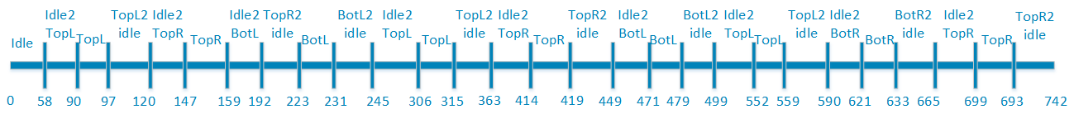


Figure 2.9: Division of motions for the Leonardo character.

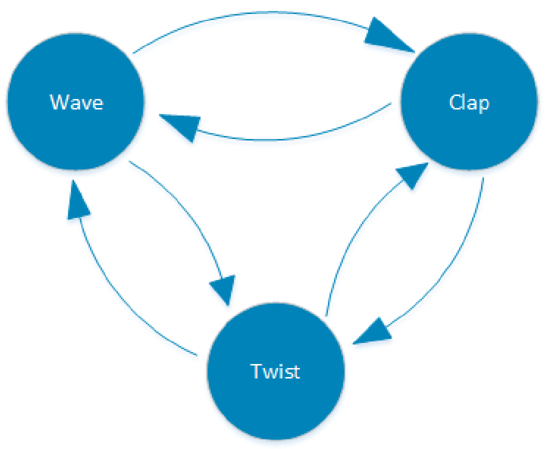


Figure 2.10: Motion graph for the Cathrine character.

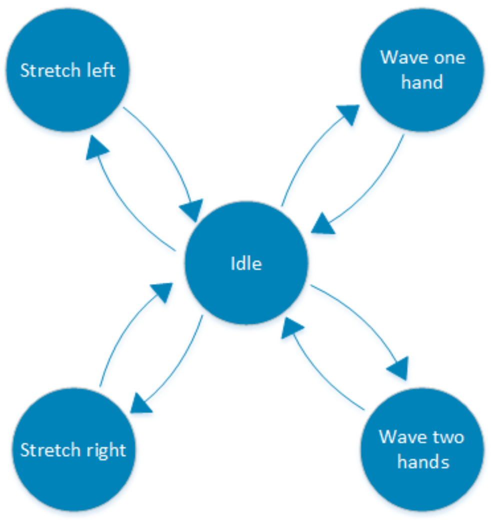


Figure 2.11: Motion graph for the Marco character.

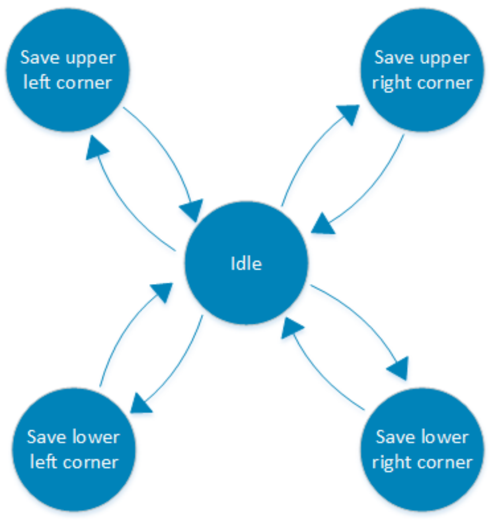


Figure 2.12: Motion graph for the Leonardo character.

3. Segmentation

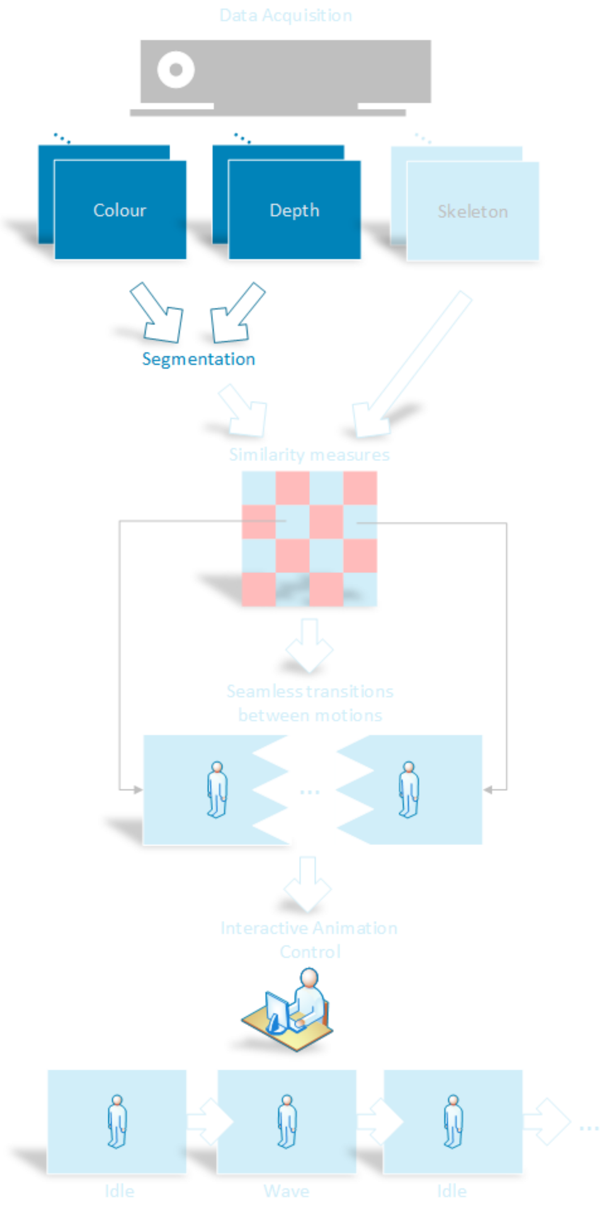


Figure 3.1: Illustration of the overall system design.

When calculating the similarity measures of the person from depth and colour frames, a measure which is not disturbed by the background is wanted. Therefore a segmentation of the person is needed, where a maximum of the person is preserved and a maximum of the background is removed. The segmentation labels the pixels to a class, and in this case, the pixels are either classified as foreground, i.e. the person, or background.

The segmentation can be done using either the colour or depth frames. If the colour frame is used, there has to be taken account for illumination changes within the frames. In order to reduce this illumination problem, [Bouwmans et al. 2008] suggests to convert the colour frames from RGB to YCbCr, where the Y channel (the luminance channel) is not taken into account, and therefore reduce the illumination problems.

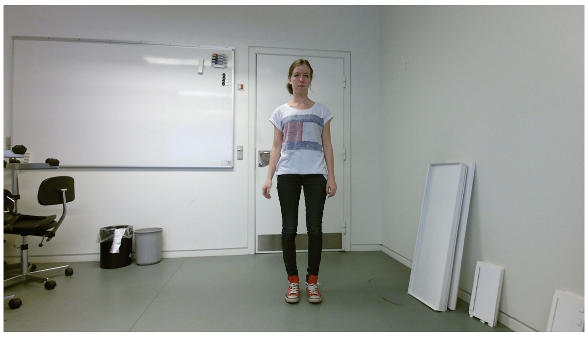


Figure 3.2: Colour frame 100.

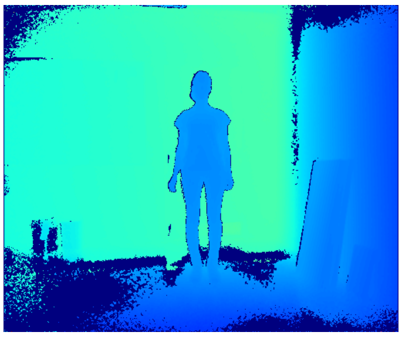


Figure 3.3: Depth frame 100.

Another problem with using the colour frames for segmentation of the person is cluttering, i.e. similarity in appearance of colour of the person and the background. A solution is to use the depth frames instead, which avoids the cluttering and contains less illumination problems. The reason why it is less and not avoided, is that the IR data stream, which the depth frames are based on, can vary between very light and vary dark materials.

Since the camera is static, the only motion present in the in the video is the motion of the person, which is why a model of the background, as described in next section, could be used for segmenting the person.

3.1 Segmentation Based on Modelling the Background

One way for segmenting the person from the background is by using one or more frames of the background containing no person and then subtract the frame containing the person. One problem with using just a single background frame is that it can be highly sensitive to noise according to which background frame that is chosen.

In order to cope with the temporal changes in noise, the background is learned using multiple frames. This means a background model is made by calculating the Gaussian probability density function of each pixel across a known amount of background frames.

When having all pixels in the background model described by its mean, μ and variance, σ^2, the frame which needs to be segmented, F, is firstly subtracted with the mean of the background model. The threshold is then set according to a multiple, k, of the standard deviation, σ, of the background model. If a pixel is within a multiple of the standard deviation, it is determined as foreground and set to 1 and therefore a part of the person, otherwise it is determined as background and set to 0. This is also shown in Equation 3.1.

$$S = \begin{cases} 1, & \text{if } |F - \mu| \leq k\sigma. \\ 0, & \text{otherwise.} \end{cases} \tag{3.1}$$

Having 15 background frames to model the background, the segmented output of frame 100 and frame 800 thresholded with different standard deviations are shown in Figure 3.4 to Figure 3.7.

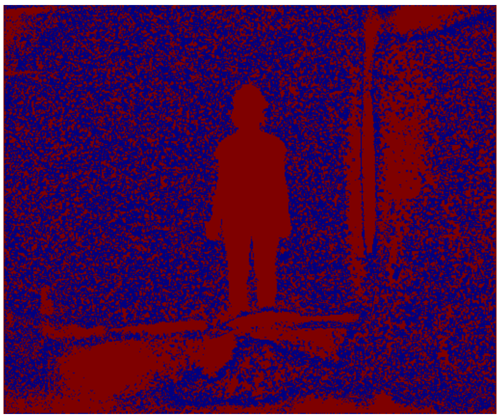


Figure 3.4: Segmented frame 100 with 1 standard deviation.

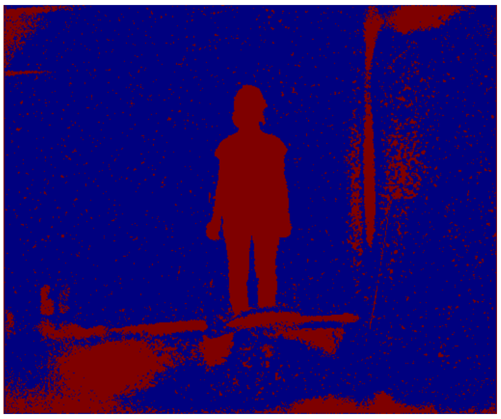


Figure 3.5: Segmented frame 100 with 3 standard deviations.

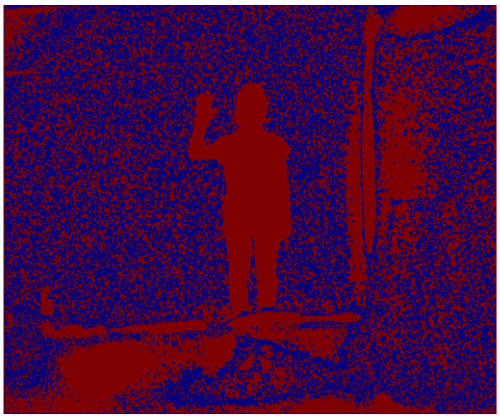


Figure 3.6: Segmented frame 800 with 1 standard deviation.

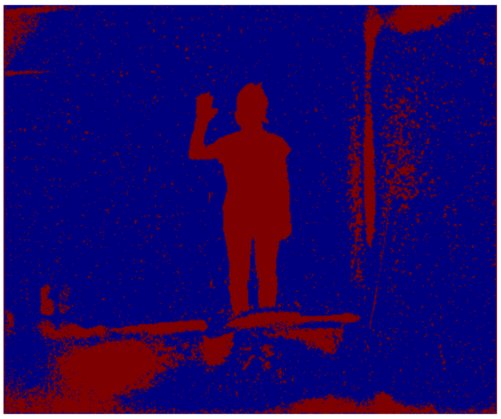


Figure 3.7: Segmented frame 800 with 3 standard deviations.

As shown in Figure 3.4 to Figure 3.7, noise around the person occurs and most of all there is a problem due to the small difference in distances between the feet and the floor, which means that the feet are not segmented from the floor. In order to solve this, a solution could be to look at the floor as a plane and then exclude the points that is on that plane.

3.2 Segmentation Based on Planes

In order to segment the feet from the floor using a plane, three points of the floor needs to be known. When doing a segmentation based on a Gaussian model for each pixel as in section 3.1, there is a problem with determining which points are part of the floor. One solution is that points of the floor is known beforehand by user interaction. Because that it is already known that points of the floor has a similar depth to the body a simpler a faster segmentation can be used to determine the points automatically.

This can be done by using the information from the skeleton data to find the range of the body in the x-, y- and z-direction plus an extra margin. This approach requires that the surroundings do not have any furniture or items that have the same range of the person plus the margin. This margin is then chosen to be 20 cm, which maintains the entire person. An example of using the skeleton for segmentation of frame 100 and 800 with the extra margin in each direction is shown in Figure 3.8 and Figure 3.9.

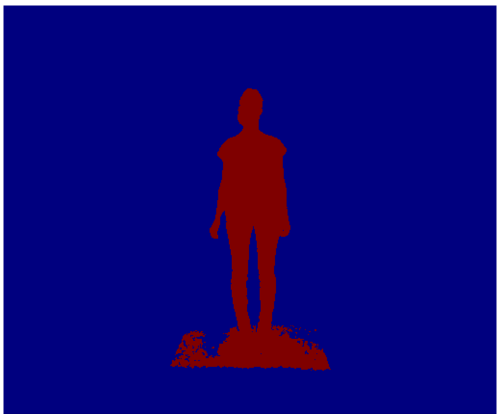


Figure 3.8: Segmented frame 100 based on the skeleton plus a margin of 20 cm in each direction.

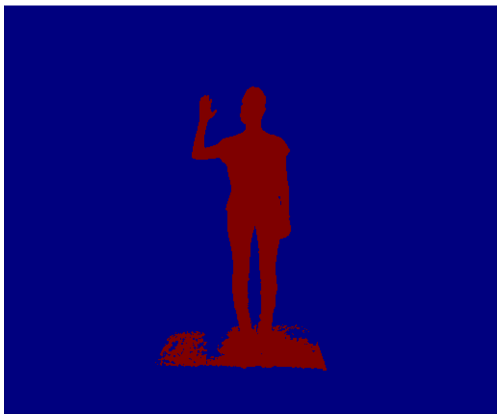


Figure 3.9: Segmented frame 800 based on the skeleton plus a margin of 20 cm in each direction.

The extra margin can provide three points of the floor, which can be used for creating a plane of the floor. These points are the minimum x-value, maximum x-value and minimum z-value which are illustrated in Figure 3.10.

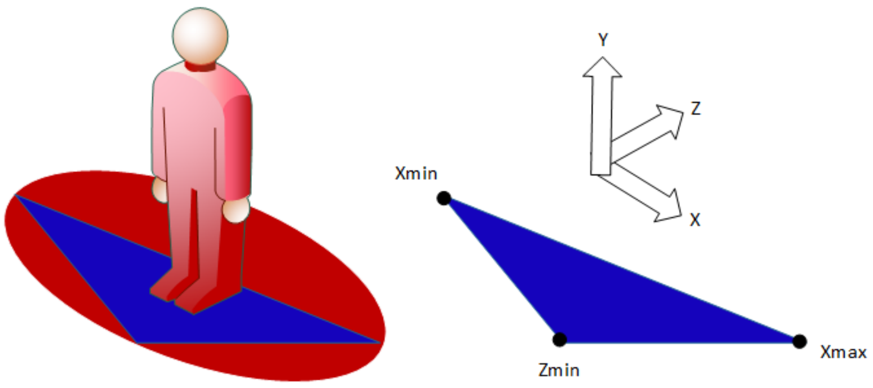


Figure 3.10: Illustration of how the plane of the floor is extracted.

By having the normal vector, (a, b, c), which is the cross product of two vectors created by the three points, an equation of the plane can be made in Equation 3.2, where $(x_0, y_0, z0)$ is a known point on the plane, i.e. one of the three points.

$$a(x - x_0) + b(y - y_0) + c(z - z0) = 0 \tag{3.2}$$

The distance from a plane α with the normal vector, (a, b, c), to a point P, $(x_0, y_0, z0)$, is calculated using Equation 3.3.

$$dist_{\alpha,P} = \frac{|a(x - x_0) + b(y - y_0) + c(z - z0)|}{\sqrt{a^2 + b^2 + c^2}} \tag{3.3}$$

Before determining the plane of the person, a conversion from depth to camera space is done. This is needed in order to avoid interpolation between the pixels in the depth frame when determining whether the points/pixels are close to the plane.

If a point is therefore within a known distance of the plane, it is removed. If the distance is too low, most of the floor is not removed as shown in figures Figure 3.11 to Figure 3.13. On the other hand if the distance is too high, both the floor and the feet are removed as shown in Figure 3.14 to Figure 3.16. A value in between must therefore be chosen where a maximum of person is preserved and a maximum of the floor is removed. This distance is evaluated in subsection 3.2.2.

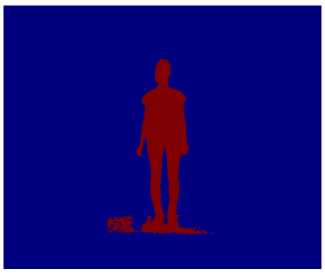


Figure 3.11: Frame 100 with removed points within 1 cm of the plane.

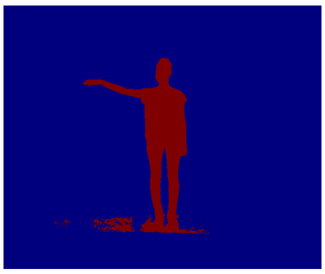


Figure 3.12: Frame 450 with removed points within 1 cm of the plane.

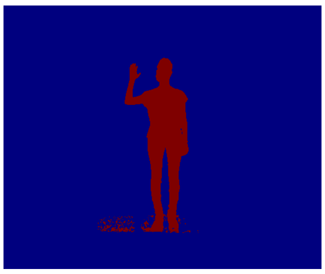


Figure 3.13: Frame 800 with removed points within 1 cm of the plane.

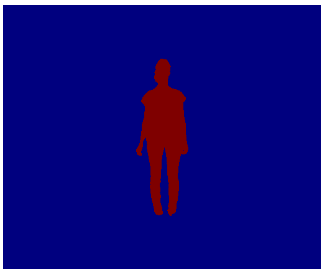


Figure 3.14: Frame 100 with removed points within 10 cm of the plane.

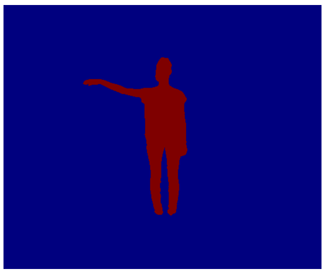


Figure 3.15: Frame 450 with removed points within 10 cm of the plane.

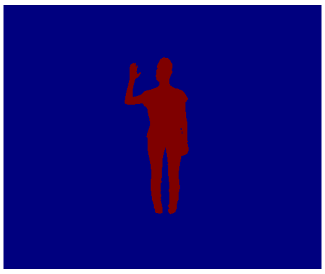


Figure 3.16: Frame 800 with removed points within 10 cm of the plane.

3.2.1 Average of Planes

Figure 3.11 to Figure 3.16 have been segmented individually, which means a new plane is made for each frame. This can result in different segmentations from frame to frame caused by small pixel changes. As an example, frame 100 in Figure 3.11 needs a higher distance to remove points on the floor than frame 800 in Figure 3.13. Since the camera is static, it is not necessary to find a new plane for each single frame because the floor is also static, why only one single plane is used for all frames.

This single plane is found by an average over a number of planes from the first frames. In order to know how many frames that should be used, the sum of differences between the points that describes the plane, i.e the normal vector and the known point, are evaluated. The number of frames which must be used, should therefore be at the point where adding additional frames does not have a significant change to the description of the plane.

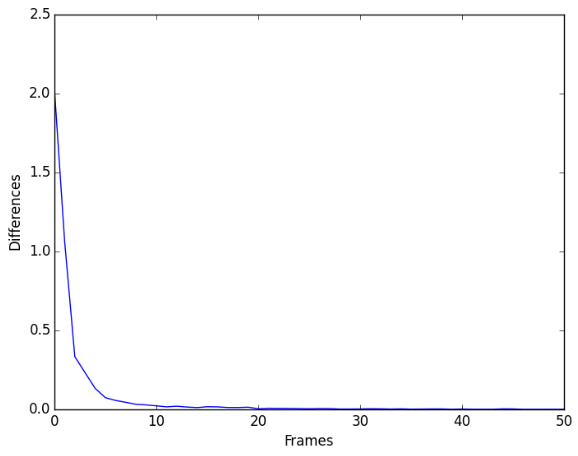


Figure 3.17: Average of distances between planes with increasing number of frames.

Figure 3.17 shows that after 20 frames, the sum of differences converge, which is why the 20 first frames are chosen for determining the average plane.

The improvement can be shown in subsection 3.2.2 and Appendix A, where points are removed at different distances to the plane.

3.2.2 Evaluation of Distances

In this section, the best distance to the plane that removes the maximum part of the floor and a minimum part of the person is evaluated. Instead of labelling the foreground and background manually for every frame to evaluate the segmentation, a visual evaluation is done instead, where 10 different frames spaced different places in the video are investigated. The frames are segmented using different distances, and the output from one frame is shown in this section, and the rest 9 frames are shown in Appendix A.

Since a distance of 10 cm to the plane where shown in Figure 3.14 to Figure 3.16 to be too high, the values that are evaluated are from 5 cm to 9 cm with an interval of 1 cm, which are shown in Figure 3.18 to Figure 3.22.

Choosing a distance of 8 and 9 cm in Figure 3.18 and Figure 3.19, shows that the floor is removed but parts of the feet are also removed. This means these values are too high. On the other hand, when choosing a distance of 5 and 6 cm in Figure 3.20 and Figure 3.21, shows that not all of the floor is removed and therefore is a value that is too low.

Choosing a distance of 7 cm in Figure 3.22, shows that a small amount of the floor is still present, but the whole person is intact. It is possible that the remaining noise can be removed by post-processing, which is described in next section.

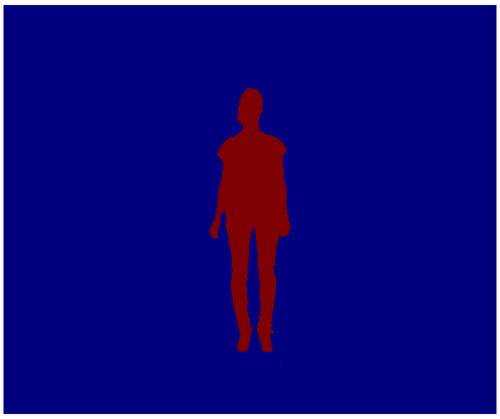


Figure 3.18: Frame 100 segmented with a distance of 8 cm to the plane.

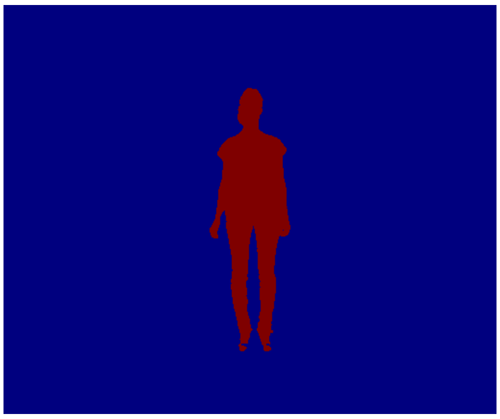


Figure 3.19: Frame 100 segmented with a distance of 9 cm to the plane.

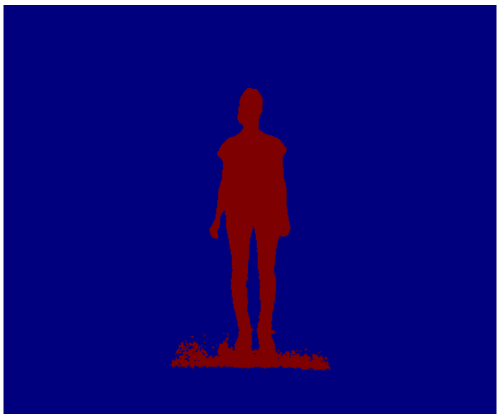


Figure 3.20: Frame 150 segmented with a distance of 5 cm to the plane.

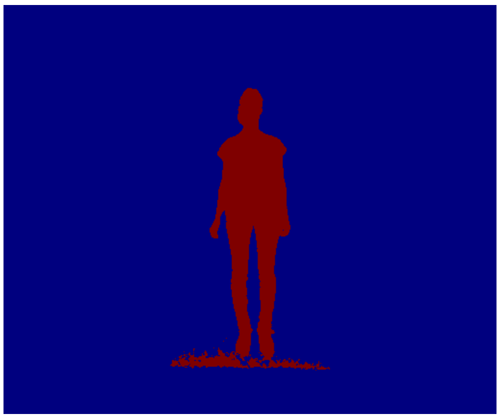


Figure 3.21: Frame 100 segmented with a distance of 6 cm to the plane.

3.2.3 Post-processing

One way to remove the noise from the segmentation could be by using binary morphology. The downside by using this, is that the shape of the body is not entirely preserved, and also the ideal size of the structured element must be the same size as the noise which can vary between frames. Another approach is therefore used. By assuming that the output from the plane segmentation contains connected BLOBs (Binary Large OBjects) (Moeslund, 2012), where the biggest one is the body, the detection of the pixel sizes of all the BLOBs can be used to remove the smaller ones. In order to figure out whether pixels are connected or not, a connected component analysis is done based on 4-connectivity or 8-connectivity as illustrated in Figure 3.23.

Since BLOBs are unlikely to belong to the body when they touch corners as in Figure 3.23, a 4-connectivity is therefore chosen when establishing the size of the BLOBs. Further a 4-connecivity needs fewer computations and the process of finding BLOBs in an image can therefore happen faster.

An example of removing the noise from the output of the plane segmentation in Figure 3.22, where the biggest BLOB is kept shown in Figure 3.24.

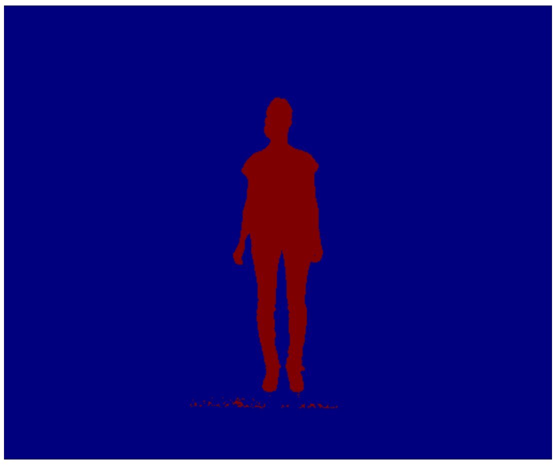


Figure 3.22: Frame 100 segmented with a distance of 7 cm to the plane.

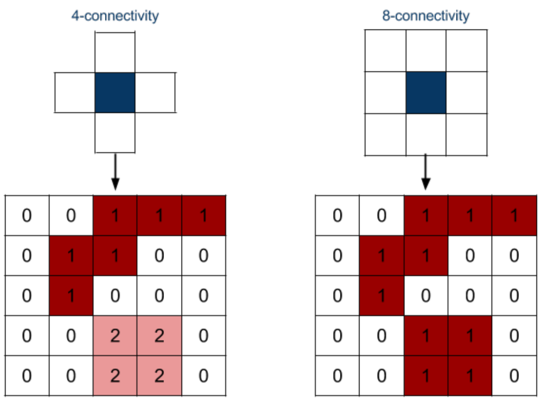


Figure 3.23: Illustration of applying 4- and 8-connectivity to the same image.

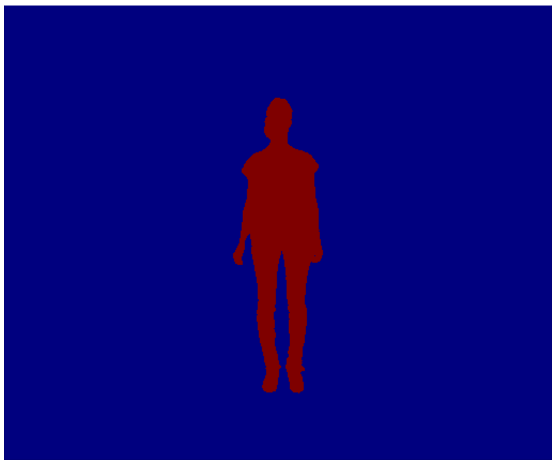


Figure 3.24: Frame 100 after removing the noise.

The 9 other frames from the evaluation in Appendix A also include the removed noise with the chosen distance of 7 cm in the end of the appendix, which all by a visual evaluation have an acceptable segmentation.

3.3 Segmentation Overview

Flow of the segmentation implemented as *processData.py*, where the input is the skeleton, depth and colour frames from a dataset:

- Find range of body in each direction using the skeleton data plus a safety margin
- Find the average plane of the floor for the first 20 frames.
- Exclude points close to the plane
- Reduce additional noise by preserving only the biggest BLOB, i.e. the person.

Boolean bitwise AND operation is applied to the output frame and the original depth frame. This depth frame is then further used for mapping the colour frame onto the depth frame.

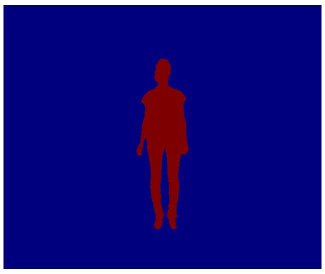


Figure 3.25: Frame 100 output from segmentation.

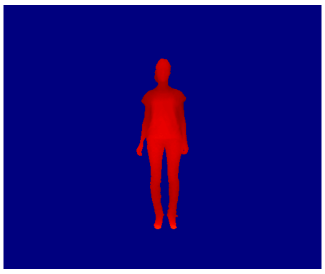


Figure 3.26: Frame 100 with its corresponding depth frame.

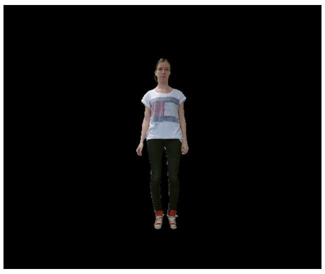


Figure 3.27: Frame 100 mapped to colour using the depth information in Figure 3.26.

4. Similarity Measures

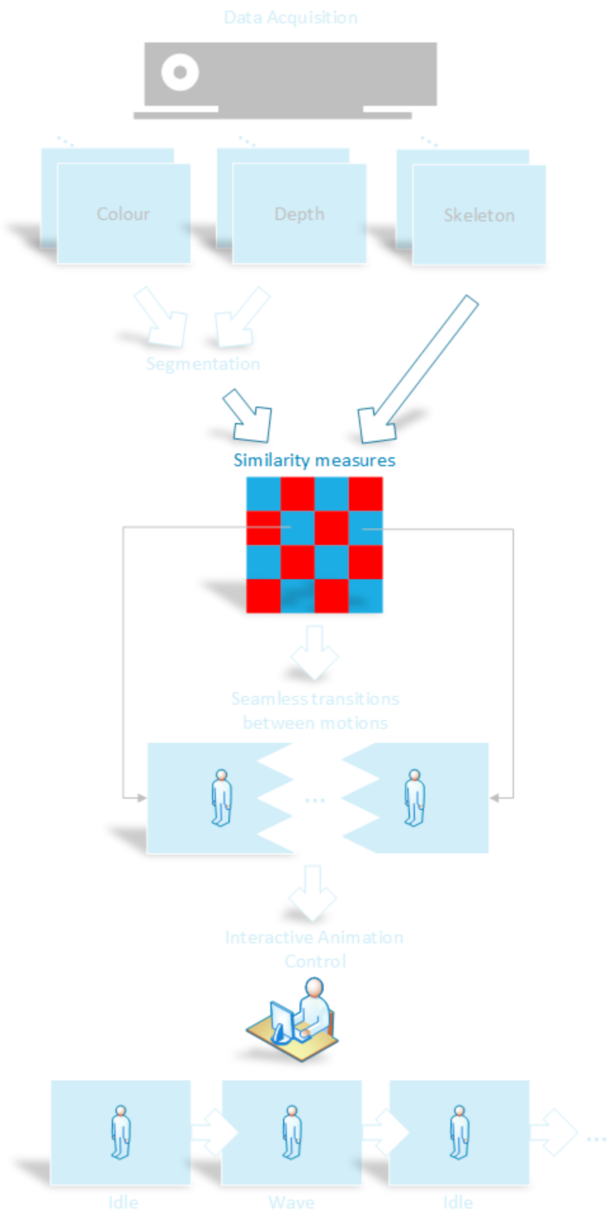


Figure 4.1: Illustration of the overall system design.

Having segmented the person and using the masked frame to get the depth and colour information, similarity measures for each of the three modalities, skeleton, depth and colour, are made. These similarity measures should be used for determining the best transition amongst other frames in the video where the pose and appearance have the best match.

For each modality the similarity between frames are calulated as the Eucledian distance (L2 distance) between two feature vectores $p = (p_1, p_2, ..., p_n)$, and $q = (q_1, q_2, ..., q_n)$ as in Equation 4.1.

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + ... + (p_n - q_n)^2} = \sqrt{\sum_{n=1}^{N} (p_n - q_n)^2} \tag{4.1}$$

The similarity measure is calculated between all possible frame pairs, and having N number of frames this will give an N $\times$ N matrix as shown in Equation 4.2. Since the frames are also compared with itself, the self-similarity matrix is zero on the diagonal and symmetrical, so $S_{p,q} = S_{q,p}$.

$$S = \begin{bmatrix} 0 & S_{0,1} & ... & S_{0,N} \\ S_{1,0} & 0 & ... & S_{1,N} \\ \vdots & \vdots & \ddots & \vdots \\ S_{N,0} & S_{N,1} & ... & 0 \end{bmatrix} \in R^{N \times N} \tag{4.2}$$

After individual normalization of each similarity measure by dividing all values with the maximum value, they are combined into one similarity measure, where the weight for each measure, α, β and γ, can be set, as shown in Equation 4.3.

$$S = \alpha S_{skel} + \beta S_{depth} + \gamma S_{colour} \quad \text{where} \quad \alpha, \beta, \gamma \in [0,1] \tag{4.3}$$

The similarity measures for the skeleton, S_{skel}, the depth data, S_{depth}, and the colour data, S_{colour}, are described in the next sections.

4.1 Similariy Measure of Skeleton Data

Using the joints from the skeleton as a similarity measure can give a good starting point in evaluating the different positions of the body between each frame. The feature vector for each frame is made by using all the positions of the skeleton's joints in world space in 3D, which are illustrated in 2D in Figure 4.2 to Figure 4.4.

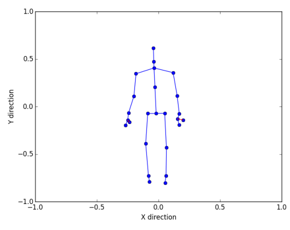


Figure 4.2: Skeleton's joints positions for frame 100 in 2D.

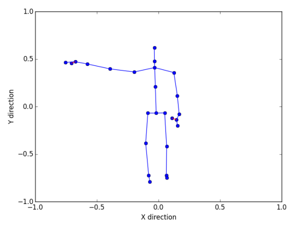


Figure 4.3: Skeleton's joints positions for frame 450 in 2D.

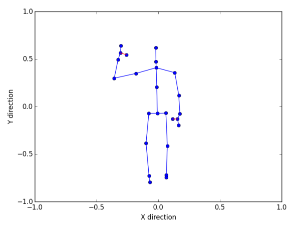


Figure 4.4: Skeleton's joints positions for frame 800 in 2D.

A feature vector for a single frame, p, is therefore described as Equation 4.4, where N is the number of joints, which is 25.

$$skel_p = [joint0_x, joint0_y, joint0_z, ..., jointN-1_x, jointN-1_y, jointN-1_z] \in R^{1\times(N\cdot3)} \tag{4.4}$$

Having this feature vector for one frame, the similarity measure between two frames, p and q, is calculated using Equation 4.1 as shown in Equation 4.5.

$$S_{skel} = d(skel_p, skel_q) \tag{4.5}$$

The self-similarity matrix of the skeleton's joints for all frames is shown in Figure 4.5. The matrix illustrates similar motions throughout the capture.

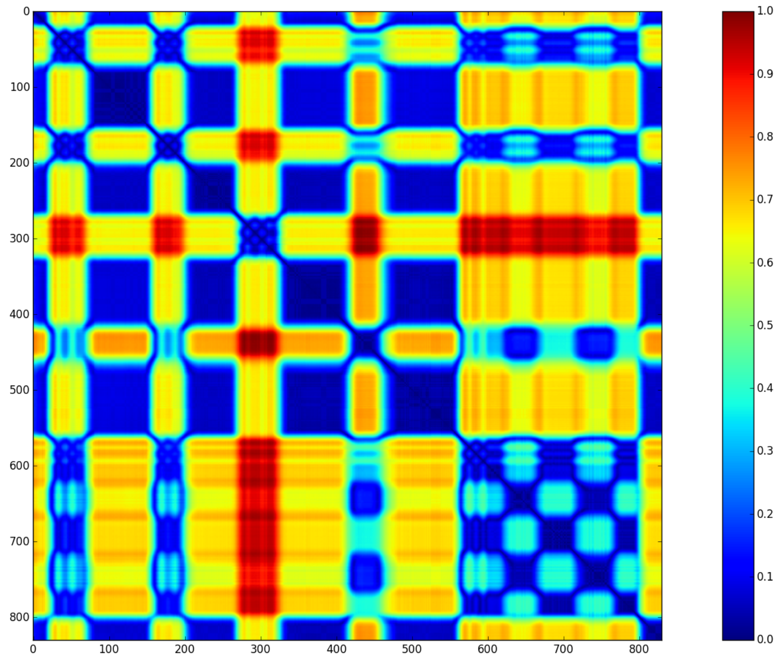


Figure 4.5: The self-similarity matrix of the skeleton's joints mapped from similar, dark blue, to dissimilar, red.

4.2 Similarity Measure of Depth Data

The depth data can provide more information about the position of the body. This is done by evaluating each depth value from each pixel in the segmented frame, which corresponds to the size of the depth frame of 512 × 424. Examples of the segmented depth frames corresponding to the skeletons in Figure 4.2 to Figure 4.4 are shown in Figure 4.6 to Figure 4.8.

The feature vector for a depth frame is the unravelled segmented frame with depth values of each pixel position as shown in Equation 4.6, where N is 512 and M is 424.

$$depth_p = [D_{x0,y0}, D_{x0,y1}, ..., D_{x0,yM}, D_{x1,y0}, D_{x1,y1}, ..., D_{xN-1,yM-1}] \in R^{1\times(N\cdot M)} \tag{4.6}$$

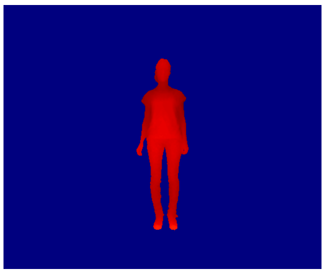


Figure 4.6: Segmented depth frame 100.

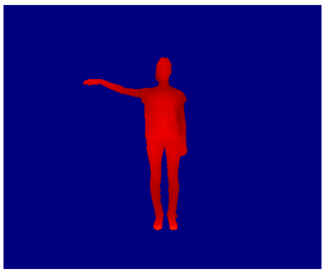


Figure 4.7: Segmented depth frame 450.

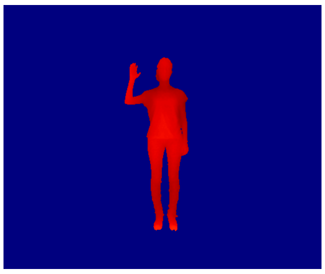


Figure 4.8: Segmented depth frame 800.

The similarity between two frames, p and q, is done in the same way as for the skeleton joints in Equation 4.5 as shown in Equation 4.7.

$$S_{depth} = d(depth_p, depth_q) \tag{4.7}$$

The self-similarity matrix of the depth data for all frames is shown in Figure 4.9.

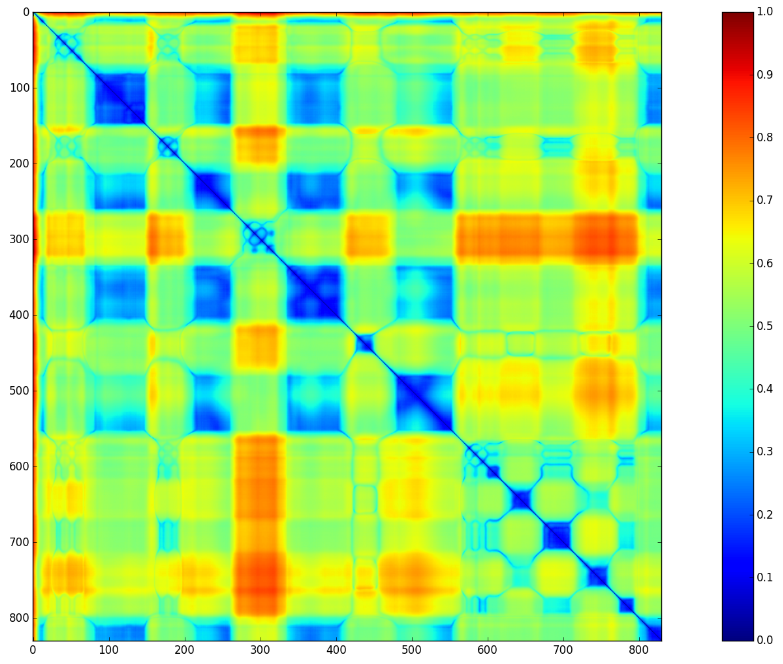


Figure 4.9: The self-similarity matrix of the depth data mapped from similar, dark blue, to dissimilar, red.

4.3 Similarity Measure of Colour Data

The downside by only using skeleton and/or depth data, is that is does not tell anything about the appearance of the person, which is why the colour data is also introduced as a similarity measure. Just as the similarity measure for the depth, the colour is also based on the segmented frame, where each colour pixel is evaluated in the same way. Examples of the colour frames corresponding to the skeletons in Figure 4.2 to Figure 4.4 and the depth frames in Figure 4.6 to Figure 4.8 are shown in Figure 4.10 to Figure 4.12.

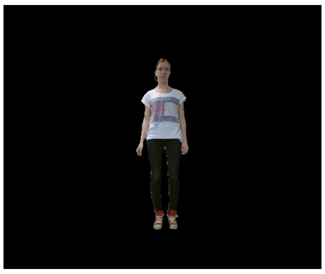


Figure 4.10: Segmented colour frame 100.

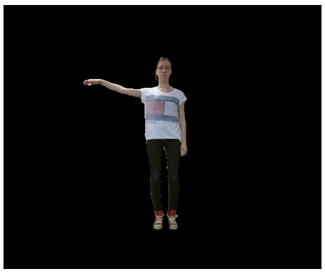


Figure 4.11: Segmented colour frame 450.

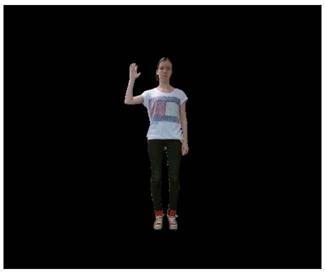


Figure 4.12: Segmented colour frame 800.

Since the segmented frame masked with the colour frame is 512 × 424 × 3, the feature vector for the colour frame is therefore described as in Equation 4.8, where N is 512 and M is 424.

$$colour_p = [R_{x0,y0}, G_{x0,y0}, B_{x0,y0}, , R_{x0,y1}, G_{x0,y1}, B_{x0,y1}, ..., R_{xN-1,yM-1}, G_{xN-1,yM-1}, B_{xN-1,yM-1}] \in R^{1\times(N\cdot M\cdot 3)} \tag{4.8}$$

Just as for the skeleton joints and depth data in Equation 4.5 and Equation 4.7, the similarity between two frames, p and q, is described in Equation 4.9.

$$S_{colour} = d(colour_p, colour_q) \tag{4.9}$$

The self-similarity matrix of the colour data for all frames is shown in Figure 4.13.

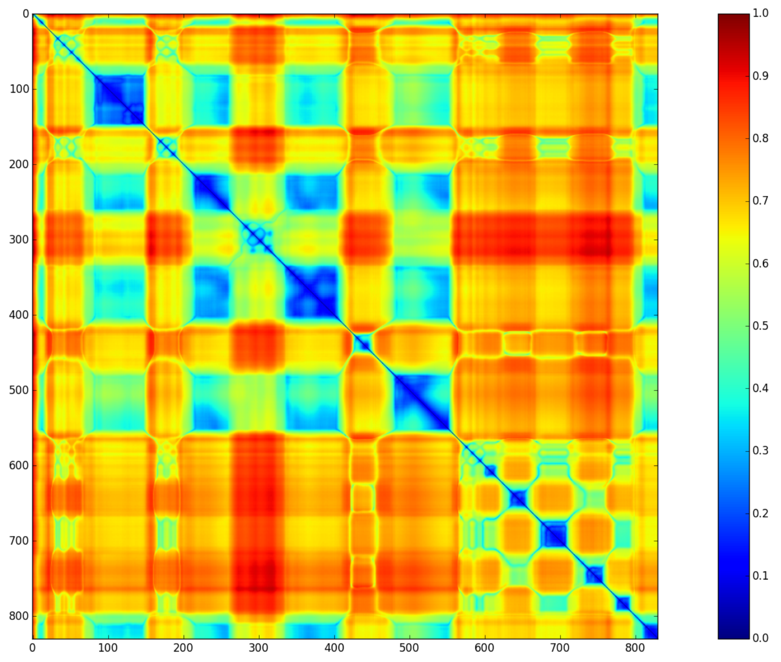


Figure 4.13: The self-similarity matrix of the colour data mapped from similar, dark blue, to dissimilar, red.

The self-similarity matrices for the three test datasets are shown in Appendix X.

4.4 Transitions Based on Similarity Measures

Using a combination of the similarity measures, the best transition for one motion to another is found by the smallest similarity measure among all frames in the two motions, where the transitioning frames are referred to the source and target frame as shown in Figure 4.14.

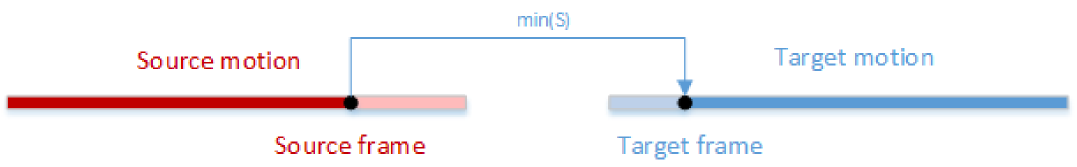


Figure 4.14: Illustration showing a transition between a source and a target motion, where the similarity measure, S, over all frames is minimized.

According to different settings of the weights in Equation 4.3, this could result in different source and target frames. In order to illustrate this, the source frame is for simplicity already established where the best match is found by evaluating all other frames in the video. Choosing frame number 700, means that all values in row or column 700 in the self-similarity matrix are evaluated, where 700 on the x-axis is 0 according to the self-similarity.

Just looking at each of the similarity measures individually, Figure 4.15 to Figure 4.17 shows the similarity measure for all frames compared to frame 700 and the possible target frames with the smallest similarity measures.

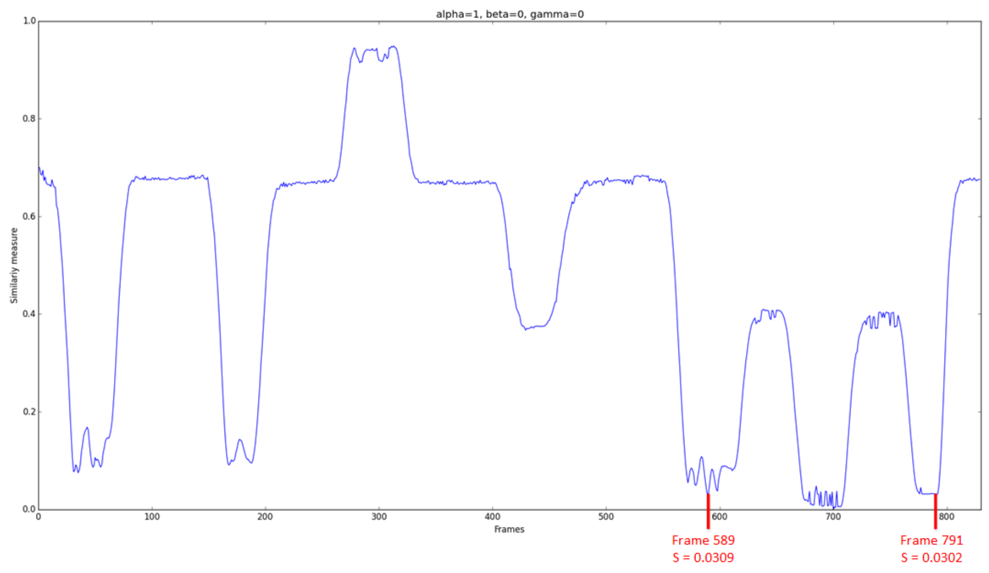


Figure 4.15: Similarity measure for all frames compared to frame 700, where α=1, β=0 and γ=0, i.e. only the skeleton is taken into account.

The depth and colour similarity measures both agree that frame 597 is the best match, whereas the skeleton has frame 791 as the best match. By looking at these two possible target frames as shown in Figure 4.18, it is shown that both frames are similar to the source frame. Yet, the best target frame from manually evaluating the pose and appearance is in this case frame 791, i.e. only looking at the skeleton's similarity measure.

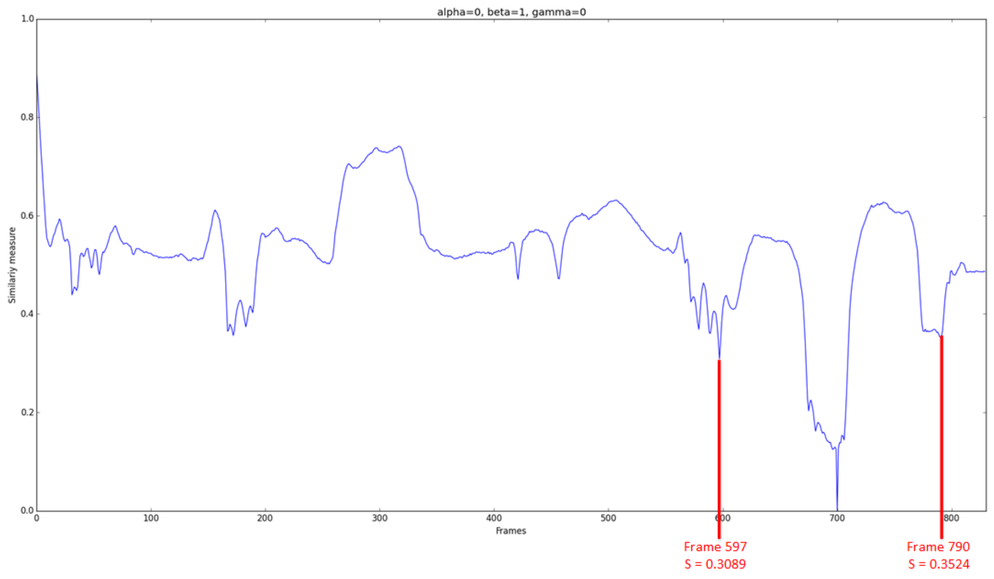


Figure 4.16: Similarity measure for all frames compared to frame 700, where α=0, β=1 and γ=0, i.e. only the depth is taken into account.

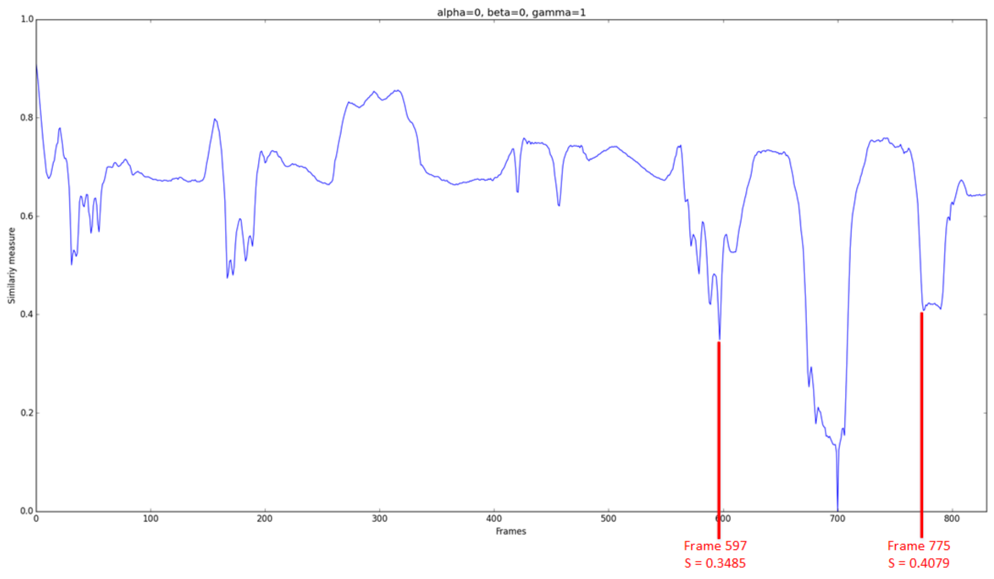


Figure 4.17: Similarity measure for all frames compared to frame 700, where α=1, β=0 and γ=1, i.e. only the colour is taken into account.

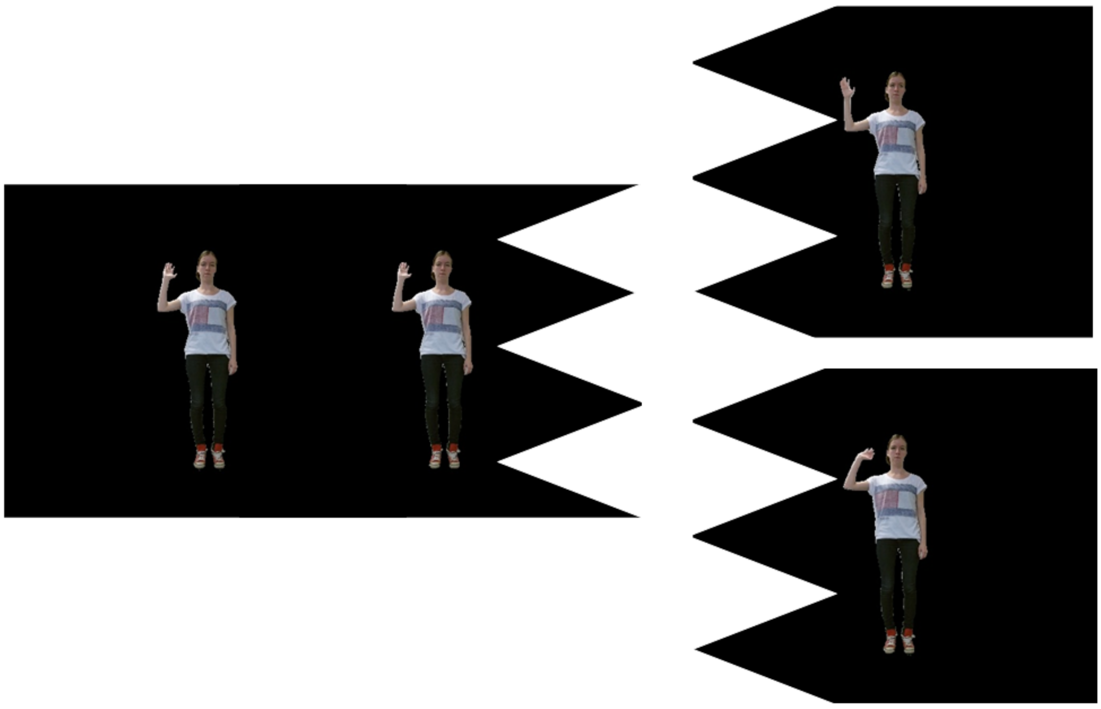


Figure 4.18: Illustration of source frame 700 and the two possible target frames, frame 791 (top) and frame 597 (bottom).

This is not necessarily the case when transitioning between other frames. For example if the source frame was frame 703 shown in Figure 4.19 instead, which is very similar to frame 700. The best match is frame 783 as shown in Figure 4.20, which shows that the skeleton measure can be noisy, and therefore another setting of the similarity measure would be better.

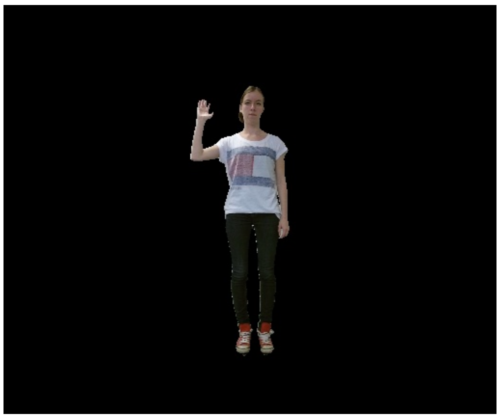


Figure 4.19: Frame 703.

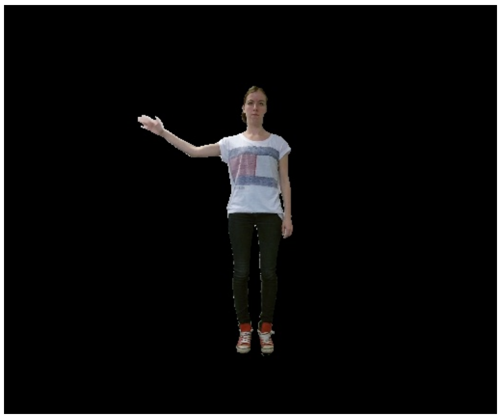


Figure 4.20: Frame 783.

Nevertheless, even though frame 791 is the best target frame between all other frames for transitioning from frame 700, the problem with transitioning to other places in the video, is that the pose and appearance of the person in the source and target frame is still different and can make an unrealistic jump in the resulting video, which needs to be handled.

5. Transitioning Between Motions

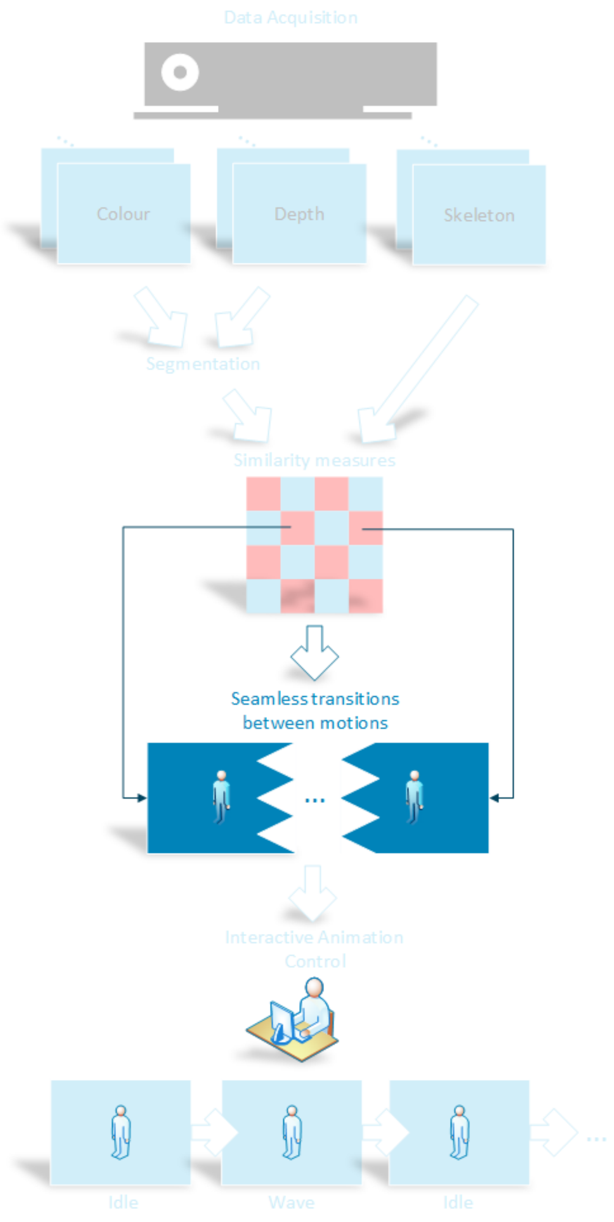


Figure 5.1: Illustration of the overall system design.

The best transition between two motions is determined by the smallest similarity measure, where a source and target frame are found. Since the source and target frames can have a difference in shape and appearance, when transitioning between these frames, it will then make a jump in the resulting video, which is unwanted. Smoothing the transitions will therefore be needed, which can be done by creating new intermediate frames between the source and target frames.

5.1 Smoothing Transitions Using Alpha Blending

It is possible that there is a difference in the shape and appearance of the person when transitioning, and in order to make a smooth transition, new frames between the source and target frame are created. These frames could be made by using alpha blending described in Equation 5.1, where each pixel in a source frame, f_s, and a target frame, f_t, are blended by an α-value giving the resulting blended frame, b.

$$b(x,y) = (1-\alpha)f_s(x,y) + \alpha f_t(x,y) \quad \text{where} \quad \alpha \in [0,1] \tag{5.1}$$

Using the alpha blending, it is possible to mix the two frames either equally, by setting α=0.5, or give the two frames different importance. This is useful when a number of intermediate frames should be created, since it would be possible to gradually change between the source and target frame by changing the α-value as shown in Figure 5.2.

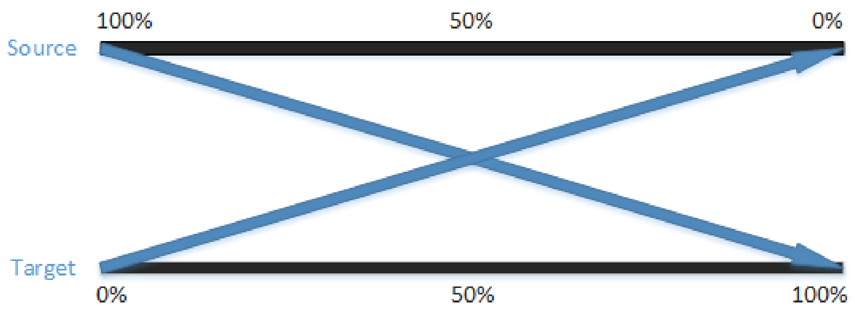


Figure 5.2: Illustration of gradually transitioning from a 100%, i.e α = 1, source frame to a 100%, i.e. α = 0, target frame by changing the value of importance of each frame.

An example of smoothing transitions using the alpha blending between a source and target frame from Figure 5.3 to Figure 5.4, with three intermediate frames are shown in Figure 5.5.

Using just alpha blending for transitioning between a source and target frame will appear smooth whenever there is a difference in appearance, e.g. wrinkles on a shirt. But problems occur when the shape is not similar, which will result in ghosting as shown in Figure 5.5. The final transition might appear smooth, but not necessarily perceived as being realistic. A different approach for creating intermediate frames is therefore exploited in section 5.2.

5.2 Smoothing Transitions Using Optical Flow

The problem with using alpha blending occurs when the shape between the source and target frame was too different. So creating new frames with intermediate poses between the source

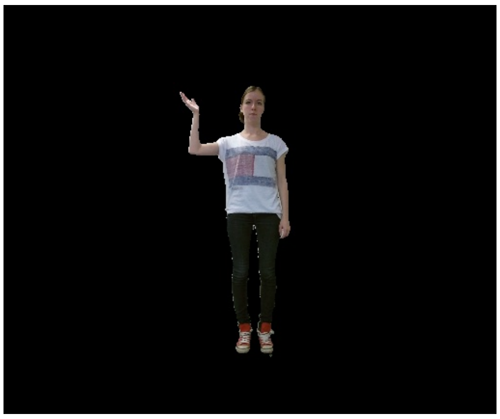


Figure 5.3: Source frame - frame 603.

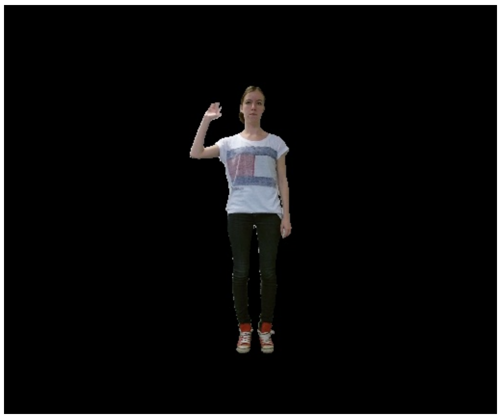


Figure 5.4: Target frame - frame 611.

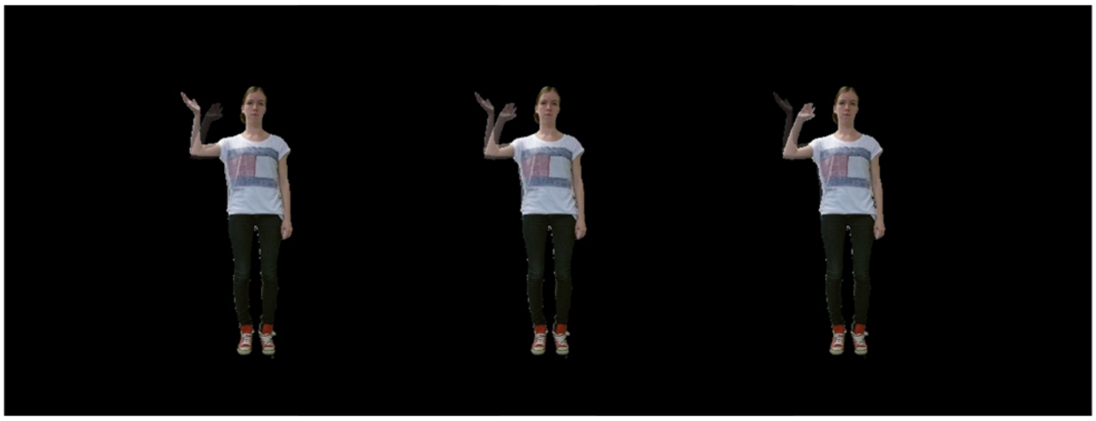


Figure 5.5: Three intermediate frames between the source and target frame from Figure 5.3 and Figure 5.4 made with alpha blending. From left α = 0.25, 0.5 and 0.75.

and target frame instead could, compared to the alpha blending, increase the smoothness and perceived realism of the transition. Inspired by (Casas et al., 2013) and (Fechteler et al., 2014), where new 3D meshes are created by interpolation between the source and target frames, a novel approach is therefore suggested, where the displacement in shape between the frames is estimated by optical flow and then used for create intermediate poses by moving the pixels accordingly.

The optical flow between two frames estimates the changes in illumination whenever a camera is moved or an object is displaced. A number of different optical flow algorithms exist, but what they all have in common is the brightness constancy constraint, where the pixels are assumed to have the same intensity values when displaced. Having this constraint it is possible to estimate the displacement by searching for similar correspondences, as illustrated in Figure 5.6.

This means that for each pixel in the source frame will have a flow vector which determines the estimated flow towards the target frame. The displacement of the pixels can therefore be described in Equation 5.2.

$$f_{target}(x,y) \sim f_{displaced}(x,y) = (f_{source}(x) + flow(x), f_{source}(y) + flow(y)) \tag{5.2}$$

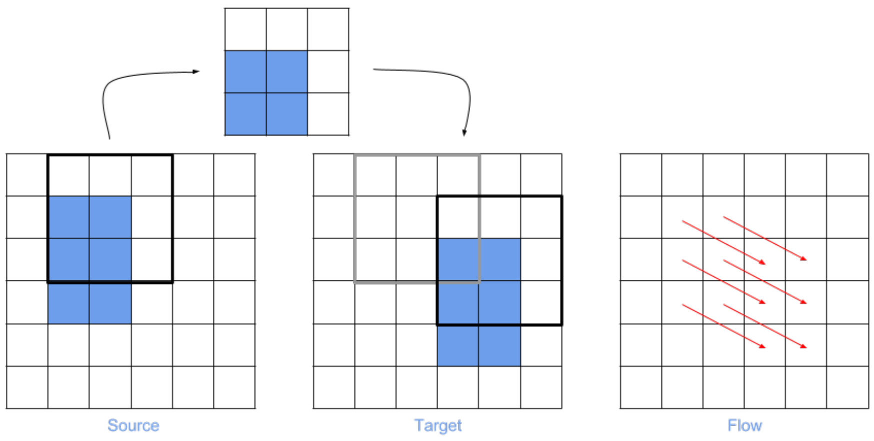


Figure 5.6: Illustration of the optical flow between two frames with a displaced box.

Since the optical flow estimates the displacements in a specified size of neighboorhood, it assumes that the displacement is not too big. But in order to increase robustness of the estimation and also estimate larger displacement than the size of the neighbourhood an image pyramid could be used. An illustration of such is shown in Figure 5.7

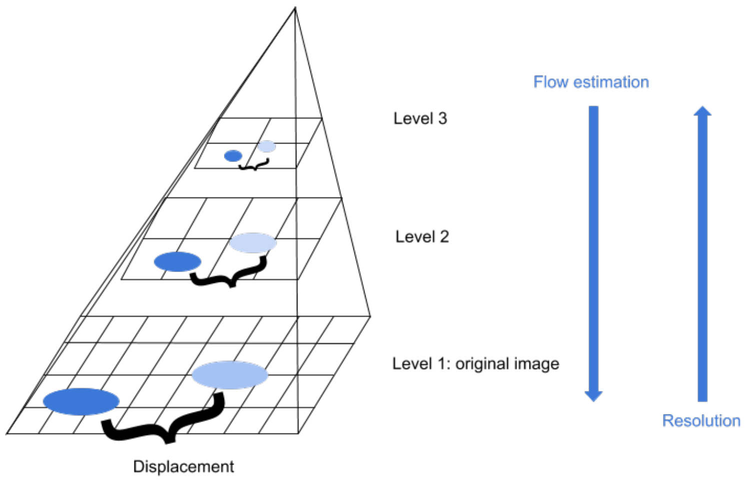


Figure 5.7: Illustration of an image pyramid with 3 levels, where the higher the level the smaller is the displacement.

The first and lowest level in the pyramid is the original frame, and at each level the frame is downsampled. The estimation of the flow then starts at the top level, i.e. at the lowest resolution, and then uses the estimated displacement from that level as an initial guess in the next level and so forth.

Since the flow is estimated by looking at the illumination changes, the frames are converted from RGB to gray-scale. This is done by weighting the different colour channels as described in Equation 5.3 (Moeslund, 2012).

$$I = w_R \cdot R + w_G \cdot G + w_B \cdot B \quad \text{where} \quad w_R + w_G + w_B = 1 \tag{5.3}$$

The weights for the colour channels could be assigned if a colour channel had more importance than another, e.g. if the person was only wearing red clothing, hence a larger importance for the red colour channel. But since each character in the database wears different colour clothing, the values are instead assigned according to the standardized weights for the individual colour channels optimized for the human visual system, which are w_R = 0.299, w_G = 0.587 and w_B = 0.114.

A common way of estimating the optical between two frames is by using the iterative Lucas-Kanade optical flow, where the flow is estimated by finding corresponding intensity values in the other frame where the displacement between the two frames are minimized (Lucas et al., 1981). Since the Lucas-Kanade algorithm is sparse it only needs a set of features present in both frames to estimate the flow. These features are extracted by a keypoint detector such as SURF, SIFT or Harris corner.

But instead of extracting features to track and determine how to move the pixels between the keypoints when creating new poses, a possibility is to estimate the flow for every pixel, hence using a dense optical flow. This approach is also used for alignment of textures to a geometric proxy mesh in (Casas et al., 2014), where the Farneback Optical Flow is used.

5.2.1 Farneback's Optical Flow

The dense optical flow using Gunner Farneback's algorithm is based on approximating each neighbourhood as a quadratic polynomial as described in Equation 5.4, where A is a symmetrical matrix, b is a vector and c is a scalar (Farnebäck, 2003).

$$f(x) \sim x^T A x + b^T x + c \tag{5.4}$$

This means that all neighbourhoods, e.g. in Figure 5.6, are described by a quadratic polynomial, where the coefficients are found by a least square fit. Using these polynomials to compare the neighbourhood with the other neighbourhoods in the other frame, the best fit for the displacement can be found. So for example displacing a polynomial $f_1(x)$ with d, a new polynomial $f_2(x)$ is constructed (Farnebäck, 2003):

$$\begin{aligned} f_1(x) &= x^T A_1 x + b_1^T x + c_1 \\ f_2(x) &= f_1(x-d) = (x-d)^T A_1 (x-d) + b_1^T (x-d) + c_1 \\ &= x^T A x + (b_1 - 2A_1 d)^T x + d^T A_1 d - b_1^T d + c_1 \\ &= x^T A_2 x + b_2^T x + c_2 \end{aligned} \tag{5.5}$$

Where due to the brightness constancy constraint, the coefficients can be equated as:

$$\begin{aligned} A_1 &= A_2 \\ b_2 &= b_1 - 2A_1 d \\ c_2 &= d^T A_1 d - b_1^T + c_1 \end{aligned} \tag{5.6}$$

By assuming that A is non-singular, meaning that it is invertible, the displacement of d can therefore be solved as:

$$\begin{aligned} b_2 &= b_1 - 2A_1 d \\ 2A_1 d &= -(b_2 - b_1) \\ d &= -\frac{1}{2}A_1^{-1}(b_2 - b_1) \end{aligned} \tag{5.7}$$

In principle, searching for the polynomial with the best fit could be done in the whole frame. But since this turns out to be too noisy, it is assumed that the pixels only undergoes a small displacement, which means searching for the best fit by minimizing displacements together with the error between the coefficients is only done within a surrounding neighbourhood of pixels.

Farneback's optical flow is implemented in OpenCV as *calcOpticalFlowFarneback* (documentation), which takes eight input values as listed below, where the output of each pixel is a flow vector, where the magnitude of 1 corresponds a displacement of one pixel horizontal or vertically.

- Source frame
- Target frame
- Image pyramid scale
- Levels in image pyramid
- Kernel size of average filter
- Iterations in each pyramid level
- Kernel size of the searching neighbourhood
- Standard deviation for the kernel of the searching neighbourhood

According to default settings in the OpenCV implementation (documentation), a classical image pyramid is used with a value of 0.5, which means the next layer is half the size of the previous layer. Further the size of the neighbourhood in the search for the displacement is set to 5, with the standard deviation set to 1.1. In order to set the rest of the parameters, i.e. levels in the pyramid, average kernel size, and the number of iterations in each pyramid level, an evaluation is done in subsection 5.2.2.

5.2.2 Evaluation of Optical Flow Settings

An evaluation of Farneback's optical flow by changing its parameters is done in order to set appropriate paramters as well as understanding its restrictions. The evaluation is done by analyzing the flow vectors for a known movement, which in this case is moving an arm up from frame 283 to 286 as shown in Figure 5.8, where the difference is shown by alpha blending.

Looking at the implementation of the Farneback's optical flow in MATLAB, it has the same default values as the OpenCV implementation. Further it has also default values for the number of levels in the pyramid set to 3, the averaging kernel size set to 15x15, and the

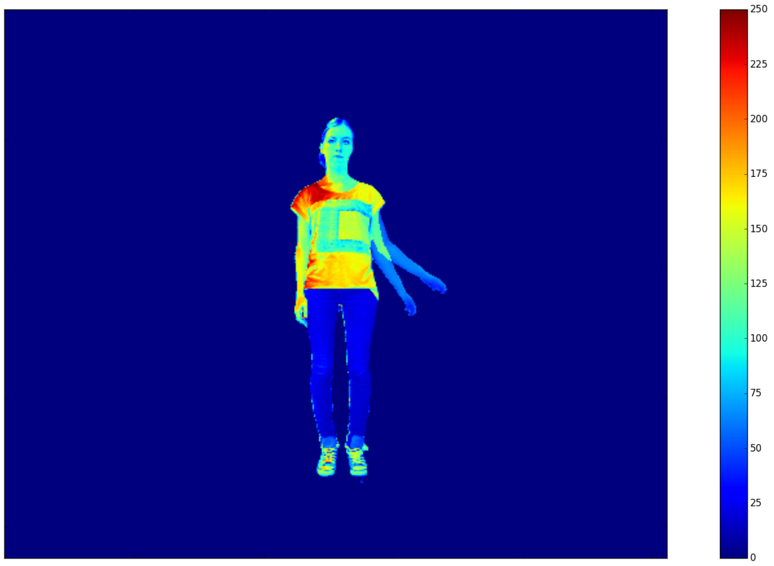


Figure 5.8: The difference between the frame 283 and 286 shown by alpha blending, which is used for evaluation of the settings for the optical flow.

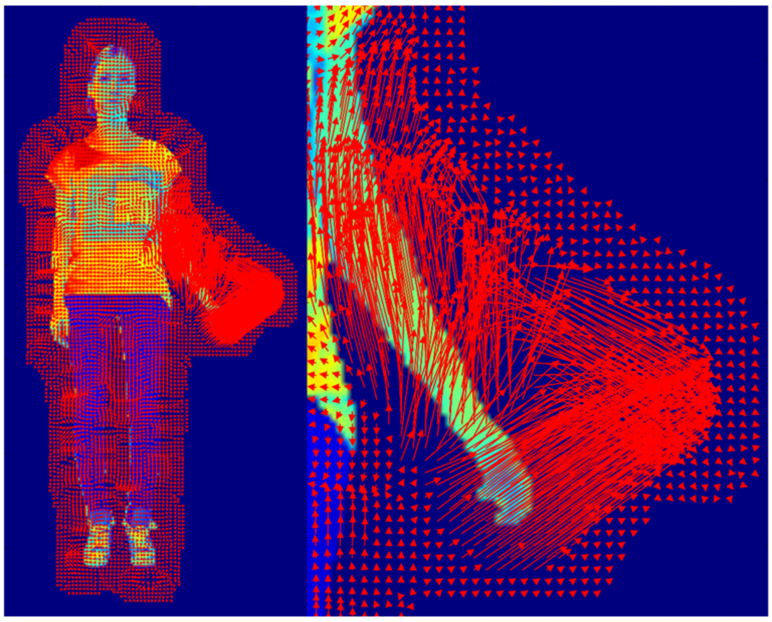


Figure 5.9: The estimated flow a with the default MATLAB settings.

number of iterations in each level of the pyramid set to 3. Using these settings will result in a flow shown in Figure 5.9.

Choosing less levels in the pyramid, e.g. 1, it will in this case make the flow estimation worse as shown in Figure 5.10, which is because the displacement between the pose is too big.

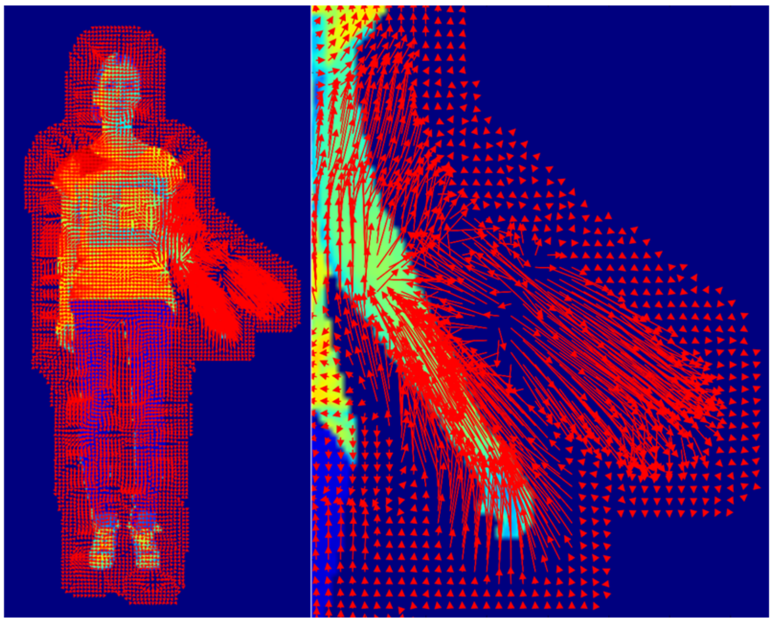


Figure 5.10: Estimated flow with levels=1, window=15, iterations=3, between frame 283 and 286

Keeping in mind that the pyramid levels enables the optical flow to track large displacements, would mean that if the displacement is less it would be possible to give a better estimate of the flow. This is shown by estimating the flow between fame 283 and 284 shown in Figure 5.11.

Choosing a higher number of pyramid levels than 3 and therefore downsampling the frame would not necessarily add more robustness to the frame, since the frame at the last level would be very small. Setting the number of pyramid levels to 3, would in this case be an appropriate value.

As shown in Figure 5.9 noise appears in the form of the flow vectors not moving uniformly, and using a smaller averaging kernel size on the frame before estimating the flow will only add more noise as shown in Figure 5.12, where a kernel size of 5x5 is used.

By using a larger kernel size could eliminate some of the noise and also make the flow smoother and more uniform as shown in Figure 5.13, where a kernel size of 25x25 is used. Compared to Figure 5.9, using a larger kernel size will eliminate noise and also improve the robustness of the flow estimation, which is why a kernel size of 25x25 would be used.

According to setting the number of iterations, the algorithm performs a search at each level to find the best match and will after a number of iterations converge. So choosing a higher number of iterations and make sure that the algorithm reach convergence or closer to convergence, could increase the robustness to noise. Choosing the number of iterations of 10 would result in a flow as shown in Figure 5.14. Comparing this to the estimated flow in Figure 5.13, then choosing a higher number of iterations is an improvement according to the noise in the areas where the structure is very uniform, e.g. the pants. This also means

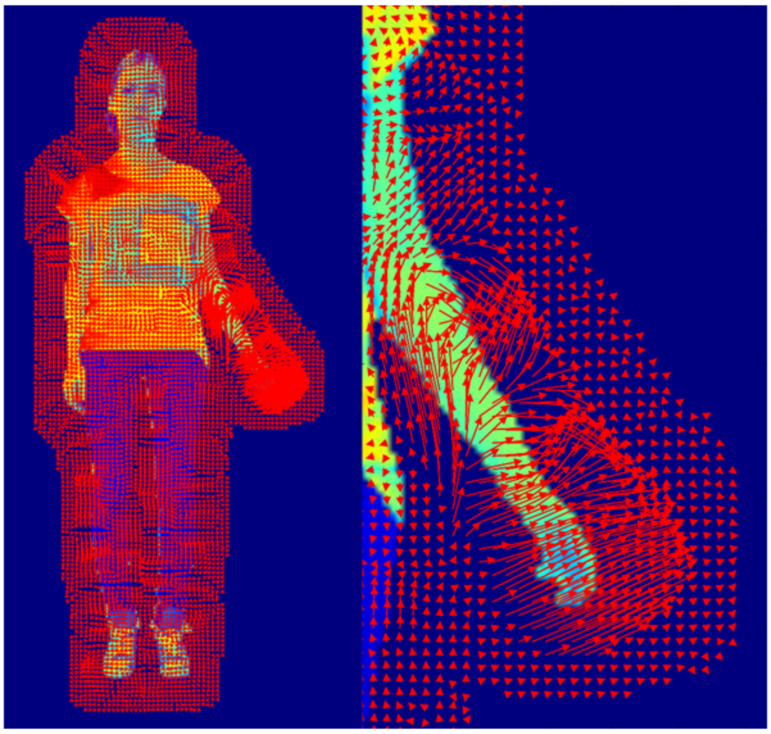


Figure 5.11: Estimated flow with levels=1, window=15, iterations=3, between frame 283 and 284

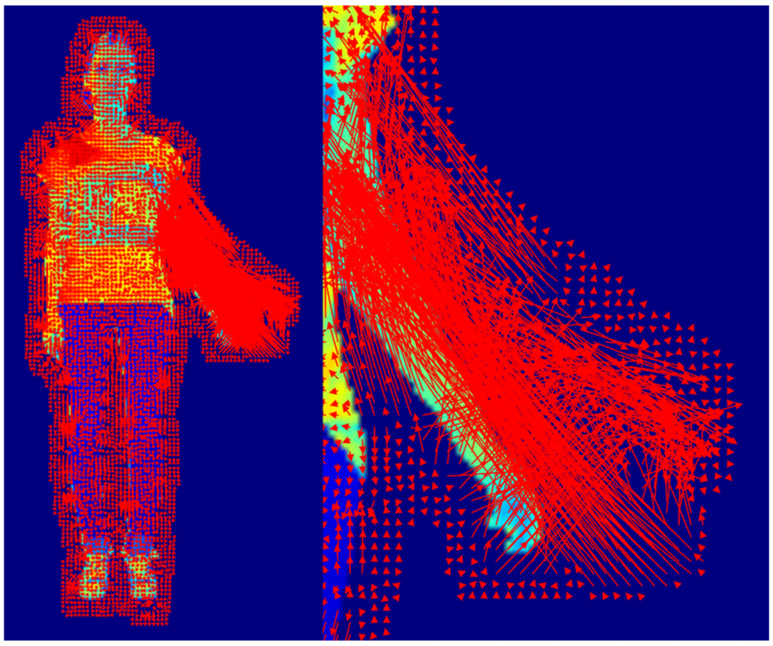


Figure 5.12: Estimated flow with levels=3, window=5, iterations=3, between frame 283 and 286

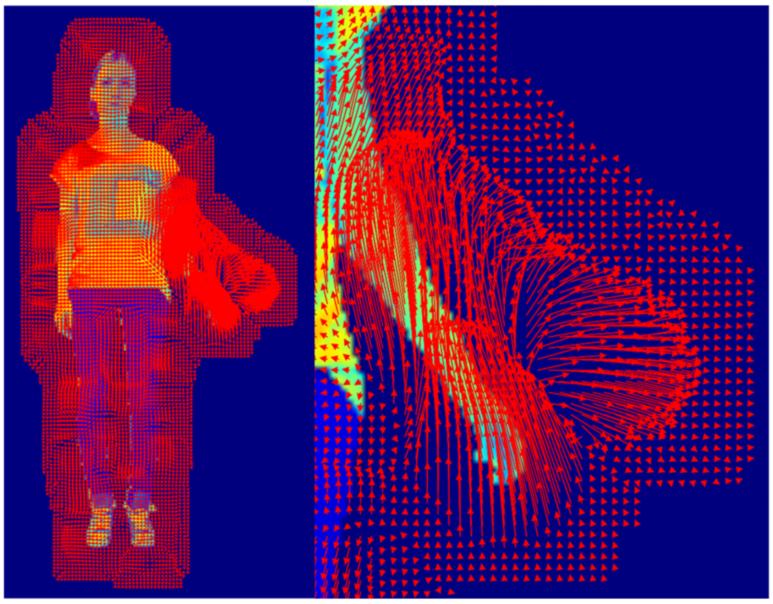


Figure 5.13: Estimated flow with levels=3, window=25, iterations=3, between frame 283 and 286

that setting the number iterations to only 3, will in this case not be enough for making the algorithm to converge.

The resulting parameter settings used in the Farneback's optical flow are therefore:

- Pyramid scale = 0.5
- Levels in pyramid = 3
- Kernel size of average filter = 25
- Iterations in each pyramid level = 10
- Kernel size of pixel neighbourhood = 5
- Standard deviation of kernel = 1.1

The flow estimation using these parameters can vary according to the size of the person, the frame and the size of the displacement field. The optical flow is restricted to only estimate the flow between small displacements. If the poses differs too much, it will be hard to make a correct flow estimation. In order not to have positions that vary too much, is something that should be minimized by a combination of the parameters in the similarity measure, which will then find a good transition between a source and target frame with a small displacement.

5.3 Interpolation of Pixels

Having estimated the dense optical flow between the source and target frame, a position for where every pixel should go is provided, which can be used for making a linear interpolation of the pixels.

A problem occurs, since the flow vectors are not necessarily integers, which means the destination for a pixel becomes a position that is not possible. In order to handle this problem,

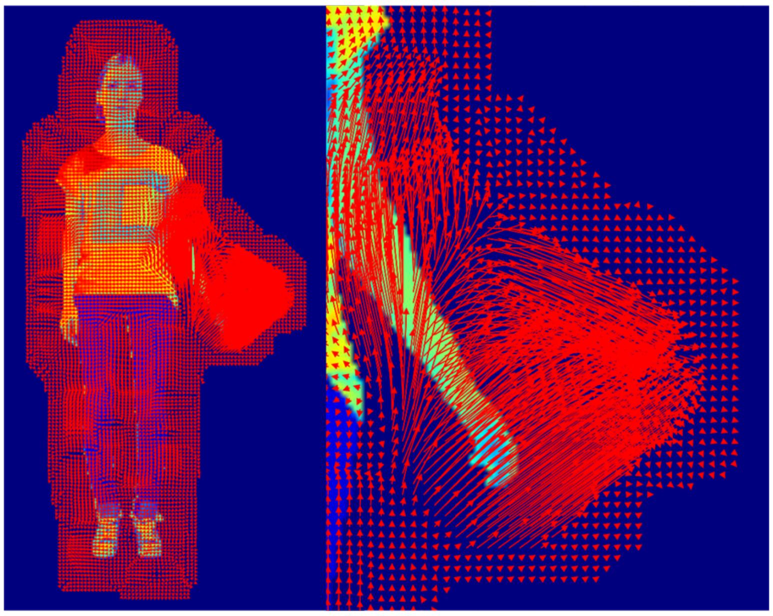


Figure 5.14: Estimated flow with levels=3, window=25, iterations=10, between frame 283 and 286

the values are rounded off to the nearest possible pixel, also called zeroth-order interpolation (Moeslund, 2012). E.g. flow vectors like (2.3, 3.7) are interpolated to (2, 4).

Gradually moving the pixels in the direction of the flow, will then create intermediate poses. This is done by adding each pixel a percentage of its flow vector as in Equation 5.2, which will result in frames shown in Figure 5.15 where the pixels are moved 0%, 25%, 50% and 75% in the direction of the flow, i.e. forward mapping is performed.

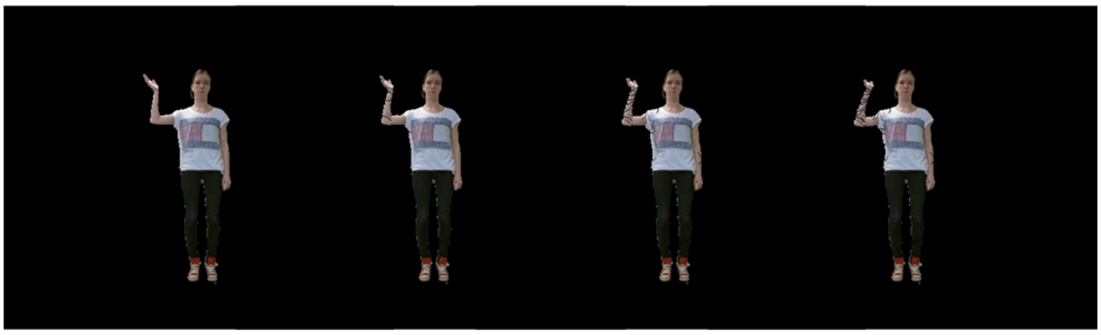


Figure 5.15: Example of where pixels are gradually moved in the direction of the flow between frame 603 and 611. From left - 0%, 25%, 50% and 75% of the flow vectors.

This works well for creating intermediate poses between a source and target frame, but holes in the resulting frames appears. This is can be due to rounding errors when doing the zeroth-order interpolation of the flow vector, but it also happens when a pixel does not have a destination of a flow vector from another pixel. This makes sense when e.g. moving the arm, where the previous position of the arm should be empty in the new frame. But this can also happen inside the body, which must be handled.

Usually the problem with holes in the output frame is handled by performing backward mapping (Moeslund, 2012), where pixel values in the output frame are found in the input frame using the inverse transformation, which is illustrated in Figure 5.16.

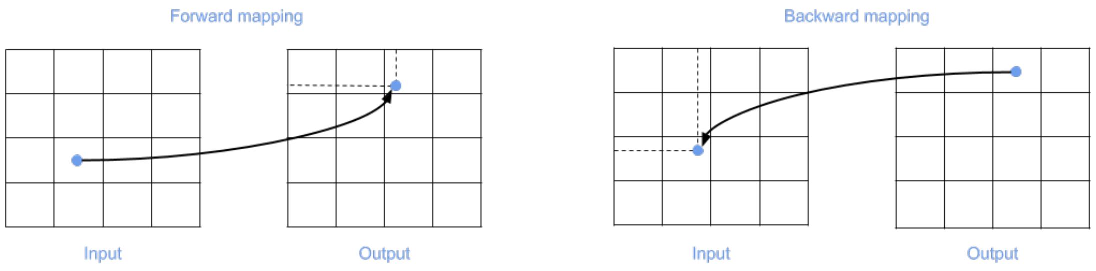


Figure 5.16: Illustration of performing forward and backward mapping.

This could work since pixels have its own transformation from the optical flow vector. But a problem is still if a pixel does not have a destination from the optical flow, meaning it has no transformation thus no inverse transformation. In order to give this pixel a colour value, would then be by evaluating the surrounding pixels to either see what their inverse transformations are or what their colour value is in the output frame. This is the same issue found in forward mapping. Due to the extra computational time to perform backward mapping and still have holes in the output frame, which needs to be filled, the forward mapping is therefore chosen. So the pixels that need to be filled, are found by evaluating the surrounding pixels in the output frame.

5.3.1 Post-processing

In order to fill the holes which appears when moving the pixels in the direction of the flow, the surrounding pixels are evaluated by applying a filter. One way of using a filter to fill out the holes is to apply it to the whole frame as shown in Figure 5.17 where a mean filter of size 3x3, 5x5 and 7x7 is applied to frame where the pixels are moved 75% in the direction if the flow from Figure 5.15. The problem with using this approach is that the holes a not completely filled, and choosing a larger filter would only create additional blur to the resulting frame.

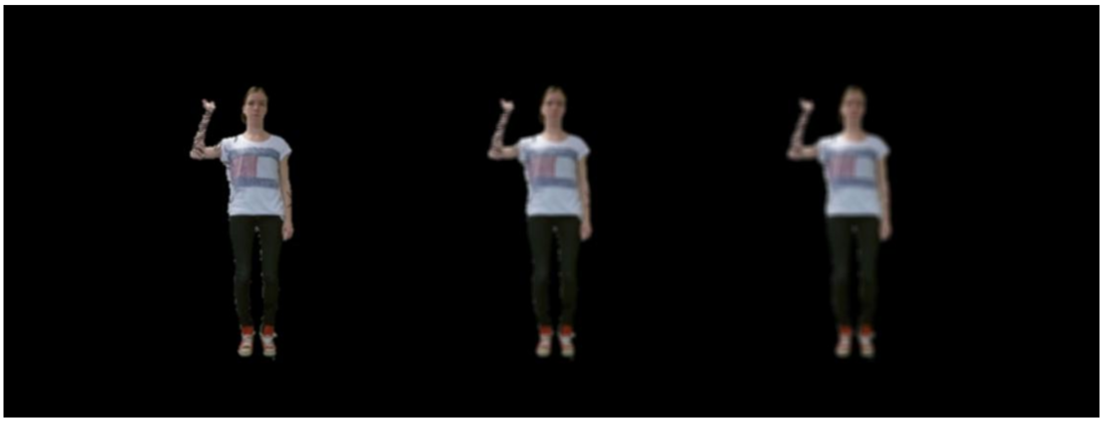


Figure 5.17: Applying a mean filter to moving the pixels 75% in the direction of the flow from Figure 5.15. From left - kernel sizes 3x3, 5x5 and 7x7.

In order to preserve the person in frame without blur, a search through that frame for holes, i.e. black pixels ([0,0,0]) is done. These pixels are then replaced with the colour value from the one of the filtered frame in Figure 5.17 at the same position. The resulting frame with the different kernel sizes are shown in Figure 5.18.

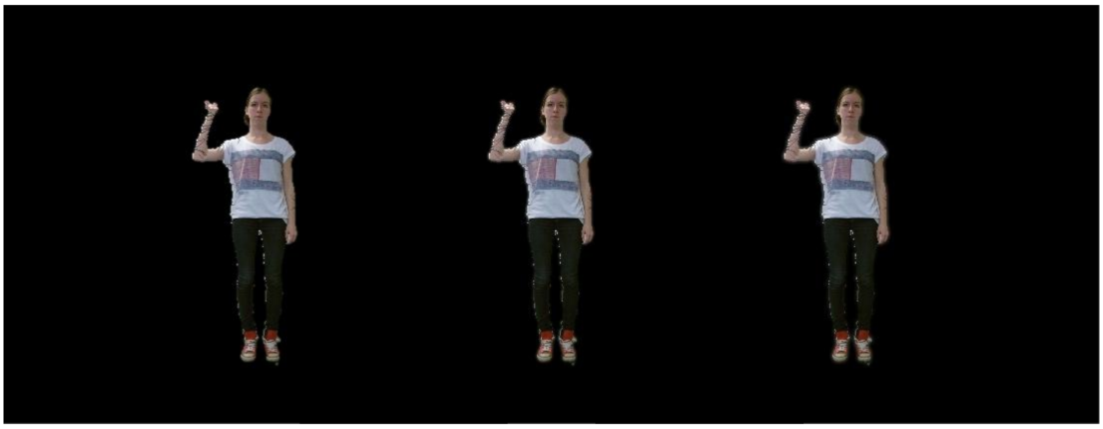


Figure 5.18: Replacing holes in with the values from the filtered frames in from Figure 5.17. From left - kernel sizes 3x3, 5x5 and 7x7.

The holes are still not completely filled, since the mean filter takes the value of the hole into account when calculating the mean value. One way to solve this is to create a new filter that not does not use this value. This means instead of using a mean filter of, e.g. size 3x3 as in Equation 5.8.

$$A = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{5.8}$$

The filter will instead be as in Equation 5.9.

$$A = \frac{1}{8}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{5.9}$$

This only solves the problem when none of the surrounding pixels within the applied filter, does not contain holes. This means an adaptive filter should be made according to how the holes are placed, which is a solution that is computationally demanding. Another issue when using a mean filter, is that the resulting colour might not have the same value or be a very different value according to the surrounding pixels. Using a median filter instead can solve these problems, since a colour value is chosen between already existing values. The result from using a median filter on the frame instead is shown in Figure 5.19.

Compared to the mean filtered frames in Figure 5.18, using the median filter does provide a better resulting frame. The kernel size of 3x3 as shown in Figure 5.19 is too small, since it does not fill all holes, but using a filter of 5x5 or above will. Addtionally, the kernel should not be too large either. When zooming in on the frames where a kernel size of 5x5 and 7x7 are used in Figure 5.20, it is shown that using a kernel of 7x7 or higher adds a extra pixels to the body, most visible at the neck, and also a more blurred output. A median filter with a kernel size of 5x5 is therefore used.

After filling out the holes when moving the pixels in the direction of the flow, a problem occurs, because moving the pixels in the full direction of the flow still vary according to the actual target frame as shown in Figure 5.21, which could create a jump in the resulting video.

In order to solve that problem, the flow from the target frame to the source frame is also

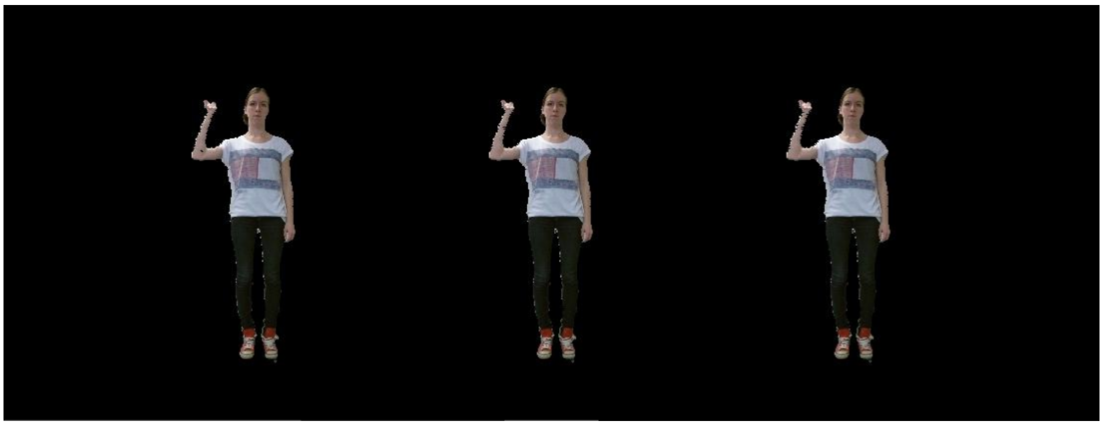


Figure 5.19: Replacing holes in with the values from median filtered frames. From left - kernel sizes 3x3, 5x5 and 7x7.

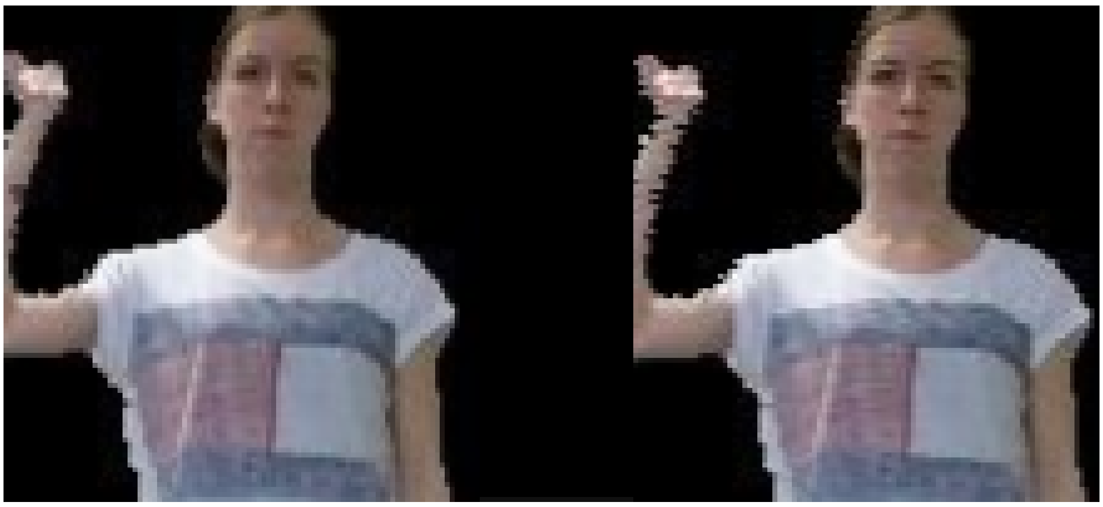


Figure 5.20: Zoomed frames of using the kernel sizes 5x5 and 7x7 from Figure 5.19.

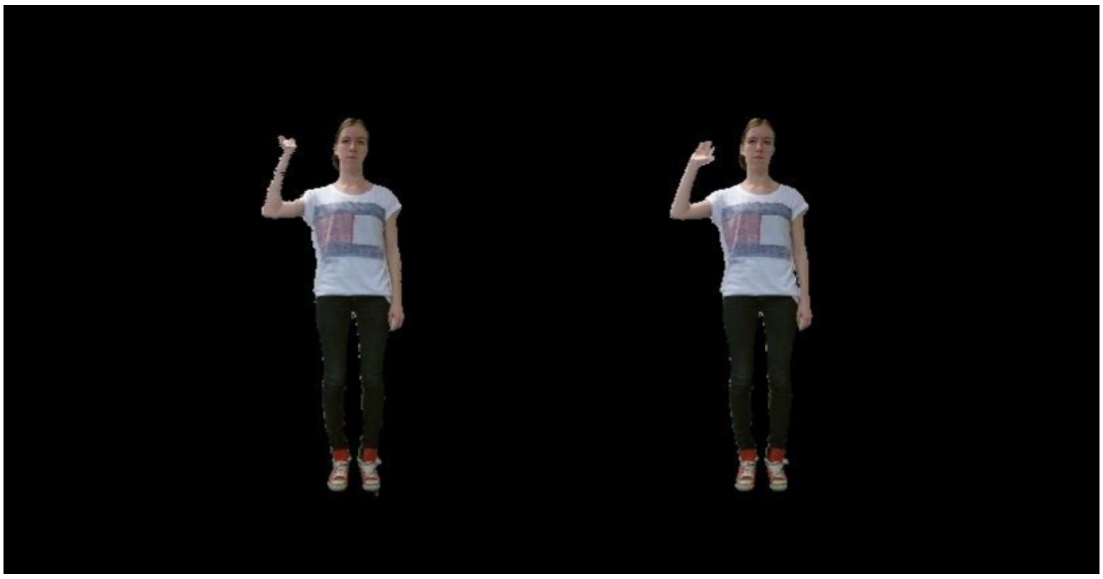


Figure 5.21: The moved pixels 75% in the direction of the flow estimated from source to the target frame, and the target frame 611.

used as shown in Figure 5.22, where the pixels are moved in each direction of the flow 25% , 50% and 75% of the way. Here it shows that the frames in the top row are more close to the source frame, and the same way for the bottom row have frames more close to the target frame. A solution is then to merge top and bottom row by alpha blending.

The resulting intermediate frames will therefore be a blend of the top and bottom row in Figure 5.22.

5.4 Transitioning Overview

The flow of creating intermediate frames when transitioning, implemented as *intermediate-Frames.py*:

- Find Farneback's optical flow from source to target and target to source
- Move gradually pixels in the direction of their individual flow vectors
- Fill out holes by using a median filter
- Blend the frames from each direction.

The amount of intermediate frames are set to 5, which are further evaluated in chapter 7.

An example for calculating the middle frame between a source and target frame, i.e. when pixels are moved 50% along their flow vectors, is illustrated in Figure 5.23.

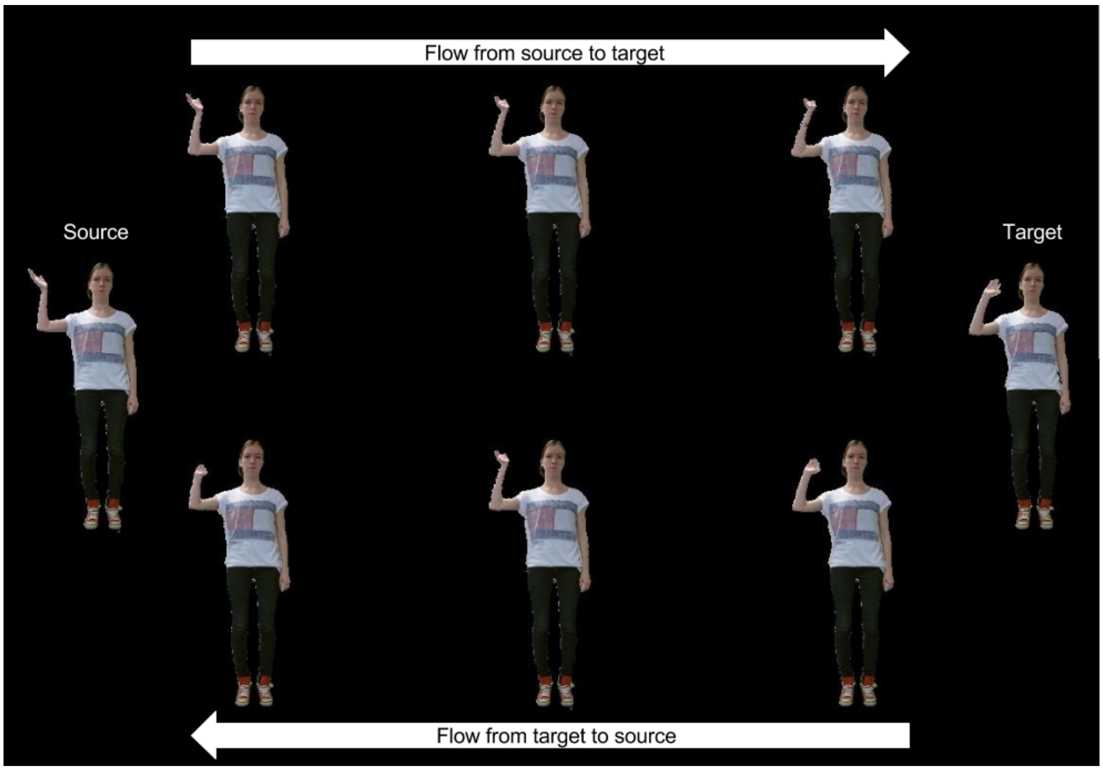


Figure 5.22: The flow from source to target (top row) and from target to source (bottom row), where the pixels are moved 25%, 50% and 75% in the direction of the flow.

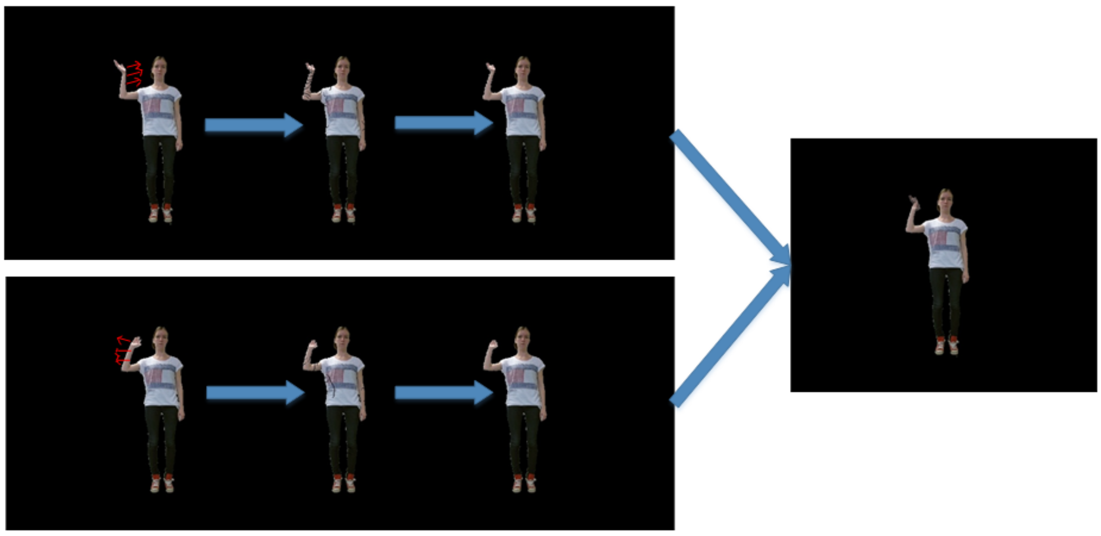


Figure 5.23: Illustration of how an intermediate frame between a source frame (frame 603) and a target frame (frame 611) is made.

6. Animation Control

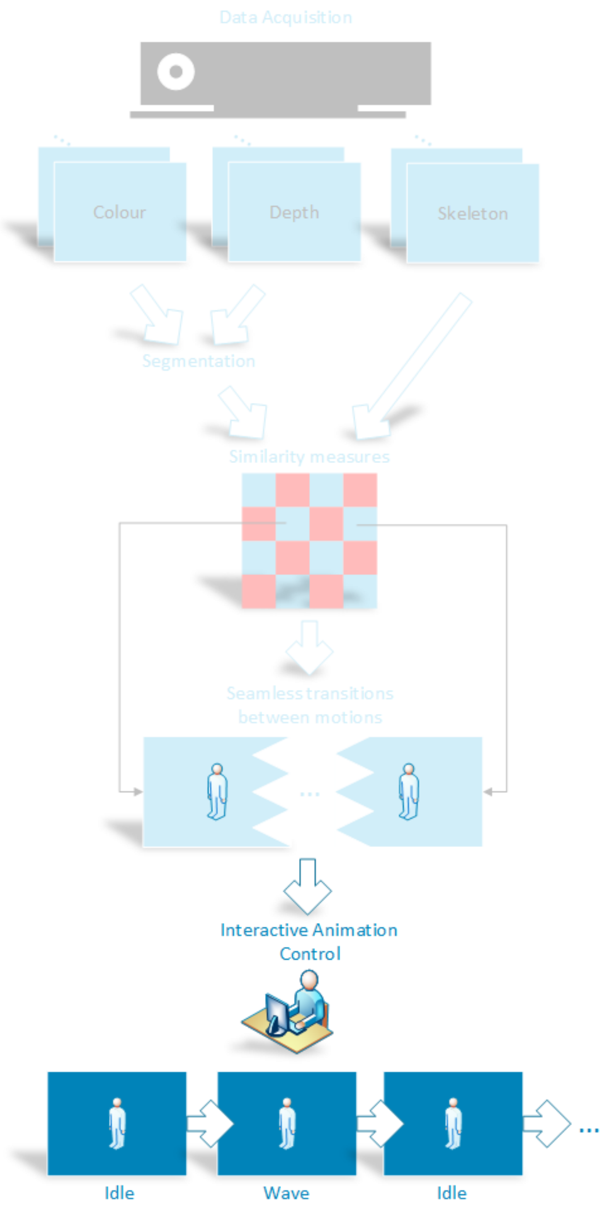


Figure 6.1: Illustration of the overall system design.

A program for a user to create videos of characters performing a chosen set of motions is in this chapter explained. Three possible characters can be chosen, Cathrine, Marco and Leonardo, which each has different motion options.

The Cathrine character can perform three different dance motions, which are wave, clap and twist as shown in Figure 6.2.

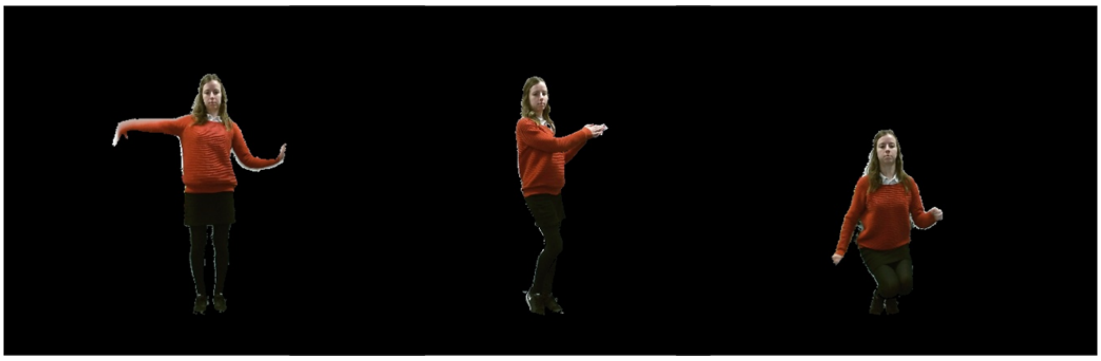


Figure 6.2: The motions of the Cathrine character. From left - wave (frame 70), clap (frame 220) and twist (frame 300).

The Marco character can perform four different motions, which are wave with one hand, wave with both hands, stretch to the right and stretch to the left as shown in Figure 6.3.

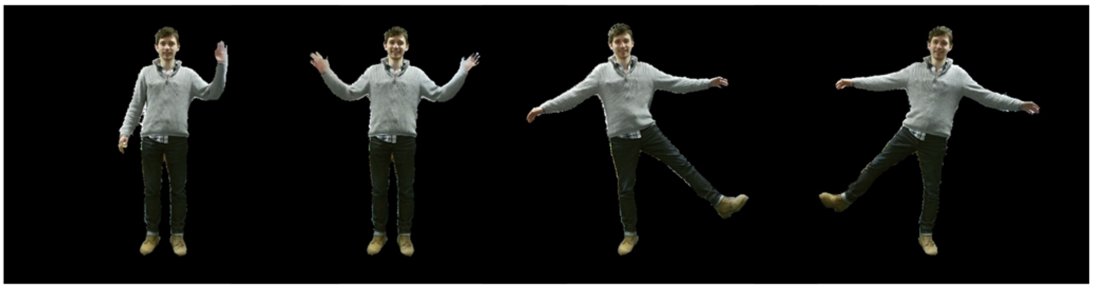


Figure 6.3: The motions of the Marco character. From left - wave one hand (frame 80), wave both hands (frame 150), stretch right (frame 200) and stretch left (frame 245).

The Leonardo character can perform four different goalie motions, which are save top left corner, save top right corner, save bottom left corner and save bottom right corner as shown in Figure 6.4.

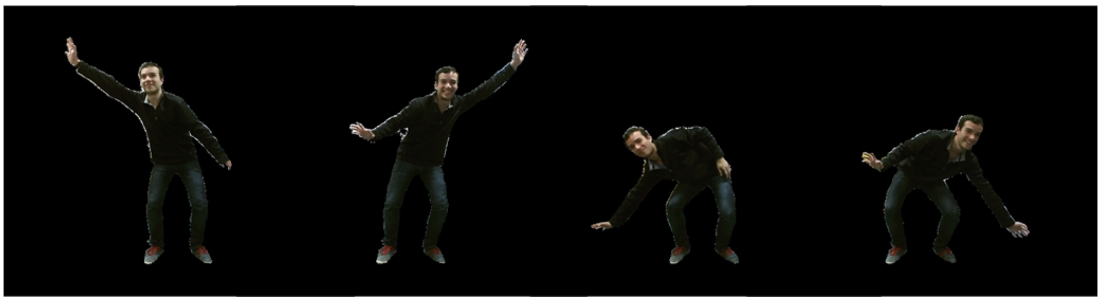


Figure 6.4: The motions of the Leonardo character. From left - save top left (frame 90), save top right (frame 150), save bottom left (frame 230), save bottom right (frame 630).

6.1 Choosing a Character

The first part of the program, the user has to choose one of the three characters by a number, i.e. Cathrine=1, Marco=2 or Leonardo=3. If a different number is pressed, then the Cathrine character is chosen as default.

Each character has a corresponding file containing meta data about its motions which is loaded in to the program when choosing a character. This meta data file contains an array of the possible motion names, an array of where the motions are divided in the original video, a database of the motions and finally a motion graph determining how to shift between motions.

Therefore, when choosing a character, the program loads the following:

- An array of motion names
- An array of where motions are divided in the video
- The motion graph
- The self-similarity matrix of the skeleton
- The self-similarity matrix of the depth
- The self-similarity matrix of the colour
- The segmented colour frames

The similarity matrices are used for determining the best transition between the motions.

6.2 Choosing a Sequence of Motions

When the above are loaded into the program, the possible motions from the array of motion names are then presented to the user. This is done by looping through the list of names. The output for each of the characters presented to the user are shown in Figure 6.5.

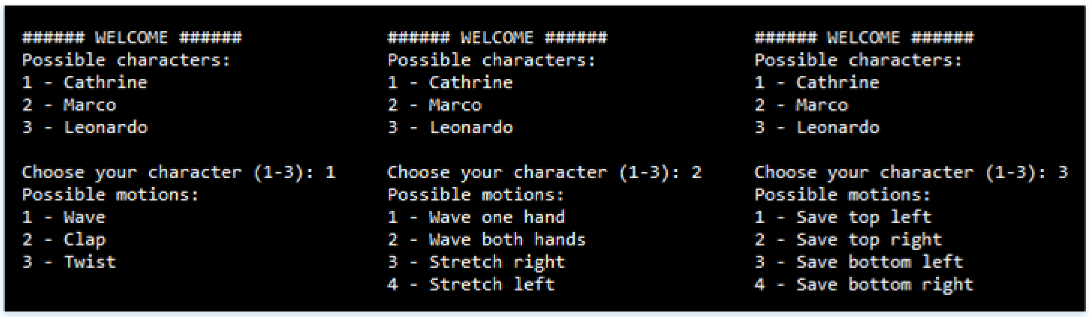
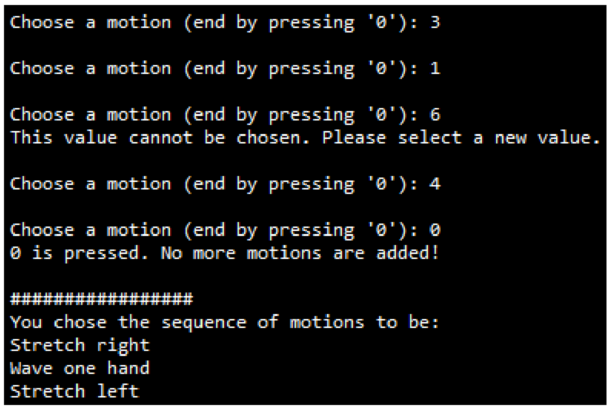

```
###### WELCOME ######            ###### WELCOME ######            ###### WELCOME ######
Possible characters:             Possible characters:             Possible characters:
1 - Cathrine                     1 - Cathrine                     1 - Cathrine
2 - Marco                        2 - Marco                        2 - Marco
3 - Leonardo                     3 - Leonardo                     3 - Leonardo

Choose your character (1-3): 1   Choose your character (1-3): 2   Choose your character (1-3): 3
Possible motions:                Possible motions:                Possible motions:
1 - Wave                         1 - Wave one hand                1 - Save top left
2 - Clap                         2 - Wave both hands              2 - Save top right
3 - Twist                        3 - Stretch right                3 - Save bottom left
                                 4 - Stretch left                 4 - Save bottom right
```

Figure 6.5: The possible motions to choose from when a character is chosen.

The user then chooses a sequence of motions, the same way as choosing a character, i.e. by a number. This runs as a loop as shown in Figure 6.6, where the chosen number for each loop is appended to a list. If a number is outside the possible list of numbers, the loop starts over again without appending a number to the list. In order to end appending additional motions,

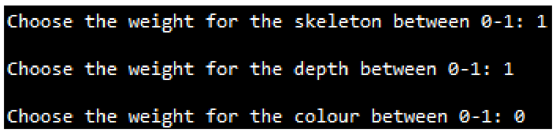


```
Choose a motion (end by pressing '0'): 3

Choose a motion (end by pressing '0'): 1

Choose a motion (end by pressing '0'): 6
This value cannot be chosen. Please select a new value.

Choose a motion (end by pressing '0'): 4

Choose a motion (end by pressing '0'): 0
0 is pressed. No more motions are added!

################
You chose the sequence of motions to be:
Stretch right
Wave one hand
Stretch left
```

Figure 6.6: An example of choosing motions for the Marco character.

0 is pressed. Hereafter the sequence of the chosen motions are presented to the user as shown Figure 6.6.

Before creating the animation, the weights for the similarity measure in Equation 4.3 are also selected by the user as shown in Figure 6.7. These values are then used for determining the best transition between the chosen motions.



```
Choose the weight for the skeleton between 0-1: 1

Choose the weight for the depth between 0-1: 1

Choose the weight for the colour between 0-1: 0
```

Figure 6.7: An example of choosing the weights for the similarity measure, where α=β=1 and γ=0.

6.3 Determine Transitions

Having the list of motions which the chosen character should perform together with the weights for the similarity measure, the best transition between each motion are found.

For each transition there is a source motion and a number of possible target motions as illustrated in Figure 6.8. In order to find the best transition with the smallest similarity measure, all frames in the source motion are compared with all frames, as in this example, two target motions. This leads to two similarity measures, S_1, and S_2, which each determines the transition from the source motion to one of the target motions. The final transition is chosen by the smallest of those two similarity measures, which gives the transitioning source and target frame marked by the black dot.

This search for the best source and target frame between two transitioning motions is implemented as *findSourceTargetSingleMulti*.

The chosen target motion then becomes the new source motion, and going through the motion graph, a new set of possible target motions are available. The search for the next transition is done again in the same way as illustrated in Figure 6.9, where all frames in the red bar are appended to the video.

Between each transition, 5 intermediate frames are generated and inserted in the video sequence. When determining the source and target frame, it is possible that the transition

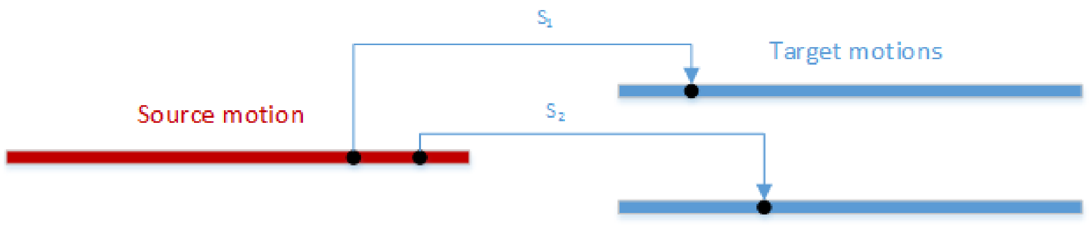


Figure 6.8: Illustration showing possible transitions between one source motion and number of possible target motions with their corresponding similarity measures, S_1, and S_2.

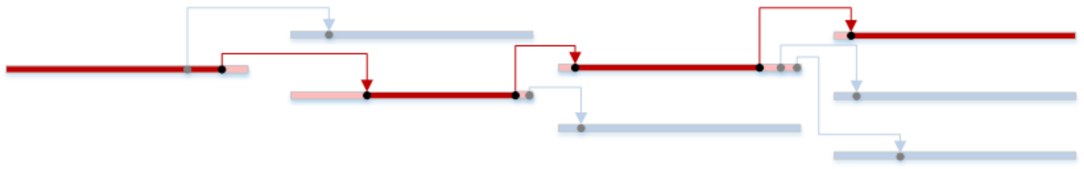


Figure 6.9: Illustration of searching for the best transitions through multiple motions and the resulting video sequence marked with red.

with the smallest similarity measure is the actual next frame in the original video. This means, that no intermediate frames are needed, so frames are only inserted if the source and target frames are not consecutive frames.

Since the capture of the Cathrine and Leonardo character have motions which are performed multiple times, there are therefore a number of possible motions to start the video sequence from as shown in Figure 6.10.

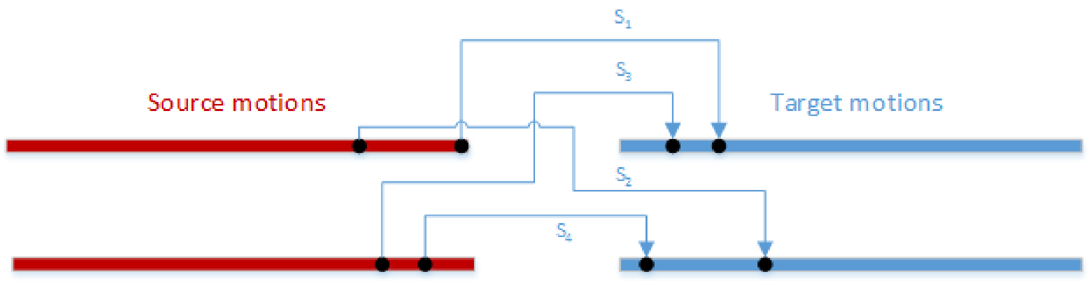


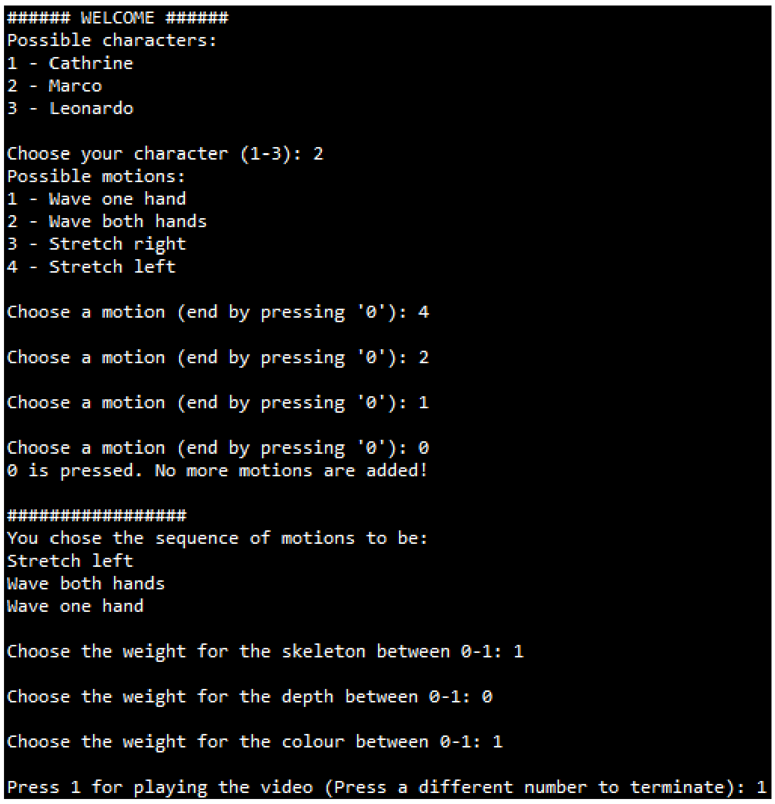
Figure 6.10: Illustration showing possible transitions between two source motions two possible target motions with their corresponding similarity measures, S_1, S_2, S_3 and S_4.

One way of choosing the starting motion could be randomized, but this does not necessary result in the best transition. Therefore all possible transitions are evaluated. According to the implementation, this means *findSourceTargetSingleMulti* is called as many times as the number of source motions to start from, and the transition with the minimum similarity measure is then chosen.

6.4 Animation Control Overview

An overview of the different steps in the animation control are listed below implemented as *interactiveControl.py*, and an example of running this script and choosing the Marco character is shown in Figure 6.11.

- Choose character
- Choose motion sequence
- Choose settings for the similarity measure
- Determine transitions and create intermediate frames
- Show video



```
###### WELCOME ######
Possible characters:
1 - Cathrine
2 - Marco
3 - Leonardo

Choose your character (1-3): 2
Possible motions:
1 - Wave one hand
2 - Wave both hands
3 - Stretch right
4 - Stretch left

Choose a motion (end by pressing '0'): 4

Choose a motion (end by pressing '0'): 2

Choose a motion (end by pressing '0'): 1

Choose a motion (end by pressing '0'): 0
0 is pressed. No more motions are added!

#################
You chose the sequence of motions to be:
Stretch left
Wave both hands
Wave one hand

Choose the weight for the skeleton between 0-1: 1

Choose the weight for the depth between 0-1: 0

Choose the weight for the colour between 0-1: 1

Press 1 for playing the video (Press a different number to terminate): 1
```

Figure 6.11: Example of the animation control, where the Marco-character is chosen together with three motions.

7. *User Study*

The purpose of performing a user study is to test how the realism is perceived when transitioning between motions, and whether there is a perceived difference when transitioning with or without animation. In order to test this, the area within psychophysics is exploited, where the human perception can be measured through psychometric methods.

When observing the world, as illustrated in Figure 7.1, information, e.g. light and sound, is received to the brain through various numbers of receptors, e.g. eyes and ears, and the perception is then the interpretation of the given information. Perception can therefore be described as the sum of all impressions.

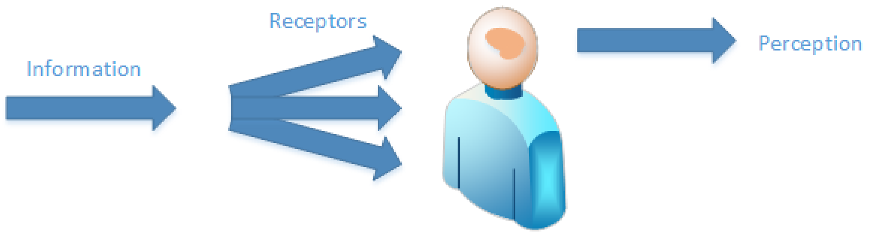


Figure 7.1: Illustration of how information is perceived.

But the way people interpret the received information can be very different, since it can depend on each person's personality and experiences. E.g. professional musicians can hear if a beat is out of sync, whereas non-experts might not hear that because they do not have as much experience as the musicians. The test subjects will in this user study be non-experts.

Within psychophysics, a user study involves a scenario where test subject are provided with some well known events, i.e. stimuli, which the subject should respond to (Poulsen, 2005). In this case the stimuli are videos with different known settings, which will be further described in section 7.1, and the method of how subjects should respond is described in section 7.2.

7.1 Stimuli

The test subjects are presented with three different sets of videos, where

- The transition between motions is made with no animation
- The transition between motions is made with animation based on alpha blending
- The transition between motions is made with animation based on optical flow

The hypothesis to test is the videos with animations based on optical flow will have a better perceived realism than the the other videos.

Since it is known from section 4.4 that different parameter settings for the similarity measure will find different transition points in the video and thus create different output videos, a number of settings are therefore tested. Knowing that there are infinite combinations of parameter settings to be tested, only a selection of combinations are chosen. The 10 different parameter settings which are tested, are presented in Table 7.1.

Parameter Settings

α	β	γ	# frames
1	-	-	-
1	-	-	5
1	1	-	-
1	1	-	5
1	1	-	5*
1	-	1	5
1	1	1	5
-	1	-	5
-	-	1	5
-	1	1	5

Table 7.1: The parameter settings for each video. *Blended frames.

Having three different characters, videos with the different parameter settings are generated for each character, i.e. 30 videos are generated. Each character will perform three different motions, presented in Table 7.2, which will be the same for all videos.

Motions in Videos

Cathrine	**Marco**	**Leonardo**
Wave	Wave two hands	Save bottom left corner
Twist	Stretch left	Save top right corner
Clap	Wave one hand	Save top left corner

Table 7.2: Overview of the input motions for each character.

The different combinations of the similarity measure, will then find different source and target frames between each motion and then perform a transition.

7.2 Method of Response

Dividing the test in three parts, each part for each character, the output of the user study should be an order of which parameter settings are the best according to their realism. This means all 10 stimuli is presented, and the test subject must then order them.

The problem with this approach, is that it can be confusing when then the subject is presented to many stimuli. It also puts a large demand on their memory of how previous videos looked like. The solution is to perform a pairwise comparison, where the user is presented with only two stimuli at a time (Lawrence E. Marks, 2002) This is also the approach used in (Casas et al., 2014) when the realism of their animation was evaluated in a user study.

When performing a paired comparison, all combinations must be tested as illustrated in Figure 7.2, which means the user has to make $10^2 - 10 = 90$ combinations for each character. But if a no order effect is assumed, i.e. the order in which the stimuli are presented, it can be reduced by half to 45 combinations. To avoid the test subject from predicting which videos to appear next, they are therefore presented in a shuffled order.

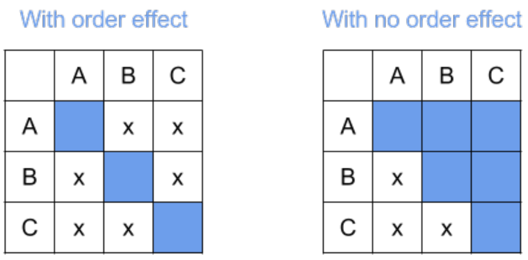


Figure 7.2: Illustration of the combinations in a pairwise comparison that must be tested (marked by x) with and without order effect.

After the test subjects are presented with two videos at a time, they are then asked three questions:

1. Which video do you prefer?
2. How would you rate the realism of the top video
3. How would you rate the realism of the bottom video

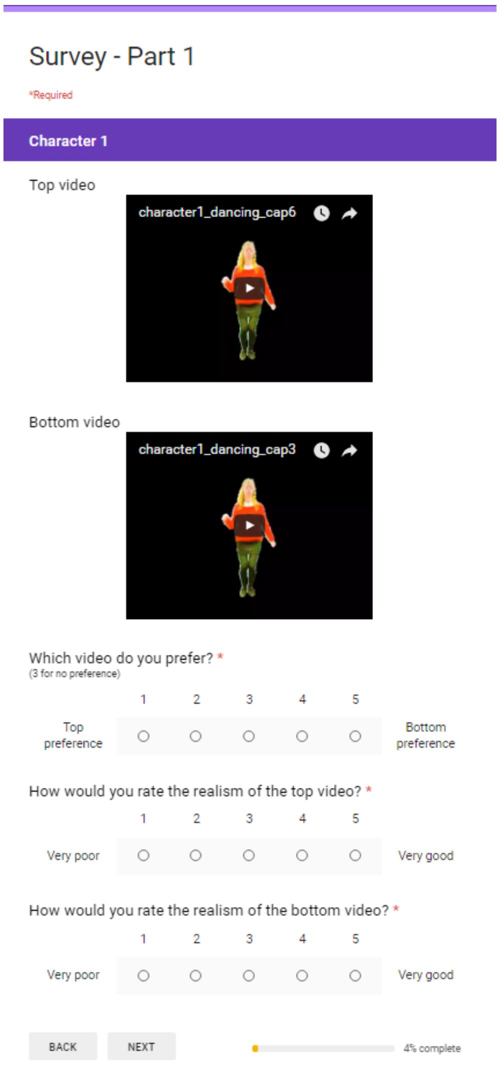
Test subjects could answer these questions on a nominal scale, i.e. for the first question, the user can choose either between top preference or bottom preference, and the realism for each video can be chosen to be either bad or good. But if a ordinal scale is chosen instead, where the test subject can give their answer on a scale from e.g. 1 to 5. This means more information about their perception can be retrieved.

The Likert scale is therefore used, which is a category scale (Lawrence E. Marks, 2002). Here the test subject will in terms of preference rate on a scale from top preference to bottom preference, and the realism will be rated from very bad to very good. It is important to note that increasing the points the scale will also increase the workload for the test subject. Since the workload is already high due to the many comparisons, the 5 point scale is therefore used, which is also the same size used in (Casas et al., 2014). For any scale it is also important for the test subject to have the possibility to give a neutral answer. For example if the video is neither bad nor good. This is why an uneven number for the scale chosen, so the test subject in that situation has the opportunity to answer 3.

7.3 Test Setup

The user study is made as a web-based survey using Google forms, where the videos must be inserted from YouTube. The videos can therefore in addition to be watched from the attachment also be watched online on the YouTube account called *c thomsen*. Since the test subject has the opportunity to view the videos more than once, it is therefore assumed that the order in which the videos are presented has no effect on the rating.

The stimuli together with the questions presented to the user is shown in Figure 7.3, which is the first comparison of the first part in the survey. At the start of the survey the test subject is introduced to the user study, together with a statement of consent, which is shown in Figure 7.4.


Survey - Part 1

*Required

Character 1

Top video

character1_dancing_cap6

Bottom video

character1_dancing_cap3

Which video do you prefer? *
(3 for no preference)

	1	2	3	4	5	
Top preference	○	○	○	○	○	Bottom preference

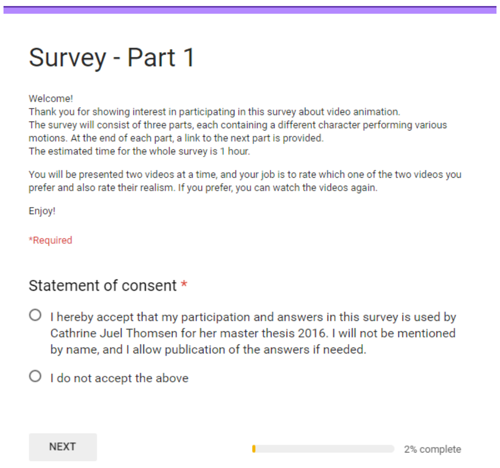
How would you rate the realism of the top video? *

	1	2	3	4	5	
Very poor	○	○	○	○	○	Very good

How would you rate the realism of the bottom video? *

	1	2	3	4	5	
Very poor	○	○	○	○	○	Very good

Figure 7.3: Paired comparisons in the survey.


Survey - Part 1

Welcome!
Thank you for showing interest in participating in this survey about video animation.
The survey will consist of three parts, each containing a different character performing various motions. At the end of each part, a link to the next part is provided.
The estimated time for the whole survey is 1 hour.

You will be presented two videos at a time, and your job is to rate which one of the two videos you prefer and also rate their realism. If you prefer, you can watch the videos again.

Enjoy!

*Required

Statement of consent *

○ I hereby accept that my participation and answers in this survey is used by Cathrine Juel Thomsen for her master thesis 2016. I will not be mentioned by name, and I allow publication of the answers if needed.

○ I do not accept the above

Figure 7.4: Statement of consent in the survey.

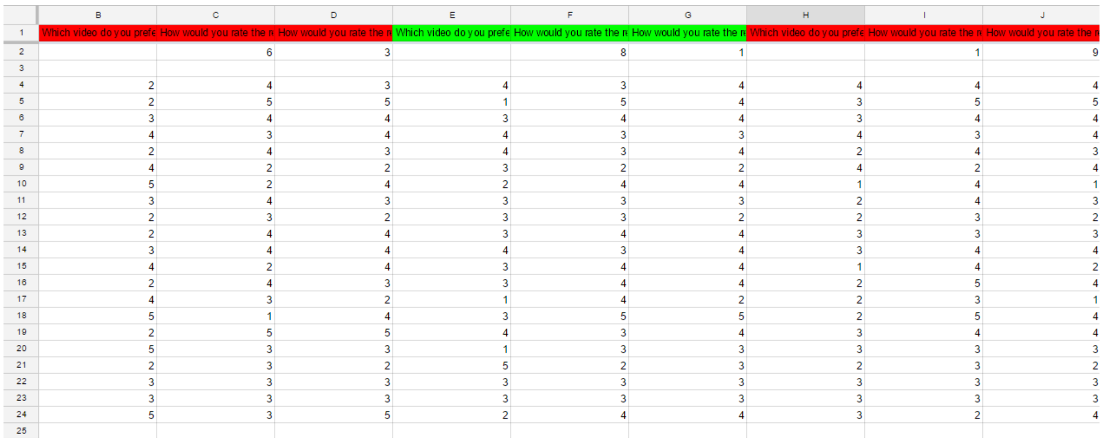
The survey is divided in three different parts, each part corresponding to one of the characters having 45 comparisons. Three Google Forms is created, and in the end of the first and second part of the survey, a link was provided to the next part.

In order to avoid the test subjects from figuring out which settings the videos have in the different characters, their namings are shuffled as summarized in Appendix B.

A pilot test was conducted to make sure that all videos could be played and all questions were identical. The outcome of the pilot test was also to estimate the time for completion of the whole test, so the test subject knows how much time he/she has to spend. The estimated time for completion was approximately 1 hour, as also shown in Figure 7.4.

7.4 Results

Answers from each of the three parts in the survey are saved in a Google spreadsheet. An example in Figure 7.5 shows the results of the first three comparisons for the first part in the survey, i.e. the Cathrine character.



	B	C	D	E	F	G	H	I	J
1	Which video do you prefe	How would you rate the re	How would you rate the re	Which video do you prefe	How would you rate the re	How would you rate the re	Which video do you prefe	How would you rate the re	How would you rate the re
2		6	3		8	1		1	9
3									
4	2	4	3	4	3	4	4	4	4
5	2	5	5	1	5	4	3	5	5
6	3	4	4	3	4	4	3	4	4
7	4	3	4	4	3	3	4	3	4
8	2	4	3	4	3	4	2	4	3
9	4	2	2	3	2	2	4	2	4
10	5	2	4	2	4	4	1	4	1
11	3	4	3	3	3	3	2	4	3
12	2	3	2	3	3	2	2	3	2
13	2	4	4	3	4	4	3	3	3
14	3	4	4	4	3	4	3	4	4
15	4	2	4	3	4	4	1	4	2
16	2	4	3	3	4	4	2	5	4
17	4	3	2	1	4	2	2	3	1
18	5	1	4	3	5	5	2	5	4
19	2	5	5	4	3	4	3	4	4
20	5	3	3	1	3	3	3	3	3
21	2	3	2	5	2	3	2	3	2
22	3	3	3	3	3	3	3	3	3
23	3	3	3	3	3	3	3	3	3
24	5	3	5	2	4	4	3	2	4
25									

Figure 7.5: Example of the results of three comparisons presented in a spreadsheet.

Every comparison has three questions, which corresponds to three columns and row 2 shows the two videos which are compared. The first column is the rating for the preference between the two videos presented to the test subject, the second column is the rated realism for the top video and the third column is the rated realism for the bottom video. This is repeated for each comparison which are divided by colours as shown in Figure 7.5.

The mean and standard deviation of the ratings for the perceived realism is found by gathering the data for each individual video. So for example the data for video in 1 in Figure 7.5 are gathered from column G and I.

In order to do the same for the preference the data is firstly divided so that 1 corresponds to low preference and 5 corresponds to a high preference. So the first answer in row 4 in the first comparison in Figure 7.5, shows a preference of 2, which means a slightly higher preference to the top video than the bottom video. The top video has therefore a preference of 4 and the bottom video has a preference of 2. This means the preference for the bottom video corresponds to the results in the first column, and the preference for the top video corresponds to results in first column, where each answer is subtracted with six.

The results for each character is summarized in Table 7.3 to Table 7.5, and the mean and standard deviation for the preference and perceived realism are illustrated in Figure 7.6 to Figure 7.11.

Results - Character 1

α	β	γ	# f	Preference	Realism
1	-	-	-	2.77 ($\pm$0.98)	3.10 ($\pm$1.08)
1	-	-	5	3.29 ($\pm$0.79)	3.71 ($\pm$0.81)
1	1	-	-	2.99 ($\pm$0.99)	3.24 ($\pm$1.04)
1	1	-	5	3.29 ($\pm$0.85)	3.67 ($\pm$0.87)
1	1	-	5*	2.39 ($\pm$0.98)	2.78 ($\pm$0.96)
1	-	1	5	3.23 ($\pm$0.85)	3.59 ($\pm$0.87)
1	1	1	5	3.33 ($\pm$0.85)	3.70 ($\pm$0.83)
-	1	-	5	3.33 ($\pm$0.82)	3.67 ($\pm$0.83)
-	-	1	5	2.14 ($\pm$1.11)	2.35 ($\pm$1.20)
-	1	1	5	3.25 ($\pm$0.86)	3.69 ($\pm$0.90)

Table 7.3: Results for the Cathrine character.

Results - Character 2

α	β	γ	# f	Preference	Realism
1	-	-	-	2.51 ($\pm$0.86)	3.11 ($\pm$0.88)
1	-	-	5	3.34 ($\pm$0.67)	3.93 ($\pm$0.79)
1	1	-	-	2.52 ($\pm$0.78)	3.04 ($\pm$0.95)
1	1	-	5	3.37 ($\pm$0.69)	3.98 ($\pm$0.80)
1	1	-	5*	2.30 ($\pm$0.91)	2.87 ($\pm$1.06)
1	-	1	5	3.49 ($\pm$0.72)	3.97 ($\pm$0.86)
1	1	1	5	3.41 ($\pm$0.65)	3.94 ($\pm$0.76)
-	1	-	5	3.27 ($\pm$0.67)	3.89 ($\pm$0.81)
-	-	1	5	2.38 ($\pm$0.86)	3.01 ($\pm$0.92)
-	1	1	5	3.41 ($\pm$0.72)	3.91 ($\pm$0.80)

Table 7.4: Results for the Marco character.

Results - Character 3

α	β	γ	# f	Preference	Realism
1	-	-	-	2.52 ($\pm$0.86)	2.67 ($\pm$0.86)
1	-	-	5	3.66 ($\pm$0.86)	3.67 ($\pm$1.03)
1	1	-	-	2.42 ($\pm$0.83)	2.63 ($\pm$0.81)
1	1	-	5	3.66 ($\pm$0.80)	3.76 ($\pm$0.94)
1	1	-	5*	2.08 ($\pm$0.89)	2.37 ($\pm$0.86)
1	-	1	5	3.79 ($\pm$0.83)	3.88 ($\pm$0.94)
1	1	1	5	3.51 ($\pm$0.81)	3.65 ($\pm$0.93)
-	1	-	5	2.77 ($\pm$0.79)	3.05 ($\pm$0.89)
-	-	1	5	2.78 ($\pm$0.85)	3.05 ($\pm$0.95)
-	1	1	5	2.81 ($\pm$0.79)	3.12 ($\pm$0.90)

Table 7.5: Results for the Leonardo character.

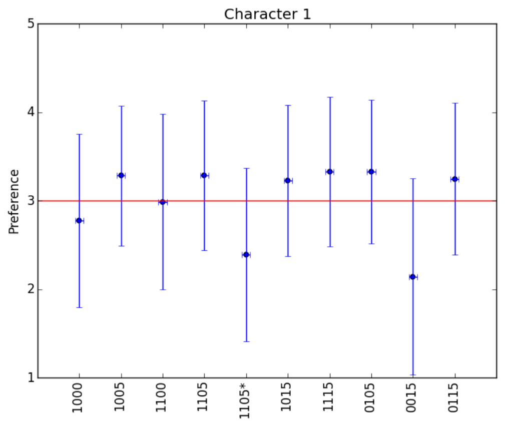


Figure 7.6: Test results for the preference for the Cathrine character.

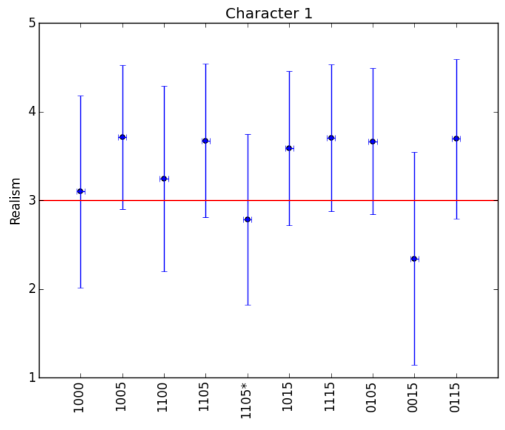


Figure 7.7: Test results for the perceived realism for the Cathrine character.

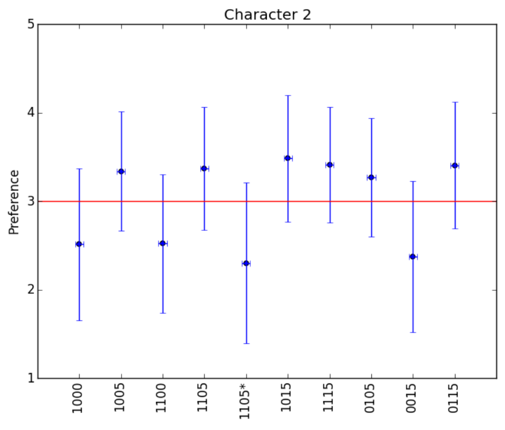


Figure 7.8: Test results for the preference for the Marco character.

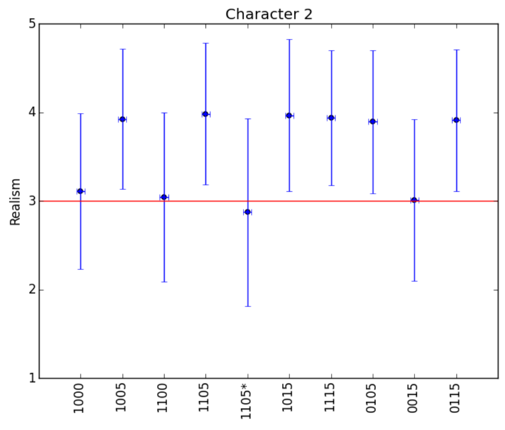


Figure 7.9: Test results for the perceived realism for the Marco character.

7.4.1 Discussion of Results

The results show that the animation based on alpha blending was rated as one of the lowest for all three characters, and this is followed by the transitioning with no animation, which also scored a low preference and perceived realism. As an example, the transition between two first motions for the Leonardo character using the skeleton and depth information with no animation is shown in Figure 7.12, where the transition is done between frame 491 and 376, which shows a difference in shape, and therefore creates a jump in the output video.

Using 5 frames based on alpha blending with the same similarity settings instead, creates a unrealistic transition, which is shown in Figure 7.13, whereas the animation based on optical flow, shown in Figure 7.14, creates a much smoother transition with no blur.

Interestingly, the transitions with animation based on optical flow using only the colour data in the similarity measure, received a very low score, and for the Cathrine character, it even got the lowest score. Comparing these transitions with the transitions where the skeleton is added, is shown in Figure 7.15, where they both have the same target frame, but different source frames.

The resulting animation for each of the settings are shown in Figure 7.16 and Figure 7.17,

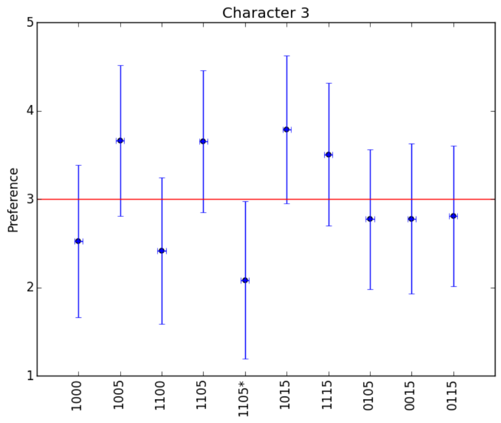


Figure 7.10: Test results for the preference for the Leonardo character.

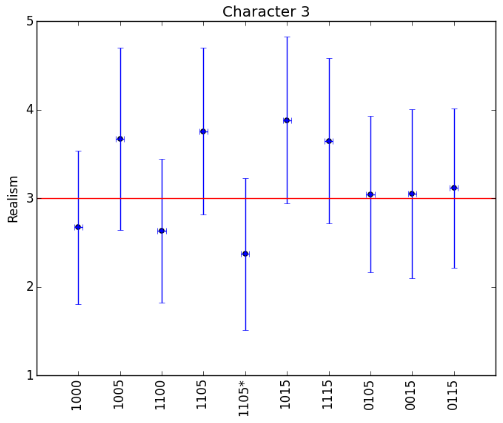


Figure 7.11: Test results for the perceived realism for the Leonardo character.

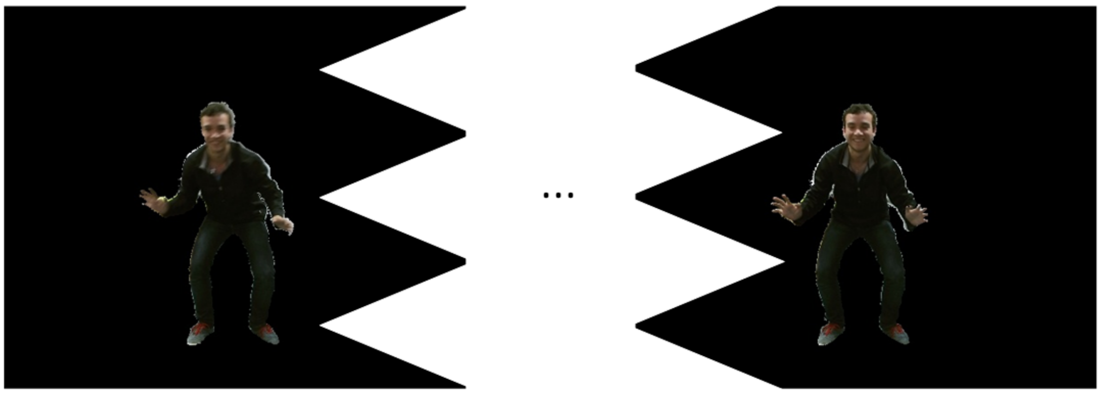


Figure 7.12: Transition between frame 491 and 376 using the skeleton and depth information in the similarity measure.

where using the colour information causes the transition to be unrealistic. This is due to the fact that the pose between the source and target frame is too different, and the optical flow therefore fails since it is restricted to a small difference in pose as mentioned in section 5.2.

The same issue is also visible for the Marco character, where the source and target frame found only by the colour information are too different as shown in Figure 7.18. Adding the skeleton information creates again a better match between the source and target frame.

The resulting animation between the source and target frames using the two different parameter settings are shown in Figure 7.19 and Figure 7.20, where optical flow again fails to estimate the flow correctly due to the large difference in pose.

The results also shows that using the skeleton information only or adding it to the colour an depth information will give a better match in terms of difference in poses between a source and target frame, which is why the the videos where the skeleton information are used received higher scores for all characters. Since the skeleton only provides information about the pose, whereas the colour information only provides information about the appearance, it therefore makes sense that skeleton overrules the other similarity measures because the output from transitioning using the similarity measures should be the best match in pose.

Adding colour or depth information to skeleton also showed to have small impact in the

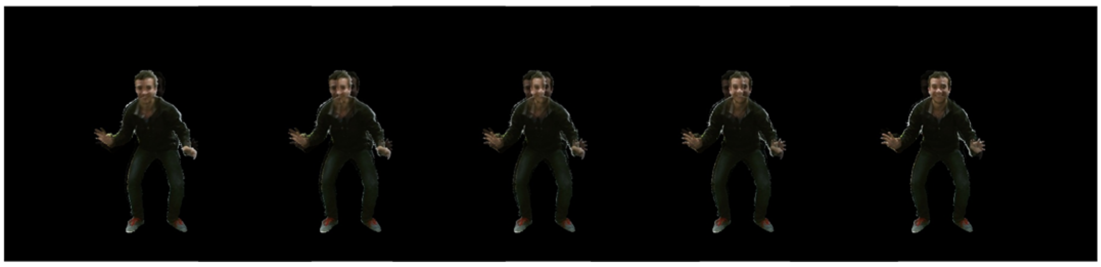


Figure 7.13: Animation based on alpha blending between the source and target frame found by the skeleton and depth information.

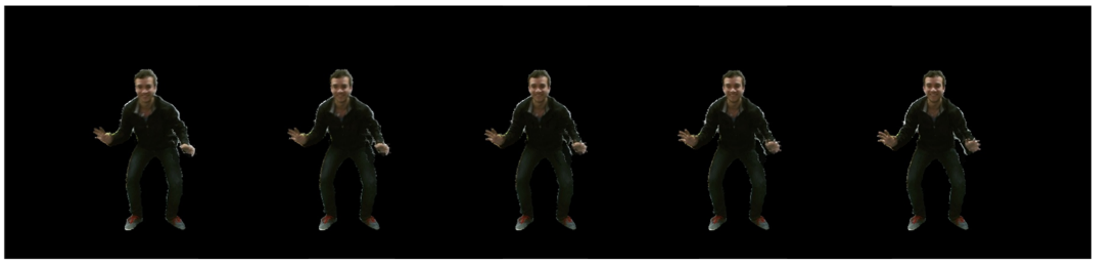


Figure 7.14: Animation based on optical flow between the source and target frame found by the skeleton and depth information.

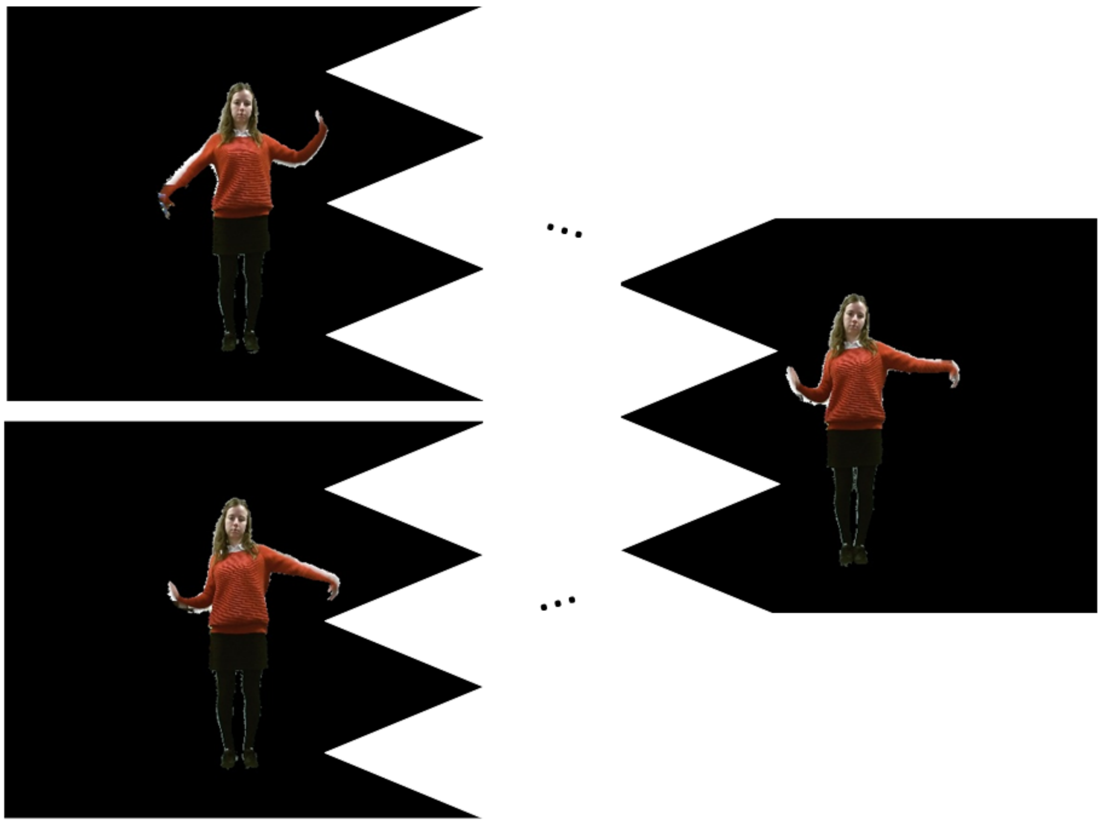


Figure 7.15: Two source frames with the same target frame (frame 554) found by two different similarity measure settings. Top source frame (frame 65) is found using the colour information only, whereas the bottom source frame (frame 88) is found using colour and skeleton information.

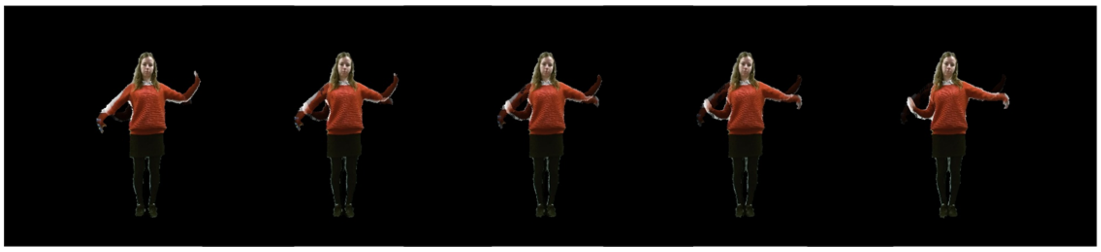


Figure 7.16: Animation based on optical flow between the source and target frame found by the colour information.

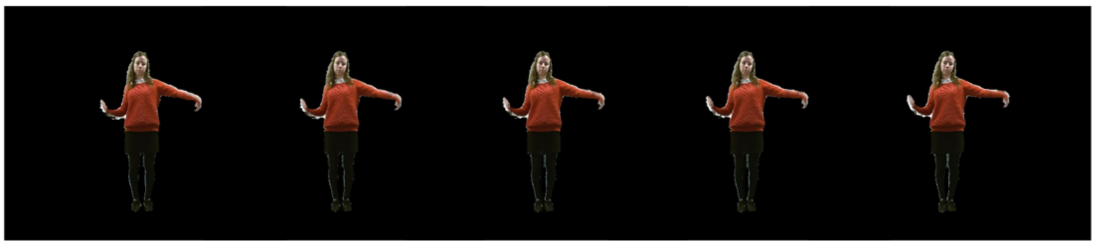


Figure 7.17: Animation based on optical flow between the source and target frame found by the colour and skeleton information.

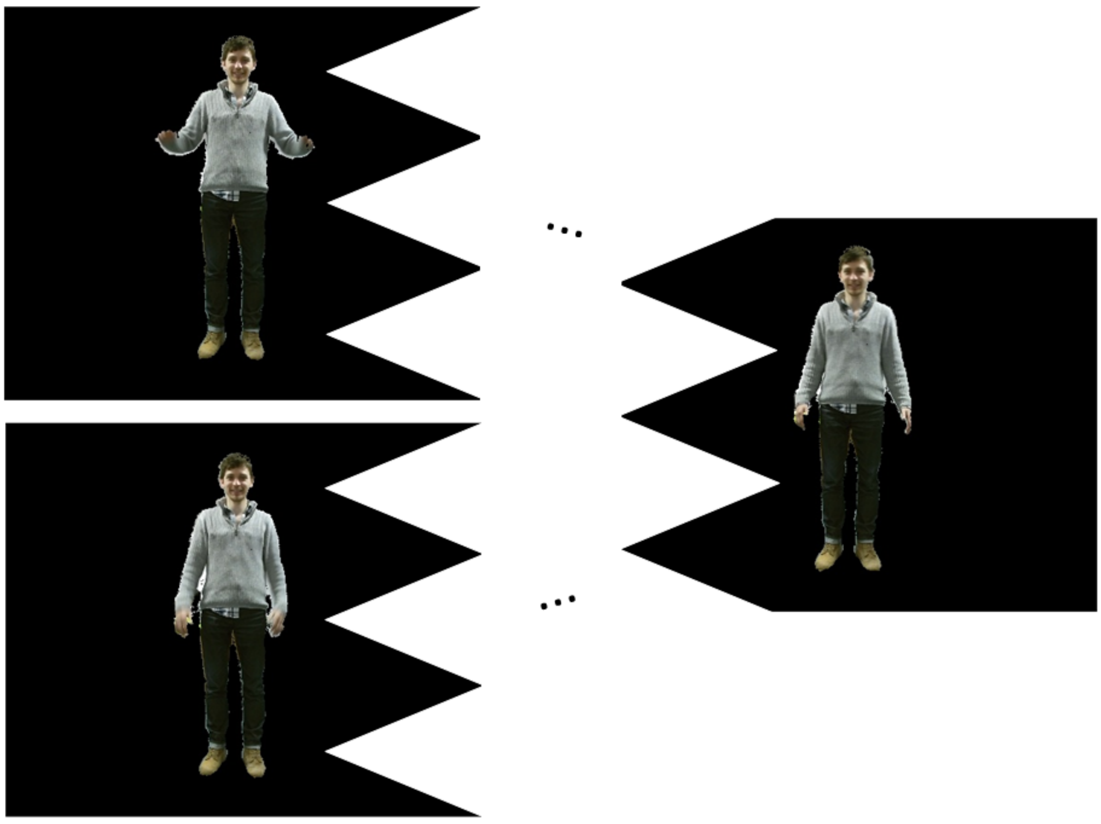


Figure 7.18: Two source frames with the same target frame (frame 224) found by two different similarity measure settings. Top source frame (frame 168) is found using the colour information only, whereas the bottom source frame (frame 173) is found using colour and skeleton information.

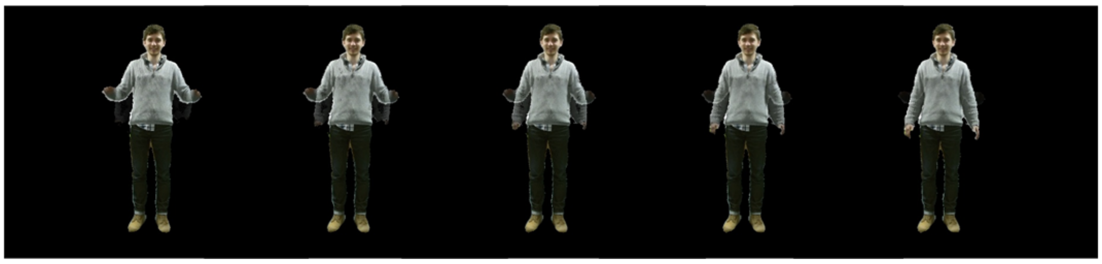


Figure 7.19: Animation based on optical flow between the source and target frame found by the colour information.

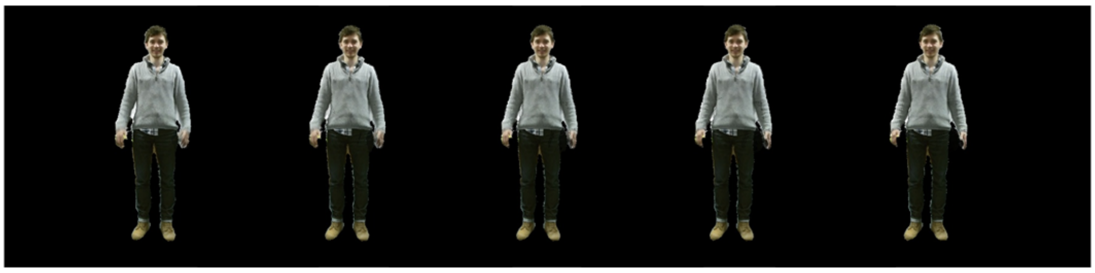


Figure 7.20: Animation based on optical flow between the source and target frame found by the colour and skeleton information.

preference and the perceived realism in the videos for all characters. This is because the source and target frames only vary by one or two frames, which in the end will create a very similar transition. An example is given for the Cathrine character, where the transitioning frames for using skeleton, shown in Figure 7.21, using skeleton and depth, shown in Figure 7.22, and using skeleton and colour, shown in Figure 7.15, only vary with a few frames and the difference in poses are still very small.

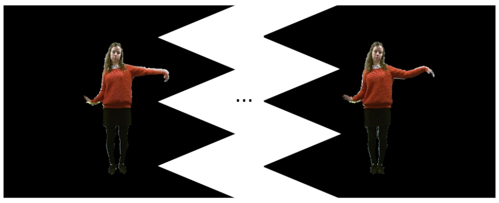


Figure 7.21: The source frame (frame 90) and target frame (frame 557) from only using the skeleton information.

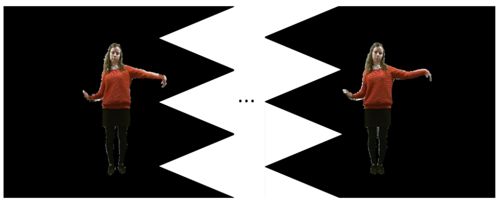


Figure 7.22: The source frame (frame 89) and target frame (frame 556) from using the skeleton and depth information.

Looking at the animation in Figure 7.14, Figure 7.17 and Figure 7.20, the transitions appear smooth and the reason why they did not received an even higher score could be due to the chosen number of intermediate frames.

When playing the video with the animation in Figure 7.17 for the Cathrine character, it will create a small pause when the transition happens, since the movements before and after the transition are performed quickly. This means the number of intermediate were in this case set too high. On the other hand, the number of frames can also be set too low, which will result in transitions that happen too fast where the change of pose and appearance in the animation is noticeable, which is the case for the Marco character. So analyzing the motion before and after the transition could help establishing an appropriate number of intermediate frames, which could create a better animation with higher rate of realism.

8. Evaluation

Having implemented an interactive animation control and evaluated the animation through a user study, an overall conclusion is made followed by a discussion of the future work.

8.1 Conclusion

The conclusion is based on the problem statement:

How should an interactive animation with seamless transitions which uses captured sequences of a person be constructed?

Three different people performing various motions were captured by acquiring the skeleton, depth and colour data from a second generation Kinect for Windows. The motions in each capture were manually divided and inserted in a motion graph to assure only feasible transitions between motions would happen.

Segmenting the colour and depth frames by learning a background model proved to be a poor segmentation, thus another approach was suggested. Using the skeleton information to establish the range of the body in the depth frame gave a segmentation where the person and a part of the floor was left. The floor was then segmented by averaging a plane over the first 20 frames of a captured video and exclude points close to the plane in the rest of the video. Additional noise left in the frames were removed by preserving only the biggest BLOB using a 4-connectivity connected component analysis. This segmentation was visually evaluated to be acceptable.

The best transition between a source and target motion was found by a similarity measure by minimizing the Euclidean distance of a combination of skeleton, depth and colour data between all possible frames. In order to create seamless transitions between motions, intermediate frames were needed. The suggested approach was to create new poses by estimating the dense optical flow between the transitioning frames in each direction and then move the pixels step wise towards the flow and blend the frames from each direction.

An interactive animation control was implemented, where it is possible for a user to control three different characters. Animations between the best transitions are made using a fixed amount of intermediate frames from the suggested approach.

The suggested animation based on optical flow was compared with animations using a direct alpha blending between the transitioning frames and not performing any animation. This was evaluated through a web based user study between three different characters. The results showed consistency over all characters where the suggested animation with different similarity measure settings had a higher rate of preference and perceived realism than using animations based on alpha blending and using no animation. The user study also showed possible ways for improving the suggested animation by establishing an adaptive number of intermediate frames instead of being fixed.

8.2 Future Work

This section describes possible topics for future work, which could improve the video animation of people.

Looping motions

In order for a person to perform the same motion multiple times, looped motions, as illustrated in Figure 8.1, could be implemented. These loops could be created by minimizing the error in the self-similarity matrix within that motion together with maximizing the length of the looped sequence as in (Schödl et al., 2000). Looped motions could also be created by transitioning between the end and the start of a motion, but this of course requires that the end and start frame are similar enough to perform a smooth transition.

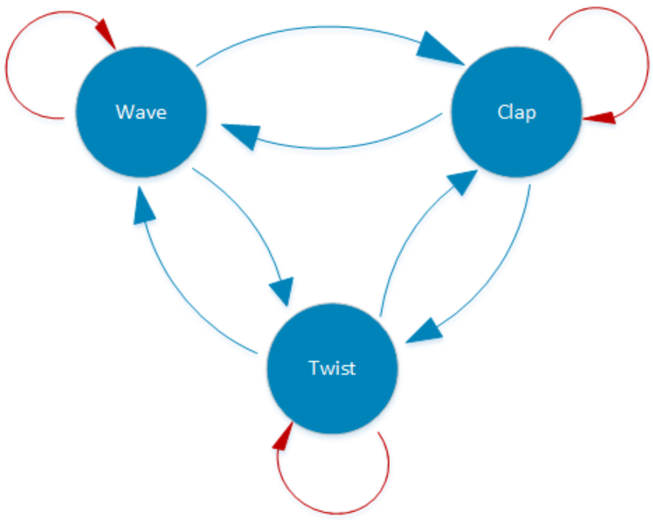


Figure 8.1: Illustration of the motion graph for the Cathrine character with looped motions added.

Another possibility to determine the length of the looped motion could be if the user determined the amount of time that the person has to perform the motion, e.g. Wave(10 seconds), Twist (3 seconds), clap (5 seconds). This information could then be used to create a looped motion with a number of frames that corresponds to the set time.

Adaptive number of intermediate frames

The number of intermediate frames should not be fixed to the same number for all transitions, but should vary according to the the speed of motion before and after transitioning. But even if the motions before and after the transition are fast does not necessarily mean that a small number of intermediate frames need to be added, because if the difference in pose are large, i.e. above a threshold, more intermediate frames are needed. This means the similarity between the source and target frame should also be taken into account to determine the number of intermediate frames between a transition.

Having an adaptive number of intermediate frames, it would be interesting to see how far the system could be pushed by minimizing the amount of intermediate frames and still maintain a high perceived realism. This could be investigated by performing a user study using the method of limits for determining a threshold (Poulsen, 2005).

Using this method, the test subjects would be presented with ascending or descending number of intermediate frames in a video until their response change from unrealistic to realistic. The output of this study would then be the smallest number of intermediate frames needed in an animation, where the realism of the transition is still maintained.

Animation test

The user study performed in chapter 7 was made, where the different characters was presented on a black background. This could mean that small differences in the animation or segmentation would be easier to spot than inserting the characters into another video with a moving background where less focus is on the person. Inserting the same videos on a moving background, would therefore be interesting to see if they provide higher rate of realism.

Automatically dividing motions

Manually dividing motions could with many videos be very time consuming, which is why a method for performing this automatically could be investigated.

One way to divide the motions automatically could be by performing frame clustering as in (Klaudiny et al., 2012), where similar successive frames has a low similarity measure around the diagonal of the self-similarity matrix, which can be utilized by clustering. Each cluster will then be equivalent to one motion as illustrated in Figure 8.2.

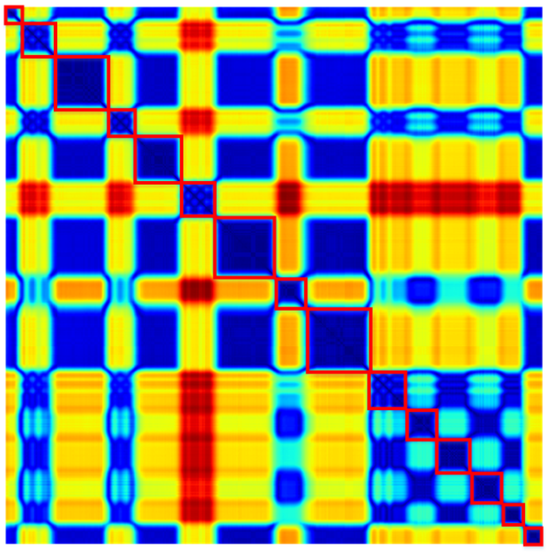


Figure 8.2: An example of how the automatic division should be done by using frame clustering marked by a red square.

Automatically labelling motions

If a database with labelled motions were available, labelling motions from a new capture automatically could then be possible. So whenever a new capture is done, the new motions are held up against the database. Motions could then be automatically labelled if the new capture have similar motions to the ones in the database, so frame clustering approach could therefore also be performed here. If motions are very different from the ones in the database, then the user should label the motion him-/herself and in that way adding new motions to the database.

An example is given below, where the database is the evaluating dataset, where a person is waving with left and right hand, and the new capture is the dataset for the Marco character, where one of his motions is waving with left hand. Finding the similarity for, e.g. the skeleton across all frames will result in a similarity matrix as shown in Figure 8.3, where the x-axis is the frames the database and the y-axis is the new capture.

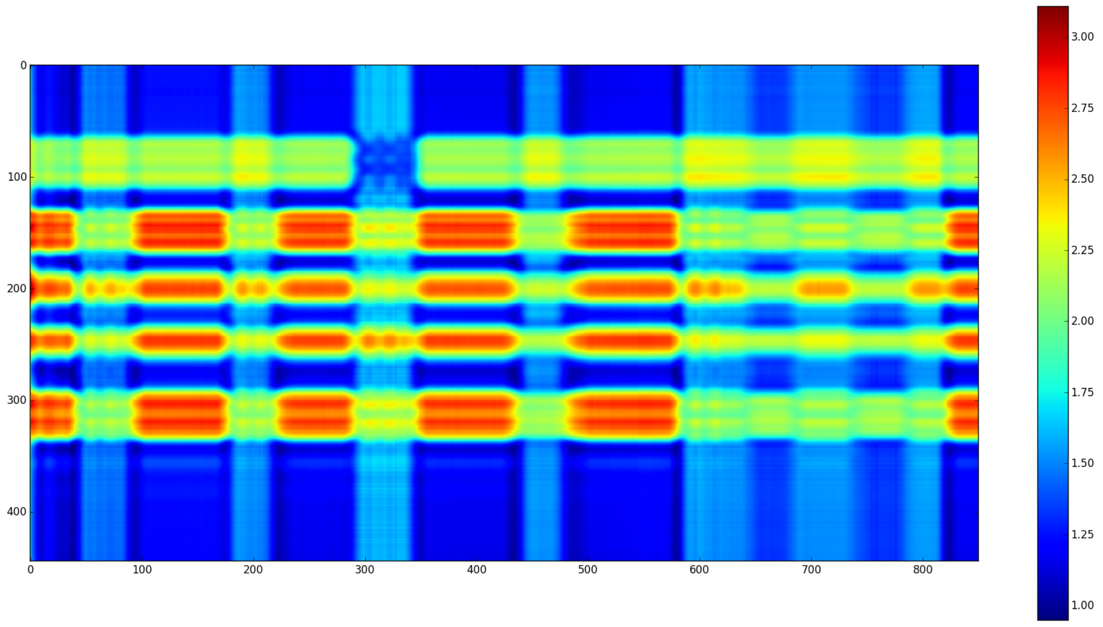


Figure 8.3: Similarity of the skeleton between the evaluating dataset (x-axis) and the dataset with the Marco character (y-axis).

By thresholding the similarity measure in Figure 8.3 with 1.5, it is then visible that there is a high similarity measure between the wave motion in the database and the wave motion in the new capture shown in Figure 8.4. The other motions with the high similarity measure are idle periods as shown in Figure 8.5.

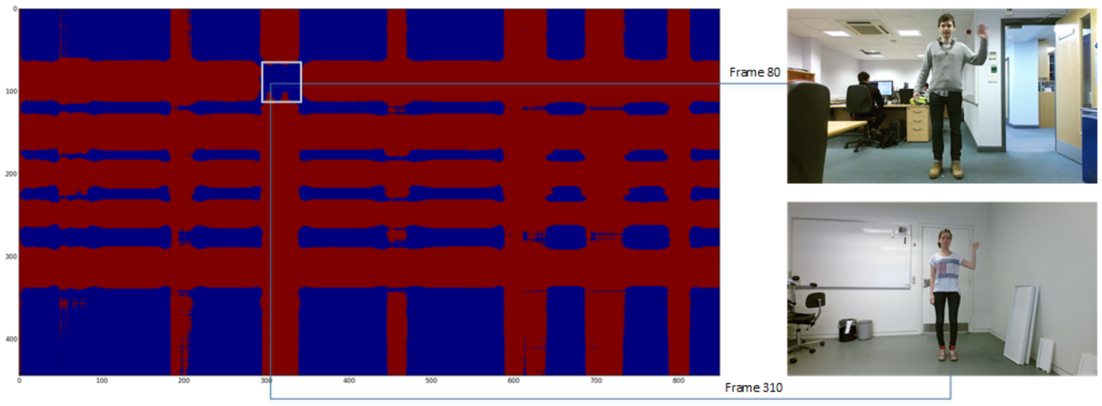


Figure 8.4: Thresholded Figure 8.3, where the high similarity for the wave motion is highlighted with two similar frames.

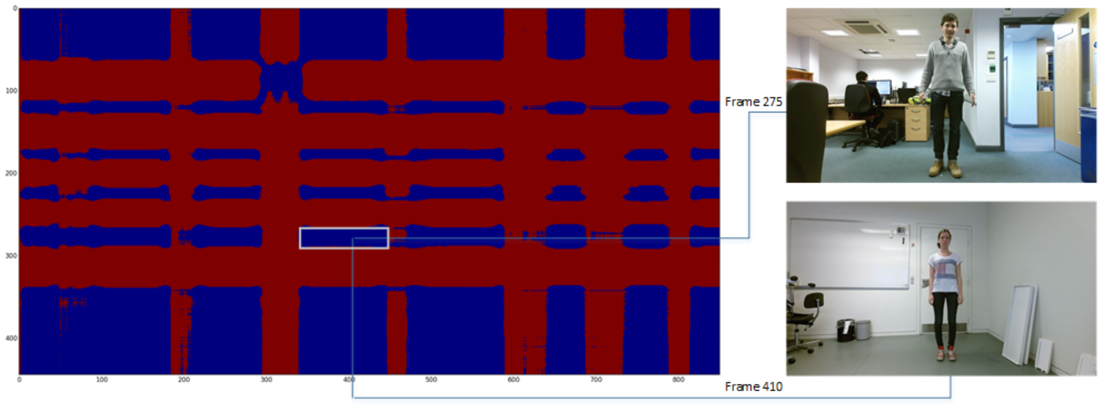


Figure 8.5: Thresholded Figure 8.3, where the high similarity for the idle period is highlighted with two similar frames.

A. *Evaluation of Distances*

Distance of 5 cm

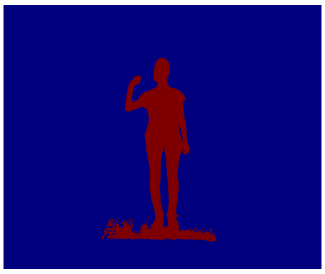


Figure A.1: Frame 70 segmented with a distance of 5 cm to the plane.

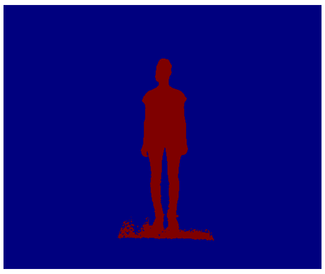


Figure A.2: Frame 150 segmented with a distance of 5 cm to the plane.

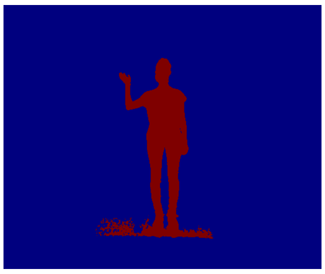


Figure A.3: Frame 200 segmented with a distance of 5 cm to the plane.

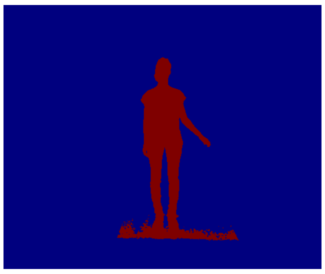


Figure A.4: Frame 285 segmented with a distance of 5 cm to the plane.

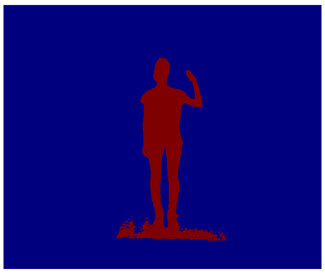


Figure A.5: Frame 300 segmented with a distance of 5 cm to the plane.

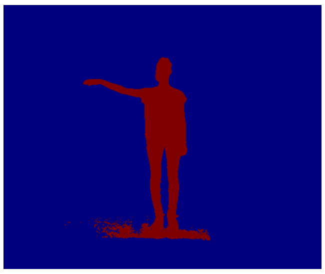


Figure A.6: Frame 450 segmented with a distance of 5 cm to the plane.

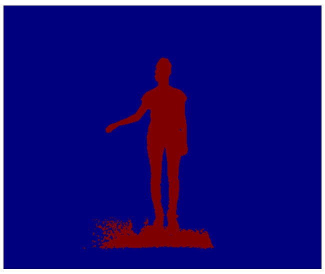


Figure A.7: Frame 580 segmented with a distance of 5 cm to the plane.

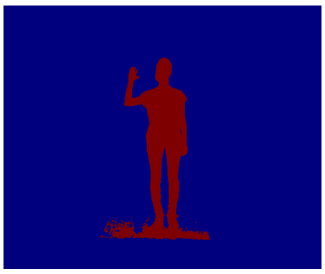


Figure A.8: Frame 700 segmented with a distance of 5 cm to the plane.

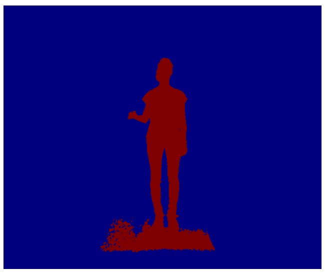


Figure A.9: Frame 820 segmented with a distance of 5 cm to the plane.

Distance of 6 cm

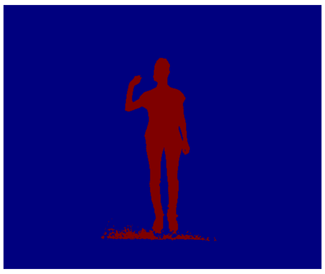


Figure A.10: Frame 70 segmented with a distance of 6 cm to the plane.

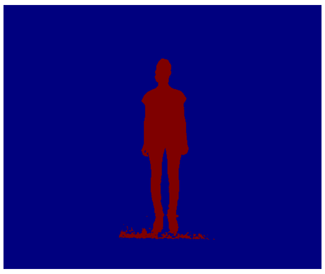


Figure A.11: Frame 150 segmented with a distance of 6 cm to the plane.

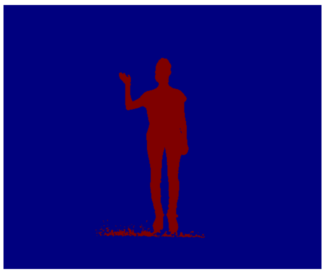


Figure A.12: Frame 200 segmented with a distance of 6 cm to the plane.

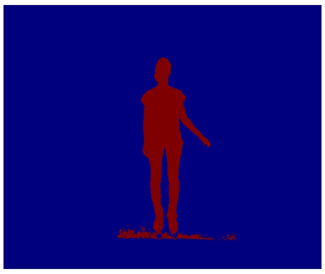


Figure A.13: Frame 285 segmented with a distance of 6 cm to the plane.

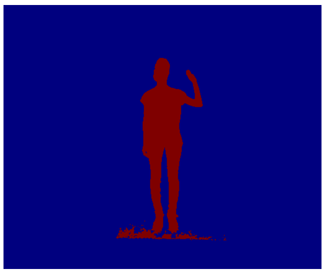


Figure A.14: Frame 300 segmented with a distance of 6 cm to the plane.

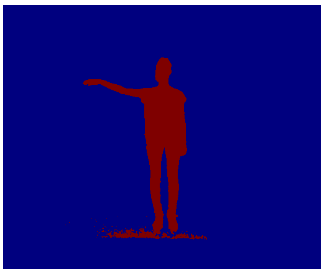


Figure A.15: Frame 450 segmented with a distance of 6 cm to the plane.

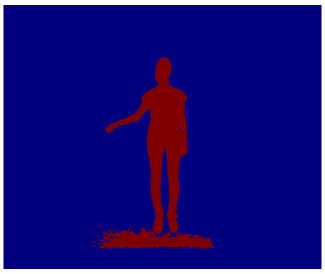


Figure A.16: Frame 580 segmented with a distance of 6 cm to the plane.

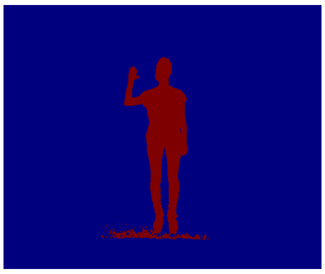


Figure A.17: Frame 700 segmented with a distance of 6 cm to the plane.

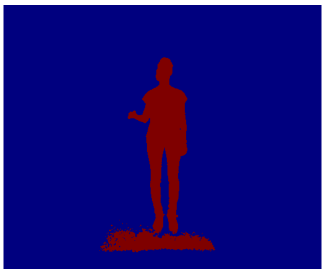


Figure A.18: Frame 820 segmented with a distance of 6 cm to the plane.

Distance of 7 cm

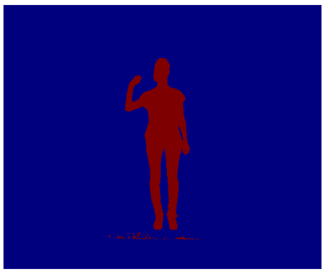


Figure A.19: Frame 70 segmented with a distance of 7 cm to the plane.

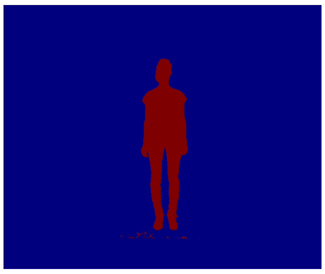


Figure A.20: Frame 150 segmented with a distance of 7 cm to the plane.

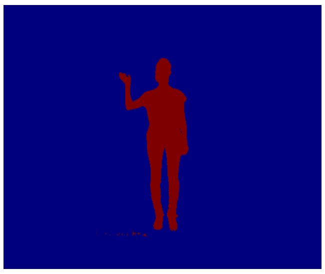


Figure A.21: Frame 200 segmented with a distance of 7 cm to the plane.

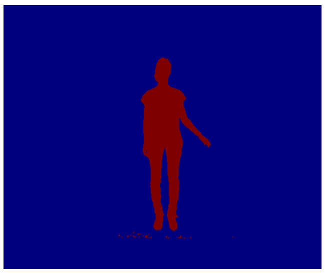


Figure A.22: Frame 285 segmented with a distance of 7 cm to the plane.

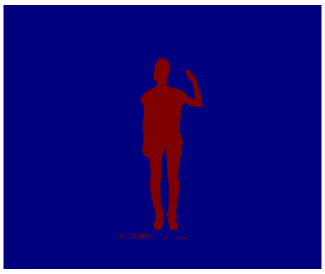


Figure A.23: Frame 300 segmented with a distance of 7 cm to the plane.

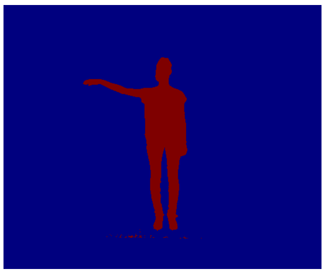


Figure A.24: Frame 450 segmented with a distance of 7 cm to the plane.

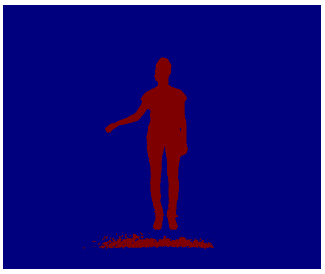


Figure A.25: Frame 580 segmented with a distance of 7 cm to the plane.

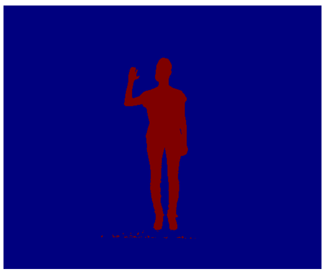


Figure A.26: Frame 700 segmented with a distance of 7 cm to the plane.

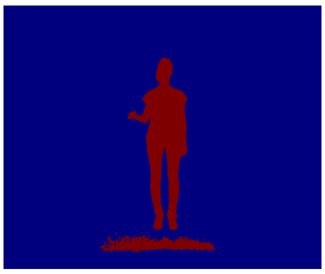


Figure A.27: Frame 820 segmented with a distance of 7 cm to the plane.

Distance of 8 cm

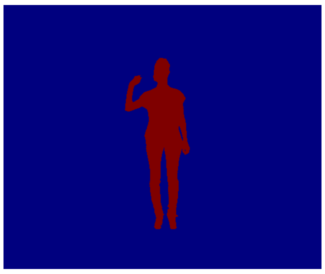


Figure A.28: Frame 70 segmented with a distance of 8 cm to the plane

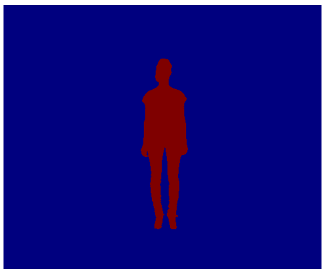


Figure A.29: Frame 150 segmented with a distance of 8 cm to the plane

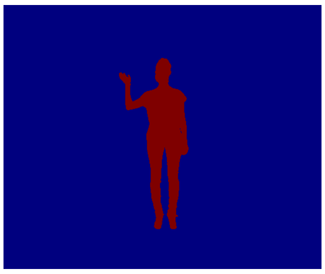


Figure A.30: Frame 200 segmented with a distance of 8 cm to the plane

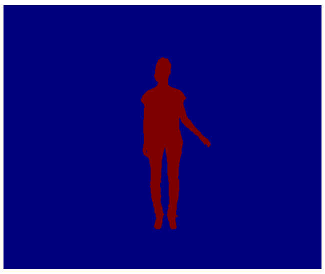


Figure A.31: Frame 285 segmented with a distance of 8 cm to the plane.

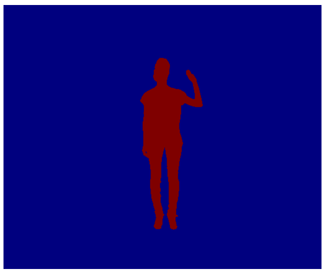


Figure A.32: Frame 300 segmented with a distance of 8 cm to the plane.

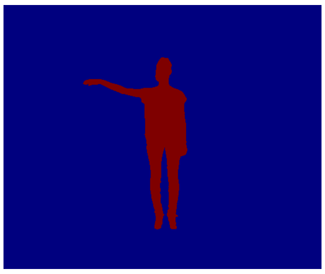


Figure A.33: Frame 450 segmented with a distance of 8 cm to the plane.

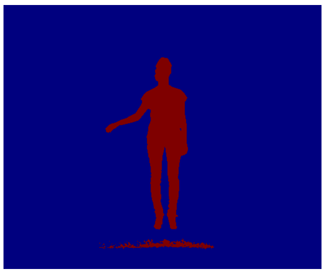


Figure A.34: Frame 580 segmented with a distance of 8 cm to the plane.

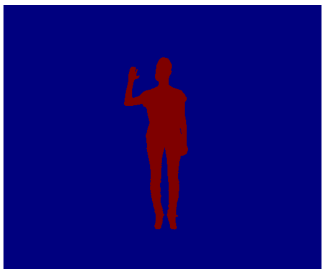


Figure A.35: Frame 700 segmented with a distance of 8 cm to the plane.

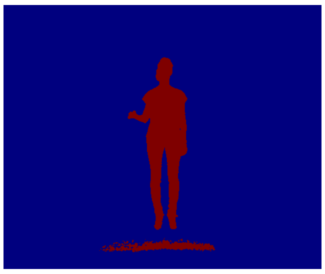


Figure A.36: Frame 820 segmented with a distance of 8 cm to the plane.

Distance of 9 cm

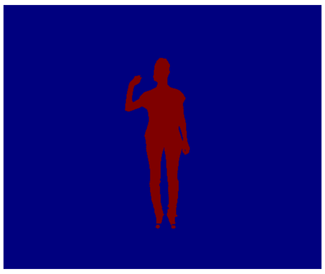


Figure A.37: Frame 70 segmented with a distance of 9 cm to the plane.

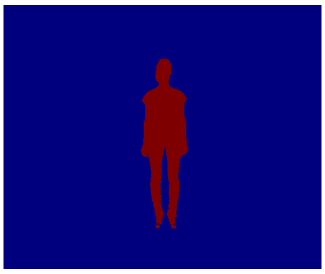


Figure A.38: Frame 150 segmented with a distance of 9 cm to the plane..

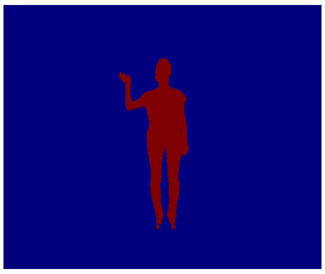


Figure A.39: Frame 200 segmented with a distance of 9 cm to the plane.

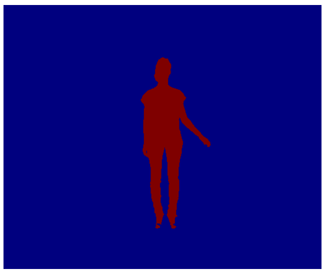


Figure A.40: Frame 285 segmented with a distance of 9 cm to the plane.

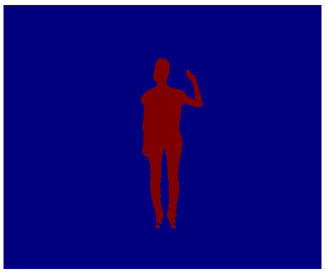


Figure A.41: Frame 300 segmented with a distance of 9 cm to the plane.

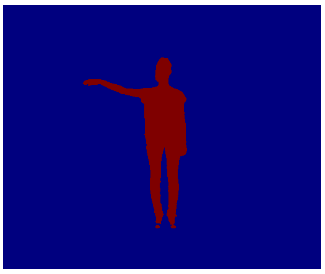


Figure A.42: Frame 450 segmented with a distance of 9 cm to the plane.

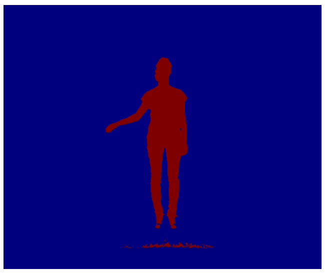


Figure A.43: Frame 580 segmented with a distance of 9 cm to the plane.

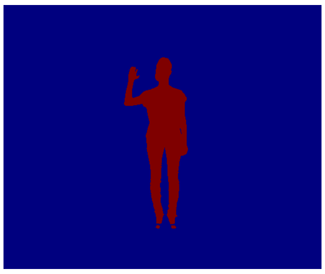


Figure A.44: Frame 700 segmented with a distance of 9 cm to the plane.

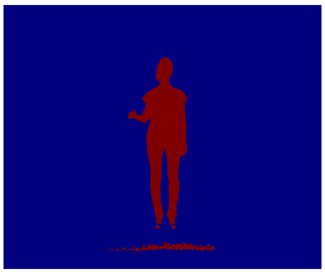


Figure A.45: Frame 820 segmented with a distance of 9 cm to the plane.

Distance of 7 cm - Noise Removed

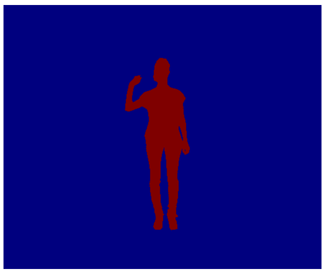


Figure A.46: Frame 70 segmented with a distance of 7 cm to the plane and noise removed.

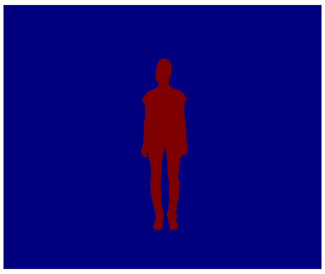


Figure A.47: Frame 150 segmented with a distance of 7 cm to the plane and noise removed.

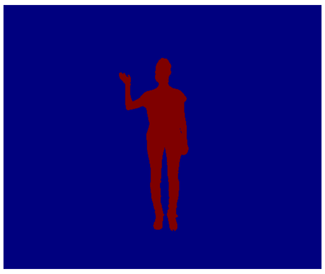


Figure A.48: Frame 200 segmented with a distance of 7 cm to the plane and noise removed.

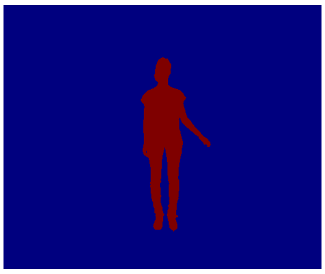


Figure A.49: Frame 285 segmented with a distance of 7 cm to the plane and noise removed.

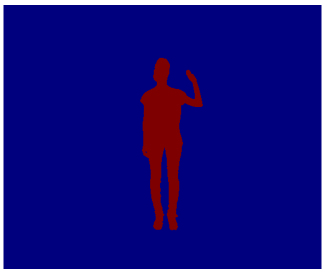


Figure A.50: Frame 300 segmented with a distance of 7 cm to the plane and noise removed.

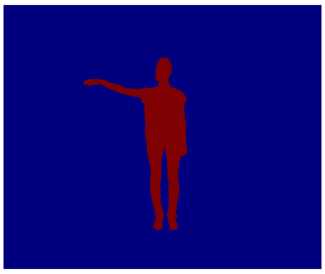


Figure A.51: Frame 450 segmented with a distance of 7 cm to the plane and noise removed.

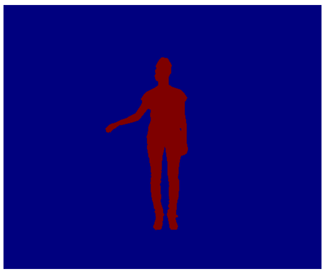


Figure A.52: Frame 580 segmented with a distance of 7 cm to the plane and noise removed.

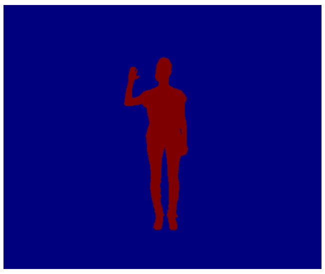


Figure A.53: Frame 700 segmented with a distance of 7 cm to the plane and noise removed.

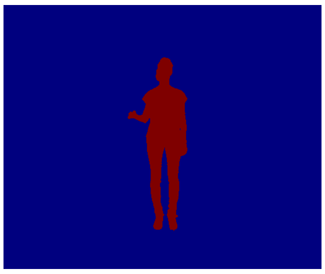


Figure A.54: Frame 820 segmented with a distance of 7 cm to the plane and noise removed.

B. Survey Video Namings

Shuffled video namings - Character 1

α	β	γ	# f	Attachment name	YouTube name
1	-	-	-	testDir/cathrine/a1_b0_g0_f0	character1_dancing_cap6
1	-	-	5	testDir/cathrine/a1_b0_g0_f5	character1_dancing_cap3
1	1	-	-	testDir/cathrine/a1_b1_g0_f0	character1_dancing_cap9
1	1	-	5	testDir/cathrine/a1_b1_g0_f5	character1_dancing_cap7
1	1	-	5*	testDir/cathrine/a1_b1_g0_f5Blend	character1_dancing_cap10
1	-	1	5	testDir/cathrine/a1_b0_g1_f5	character1_dancing_cap8
1	1	1	5	testDir/cathrine/a1_b1_g1_f5	character1_dancing_cap5
-	1	-	5	testDir/cathrine/a0_b1_g0_f5	character1_dancing_cap1
-	-	1	5	testDir/cathrine/a0_b0_g1_f5	character1_dancing_cap4
-	1	1	5	testDir/cathrine/a0_b1_g1_f5	character1_dancing_cap2

Table B.1: The namings of the survey videos for the Cathrine character in the attachment and the shuffled namings on YouTube. *Blended frames.

Shuffled video namings - Character 2

α	β	γ	# f	Attachment name	YouTube name
1	-	-	-	testDir/marco/a1_b0_g0_f0	character2_random_cap10
1	-	-	5	testDir/marco/a1_b0_g0_f5	character2_random_cap3
1	1	-	-	testDir/marco/a1_b1_g0_f0	character2_random_cap2
1	1	-	5	testDir/marco/a1_b1_g0_f5	character2_random_cap6
1	1	-	5*	testDir/marco/a1_b1_g0_f5Blend	character2_random_cap1
1	-	1	5	testDir/marco/a1_b0_g1_f5	character2_random_cap8
1	1	1	5	testDir/marco/a1_b1_g1_f5	character2_random_cap7
-	1	-	5	testDir/marco/a0_b1_g0_f5	character2_random_cap4
-	-	1	5	testDir/marco/a0_b0_g1_f5	character2_random_cap9
-	1	1	5	testDir/marco/a0_b1_g1_f5	character2_random_cap5

Table B.2: The namings of the survey videos for the Marco character in the attachment and the shuffled namings on YouTube. *Blended frames.

Shuffled video namings - Character 3

α	β	γ	# f	Attachment name	YouTube name
1	-	-	-	testDir/leonardo/a1_b0_g0_f0	character3_goalie_cap9
1	-	-	5	testDir/leonardo/a1_b0_g0_f5	character3_goalie_cap1
1	1	-	-	testDir/leonardo/a1_b1_g0_f0	character3_goalie_cap2
1	1	-	5	testDir/leonardo/a1_b1_g0_f5	character3_goalie_cap5
1	1	-	5*	testDir/leonardo/a1_b1_g0_f5Blend	character3_goalie_cap4
1	-	1	5	testDir/leonardo/a1_b0_g1_f5	character3_goalie_cap3
1	1	1	5	testDir/leonardo/a1_b1_g1_f5	character3_goalie_cap7
-	1	-	5	testDir/leonardo/a0_b1_g0_f5	character3_goalie_cap6
-	-	1	5	testDir/leonardo/a0_b0_g1_f5	character3_goalie_cap10
-	1	1	5	testDir/leonardo/a0_b1_g1_f5	character3_goalie_cap8

Table B.3: The namings of the survey videos for the Leonardo character in the attachment and the shuffled namings on YouTube. *Blended frames.

C. *Self-similarity Matrices for the Test Datasets*

Character 1

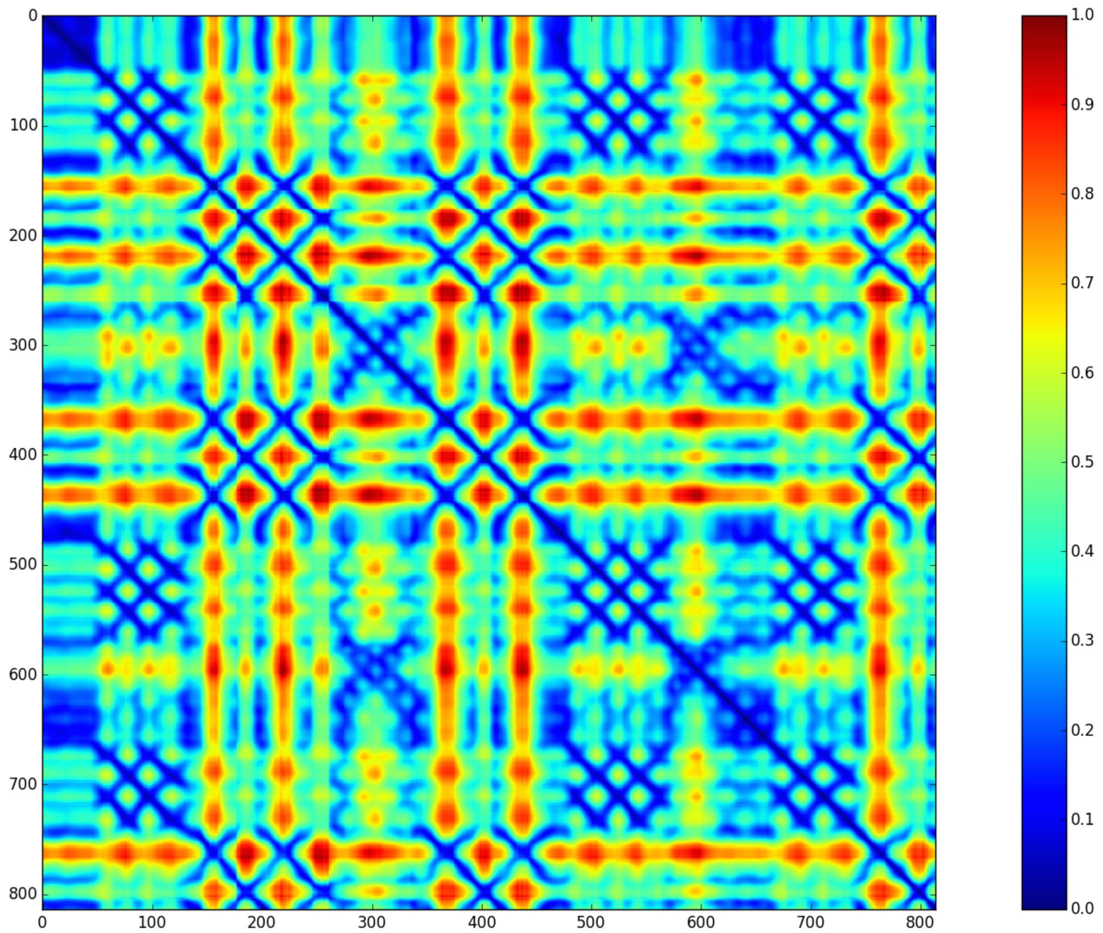


Figure C.1: The self-similarity matrix of the skeleton data for the Cathrine character mapped from similar, dark blue, to dissimilar, red.

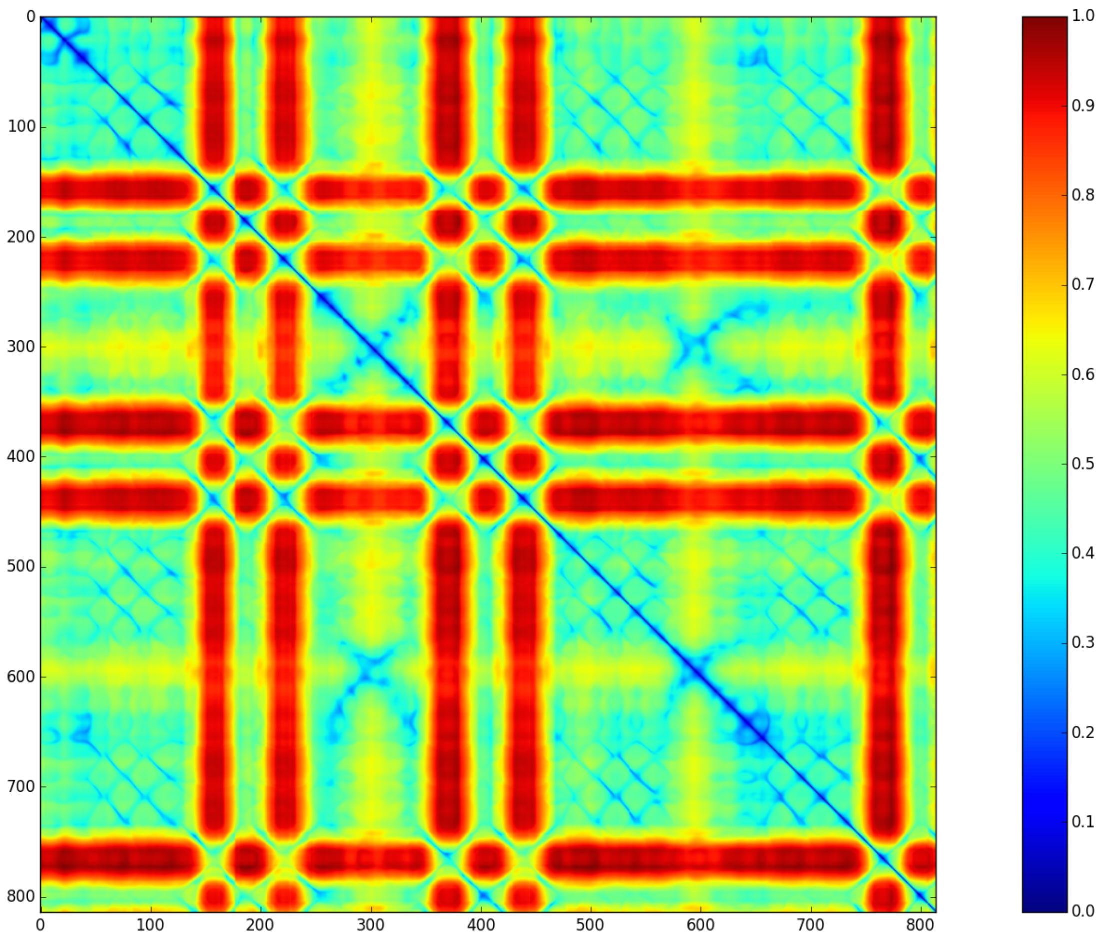


Figure C.2: The self-similarity matrix of the depth data for the Cathrine character mapped from similar, dark blue, to dissimilar, red.

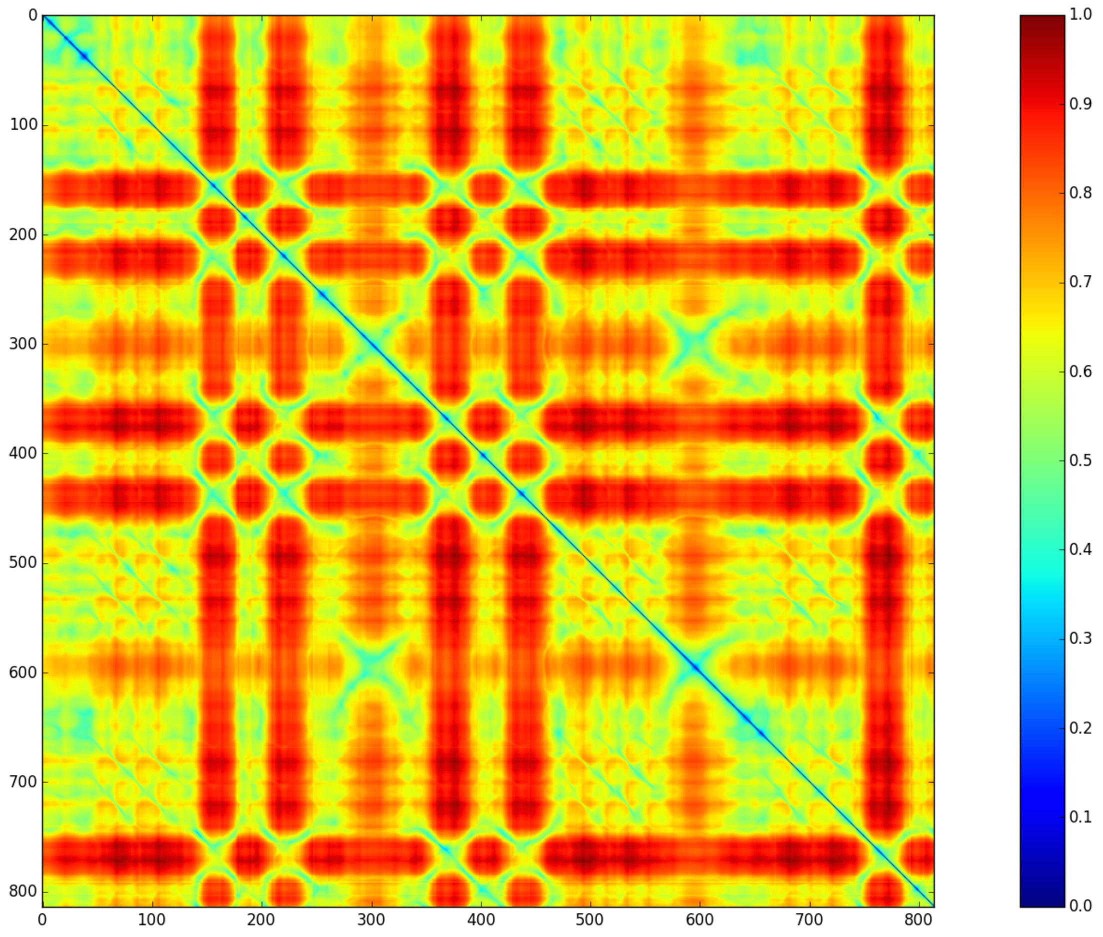


Figure C.3: The self-similarity matrix of the colour data for the Cathrine character mapped from similar, dark blue, to dissimilar, red.

Character 2

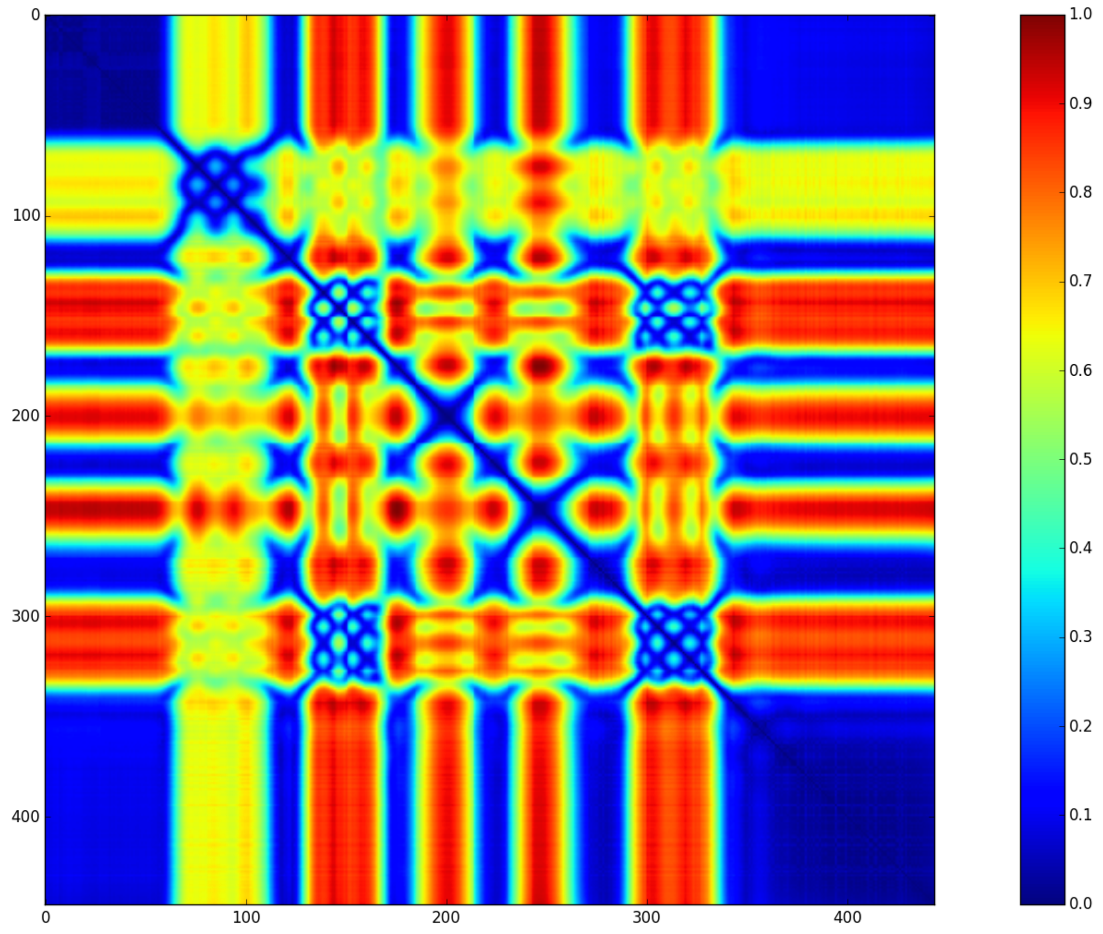


Figure C.4: The self-similarity matrix of the skeleton data for the Marco character mapped from similar, dark blue, to dissimilar, red.

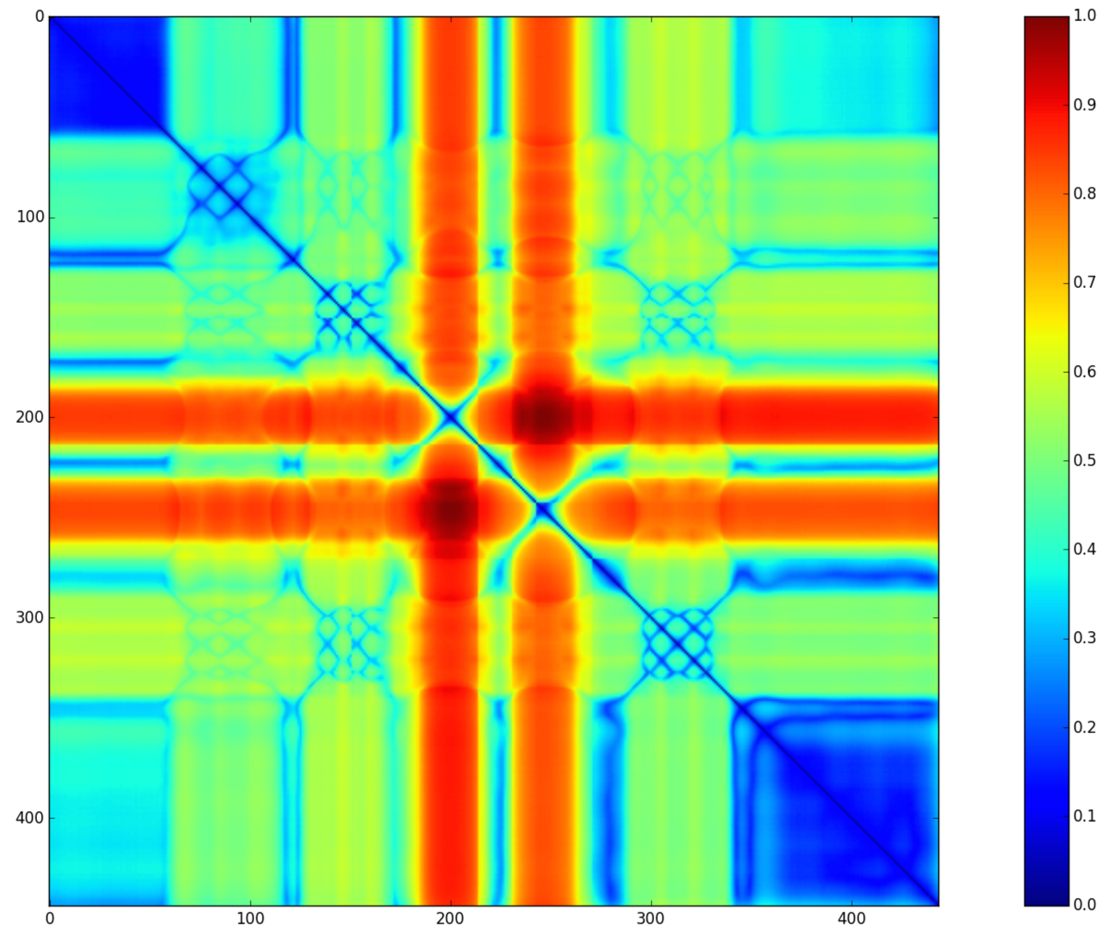


Figure C.5: The self-similarity matrix of the depth data for the Marco character mapped from similar, dark blue, to dissimilar, red.

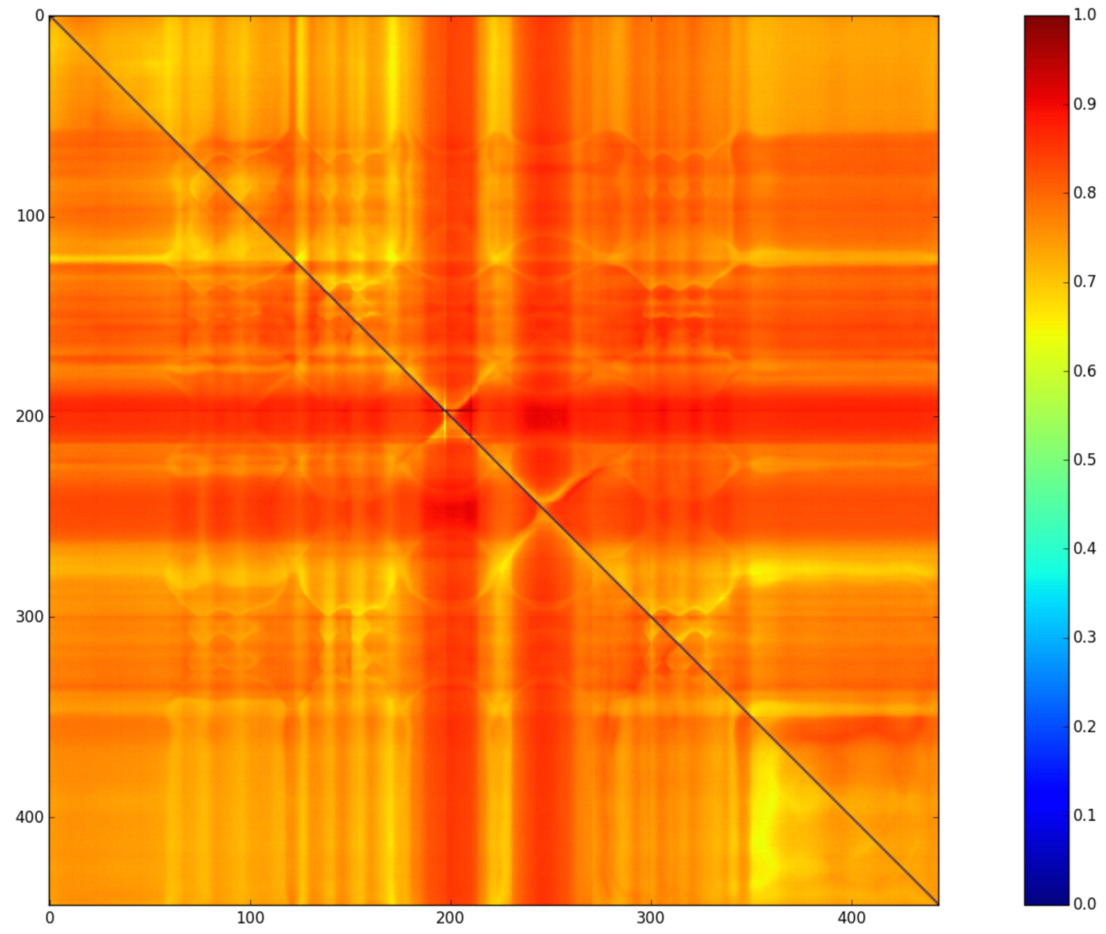


Figure C.6: The self-similarity matrix of the colour data for the Marco character mapped from similar, dark blue, to dissimilar, red.

Character 3

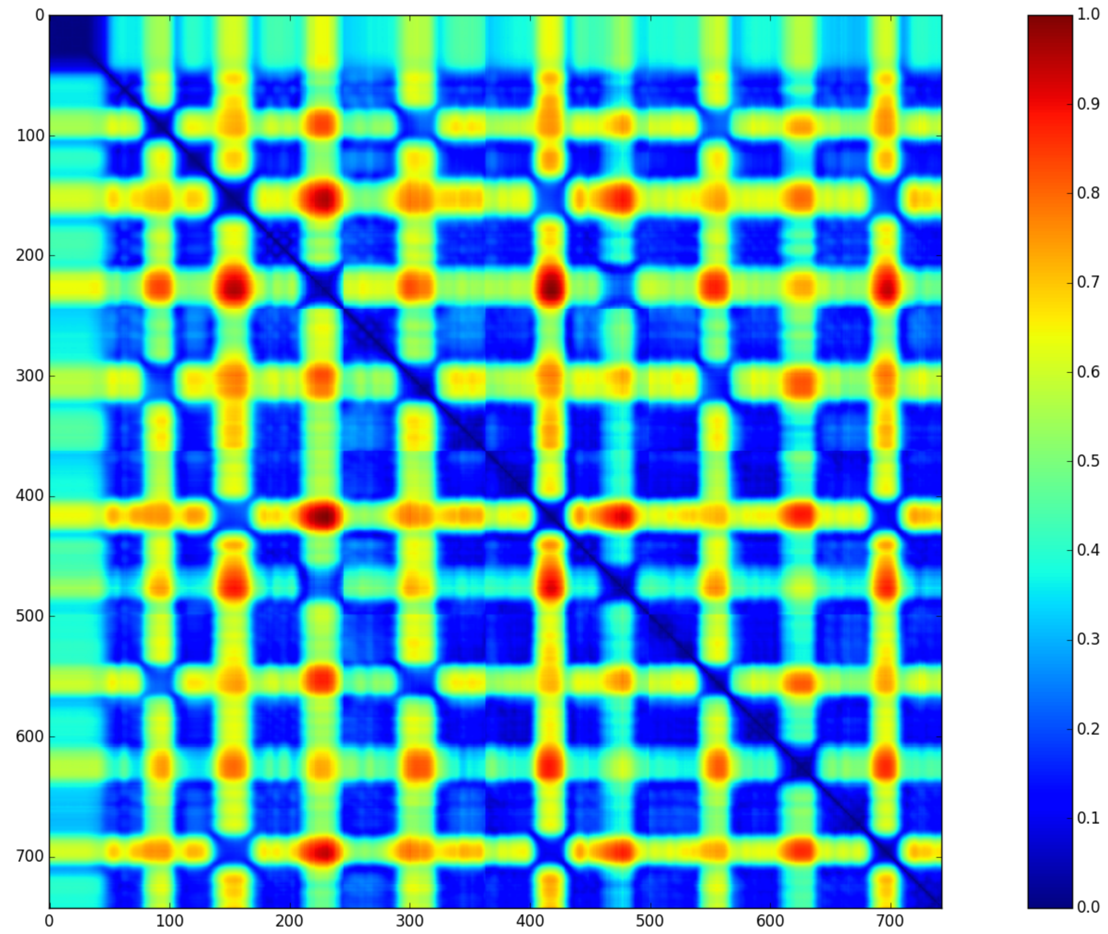


Figure C.7: The self-similarity matrix of the skeleton data for the Leonardo character mapped from similar, dark blue, to dissimilar, red.

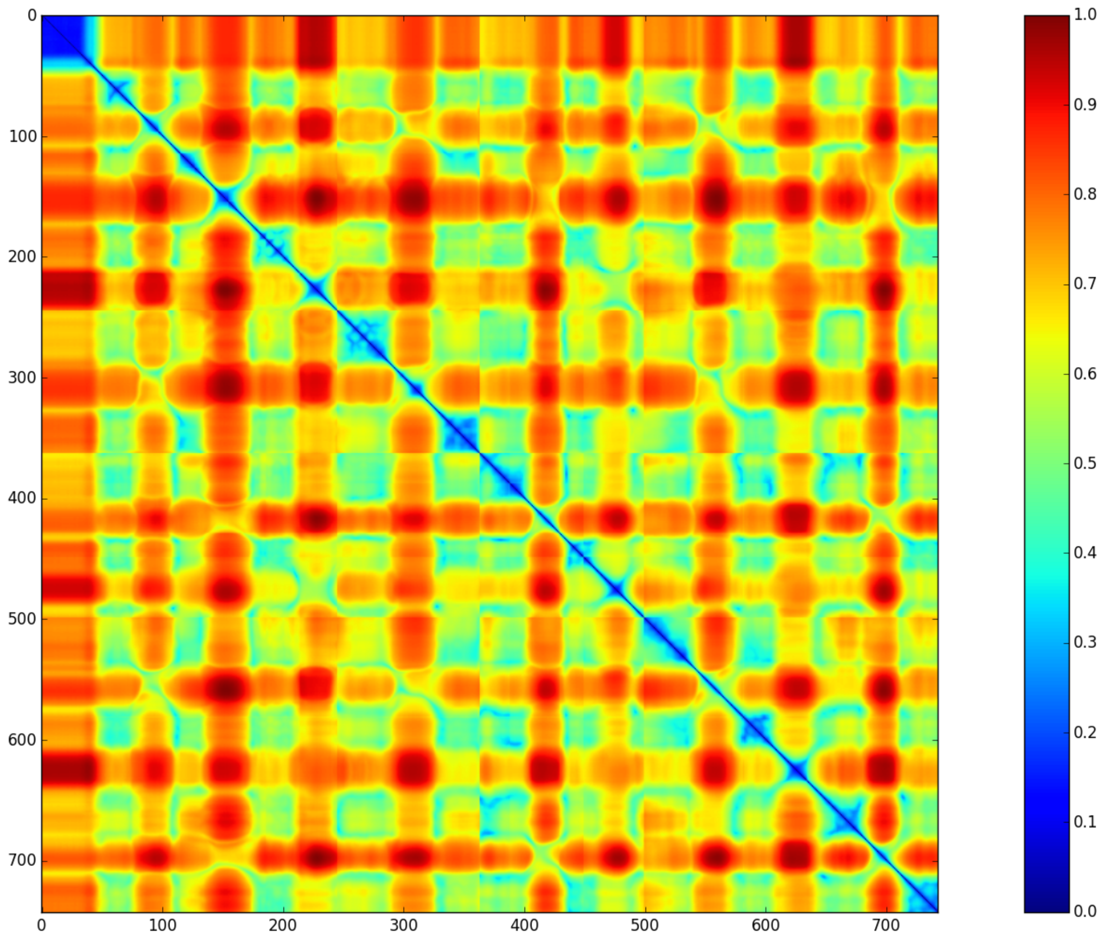


Figure C.8: The self-similarity matrix of the depth data for the Leonardo character mapped from similar, dark blue, to dissimilar, red.

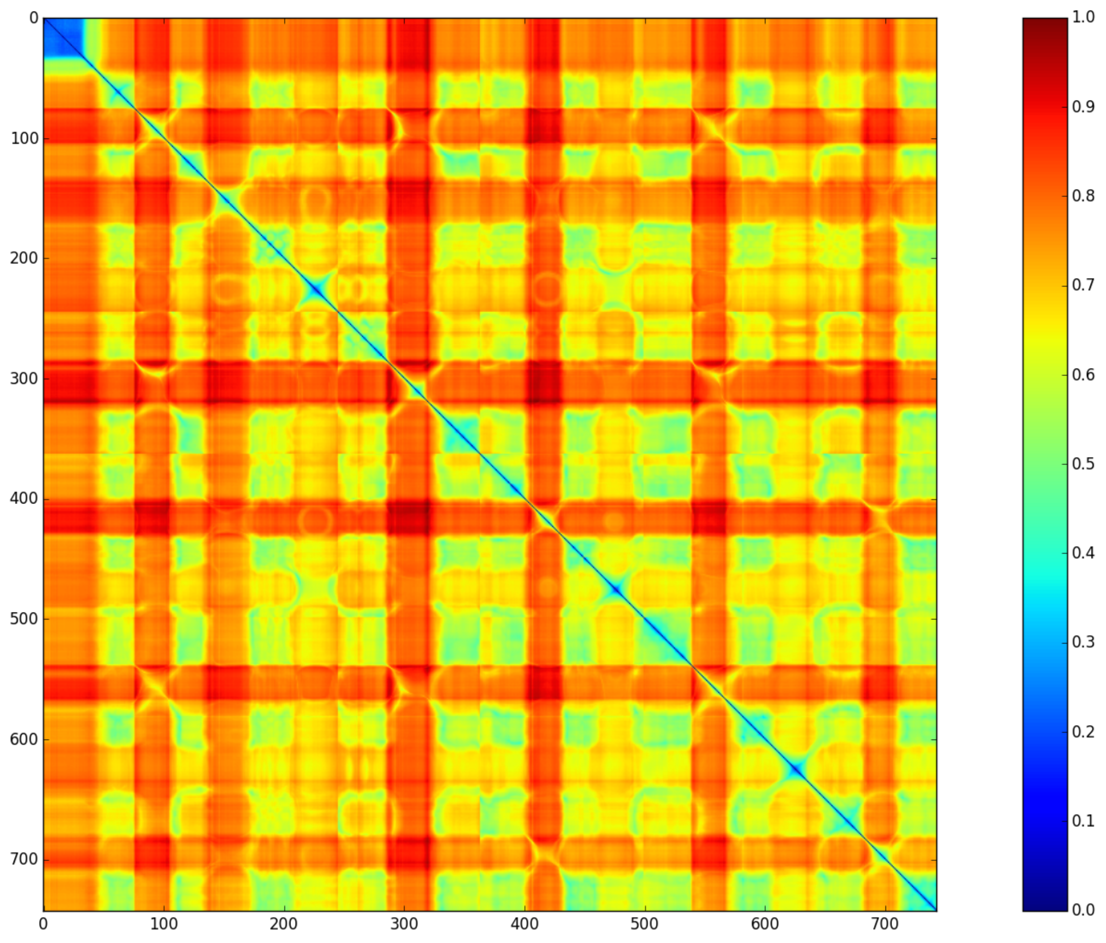


Figure C.9: The self-similarity matrix of the colour data for the Leonardo character mapped from similar, dark blue, to dissimilar, red.

D. Structure of Attachments

The structure of the attachments is listed below:

A - Source Code

B - User Study Videos

C - User Study Results

Bibliography

Chris Bahnsen. Kinectv2recordings - data acquisition for the kinect v2 for windows.

Chris Budd, Peng Huang, Martin Klaudiny, and Adrian Hilton. Global non-rigid alignment of surface sequences. *International Journal of Computer Vision*, 102(1-3):256–270, 2013.

Dan Casas, Margara Tejera, Jean-Yves Guillemaut, and Adrian Hilton. 4d parametric motion graphs for interactive animation. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pages 103–110. ACM, 2012.

Dan Casas, Margara Tejera, Jean-Yves Guillemaut, and Adrian Hilton. Interactive animation of 4d performance capture. *Visualization and Computer Graphics, IEEE Transactions on*, 19(5): 762–773, 2013.

Dan Casas, Marco Volino, John Collomosse, and Adrian Hilton. 4d video textures for interactive character appearance. In *Computer Graphics Forum*, volume 33, pages 371–380. Wiley Online Library, 2014.

OpenCV 2.4.13.0 documentation. Motion analysis and object tracking. URL `http://docs.opencv.org/2.4/modules/video/doc/motion_analysis_and_object_tracking.html#calcopticalflowfarneback`.

Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Image analysis*, pages 363–370. Springer, 2003.

Philipp Fechteler, Wolfgang Paier, and Peter Eisert. Articulated 3d model tracking with on-the-fly texturing. In *Image Processing (ICIP), 2014 IEEE International Conference on*, pages 3998–4002. IEEE, 2014.

Peng Huang, Adrian Hilton, and Jonathan Starck. Shape similarity for 3d video sequences of people. *International Journal of Computer Vision*, 89(2-3):362–381, 2010.

Martin Klaudiny, Chris Budd, and Adrian Hilton. Towards optimal non-rigid surface tracking. In *Computer Vision–ECCV 2012*, pages 743–756. Springer, 2012.

George A. Gescheider Lawrence E. Marks. *Psychophysical Scaling*, volume 4. 2002.

Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. In *IJCAI*, volume 81, pages 674–679, 1981.

Microsoft. Kinect hardware, a. URL `https://developer.microsoft.com/en-us/windows/kinect/hardware`.

Microsoft. Skeletal tracking, b. URL `https://msdn.microsoft.com/en-us/library/hh973074.aspx`.

Microsoft. Microsoft to consolidate the kinect for windows experience around a single sensor, 2015. URL https://blogs.msdn.microsoft.com/kinectforwindows/2015/04/02/microsoft-to-consolidate-the-kinect-for-windows-experience-around-a-single-ser

Thomas B Moeslund. *Introduction to video and image processing: Building real systems and applications*. Springer Science & Business Media, 2012.

Torben Poulsen. *Psychoacoustic Measuring Methods. Version 2.2*. 2005. 31230-08.

Arno Schödl, Richard Szeliski, David H Salesin, and Irfan Essa. Video textures. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 489–498. ACM Press/Addison-Wesley Publishing Co., 2000.

Andy Serkis. Gollum in the lord of the rings - performance capture. URL http://www.serkis.com/performance-capture-gollum.htm.

Jonathan Starck and Adrian Hilton. Surface capture for performance-based animation. *Computer Graphics and Applications, IEEE*, 27(3):21–31, 2007.

Zhengyou Zhang. Microsoft kinect sensor and its effect. *MultiMedia, IEEE*, 19(2):4–10, 2012.