UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO
DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO E AUTOMAÇÃO

LEANDRO MARTIN GUERTZENSTEIN ANGARE

# A DATA-DRIVEN METHOD FOR INTER-SIGNS ANIMATION IN SIGN LANGUAGES

# MÉTODO PARA ANIMAÇÃO ENTRE SINAIS EM LÍNGUAS DE SINAIS

CAMPINAS
2021

LEANDRO MARTIN GUERTZENSTEIN ANGARE

# A DATA-DRIVEN METHOD FOR INTER-SIGNS ANIMATION IN SIGN LANGUAGES

# MÉTODO PARA ANIMAÇÃO ENTRE SINAIS EM LÍNGUAS DE SINAIS

Dissertação apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Elétrica, na Área de Engenharia de Computação.

Dissertation submitted to the School of Electrical and Computer Engineering of the State University of Campinas in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering, in the area of of Computer Engineering.

Orientador: Prof. Dr. José Mario De Martino

ESTE TRABALHO CORRESPONDE À VERSÃO FINAL DA DISSERTAÇÃO DEFENDIDA PELO ALUNO LEANDRO MARTIN GUERTZENSTEIN ANGARE, ORIENTADO PELO PROF. DR. JOSÉ MARIO DE MARTINO.

CAMPINAS

2021

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Rose Meire da Silva - CRB 8/5974

Angare, Leandro Martin Guertzenstein, 1986-
An42d A data-driven method for inter-signs animation in sign languages / Leandro Martin Guertzenstein Angare. – Campinas, SP : [s.n.], 2021.

Orientador: José Mario De Martino.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Animação por computadores. 2. Avatares. 3. Ajuste de curva. I. De Martino, José Mario, 1958-. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título

Informações para Biblioteca Digital

**Título em outro idioma:** Método para animação entre sinais em línguas de sinais
**Palavras-chave em inglês:**
Computer animation
Avatars
Curve fitting
**Área de concentração:** Engenharia de Computação
**Titulação:** Mestre em Engenharia Elétrica
**Banca examinadora:**
José Mario De Martino [Orientador]
Léo Pini Magalhães
Maria Andréia Formico Rodrigues
**Data de defesa:** 22-11-2021
**Programa de Pós-Graduação:** Engenharia Elétrica

**Identificação e informações acadêmicas do(a) aluno(a)**
- ORCID do autor: https://orcid.org/0000-0002-5211-852
- Currículo Lattes do autor: http://lattes.cnpq.br/2001080833028559

**UNIVERSIDADE ESTADUAL DE CAMPINAS**
**FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO**

A Comissão Julgadora dos trabalhos de Defesa de Dissertação de Mestrado, composta pelos Professores Doutores a seguir descritos, em sessão pública realizada em 22 de novembro de 2021, considerou o candidato Leandro Martin Guertzenstein Angare aprovado.

Prof. Dr. José Mario De Martino (Presidente)

Prof. Dr. Léo Pini Magalhães

Profa. Dra. Maria Andréia Formico Rodrigues

*A Ata de Defesa com as respectivas assinaturas dos membros da Comissão Julgadora encontra-se no SIGA (Sistema de Fluxo de Dissertação/Tese) e na Secretaria de Pós-Graduação da Faculdade de Engenharia Elétrica e de Computação.*

**AGRADECIMENTOS**

Este trabalho é fruto de um longo processo de pesquisa, estudos, tentativas que fruíram em erros e acertos e com o qual espero poder contribuir para o desenvolvimento da animação digital, esta incrível confluência de arte e tecnologia pela qual tenho imensa apreciação. Certamente que este trabalho jamais poderia ter sido feito sem ajuda de uma imensa quantidade de pessoas, dentre elas meus agradecimentos:

Ao professor José Mario De Martino, cuja orientação desde que iniciei os estudos no programa de graduação na UNICAMP me motivaram a seguir na busca do conhecimento e cuja paciência ímpar possibilitou a realização deste projeto;

À minha família, que sempre esteve ao meu lado e, em especial, à minha mãe que sempre me apoiou;

Aos colegas do Departamento de Engenharia de Computação e Automação (DCA) que me ajudaram imensamente neste projeto, com imenso destaque à Carolina Monteiro Ferreira cujo projeto de iniciação científica auxiliou imensamente a conclusão desta dissertação;

Ao Centro de Tecnologia da Informação Renato Archer (CTI), disponibilizando o equipamento e laboratório para realização das sessões de captura de movimento e em especial a Ângelo Brandão Benetti;

À memória do diretor de animação Clovis Vieira, que inspirou a mim e tantos outros animadores no Brasil.

O presente trabalho foi realizado com apoio do CNPq, Conselho Nacional de Desenvolvimento Científico e Tecnológico – Brasil (Processo 160941/2013-0).

## RESUMO

A presente dissertação expõe a pesquisa que culminou na proposição de um método para síntese da animação de transições entre sinais de um avatar sinalizante da Língua de Sinais Brasileira (Libras). Nos últimos anos, tem sido cada vez mais incentivada e presente no dia a dia a acessibilidade em materiais multimídia (filmes, televisão, internet, etc.) fruto da demanda de uma parcela considerável da população que se sente mais confortável comunicando-se em língua de sinais. Uma forma de melhorar a acessibilidade de conteúdo multimídia em língua de sinais é a apresentação do conteúdo por meio de janela de língua de sinais. Nessa janela é exibido o vídeo de um apresentador real ou virtual. Um apresentador virtual de língua de sinais (avatar sinalizante) deve, assim como um intérprete humano, transmitir informação em língua de sinais de forma inteligível. Uma abordagem para se atingir essa meta é a concatenação de animações de sinais armazenadas em uma base de dados. Tal concatenação, objeto desta pesquisa, deve ser feito de tal forma que não prejudique a legibilidade, buscando movimentos suaves e naturais para o avatar. Pesquisa inicial realizada no âmbito desse trabalho de mestrado buscava uma solução no campo da animação procedural baseada em regras cinemáticas derivadas da Lei de Donders, porém como não há consenso em trabalhos acadêmicos de biomecânica humana para tal a solução adotada foi um sistema de animação híbrido entre baseado em dados (*data-driven*) e procedural. Esta solução foi possível criando-se um banco de dados de conjuntos de curvas de animação de transições entre sinais gravadas de intérpretes com equipamento de captura de movimento. Estas curvas de animação para serem armazenadas e manipuladas foram ajustadas em segmentos de curvas paramétricas. No presente trabalho foram exploradas a formulação de curvas de Hermite e de B-Splines, a primeira através de método próprio e a segunda utilizando-se de algoritmo proposto por Paul Dierckx que utiliza o gradiente conjugado para ajustar, por meio de processo de minimização, o conjunto de nós e coeficientes aos dados. Associado a estas curvas de animação estão conjuntos de parâmetros que descrevem os momentos anteriores e posteriores à transição utilizando Transformada Discreta de Cossenos (DCT) das curvas de animação dos sinais. Esses parâmetros são utilizados para encontrar o conjunto de curvas de animação de transição mais próximo ao intervalo que se quer sintetizar, utilizando distância mínima quadrática. Por fim, é apresentado um comparativo entre diferentes curvas utilizadas para sintetizar animações entre sinais de Libras utilizando como parâmetro o erro quadrático médio (RMSE) entre as animações geradas e o sinal original.

**Palavras-Chave:** Animação por Computador, Avatares, Ajuste de Curva.

## ABSTRACT

This dissertation presents the research which culminated in a method for inter-signs transition animation for a virtual Brazilian Sign Language (Libras) interpreter. In recent years, accessible multimedia material (films, television, internet, etc.) has been increasingly promoted and has become more present in our daily life, due to the demand of a population that feels more comfortable communicating in sign language. One way to improve the accessibility of multimedia content is to use an overlaid sign language window, presenting the content. This window displays the video of a real or virtual presenter. A virtual sign language presenter (signing avatar) must, like a human presenter, transmit information in sign language intelligibly. An approach to reach this goal is to concatenate animations of signs stored in a database. Such concatenation, which is this research's objective, must be made in a way as to not impact legibility, seeking natural and smooth movements. An initial research activity sought a procedural animation solution based on kinematic rules following a Donders Law's type of model, but since there was no consensus in human bio-mechanic academic studies for it the adopted solution was a hybrid data-driven and procedural animation system. This solution was possible by creating a database of animation curves recorded from sign language interpreters using motion capture (mocap). These animation curves, to be stored and manipulated, were fitted into parametric curve segments. In the present work, the formulation of Hermite and B-Spline curves were explored, the first using a method developed by the researchers and the latter using an algorithm proposed by Paul Dierckz using the conjugated gradient to adjust, through a minimization process, the knots and the coefficients of the B-Spline. Associated with these animation curves are parameter sets describing the moments prior and after transitions using Discrete Cosine Transformation (DCT) in the signs' animation curves. These parameters are used to find the closest transition animation curves to the transition to be synthesized, using minimum square distance. It is also presented a comparison between curves used to synthesize animations between Libras signs using as parameter Root Mean Square Error (RMSE) between the generated animations and the original signal.

**Keywords:** Avatars, Computer Animation, Curve Fitting.

## LISTA DE FIGURAS

## LISTA DE TABELAS

**SUMÁRIO**

# 1 INTRODUCTION

According to the *Instituto Brasileiro de Geografia e Estatistica* (Brazilian Institute of Geography and Statistics), IBGE (2012), the most recent Brazilian census shows that 9,71 million citizens can be considered having some sort of auditory deficiency of which 2,14 million (approximately 1.12% of the total population) are deaf or possess severe auditory deficiency. The government has shown concern on how to better integrate these citizens with new educational approaches. he existence of a great educational gap in the education of individuals with severe auditory deficiency or completely deaf in relation to those that present no auditory deficiency is widely documented (SOBEL, 2006). This is attributed to the language used to educate, in our case Portuguese, which is not the language used in their daily life, the Brazilian Sign Language (Libras). This means that both in oral and written communication those citizens have to deal with additional challenges, similar to a Brazilian student trying to learn calculus with a Russian textbook.

The government has also put forth the normative instruction (*instrução normativa*) 116, on December 18th 2014, by the Brazilian National Movie Agency (ANCINE). In its first paragraph, it instructs that henceforth all productions made with public resources should include a Libras translation. This creates a big demand for the production of audiovisual Libras content, which a signing avatar could supply.

These welcome initiatives put a high demand for Portuguese – Libras interpreters. A possible solution to meet demands when interpreters are not available is the use of a 3D signing avatar. The avatar can provide students with self-sufficient means and independence to learn and study at home, especially in early school years. Audiovisual content creators that need to adequate their production to the ANCINE normative instruction and have no means to hire a Libras interpreter can also benefit from a signing avatar, especially when dealing with interactive media.

Signing avatars have been the subject of extensive research as can be seen in MOREIRA et al (2011), AMARAL et MARTINO (2012), and SOARES et al (2017). This dissertation inserts itself in this collective endeavor by proposing a new method for transitioning arms’ movements between Brazilian Sign Language signs. This method shows its relevance when we consider generating real-time sign sentences, whereupon the 3D avatar has a database of signs and needs to concatenate between them to create new sentences, and this process needs to be done in a way as to create an animation that will look natural to facilitate understanding.

In summary, the goal is to present a method that synthesizes a natural-looking and smooth transition between two Libras signs, which would increase intelligibility and make the movement of the signing avatar seem more natural.

To achieve this goal, extensive research into computer animation has been done in order to understand how real-life arm movement is created and how this movement can be translated into digital information that can be manipulated for the signing avatar goal. An initial attempt was to find a model that could predict all arm movement, such as Minimum Work or Minimum Torque Change, but upon reviewing the literature on the bio-mechanics of arm movement this approach did not seem easily achievable. However, this literature review suggested us an approach whereupon we could synthesize new movement based on already existing similar movement (i.e. similar initial and final arm poses). For that, we decided to use motion capture (mocap) to create a database of real-life interpreters signing sentences in Libras and use the animation curves from these mocap movements to synthesize new transitions between signs.

To implement the method, we had to understand some questions pertinent to our research: how previous work in computer animation and in mechanics of arm movement can guide the proposition of a method to synthesize new movement, how to store and manipulate animation curves, and how to use preexisting animation to synthesize new ones. These questions are explored in the next chapters, organized as follows.

Chapter 2 presents the literature review aimed at finding models to human arm movement in real life, defining the propositions used in the method. The chapter also presents a review of animation and computer animation in context with this research, defining terms and concepts used in the later chapters.

Chapter 3 presents the methodology developed based on the propositions discussed in Chapter 2.

Chapter 4 shows the results obtained by applying the method described in Chapter 3 to a 3D Libras signing avatar. The chapter also presents a quantitative comparison based on the errors between the animation curves synthesized by the method and the original mocap animation.

Chapter 5 presents conclusions reached upon analyzing the data from the previous chapter and suggestions on how improvements can be made to the method presented, and further research is also discussed.

Partial results of the research have been published in the papers DE MARTINO et al. (2016) and ANGARE et al. (2015).

# 2 OVERVIEW OF CHARACTER ANIMATION

In this chapter, we review the main published works that establish the theoretical and practical foundation of this dissertation. The chapter presents the underlining concepts for the choices made in the method proposed. The chapter is divided into two sections:

Section 2.1 **Animation**: defining some of the major concepts of animation contextualized in our work, a more detailed text about these concepts is presented in Appendix D; we invite the more interested reader to venture into its pages since for academic brevity and focus we could not expand on the topic in this chapter;

Section 2.2 **Bio-Mechanics of Arm Movement**: discusses relevant research focusing theoretical aspects of arm movement modeling and propositions that constitute the foundations of the method proposed, answering the question presented in Section 2.1

## 2.1 ANIMATION

Research has been conducted towards the creation of a signing avatar, and the majority of work uses solutions already provided by software to transition between signs, recurring to the use of simple curves interpolation. GONCALVES, TODT et GARCIA (2015) and MURTAGH (2011) both use Blender, an open-source free-to-use 3D software available online, to interpolate the movement between Brazilian and Irish signs, respectively, in a virtual avatar. FERREIRA (2017) also presented an approach for a virtual avatar in Portuguese Sign Language, but the specifics of how the avatar transitions from sign to sign is overlooked and can only be inferred to be done by simple curve interpolation.

The previously mentioned research works reflect the low focus current research on signing avatars has on character animation. Our research proposes then to improve upon signing avatar animation, with a method for improving the smoothness of the movements of the signing avatar and thus increase intelligibility.

It is important to understand where in the animation pipeline the proposed method is inserted. For that, we can refer to a general outline of a 3D animation pipeline as described by PARENT (2012) and shown in Figure 2.1.

**Figure 2. 1. 3D Computer Animation Pipeline.**

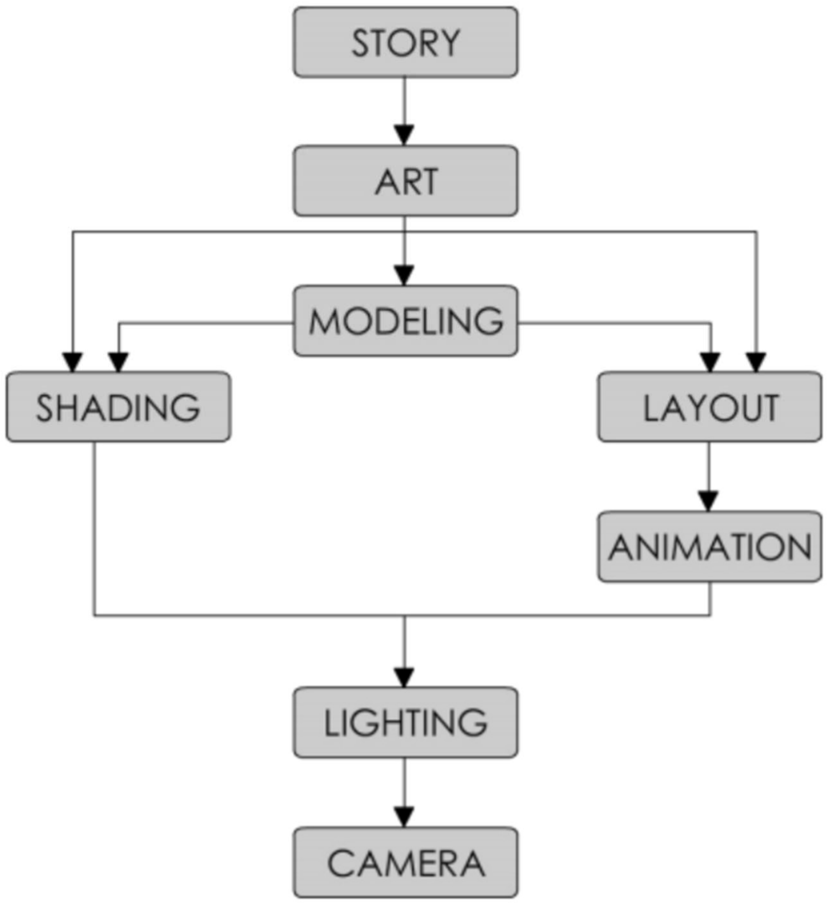


Image from PARENT (2012).

To implement our method we use a virtual avatar that has been designed (Art), Modeled, Shaded, set up in layout with the set, and is lighted to a final camera render. All these steps shown in Figure 2.1 are important to develop an animation piece, but our method will deal with only the synthesis of movement (Animation), that is the creation of animation curves comprised of a sequence of floating-point numbers that describe a virtual skeleton's joints' angles in a time range.

To understand how computers synthesize animation generally (i.e. what is usually done in most animated films nowadays) and in the context of the virtual characters we can make a parallel with the traditional hand-drawn process described by THOMAS, F. et JOHNSTON, O. (1995) and SHAMUS, C. (1990), summarized in Figure 2.2.

**Figure 2. 2. Diagram showing the movement creation pipeline in a cel system studio. The animation department (central blocks) receives layout, which is the background and the different layers and rough positions of characters, from the Layout Department (left block) and delivers the drawn frames for the movie for the Ink and Paint Department (right block) which is responsible for transferring the drawings to cel and coloring them. Inside the animation department, we can see the different animation professionals (animators, assistant animator, and inbetweener) and what they are responsible for (key poses, break down poses, and in-betweens, respectively).**

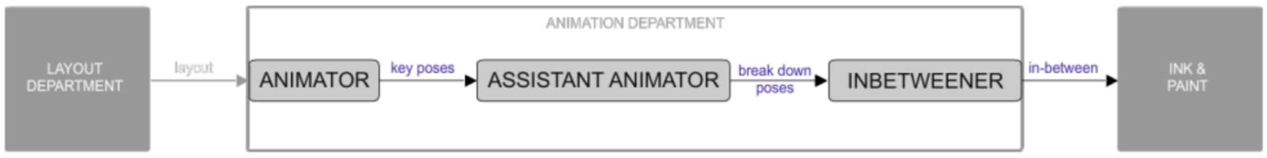


Image from the author.

While most computer animation software will try to fill the role of the inbetweener, creating the intermediate positions between different key poses, we propose in our method to go a step further and fill the role of the assistant animator as well (see Appendix D for details). For that, we don’t intend to simply interpolate between poses, but to use previously learned motions to create a more complex animation, as Frank Thomas and Ollie Johnston would put it, to decide how the avatar will move.

One way of gaining this knowledge is to use as a basis animation derived from real-life movement, which is a form of improving animation that can trace its roots to the beginning of XX$^{th}$ century animation with Fleischer’s rotoscope (FLEISCHER, M. 1917).

This mocap basis is not a solution by itself, so we refer back to PARENT (2012) on his taxonomy of 3D Computer Animation, which presents us with 3 different categories of animation to synthesize motion:

- **Artistic Animation** is the animation most commonly used when generating animation for film and television. In this type of animation the user (animator) defines specific positions of the character in time, commonly referred to as keyframes, and the system is usually only responsible for interpolating between these positions;
- **Data-Driven Animation** is where real-life information guides the generation of an animated virtual object. The most common technique in this category is the mocap, and one great work that inspired the method here presented is PULLEN et BREGLER (2002) discussed in more detail in Appendix D;
- **Procedural Animation** is usually the most computationally complex category

and it encompasses those types of animation that synthesize movement based on mathematical models. There are several types of animation in this category, some of which are clothes and hair simulation, crowds, and rag-doll systems.

Understanding the animation pipeline, specifically how animation curves enable a virtual character to move is the first step towards building a solution to the problem of 3D signing avatars' movements. Knowing how different methods create movement in virtual environments, as proposed by PARENT (2012), enable us to have a first glimpse into the direction our method can take. Will it be a more artistic approach where the animator will have more direct control; or perhaps a data-driven where animation from mocap is used to synthesize the transition between signs such as PULLEN et BREGLER (2002) proposed; or even a procedural method based on mathematical models for movements?

To better answer this, one needs to look into real life and what has been researched regarding arm movement, which is the topic for the next section.

## 2.2 BIO-MECHANICS OF ARM MOVEMENT

An understanding of the mechanics of human arm movement is required to represent it in three-dimensional animation. In the context of this dissertation, a study has been made to attain such understanding. The study focused only on research on perceived motion discarding references to neurological synapses, muscle response, etc. Presented here are some previous works done mostly in human motor and kinetics. As previously mentioned, our solution is data-driven, but that was not our first approach, which was a procedural type of animation, since a lot of research into human arm bio-mechanics had shown there was some underlying set of rules that could mathematically model its animation. We present here initially some of these works and will later show the works that have shown the strongest arguments towards a data-driven approach in opposition to a procedural one.

Before dwelling on the actual arm movement, it is noteworthy to point out the work in DONDERS (1864) since it is widely cited in the works presented below. Donders was a XIX$^{th}$ century physician that studied the human eye. One of his questions was how the human eye moves. He decomposed this movement into imaginary horizontal and vertical axis which he called the horizontal and vertical retinal meridians (Figure 2.3). The horizontal meridian is an imaginary line that crosses the eye from its center and it's parallel to a line that goes from ear to ear. This meridian represents the rotation of the eye going up (towards the forehead) and down (towards the chin). The vertical meridian is an imaginary line that crosses the eye from its center and it's parallel to a line that goes from the top the head to the mid point between the feet, this meridian represents the rotation of the eye going from one ear to the other. In his paper, the author states what is known as Donders' Law:

> ... to each position of the line of sight belongs a definite orientation of the horizontal and vertical retinal meridians relative to the coordinates of space (DONDERS, 1864)

**Figure 2. 3. Eye rotation axis.**

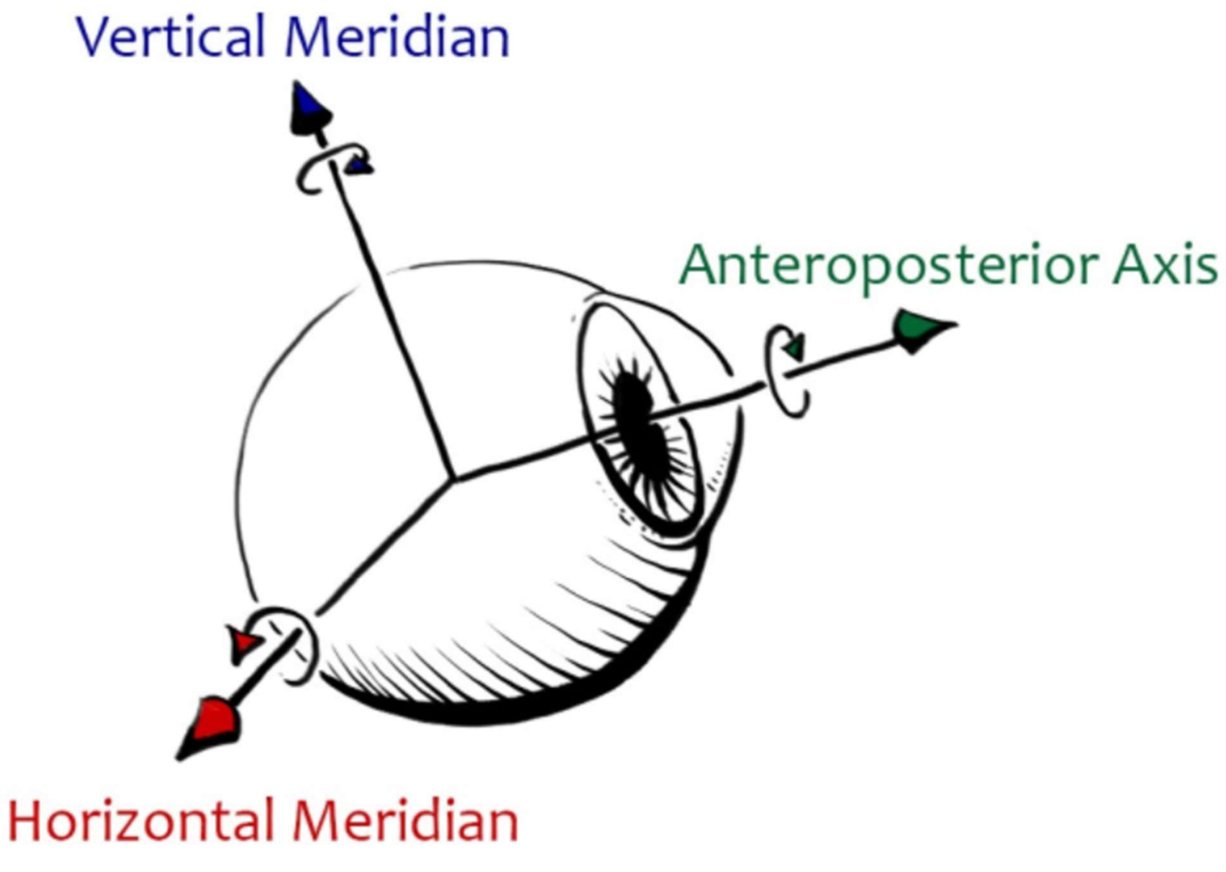


Image from the author.

That is, for every direction of the eye, although it is roughly a sphere and has three degrees of freedom, the rotations around the horizontal and vertical meridians are always the same, that is the angle in each of these axes have the same value, independent of the initial gaze orientation. This means that whatever the direction the eye is looking, the angle around the horizontal and vertical meridians are always the same. We can explain this with an example where different rotations can result in the same final result: a die rotating. Imagine we're seeing the die as in Figure 2.4 with the sides 1 up, 5 on our right, and 4 on our left, we'll call this configuration (a). If we rotate it 180 degrees clockwise on its y-axis we have the configuration (b), where side 6 was adjacent to side 4 and it's now up and side 3 that was adjacent with side 1 is now left. We now rotate it again, but on its x-axis and have the configuration (c) showing the side 2 that was adjacent with side 6 on our right and side 1 that was adjacent with side 5 on top. This same result can be obtained from configuration (a) with an 180-degree rotation on its z-axis. So a final configuration (c) can be the result of 2 different rotations! Donders showed that is not the case for the human eye, a final eye globe rotation can only be done by a definite rotation in the horizontal and vertical meridian axis.

**Figure 2. 4. Rotations of a die. We can go from an initial configuration (a) to a final configuration (c) by rotating 180º on y (configuration b) and then 180º on x, or by rotating 180º on z.**

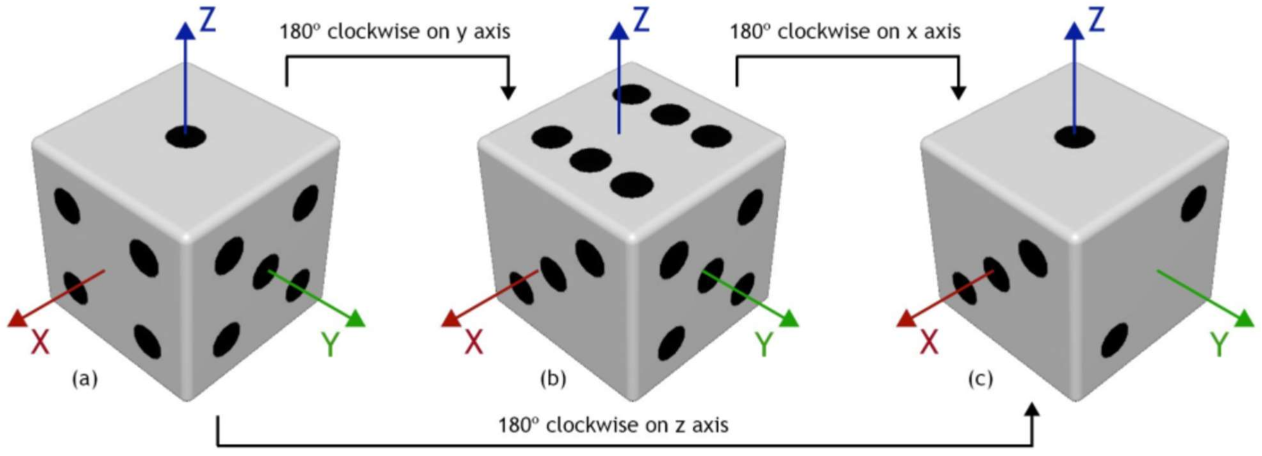


Image from the author.

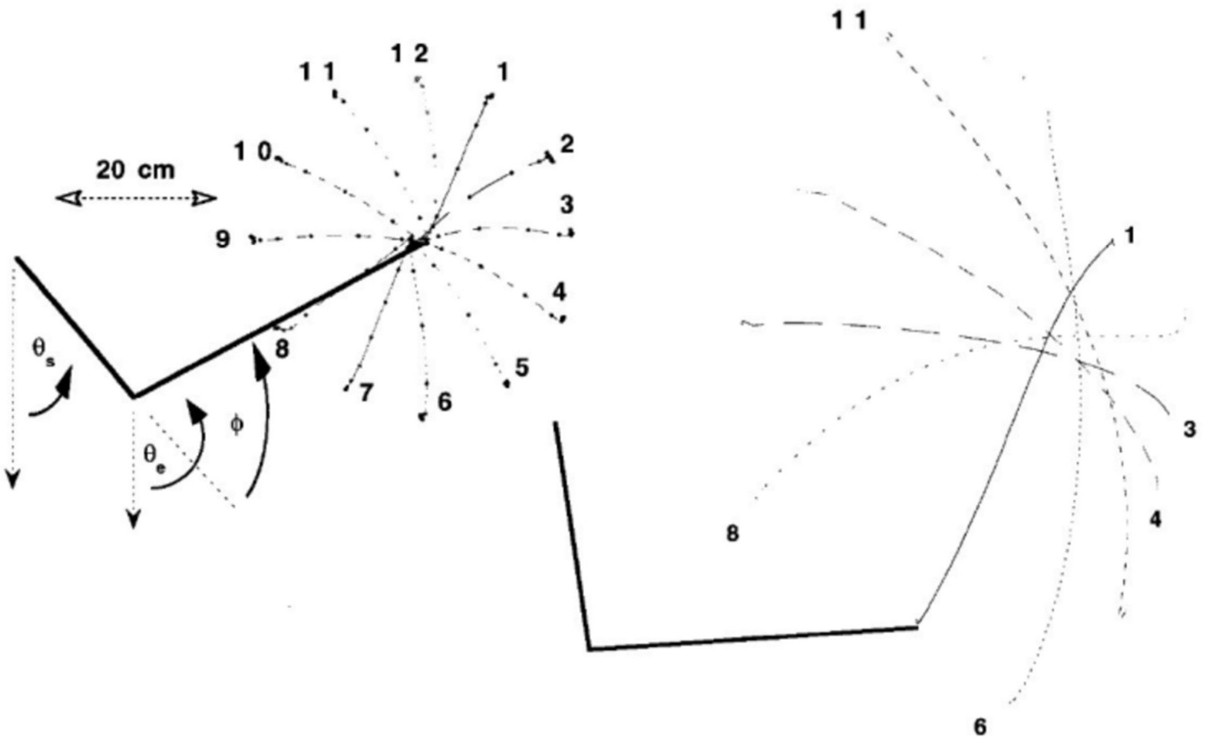
One of the explanations for Donders' Law is the fact that the eye has a very restricted rotation on its anteroposterior axis, commonly known as the line of sight (Figure 2.3) (CYBERSIGHT, 2016), and thus the eye globe can be thought as having almost only 2 DOF. If we go back to our die example from Figure 2.4 we can have an analogy saying that if it was an eye globe it cannot rotate on its z-axis (which works as the anteroposterior axis in the human eye), and if we restrict rotations on positive numbers and less than 360 degrees we can only have configuration (c) with the 180 degrees rotations in y and x. Rotation around the anteroposterior axis is used to adjust to the tilting of the head (LUDWIG et CZYZ, 2019).

Donders' Law was held true and further elaborated in various works like EWART et al (2016) and MAROTTA, J. J., MENDENDORP, W. P. et CRAWFORD, J. D (2013) out of this dissertation's scope and it was taken as an indicator in arm movement research of an underlying and defining rule that could predict exactly what rotations would be needed to achieve certain motion goals, i.e. there is a law that states exactly how an arm movement is created based on its final position. For this dissertation, this would indicate that a pure procedural animation with an underlying mathematical model could perfectly describe arm movement, since as with Donders' Law for the eye orientation it would be possible to devise a function that maps final arm position to rotations of shoulder, elbow, and wrist. As can be seen in the next pages, various researches have been done that show this function is not as straightforward as one might like, there are too many exceptions to a straightforward mathematical function to arm movement.

In the work by GOTTLIEB et al (1997), it was analyzed what the linear correspondence between elbow and shoulder torques in reaching arm movement was. Two movement sets were created: the first one (CO) with the initial hand position fixed and six subjects instructed to reach 12 small targets (cotton balls with 2 cm. diameter) in the parasagittal plane equally spaced in a circle or ellipse centered on the initial hand position similar to the hour markers in a clock (Figure 2.5), the second movement set (CX) was similar but initial hand position was outside the targets circle and the circle or ellipse diameter was increased from 40 cm to about 50 to 80 cm[1], using only three subjects.

[1] Although in the work it is not clear if the 50 to 80 cm corresponds to radius or diameter, it is assumed the latter.

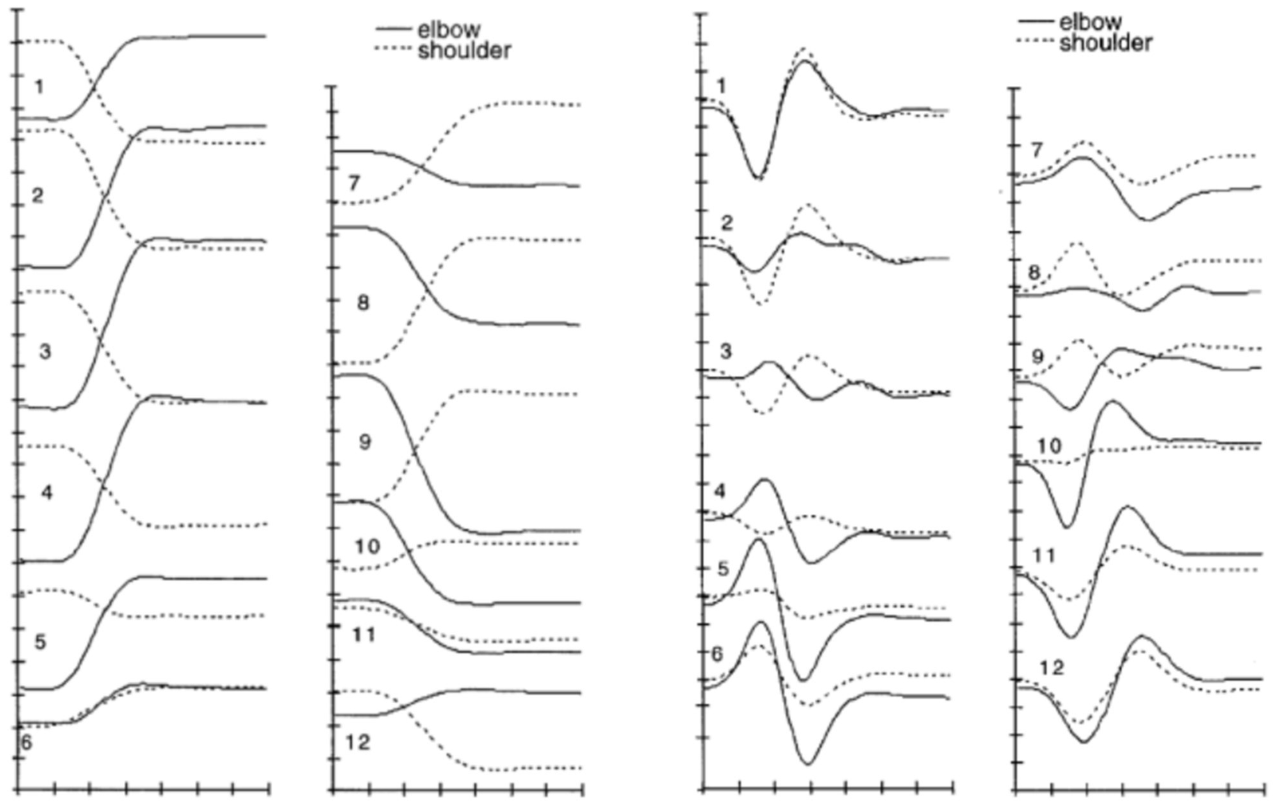
**Figure 2. 5. Experiments from GOTTLIEB et al. (1997). “A: center-out movements: subjects moved from an initial location out in front of the shoulder to a series of 12 targets, located at the hours of a clock and 20 cm from its center. Parasagittal target plane was aligned with the arm. Heavy lines show the arm segments at the initial position when the subject prepared to move. Thin lines show average finger paths to the 12 targets by subject T. Markers are drawn at 50-ms intervals. Angle variables for Eq. 2 are defined here. B: centercrossing (CX) movements: subjects started from different initial locations, elliptically arranged around a center near shoulder height. They moved to targets located at the opposite side of the ellipse. Six of the 12 movements directions are illustrated, the end points of the movements are identified by the numbers. Other 6 movements were made in the opposite directions. Heavy lines show the arm segments at the initial position of the 1 o’clock movement. Thin lines are the finger paths of individual movements.” (GOTTLIEB et al. (1997))**



Images from GOTTLIEB et al. (1997).

The movement was captured using "... electro-optical motion measurement system (OPTOTRAK-3010) recorded at 200 samples / seconds the locations of four markers attached to the shoulder, elbow, wrist, and index fingertip", and applied to a virtual skeleton with five degrees of freedom including shoulder, elbow and wrist. bi-dimensional model was used to calculate the torques, but still it is odd that the paper cites 5 degrees of freedom since a standard 3D model would have 7 DOF (3 for shoulder, 1 for elbow and 3 for wrist) and a 2D model would have only 3 DOF. Nonetheless, the authors analyzed the shoulder and animation curves using Newtonian equations and statistic data to estimate arm segments mass. The authors concluded that there is a linear relationship between shoulder and elbow torque, as can be seen in Figure 2.6. Furthermore, they propose a linear relationship between those torques (Equation 2.1), with a constant $K_d$ varying systematically with movement direction.

**Figure 2. 6. Results from GOTTLIEB et al. (1997). Shoulder and elbow torque computed in the first (left) and second (right) movement set proposed by GOTTLIEB et al, 1997. The linear relationship can be seen in the graph's shape similarities.**



Images from GOTTLIEB et al. (1997).

The authors point out an interesting analysis that when performing similar movements (in the first set corresponding to reaching targets with 10 to 20 cm distance) one can see "...fairly linear relations between joint angles and of course the hand path itself is fairly straight", but that does not hold when analyzing movements too different, i.e. targets too far apart.

DESMURGET, GRÉA et PRABLANC (1998) conducted an experiment where participants were asked to reach for targets positioned in a space in front of them. Initial configuration was vastly changed compared to GOTTLIEB et al (1997) and a new question was raised: does final hand orientation (i.e. wrist rotation) change other joints' rotations? If so, are they unique? The experiment consisted of nine right-handed participants, without any motor or neurological impediment, comfortably seated in a chair with their shoulder immobilized, preventing torso and clavicle interference, asked to reach targets in front of them. The apparatus used as targets consisted of self-illuminated cylinders that could be freely rotated, and when the experiment began lights were turned off making the room completely dark with only the cylinder's self-illumination visible. Movements were made varying the cylinder's position (3 configurations: front, right and front-right diagonal), orientation (3 configurations: 60º and 20º counter clockwise and 20º clock-wise), and initial hand position (3 possibilities: waist height, 35 cm above the waist and 20 cm right, and 60 cm above the waist and 35 cm right). Three interesting results are relevant from this experiment:

1. The authors have shown that variations of initial configuration produced different shoulder and elbow angles, that is "[...] posture reached by the arm was not the same for the different starting points".
2. Applying Principal Component Analysis (PCA) with all individuals' data it was verified that 8 out of 9 subjects the main factors clustered together indicate that "postural variations exhibited by these subjects presented a high degree of similarity". In the discussion, it is stated that "Beyond the previous observations, it is noteworthy that the behavior of the motor system was not random, but quantitatively deterministic both between and within subjects. Modifying the object configuration and/or the hand initial location had a similar effect from subject to subject."
3. Another result presented is that "... the position and orientation of the hand in space are unlikely to be controlled through separate independent neural pathways" - meaning that wrist rotations influence shoulder' and elbow's rotations.

These three results - influence of initial configuration; similarity in movements between and within subjects; and influence of hand orientation on arm posture - could be seen consistently in most of the works presented in this section, and thus were taken into consideration when devising the approach proposed (Chapter 3).

MAROTTA, MENDEDORP et CRAWFORD (2003) conducted two experiments investigating the parallel visuomotor processing theory of grasping - the idea that movement is split and processed in parallel in three independent components: transport of the arm, orientation of the hand, and grasp - and Donder's Law applied to grasping movement, stating that the final configuration of the arm system is independent from its initial state, suggesting that arm torsion is uniquely dependent on final arm position relative to the body. The first experiment involved six subjects and investigated the contribution of upper and lower arm to grasp orientation, requiring subjects to reach for a rectangular target block that rotated from 0º to 90º; the second experiment investigated upper and lower arm torsion contribution and their interaction with Donders' Law, presenting eight subjects with a "...wider range of grasping space by requiring subjects to reach out and grasp target blocks that were presented in one of three orientations (horizontal, three-quarter, and vertical) at nine different positions in the vertical presentation board", Figure 2.7. Both experiments were conducted with right-handed subjects with no motor or visual disabilities or impediments.

**Figure 2. 7. Experiments from MAROTTA, MENDENDORP et CRAWFORD (2003). Position of targets for the first (A) and second (B) experiments.**

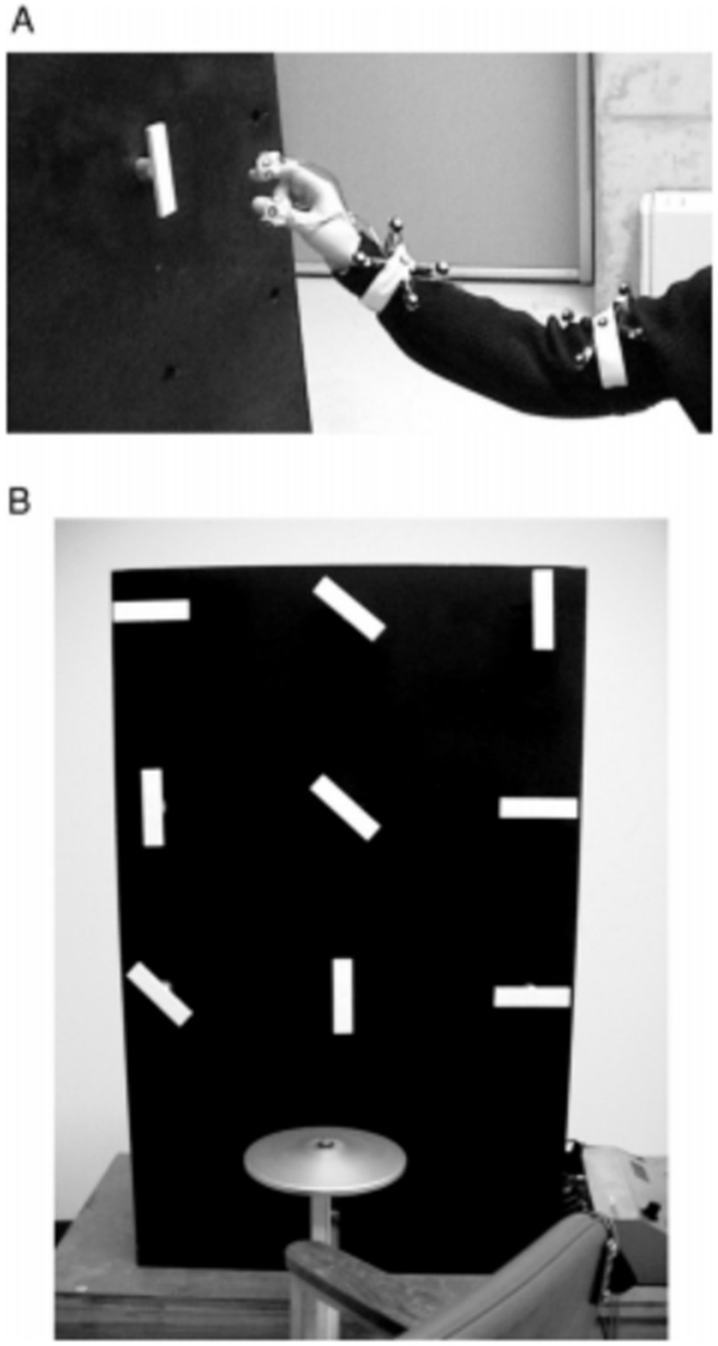


Image from MAROTTA, MENDENDORP et CRAWFORD (2003).

As a result, it was evident that arm movement and hand orientation were created in a holistic manner, that is, change in the hand's orientation and space position is the result of simultaneous torsion and rotations in shoulder, elbow, and wrist, and not created by parallel processing. Parallel visuomotor processing theories of gasping are stated by the authors as:

> One way the brain could deal with the complexity of a reaching and grasping movement is to split the total movement plan into parallel functional modules, each involving a small number of degrees of freedom (Arbib 1981; Arbib et al. 1985; Gentilucci et al. 1991; Jeannerod 1981, 1992; Jeannerod and Biguer 1982; Paulignan et al. 1991a,b; Stelmach et al. 1994). According to this theory, prehension movements directed to visual stimuli can be partitioned into three independent components: the transport of the arm, the orientation of the hand, and the grasp (Arbib 1981; Jeannerod 1988). The transport component involves proximal joints and muscle groups, while the grasping component involves distal joints and muscles (Arbib 1990; Jeannerod 1984; Jeannerod and Decety 1990; Lacquaniti and Soechting 1982; Stelmach et al. 1994). […] Within the initial acceleration of the arm transport, the hand's posture and orientation [grasp] are adapted according to the task. Hand shaping, determined by visual assessment of object dimensions, will also be assumed during the acceleration phase in anticipation of contact. (MAROTTA, MENDECORP et CRAWFORD, 2003)

Regarding Donders' Law, MAROTTA, MENDENDORP et CRAWFORD (2003) also state "... rather than beginning with Donders' Law and then calling the torsional shifts we observed violations of Donders' Law, is to consider that there are more general laws governing arm kinematics that only provide Donders' Law under restricted task circumstances, like a constant set of grasp orientations".

These experiments provide two insights into human arm movement: first, movement is processed in a holistic manner and as such should not be synthesized in computers as set of independent constrains (for instance, using Inverse Kinematics to position hand in space and then rotating wrist to reach desired hand orientation); and second that there are some rules (Donders' Law) that do hold true when analyzing movements very similar, but they do not satisfactorily model general arm movement.

ADMIRAAL, KUSTERS et GIELEN (2004) present a larger experiment aiming to analyze a broad range of arm movements, from pointing to gasping with different velocities and accuracy, and compare such movements with predictions using minimum-work model, minimum-torque-change model, and Donders' Law. As with the previous experiments, right-handed subjects with no visual or motor impediments participated, and a similar number of participants were used: 6 in pointing and 9 in reaching experiments, compared to 6 and 3 subjects in GOTTLIEB et al (1997), 9 in DESMURGET, GRÉA et PRABLANC (1998) and 6 and 8 in MAROTTA, MENDEDORP et CRAWFORD (2003). The first experiment, pointing, used stereoscopic projection to create 5 virtual yellow sphere targets with 1.5 cm diameter, laid out with one in the center of a virtual square and the remaining 4 in its vertices. Subjects were asked to start and end two movements on the central target, first going clockwise in order on the square's vertices targets and second performing the same movement but in a counter-clockwise path, each consisting of 8 cycles (repetitions). After each movement (counter and clockwise), participants were asked to point to random targets. The second experiment asked participants to point the tip of the index finger on targets that appeared in the 3D space (as opposed to the previous experiment where the targets were positioned on a 2D surface distant 100 cm) and hold it there until the target disappeared. Contrary to the first experiment, in the second experiment subjects' arms did not stretch in all the targets (shoulder, wrist, and elbow forming a straight line).

For the first experiment, pointing, the minimum-torque and minimum-torque-change models estimated very closely the 3D path done by subject's index fingertip.

Donders’ Law was not verified on the first experiment, since doing the same movement and reaching the same position, arm torsion was not the same, which would be expected. Torsion also deviated from the other methods estimations even though they did predict an increase in arm torsion with each cycle, contrary to the same value established by Donders’ Law.

When participants were asked to change the way they pointed to targets, increasing accuracy or requested to be as fast as possible, it was seen an increase or decrease in arm torsion, again not corresponding to the proposed models (Donders’ Law, Minimum-work and Minimum-torque-change models).

As in the works of DESMURGET, GRÉA et PRABLANC (1998) one can see (Figure 2.8) the same change in arm torsion depending on initial and final arm configuration, which would indicate, again, that although models did not predict every aspect of the movement (predicted trajectory on the pointing experiment, but not arm torsion), there are underlying rules that the motor system follow between and within subjects.

**Figure 2. 8. Results from ADMIRAAL, KUSTER et GIELEN (2004). Humeral axis-rotation measured in subjects as a function of initial and final target. Different symbol corresponds to different subject. It is possible to see the effects of different initial and final targets are consistent in all individuals, for instance on graph C that represents final target 3 the torsion reduces about 10 degrees for all subjects when they start movement from target 4 instead of 2.**

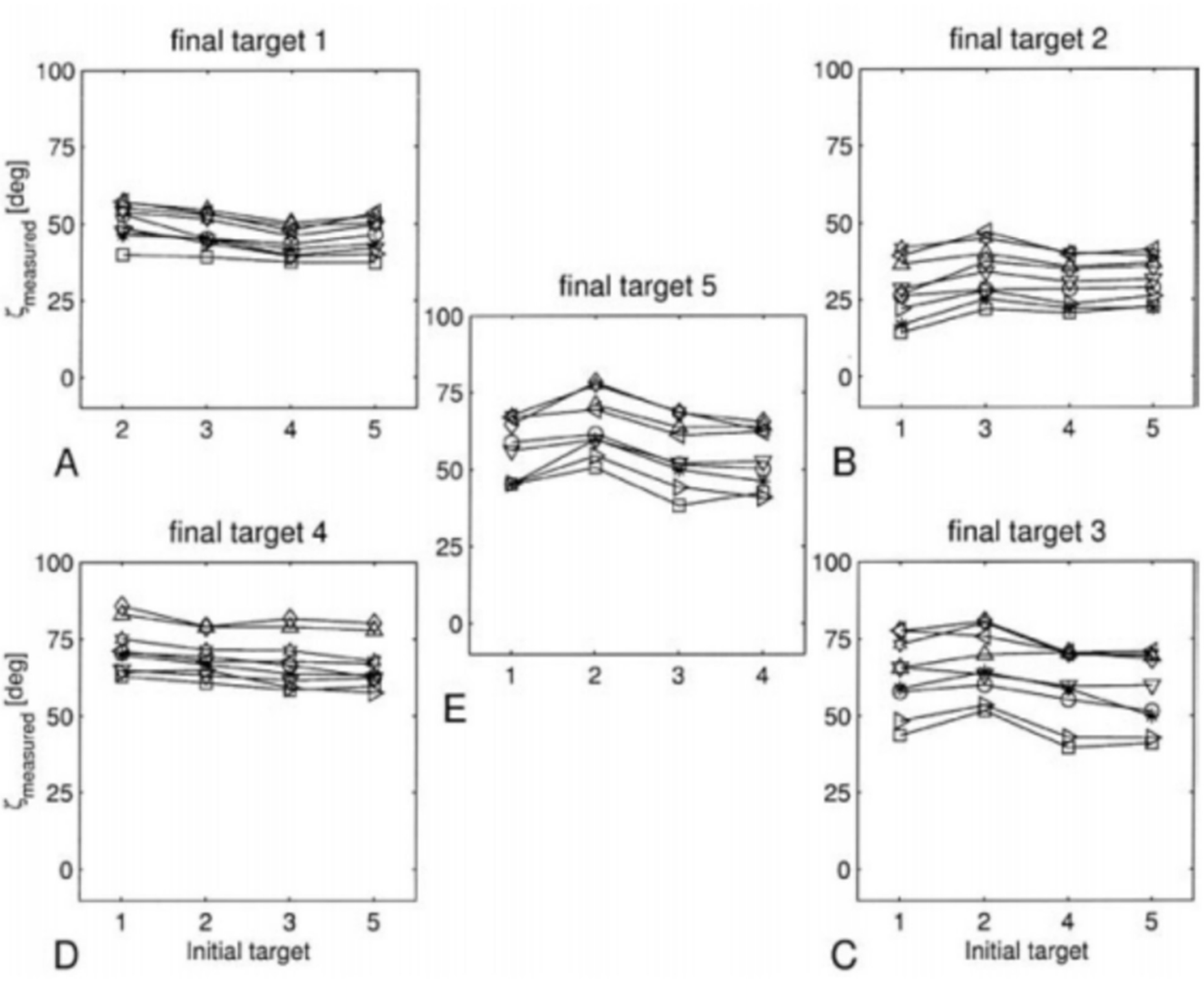


Image from ADMIRAAL, KUSTERS et GIELEN (2004).

The previous work was questioned by EWART et al. (2016), where a similar experiment was conducted using real targets instead of virtual reality. Eight right-handed subjects with no previous history of neurological or muscular disorders were asked to point (first experiment) or reach (second experiment) six different cylindrical targets departing from five or seven (number varies from subject to subject) initial hand positions. Differences in how movement was to be performed were also introduced as the previous work: comfortable movement, quick reach (fast), or grasp (accurate). Authors point out in the conclusion that:

> “Taken together our findings and aforesaid considerations suggest that a Donders' like law for the arm should be stated in three parts, as follows. At each spatial location relative to the body to which one points, there corresponds closely related postures of the arm independent of movement speed, or starting arm position. At each spatial location relative to the body at which an object with a given orientation is grasped, there corresponds closely related postures of the arm, regardless of the arm's starting position. Physiological variability is intrinsic to the rule, the postures will vary movement to movement by a few degrees. Hermens and Gielen (2004) tested the predictions of various posture-based and trajectory-based models of motor planning and control. While they claimed that their experimental results differed significantly from the predictions of all models considered, notwithstanding they acknowledged that "Of the models considered, Donders' law best predicts the experimental data” (EWART et al, 2016)

What the authors suggest thus contradicts the works previously mentioned, specifically in that there is only one configuration and movement possible for a final hand position, independent on the initial arm configuration, even though they do acknowledge variations, but not due to movement speed, but to "the variability of central neural activity".

It is important though to notice the results in Figure 2.9, where even the most "well-behaved" motion does vary significantly, especially in the grasping motion where variations seem to be on the order of 20º.

**Figure 2. 9. Results from EWART et al. (2016). Variations of arm angles for two individuals reaching 5 different targets from 7 different initial hand positions**

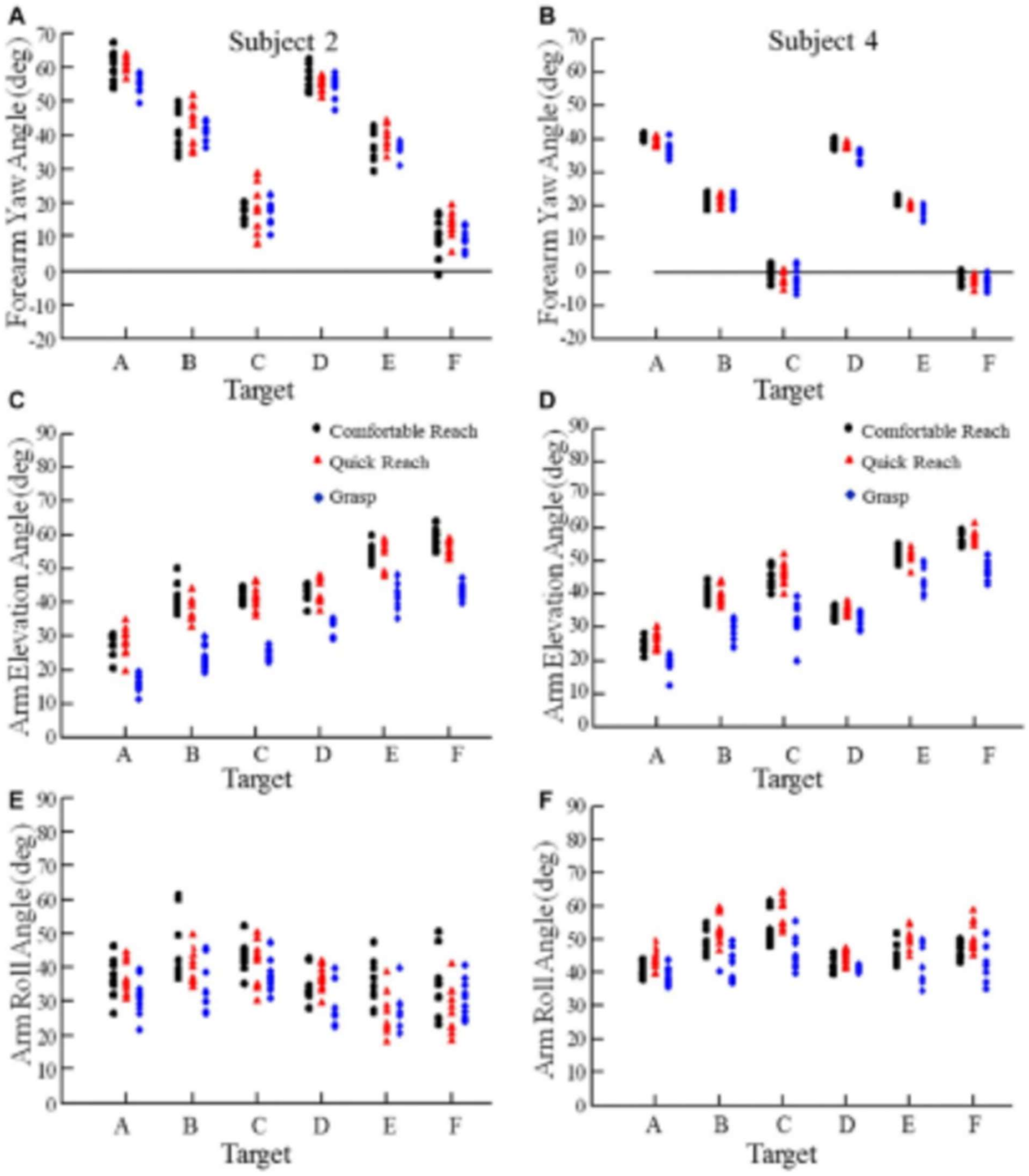


Image from EWART et al (2016).

Another problem in this experiment is that, since it was not their goal, researchers do not propose any actual model, instead they just refer to some general Donders' Law like rules.

Having reached an understanding that a straightforward mathematical model to create a procedural animation technique for our method is not possible at the moment, we decided on a hybrid data-driven / procedural method, which is proposed in the next chapter.

## 2.3 FINAL REMARKS

This chapter presented an overview of concepts and previous works that provided the base for the research to the method for inter-sign animation in Libras. It is divided into 2 2 sections, **Animation,** and **Bio-Mechanics of Arm Movement**.

In **Animation** we presented the animation production pipeline, in which it is defined the role of the animator, responsible for creating the main or key poses of the animation; the assistant animator or break down man, responsible for roughing out the poses made by the animator and creating the break down poses; and the inbetweeners, responsible for creating the intermediate drawings between poses and break downs. The latter task, because it requires less skill and is almost a mechanical process, has been replaced more and more often by software, and it is this dissertation's goal to present a computer method to replace the work of the assistant and inbetweener for arm movement in Libras.

Animation can be created in a multitude of ways. PARENT (2012) presents a taxonomy, dividing this process into 3 types: Artistic, Data-Driven, and Procedural.

To select the appropriate type of animation to use in this project we sought to understand how the arm moves. As a first approach to this method, we have sought a procedural animation  approach based on mathematical models describing arm movement. However, after studying the different models presented in previous researches, we realized that no known model provides a general extensive model to synthesize the broad movements required in sign language.

Works presented in Section 2.2, although initially intended to provide a model such as the minimum-torque to guide a procedural animation for inter-sign movement, presented a plethora of models and views towards modeling human arm movement, on which there is not even a consensus on whether there is a general universal model. But they did provide some of the propositions used to create the method proposed, which are: Similarity of movement (all experiments point out that similar final and, in most cases initial, arm configuration produce the same movement), initial configuration interference (although not a consensus, initial configuration of the arm including its rotations and velocity do appear to change the movement and even final configuration of the arm), and wrist interference (wrist rotations are not isolated and should be taken into account to synthesize whole arm animation).

Thus, instead of pursuing a global model with procedural animation, our method uses data-driven animation, based on similarity of movement, to retrieve an already existing animation similar to that intended to be synthesized. To retrieve this animation, initial configuration and wrist interference are taken into account.

While theoretical basis was covered in this chapter, the next one discusses the methodology used for developing the inter-sign approach to generate the animation of signing avatars, the focus of this dissertation.

# 3 METHODOLOGY

As presented in the introduction of this dissertation, the main objective of this research is to develop a method for transitioning arms' movements between two different Brazilian Sign Language (Libras) signs. The goal is to present a method to create natural and smooth transitions, which would increase intelligibility and make the movement seem more natural in a signing avatar.

A first question was raised: what type of computer animation is better suited for this method? In the previous Chapter we present the different animation types defined by PARENT (2012). In that chapter, we also argue that, although procedural animation seemed to be an appropriate approach to animate arm movement during signing, previous work review related to arm movement revealed that there is still no robust and generic model, as pointed out by EWART et al (2016), that would adequately represent the dynamics of arm movement during signing.

Although we could not find a robust and generic model to describe all arm movements, we gained knowledge from previous researches to guide us on the development of the proposed method. The main idea behind our method is to use segments of mocap animation to transition between signs, in a fashion inspired by PULLEN et BREGLER (2002). This idea is aligned with the 3 propositions from Section 2.2: similarity of movement, initial configuration interference, and wrist interference. When looking for a similar mocap movement for the transition between signs we use the initial and final configuration of the avatar's arm and wrist, thus we abide by the propositions: the movement to be used to transition will consider the initial configuration of arm; wrist configuration will interfere on the movement, and transitions that start and end in similar configurations will present a similar movement.

To recapitulate, what we need is a method that receives as input two Libras signs and outputs a movement transitioning between these signs.

Our first step was to create a database of existing transitions, similar to what PULLEN et BREGLER (2002) used as the basis to synthesize animation. Our database is composed of mocap data of 4 Libras interpreters signing four sentences each. The sentences were specially designed to have a wide range of arm movement between signs. Mocap data was transferred to a virtual avatar and cleaned. Segmentation was done, indicating beginning and end of signs and transitions between them. For each transition between signs we stored 7 animation curves – 3 for shoulder's rotation in x, y, and z, 1 for elbow rotation in x, and 3 for wrist's rotation in x, y, and z – as Hermite and as B-Spline curves (more details on curve fitting is found in Appendix B), so we can compare performance between those two curve representations. This transition **curve database** is used to retrieve a similar transition that will serve as a basis to synthesize the new transition animation.

This similarity is based on how the arm finishes the sign we want to transition from (first sign) and how the arm starts the sign we want to transition to (second sign), in accordance with the propositions shown in Chapter 2. To mathematically model these similarities, **parameters** were extracted from the mocap animations. These parameters are obtained by using Discrete Cosine Transformations (DCT) on *N* final frames of the first sign's animation curves and *N* starting frames of the second sign's animation curves. More information about the DCT process can be found in Section 3.5. These DCTs were stored for each mocap transition in a separate **parameter database**, which is used to compare with the signs we want to transition.

These two steps, construction of curves database and parameters database, were processed offline and thus did not encumber the system with unnecessary calculations.

Having a database of existing mocap transitions and an associated parameters database, the method can now use them to find a similar transition to the one desired between signs and use it to create a movement connecting such consecutive signs. To achieve this, our method establishes 3 steps which are outlined here and discussed in detail in the next sections:

1. **Parameter Extraction**: to find a similar transition in the mocap database, we need to first extract the parameters from the signs we want to transition. To conform with the proposition that similar initial and final arm configuration (joints' positions and rotations) create a similar movement, that is the path that arm makes when going from the same initial and final position tend to be similar across different humans, we need to compare the transitions we have in the database with the transition information we have from the signs (i.e. animation curves). As with the parameters extracted from the mocap transitions, we also use the first sign's last *N* frames and the second sign's initial N frames to create a DCT of the animation curves. More details in

Section 3.5.

2. **Transitions Search**: having the signs animation curves' DCT (comparison parameter) we use it to find a similar transition in the curves database, that is, a mocap transition that has a similar DCT to the signs we want to transition. This is done by Euclidean distance between the input parameters and all the transitions in the database, with the closest being chosen. More details in Section 3.6;
3. **Curves Processing**: Apply the similar transition animation curves found in the database in the interval between the input signs, adjusting them to the signing arms positions and maintaining the animation's speed and acceleration, that is maintaining C2 continuity on the animation curves. More detail in Section 3.7.

Each step is performed by different modules that have specific functionalities. Figure 3.1 presents the method overview with these modules in mind, how they interact with each other and with data from input and the curves and parameters databases. The modules and interactions between them are described below.

**Figure 3. 1. Method overview. The animation interface passes through a sequence of <u>frames</u> describing disconnected signs which are processed by the parameter extractor in order to get <u>descriptors</u> which are used to find a closely related curve in the descriptor database by a transitions search module; once the closely similar curve is found, its id is then passed on to the curves processor which, having received also the signs frames from the animation interface in order to produce smooth concatenations, obtains the <u>curve</u> from the database and then returns the respective transitions <u>frames for the final animation</u>.**

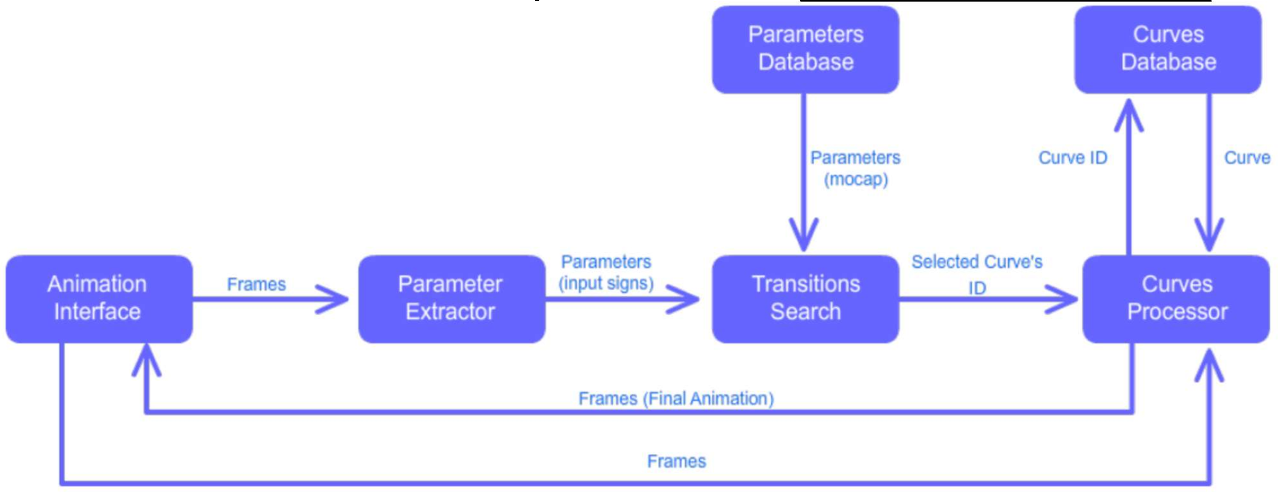


Image from the author.

The input comes from the **animation interface** that requests for transitions between signs; this could be software controlling the signing avatar or a user in a 3D program. In our case, we used a 3D software that had a virtual avatar signing Libras sentences from mocap sessions but with the interval between each sign being cut off. Whichever the animation interface, the main point is that it already has a set of animation curves describing a virtual skeleton's movement that comprises two consecutive signs, and those are used to feed the system. These animation curves are presented as vectors of consecutive floating-point numbers, one vector for each animation curve and each number indicating the animation curve's value (that is, joint rotation) in the specified time in frames.

The **parameter extractor** analyzes frame data (i.e. animation curves of the first and second signs) and retrieves the DCT parameters that describe the curves in a more meaningful way for the system to compare them with the mocap curves parameters stored in the **parameters database**. This is necessary since a direct comparison of animation curves' values will require a great amount of comparisons and not necessarily take into account important characteristics such as speed and acceleration. A curve transformation (DCT) was thus used so these parameters, velocity and acceleration, could be incorporated. To create this descriptor database several mocap sessions were performed following a protocol comprised of simple sentences with wide range of motion, these were created with the aid of LIBRAS interpreters and are further detailed in Section 3.1.

It is also important to point out again the separation between **parameter** and **animation curve database**; since all searches are based on parameters it's possible to select between different curves implementations (Hermite and B-Spline) when applying the desired transition. That means that the method searches for the closest transition parameter in the parameters database and with that can decide to use the implemented Hermite or spline representation of animation curves for the mocap transition.

A **transition search** is done, comparing parameters from the initial and final arm configuration of signs to the mocap transition parameters database in order to find the closest match, i.e. a mocap transition that had similar initial and final arm configuration the signs we want to animate. This conforms to previous researches shown in Section 2.2 by which, even though there doesn't seem to be a universal model for arm movement, initial and final configuration of the arm will lead to very similar movement (the 3 propositions). The proposed method also differs here from the work of PULLEN et BREGLER (2002) in that we compare the seven animation curves of the arm (shoulder x, y and z rotation, elbow x rotation, and wrist x, y and z rotation) while they just used one animation curve to find a similar mocap animation in their database.

Having found the transition that has the closest parameters to the interval between signs we want to animate, it is necessary to insert the animation curves from the mocap database in the frame range between the first and last signs (interval to be animated). To better manipulate these animation curves they were stored as parametric curves in the curves database, more specific B-Spline and concatenated Hermite curves. To fit the mocap animation into Hermite curves a new algorithm is proposed by us, and for B-Spline curve fitting it was used the algorithm proposed by DIERCKX (1993) (see Appendix A for details). These curves are inserted between signs by manipulating knots (B-Spline curves) and curve points and tangents (Hermite curves), and this is described in Section 3.7. With the mocap curves applied in the frame range between signs we successfully animate a transition between Libras signs with an animation that is based on real-life interpreters, thus providing a natural movement to the signing avatar.

As stated before, the method proposed can also be divided into 2 different stages: **offline** and **online** processing. Offline processing is all the work done to create the curves and parameters database. Online processing is when a transition is required to concatenate two signs, and the system is in use by a user requesting to animate an avatar between two signs.

Finally, we applied this method to a set of Libras sentences by excluding manually the original inter-sign transition frames and using different curve models to synthesize a new movement. The new interpolated animation curves are compared to the original mocap data using RMSE. Details on these evaluations can be found in Chapter 4.

## 3.1 DEVISING MOTION CAPTURE MOVEMENT

The method proposed is based on a richly populated bank of transition movements. With this in mind, four different sentences were proposed to be signaled which would encompass a broad arrange of arm movement for transitions and at the same time had a meaning by themselves. These sentences were:

1. Hello, my name is Machado, I'm going home. Would you like to eat with me?

(Portuguese version: Olá, meu nome é Machado, vou para casa. Você quer comer comigo?)

2. The yellow painting is very beautiful. Who did it? (Portuguese version: O quadro amarelo é muito bonito. Quem pintou?)
3. I drove fast to my home to check the oven. (Portuguese version: Dirigi rápido para minha casa para verificar o forno.)
4. On Mondays, I love eating fried pastry at the farmer's market. (Portuguese version: Às segundas, adoro comer pastel na feira.)

A fifth, more complex and long sentence was devised but, after signaling by a first interpreter, it was deemed too complex to arrive at a consensus on how it should be signed by the other interpreters. This was an unforeseen problem and is discussed further in Chapter 5.

The four sentences used were devised using the ACESSA BRASIL (2017) system and the advice of one of the interpreters. Before capturing, a consensus on the signs used to each sentence was reached between interpreters having in mind standardized signaling and although those signs were carried out on the motion capture sessions, there were minor deviations on order, which can be accounted for personal traits of each interpreter. Instead of forcing the interpreters to follow a strict sign order, it was deemed more profitable for the project to have a wider range of transitions.

## 3.2 MOTION CAPTURE SESSION AND CLEAN UP

Motion capture took place in two sessions in the motion capture laboratory at Centro de Tecnologia da Informação Renato Archer (CTI), using 8 Vicon infrared cameras at 120 frames per second. On each interpreter, there were positioned 51 reflexive markers on the hips and up (Figure 3.2), that is, excluding legs since they are not used to sign. Markers were positioned at fingertips but due to low accuracy on the capture system they could not be used, so instead fingers animations were created manually (see next chapter). Each interpreter also signed a Consent to Use of Image and Captured Material for scientific purposes, available in Appendix C. Sessions were also recorded with a standard mini DV recorder, generating video references at 30 frames per second and 720 x 480 pixels in resolution.

**Figure 3. 2. Motion capture session at CTI facility. 2 infrared cameras can be seen on both sides of the interpreter, who is wearing the standard black clothes with the reflective markers (grey spheres).**

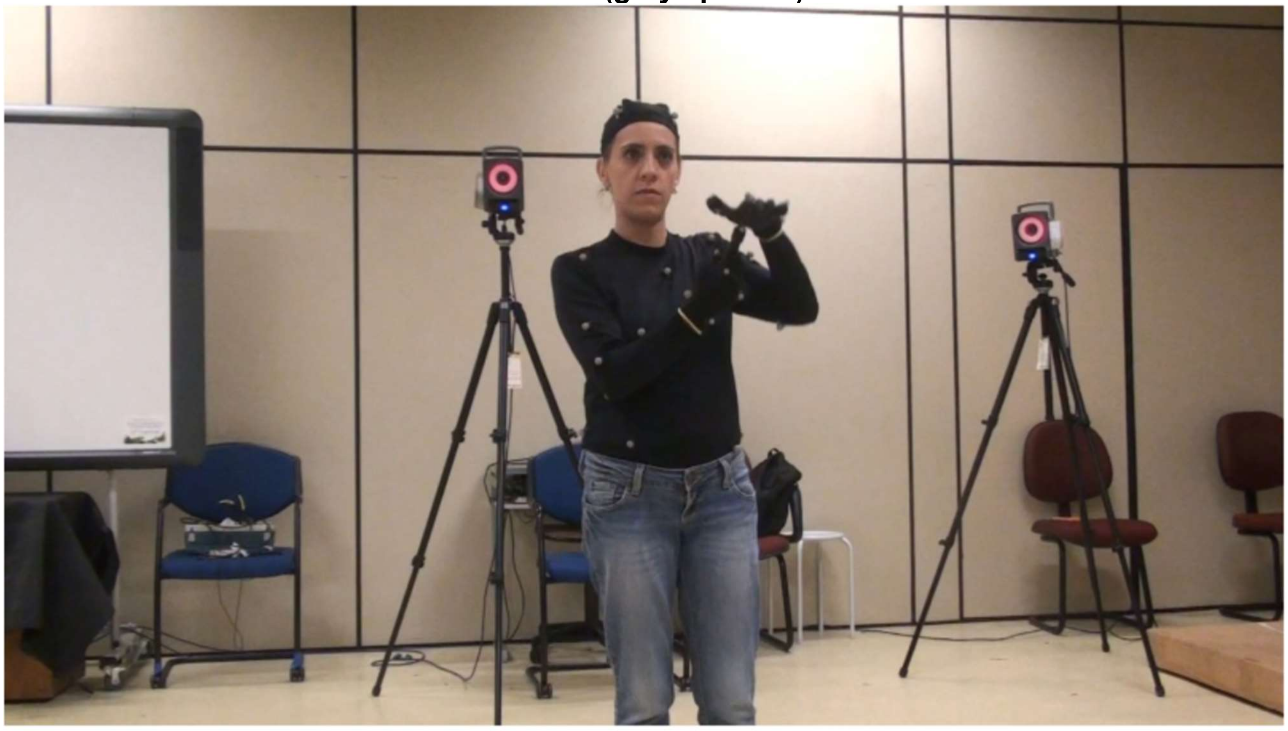


Image from the author.

Each infrared camera captures a bi-dimensional grayscale image at a rate of 120 samples per second, with each bright spot being one of the reflective markers. The 8 images generated are then processed using Vicon software to estimate each three-dimensional position.

With each marker position in 3D space, it is possible to discern a virtual skeleton to allow the deformation of a polygonal mesh in a process called solving motion. Again it was used Vicon's software and default skeleton to achieve this. This process, however, needs to be supervised since two common errors arise:

1. **bad labeling:** each marker, in order to be used to estimate a joint's position and rotation, needs to be identified by a unique label. Sometimes the system does not assign the correct label and so it is up to the motion capture technician to identify this error and correct it.
2. **gaps:** when there is no sufficient information about a specific marker in one or more cameras the system cannot estimate its position in space, creating gaps

in which there is no marker present as shown in Figure 3.3. These have to be addressed since a lack of marker in a single frame can create a noise that could make a movement seem unnatural. To correct this the software offers basically two tools: a spline and a rigid body interpolator. The first one creates a smooth transition in the gap but can produce unrealistic or unnatural results, the second one takes into account marker distance between itself and neighboring markers that move together in a rigid body fashion, such as in the head or the markers located in the forearm. This is a time-consuming and very important part of the process, and sometimes when there are too many gaps movement cannot be reconstructed and so the captured data is lost.

**Figure 3. 3. Vicon's Blade software interface (cropped) in which we can see the markers (blue and white spheres) on the left and the graphical representation of its position in three-dimensional coordinates (left) on the graph editor. Each black dot on the graph represents a keyframe, that is, a position in space estimated in that particular frame by the system. A gap is present from frames 1409 to 1506, represented by a continuous curve.**

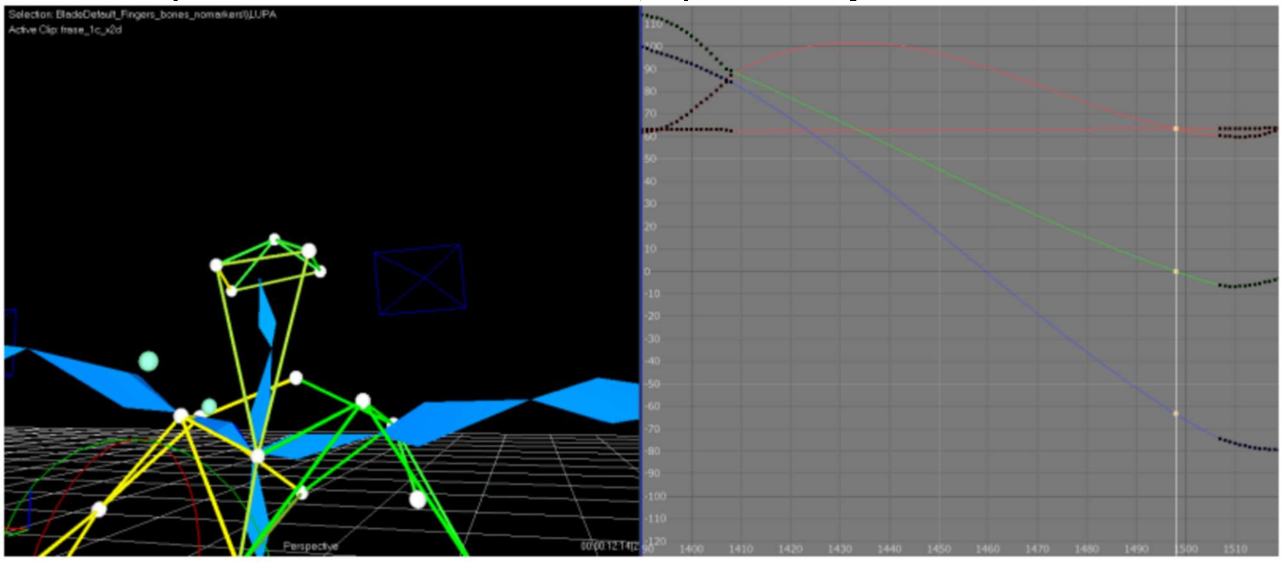

Selection: BladeDefault_Fingers_bones_nomarkers\LUPA
Active Clip: frase_1c_x2d
Perspective
1400
1410
1420
1430
1440
1450
1460
1470
1480
1490
1500
1510

Image from the author, generated in Vicon Blade.

Processing the markers' 2D images into 3D positions uses computer vision algorithms and was taken from the system provided by Vicon. If the reader is interested, a survey recommended on the topic is presented in MOESLUND, HILTON et KRÜGER (2006).

After this process the virtual skeleton is animated with the signs, that is, for each joint's DOF an animation curve describing the change of rotation values (in degrees) is attached. For each interpreter's sentence a skeleton was exported to an fbx file which was used on the next stage to apply to a 3D model and create the transitions and sign labeling.

## 3.3 SENTENCE SEGMENTATION

The skeleton provided by the Vicon system lacks two important features: finger animation and sign segmentation.

For the first feature, finger animation, a supplementary skeleton chain was created in each hand, including proximal, middle and distal virtual phalanges for the five fingers[2] and a control attached to the wrist joint by which the animator could control the joints' rotations (hyper-extension / flexion and adduction / abduction) in a 1 to 1 connection as described in O'HAILEY (2013). The software used for this process was Autodesk Maya.

For the second feature, sign segmentation, a manual approach was chosen over automatic systems such as the one presented in STARNER et PENTLAND (1998) since to use them it would be necessary a training phase for the machine learning requiring already segmented signed sentences. This approach consists of playing the mocap animation with the virtual character and noting down in which frames the signs started and ended, with this process being validated by the Mini DV video references described in the previous section and later with one of the interpreters.

The manual approach uses Python scripts to create an **XML sentence segmentation file** for each interpreter, indicating segments of each sentence with signs' and transitions' start and end information. XML was used for its wide adoption, including its use in sign language projects (AMARAL, 2012), and its human-readability (JONES et DRAKE, 2002). The latter has an advantage of smaller files, but that is not an issue since sentences are not as long as to create big files. Using Autodesk Maya's timeline and script editor it was possible to iterate through each sentence and determine each of the segments. An example of this XML file can be seen below in Code 3.1.

The first, root, element in the XML file is the interpreter's name used mainly to identify uniquely each sentence for the human operator, since it is not used in the system in any manner whatsoever. Each root element has a sentence attached to itself, in this project ranging from 1 to 4, but could be indefinitely expanded. In each sentence there are the

[2] The thumbs' virtual skeleton had also only three joints, namely the metacarpal, proximal and distal phalange.

segments needed for the project, each corresponding to a child element, which has the word being signed or the transition number as its ID; these elements include integer attributes *start* and *end* indicating start and end frame as a closed interval (e.g. [0, 235]), an integer attribute *pos* indicating the position of the element in the sentence's segments sequence, and a boolean attribute **word** indicating if the segment is a word (or, more correctly, a sign) in which case it is true.

**Code 3. 1. Code for the XML file segmenting sentences for each interpreter. Note that in each sentence there is an Intro (first element) and an End (last element) elements which are the interpreter's movement going to and reaching the T position.**

```
<Daniele>
  <frase1>
    <Intro end="235" pos="1" start="0" word="False"/>
    <TRANSITION1 end="292" pos="2" start="236" word="False"/>
    <Ola end="355" pos="3" start="293" word="True"/>
    <TRANSITION2 end="424" pos="4" start="356" word="False"/>
    <meu_Nome end="473" pos="5" start="425" word="True"/>
    <TRANSITION3 end="550" pos="6" start="478" word="False"/>
    <machado end="663" pos="7" start="551" word="True"/>
    <TRANSITION4 end="705" pos="8" start="664" word="False"/>
    <vou end="760" pos="9" start="706" word="True"/>
    <TRANSITION5 end="814" pos="10" start="761" word="False"/>
    <casa end="915" pos="11" start="815" word="True"/>
    <TRANSITION6 end="950" pos="12" start="916" word="False"/>
    <voce end="964" pos="13" start="951" word="True"/>
    <TRANSITION7 end="1016" pos="14" start="965" word="False"/>
    <comer end="1066" pos="15" start="1017" word="True"/>
    <TRANSITION8 end="1120" pos="16" start="1067" word="False"/>
    <comigo end="1300" pos="17" start="1121" word="True"/>
    <TRANSITION9 end="1406" pos="18" start="1301" word="False"/>
    <End end="1610" pos="19" start="1407" word="False"/>
  </frase1>
  <frase2>
    <Intro end="216" pos="1" start="0" word="False"/>
    <TRANSITION1 end="303" pos="2" start="217" word="False"/>
    <Quadro end="365" pos="3" start="304" word="True"/>
    <TRANSITION2 end="424" pos="4" start="366" word="False"/>
    <amarelo end="473" pos="5" start="425" word="True"/>
    <TRANSITION3 end="508" pos="6" start="474" word="False"/>
    <bonito end="624" pos="7" start="509" word="True"/>
    <TRANSITION4 end="661" pos="8" start="624" word="False"/>
    <muito end="676" pos="9" start="662" word="True"/>
    <TRANSITION5 end="704" pos="10" start="677" word="False"/>
    <pintou end="793" pos="11" start="705" word="True"/>
    <TRANSITION6 end="818" pos="12" start="794" word="False"/>
    <quem end="843" pos="13" start="819" word="True"/>
    <TRANSITION7 end="882" pos="14" start="844" word="False"/>
    <End end="1179" pos="15" start="883" word="False"/>
  </frase2>
  <frase3>
    <Intro end="250" pos="1" start="0" word="False"/>
    <TRANSITION1 end="320" pos="2" start="251" word="False"/>
    <Dirigi end="346" pos="3" start="321" word="True"/>
    <TRANSITION2 end="376" pos="4" start="347" word="False"/>
    <muito end="400" pos="5" start="377" word="True"/>
    <TRANSITION3 end="443" pos="6" start="401" word="False"/>
    <rapido end="498" pos="7" start="444" word="True"/>
    <TRANSITION4 end="546" pos="8" start="499" word="False"/>
    <ate end="611" pos="9" start="547" word="True"/>
    <TRANSITION5 end="640" pos="10" start="612" word="False"/>
    <casa end="676" pos="11" start="641" word="True"/>
    <TRANSITION6 end="705" pos="12" start="677" word="False"/>
    <ver end="731" pos="13" start="706" word="True"/>
    <TRANSITION7 end="755" pos="14" start="732" word="False"/>
    <forno end="958" pos="15" start="756" word="True"/>
    <TRANSITION8 end="1055" pos="16" start="959" word="False"/>
    <End end="1433" pos="17" start="1056" word="False"/>
  </frase3>
  <frase4>
    <Intro end="229" pos="1" start="0" word="False"/>
    <TRANSITION1 end="272" pos="2" start="230" word="False"/>
    <Segundas end="381" pos="3" start="273" word="True"/>
    <TRANSITION2 end="436" pos="4" start="382" word="False"/>
    <adoro end="497" pos="5" start="437" word="True"/>
    <TRANSITION3 end="538" pos="6" start="498" word="False"/>
    <comer end="568" pos="7" start="539" word="True"/>
    <TRANSITION4 end="619" pos="8" start="569" word="False"/>
    <pastel end="755" pos="9" start="620" word="True"/>
    <TRANSITION5 end="803" pos="10" start="756" word="False"/>
    <feira end="954" pos="11" start="804" word="True"/>
    <TRANSITION6 end="1079" pos="12" start="955" word="False"/>
    <End end="1246" pos="13" start="1080" word="False"/>
  </frase4>
</Daniele>
```

This sentence segmentation file structure was successfully used in the project, but its structure deviates greatly from what would be considered a good XML file structure. This issue is addressed in Chapter 5.

Joints rotations on each frame for all animations were also exported in an **XML DOF rotations file** (Code 3.2) with the root element indicating the total length of the animation; children element were the joints for left and right arm, namely *Humerus*, *Elbow* and *Wrist*; and the leaf nodes were each joints degree of freedom, containing the rotation values for each frame as a floating-point number.

**Code 3. 2. Code for the XML DOF rotations file. This file structure is used to store the rotation information for each arm's 7 animation curves. These animation curves are stored as a floating-point vector inside each attribute element, represented here for simplicity as […].**

```
<Daniele end="1623" start="0">
  <R_Humerus>
    <R_Humerus.rotateX> […] </R_Humerus.rotateX>
    <R_Humerus.rotateY> […] </R_Humerus.rotateY>
    <R_Humerus.rotateZ> […] </R_Humerus.rotateZ>
  </R_Humerus>
  <R_Elbow>
    <R_Elbow.rotateX> […] </R_Elbow.rotateX>
  </R_Elbow>
  <R_Wrist>
    <R_Wrist.rotateX> […] </R_Wrist.rotateX>
    <R_Wrist.rotateY> […] </R_Wrist.rotateY>
    <R_Wrist.rotateZ> […] </R_Wrist.rotateZ>
  </R_Wrist>
  <L_Humerus>
    <L_Humerus.rotateX> […] </L_Humerus.rotateX>
    <L_Humerus.rotateY> […] </L_Humerus.rotateY>
    <L_Humerus.rotateZ> […] </L_Humerus.rotateZ>
  </L_Humerus>
  <L_Elbow>
    <L_Elbow.rotateX> […] </L_Elbow.rotateX>
  </L_Elbow>
  <L_Wrist>
    <L_Wrist.rotateX> […] </L_Wrist.rotateX>
    <L_Wrist.rotateY> […] </L_Wrist.rotateY>
    <L_Wrist.rotateZ> […] </L_Wrist.rotateZ>
  </L_Wrist>
</Daniele>
```

The two XML files, sentence segmentation and DOF rotations, were used on the next steps to create parametric curves and parameters files for each transition as shown next.

## 3.4 CURVE FITTING

Transition curve information contained in the XML DOF files is a discrete sequence of floating-point values that, together with sentence segmentation files can provide a somewhat satisfactory transition as it was used in PULLEN et BREGLER (2002) and in LEE et al (2002), but it does have significant disadvantages compared to parametric curve descriptions:

- **larger files** since it needs to store each rotation value for each frame in the transitions, which are sometimes more than a hundred frames long;
- **interpolation problems** occurring when the desired curve length is not the same as the original, requiring discrete signal scaling interpolation similar to algorithms used in 2D image processing;
- **noise** in motion capture signal is very frequent and a parametric representation can work as a filter, producing smoother curves;
- **concatenation** on curve's boundaries, which would require some sort of curve fitting to create a smooth transition between signs and transitions;

Parametric curve representation was thus used instead, which was stored in the curves XML files in the curves database as presented later in this section in Code 3.3.A and Code 3.3.B.

A first attempt towards parametric representation was to use polynomial fitting; that is, adjusting the $n$ data points [$x_i$, $y_i$], with $x_i$ being the frame and $y_i$ the animation curve's rotation value, to a function of the type:

$$f(x) = \bar{y} = \sum_{j=0}^{k} a_j . x^j \tag{3.1}$$

where $k$ is the degree of the polynomial (0 is a constant value) and the $a_i$ terms are the series of coefficients that shape the curve.

A residue Q denotes the difference between the approximated function and the given

DOF rotation value *y<sub>i</sub>*. It can be calculated in various ways and the most common being the quadratic residue, denoted as:

$$Q = \sum_{i=1}^{n}(y_i - \sum_{j=0}^{k} a_j . x_i^j)^2 \qquad \textbf{\textit{(3.2)}}$$

A best fit will minimize this residue. Which means:

$$\frac{\partial Q}{\partial a_j} = 0; forj = 1,2,..,n$$

To solve Equation 3.2 for its minimum, it can be rewritten in matrix form (RUGGIERO et LOPES, 2010) as:

$$a = (X^T X)^{-1}.X^T.y \qquad \textbf{\textit{(3.3)}}$$

where

$$a=\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}, X=\begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^k \\ 1 & x_2 & x_2^2 & \dots & x_2^k \\ \dots & \dots & \dots & \ddots & \dots \\ 1 & x_n & x_n^2 & \dots & x_n^k \end{bmatrix}, y=\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

This is known as the least squares fit.

MATLAB was used to apply this method on the DOF curves, generating the curves shown in the images below.

**Figure 3. 4. Polynomial fitting of two different transition DOF curves.**

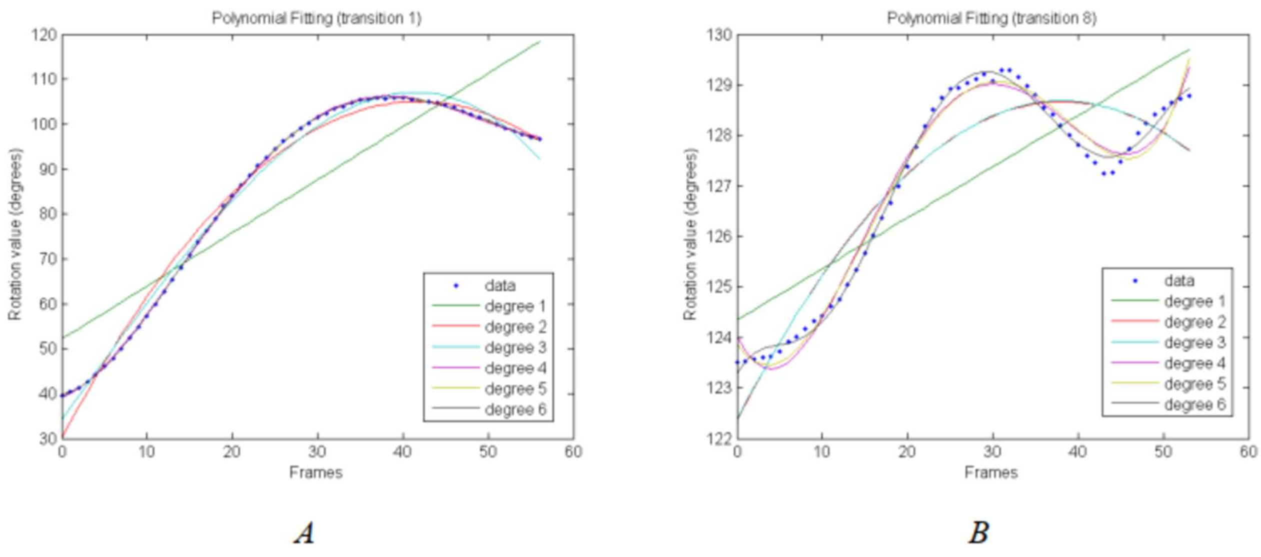


Image generated in MATLAB.

It is clear that by increasing the curve's degree we get a better approximation with the data (as can be seen with the decreasing RMSE values as we increase curve degree in Figure 3.5), this can also be seen in Figure 3.4. Although increasing curve degree does approximate better, it is not a good choice, and that can be seen from the example in Figure 3.4.B, where a curve with degree 6 is far from a good fit, presenting inflections that do not show the real behavior of the animation curve (its velocity and acceleration) specially visible in the vicinity of frames 3 and 50. Also, the boundaries of the function (frames 0 and 52) are still far off from the original curve both in value and in slope, which would create problems on the join points between the transition and the sign curves.

**Figure 3. 5. Root mean square error (RMSE) of polynomial fitting functions in MATLAB applied to 8 transitions of a DOF. RMSE in MATLAB is defined asRMSE=√($\sqrt{\frac{Q}{n-m}}$) where n are the number of samples (frames, in this context) and m is the number of fitted coefficients (curve degree, in this context). It is noteworthy to see that there is a great decrease in RMSE from degree 3 to 5, indicating that a complex curve is necessary on most cases.**

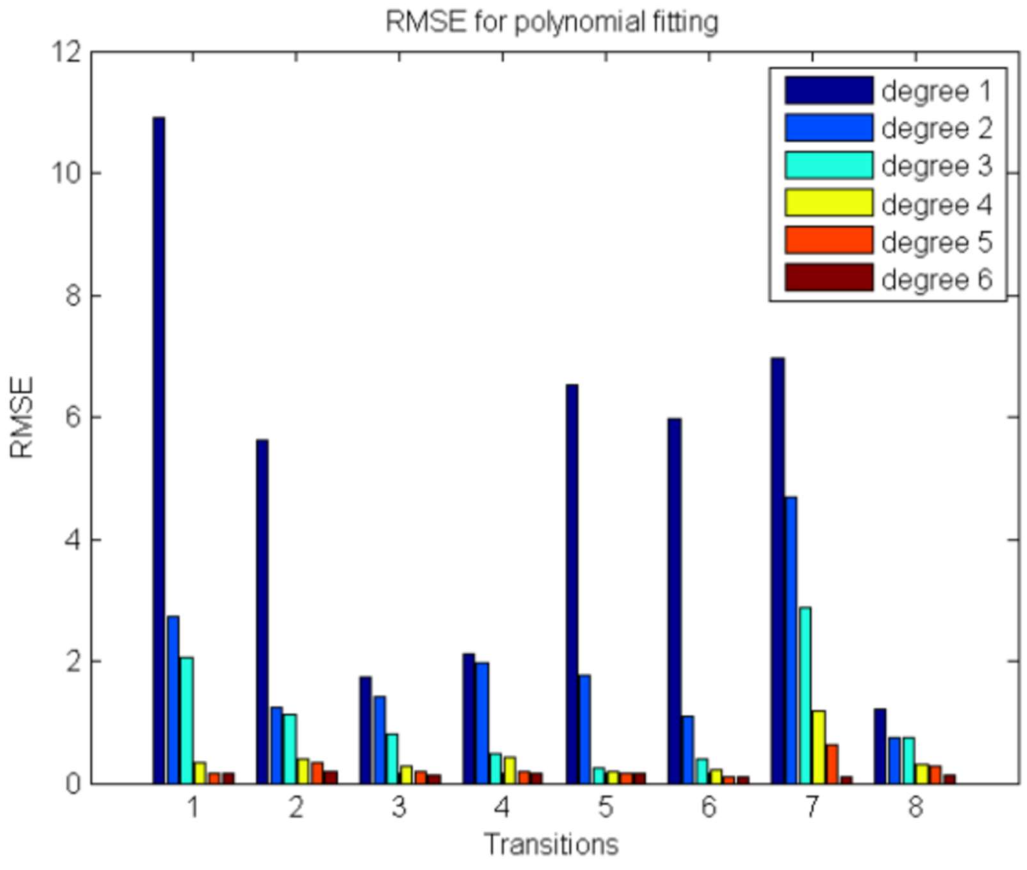


Image generated in MATLAB.

Since polynomial fitting did show a poor choice, two other curve types were used to fit the data into models: polynomial B-Spline and piece-wise cubic Hermite spline curve.

B-Spline fitting was done using the algorithm presented in DIERCKX (1993) which is similar to the polynomial fitting using least squares fit. Hermite fitting was done by an algorithm devised for this project. Since these curve fitting are complex, they are discussed in depth in Appendix A.

Both curve fitting algorithms produced similar curves, which were stored in Hermite and Spline XML files, shown below.

**Code 3. 3. XML file for transition animated curves using Hermite Curves. Numbers have been truncated to facilitate reading.**

```
<transition id="da01011" type="hermite">
  <info fps="120" time_units="frames"/>
  <description>
    <object name="R_Humerus">
      <attribute delta_time="56" delta_value="58.01" name="rotateX" value0="-1.78">
        <segment index="1" p0="[0,0]" p1="[1,1]" t0="[1.61,1.52]" t1="[2.72,4.92]"/>
      </attribute>
      <attribute delta_time="56" delta_value="29.63" name="rotateY" value0="-25.07">
        <segment index="1" p0="[0,1]" p1="[0.78,0]" t0="[0.59,-1.89]" t1="[0.039,0.01]"/>
        <segment index="2" p0="[0.78,0]" p1="[1,0.18]" t0="[0.011,0.01]" t1="[0.21,0.41]"/>
      </attribute>
      <attribute delta_time="56" delta_value="49.04" name="rotateZ" value0="38.13">
        <segment index="1" p0="[0,0.89]" p1="[0.29,0.99]" t0="[0.39,0.38]" t1="[0.15,-0.01]"/>
        <segment index="2" p0="[0.28,0.99]" p1="[1,0]" t0="[1.19,-0.02]" t1="[0.86,-1.08]"/>
      </attribute>
    </object>
    <object name="R_Elbow">
      <attribute delta_time="56" delta_value="67.75" name="rotateX" value0="32.63">
        <segment index="1" p0="[0,0.06]" p1="[0.16,0]" t0="[0.21,-0.18]" t1="[0.01,0.01]"/>
        <segment index="2" p0="[0.16,0]" p1="[1,0.99]" t0="[0.97,0.020]" t1="[1.44,-0.21]"/>
      </attribute>
    </object>
    <object name="R_Wrist">
      <attribute delta_time="56" delta_value="31.95" name="rotateX" value0="80.05">
        <segment index="1" p0="[0,0]" p1="[0.51,1]" t0="[0.026,0.056]" t1="[0.02,-0.01]"/>
        <segment index="2" p0="[0.51,1]" p1="[0.94,0.55]" t0="[0.02,-0.01]" t1="[0.28,-0.0]"/>
        <segment index="3" p0="[0.94,0.55]" p1="[1,0.56]" t0="[0.13,-0.01]" t1="[0.21,-0.1]"/>
      </attribute>
      <attribute delta_time="56" delta_value="9.72" name="rotateY" value0="36.83">
        <segment index="1" p0="[0,0.38]" p1="[0.1,0.01]" t0="[0.36,-0.80]" t1="[0.15,-0.06]"/>
        <segment index="2" p0="[0.1,0.01]" p1="[0.5,0.98]" t0="[0.44,-0.19]" t1="[0.5,0.08]"/>
        <segment index="3" p0="[0.5,0.98]" p1="[0.7,0.63]" t0="[0.32,0.05]" t1="[0.28,0.01]"/>
        <segment index="4" p0="[0.7,0.63]" p1="[1,0.36]" t0="[0.46,0.017]" t1="[0.55,-0.44]"/>
      </attribute>
      <attribute delta_time="56" delta_value="46.31" name="rotateZ" value0="-18.23">
        <segment index="1" p0="[0,0.8]" p1="[0.25,0.99]" t0="[0.21,0.5]" t1="[0.02,0.01]"/>
        <segment index="2" p0="[0.25,0.99]" p1="[1,0]" t0="[0.68,0.02]" t1="[2.05,-1.65]"/>
      </attribute>
    </object>
  </description>
</transition>
```

**Code 3. 4. XML file for transition animated curves using B-Spline curves. Numbers have been truncated to facilitate reading.**

```
<transition id="da01011" type="spline">
  <info fps="120" time_units="frame"/>
  <description>
    <object name="R_Humerus">
      <attribute delta_time="57" delta_value="58.0172" name="rotateX" value0="-1.7817">
        <g>6</g>
        <coef>
          [-1.7424, 0.43291, -0.20011, 6.6217, 1.986, 18.9936, 34.9427, 48.0801, 53.9813, 56.0629]
        </coef>
        <knots>[0, 0, 0, 0, 0.08318, 0.18719, 0.55111, 0.70267, 0.81888, 0.9149, 1, 1, 1, 1]</knots>
      </attribute>
      <attribute delta_time="57" delta_value="29.6394" name="rotateY" value0="4.5603">
        <g>7</g>
        <coef>
          [4.5235, 0.98408, -4.6605, -13.8159, -17.7027, -25.9618, -24.1599, -22.8483, -21.036, -20.5641, -19.5233]
        </coef>
        <knots>
          [0, 0, 0, 0, 0.12219, 0.27525, 0.52061, 0.80666, 0.85118, 0.91288, 0.97822, 1, 1, 1, 1]
        </knots>
      </attribute>
      <attribute delta_time="57" delta_value="49.0459" name="rotateZ" value0="81.6425">
        <g>6</g>
        <coef>
          [81.5332, 85.8763, 87.8579, 85.8724, 70.6242, 54.3305, 46.8823, 40.0148, 39.0731, 38.1365]
        </coef>
        <knots>
          [0, 0, 0, 0, 0.25281, 0.4778, 0.77339, 0.80172, 0.85982, 0.97273, 1, 1, 1, 1]
        </knots>
      </attribute>
    </object>
    <object name="R_Elbow">
      <attribute delta_time="57" delta_value="67.7594" name="rotateX" value0="37.0639">
        <g>6</g>
        <coef>
          [37.0859, 35.5741, 32.8384, 31.8923, 37.5239, 52.3981, 88.9373, 101.7003, 100.0563, 99.7765]
        </coef>
        <knots>
          [0, 0, 0, 0, 0.070083, 0.23474, 0.32213, 0.3576, 0.62896, 0.90912, 1, 1, 1, 1]
        </knots>
      </attribute>
    </object>
    <object name="R_Wrist">
      <attribute delta_time="57" delta_value="31.9551" name="rotateX" value0="80.0529">
        <g>7</g>
        <coef>
          [80.0159, 83.6454, 80.3447, 99.0212, 108.132, 112.7727, 107.4681, 104.0151, 98.6388, 98.2509, 98.5793]
        </coef>
        <knots>
          [0, 0, 0, 0, 0.12002, 0.36672, 0.4488, 0.52244, 0.53399, 0.76284, 0.89119, 1, 1, 1, 1]
        </knots>
      </attribute>
      <attribute delta_time="57" delta_value="9.7216" name="rotateY" value0="40.613">
        <g>7</g>
        <coef>
          [40.6638, 40.2077, 35.2122, 38.7542, 39.9383, 47.5721, 47.6438, 42.9002, 43.9283, 40.8801, 40.3503]
        </coef>
        <knots>
         [0, 0, 0, 0, 0.13708, 0.24378, 0.34221, 0.34353, 0.62954, 0.69192, 0.69976, 1, 1, 1, 1]
        </knots>
      </attribute>
      <attribute delta_time="57" delta_value="46.3108" name="rotateZ" value0="18.9881">
        <g>6</g>
        <coef>
          [19.2777, 24.8688, 26.1465, 29.6811, 18.2527, 13.2059, -6.1011, -12.1528, -17.7031, -17.7755]
        </coef>
        <knots>
          [0, 0, 0, 0, 0.17639, 0.34221, 0.39279, 0.50308, 0.60209, 0.81994, 1, 1, 1, 1]
        </knots>
      </attribute>
    </object>
  </description>
</transition>
```

As stated before, the XML transition curves files were devised prioritizing human readability, therefore the elements and their attributes are straightforward with their nomenclature and use:

- **transition** is the root element and contains the primary key *id* for each transition and the curve *type* indicating whether the curves described within the file are B-Spline or Hermite curves;
- **info** element presents basic information about the curves units in time (frames, seconds, milliseconds...) and value (degrees, radians, meters...), which could be used if the database was populated with animation curves created with different units. Since we controlled all data stored we made sure that they all used the same units (frames and degrees) so this element was not used in the current implementation;
- **description** contains all object's attributes animation curves information in its children elements, with their direct elements being each joint (*R_Humerus*, *R_Elbow*, *R_Wrist*...) and, in turn, their own direct elements being each DOF / attribute (rotate x, y, and z for humerus and wrist joints and rotate x for elbow);
- **attribute** elements, as stated above, hold each joint's individual animation curve (1 for each DOF), as well as information on the original curves values, such as initial value (*value0*) and the difference between the maximum and minimum values in the range (*delta_value*);
- **segment (Hermite curves)** elements hold each Hermite curve segment's information about its points (*p0*, *p1*) and its tangents (*t0*, *t1*) in a 2-dimensional point. These are the values presented in Hermite formulation in Equation A.2;
- **g (B-Spline curves)** indicates each curve's number of sub-intervals as shown in Section A.2;
- **coef** and **knots (B-Spline curves)** are the coefficient and knots vectors as defined in Section A.2;

These XML files were stored in the Curves Database to be retrieved when needed.

The next step to the method was to find the best match in the curves database for the segment being concatenated. For that, it was used the propositions presented in Chapter 2. What was needed next was a way to describe the curves in a manner as to allow them to be compared and thus used to concatenate similar movements.

## 3.5 PARAMETERS EXTRACTION

Following the propositions raised by previous research in bio-mechanics of arm movement (Section 2.2), to correctly describe a curve and be able to compare it with other arm movements we need the 7 DOFs of the shoulder, elbow, and wrist joints. We also need to take into account not only the final DOFs values but also their initial values as well as velocity and acceleration (initial configuration interference).

A first approach was to calculate the rotation, velocity, and acceleration of each 7 DOFs from the start and end position, arriving at a total of 42 parameters for each arm's transition. Since arm movement is represented as a discrete sequence of rotation values, it is possible to obtain the parameters easily as a means of values, for rotations, or means of difference of values, for velocity and acceleration for the required amount of frames (window of reference) from before and after the transition. This approach though is very prone to noise and, as the window of reference grows, the values become less descriptive of the curves as they become more of an average of rotation, velocity and acceleration that in themselves can't describe the shapes of curves.

The approach used was to consider the segments of the rotational curve as a sign and operate on frequency. To operate in such domain a transformation is in order, and it was chosen the Discrete Cosine Transformation, DCT, over others (such as Discrete Fourier Transformation, DFT, and Fast Fourier Transformation, FFT, which are the other most common transformations used in electronic signal processing) for its simplicity, and thus faster computation than Fourier Transformation.

Discrete Cosine Transformation used is the DCT-II (SALOMON et MOTTA, 2010):

$$X_k = \sum_{n=0}^{N-1} x_n . \cos\left[\frac{\pi}{N} . \left(n + \frac{1}{2}\right) . k\right] \qquad k = 0, \dots, N-1. \qquad \textbf{\textit{(3.4)}}$$

where $x_n$ is the n-th rotation value in the segment of size *N*.

To comply with results from MATLAB processing, the term $x_n$ was multiplied by 1/√(N-1) if *n* = 0 and by √2/√(N-1) if *n* ≠ 0, resulting in:

$$X_k = \sum_{n=0}^{N-1} . \frac{x_n}{\sqrt{N-1}} . \cos\left[\frac{\pi}{N} . \left(n + \frac{1}{2}\right) . k\right] \quad k = 0, \dots, N-1; \quad w = \frac{1 \; if \; n-0}{\sqrt{2} \; if \; n \neq 0} \qquad \textbf{\textit{(3.5)}}$$

The choice of *N* was based on the following criteria: *N* could not be too small (lower boundary), that it should be close to the amount of $X_k$ used as parameters to compare curves, avoiding errors produced by noise and it could not be too big (upper boundary) as to describe a sign curve and not just the immediate end and beginning.

The lower boundary for N can be determined by the amount of coefficients *N* intended to be used as parameters. Since we want to determine position, velocity, and acceleration we need at least 3 components of the original mocap signal, since with only 1 sample we can determine position, with 2 samples we can determine velocity dividing the position by time and with 3 samples we can again divide both velocities to get acceleration. This will indicate:

$N \geq 3$

For the higher boundary it was necessary to look at the duration of the signs. This is very particular for each interpreter, taking each more or less time to sign, so it was chosen the lowest sign duration average (52,59 frames at 120 fps, which amounts to about 0,43 seconds). Considering that this range has, at least, the end of the previous transition and beginning of the next we can divide it by 2 and have an estimation of:

$N \leq 26$

Since *N* has to be an integer, it was rounded down.

It is noteworthy to point out that to devise the lowest sign duration average mentioned previously we took into account some signs that were very fast, last than 10 frames. This is further discussed in Section 5.2.4.

Having defined a range of possible values for *N* tests were done to determine a good number, which was defined as 10.

The last variable needed was *k*, which is the number of DCT coefficients to be used as the parameters to describe the transition's beginning and end. $X_0$ is needed for it contains the average of *N* rotation values before and after the transition, higher frequencies are not interesting since they store mostly noise information from mocap. A look at the DCT basis function (Figure 3.6) reveals an interesting property: $X_1$ will store information on how, on average, the rotation value changes over time, and $X_2$ will store information on whether there is a change within the range but not on the boundary (beginning and end). The highest frequencies, especially for a lower *N* as has been used in the algorithm, will not yield significant data to compare curves. Thus, *k* was chosen as 2, that is 3 components of the DCT on the segments.

**Figure 3. 6. DCT basis functions.**

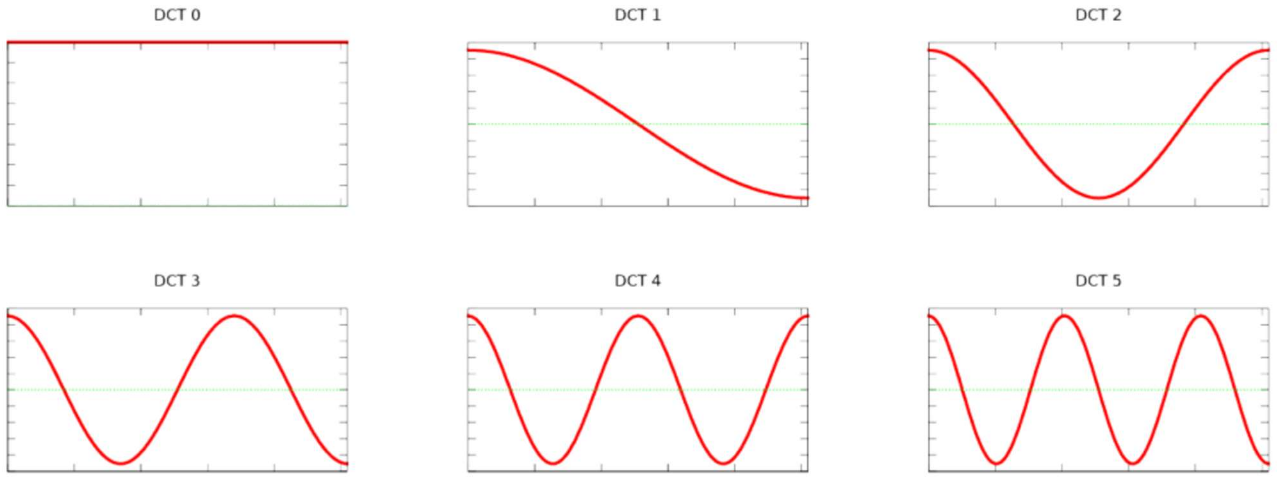


Image adapted from HERRERA et RUIZ (2014).

Transitions are thus described as a vector of 42 parameters: for each of the 7 DOFs (X, Y and Z shoulder rotation, X elbow rotation, and X, Y and Z wrist rotations) there are 6 DCT bases ($X_0$, $X_1$ and $X_2$ for the 10 frames prior to the transition an $X_0$, $X_1$ and $X_2$ for the 10 frames after the transition).

For each transition, an XML file is created containing the curve parameters (we used the word descriptors in the XML file so as to not create confusion with a curve descriptor such as knot counts, degrees, etc.). These files are stored in the Parameters Database and used in the Transitions Search.

**Code 3. 5. Example of an XML transition parameter file. Values have been truncated.**

```
<TransitionDescriptor fps="120" id="da01011" rotations="degrees" time="frames">
  <R_Humerus.rotateX in_dct00="-9.82" in_dct01="-0.79" in_dct02="0.33" out_dct00="99.52" out_dct01="-1.71" out_dct02="-
0.032"/>
  <R_Humerus.rotateY in_dct00="28.95" in_dct01="1.67" in_dct02="-0.034" out_dct00="-32.79" out_dct01="-0.86"
out_dct02="0.047"/>
  <R_Humerus.rotateZ in_dct00="125.96" in_dct01="-1.70" in_dct02="0.03" out_dct00="65.35" out_dct01="0.53"
out_dct02="0.054"/>
  <R_Elbow.rotateX in_dct00="82.30" in_dct01="1.47" in_dct02="-0.03" out_dct00="172.15" out_dct01="0.53"
out_dct02="0.027"/>
  <R_Wrist.rotateX in_dct00="133.79" in_dct01="0.35" in_dct02="0.46" out_dct00="170.99" out_dct01="-0.53" out_dct02="-
0.41"/>
  <R_Wrist.rotateY in_dct00="73.75" in_dct01="0.44" in_dct02="0.18" out_dct00="69.37" out_dct01="0.37"
out_dct02="0.14"/>
  <R_Wrist.rotateZ in_dct00="22.96" in_dct01="-0.07" in_dct02="0.72" out_dct00="-31.01" out_dct01="-0.20" out_dct02="-
0.45"/>
</TransitionDescriptor>
```

When the user requires a new transition between two signs, that are given as a set of discrete DOF animation curves, parameters are extracted using the DCT as shown in the previous section and compared to those in the parameters database.

The comparison is made by a weighted Euclidian distance, *d*, as shown in the Equation 3.6 below.

$$d = \sum_{i=0}^{41} w_i . \sqrt{x_i^2 - y_i^2} \qquad \textbf{\textit{(3.6)}}$$

$X_i$ is the vector containing the parameters for a curve in the parameters database; $Y_i$ is the vector containing the parameters from the desired gap (that is, the DCT from the first sign's *N* last rotation values and the second sign's *N* first rotation values); and $w_i$ is a vector containing the weight of each parameter.

$w_i$ is an important element for the distance measurement since it normalizes the values from the DCT elements. This is needed because the DCT has a much higher value on the 0 component (direct signal) than on the other 1 and 2 components (alternating signals), as can be seen from the example in Code 3.4.

For each transition in the parameters database, we get a value of *d*, the smallest one is then chosen, indicating the closest transition to the gap desired. The final step thus is taken, that is, to concatenate the two signs with the transition stored in the transitions database.

## 3.6 CURVE PROCESSING

Curve processing is divided into two stages: curve adjustment, which is adjusting both the transition curves' values (rotation degree) and time (transition duration); and join point adjustment, that is to smooth out the values on the transition boundaries (join points) with the signs. Since signs are very specific and should not be tampered with as to avoid a misinterpretation, the method does not modify the signs' curves like the concatenation presented in PULLEN and BREGLER (2002).

Since curve processing is significantly different from B-Spline and Hermite, they were treated and will be shown next separately.

### 3.6.1 Transition Curves Adjustment

Curve adjustment is the process of fitting the first transition animation curve value ($y_0$) to the last value from the first sign animation curve ($v_0$) and the last transition animation curve value ($y_N$) to the first value from the second sign animation curve ($v_1$). Also, if determined by the user a specific time value for the transition, it needs to scale in time accordingly.

**Hermite**

The transformation for the Hermite spline can be decomposed on its x (time) and y (rotation value) components as a scale in x and a skewing in y as so:

$$x' = x.t \quad \textbf{\textit{(3.7)}}$$

$$y' = (y - y_0).\varDelta Y + (\varDelta v - (y_N - y_0).\varDelta Y).x + v_0 \quad \textbf{\textit{(3.8)}}$$

Where:

- *x'* is the new time value;
- *x* is the old time value (normalized in the range 0 to 1);
- *t* is the total amount of time for the transition, given by the user;
- *y'* is the new rotation value;
- *y* is the old rotation value (normalized in the range 0 to 1 from minimum to maximum);
- $\varDelta y = y_N - y_0$;
- $\varDelta v = v_1 - v_0$;
- $\varDelta Y$is the total amount of rotation in the animation curve (stored in the XML file as the attribute *delta_value* as can be seen in Code 3.3.B), multiplying any value in the transition curves ($y_i$) by it will yield the original transition rotation value, since the values stored are normalized;

For each tangent defined as the vector *[$t_x$,$t_y$]* the transformation is given as:

$$t_x' = t_x.t \quad \textbf{\textit{(3.9)}}$$

$$t_y' = t_y.\varDelta Y + (\varDelta v - (y_n - y_0).\varDelta Y).x \quad \textbf{\textit{(3.10)}}$$

**B-Spline**

The same idea of skewing over scaling is applied on the B-Spline curve. An inconvenient from the B-Spline used compared to the Hermite curve is that to apply the skew we need the *x* (time) component of each given spline component as can be seen on the Equation 3.8, and for B-Splines the information given is only a vector of knots (*t*)*,* a vector of coefficients (*c*) and the curve order (*k*).

This inconvenience can be addressed if we look on the formulation of the B-Spline (DIERCKX, 1993):

$s(x) = \sum_{i=1}^{n} c_i . N_{i,k}(x)$

For each basis spline *N* we assign a multiplying coefficient *c* that can be interpreted as the weight for that basis on the overall curve, similar to the control points when the B-Spline represents a curve of dimension 2 or greater. In Figure 3.7 it is possible to see the basis splines, and the peaks of each are the points in x that the curve exert greatest influence, notably on the edges we have $N_{1,3}$ and $N_{4,3}$ with weight of 1 indicating that the B-Spline curve will pass through these initial and end points, since the knot vector repeats *k* times the first and last values generating a closed curve.

**Figure 3. 7. Basis splines for a B-Spline curve with a knot vector t = [0 0 0 0.5 1 1 1].**

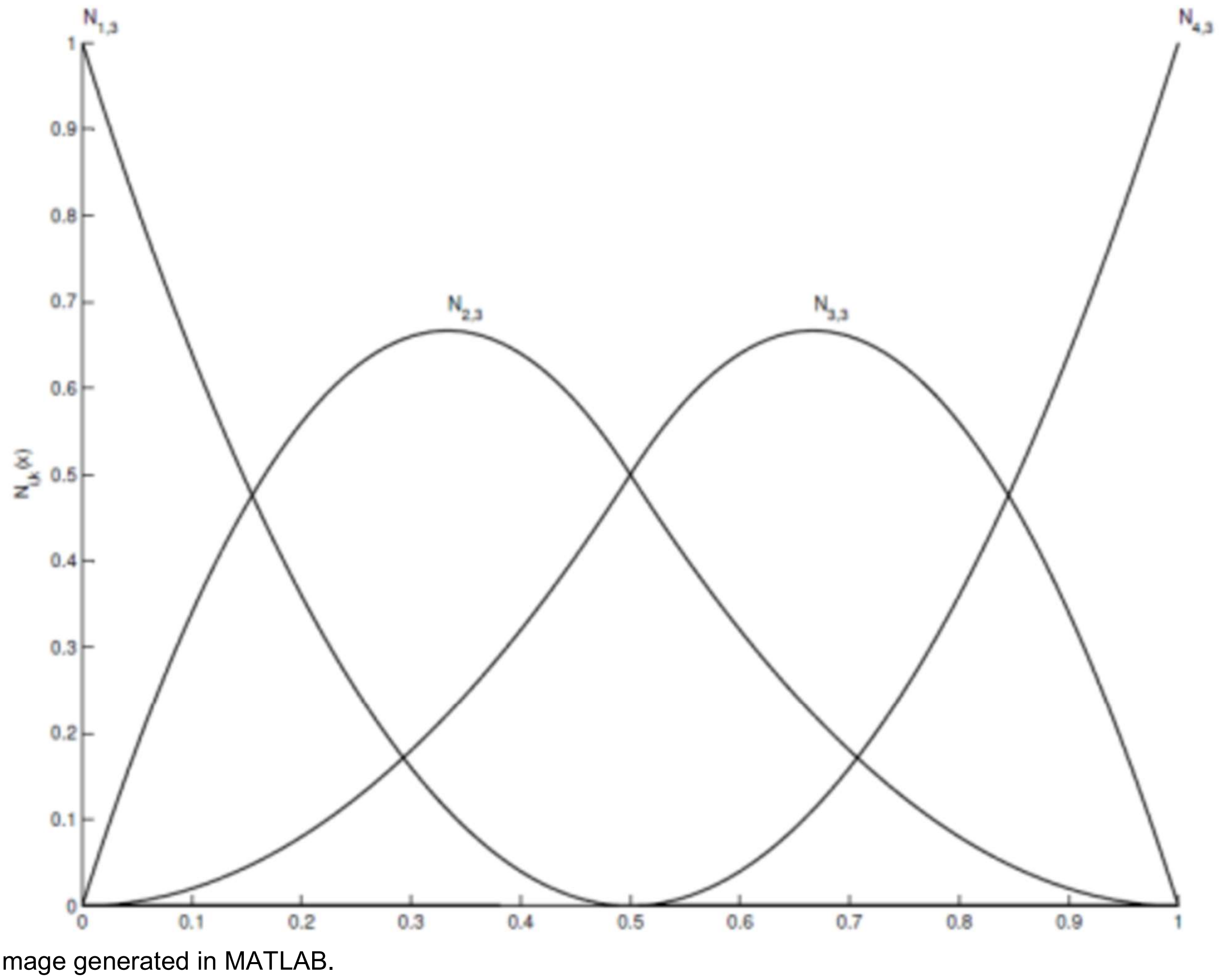


Image generated in MATLAB.

For periodic knot vectors it would be easy to find a general formula for the basis function peaks. But since the algorithm used in spline fitting (DIERCKX, 1993) changes the knot vectors, the peaks must be found by another way. Each basis function peak can be obtained by finding its derivative's zero function, which is given as:

$$\frac{dN_{i,k}(x)}{dx} = N_{i,k}'(x) = \frac{k}{t_{i+k}-t_i}.N_{i,k-1}(x) - \frac{k}{t_{i+k+1}-t_{i+1}}.N_{i+1,k-1}(x) \tag{3.11}$$

Thus, for each coefficient $c_i$ it is possible to get its relative x coordinate when $N_{i,k}' = 0$.

A final remark is that $N_{i,k}'$ can be zero not only when max($N_{i,k}$), but also when $N_{i,k} = 0$, that is outside the basis function influence under the vector x, so the range for search of $N_{i,k}' = 0$ has to be contained where $N_{i,k} > 0$.

To find the root of $N_{i,k}'$ within the range established, it was used the bisection method as proposed by RUGGIERO et LOPES, 2010.

Therefore, to join the B-Spline transition curves with the signs' curves it was used a function analog to the Hermite skewing presented in Equation 3.8 applied to the coefficient vector, which is:

$$c_i' = (c_i - c_0) + (\Delta v - (c_N - c_0)).x_i + v_0 \tag{3.12}$$

$x_i$ is the normalized x (that is, in the range 0 to 1) found in the maximum value of the corresponding i-th basis function for $c_i$.

### 3.6.2 Join Point Adjustment

Join point is the boundary frames between the curves (first sign to transition and transition to second sign). To smooth out these values and maintain the rotation values from the sign, only operations on transition curves are acceptable. Again, for each curve, the method varies, but it is the same idea of maintaining tangent continuity at the boundaries.

Since noise is present in the motion capture signs, instead of using the direct tangent which would be given by the difference between 2 consecutive frames before and after a transition that has a length of *N* frames, a small range was established, and the tangent was calculated as a means from the *W* closest frames, as such:

$$\tan_{in} = \sum_{i=-W}^{0} \tan_i - \tan_{i+1} \tag{3.13}$$

$$\tan_{out} = \sum_{i=N}^{N+W} \tan_i - \tan_{i+1} \tag{3.14}$$

Where:

- *N* is the transition size (time, in frames);
- *W* is the window, number of frames to be considered;
- *tan* are the tangent values;

**Hermite**

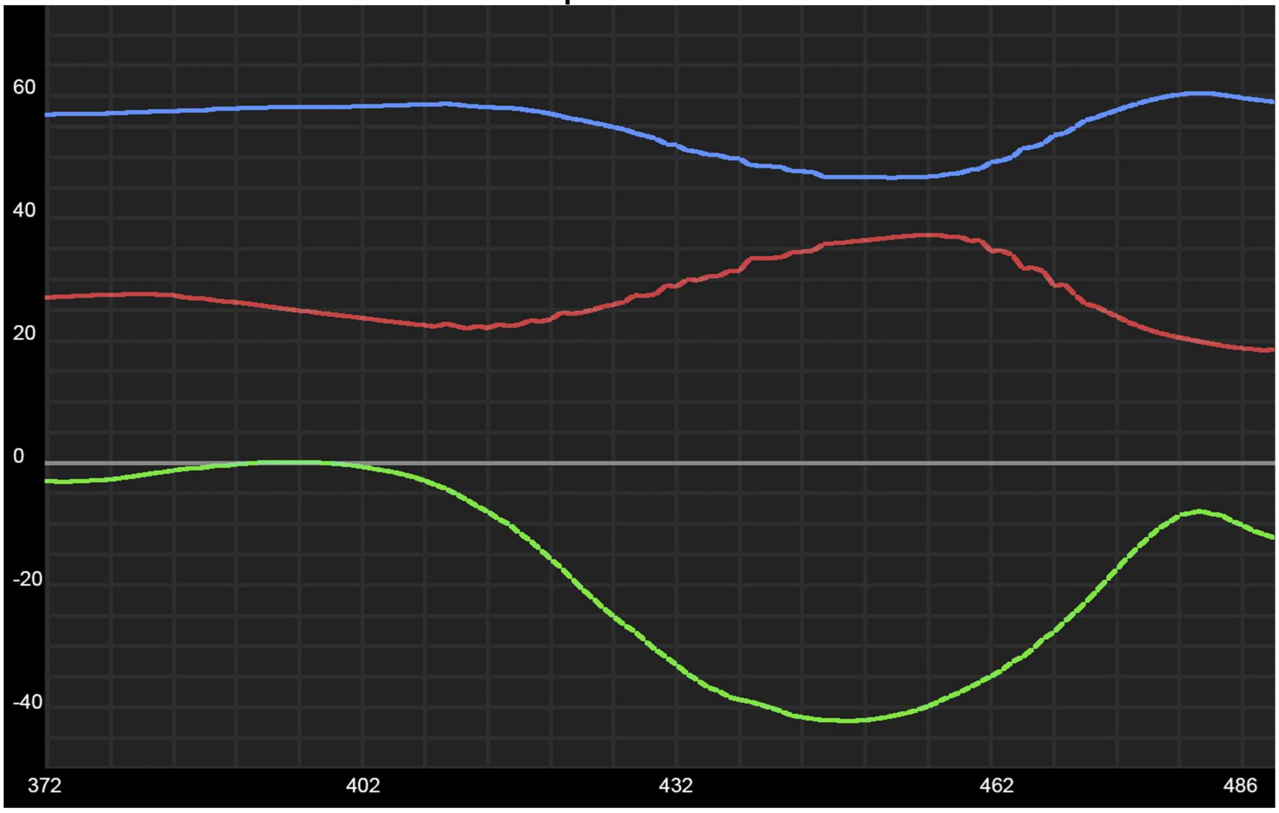
Hermite adjustment is done by modifying the first tangent of the first segment and applying the same inclination as to *tan*$_{in}$, and applying the same inclination as *tan*$_{out}$ to the second tangent of the last segment.

Therefore, it is easier to manipulate tangents in a polar coordinate system. The vectors are represented by an angle *θ* and a module *r* instead of *x* and *y* components, and the transformation from the Cartesian to polar coordinate system is given by:

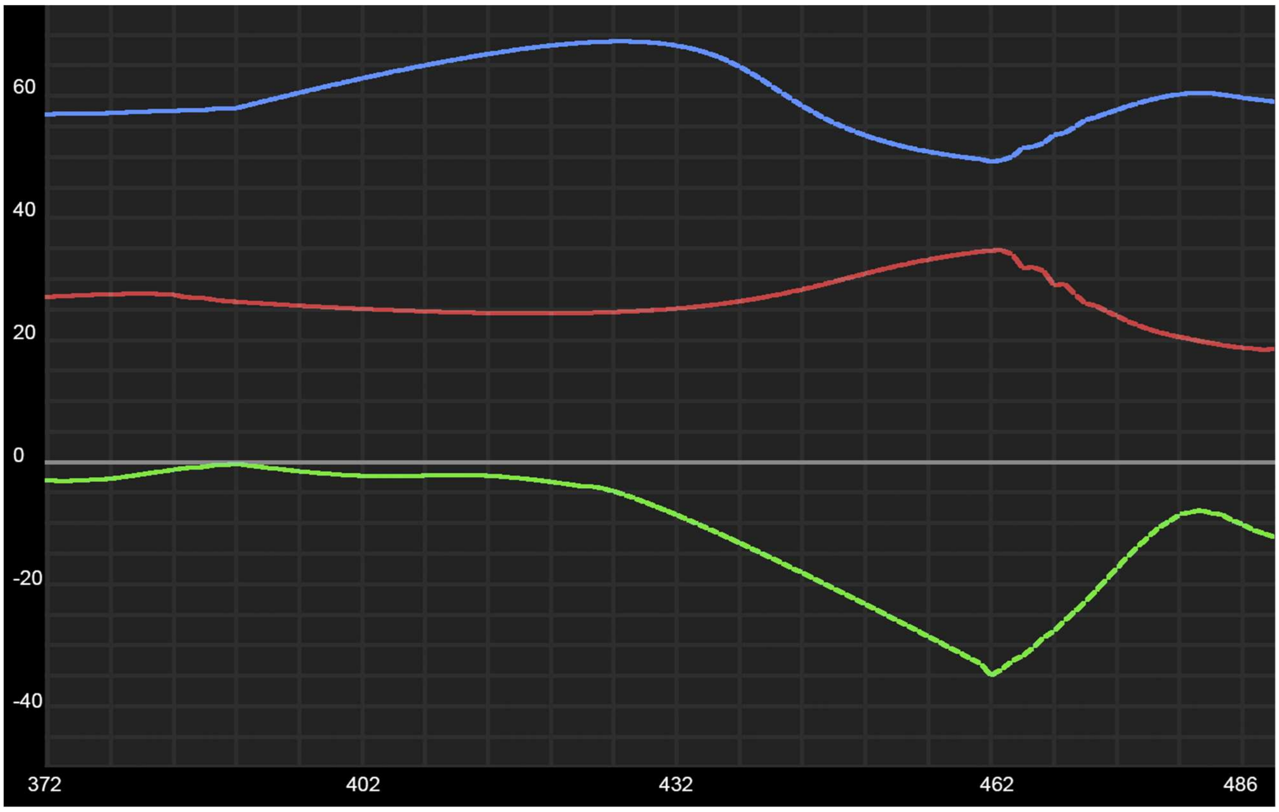
$$\begin{gathered} r=\sqrt{x^2+y^2} \\ \theta=tan^{-1}\left(\frac{y}{x}\right) \end{gathered} \qquad \textit{(3.15)}$$

A first approach would simply modify the values of *θ* on the transition curves to match the sign's initial and final tangent. This can cause steep changes in rotation values which are not natural (Figure 3.8). To address this issue, it is necessary to also change the transition tangents' modules based on the difference of angle with the signs' tangents.

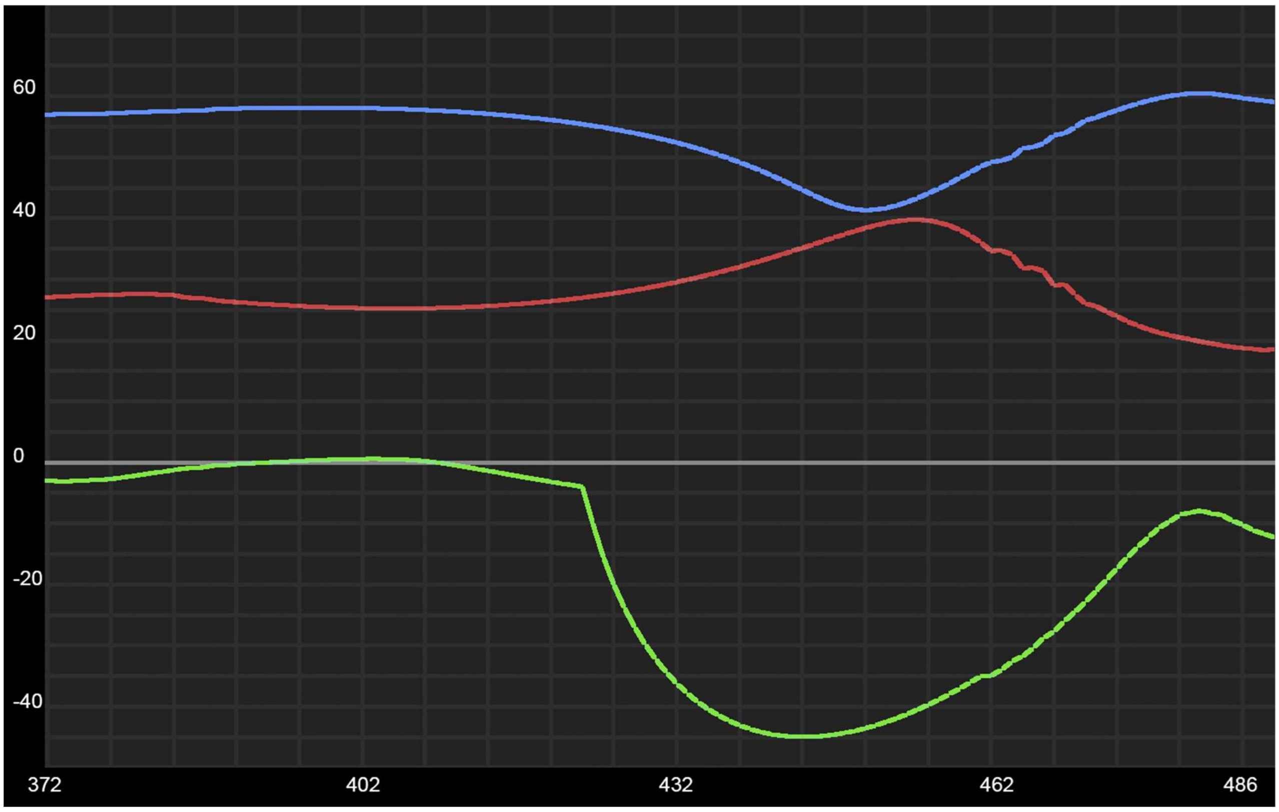
**Figure 3. 8. Hermite tangents adjustments. Original animation curve from motion capture data (A). Hermite transition curves have been applied to the range [390, 463] without any join point adjustment (B), resulting in a discontinuing curve especially visible on the rotation Y (green curve) on the frame 463. Join point adjustment is later applied, first with only the tangent angle (C) and later with the diminishing tangent module (D) given by Equation 3.16.**



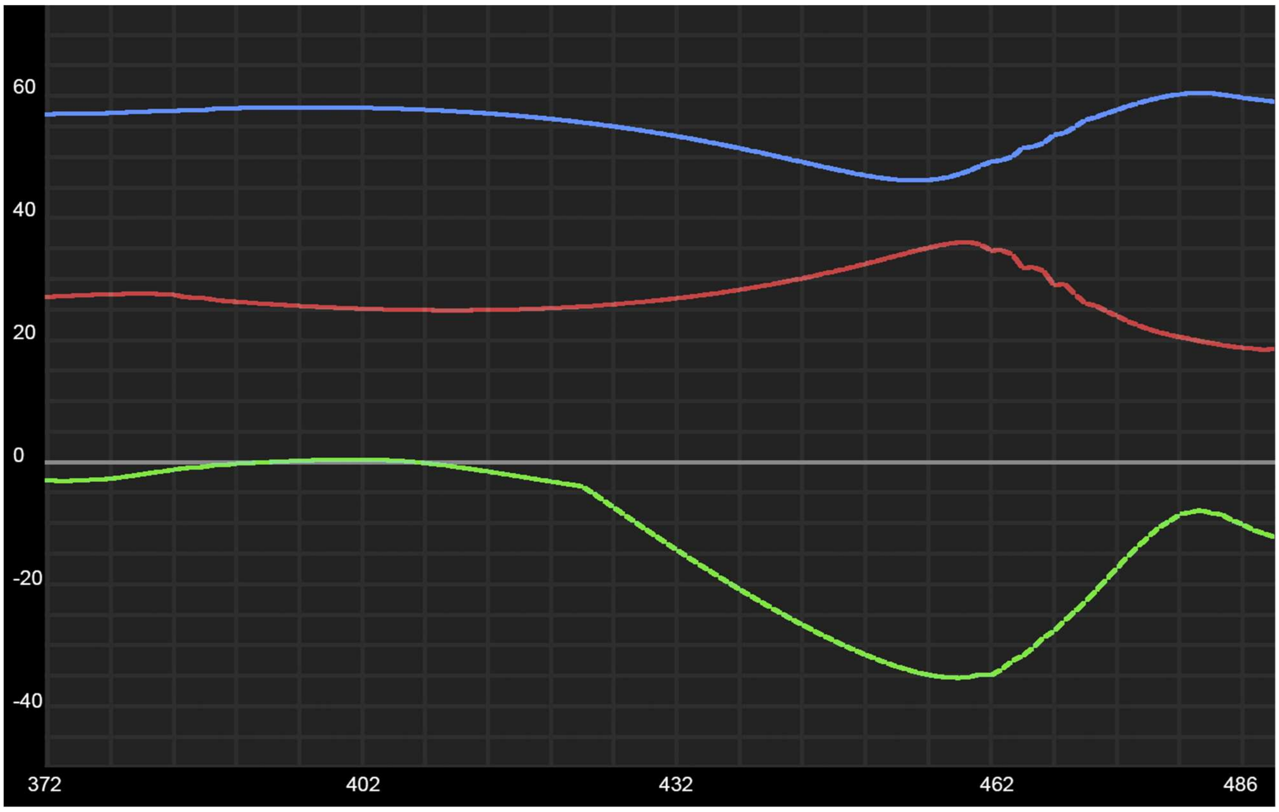
A


60
40
20
0
-20
-40
372
402
432
462
486

B


60
40
20
0
-20
-40
372
402
432
462
486

C



D

Images generated in Autodesk Maya.

The formula used was hence:

$$r' = r.\frac{(\pi - \Delta\theta)}{\pi}\theta' = \theta^{-} \qquad \textbf{\textit{(3.16)}}$$

Where:

- *r'* is the new transition tangent module;
- *r* is the tangent module from the transition curve's tangents;
- *Δθ* is the difference between the sign and the transition curve angle, that is *θ* - *θ*;
- *θ'* is the new transition tangent angle;
- *θ* is the angle from the sign animation curve;

**B-Spline**

For the B-Spline curves, the adjustment made is on the coefficients *c*, namely on the second ($c_2$) and on the second to last coefficients ($c_{N-1}$).

The tangents $tan_{in}$ and $tan_{out}$ provide the rate by which the rotation values (y) change over time (x), and this can be used to align $c_2$ and $c_{N-1}$ with the signs' tangents, therefore maintaining continuity on the join points. To estimate the value of x for the coefficients, the same method of derivative of basis functions from the previous section was used.

$$c_2{}' = c_1{}' + \theta \quad . t . x_2 c_{N-1}{}' = c_N{}' - \theta_{out} . t . x_{N-1} \qquad \textbf{\textit{(3.17)}}$$

## 3.7 FINAL REMARKS

The method proposed can be seen as a mixture between data-driven and procedural animation. While it does use data from motion capture to create transitions between signs, it does so by selecting the best transition in the set according to the propositions discussed in Chapter 2.

Transitions used, after captured from interpreters and segmented, were stored as Hermite or B-Spline curves in order to better manipulate them. This also became an interesting tool to compare the performance of different curve types, which will be shown in the results in the next chapter.

The selection of transition to be used between signs was done by comparing the DCT of the frames before and after the transition, that is the last frames for the first sign and the first frames for the second sign, resulting in a selection that not only takes into consideration the arms initial and final posture (joint rotations) but also the movement before and after.

Joining the animation curves from transitions and signs was a challenge on its own when dealing with the B-Spline curve. Since the algorithm proposed to fit the curve treated it as 1-dimensional the skewing applied along the curve had to use the derivative of basis functions to find the appropriate amount of distortion to apply to coefficients proportionally to their "x" value.

The last step was then to compare the results of the method proposed, which can be seen in the next chapter.

# 4 RESULTS AND DISCUSSIONS

In the previous chapters we have shown how this project started with research into digital animation and human arm movement models, trying to find a procedural animation (PARENT, 2012) model that would enable us to synthesize arm movement between signs in Libras avatars. Several works on Bio-Mechanics of Arm Movement such as MAROTTA, MENDENDORP et CRAWFORD (2003), and ADMIRAAL et al (2004) showed us that, although a universal model doesn't seem to exist which could support such procedural animation approach, some underlying rules helped us develop the method, providing us the 3 propositions adopted. We've then presented the model we used for transitioning between signs, which consists of a data-driven animation system, using previous mocap interpreters inter-sign transitions to synthesize new movement. A reminder of the different steps used in the method can be seen in the figure below which is a reproduction of Figure 4.1.

**Figure 4. 1. Method overview.**

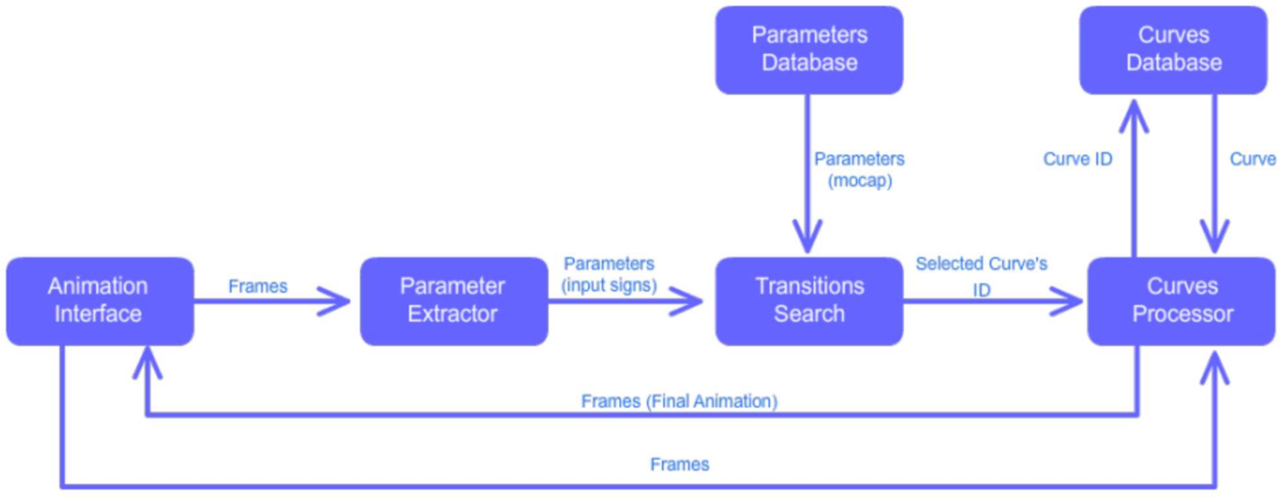


Image from the author

The method proposed was implemented as a series of MATLAB and Python / Autodesk Maya programs with results that are shown here.

This chapter will be divided into 2 sections: 4.1. Method Application, where it is shown and discussed how the proposed method was used to synthesize new transition animation; and 4.2. Results, where the results of these synthesized animation curves are shown.

## 4.1 METHOD APPLICATION

This section presents the steps done to synthesize inter-sign transition movement. As the details and discussions about implementations have been addressed in previous chapters, here we will focus on how we applied the method to real data, pointing out issues when they arose. A more detailed discussion into these issues and how we envision they can be dealt with in the future is presented on Section 5.2.

Using Autodesk Maya and its Python API we loaded up each individual mocap sentence. These, of course, had the inter-sign transitions animations, so the first step to run our method was to delete the animation curves' keyframes on the intervals defined in the XML sentence segmentation files, leaving us with a signing virtual avatar with signs in place but with arms jumping from one sign to the other.

To find the closest match, the XML database containing the transition parameters is loaded and, for each transition in the sentence, we exclude it from the search to force the method to find a different transition than the current one (since the closest match to any transition is the transition itself).

Having found the closest match to the transition to be interpolated, we used both Hermite and B-Spline curve models stored in the XML curves database to transition from one sign to the next.

B-Spline curve model, being based on the work of DIERCKX (1993) didn't pose many problems besides the interpretation of the algorithm and implementation under the tools used in this project as shown in Appendix A.2.

Hermite curve-fitting as shown in Appendix A.1 was a more challenging task, but it provided us with an algorithm that was both very solid (there were no curves that had a too high RMSE value as compared to the B-Spline fitting that had a very high RMSE in some animation curves as the case in the Right Wrist’s rotation X in interpreter KA’s first sentence transition 9 shown in Appendix B) and fast (DIERCKX algorithm would take minutes to process the transitions in a sentence while Hermite curve fitting could process them in a matter of seconds in the same machine, both inside MATLAB). There are some things we would like to improve in the algorithm, for sure, and we show here how we can do that.

Determination of *Δy* to select the best splicing points from Equation A.5 could be improved. The equation is copied below:

$$\Delta y = \frac{[max(x)-min(x)]}{100} \qquad \textbf{\textit{(4.2)}}$$

This produces good, but in some animation curves that had too much noise and a small amplitude it did produce a very noisy Hermite curve as can be seen in Figure 4.1. Another solution could be to use a defined number for all curves based on the context of human arm joints in sign language animation, that is with a large set of mocap data we could retrieve what are the minimal rotation values that usually occur on a cleaned animation curve and use that to create a new Δy to filter out noise values.

**Figure 4. 2. Example of a bad splicing point selection due to high frequency and low animation curve amplitude. This case suggests a different method to determine Δy.**

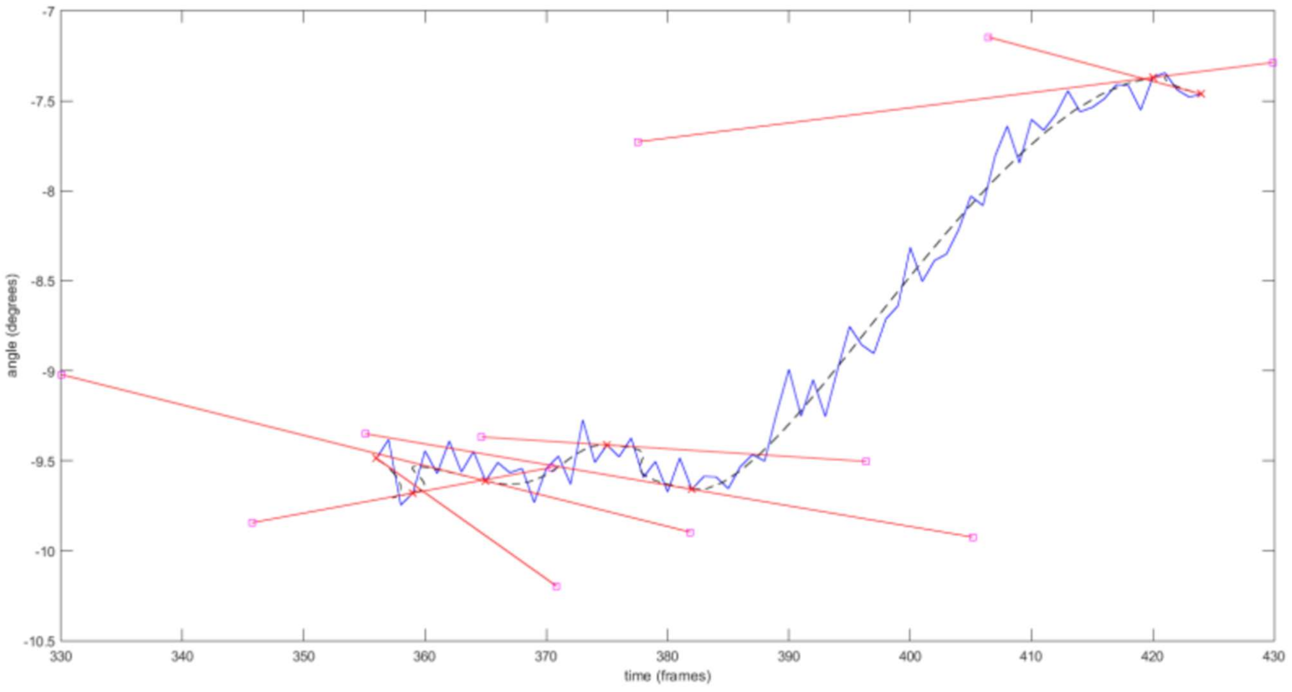


Image generated in MATLAB.

Another point that we found some difficulties was the tangent determination process. We determined tangent inclination by an average of close points' inclination (in lieu of just the curve derivative between the splicing point and its direct neighbors) due to the high noise encountered in the mocap data. A first approach though was with only local inclination (direct neighbors), that is, for a splicing point at frame *i* between Hermite curves *k* and *k+1*, was tried using the following formula:

$$t_1^k = y_i - y_{i-1}$$
$$t_0^{k+1} = y_{i+1} - y_i \qquad \textbf{\textit{(4.2)}}$$

This yielded poor tangent inclinations due to noise in mocap data, thus it was used a range of points inclination as shown in Equation A.6. For the intermediate tangents, those that are in the splicing points, these produced good results, but the same cannot be said for the first tangent in the first curve, $t_0^0$, and the second tangent in the last curve, $t_1^K$ (*K* is the last Hermite curve segment in the animation curve).

An approach that could be attempted is to have the next step of the algorithm, tangent weight determination, also look for a better inclination whereas now the algorithm created only searches for a better tangent weight; this means creating independent $k_0$ and $k_1$ factors for the x and y coordinates of the tangents.

Having the curves adjusted into Hermite or B-Spline models, the next step was to create an algorithm that would join the transition's animation curves with the signs' animation curves.

Our first approach to match transition animation curves to those of the signs to be concatenated was to scale the animation curves so as the first value matches the first sign's last value and the last value matches the second sign's first value, as in the equation below:

$$y' = \frac{y - y_0}{\Delta y} . \Delta v + v_0 \qquad \textbf{\textit{(4.3)}}$$

This approach though created big distortions in the transition curves as can be seen in Figure 4.2. The problems arise when $\Delta y$ (difference between initial and final transition values) is very small, causing the transition animation curve's values ($y'_0, y'_1, \ldots , y'_n$) to be too high.

**Figure 4. 3. Difference between skewing and scaling in three animation curves for x, y and z rotations of a joint. (A) shows the original motion capture animation curves. (B) we deleted the curves values on intervals [1044,1092] and [1151,121] and applied the method with Hermite curves using skewing to match initial and final transition values. If instead it was used scale, the resulting curve can be seen on (C), and with the y (rotation values) axis scaled up (D) it is possible to see a valley around -9000 degrees. Images generated in Autodesk Maya.**

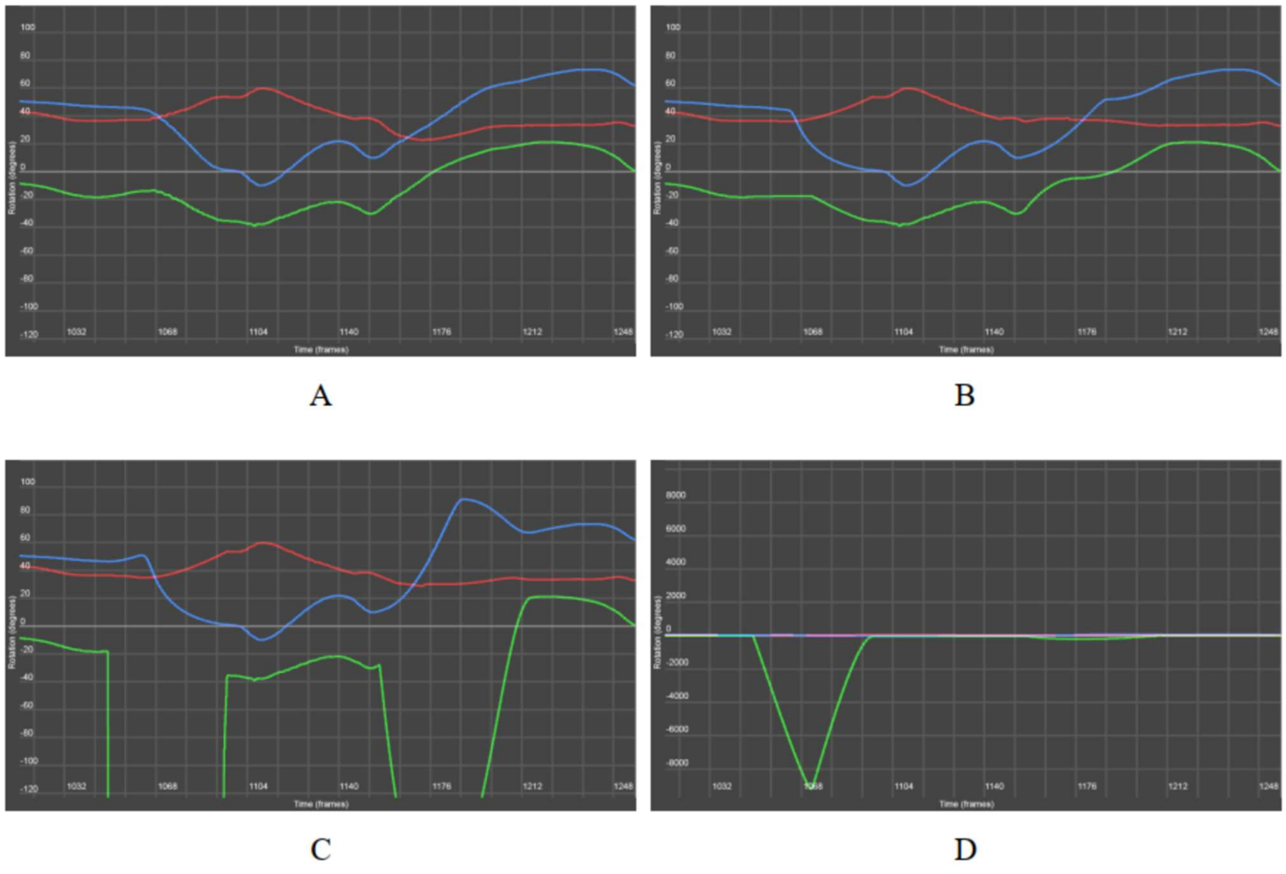


Image generated in Autodesk Maya.

Another important factor to notice if using the scale formula is that as *Δy* approaches 0 numerical errors are prone to increase to the point of, if *Δy* = 0, a zero division error occurs.

For these reasons we used the skewing presented on Equations 3.7 to 3.10 in Section 3.7.1 on the transition animation curves instead of scaling them, successfully avoiding issues with too low *Δy*.

Having found a closer transition in the database based on the parameters proximity, interpolation between signs was done using both Hermite and B-Spline curves (stored in the curves database) as well as simple linear interpolation and what we called simple Hermite, which consisted of a Hermite curve with tangents half the length of the transition range and orientations following the end of first sign's animation curve and beginning of second sign's animation curve, in the same fashion as explained with the Hermite curve fitting shown in Appendix A.1.

For each animation curve (shoulder rotation in x, y, and z; elbow rotation in x; and wrist rotation in x, y, and z) we calculated the RMSE between the original mocap values and the four different curves (linear, simple Hermite, Hermite, and B-Spline). Below we present a table that shows RMSE results for an interpreter's sentence with 6 intervals as well as the animation curves graphs for elbow rotation in x, y, and z with original (mocap) data and the interpolation curves before mentioned. The results are presented and discussed in the next section.

## 4.2 RESULTS

We successfully applied the method to 3 sets (1 for each interpreter) of the same 4 sentences that were recorded in the mocap studio at CTI as presented in Section 3.2. Although our initial goal was to apply the method on all the sentences of the 4 interpreters recorded we only applied the method to 3, since the first interpreter recorded had a very different sign structure than the others. This was an unexpected setback that we learned as we conducted the experiments, since it was our (mis) understanding that sign language had a fairly regular structure that would result in similar, if not equal, translations from spoken language. We were lucky to be able to detect this on our second interpreter recording session and thus we've managed to create the same translation with the latest three interpreters. On top of that, the virtual skeleton used for the first interpreter was not the same as the others, requiring an animation re-targeting to conform with the skeleton used in the other three (that were the same), which was not viable due to time constraints.

The animation transitions recorded for the first interpreter were kept in the animation curves and parameters database and could be used to synthesize the transitions of the other three interpreters. With this, we managed to populate the database with erroneous data that was later used (below) to validate our method, since if a lot of transitions were selected from this first interpreter data that would indicate the transition search part of the method didn’t have enough significance.

**Table 4. 1. Transitions selected by the method. Interval ID columns identify the transition to be synthesized and Matched ID columns refers to the transition selected by the method. In red are highlighted transitions selected from the first interpreter (DA) and in light blue are highlighted transitions that have been successfully selected from a similar transition from another interpreter.**

| Interpreter 1 (KA) | |
|---|---|
| Interval ID | Matched ID |
| ka0101 | th04012 |
| ka0102 | ka01041 |
| ka0103 | ka04021 |
| ka0104 | ka01071 |
| ka0105 | ro01091 |
| ka0106 | th01061 |
| ka0107 | ka01041 |
| ka0108 | th04041 |
| ka0109 | ro01041 |
| ka0110 | ro02031 |
| ka0111 | ka01031 |
| | |
| ka0201 | ka03011 |
| ka0202 | th02021 |
| ka0203 | th01071 |
| ka0204 | th01031 |
| ka0205 | ka04031 |
| ka0206 | th02061 |
| | |
| ka0301 | ka02011 |
| ka0302 | ka01101 |
| ka0303 | ka04021 |
| ka0304 | th01101 |
| ka0305 | da04041 |
| ka0306 | th03061 |
| | |
| ka0401 | ro04011 |
| ka0402 | th04021 |
| ka0403 | th01041 |
| ka0404 | da01071 |
| ka0405 | ka04061 |
| ka0406 | ka04051 |

| Interpreter 2 (RO) | |
|---|---|
| Interval ID | Matched ID |
| ro0101 | da02032 |
| ro0102 | ro01051 |
| ro0103 | ro02031 |
| ro0104 | ro01101 |
| ro0105 | ro01091 |
| ro0106 | ro02041 |
| ro0107 | ro01051 |
| ro0108 | ka01071 |
| ro0109 | ro01051 |
| ro0110 | ro03051 |
| ro0111 | ro01071 |
| | |
| ro0201 | th01011 |
| ro0202 | ro01091 |
| ro0203 | ro01031 |
| ro0204 | th04051 |
| ro0205 | ro01101 |
| ro0206 | ro04061 |
| | |
| ro0301 | ro02011 |
| ro0302 | ka01071 |
| ro0303 | ro03041 |
| ro0304 | ro03051 |
| ro0305 | ro01101 |
| ro0306 | ro01021 |
| ro0307 | th03061 |
| | |
| ro0401 | th04011 |
| ro0402 | th04021 |
| ro0403 | ro03051 |
| ro0404 | ro02031 |
| ro0405 | ro01041 |
| ro0406 | ro02061 |

| Interpreter 3 (TH) | |
|---|---|
| Interval ID | Matched ID |
| th0101 | th02011 |
| th0102 | th02031 |
| th0103 | ka02041 |
| th0104 | th02051 |
| th0105 | ro02041 |
| th0106 | ka01061 |
| th0107 | th04041 |
| th0108 | th03021 |
| th0109 | th03051 |
| th0110 | th03041 |
| th0111 | th04061 |
| | |
| th0201 | th03011 |
| th0202 | ka02021 |
| th0203 | th04041 |
| th0204 | th04021 |
| th0205 | th01041 |
| th0206 | th04061 |
| | |
| th0301 | th02011 |
| th0302 | th01081 |
| th0303 | th01061 |
| th0304 | th01101 |
| th0305 | th01091 |
| th0306 | th02061 |
| | |
| th0401 | ro04011 |
| th0402 | ro04021 |
| th0403 | th01031 |
| th0404 | th01071 |
| th0405 | ro02041 |
| th0406 | th02061 |

Table 4.1 shows some errors (highlighted in red) in which the method found as a closer transition to another one from the first interpreter (*da*). There were 3 occurrences of such an error in the 88 transitions (3.41%), this is much less than the expected 25% occurrences that would happen if we selected at random a transition from the database (the whole transitions database have a bit over a quarter of transitions originated from the first interpreter, 30 out of the total 118). On the other hand, the light blue transitions, which are

the expected selection of the same transition from a different interpreter, represent 12.5% of all the transitions, which is much higher than the expected 2.28% (2 in 88) occurrences if transitions were selected at random.

Another noteworthy result is seen in Table 4.2. This table shows transitions that were selected from the same interpreter. The method matched 56 out of the 88 occurrences (63.64%) with a transition from the same interpreter, and this is much higher than the expected 25%, if we count the first interpreter's transitions, or 33%, if we count only the last three interpreters that had the same movement and virtual skeleton.

**Table 4. 2. Transitions selected by the same interpreter. Same transitions from the previous table, but now those that are selected from the same interpreter are highlighted.**

| Interpreter 1 (KA) | |
|---|---|
| Interval ID | Matched ID |
| ka0101 | th04012 |
| ka0102 | ka01041 |
| ka0103 | ka04021 |
| ka0104 | ka01071 |
| ka0105 | ro01091 |
| ka0106 | th01061 |
| ka0107 | ka01041 |
| ka0108 | th04041 |
| ka0109 | ro01041 |
| ka0110 | ro02031 |
| ka0111 | ka01031 |
| | |
| ka0201 | ka03011 |
| ka0202 | th02021 |
| ka0203 | th01071 |
| ka0204 | th01031 |
| ka0205 | ka04031 |
| ka0206 | th02061 |
| | |
| ka0301 | ka02011 |
| ka0302 | ka01101 |
| ka0303 | ka04021 |
| ka0304 | th01101 |
| ka0305 | da04041 |
| ka0306 | th03061 |
| | |
| ka0401 | ro04011 |
| ka0402 | th04021 |
| ka0403 | th01041 |
| ka0404 | da01071 |
| ka0405 | ka04061 |
| ka0406 | ka04051 |

| Interpreter 2 (RO) | |
|---|---|
| Interval ID | Matched ID |
| ro0101 | da02032 |
| ro0102 | ro01051 |
| ro0103 | ro02031 |
| ro0104 | ro01101 |
| ro0105 | ro01091 |
| ro0106 | ro02041 |
| ro0107 | ro01051 |
| ro0108 | ka01071 |
| ro0109 | ro01051 |
| ro0110 | ro03051 |
| ro0111 | ro01071 |
| | |
| ro0201 | th01011 |
| ro0202 | ro01091 |
| ro0203 | ro01031 |
| ro0204 | th04051 |
| ro0205 | ro01101 |
| ro0206 | ro04061 |
| | |
| ro0301 | ro02011 |
| ro0302 | ka01071 |
| ro0303 | ro03041 |
| ro0304 | ro03051 |
| ro0305 | ro01101 |
| ro0306 | ro01021 |
| ro0307 | th03061 |
| | |
| ro0401 | th04011 |
| ro0402 | th04021 |
| ro0403 | ro03051 |
| ro0404 | ro02031 |
| ro0405 | ro01041 |
| ro0406 | ro02061 |

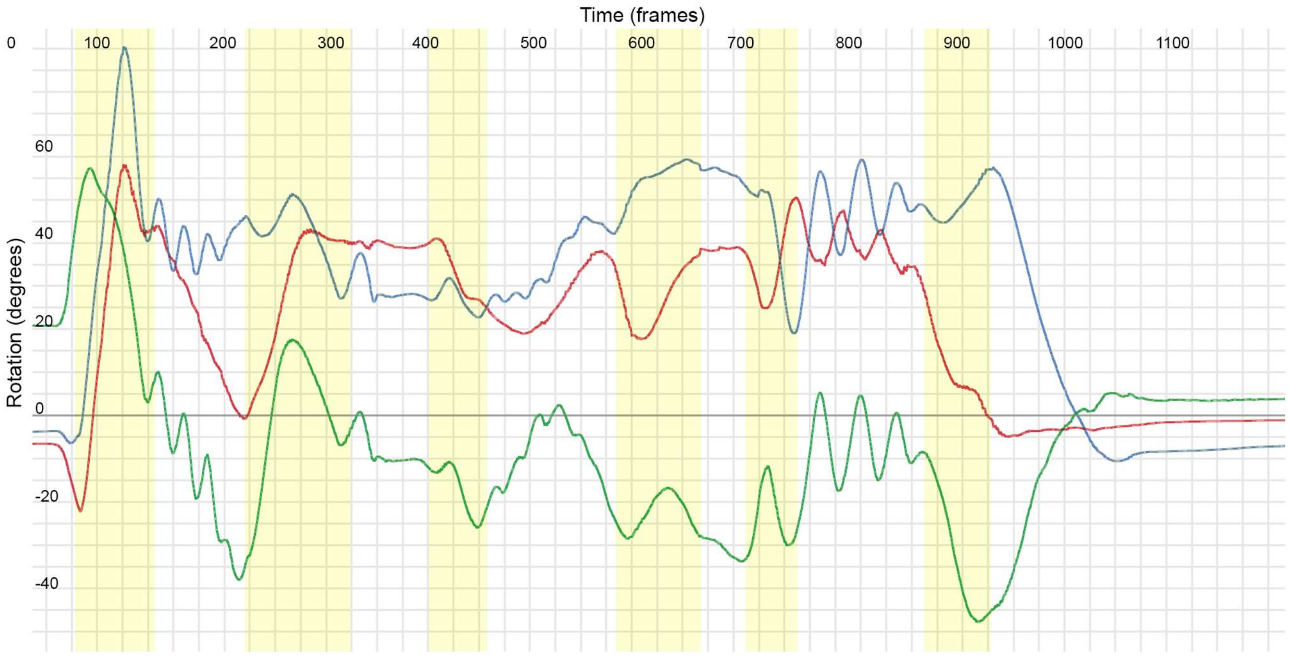
| Interpreter 3 (TH) | |
|---|---|
| Interval ID | Matched ID |
| th0101 | th02011 |
| th0102 | th02031 |
| th0103 | ka02041 |
| th0104 | th02051 |
| th0105 | ro02041 |
| th0106 | ka01061 |
| th0107 | th04041 |
| th0108 | th03021 |
| th0109 | th03051 |
| th0110 | th03041 |
| th0111 | th04061 |
| | |
| th0201 | th03011 |
| th0202 | ka02021 |
| th0203 | th04041 |
| th0204 | th04021 |
| th0205 | th01041 |
| th0206 | th04061 |
| | |
| th0301 | th02011 |
| th0302 | th01081 |
| th0303 | th01061 |
| th0304 | th01101 |
| th0305 | th01091 |
| th0306 | th02061 |
| | |
| th0401 | ro04011 |
| th0402 | ro04021 |
| th0403 | th01031 |
| th0404 | th01071 |
| th0405 | ro02041 |
| th0406 | th02061 |

These two tables can be interpreted as a result of the propositions exposed in Chapter 2, and in particular similarity of movement. Table 4.2 results show that the peculiarities of a single individual's movement seem to override a more general model, since instead of selecting a transition of a similar movement from another interpreter, the method has selected as a closer transition another movement from the same person. But, interesting to note, the remaining 32 transitions that were not chosen from the same person were in a great number (11, or more than 33%) chosen from the same movement but from another interpreter, which does contribute to confirm the first proposition.

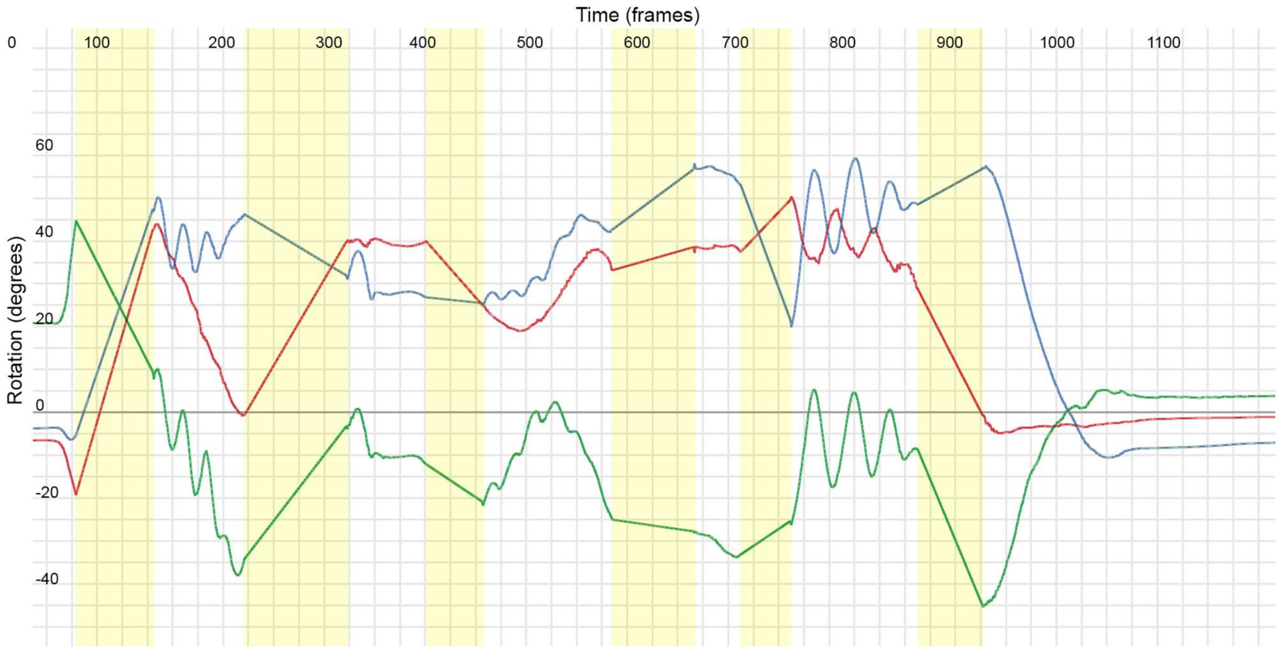
**Table 4. 3. RMSEs for a Libras sentence with 6 transitions. Under TRANSITION X it's shown the transition's id stored in the system (for instance, the first one, ka0401, corresponds to the interpreter ka's fourth sentence, and first transition) and below it, under transition selected, which transition was selected in the database (for instance, to the ka0401 the system selected as the closest match ro04011, which corresponds to the interpreter ro's fourth sentence, first transition, and right arm). The four rightmost columns show the RMSE on each animation curve between the original mocap value and the interpolated linear, simple Hermite, Hermite, and Spline.**

| | Transition | RMSE | | | |
|---|---|---|---|---|---|
| Object | | linear | sHermite | Hermite | spline |
| *TRANSITION 1* | ka0401 | | | | |
| *transition selected:* | ro04011 | | | | |
| R_Humerus.rotateX | | 17.21 | 20.38 | 17.42 | 21.06 |
| R_Humerus.rotateY | | 12.81 | 7.07 | 12.01 | 10.26 |
| R_Humerus.rotateZ | | 25.61 | 27.80 | 28.26 | 26.17 |
| R_Elbow.rotateX | | 16.91 | 8.76 | 2.54 | 6.93 |
| R_Wrist.rotateX | | 28.18 | 30.73 | 30.41 | 25.63 |
| R_Wrist.rotateY | | 6.07 | 8.55 | 6.78 | 13.33 |
| R_Wrist.rotateZ | | 30.23 | 29.80 | 30.29 | 29.84 |
| **Average value:** | | **19.57** | **19.01** | **18.24** | **19.03** |
| *TRANSITION 2* | ka0402 | | | | |
| *transition selected:* | th04021 | | | | |
| R_Humerus.rotateX | | 9.66 | 9.26 | 7.34 | 5.33 |
| R_Humerus.rotateY | | 19.02 | 17.15 | 11.30 | 12.69 |
| R_Humerus.rotateZ | | 4.87 | 7.07 | 6.78 | 7.05 |
| R_Elbow.rotateX | | 5.16 | 3.43 | 1.86 | 1.92 |
| R_Wrist.rotateX | | 19.36 | 17.67 | 7.26 | 7.30 |
| R_Wrist.rotateY | | 8.88 | 8.32 | 2.59 | 2.44 |
| R_Wrist.rotateZ | | 23.09 | 22.86 | 12.44 | 12.01 |
| **Average value:** | | **12.86** | **12.25** | **7.08** | **6.96** |
| *TRANSITION 3* | ka0403 | | | | |
| *transition selected:* | th01041 | | | | |
| R_Humerus.rotateX | | 1.71 | 1.81 | 1.28 | 1.44 |
| R_Humerus.rotateY | | 3.02 | 2.77 | 2.55 | 2.47 |
| R_Humerus.rotateZ | | 2.33 | 2.75 | 2.32 | 2.40 |
| R_Elbow.rotateX | | 3.83 | 4.10 | 4.92 | 4.56 |
| R_Wrist.rotateX | | 2.46 | 1.86 | 0.75 | 5.05 |
| R_Wrist.rotateY | | 8.35 | 4.08 | 4.60 | 4.26 |
| R_Wrist.rotateZ | | 7.44 | 6.71 | 10.83 | 9.81 |
| **Average value:** | | **4.16** | **3.44** | **3.89** | **4.28** |
| *TRANSITION 4* | ka0404 | | | | |
| *transition selected:* | da01071 | | | | |
| R_Humerus.rotateX | | 9.37 | 6.67 | 13.06 | 12.74 |
| R_Humerus.rotateY | | 4.80 | 5.19 | 3.94 | 2.80 |
| R_Humerus.rotateZ | | 5.19 | 4.24 | 1.90 | 1.86 |
| R_Elbow.rotateX | | 13.63 | 9.31 | 5.24 | 8.32 |
| R_Wrist.rotateX | | 10.77 | 21.91 | 34.15 | 32.83 |
| R_Wrist.rotateY | | 18.74 | 11.57 | 31.56 | 31.12 |
| R_Wrist.rotateZ | | 15.05 | 27.34 | 62.00 | 59.46 |
| **Average value:** | | **11.08** | **12.32** | **21.69** | **21.30** |
| *TRANSITION 5* | ka0405 | | | | |
| *transition selected:* | ka04061 | | | | |
| R_Humerus.rotateX | | 8.79 | 8.95 | 5.67 | 4.54 |
| R_Humerus.rotateY | | 7.24 | 7.94 | 9.78 | 7.59 |
| R_Humerus.rotateZ | | 5.73 | 6.48 | 7.25 | 8.74 |
| R_Elbow.rotateX | | 2.62 | 1.88 | 0.54 | 3.71 |
| R_Wrist.rotateX | | 4.93 | 7.33 | 8.79 | 14.30 |
| R_Wrist.rotateY | | 29.14 | 23.94 | 31.99 | 42.22 |
| R_Wrist.rotateZ | | 16.08 | 9.75 | 4.69 | 28.69 |
| **Average value:** | | **10.65** | **9.47** | **9.82** | **15.69** |
| TRANSITION 6 | ka0406 | | | | |
| transition selected: | ka04051 | | | | |
| R_Humerus.rotateX | | 3.97 | 4.64 | 6.30 | 6.05 |
| R_Humerus.rotateY | | 5.44 | 4.96 | 14.03 | 13.48 |
| R_Humerus.rotateZ | | 3.57 | 2.28 | 10.46 | 10.96 |
| R_Elbow.rotateX | | 5.23 | 1.48 | 2.02 | 6.17 |
| R_Wrist.rotateX | | 6.68 | 1.03 | 6.99 | 7.55 |
| R_Wrist.rotateY | | 8.25 | 4.63 | 30.86 | 32.72 |
| R_Wrist.rotateZ | | 10.46 | 3.00 | 24.50 | 30.72 |
| **Average value:** | | **6.23** | **3.14** | **13.60** | **15.38** |

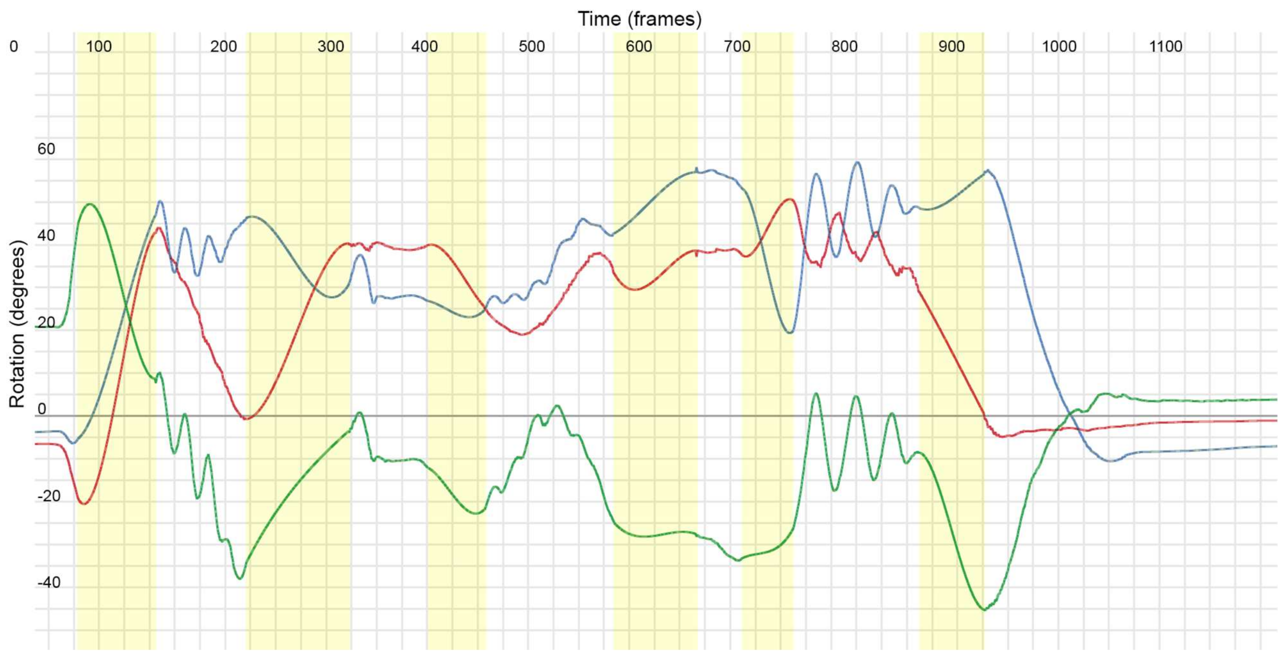
**Figure 4. 4. Animation curves for interpreter ka's fourth sentence shoulder rotation (x-axis is represented in red, y-axis in green, and z-axis in blue). Highlighted in yellow are the areas corresponding to the transition segment that was interpolated by the method. (a) is the original data, (b) the linear, (c) simple Hermite, (d) Hermite and (e) Spline curves interpolation.**



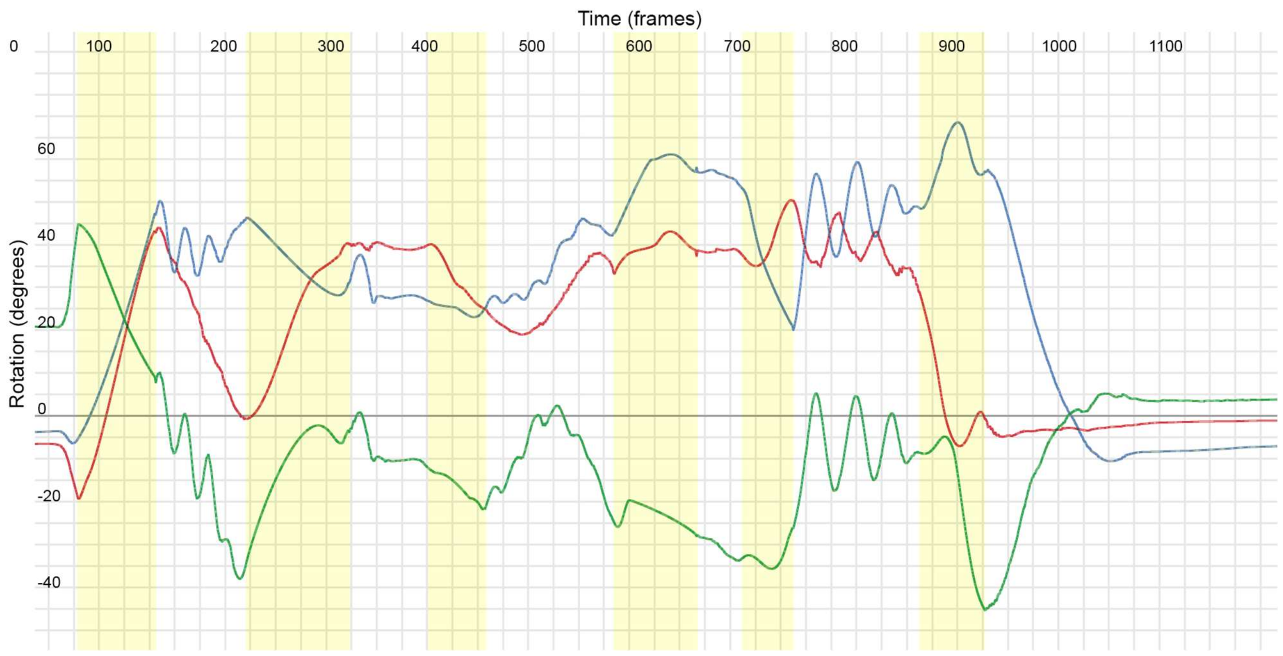
*(a) Original*



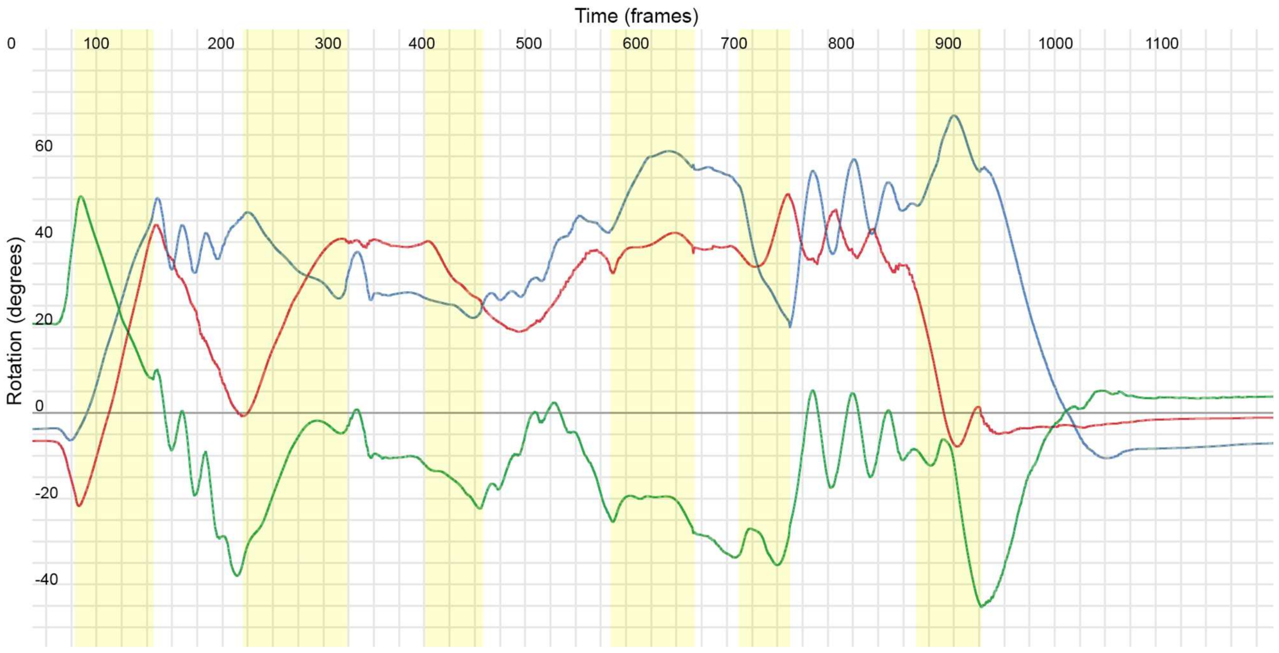
*(b) linear*



*(c) simple Hermite*



*(d) Hermite*



*(e) Spline*

Images generated in Autodesk Maya.

Figure 4.3 shows some of the features the proposed method can synthesize that wouldn't be possible in a simple curve interpolation solution. Focusing on the elbow's *y* rotation (green curve) for the second transition we can see that linear interpolation is not a satisfactory model because it produces a sharp change in C2 continuity, which is usually perceived as a very unnatural movement; simple Hermite curve interpolation produces a smooth transition but doesn't have some of the details that Hermite and Spline transitions derived from other mocap animation has, such as the peak inside the transition that is present in the original data as well.

This sentence (*ka04*, or fourth sentence of the *ka* interpreter) is presented here because it contains in its fourth transition an error in the closest match step, since it selected as the closest match another transition from the first interpreter (*da*), which as mentioned before shouldn't have happened since the interpreter motion and virtual skeleton were very different from the other three. We can see this fourth transition (identified as *ka0404*) has a better visual result with a simple Hermite curve (Figure 4.3.c), since Hermite and Spline curves (Figures 4.3.d and Figures 4.3.e, respectively) present some characteristics that are not similar at all with the original data and are even the opposite with a peak on the *x* (red) curve while in original we have a valley. Interestingly, though, this is not shown in the RMSE results; we can deduce that the best RMSE linear result for transition *ka0404* (Table 4.3) is due to the fact that the original transition animation curves have a complex shape (i.e. multiple changes in first and second derivative), and a linear interpolation curve, being a rough average (weighted average) between the initial and final values, would bear a lower RMSE.

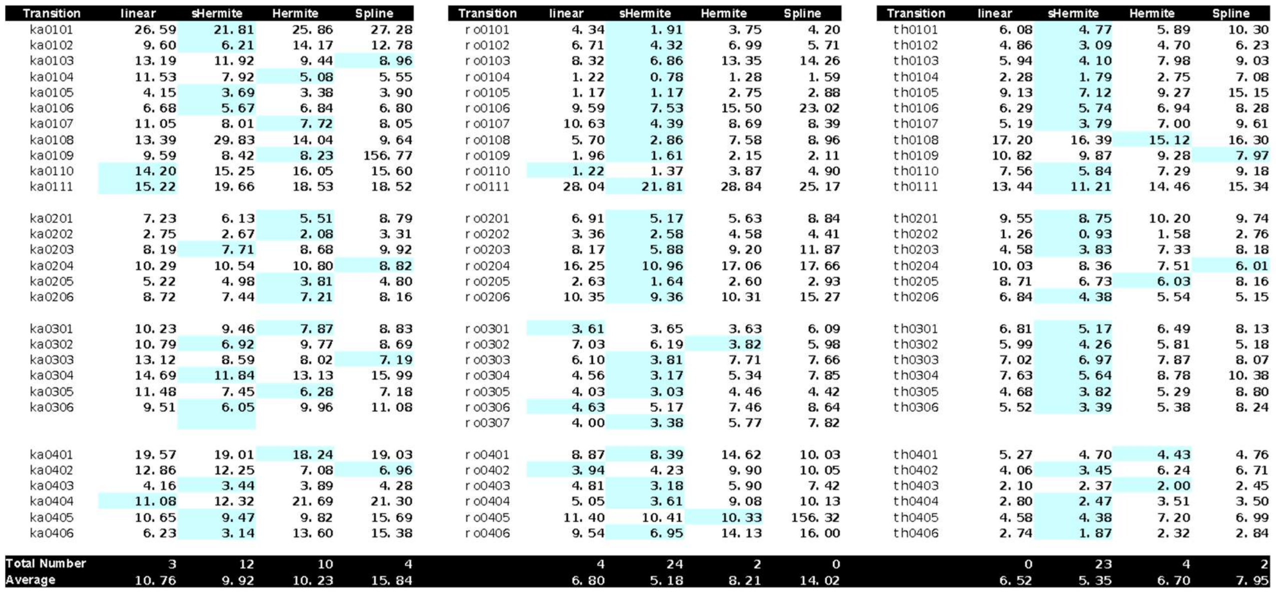
Analyzing just the RMSE we can see in Table 4.3 that the average values for the different curve types are not very different, but when we count the number of occurrences for each curve as best RMSE we can see that simple Hermite curves are by far the lowest in score (therefore better fit) with 59 transitions having lowest RMSEs as simple Hermite; followed by the method's Hermite curve using mocap transition information with 16; and in last linear and spline curves appear almost tied, with 7 and 6, respectively.

RMSE results could be complemented with a more subjective evaluation, where people fluent in Libras would be able to score the different animation curves transitions. This was envisioned in the project, but due to time constraints, it had to be left behind. We can, though, see some of the animation curves' graphs, like the one shown in Figure 4.3, and reach some conclusion.

What seems to be the big advantage of the simple Hermite model is that it produces a smoother, less complex (i.e. presence of peaks and valleys), curve than the others, and since transitions between sign languages in our mocap database are usually fast this Hermite model fits better with expected animation curves shapes. When we have more complex animation curves shapes, Hermite and spline curves are the best interpolation models, especially when they are originated from the same movement (have the same initial and final signs), as can be seen in the first and second transitions in Figure 4.3 that show more complex animation curves in the original mocap data and thus have the Hermite (first transition) and spline (second transition) yielding the best RMSE.

A way of applying this knowledge gained with the results and analysis of the proposed method is if we take inspiration from the work of PULLEN and BREGLER (2002). There could be a transition animation curve decomposition into higher and lower frequency bands and, when using it to interpolate between signs, combine a Hermite curve model in the low frequency with a mocap-based high frequency band to give the texturing PULLEN and BREGLER (2002) referred to.

**Table 4. 4. Average RMSEs by transitions. This table is a summary of the average RMSE found when interpolating the transitions between signs using the 4 different curve types. The full RMSE for one interpreter's 4 sentences transitions (that is, with each animation curve RMSE) can be found in Appendix B.**



| Transition | linear | sHermite | Hermite | Spline |
|---|---|---|---|---|
| ka0101 | 26.59 | 21.81 | 25.86 | 27.28 |
| ka0102 | 9.60 | 6.21 | 14.17 | 12.78 |
| ka0103 | 13.19 | 11.92 | 9.44 | 8.96 |
| ka0104 | 11.53 | 7.92 | 5.08 | 5.55 |
| ka0105 | 4.15 | 3.69 | 3.38 | 3.90 |
| ka0106 | 6.68 | 5.67 | 6.84 | 6.80 |
| ka0107 | 11.05 | 8.01 | 7.72 | 8.05 |
| ka0108 | 13.39 | 29.83 | 14.04 | 9.64 |
| ka0109 | 9.59 | 8.42 | 8.23 | 156.77 |
| ka0110 | 14.20 | 15.25 | 16.05 | 15.60 |
| ka0111 | 15.22 | 19.66 | 18.53 | 18.52 |
| ka0201 | 7.23 | 6.13 | 5.51 | 8.79 |
| ka0202 | 2.75 | 2.67 | 2.08 | 3.31 |
| ka0203 | 8.19 | 7.71 | 8.68 | 9.92 |
| ka0204 | 10.29 | 10.54 | 10.80 | 8.82 |
| ka0205 | 5.22 | 4.98 | 3.81 | 4.80 |
| ka0206 | 8.72 | 7.44 | 7.21 | 8.16 |
| ka0301 | 10.23 | 9.46 | 7.87 | 8.83 |
| ka0302 | 10.79 | 6.92 | 9.77 | 8.69 |
| ka0303 | 13.12 | 8.59 | 8.02 | 7.19 |
| ka0304 | 14.69 | 11.84 | 13.13 | 15.99 |
| ka0305 | 11.48 | 7.45 | 6.28 | 7.18 |
| ka0306 | 9.51 | 6.05 | 9.96 | 11.08 |
| ka0401 | 19.57 | 19.01 | 18.24 | 19.03 |
| ka0402 | 12.86 | 12.25 | 7.08 | 6.96 |
| ka0403 | 4.16 | 3.44 | 3.89 | 4.28 |
| ka0404 | 11.08 | 12.32 | 21.69 | 21.30 |
| ka0405 | 10.65 | 9.47 | 9.82 | 15.69 |
| ka0406 | 6.23 | 3.14 | 13.60 | 15.38 |
| Total Number | 3 | 12 | 10 | 4 |
| Average | 10.76 | 9.92 | 10.23 | 15.84 |

| Transition | linear | sHermite | Hermite | Spline |
|---|---|---|---|---|
| ro0101 | 4.34 | 1.91 | 3.75 | 4.20 |
| ro0102 | 6.71 | 4.32 | 6.99 | 5.71 |
| ro0103 | 8.32 | 6.86 | 13.35 | 14.26 |
| ro0104 | 1.22 | 0.78 | 1.28 | 1.59 |
| ro0105 | 1.17 | 1.17 | 2.75 | 2.88 |
| ro0106 | 9.59 | 7.53 | 15.50 | 23.02 |
| ro0107 | 10.63 | 4.39 | 8.69 | 8.39 |
| ro0108 | 5.70 | 2.86 | 7.58 | 8.96 |
| ro0109 | 1.96 | 1.61 | 2.15 | 2.11 |
| ro0110 | 1.22 | 1.37 | 3.87 | 4.90 |
| ro0111 | 28.04 | 21.81 | 28.84 | 25.17 |
| ro0201 | 6.91 | 5.17 | 5.63 | 8.84 |
| ro0202 | 3.36 | 2.58 | 4.58 | 4.41 |
| ro0203 | 8.17 | 5.88 | 9.20 | 11.87 |
| ro0204 | 16.25 | 10.96 | 17.06 | 17.66 |
| ro0205 | 2.63 | 1.64 | 2.60 | 2.93 |
| ro0206 | 10.35 | 9.36 | 10.31 | 15.27 |
| ro0301 | 3.61 | 3.65 | 3.63 | 6.09 |
| ro0302 | 7.03 | 6.19 | 3.82 | 5.98 |
| ro0303 | 6.10 | 3.81 | 7.71 | 7.66 |
| ro0304 | 4.56 | 3.17 | 5.34 | 7.85 |
| ro0305 | 4.03 | 3.03 | 4.46 | 4.42 |
| ro0306 | 4.63 | 5.17 | 7.46 | 8.64 |
| ro0307 | 4.00 | 3.38 | 5.77 | 7.82 |
| ro0401 | 8.87 | 8.39 | 14.62 | 10.03 |
| ro0402 | 3.94 | 4.23 | 9.90 | 10.05 |
| ro0403 | 4.81 | 3.18 | 5.90 | 7.42 |
| ro0404 | 5.05 | 3.61 | 9.08 | 10.13 |
| ro0405 | 11.40 | 10.41 | 10.33 | 156.32 |
| ro0406 | 9.54 | 6.95 | 14.13 | 16.00 |
|  | 4 | 24 | 2 | 0 |
|  | 6.80 | 5.18 | 8.21 | 14.02 |

| Transition | linear | sHermite | Hermite | Spline |
|---|---|---|---|---|
| th0101 | 6.08 | 4.77 | 5.89 | 10.30 |
| th0102 | 4.86 | 3.09 | 4.70 | 6.23 |
| th0103 | 5.94 | 4.10 | 7.98 | 9.03 |
| th0104 | 2.28 | 1.79 | 2.75 | 7.08 |
| th0105 | 9.13 | 7.12 | 9.27 | 15.15 |
| th0106 | 6.29 | 5.74 | 6.94 | 8.28 |
| th0107 | 5.19 | 3.79 | 7.00 | 9.61 |
| th0108 | 17.20 | 16.39 | 15.12 | 16.30 |
| th0109 | 10.82 | 9.87 | 9.28 | 7.97 |
| th0110 | 7.56 | 5.84 | 7.29 | 9.18 |
| th0111 | 13.44 | 11.21 | 14.46 | 15.34 |
| th0201 | 9.55 | 8.75 | 10.20 | 9.74 |
| th0202 | 1.26 | 0.93 | 1.58 | 2.76 |
| th0203 | 4.58 | 3.83 | 7.33 | 8.18 |
| th0204 | 10.03 | 8.36 | 7.51 | 6.01 |
| th0205 | 8.71 | 6.73 | 6.03 | 8.16 |
| th0206 | 6.84 | 4.38 | 5.54 | 5.15 |
| th0301 | 6.81 | 5.17 | 6.49 | 8.13 |
| th0302 | 5.99 | 4.26 | 5.81 | 5.18 |
| th0303 | 7.02 | 6.97 | 7.87 | 8.07 |
| th0304 | 7.63 | 5.64 | 8.78 | 10.38 |
| th0305 | 4.68 | 3.82 | 5.29 | 8.80 |
| th0306 | 5.52 | 3.39 | 5.38 | 8.24 |
| th0401 | 5.27 | 4.70 | 4.43 | 4.76 |
| th0402 | 4.06 | 3.45 | 6.24 | 6.71 |
| th0403 | 2.10 | 2.37 | 2.00 | 2.45 |
| th0404 | 2.80 | 2.47 | 3.51 | 3.50 |
| th0405 | 4.58 | 4.38 | 7.20 | 6.99 |
| th0406 | 2.74 | 1.87 | 2.32 | 2.84 |
|  | 0 | 23 | 4 | 2 |
|  | 6.52 | 5.35 | 6.70 | 7.95 |

.

This chapter has shown some of the issues we've encountered while researching and developing the method presented. We've shown that we've navigated through a plethora of subjects in computer science, ranging from data structure in animation curves and parameters database to curve decomposition in DCT form and complex curve fitting algorithms for the Hermite and B-Spline curves. As expected, no mastery can be achieved at this level and with so many different and complex subjects, and as a result a lot of the work could be improved, especially within the Hermite curve fitting algorithm.

Nonetheless, the results yielded by the method have given us hope that there is something very promising that we've manage to scratch. Putting aside the low RMSEs from the Hermite and B-Spline curve fitting algorithms implemented, we've shown that the three propositions raised from previous research on human arm movement in Chapter 2 (similarity of movement, initial configuration interference and wrist interference) held true when our method found the closest transition curves match in the same interpreter (63.64% of the occurrences) or another interpreter doing the same movement (12.5%). RMSE results, though, seem to suggest that a simpler approach (simple Hermite curve model) is usually better since the transition time between signs are small enough that high frequency details in Hermite and Spline curve fitting were basically noise. This is not true when we have a more complex original animation curve like the first and second transitions in Figure 4.3 where our method successfully matched a transition animation curve set from another interpreter doing the same movement. This could imply that a more complex method, mixing high and low frequency bands like PULLEN and BREGLER (2002) used in their work could yield more accurate results.

This will be further talked about in the next and final chapter, where we conclude our work, giving a brief review of what has been done and propose further works based on what has been researched and implemented.

# 5 CONCLUSION AND FURTHER WORK

In this chapter we discuss the results and to which degree they have answered the questions on how to develop a method for transitioning between signs in Libras in the next sections (5.1.1.Similarity of Movement, 5.1.2.Curve Models and 5.1.3.B-Spline versus Hermite). In Section 5.2 we present further work envisioned in this research.

### 5.1.1 Similarity of Movement

Chapter 2 has presented previous works on the biomechanics of arm movement and thus gave us the 3 propositions that guided our method:

- Similarity of Movement
- Initial Configuration Interference
- Wrist Interference

These suggest that similar arm configuration – that is, an arm that starts and ends movement with very similar position, velocity, and acceleration – will create similar movement.

One way to verify this hypothesis is to look at the correspondences the method found when searching for the closest match between the signs to be concatenated and the transitions stored in the databases. This information is given in Tables 4.1 and 4.2 and can be summarized as:

- 11 (12.5%) transitions selected from the expected similar transition – same sentence and same transition. If selected at random, the expected percentage would be 1.69%.
- 56 (63.64%) transitions selected from the same interpreter. If selected at random, the expected percentage would be 24.58%.
- 3 (3.41%) transitions selected from the corrupted data set (as explained in Section 4.2). If selected at random, the expected percentage would be 25.42%.

These numbers reveal that the transitions search algorithm performed as expected, since the numbers of similar and same interpreter transitions correspondences were much higher than if made by a random selection. Similarly, the correspondence to the corrupted data set was as well much lower than if made by a random selection.

Besides supporting the 3 propositions before-mentioned, this also can be seen as indicative of the validity of using DCT as parameters to compare similar arms configurations.

Another conclusion we can assert is that an interpreter's own personal way of moving in a certain sense overrides what the authors have pointed out in Chapter 2 as a more general rule of arm movement, and we can see that by the fact that in 63.64% of the cases the transitions selected were those from the same interpreter while only in 12.5% the transitions selected were those expected (same transition in the same sentence but from another interpreter). This means that although the movement done could be predicted by the same model, the way an interpreter moves – i.e. positions its arms after the end of the first sign and at the beginning of the second one – has more similarities between one's own movement, even if for a different target, than those of different interpreters aiming towards the same target.

### 5.1.2 Curve Models

In Chapter 4 we have shown the results of applying different animation curves to synthesize transition animation between signs in terms of the RMSE between the generated curves and the original mocap data.

We can synthesize this information in the following Table:

**Table 5. 1. Statistics for different curve types. The average RMSE column presents the average RMSE for all the animation curves in all transitions. The number of Lowest RMSE column indicates in how many of the 88 transitions the respective curve type scored the lowest RMSE.**

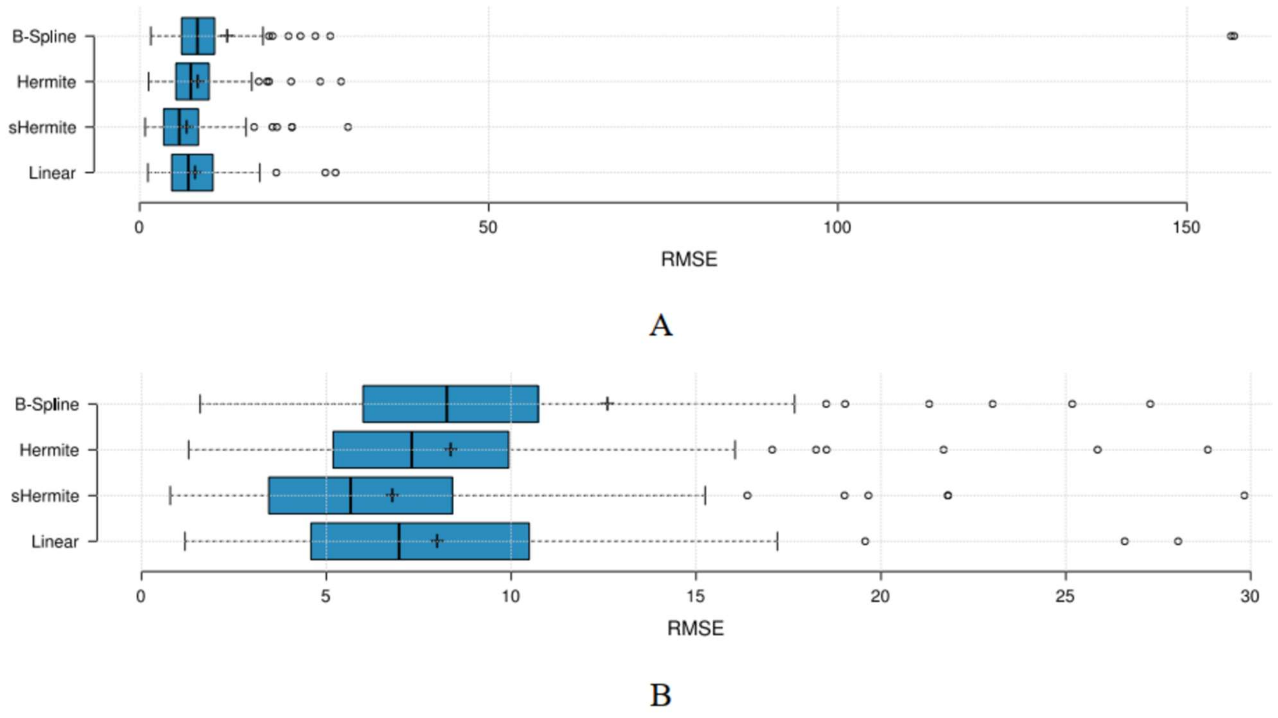
| Curve Type | Average RMSE | Number of Lowest RMSE |
|---|---|---|
| Linear | 8,06 | 7 |
| sHermite | 6,84 | 57 |
| Hermite | 8,41 | 17 |
| B-Spline | 12,67 | 7 |

The first conclusion we can assert from the data presented is that a better general fit, that is the lowest RMSE considering all animation curves in all transitions, is the simple Hermite (sHermite) curve, and that could be a result of the short transition time, averaging

65.318 frames or 0.54 seconds. We can speculate that in such a short time frame the best results are simpler curves that maintain C2 curve continuity.

Linear and Hermite curves have a very similar Average RMSE score, but we can see from the boxplot in Figure 5.1 below that the Linear curve has a bigger deviation again probably due to the short transition times.

**Figure 5. 1. Boxplot of RMSE values for each curve type (A) with zoom in the range 0-30 (B). Outliers are represented as the white circles and the crosses represent average RMSE values.**



A last observation to be made with these data is that B-Spline curves presented 2 outliers with very high values (that can be seen in Figure 5.1.A) and this is due to the coefficients generated by the fitting algorithm (Appendix A) with one example presented in Code 5.1.

**Code 5. 1. B-Spline coefficients from one of the transition animation curves that caused high RMSE results (outliers). The problematic coefficients are highlighted in red.**

```
[17.2778, -178277.7668, 34585.2352, -6149.7985, 22.2879, 14.6077, 16.6538, 17.6752, 23.5036, 25.4611, 24.9802]
```

### 5.1.3 B-Spline versus Hermite

As we have shown in the previous section, there is a big difference in performance on our method using B-Spline and Hermite curve models, namely the RMSE on Hermite curves are lower and there were no outliers.

One explanation for this is the fact that the B-Spline curves were much more complex, presenting a larger number of control points (coefficients and knot vector) if compared to Hermite's (points and tangents). Below is an example of one of the curves (X rotation in humerus joint):

For **B-Spline** fitting we have 10 coefficients (-1.7424, 0.43291, -0.20011, 6.6217, 1.986, 18.9936, 34.9427, 48.0801, 53.9813, 56.0629) and 6 middle knot values (0, 0, 0, 0, 0.08318, 0.18719, 0.55111, 0.70267, 0.81888, 0.9149, 1, 1, 1, 1) with a total of 16 values to define the curve, while for **Hermite** fitting the same curve is represented by only 8 values: p0=[0,0], p1=[1,1], t0=[1.6124,1.5229] and t1=[2.721,4.9222]. This higher number of descriptors for the B-Spline representation was consistent throughout the curves in the database.

B-Spline algorithm gave more precise fitting results, with consistent lower RMSE in comparison to the Hermite fitting algorithm, but also that came with the cost of speed. Running the same Matlab R2015 in a lapotop computer (2.1 GHz i7-3621QM processor, 8 GB RAM) each sentence would take an average of about 30 minutes to complete the Hermite fitting, while B-Spline fitting for the same sentences would take 4 times more, sometimes even needing 3 hours to complete the process.

## 5.2 FURTHER WORK

Previous section has shown in short how we developed the method to synthesize inter-sign arm movement in a virtual Libras interpreter. We have applied our method to gaps between mocap signs and analyzed the results in detail in Chapter 4.

In this final section, we present some ideas to push the work here presented forward, in ways that either due to time or limited scope of this project we couldn't have worked on.

### 5.2.1 XML files

Mocap data was stored in simple XML files (Chapter 3) so as to be easily available for the method to retrieve animation curves and transition parameters. For that, we also designed XML files storing sentence segmentation (i.e. frames ranges in mocap animation that correspond to signs and transitions) in a structure that could be both easy to navigate in the API provided to read XML files as well as be readable by humans.

The XML sentence segmentation file described in Chapter 3 presents a workable but not well-formed file and thus does not conform to the XML syntax rules (RAY, 2003). To address this issue, the main modification that we would envision in a further version of this project is for each element to have a name that is used for all the elements of such type. This would be the case for all the elements in the XML sentence segmentation files, including the root element. The proposed alterations are as followed:

- Root element: the element name should clarify that this is a segmentation file, having attributes describing the version, interpreter's name, and possibly creation date, resulting in a structure like **<sseg date="01-02-2013" interpreter="name" version="1.0">** .
- Sentence element: should be translated to the English word *sentence* instead of the Portuguese *frase* and contain the number as an attribute, not as its name, resulting in a structure like **<sentence number="1">.**
- Segment element: its name should be the same for all elements, the identification of each segment should be expressed in a new attribute, *type*, the *position* attribute should also be renamed to a more precise word, *order*, and the *word* attribute could be taken off since it can be implied by the new attribute, resulting in a structure like **<segment end="100" order="1" start="1" type="sign">.**
- Additionally, it should be interesting to include a reference to the file(s) containing the animation, as an additional element child of the root element.

### 5.2.2 Hermite Curves Fitting

We have presented in Appendix A the method by which we fitted Hermite curves into the mocap animation curves by:

1. splitting them into their peaks and valleys;
2. evaluating the tangent inclination on them by averaging the tangents around those splitting points;
3. and lastly using a brute force method for finding the best tangent (or weight) computing the Hermite curve value's RMSE for different tangent weights and selecting that with lower value.

This method presents a fast and satisfactory way of finding the tangents for the Hermite curve segments, but another approach would be to develop the method proposed in the B-Spline fitting algorithm by DIERCKX (1993).

The Hermite curve formula can be written as:

$$p(x) = (2x^3 - 3x^2 + 1)p_0 + (x^3 - 2x^2 + x)t_0 + (-2x^3 + 3x^2)p_1 + (x^3 - x^2)t_1; x \in [0,1] \quad \textbf{\textit{(5.2)}}$$

And we can use the formulation presented in the B-Spline fitting algorithm section with a Hermite curve representation instead of a B-Spline, therefore we would try to find *r* in the equation:

$$r = min \sum_{i=1}^{n} (y_i - p(x_i))^2 \quad \textbf{\textit{(5.3)}}$$

This would need to be written in the matrix form and solution yielded by using the pseudo-inverse method as with the B-Spline curve fitting.

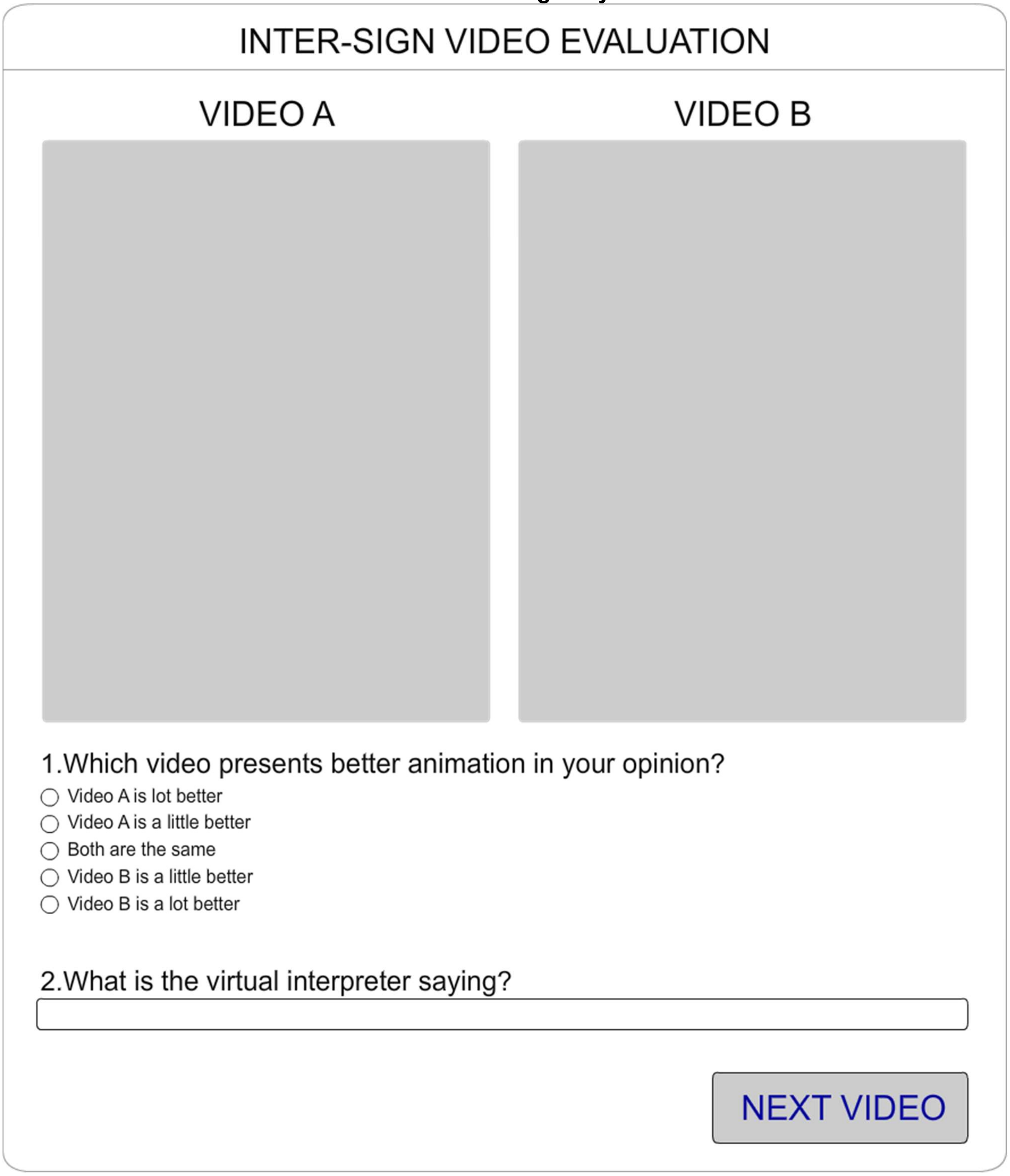
### 5.2.3 Subjective Evaluation

We've presented in Chapter 4 the method's results by comparing the synthesized transition animation curves with the original mocap animation curves using RMSE. This provides a numerical value to compare and evaluate the different curve models used (linear, simple Hermite, Hermite, and B-Spline).

In future works, it would be interesting to evaluate in a more subjective approach as well. We envision this to be done by having deaf or Libras proficient individuals compare two different videos of different animation curves models. The videos would be randomly selected from the set of transitions created with the method as well as the original mocap animation.

Besides evaluating how natural the different transition animations are perceived we could also ask participants to write down what are the words being signed by the virtual avatar, enabling us to evaluate the animation's intelligibility.

**Figure 5. 2. Wireframe for the proposed animation curves models subjective evaluation. Question 1 would have the participants compare between two different animation curves transitions of the same two signs and decide which, if any, is better and Question 2 can be used to evaluate the intelligibility of the model.**



INTER-SIGN VIDEO EVALUATION

VIDEO A

VIDEO B

1.Which video presents better animation in your opinion?

- ◯ Video A is lot better
- ◯ Video A is a little better
- ◯ Both are the same
- ◯ Video B is a little better
- ◯ Video B is a lot better

2.What is the virtual interpreter saying?

Image from the author.

### 5.2.4 Parameters Extraction and Best Match

In Section 3.5 it is shown how we established the transition animation parameter as DCT coefficients from the end of the first sign and the beginning of the second, using the DCT-II formulation by SALOMON et MOTTA (2010), which is:

$$X_k = \sum_{n=0}^{N-1} x_n . \cos[\frac{\pi}{N} . (n + \frac{1}{2}) . k] k = 0, \ldots, N-1. \quad \textbf{\textit{(5.4)}}$$

Regarding the definition of the *N* discrete rotation values used to extract the parameters from the DCT, a higher boundary could be expected to be used as the lowest sign duration, instead of the average length of mocap signs. There are some signs that, during segmentation, were considered as an instantaneous position (there is no movement, and the sign is just a static pose) having a duration of less than 10 frames (with the quicker one lasting only 6 frames), which is too short and prone to noise errors. One such sign is "house", which is configured as the two hands with fingers extended in front of the body rotated at about 45 degrees and index, middle, and ring finger touching their respective from the opposite hand (Figure 5.2). In this case, the readability of the sign is accomplished not only on the sign itself but on the transitions. When integrating this method with a virtual avatar pipeline, this limitation has to be addressed either by decreasing the value of *N* when extracting parameters or specifying a minimum duration of signs which should be equal to or greater than *N*.

**Figure 5. 3. Interpreter signing "house".**

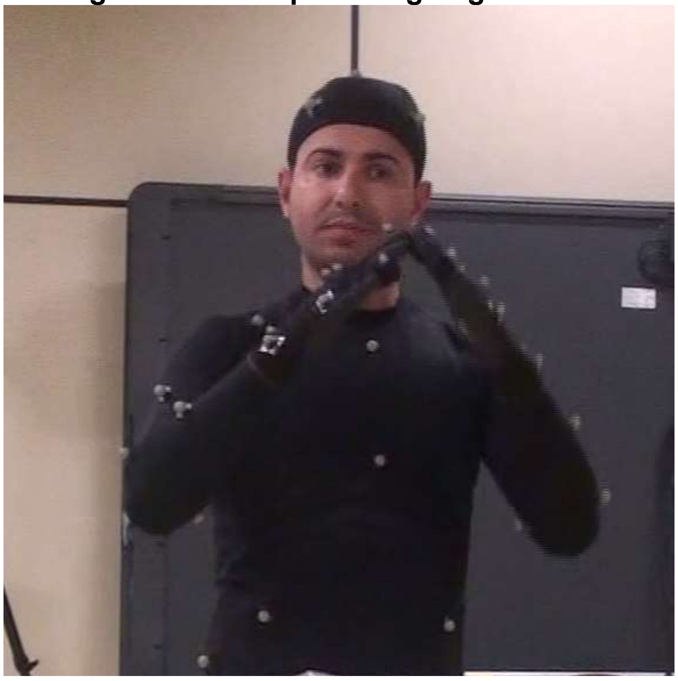


Image from the author.

For each mocap transition, we stored a parameter in the parameters database as exposed in the aforementioned section which is an XML file in the form presented in Code 5.2.

**Code 5. 2. Example of an XML transition parameter file. Values have been truncated.**

```
<TransitionDescriptor fps="120" id="ka01011" rotations="degrees" time="frames">
  <R_Humerus.rotateX in_dct00="-3.88" in_dct01="0.29" in_dct02="0.12" out_dct00="92.00" out_dct01="-1.09"
out_dct02="0.27"/>
  <R_Humerus.rotateY in_dct00="23.06" in_dct01="-0.12" in_dct02="0.07" out_dct00="-114.60" out_dct01="-0.57"
out_dct02="0.32"/>
  <R_Humerus.rotateZ in_dct00="-1.84" in_dct01="-0.027" in_dct02="0.003" out_dct00="13.64" out_dct01="0.413"
out_dct02="-0.11"/>
  <R_Elbow.rotateX in_dct00="4.07" in_dct01="-0.14" in_dct02="-0.10" out_dct00="136.05945014" out_dct01="0.53"
out_dct02="-0.06"/>
  <R_Wrist.rotateX in_dct00="-19.98" in_dct01="-0.33" in_dct02="0.087" out_dct00="138.63" out_dct01="-1.71" out_dct02="-
1.54"/>
  <R_Wrist.rotateY in_dct00="-41.80" in_dct01="-0.72" in_dct02="0.03" out_dct00="-128.60" out_dct01="-0.17"
out_dct02="0.05"/>
  <R_Wrist.rotateZ in_dct00="-32.72" in_dct01="0.33" in_dct02="-0.05" out_dct00="-192.63" out_dct01="1.67"
out_dct02="1.73"/>
</TransitionDescriptor>
```

We can see in the above example that the first DCT bands (*in_dct00* and *out_dct00*) are orders of magnitude bigger than the other bands, like in right wrist's rotate X we have -19.9859413413 in the *in_dct00* band while the value for *in_dct02* is 0.0870298131567. In practical terms, this means that the first DCT bands will overwhelm the values of the other bands when calculating transitions distances.

Principal Component Analysis (PCA) could help improve the method in 2 important ways: balancing the DCT bands and improving transitions matching.

To balance the DCT bands, PCA suggests a variance scaling on each vector of the parameters of the transition. This means for each of the parameters (*in_dct00*, *in_dct01*, *in_dct02*, *out_dct00*, *out_dct01* and *out_dct02*) we would normalize their values as:

$$dct' = \frac{dct - \bar{dct}}{S} \qquad \textit{(5.5)}$$

Where:

- *dct* is the parameter value before normalization
- *dct* is the mean value of the parameter in the database
- *S* is the standard deviation for the DCT band in the database

Improvement in transition matching can be possible by decreasing the number of parameters to be compared. With the current implementation, we have 42 parameters (6 DCT bands for each of the 7 animation curves in the shoulder, elbow, and wrist joints), this creates a very low density and sparse distribution of data, for we are dealing with a 42 dimensions point cloud, each point being a transition.

If we reduce the parameters dimensionality by using just the most significant PCA components we can have a more meaningful Euclidian distance comparison to match the desired mocap transition, that is, avoiding the curse of dimensionality for distance functions as exposed in MARIMONT et SHAPIRO, 1979.

### 5.2.5 Other Applications

The method presented in this project addresses the issue of synthesizing inter-sign arm transitions in a 3D virtual Libras interpreter, a very specific problem. But it is our belief that this same method can be generalized to different virtual joint chains and in other situations besides inter-sign transitions.

Virtual interpreter arms are a linked chain of virtual joints (shoulder, wrist and elbow) that behave as a unit towards one goal, i.e. going from the end of a sign position to the beginning of the next sign's, as exposed by the researches presented in Section 2.2. We believe that the same method presented here can also be applied to other virtual joint chains, such as legs and torso animation, since the method has no hard requirement or design into the number of joints and its DOFs. Animation curve fitting and processing are done on an individual basis, so adding or subtracting animation curves won't affect these processes, and Parameters Extractions can be easily adapted with more or fewer parameters.

Transitions database is populated with inter-sign arm movement, but we believe that it could be populated with different animations in order to produce a different movement. This could lead to other applications of the method, such as in computer games where a character would need to switch between fighting animations.

## REFERENCES

**A Computer Animated Hand**. Production: CATMULL, E.; PARKE, F. VHS. University of Utah, 1972. 1 video tape (1 min), VHS.

ADAMS, J. A.; ROGERS, D. F. **Mathematical Elements for Computer Graphics**. 2. ed. New York: McGraw-Hill Publishing, 1990.

ADMIRAAL, M. A.; KUSTERS, M. J. M. A. M.; GIELEN, S. C. A. M. Modeling kinematics and dynamics of human arm movements. **Motor control**, [*S. l.*], v. 8, n. 3, p. 312-338, 2004. DOI: 10.1123/mcj.8.3.312. Available on: https://journals.humankinetics.com/view/journals/mcj/8/3/article-p312.xml. Access: Jan. 18, 2022.

AMARAL, W. M. **Sistema de Transição da Língua Brasileira de Sinais Voltado à Produção de Conteúdo Sinalizado por Avatares 3D**. 2012. 217 pages. Thesis (PhD in Electric Engineering) – Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, Campinas, 2012.

AMARAL, W. M.; MARTINO, J. M. **Formalismo, implementação e avaliação de um sistema de transcrição para gerar conteúdo em língua de sinais em ambiente virtual** *In*: Encontro dos Alunos e Docentes do Departamento de Engenharia de Computação e Automação Industrial, 5., 2012. **Proceedings [...]**, Campinas: EADCA, 2012.

ANGARE, L.; MARTINO, J;, FERREIRA, C. **Inter-Sign Arm Movement in Libras.** *In*: International Workshop on Assistive Technology – IWAT. **Proceedings […]**, UFES, 2015.

BEANE, A. **3D Animation Essentials**. Indianapolis: Wiley, 2012.

BECK, J. **Animation Art**: From Pencil to Pixel, the History of Cartoon, Anime & CGI. UK: HarperDes, 2004.

BENDAZZI, G. **Cartoons:** One Hundred Years of Cinema Animation. Indiana: Indiana University Press, 1995. 540 p. ISBN 978-0253209375.

BURT, P. J. E.; ADELSON, E. H. The laplacian pyramid as a compact image code. **IEEE Transactions on Communications**, [*S. l.*], v. 31, n. 532-540, Apr. 1983.

CARVALHO, L. O. **Inbetweening Para Animações 2D Usando Linhas-Guias**. 2016. 84 pages. Thesis (PhD in Systems and Computer Engineering) – Universidade Federal do Rio de Janeiro, Rio de Janeiro, Jun. 2016.

COHEN, M. F. Interactive Spacetime Control for Animation. **Computer Graphics**, [*S. l.*], n. 26, vol. 2, p. 293-302, Jul. 02, 1992. Available on: https://dl.acm.org/doi/pdf/10.1145/133994.134083?casa_token=hq0VvU2UDF4AAAAA%3AugwfeXjP4QxKERa-8iXOjL4Ola_l6rz5bAPI6GWXhN3BGIKwh1pCcf0YY7DwoxSeccV9JAGAWXDujA. Access: Jan. 2022.

CYBERSIGHT. **Donders' and Listing's Laws**. [*S. n.: s. d.*]. Available on: http://www.cyber-sight.org/bins/content_page.asp?cid=735-2858-4397-4404-4654-4704-4705. Access: Nov. 2016.

DARLEY, A. Second-order Realism and Post-modernist aesthetics in Computer Animation. **A Reader in Animation Studies**. Barnet: UK: John Libbey Publishing Ltd, eBook ed. 2011.

DESMURGET, M.; GRÉA, H.; PRABLANC, C. Final posture of the upper limb depends on the initial position of the hand during prehension movements. **Experimental brain research**, [*S. l.*], v. 119, n. 4, p. 511-516, Apr. 1998. Available on: https://link.springer.com/article/10.1007/s002210050367. Access: Jan. 2022.

DIERCKX, P. **Curve and Surface Fitting with Splines**. Oxford University Press: Oxford, 1993.

DONDERS, F. C. **On the Anomalies of Accommodation and Refraction of the Eye**. London: The New Sydenham Society, 1864.

EWART, S. et al. A donders' like law for arm movements: The signal not the noise. **Frontiers in human neuroscience**, [*S. l.*], v. 10, p. 136, Mar. 30, 2016. DOI: 10.3389/fnhum.2016.00136. Available on: https://www.frontiersin.org/articles/10.3389/fnhum.2016.00136/full. Access: Jan. 2022.

FERREIRA, A. C. A. **3D Character Animation Using Sign Language**. 2017. Dissertation (Masters in Computer Science) – Departamento de Ciência de Computadores, Departamento de Ciência de Computadores, Porto, 2017.

FLEISCHER, M. **Method of Producing Moving-Picture Cartoons**. US Pat. 1242674 A, 9 Oct. 1917. Google Patents. Available on: https://patents.google.com/patent/US1242674A/en. Access: Jan. 2022.

GONCALVES, D. A.; TODT, E.; GARCIA, L. S. 3D Avatar for Automatic Synthesis of Signs for The Sign Languages. *In*: International Conference on Computer Graphics, Visualization and Computer Vision, 23., Czech Republic, 2015. **Proceedings […]**, Plzen: University of West Bohemia, 12 Jun. 2015, p. 17-23. Available on: https://dspace5.zcu.cz/handle/11025/29661. Access: Jan. 2022.

GOTTLIEB, G. L. et al. Directional control of planar human arm movement. **Journal of neurophysiology**, [*S. l.*], v. 78, n. 6, p. 2985-2998, 01 Dec. 1997. DOI: 10.1152/jn.1997.78.6.2985. Available on: https://journals.physiology.org/doi/full/10.1152/jn.1997.78.6.2985. Access: Jan. 2022.

HALAS, J.; SITO, T.; WHITAKER, H. **Timing for Animation**. 2. ed. Oxford: Focal Press, 2009.

HERRERA, J. F. R.; RUIZ, V. G. **Image coding fundamentals**. Nov. 16, 2014. Available on: https://w3.ual.es/~vruiz/Docencia/Apuntes/Coding/Image/00-Fundamentals/index.html. Access: Jan. 2022.

HISTORY of the Spline Computational Curve Design. **Alatown**, [*S. l.*], Jan. 9 2015. Available on: http://www.alatown.com/spline. Access: Jul. 2018.

INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA - IBGE. **Censo Demográfico 2010:** Características Gerais da População, Religião e Pessoas com Deficiência. IBGE: Rio de Janeiro, 2012. ISSN 01043145. Available on: http://biblioteca.ibge.gov.br/index.php/biblioteca-catalogo?view=detalhes&id=794. Access: Jan. 2022.

JAIN, E.; SHEIKH, Y.; HODGINS, J. Leveraging the talent of hand animators to create three-dimensional animation. *In*: Eurographics Symposium on Computer Animation, 9., Louisiana, 2009. **Proceedings […]**, Association for Computing Machinery: Louisiana, Aug. 1st, 2009. p. 93-102. DOI: 10.1145/1599470.1599483 10.1145/1599470.1599483. Available on: https://dl.acm.org/doi/abs/10.1145/1599470.1599483?casa_token=C1jAFT-gJcQAAAAA:KwzpWxxeOoAkcoho4bgPWDQ5uSneouI6WKAVECrh5St7xTDaKUor1Yk09dZ2aaWbjArflxLFnafvxQ. Access: Jan. 2022.

JAIN, Eakata, SHEIKH, Yaser et HODGINGS, Jessica. **Leveraging the Talents of Hand Animators to Create Three-Dimensional Animation** in *Proceedings of the 2009 ACM SIGGRAPH /* Eurographics Symposium on Computer Animation, pp. 93-102, 2009.

JOBSON, C. **Newly Digitized 'Phenakistoscope' Animations That Pre-Date GIFs by Over 150 Years**, [*S. l.*], Oct. 27 2015. Available on: https://www.thisiscolossal.com/2015/10/newly-digitized-phenakistoscopes. Access: Jan. 2022.

JONES, C. A.; DRAKE, F. L. **Python and XML**. Sebastopol: O'Reilly, 2002.

KIM, J.; CORDIER, F.; MAGNENAT-THALMANN, N. Neural Network-based Violonist's Hand Animation. *In*: Computer Graphics International, Switzerland, 2000. **Proceedings […]**, IEEE: Geneva, pp. 37-41, Jun. 19-24, 2000. DOI: 10.1109/CGI.2000. Available on: https://ieeexplore.ieee.org/abstract/document/852318. Access: Jan. 2022.

LEE, J. et al. Interactive control of avatars animated with human motion data. *In*: Annual conference on Computer graphics and interactive techniques, 29., San Antonio Texas, 2002. **Proceedings […],** New York:
Association for Computing Machinery, Jul. 23-26, 2002. p. 491-500, v. 21, n. 3. Available on: https://dl.acm.org/doi/abs/10.1145/566570.566607. Access: Jan. 2022.

LIBRAS - Dicionário da Língua Brasileira de Sinais. **Acessa Brasil**, [*S. n.: s. d.*]. Available on: http://www.acessibilidadebrasil.org.br/libras. Access: Feb. 22, 2017.

LIU, Z.; GORTLER, S. J.; COHEN, M. F. Hierarchical spacetime control. In: Annual conference on Computer graphics and interactive techniques, 21., 1994. **Proceedings […]**, New York: Association for Computing Machinery, Jul. 24, 1994. p. 35-42. DOI: 10.1145/192161.192169. Available on: https://ieeexplore.ieee.org/abstract/document/852318. Access: Jan. 2022.

LUDWIG, P. E.; LOPEZ, M. J.; SEVENSMA, K. E. Anatomy, head and neck, eye cornea.

**StatPearls**, [*S. l.*], 2017. Available on: https://www.ncbi.nlm.nih.gov/books/NBK470534. Access: Mar. 2019.

MACHADO, A. **Pré-Cinemas & Pós-Cinemas**. 4. ed. Campinas: Papirus, 2007.

MAKEHUMAN. **MakeHuman: Open Source Tool for Making 3D Characters**. [*S. l.: s. d.*]. Available on: http://www.makehuman.org. Access: Nov. 2016.

MANOVICH, L. "Real" Wars: Esthetics and Professionalism in Computer Animation. **Design Issues**, Massachusetts, v. 8, n. 1, 1991. DOI: 10.2307/1511450. Available on: https://www.jstor.org/stable/1511450?origin=crossref&seq=1#metadata_info_tab_contents. Access: Jan. 2022.

MARIMONT, R. B.; SHAPIRO, M. B. Nearest neighbour searches and the curse of dimensionality. **IMA Journal of Applied Mathematics**, [*S. l.*], v. 24, n. 1, p. 59-70, Aug. 1st, 1979. DOI: 10.1093/imamat/24.1.59. Available on: https://academic.oup.com/imamat/article-abstract/24/1/59/668588. Access: Jan. 2022.

MAROTTA, J. J.; MEDENDORP, W. P.; CRAWFORD, J. D. Kinematic rules for upper and lower arm contributions to grasp orientation. **Journal of Neurophysiology**, [*S. l.*], v. 90, n. 6, p. 3816-3827, 2003. Available on: https://journals.physiology.org/doi/full/10.1152/jn.00418.2003. Access: Jan. 2022.

MARTINO, J. M. et al. Signing avatars: making education more inclusive. **Universal access in the information society**, [*S. l.*], v. 16, n. 3, p. 793-808, Nov. 3, 2016. Available on: https://link.springer.com/article/10.1007/s10209-016-0504-x. Access: Jan. 2022.

MELIKHOV, K. et al. Frame skeleton based auto-inbetweening in computer assisted cel animation. *In*: 2004 International Conference on Cyberworlds, Japan, 2004. **Proceedings […]**, IEE: Tokyo, Nov. 18-20, 2004, p. 216-223. Available on: https://ieeexplore.ieee.org/abstract/document/1366177?casa_token=ulM8C0Y5OCMAAAA A:hwXyEICrDdC687m1aiCfZo4uXW3eHtbGLC5gVHvPuY2B5McLZA0vGAIPFQEX2kfi-AEW8jQIVLA. Access: Jan. 2022.

MOESLUND, T. B.; HILTON, A.; KRÜGER, V. A survey of advances in vision-based human motion capture and analysis. **Computer vision and image understanding**, [*S. l.*], v. 104, n. 2-3, p. 90-126, Nov.-Dec., 2006. DOI: 10.1016/j.cviu.2006.08.002. Available on: https://www.sciencedirect.com/science/article/abs/pii/S1077314206001263?via%3Dihub. Access: Jan. 2022.

MOREIRA, J. R. et al. Rumo a um sistema de tradução Português-LIBRAS. *In*: Workshop de Informática na Escola. 2011. **Proceedings [...]**, Aracaju, Nov. 21-25, 2011. p. 1543-1552. ISSN: 2176-4301. Available on: http://br-ie.org/pub/index.php/wie/article/view/1985/1744. Access: Jan. 2022.

MURTAGH, I. Towards a Linguistically Motivated Irish Sign Language Conversational Avatar Conversational Avata. **The ITB Journal**, [*S. l.*], v. 12, n. 21, p. 73, 2011. Available on: https://itb.ie/files/journal/issue-21.pdf#page=73. Access: Jan. 2022.

O'HAILEY, T. **Rig it Right! Maya Animation Rigging Concepts**. Massachusetts: Focal

Press, 2013.

O'ROURKE, M. **Principles of Three-Dimensional Computer Animation:** Modeling, Rendering and Animationg with 3D Computer Graphics. New York: W. N. Norton & Company, 1998.

PARENT, R. **Computer Animation: Algorithms & Techniques**. 3. ed. Waltham: Morgan Kaufman, 2012.

PULLEN, K.; BREGLER, C. Motion capture assisted animation: Texturing and synthesis. *In*: Annual conference on Computer graphics and interactive techniques, 29., San Antonio Texas, 2002. **Proceedings […]**, New York: Association for Computing Machinery, Jul. 2002, p. 501-508. DOI: 10.1145/566570.566608. Available on: https://dl.acm.org/doi/abs/10.1145/566570.566608. Access: Jan. 2022.

RAY, E. T. **Learning XML**. 2. ed. Sebastopol: O'Riley Media Inc, 2003.

RUGGIERO, M. A. G.; LOPES, V. L. R. **Cálculo Numérico**: Aspectos Teóricos e Computacionais. 2. ed. São Paulo: Pearson, 2010.

SÁENZ VALIENTE, R. **Arte y Técnica de la Animación:** Clásica, corpórea, computada, para juegos o interactiva. Buenos Aires: Ediciones de la Flor, 2011.

SALOMON, D.; MOTTA, G. **Handbook of Data Compression**. 5. ed. New York: Springer, 2010.

SHAMUS, C. **Animation: From Script to Screen**. New York: St. Martin's Press, 1990.

SOARES, R. et al. AnyLanguage-To-LIBRAS: Evaluation of an Machine Translation Service of Any Oralized Language for the Brazilian Sign Language. *In*: Brazillian Symposium on Multimedia and the Web, 23., 2017. **Proceedings […]**, Gramado, Oct. 2017. p. 481-488. DOI: 10.1145/3126858.3126871. Available on: https://dl.acm.org/doi/abs/10.1145/3126858.3126871?casa_token=EyLjueTQpMAAAAAA:r1Zlf3oUbBagN1ai5jd0p4_pH94YhUNdArQ7r7BN-kmZ_1L_t-EZiG6AB4uDgKVsbQyhgrzlbwi8UQ. Access: Jan. 2022.

STARNER, T.; PENTLAND, A. Real-time american sign language recognition from video using hidden markov models. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, [*S. l.*], v. 20, n. 12, p. 1371-1375, 1998. Available on: https://link.springer.com/chapter/10.1007/978-94-015-8935-2_10. Access: Jan. 2022.

THE MATHWORKS INC. **Polynomial Roots - MATLAB roots.** Available on: https://www.mathworks.com/help/matlab/ref/roots.html. Access: Dec. 2017.

THOMAS, F.; JOHNSTON, O. **The Illusion of Life:** Disney Animation. New York: Disney Editions, 1995.

TOWNSEND, A. **On the Spline**. Jan. 9, 2015. Available on: http://www.alatown.com/spline. Access: Jan. 2022.

U. S. NATIONAL PARK SERVICE. **Make a Thaumatrope**. available at <https://www.nps.gov/articles/make-a-thaumatrope.htm>, [*S. n.: s. d.*]. Access: Jan. 2022.

WATKINSON, M. **Real Time Character Animation:** a Generic Approach to Ragdoll Physics. 2009. Dissertation (Masters in Computer Science) – Faculty of Engineering and Computing, Coventry University, Coventry, 2009. Available on: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.455.2606&rep=rep1&type=pdf. Access: Jan. 2022.

WHITED, B. et al. Betweenit: An interactive tool for tight inbetweening. **Computer Graphics Forum**, Oxford, UK: Blackwell Publishing Ltd, Jun. 7, 2010. p. 605-614. DOI: 10.1111/j.1467-8659.2009.01630.x. Available on: https://onlinelibrary.wiley.com/doi/10.1111/j.1467-8659.2009.01630.x. Access: Jan. 2022.

# APPENDIX A – OVERVIEW OF CURVE FITTING

As mentioned in Section 3.2, motion capture data is presented as discrete animation curves, consisting of floating-point vectors representing rotations of shoulder, elbow, and wrist in a time range. It can be represented as a vector as follow:

$$Y^k = [y_{0,}^k y_{1,}^k y_{2,}^k \dots, y_{n-1}^k]; \qquad \textit{(A. 1)}$$

Where $Y^k$ is the *k*-th animation curve containing *y* individual values in a time range of *n* frames. In this project's context, *k = [0, 1, 2, …, 6]* since we have 7 animation curves for the 3 arm joints (3 DOFs in the shoulder, 1 DOF in the elbow, and 3 DOFs in the wrist). Since each animation curve is fitted independently the *k* index will be dismissed in further data vector representations.

As explained in Chapter 3, the storage and manipulation of animation curves as discreet values has a lot of drawbacks that led to the implementation of curve fitting methods.

Curve fitting is the process by which a mathematical model is used to approximate a data set. Given a data of *y'* we want to find a mathematical function *f(x) = y* such as to minimize the error (or distance) between *y'* and *y*, that is:

$$min|f(x) - y'|\forall x \qquad \textit{(A.2)}$$

Since we are dealing with a 2-variables function, the mathematical function used is a curve.

Two curve types fitting, Hermite and B-Spline, were implemented and the results compared. Since the curve fitting methods are very different, they will be shown and explained in the next sections separately.

## A.1 HERMITE CURVE FITTING

Hermite curves are polynomials defined by their start point ($p_0$), end point ($p_1$), initial tangent ($t_0$) and final tangent ($t_1$), as we can see in some examples in Figure A.1. The polynomial can have any degree but one of the most commonly used is the 3rd degree polynomial, which was also used in the method here described since for each DOF the first derivative represents velocity and second derivative acceleration, thus higher degree polynomials would have a higher complexity with no direct analogy to the real-life data represented by the curves.

**Figure A. 1. Hermite curves example. These Hermite curves have the same initial and final point values, p0 = [0,0] and p1 = [1,1], but with different tangent values. Top left: t0 = [1,0], t1 = [1,0]; Top right: t0 = [0,1], t1 = [0,1]; Bottom left: t0 = [1,0], t1 = [0,1]; Bottom right: t0 = [0,1], t1 = [1,0]. Images generated in MATLAB.**

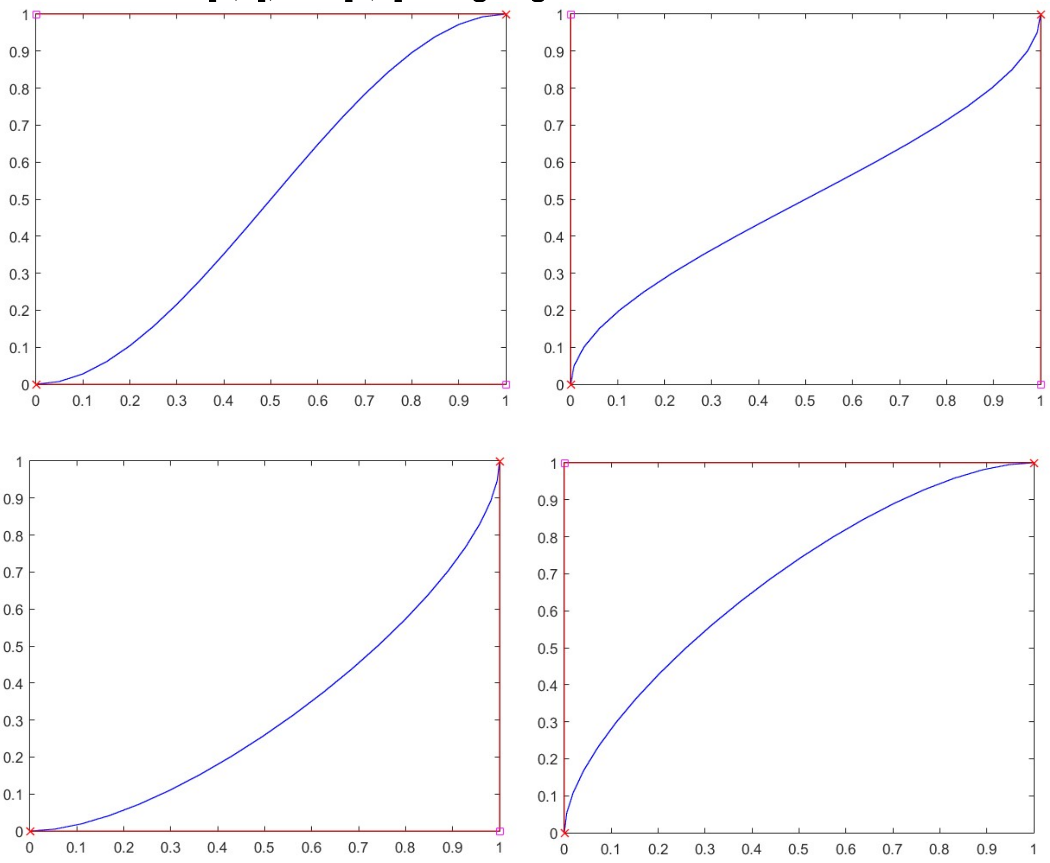


Image generated in MATLAB

A third-degree Hermite curve can be interpolated as follow:

$p(t) = (2.t^3 - 3.t^2 + 1).p_0 + (t^3 - 2.t^2 + t).t_0 + (-2.t^3 + 3.t^2).p_1 + (t^3 - t^2).t_1; t \in [0,1]$ ***(A.3)***

A Hermite curve describes only a segment of the total animation curve, so to fit the animation curve mocap data it was used several segments connected by their end points, that is:

$$\begin{Bmatrix} p_1^n = p_0^{n+1} \\ t_1^n = t_0^{n+1} \end{Bmatrix} \quad \textbf{\textit{(A.4)}}$$

The conditions expressed in Equation A.3 state that the end point ($p_1$) of the nth segment must be equal to the first point ($p_0$) of the n+1 segment, and the last tangent ($t_1$) of the nth segment should also be equal to the first tangent ($t_0$) on the n+1 segment, thus preserving a geometric continuity of $G^2$ (although the conditions presented in the equation do imply a $C_2$ parametric continuity, the algorithm changes the tangents sizes in further steps as shown in Section A.1.2, thus providing only geometric continuity $G^2$ between segments).

The curve fitting method proposed is based on the idea that each animation curve can be spliced on the local minima and maxima, that is where f''(x) = 0. After finding these initial splicing point candidates, a final selection is based on their distance in order to not splice on a too-small segment or have sign noise influence on the splicing, this will give $p_0$ and $p_1$ for each Hermite segment.

Splicing the animation curve into smaller Hermite curves yields to the next step where tangents ($t_0$ and $t_1$) are defined. To find the best tangents they are treated in 2 steps, first defining its inclination (tangent) and second its length.

Tangents inclinations are determined by the average inclination on the data points near each splicing point.

Tangents weights (or sizes) are determined by comparing the different tangent sizes on each Hermite curve and finding the best fit to the mocap data using RMSE.

Finally, an XML file is created to be used in later steps of the method proposed (Section 3.4).

These steps will be further explained next.

### A.1.1 Curve Splicing

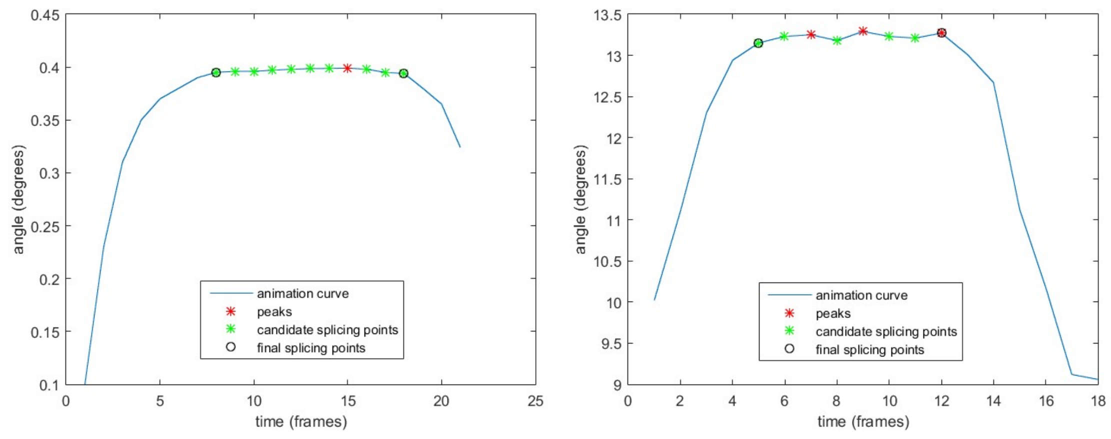
The first step to fit a Hermite Curve on the motion capture data is to splice it into segments which will be then fitted individually. A good set of splicing points is thus fundamental to achieve an overall good curve fitting. To select this best set first it is necessary to find candidate points, which are those where the second derivative of the animation curve is 0. Since the motion capture data is comprised of discrete values, these first candidate points are defined as:

$$x \ni i \Leftrightarrow (y_i - y_{i-1}).(y_{i+1} - y_i) \leq 0 \qquad \textbf{\textit{(A.5)}}$$

There are two situations that require refining on the selection of splicing points:

- A "plateau" on the signal (that is, the value on the animation curve remains constant);
- An agglomeration of consecutive prospective splicing points, probably due to a very high-frequency noise;

**Figure A. 2. complications when selecting splicing points. A plateau of relatively equal values on the animation curve (left) and agglomeration of consecutive splicing points due to high-frequency noise (right). Images generated in MATLAB.**



Images generated in MATLAB.

These situations can be dealt with by defining local errors in time (*Δx*) and value (*Δy*):

• Δy is used in the plateau scenario. This defines the minimum (rotation) value of adjacent points for them to be considered equal. It is an error range when dealing with motion capture data that may either present noise or too small rotation values on joints that can be considered the same value. After the initial set of splicing points candidates are defined, the algorithm goes through each candidate's vicinity and, if the neighbor(s) values are within the current's plus or minus Δy they are put into a temporary set of neighboring points (V). After finding all the points in the neighboring points set, the algorithm takes only its first and last points and introduces them back to the candidate splicing points set ($\hat{X}$). It is noteworthy to point out that intermediate plateaus, that is ranges of curve points within Δy but without any peaks, have been present as shown in Figure A.2 but they were successfully spliced using the method here proposed due to noise presence, which created peaks where might have not. This Δy is given by the following equation:

$$\Delta y = \frac{[max(x) - min(x)]}{100} \tag{A.6}$$

• Δx is used in the agglomeration scenario. This defines the minimum distance between splicing points. After the initial set of splicing points candidates are defined and the plateau scenario is dealt with, the algorithm goes through all of the splicing points sequentially and in the case where the distance between two or more adjacent points are less than  Δx, those points are taken from the set and a new one is introduced. The new point's x coordinate is the mean value between the subtracted points.

Since there are a lot of noises in the mocap animation curves tangent inclinations could not be considered as simply the vector given by the splicing point subtracted from its neighbors. The first approach to tangent inclination determination was to just compute the average inclination of N near points to the splicing points as in the following formulas:

$$t_1=\frac{\sum_{j=i=N}^{i}(y_j-y_{j-1})}{N} \qquad t_0=\frac{\sum_{j=i}^{i+N}(y_{j+1}-y_j)}{N} \tag{A.7}$$

Where *N* is the number of closest points used to calculate the inclination.

This approach did not produce good results, since a small N would allow noise to create a bad tangent and a big N would average too many points resulting in an inclination that wouldn't reflect the inclination of a continuous curve.

The solution used was to use a filter that smooths out the curve and at the same time maintains the overall shape with its first and second derivatives. The 1-dimensional kernel (SOLOMON, 2010) used for this was [0.1538 0.1923 0.3077 0.1923 0.1538]. Results of using this filter in the mocap curve can be seen in Figure A.3.

**Figure A. 3. Intermediate plateau. An intermediate plateau around frame 275 which has been correctly spliced due to noise in the mocap data. Image generated in MATLAB.**

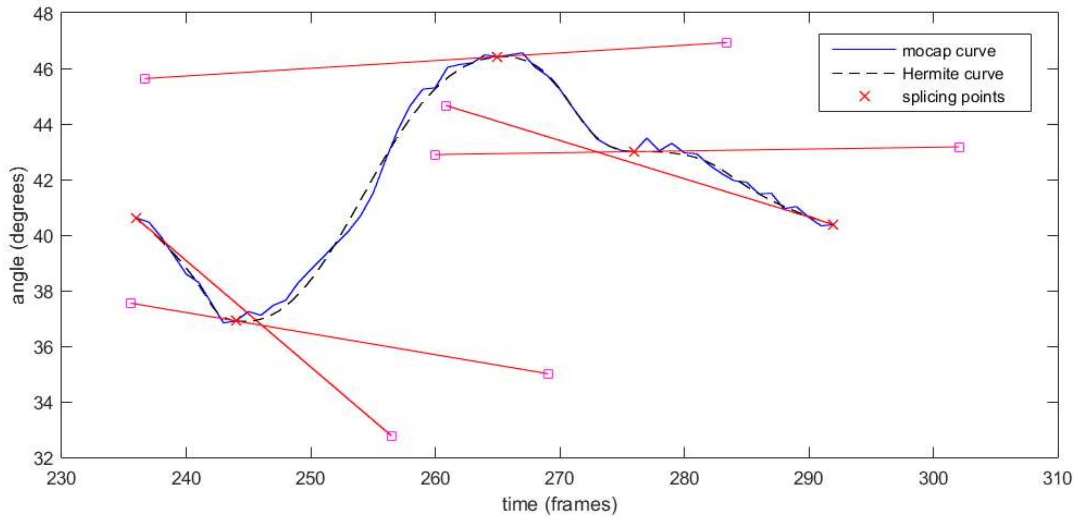


Image generated in MATLAB.

## A.1.2 Tangent Weight Determination

From the previous step tangent's direction is already taken from the mocap data. What is left to determine is their weight (or size). To do so the algorithm does a brute force comparison between different initial and final tangent sizes for each Hermite curve by multiplying the unitized tangents with a factor *k₀* and *k₁*, thus transforming Equation A.2 into:

$p(t) = (2.t^3 - 3.t^2 + 1).p_0 + (t^3 - 2.t^2 + t).t_0.k_0 + (-2.t^3 + 3.t^2).p_1 + (t^3 - t^2).t_1.k_1 t \in [0,1]$ ***(A.8)***

The factors $k_0$ and $k_1$ are defined by the user, for transition curve fitting it was used 100 equally spaced values ranging from 0 to 5 times the curve's length (frames), for each tangent's multiplying factor.

Determining the best fit was done by comparing RMSE. This needs special attention because Equation A.7 yields the Hermite curve value for a given parameter *t* and it was desirable to have a function that yields a y (rotation value) for each x (frame) in the Hermite curve. Seeing as the equation is independent for x and y coordinates, first, it is solved for *t* with x given. Starting with Equation A.2 for x:

$x = (2.t^3 - 3.t^2 + 1).p_0 + (t^3 - 2.t^2 + t).t_0.k_0 + (-2.t^3 + 3.t^2).p_1 + (t^3 - t^2).t_1.k_1$ ***(A.9)***

This can be rewritten as to put *t* in evidence in a polynomial form:

$$\boldsymbol{t^3}.(2.p_0 + t_0.k_0 - 2.p_1 + t_1.k_1)$$
$$+\boldsymbol{t^2}.(-3.p_0 - 2.t_0.k_0 + 3.p_1 - t_1.k_1)$$
$$+\boldsymbol{t}.(t_0.k_0)$$
$$-x = 0 \qquad \textbf{\textit{(A.10)}}$$

Which in turn can be written as:

$$\boldsymbol{t^3}.A + \boldsymbol{t^2}.B + \boldsymbol{t}.C + D = 0 \qquad \textbf{\textit{(A.11)}}$$

Equation A.10 is solved by using MATLAB's *roots* algorithm, which finds eigenvalues of the matrix A.11 that correspond to the polynomial's roots (THE MATHWORKS INC, 2017).

$$\begin{bmatrix} 1 & \frac{B}{A} & \frac{C}{A} & \frac{D}{A} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \textbf{\textit{(A.12)}}$$

The roots found are not necessarily valid roots, so a criterion is used in which the only selected roots should be real, no imaginary roots should be accounted - since the interpolation curve requires and exist only for real numbers - and within the closed interval

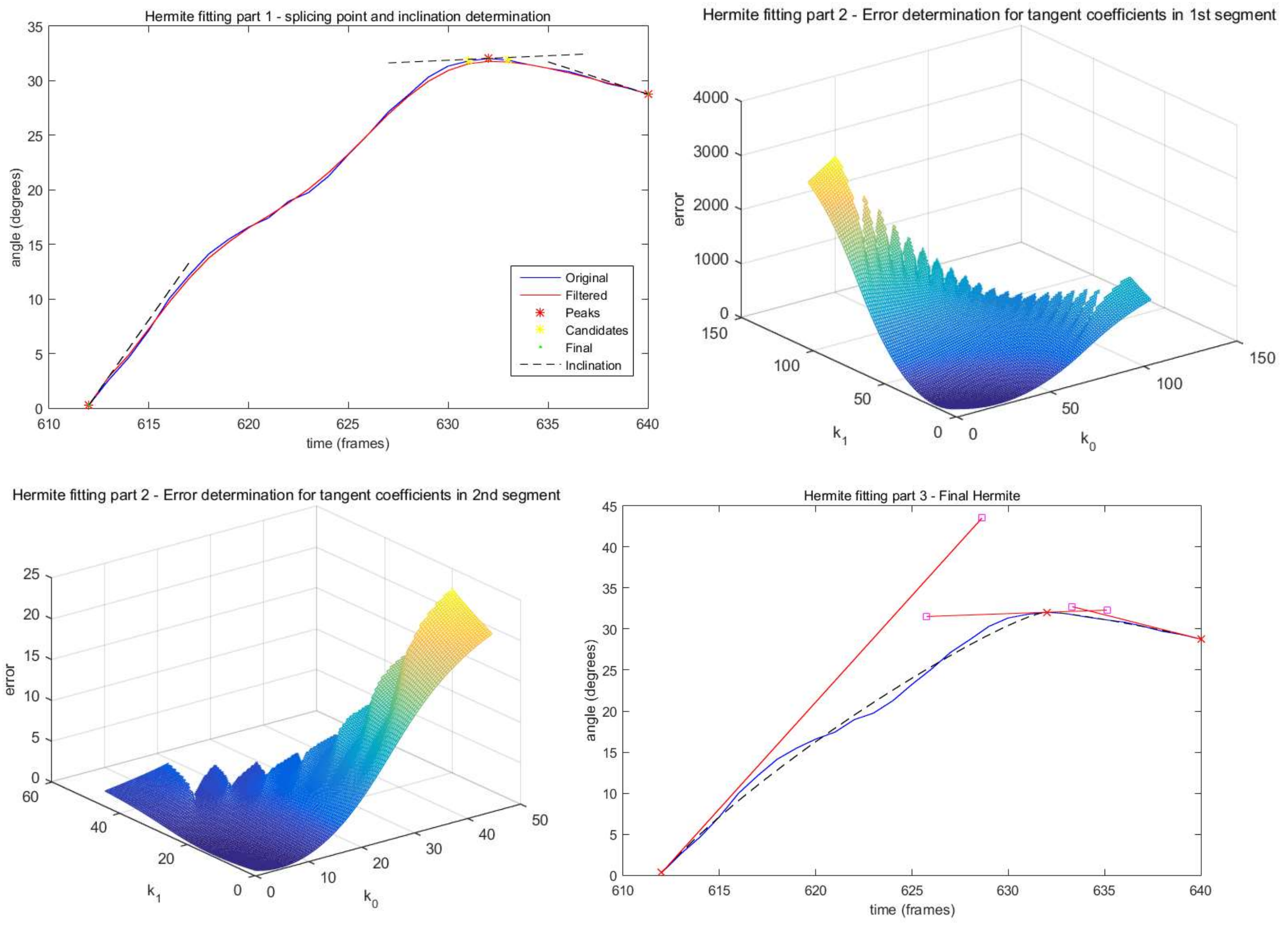
[0,1] - since the Hermite curve used is not defined outside this range. If there are more than 1 real root found, this means the Hermite curve is not a function (i.e. for a given x / frame value there are more than 1 y / rotation value), thus it can't be used as an animation curve.

With *t* given, y is calculated using the Hermite curve polynomial presented in Equation A.2 and thus an RMSE is given by finding the difference between this y and the original mocap data.

The algorithm then compares RMSEs of all given combinations of $k_0$ and $k_1$, selecting the lowest score as the final values for the tangents.

An example of the different parts of the Hermite fitting algorithm is presented in Figure A.4.

**Figure A. 4. Steps for Hermite curve fitting. Part 1 (top left) finds one splicing point, dividing the curve into two segments. On each point, including the extremities at frames 611 and 640, tangent inclinations are determined as well. For each segment we assign ranges of tangent weights k0 and k1 (top right and bottom left) and compare RMSE between the Hermite curve and original mocap data, giving the results shown. The lowest error value then determines the best fit, that is, the best values for k0 and k1 in Equation A.8. The jagged limits on the extremes of k0 and k1 ranges are due to non-existing solutions for the polynomial A.10. The bottom right image is the final Hermite curve comprised of two segments compared to original mocap data. The Hermite curve is composed by the concatenation of 2 segments given by p0=[0, 0], p1=[0.71429, 1], t0=[0.59406, 1.3619], t1=[0.22277, 0.016178] for the first segment and p0=[0.71429, 1], p1=[1, 0.8969], t0=[0.11139, 0.008089], t1=[0.23868,-0.1248] for the second segment. Note that these values are normalized.**



Images generated in MATLAB.

## A.2 B-SPLINE CURVE FITTING

A spline is a curve defined in parts by polynomial functions. Spline curves precede computer graphics as they were made using weights, also called “ducks” (Figure A.5), and a bendable ruler to draw precise curves for engineering and design, most notably on shipbuilding.

**Figure A. 5. Spline weights. Also called “ducks”, the spline weights were used for drawing curves before computer aided design.**

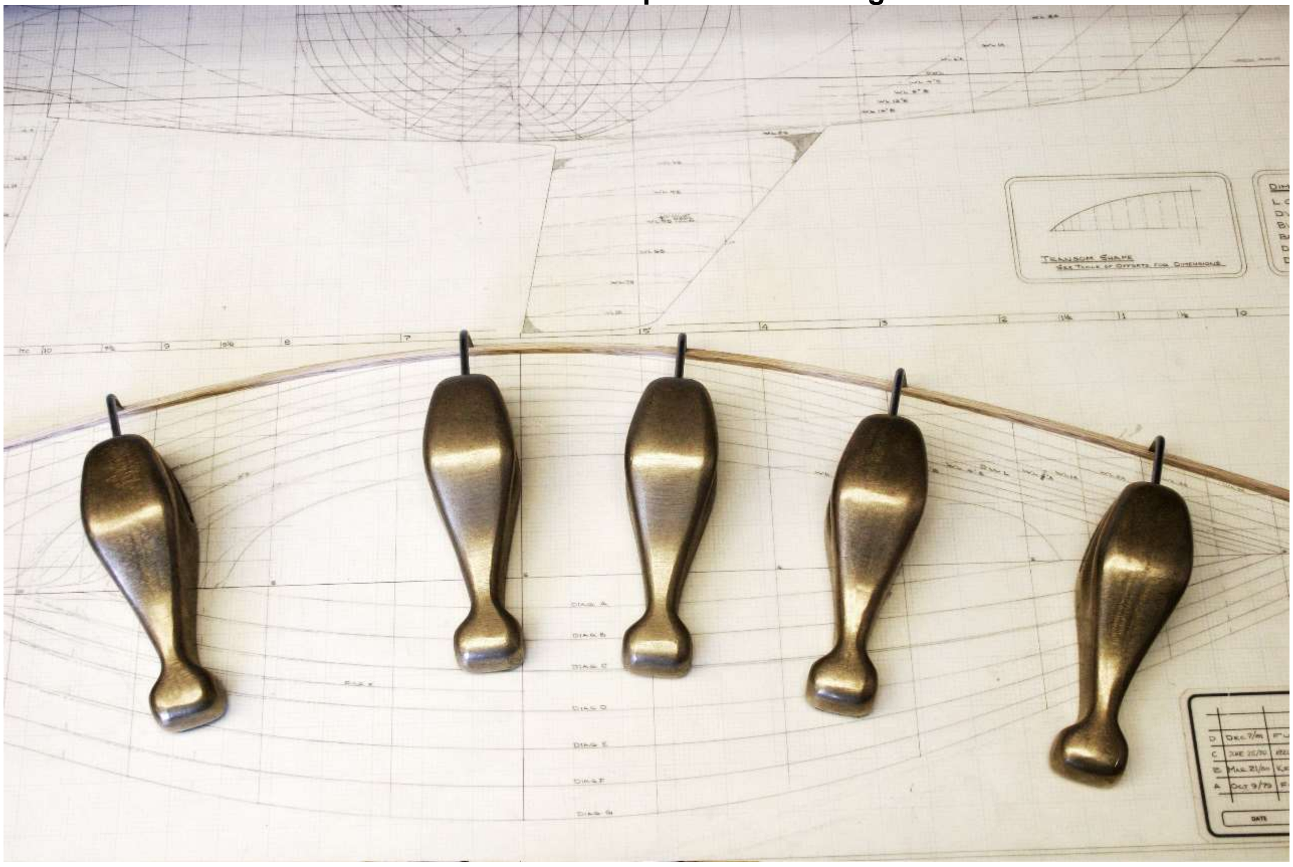


Image from TOWNSEND (2015).

A spline is a real function in the form:

$$s: [a,b] \to \mathbb{R} \qquad \textbf{\textit{(A.13)}}$$

and has a set of knots $t_j$, $j = 1, 2, \dots, (2k + g - 1)$. Where $k$ is the order of the spline (degree of polynomial + 1) and $g$ is the number of sub-intervals with knots given by:

$$[t_i, t_{i+1}], i = 0, \dots, k-1 \qquad \textbf{\textit{(A.14)}}$$

These knots can be thought of as a digital counterpart to the ducks.

B-Splines are a special type of splines constructed by linear combination of different curves called basis splines, thus the name of the curve.

Using the de Cox-Boor formulation by ADAMS and ROGERS (1990) a B-spline curve is defined as:

$$s(x) = \sum_{i=1}^{n+1} c_i . N_{i,k}(x) x_{min} < x < x_{max}, 2 \le k \le n+1 \qquad \textbf{\textit{(A.15)}}$$

The spline basis functions are the $N_{i,k}(x)$ that can be defined as:

$for\ k = 1$:

$$N_{i,1} = \begin{cases} 1, & if\ t_i \leq x < t_{i+1} \\ 0, & otherwise \end{cases} \qquad (A.16)$$

$for\ k > 1$:

$$N_{i,k}(x) = w_{i,k}(x).N_{i,k-1} + \left(1 - w_{i+1,k}\right).N_{i+1,k-1} \qquad (A.17)$$

and *w* is given by:

$$w_{i,k} = \begin{cases} \frac{x-t_1}{t_{i+k-1}-t_i}, if t_i \neq t_{i+k-1} \\ 0, otherwise \end{cases} \qquad (A.18)$$

### A.2.1 Approximation of B-Spline with Fixed Knot Vector

The B-Spline method presented here is explained in detail in DIERCKX (1993), thus only the main concepts and what was adapted from it to the problem of fitting animation curves will be presented in this work.

For the B-Splines 3 parameters were necessary to be determined on each fit:

- Spline order (*k*)
- Quantity and position of knot vector (*t*)
- B-Spline coefficients from Equation A.14 (*c*)

The most common spline order according to DIERCKX (1993) is *k* = 4 resulting in cubic splines of continuity $C^2$, physically that means angle ($C^0$), velocity ($C^1$) and acceleration ($C^2$) are maintained between segments.

To compare the fitted B-spline with original curve values, the same criteria as in the Hermite curve fitting was used, that is, Minimum Square Distance between the B-spline curve in the *x* coordinates correlated to the *n* mocap animation curve rotational values in the vector *y*:

$$r = min \sum_{i=1}^{n} (y_i - s(x_i))^2 \qquad (A.19)$$

Equation A.18 can be expanded with Equation A.14 as:

$$r = min \sum_{i=1}^{n} (y_i - \sum_{j=1}^{n} c_j.N_{j,k}(x_i))^2 \qquad (A.20)$$

This can be written in matrix form as:

$$r = min(y - E.c)^T(y - E.c) = min||y - E.c||^2 \qquad (A.21)$$

where:

$$y=\begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_m \end{bmatrix}, E=\begin{bmatrix} N_{1,k}(x_1) & N_{2,k}(x_1) & \dots & N_{n,k}(x_1) \\ N_{1,k}(x_2) & N_{2,k}(x_2) & \dots & N_{n,k}(x_2) \\ \dots & \dots & \ddots & \dots \\ N_{1,k}(x_m) & N_{2,k}(x_m) & \dots & N_{n,k}(x_m) \end{bmatrix}, c=\begin{bmatrix} c_1 \\ c_2 \\ \dots \\ c_n \end{bmatrix}$$

Solution for Equation A.20 (that is, minimization of squared Euclidean norm of *y-E.c*) can be found using the pseudo-inverse method on the equation:

$$E.c = y \tag{A.22}$$

### A.2.2 Approximation of B-Spline with Variable Knot Vector

It is shown in DIERCKX (1993) that knot quantity and position can affect greatly the fitting curve result. Too few knots are not enough for a high-frequency curve representation and a high density of knots is required where a curve has a high frequency. DIERCKX (1993) also points out that it is important to avoid coincidental or too proximal knots so as to avoid unnecessary clustering and noisy curves. To accommodate these requirements, a penalty function *P(t)* is used in Equation A.18 creating the new function to be minimized as:

$$v(t) = r(t) + p.P(t) \tag{A.23}$$

*P(t)* is given by:

$$P(t) = \sum_{i=k}^{k+g-1} \frac{1}{(t_{i+1}-t_i)}, p = e_{1}.\frac{r(t^1)}{P(t^e)} \tag{A.24}$$

where

- $t^1$ is the initial knot configuration,
- $t^e$ is the equidistant knot distribution,
- $e_1$ is a factor reflecting the desired precision for the approximation.

Since the knots divide the interval *[a,b]* equidistant distribution of knots is given by:

$$P(t^e) = \frac{g^2}{(b-a)} \tag{A.25}$$

The problem is now to minimize the non-linear function *v(t)* varying the knot vector. This can be accomplished using the Fletcher and Reeves algorithm. This algorithm, also called conjugated gradient, requires an initial knot vector, which is obtained using the method in the next section.

### A.2.3 Initial knot vector

Fletcher and Reeves algorithm will try to minimize the function *v(t)*, which is not convex, therefore the minimum found is not guaranteed to be global and could just be a local minimum to the neighborhood of $\boldsymbol{t^1}$ (initial knot vector). Hence why it is extremely important to start with a good $\boldsymbol{t^1}$**.**

The proposed method by DIERCKX (1993) defines a good initial number of knots and their positions. This is achieved by (1) establishing an initial equidistant 3-knots vector and (2) adding knots in the intervals with greater error. This method is as follow:

1. Initialize $\boldsymbol{t^1}$:

$$t^1 = \begin{bmatrix} t^1 \\ t^2 \\ t^3 \end{bmatrix} \tag{A.26}$$

with:

$$t^1 = a, \quad t^2 = \frac{a+b}{2,}, \quad t^3 = b \tag{A.27}$$

2. For $g = 1, 2, \ldots, g_{max}$ :

   I. Apply the algorithm presented in Section A.2.1 using $\boldsymbol{t^g}$ as the initial knot vector. This will determine $\boldsymbol{t^*}$ as the optimal knot position, yielding the respective spline $s_g(x)$.

   II. For $i = 1, 2, \ldots, g$ calculate the error found between each knot[3]:

$$\widetilde{r_i} = \frac{\sum_{r=j}^{j+m}(y_r - s_g(x_r))^2}{\mathrm{m} - m_i} \tag{A.28}$$

   with $t^*_i \leq x_j < x_{j+1} < \ldots \leq t^*_{i+1}$; $t^*_1 = a$, $t^*_{g+2} = b$; and $m - m_i$ defines the range (in frames, in our case) of the knots distance in the curve.

   III. If all $r_i$are smaller than an approximation error defined by the user, stop and give the current knot vector, $\boldsymbol{t^*}$, as the final knot vector. If not, define a new knot vector $\boldsymbol{t^{g+1}}$ by inserting a new knot in the current vector. Determine the new knot location, *l*, associated with the greater $r_i$as such:

[3] It is important to point out that although the author suggests the use of a weight $w_r$ for each curve point $y_r$ the algorithm implemented didn't use it. The reason being the equal importance of each sample $y_r$ (animation curve value) for each frame $x_r$ . Thus the suggested formula $w_r(y_r - s_g(x_r))$ has been substituted by the simple difference between the sampled animation curve value and its B-Spline value in the form $y_r - s_g(x_r)$.

$$\tilde{r}_l = max\{\tilde{r}_i\}, i = 1, \dots, g \quad \textbf{(A.29)}$$

Insert the new knot between the knots ***t*<sub></sub>***

$$t^{g+1} = \begin{cases} t_i^*, i = 1, \dots l \\ \frac{(t_l^* + t_{(l+1)}^*)}{2}, i = l \\ t_{i-1}^*, i = l+2, \dots, g+2 \end{cases} \quad \textbf{(A.30)}$$

## A.2.4 Fletcher and Reeves Algorithm (Conjugated Gradient)

As stated in the previous section, Fletcher and Reeves algorithm tries to find a local minimum of a function, and for that it uses an initial guess which in our case is the $t^1$ starting knot vector. The process to get this initial knot vector, shown in the previous section, can be summed up by starting with an initial 3 knots equidistant vector and inserting new knots in the curve, whereupon the approximation is worse between already existing knots.

Given the initial knot vector, the algorithm will iterate on knot vectors subtracting then the gradients of the minimum function (Equation A.30), that is the search direction vector, to arrive at a local minimum. The algorithm can be described as follow:

1. Calculate initial search direction:

$$d^1 = -\nabla v(t^1) \quad \textbf{(A.31)}$$

2. For $i$ = 1, 2, …, g – 1

(a) Minimize

$$z(a) = v.(t^i + a.d^i) \quad \textbf{(A.32)}$$

obtaining $a^*$.[4]

(b) Consider $t^{i+1} = t^i - a *. d^i$.

(c) End algorithm if

$$\frac{|v(t^{i+1}) - v(t^i)|}{v(t^i)} < e_1 \text{ and } \frac{\|t^{i+1} - t^i\|}{\|t^i\|} < e_2 \quad \textbf{(A.33)}$$

(d) Update search direction as:

$$d^{i+1} = -\nabla v(t^{i+1}) + \frac{\|\nabla v(t^{i+1})\|^2}{\|\nabla v(t^i)\|^2}.d^i \quad \textbf{(A.34)}$$

3. Make $t^1 = t^{n+k}$and go back to step 1.

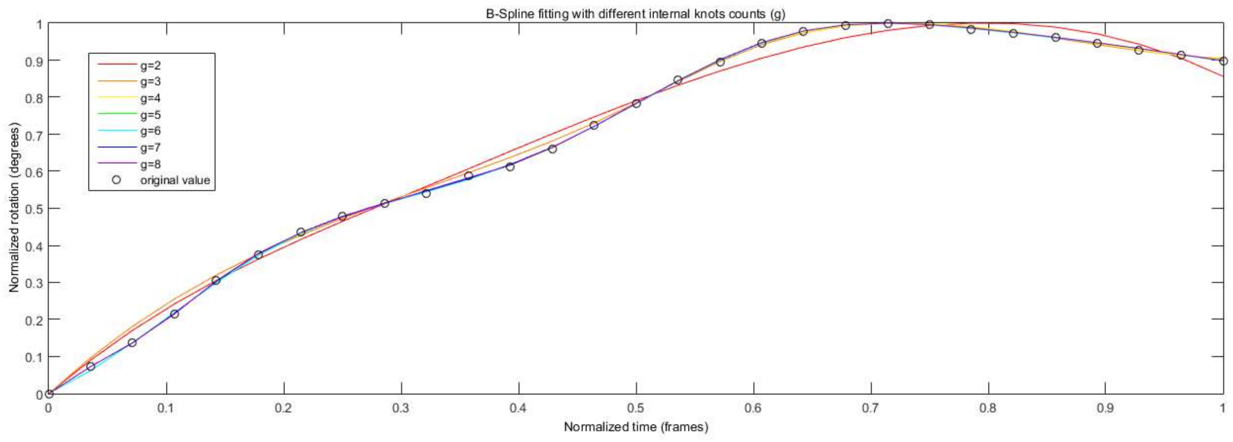
[4] The algorithm to obtain *a** can be found in DIERCKX (1993).

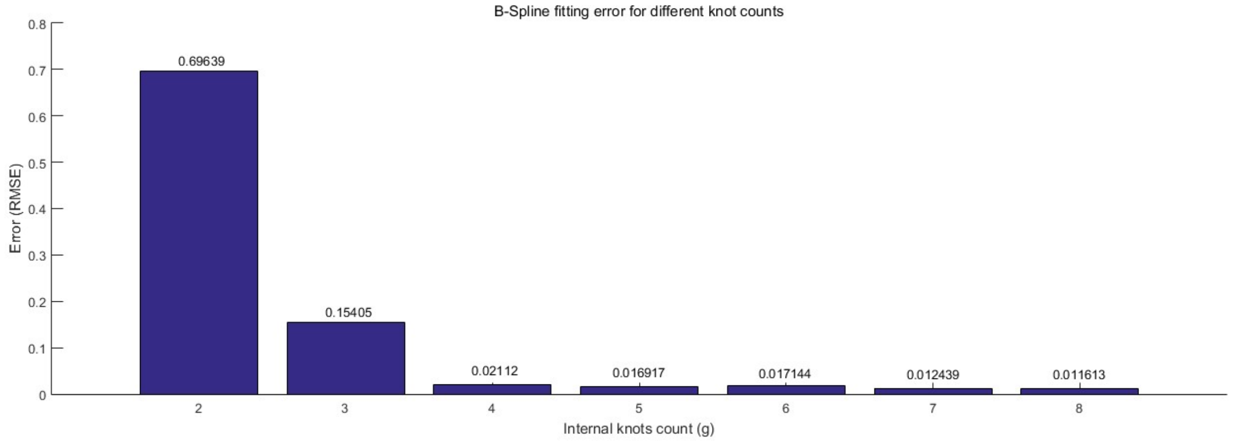
The values used in Equation A.32 to determine if the error between the current iteration and the last (e1) or the relative distance between consecutive knot vectors (*$e_2$*) were suggested in DIERCKX (1993) with values of 0.0005, but in the implementation for animation curves, it was shown that these values were too low, resulting in the algorithm not finishing its run or creating numerical errors, thus they were increased to 0.05,still yielding good results.

The algorithm to get the value of *a** can be seen DIERCKX (1993). Although not shown, in the algorithm proposed by DIERCKX (1993), *a** more often than not yielded unexpected results (for instance, *a** was attributed the value of ‘Infinite,’ caused probably by a division by 0, or attributed an imaginary number which is not expected for this problem that uses only real numbers for joint rotations, and so on), so one criterion for the algorithm to stop was to check these results. If any occurred, then the current knot vector ($\boldsymbol{t}^i$) was given as the solution.

An example of the B-Spline curve fitting with increased internal knots counts (*g*) and its associated errors (RMSE) is presented in Figure A.6.

**Figure A. 6. B-Spline fitting example. B-Spline fitting for the same data fitted in Hermite curve in Figure A.4. Top shows the different curves generated with different internal knots counts, as shown on the algorithm in the previous section. Bottom shows the different errors (RMSE) in each generated curve; this is interesting because it shows that although there was an increase in error from 5 to 6 internal knots (going from 0.016917 to 0.017144) that did not meet all the error conditions for termination as stated in Equation A.32. The spline has the coefficient vector c = [0.2969, 2.6276, 4.562, 13.6126, 16.0703, 18.5043, 21.3737, 32.8331, 31.6995, 29.8174, 28.7695] and the knot vector k = [0, 0, 0, 0, 0.093192, 0.14641, 0.30449, 0.36942, 0.36963, 0.62235, 0.78464, 1, 1, 1, 1].**





Images generated in MATLAB.

## A.3 FINAL REMARKS

As presented in Chapter 3, mocap data representing DOFs rotation values (***Y***) used for inbetweening different signs were needed to be stored in a way that could be later manipulated. This manipulation involves mainly 2 operations: stretching both time and rotation values to adjust for the new signs to be inbetweened, and tangent adjustment in the connecting points in order to create a smooth flow of movement between signs and transition animation curves (Section 3.7).

Two different curve types were used, comparing how well they would fit the purpose of the system: Hermite and B-Spline. Both used cubic polynomials at their basis, since this provides a simple curve and yet can guarantee $C^2$ which in DOFs animation curves would guarantee position, velocity, and acceleration continuity.

Cubic Hermite curve was defined in Equation A.2 as:

$$p(t) = (2.t^3 - 3.t^2 + 1).p_0 + (t^3 - 2.t^2 + t).t_0 + (-2.t^3 + 3.t^2).p_1 + (t^3 - t^2).t_1; t \in [0,1]$$

Where $p_0$ is the initial point with $t_0$ tangent, and $p_1$ the final point with $t_1$ tangent.

This of course defines only a third-degree polynomial curve, and thus is very limited. For complex DOF curves (defined as having multiple maximum and minimum points with different slopes between them) we need to segment the data curve ***Y*** and create multiple Hermite curves segments. Thus, in broad view, the Hermite curve fitting algorithm used can be stated as:

1. **Determine the splicing points.** Done by first selecting curve peaks (maximum and minimum values) and then filtering out candidate points that are either too close in time (high-frequency noise) or too close in value (plateaus) and discarding those that reside within a certain range;

2. **Determine the tangents angle** for each segment, which is obtained by averaging the closest points inclination (in this case means the average angle difference in the animation curves).

3. **Determine the tangents length** ($k_0$ and $k_1$), since these are the only 2 values needed for each segment. The Hermite curve equation becomes:

$$p(t) = (2.t^3 - 3.t^2 + 1).p_0 + (t^3 - 2.t^2 + t).t_0.k_0 + (-2.t^3 + 3.t^2).p_1 + (t^3 - t^2).t_1.k_1$$

This is solved by minimizing RMSE for each DOF rotation value (***Yi***) on each time/frame (*x*) to its corresponding point in Hermite (*p(t(x))*), remembering that Hermite curve definition is given under a parameter *t* and not on actual frame *x*.

Each tangent length is then equally divided in *n* samples between a minimum and maximum value and the Hermite segment's RMSE is calculated for each *n* times *n* curve possibilities. The one yielding the less RMSE is then used as the final value.

B-Spline curve fitting was based on the algorithm proposed by DIERCKX (1993). Given the definition of a B-Spline as in Equation A.14:

$$s(x) = \sum_{i=1}^{n+1} c_i.N_{i,k}(x)x_{min} < x < x_{max}, 2 \leq k \leq n+1$$

$c_i$ is the weight of each $N_{i,k}(x)$ basis function. To calculate each basis function it's necessary to establish a knot set $t_j$. Thus the algorithm aims to find these two vectors, $c_i$ and $t_j$, by guessing an initial estimate and doing a gradient search to find the lowest local value of the residue function $r$ that compares the spline value with the original mocap data:

$$r = min \sum_{i=1}^{n} \quad (y_i - s(x_i))^2$$

Finding the two vectors is achieved by first transforming the previous formula in the matrix form $r = min \mid\mid y - E.c \mid\mid^2$. This can be solved using the pseudo-inverse method in the equation $y = E.c$. This equation requires a known knot vector.

To solve the equation with unknown knot vectors DIERCKX (1993) proposes adding a new penalty function *p.P(t)* which helps unclutter knots, transforming the equation to:

$$v(t) = r(t) + p.P(t)$$

An initial knot estimation is required for the solving algorithm, and this is obtained by, starting with 3 equidistant knots in the interval, adding knots within intervals between knots that have the greatest error until a minimum error in the curve fitting equation is obtained.

The Fletcher and Reeves algorithm is then used with this initial knot vector. This is an iterative algorithm which, in broad view, solves the problem by:

1. Calculating an initial search direction, which is the gradient on *v(t)* for the initial knot estimation.
2. Minimize the equation *v(t)* but with the knot vector increased towards the gradient direction, creating a new knot vector in the direction searched.
3. If the approximation is within an error range defined by the user, then return the new knot vector.
4. At the end, Hermite and B-Spline curve fitting produced good results in terms of low RMSE. These were then used to concatenate Libras signs as explained in Chapter 3. Results obtained were then compared and further discussion is presented in chapters 4 and 5.

# APPENDIX B – TRANSITION CURVES RMSE COMPARISON

In this appendix we present the RMSE charts comparing RMSE between the four different curve models: linear, simple Hermite (sHermite), Hermite and B-Spline (spline). For each joint's attribute (i.e. shoulder rotation x, y and z; elbow rotation x; and wrist rotation x, y and z) it's presented its RMSE with the lowest highlighted in gray and the lowest average highlighted in blue.

We also present in red under transition 9 wrist rotation x a highlight in red indicating a value far greater than expected, and this was due to the use of a spline model that, when shortening the animation curve's length, created some abnormal behavior (only in 1 other animation curve this happened again).

It's presented here only the RMSE tables for 1 interpreter, the proposed method was also applied, and the results recorded and analyzed as explained in Chapter 5.

**Interpreter KA sentence 1 (transitions 1 to 6)**

| Animation Curve | | | linear | sHermite | Hermite | spline |
|---|---|---|---|---|---|---|
| TRANSITION 1 | *ka0101* | | | | | |
| transition selected: | *th04012* | | | | | |
| R_Humerus.rotateX | | *62.31* | 22.61 | 19.04 | 16.54 | 15.85 |
| R_Humerus.rotateY | | *78.16* | 8.14 | 3.97 | 4.15 | 11.88 |
| R_Humerus.rotateZ | | *7.00* | 28.64 | 28.47 | 13.55 | 42.21 |
| R_Elbow.rotateX | | *71.69* | 15.88 | 15.21 | 24.92 | 26.86 |
| R_Wrist.rotateX | | *90.89* | 30.78 | 19.37 | 30.90 | 28.98 |
| R_Wrist.rotateY | | *52.64* | 35.15 | 33.84 | 40.28 | 29.40 |
| R_Wrist.rotateZ | | *89.95* | 44.94 | 32.79 | 50.67 | 35.81 |
| **Average:** | | ***64.66*** | **26.59** | **21.81** | **25.86** | **27.28** |
| TRANSITION 2 | *ka0102* | | | | | |
| transition selected: | *ka01041* | | | | | |
| R_Humerus.rotateX | | *8.42* | 3.46 | 5.77 | 2.16 | 2.62 |
| R_Humerus.rotateY | | *33.67* | 8.03 | 4.21 | 9.04 | 12.13 |
| R_Humerus.rotateZ | | *8.53* | 2.38 | 2.50 | 0.80 | 8.63 |
| R_Elbow.rotateX | | *10.86* | 17.80 | 11.87 | 17.70 | 17.05 |
| R_Wrist.rotateX | | *64.40* | 19.21 | 8.88 | 20.08 | 17.63 |
| R_Wrist.rotateY | | *80.60* | 7.97 | 4.70 | 18.19 | 14.66 |
| R_Wrist.rotateZ | | *115.09* | 8.34 | 5.54 | 31.22 | 16.71 |
| **Average:** | | ***45.94*** | **9.60** | **6.21** | **14.17** | **12.78** |
| TRANSITION 3 | *ka0103* | | | | | |
| transition selected: | *th04021* | | | | | |
| R_Humerus.rotateX | | *30.66* | 4.12 | 4.49 | 1.92 | 3.84 |
| R_Humerus.rotateY | | *6.80* | 8.33 | 9.10 | 4.48 | 12.88 |
| R_Humerus.rotateZ | | *21.86* | 1.98 | 1.12 | 3.51 | 7.80 |
| R_Elbow.rotateX | | *24.17* | 4.27 | 3.22 | 3.30 | 4.60 |
| R_Wrist.rotateX | | *106.32* | 27.43 | 21.38 | 19.13 | 10.84 |
| R_Wrist.rotateY | | *58.19* | 18.68 | 19.16 | 17.02 | 13.50 |
| R_Wrist.rotateZ | | *44.65* | 27.54 | 24.99 | 16.69 | 9.28 |
| **Average:** | | ***41.81*** | **13.19** | **11.92** | **9.44** | **8.96** |
| TRANSITION 4 | *ka0104* | | | | | |
| transition selected: | *ka01071* | | | | | |
| R_Humerus.rotateX | | *16.97* | 2.53 | 2.21 | 1.87 | 1.73 |
| R_Humerus.rotateY | | *17.50* | 5.47 | 4.62 | 1.38 | 2.14 |
| R_Humerus.rotateZ | | *45.56* | 6.28 | 4.54 | 4.99 | 3.98 |
| R_Elbow.rotateX | | *5.62* | 2.34 | 0.78 | 3.25 | 2.85 |
| R_Wrist.rotateX | | *10.02* | 23.43 | 13.54 | 5.04 | 6.68 |
| R_Wrist.rotateY | | *32.25* | 18.99 | 13.91 | 5.14 | 5.94 |
| R_Wrist.rotateZ | | *109.35* | 21.68 | 15.85 | 13.86 | 15.53 |
| **Average:** | | ***33.89*** | **11.53** | **7.92** | **5.08** | **5.55** |
| TRANSITION 5 | *ka0105* | | | | | |
| transition selected: | *ro01091* | | | | | |
| R_Humerus.rotateX | | *2.60* | 6.17 | 4.41 | 7.52 | 7.58 |
| R_Humerus.rotateY | | *48.59* | 6.81 | 5.22 | 2.07 | 3.50 |
| R_Humerus.rotateZ | | *56.76* | 5.04 | 6.10 | 2.10 | 2.74 |
| R_Elbow.rotateX | | *2.10* | 3.46 | 1.98 | 1.47 | 2.26 |
| R_Wrist.rotateX | | *34.63* | 2.54 | 3.24 | 3.14 | 3.66 |
| R_Wrist.rotateY | | *47.47* | 1.25 | 2.02 | 1.49 | 1.93 |
| R_Wrist.rotateZ | | *16.53* | 3.77 | 2.88 | 5.89 | 5.66 |
| **Average:** | | ***29.81*** | **4.15** | **3.69** | **3.38** | **3.90** |
| TRANSITION 6 | *ka0106* | | | | | |
| transition selected: | *th01061* | | | | | |
| R_Humerus.rotateX | | *3.59* | 3.31 | 2.29 | 1.62 | 1.54 |
| R_Humerus.rotateY | | *13.44* | 6.45 | 9.96 | 7.83 | 7.08 |
| R_Humerus.rotateZ | | *13.11* | 1.73 | 3.75 | 1.54 | 1.46 |
| R_Elbow.rotateX | | *22.20* | 4.15 | 2.68 | 2.88 | 3.86 |
| R_Wrist.rotateX | | *46.34* | 8.80 | 6.54 | 6.47 | 6.28 |
| R_Wrist.rotateY | | *64.54* | 13.70 | 9.24 | 19.53 | 19.82 |
| R_Wrist.rotateZ | | *37.31* | 8.62 | 5.22 | 8.02 | 7.59 |
| **Average:** | | ***28.65*** | **6.68** | **5.67** | **6.84** | **6.80** |

**Interpreter KA sentence 1 (transitions 7 to 11)**

| | | | | | | |
|---|---|---|---|---|---|---|
| TRANSITION 7 | *ka0107* | | | | | |
| transition selected: | *ka01041* | | | | | |
| R_Humerus.rotateX | | *17.34* | 4.14 | 3.25 | 1.90 | 0.94 |
| R_Humerus.rotateY | | *3.25* | 4.37 | 3.53 | 3.47 | 2.59 |
| R_Humerus.rotateZ | | *34.12* | 6.42 | 4.73 | 1.69 | 3.32 |
| R_Elbow.rotateX | | *13.71* | 4.76 | 3.00 | 4.80 | 4.53 |
| R_Wrist.rotateX | | *24.34* | 25.18 | 16.11 | 23.74 | 24.53 |
| R_Wrist.rotateY | | *6.61* | 13.64 | 11.51 | 7.34 | 11.02 |
| R_Wrist.rotateZ | | *92.38* | 18.84 | 13.93 | 11.13 | 9.38 |
| **Average:** | | ***27.39*** | **11.05** | **8.01** | **7.72** | **8.05** |

| | | | | | | |
|---|---|---|---|---|---|---|
| TRANSITION 8 | *ka0108* | | | | | |
| transition selected: | *th01081* | | | | | |
| R_Humerus.rotateX | | *14.22* | 5.42 | 6.09 | 8.32 | 4.97 |
| R_Humerus.rotateY | | *3.37* | 4.53 | 3.90 | 5.26 | 2.87 |
| R_Humerus.rotateZ | | *41.26* | 5.42 | 16.02 | 14.98 | 4.91 |
| R_Elbow.rotateX | | *26.34* | 7.80 | 5.04 | 5.12 | 5.67 |
| R_Wrist.rotateX | | *68.36* | 12.19 | 63.69 | 10.05 | 10.28 |
| R_Wrist.rotateY | | *138.97* | 33.32 | 45.83 | 49.38 | 33.07 |
| R_Wrist.rotateZ | | *39.84* | 25.01 | 68.22 | 5.15 | 5.72 |
| **Average:** | | ***47.48*** | **13.39** | **29.83** | **14.04** | **9.64** |

| | | | | | | |
|---|---|---|---|---|---|---|
| TRANSITION 9 | *ka0109* | | | | | |
| transition selected: | *th01091* | | | | | |
| R_Humerus.rotateX | | *7.68* | 3.95 | 1.71 | 3.65 | 3.97 |
| R_Humerus.rotateY | | *11.54* | 10.61 | 10.08 | 9.97 | 10.33 |
| R_Humerus.rotateZ | | *7.92* | 2.60 | 3.00 | 3.99 | 2.78 |
| R_Elbow.rotateX | | *2.03* | 2.88 | 2.15 | 1.49 | 2.20 |
| R_Wrist.rotateX | | *9.04* | 2.72 | 1.53 | 1.72 | 1039.65 |
| R_Wrist.rotateY | | *56.51* | 28.45 | 24.40 | 20.76 | 25.59 |
| R_Wrist.rotateZ | | *47.50* | 15.95 | 16.09 | 16.03 | 12.89 |
| **Average:** | | ***20.32*** | **9.59** | **8.42** | **8.23** | **156.77** |

| | | | | | | |
|---|---|---|---|---|---|---|
| TRANSITION 10 | *ka0110* | | | | | |
| transition selected: | *ka03021* | | | | | |
| R_Humerus.rotateX | | *10.25* | 3.77 | 2.62 | 2.92 | 2.34 |
| R_Humerus.rotateY | | *27.02* | 3.18 | 3.32 | 2.36 | 2.65 |
| R_Humerus.rotateZ | | *11.06* | 0.77 | 0.71 | 0.35 | 0.29 |
| R_Elbow.rotateX | | *1.93* | 2.56 | 2.55 | 4.68 | 6.59 |
| R_Wrist.rotateX | | *139.13* | 29.35 | 35.37 | 38.01 | 37.62 |
| R_Wrist.rotateY | | *77.54* | 26.16 | 22.64 | 16.54 | 11.66 |
| R_Wrist.rotateZ | | *161.43* | 33.60 | 39.51 | 47.53 | 48.04 |
| **Average:** | | ***61.19*** | **14.20** | **15.25** | **16.05** | **15.60** |

| | | | | | | |
|---|---|---|---|---|---|---|
| TRANSITION 11 | *ka0111* | | | | | |
| transition selected: | *ka01031* | | | | | |
| R_Humerus.rotateX | | *10.45* | 8.92 | 7.64 | 11.67 | 11.98 |
| R_Humerus.rotateY | | *39.59* | 10.69 | 11.99 | 18.20 | 18.24 |
| R_Humerus.rotateZ | | *66.04* | 13.92 | 12.21 | 8.18 | 12.30 |
| R_Elbow.rotateX | | *109.52* | 18.09 | 14.33 | 20.16 | 20.80 |
| R_Wrist.rotateX | | *41.15* | 10.25 | 40.04 | 20.53 | 19.74 |
| R_Wrist.rotateY | | *91.41* | 33.16 | 16.43 | 25.81 | 23.34 |
| R_Wrist.rotateZ | | *32.69* | 11.54 | 34.98 | 25.14 | 23.23 |
| **Average:** | | ***55.83*** | **15.22** | **19.66** | **18.53** | **18.52** |

**Interpreter KA sentence 2**

| Animation Curve | | | linear | sHermite | Hermite | spline |
|---|---|---|---|---|---|---|
| TRANSITION 1 | *ka0201* | | | | | |
| transition selected: | *ka03011* | | | | | |
| R_Humerus.rotateX | | *62.31* | 1.25 | 2.65 | 6.08 | 8.00 |
| R_Humerus.rotateY | | *78.16* | 4.05 | 1.63 | 2.09 | 12.73 |
| R_Humerus.rotateZ | | *7.00* | 5.75 | 6.11 | 4.98 | 11.44 |
| R_Elbow.rotateX | | *71.69* | 17.44 | 13.57 | 13.10 | 15.29 |
| R_Wrist.rotateX | | *90.89* | 4.79 | 6.39 | 3.01 | 3.87 |
| R_Wrist.rotateY | | *52.64* | 11.48 | 9.64 | 5.76 | 5.78 |
| R_Wrist.rotateZ | | *89.95* | 5.85 | 2.95 | 3.56 | 4.39 |
| **Average:** | | *64.66* | **7.23** | **6.13** | **5.51** | **8.79** |
| | | | | | | |
| TRANSITION 2 | *ka0202* | | | | | |
| transition selected: | *th02021* | | | | | |
| R_Humerus.rotateX | | *8.42* | 5.61 | 5.18 | 3.66 | 9.83 |
| R_Humerus.rotateY | | *33.67* | 1.45 | 1.35 | 0.33 | 1.94 |
| R_Humerus.rotateZ | | *8.53* | 0.49 | 0.47 | 0.47 | 0.78 |
| R_Elbow.rotateX | | *10.86* | 1.59 | 1.27 | 1.59 | 1.45 |
| R_Wrist.rotateX | | *64.40* | 2.49 | 2.18 | 2.04 | 2.14 |
| R_Wrist.rotateY | | *80.60* | 4.15 | 4.33 | 3.80 | 3.82 |
| R_Wrist.rotateZ | | *115.09* | 3.49 | 3.89 | 2.66 | 3.21 |
| **Average:** | | *45.94* | **2.75** | **2.67** | **2.08** | **3.31** |
| | | | | | | |
| TRANSITION 3 | *ka0203* | | | | | |
| transition selected: | *th01071* | | | | | |
| R_Humerus.rotateX | | *30.66* | 5.24 | 4.94 | 3.16 | 3.97 |
| R_Humerus.rotateY | | *6.80* | 16.54 | 15.86 | 11.28 | 12.17 |
| R_Humerus.rotateZ | | *21.86* | 1.57 | 1.51 | 3.36 | 2.79 |
| R_Elbow.rotateX | | *24.17* | 1.57 | 1.49 | 1.37 | 1.51 |
| R_Wrist.rotateX | | *106.32* | 18.31 | 18.09 | 19.71 | 23.25 |
| R_Wrist.rotateY | | *58.19* | 3.89 | 2.79 | 12.26 | 10.32 |
| R_Wrist.rotateZ | | *44.65* | 10.21 | 9.29 | 9.59 | 15.42 |
| **Average:** | | *41.81* | **8.19** | **7.71** | **8.68** | **9.92** |
| | | | | | | |
| TRANSITION 4 | *ka0204* | | | | | |
| transition selected: | *th01031* | | | | | |
| R_Humerus.rotateX | | *16.97* | 1.87 | 1.15 | 0.87 | 0.90 |
| R_Humerus.rotateY | | *17.50* | 8.03 | 7.45 | 4.93 | 5.14 |
| R_Humerus.rotateZ | | *45.56* | 2.49 | 2.79 | 2.73 | 1.81 |
| R_Elbow.rotateX | | *5.62* | 4.52 | 5.21 | 6.01 | 6.50 |
| R_Wrist.rotateX | | *10.02* | 10.77 | 11.08 | 11.42 | 9.45 |
| R_Wrist.rotateY | | *32.25* | 18.58 | 20.10 | 21.73 | 15.03 |
| R_Wrist.rotateZ | | *109.35* | 25.79 | 25.97 | 27.90 | 22.90 |
| **Average:** | | *33.89* | **10.29** | **10.54** | **10.80** | **8.82** |
| | | | | | | |
| TRANSITION 5 | *ka0205* | | | | | |
| transition selected: | *th01041* | | | | | |
| R_Humerus.rotateX | | *2.60* | 1.55 | 1.03 | 3.87 | 2.71 |
| R_Humerus.rotateY | | *48.59* | 0.81 | 0.45 | 1.54 | 1.86 |
| R_Humerus.rotateZ | | *56.76* | 1.30 | 1.03 | 1.16 | 1.38 |
| R_Elbow.rotateX | | *2.10* | 4.40 | 4.10 | 2.71 | 3.38 |
| R_Wrist.rotateX | | *34.63* | 5.88 | 6.19 | 3.28 | 6.22 |
| R_Wrist.rotateY | | *47.47* | 3.99 | 3.92 | 3.88 | 4.32 |
| R_Wrist.rotateZ | | *16.53* | 18.59 | 18.13 | 10.21 | 13.72 |
| **Average:** | | *29.81* | **5.22** | **4.98** | **3.81** | **4.80** |
| | | | | | | |
| TRANSITION 6 | *ka0206* | | | | | |
| transition selected: | *th02061* | | | | | |
| R_Humerus.rotateX | | *3.59* | 8.32 | 7.43 | 8.06 | 6.34 |
| R_Humerus.rotateY | | *13.44* | 4.47 | 5.18 | 2.50 | 1.62 |
| R_Humerus.rotateZ | | *13.11* | 2.28 | 2.65 | 1.02 | 1.09 |
| R_Elbow.rotateX | | *22.20* | 6.33 | 1.89 | 0.63 | 7.17 |
| R_Wrist.rotateX | | *46.34* | 8.32 | 8.83 | 10.57 | 10.56 |
| R_Wrist.rotateY | | *64.54* | 11.46 | 10.38 | 17.79 | 19.30 |
| R_Wrist.rotateZ | | *37.31* | 19.85 | 15.70 | 9.89 | 11.08 |
| **Average:** | | *28.65* | **8.72** | **7.44** | **7.21** | **8.16** |

**Interpreter KA sentence 3**

| Animation Curve | | | linear | sHermite | Hermite | spline |
|---|---|---|---|---|---|---|
| TRANSITION 1 | *ka0301* | | | | | |
| transition selected: | *ka02011* | | | | | |
| R_Humerus.rotateX | | *47.13* | 8.88 | 6.72 | 9.06 | 9.08 |
| R_Humerus.rotateY | | *56.28* | 10.47 | 11.17 | 7.53 | 5.87 |
| R_Humerus.rotateZ | | *10.74* | 17.67 | 16.13 | 11.13 | 11.60 |
| R_Elbow.rotateX | | *78.14* | 5.95 | 5.12 | 2.36 | 4.77 |
| R_Wrist.rotateX | | *23.21* | 6.34 | 4.55 | 7.28 | 7.23 |
| R_Wrist.rotateY | | *83.68* | 14.36 | 13.68 | 11.81 | 11.87 |
| R_Wrist.rotateZ | | *60.29* | 7.95 | 8.85 | 5.94 | 11.36 |
| **Average:** | | *51.35* | **10.23** | **9.46** | **7.87** | **8.83** |
| | | | | | | |
| TRANSITION 2 | *ka0302* | | | | | |
| transition selected: | *ka01101* | | | | | |
| R_Humerus.rotateX | | *0.54* | 0.57 | 0.92 | 2.50 | 3.24 |
| R_Humerus.rotateY | | *5.94* | 2.42 | 1.02 | 1.16 | 2.02 |
| R_Humerus.rotateZ | | *9.22* | 3.29 | 1.38 | 0.44 | 2.76 |
| R_Elbow.rotateX | | *22.31* | 3.48 | 2.90 | 2.52 | 1.79 |
| R_Wrist.rotateX | | *99.28* | 18.31 | 9.34 | 14.67 | 21.44 |
| R_Wrist.rotateY | | *60.59* | 25.58 | 19.28 | 13.31 | 8.62 |
| R_Wrist.rotateZ | | *133.59* | 21.88 | 13.59 | 33.80 | 20.97 |
| **Average:** | | *47.35* | **10.79** | **6.92** | **9.77** | **8.69** |
| | | | | | | |
| TRANSITION 3 | *ka0303* | | | | | |
| transition selected: | *th02041* | | | | | |
| R_Humerus.rotateX | | *9.11* | 3.11 | 3.63 | 1.90 | 6.29 |
| R_Humerus.rotateY | | *5.61* | 14.46 | 13.49 | 7.76 | 7.92 |
| R_Humerus.rotateZ | | *11.19* | 4.17 | 6.70 | 2.26 | 2.75 |
| R_Elbow.rotateX | | *25.90* | 5.41 | 3.79 | 3.53 | 1.70 |
| R_Wrist.rotateX | | *45.90* | 18.79 | 7.23 | 9.29 | 10.62 |
| R_Wrist.rotateY | | *58.33* | 18.10 | 13.81 | 16.53 | 13.64 |
| R_Wrist.rotateZ | | *77.75* | 27.83 | 11.46 | 14.86 | 7.46 |
| **Average:** | | *33.40* | **13.12** | **8.59** | **8.02** | **7.19** |
| | | | | | | |
| TRANSITION 4 | *ka0304* | | | | | |
| transition selected: | *th01101* | | | | | |
| R_Humerus.rotateX | | *4.48* | 2.11 | 3.39 | 1.19 | 1.06 |
| R_Humerus.rotateY | | *1.98* | 0.91 | 0.75 | 1.88 | 2.44 |
| R_Humerus.rotateZ | | *12.48* | 2.17 | 1.77 | 1.45 | 0.59 |
| R_Elbow.rotateX | | *2.93* | 4.54 | 3.52 | 3.51 | 7.18 |
| R_Wrist.rotateX | | *100.21* | 35.92 | 28.62 | 36.56 | 40.58 |
| R_Wrist.rotateY | | *47.74* | 21.67 | 17.53 | 10.31 | 16.17 |
| R_Wrist.rotateZ | | *125.00* | 35.52 | 27.31 | 37.02 | 43.92 |
| **Average:** | | *42.12* | **14.69** | **11.84** | **13.13** | **15.99** |
| | | | | | | |
| TRANSITION 5 | *ka0305* | | | | | |
| transition selected: | *da04041* | | | | | |
| R_Humerus.rotateX | | *13.79* | 2.51 | 1.52 | 0.56 | 1.15 |
| R_Humerus.rotateY | | *6.96* | 8.03 | 6.15 | 3.36 | 6.86 |
| R_Humerus.rotateZ | | *4.39* | 3.05 | 2.04 | 1.66 | 4.29 |
| R_Elbow.rotateX | | *21.36* | 3.26 | 2.95 | 5.02 | 4.66 |
| R_Wrist.rotateX | | *0.93* | 31.04 | 10.01 | 17.91 | 15.31 |
| R_Wrist.rotateY | | *135.29* | 16.61 | 18.94 | 7.92 | 10.23 |
| R_Wrist.rotateZ | | *74.04* | 15.87 | 10.53 | 7.54 | 7.75 |
| **Average:** | | *36.68* | **11.48** | **7.45** | **6.28** | **7.18** |
| | | | | | | |
| TRANSITION 6 | *ka0306* | | | | | |
| transition selected: | *th03061* | | | | | |
| R_Humerus.rotateX | | *39.49* | 2.05 | 1.70 | 3.55 | 5.00 |
| R_Humerus.rotateY | | *36.33* | 9.08 | 7.59 | 13.10 | 11.61 |
| R_Humerus.rotateZ | | *7.05* | 5.00 | 5.81 | 4.74 | 5.37 |
| R_Elbow.rotateX | | *96.88* | 6.81 | 6.23 | 17.12 | 18.15 |
| R_Wrist.rotateX | | *47.77* | 17.99 | 4.59 | 15.11 | 14.93 |
| R_Wrist.rotateY | | *53.48* | 16.13 | 12.53 | 9.86 | 16.50 |
| R_Wrist.rotateZ | | *8.28* | 9.48 | 3.93 | 6.20 | 5.99 |
| **Average:** | | *41.33* | **9.51** | **6.05** | **9.96** | **11.08** |

**Interpreter KA sentence 4**

| Animation Curve | | | linear | sHermite | Hermite | spline |
|---|---|---|---|---|---|---|
| TRANSITION 1 | *ka0401* | | | | | |
| transition selected: | *ro04011* | | | | | |
| R_Humerus.rotateX | | *62.69* | 17.21 | 20.38 | 17.42 | 21.06 |
| R_Humerus.rotateY | | *35.70* | 12.81 | 7.07 | 12.01 | 10.26 |
| R_Humerus.rotateZ | | *53.43* | 25.61 | 27.80 | 28.26 | 26.17 |
| R_Elbow.rotateX | | *54.53* | 16.91 | 8.76 | 2.54 | 6.93 |
| R_Wrist.rotateX | | *107.54* | 28.18 | 30.73 | 30.41 | 25.63 |
| R_Wrist.rotateY | | *19.00* | 6.07 | 8.55 | 6.78 | 13.33 |
| R_Wrist.rotateZ | | *104.19* | 30.23 | 29.80 | 30.29 | 29.84 |
| **Average:** | | ***62.44*** | **19.57** | **19.01** | **18.24** | **19.03** |
| TRANSITION 2 | *ka0402* | | | | | |
| transition selected: | *th04021* | | | | | |
| R_Humerus.rotateX | | *40.94* | 9.66 | 9.26 | 7.34 | 5.33 |
| R_Humerus.rotateY | | *31.37* | 19.02 | 17.15 | 11.30 | 12.69 |
| R_Humerus.rotateZ | | *14.46* | 4.87 | 7.07 | 6.78 | 7.05 |
| R_Elbow.rotateX | | *16.91* | 5.16 | 3.43 | 1.86 | 1.92 |
| R_Wrist.rotateX | | *71.95* | 19.36 | 17.67 | 7.26 | 7.30 |
| R_Wrist.rotateY | | *36.45* | 8.88 | 8.32 | 2.59 | 2.44 |
| R_Wrist.rotateZ | | *70.06* | 23.09 | 22.86 | 12.44 | 12.01 |
| **Average:** | | ***40.31*** | **12.86** | **12.25** | **7.08** | **6.96** |
| TRANSITION 3 | *ka0403* | | | | | |
| transition selected: | *th01041* | | | | | |
| R_Humerus.rotateX | | *15.09* | 1.71 | 1.81 | 1.28 | 1.44 |
| R_Humerus.rotateY | | *8.83* | 3.02 | 2.77 | 2.55 | 2.47 |
| R_Humerus.rotateZ | | *1.37* | 2.33 | 2.75 | 2.32 | 2.40 |
| R_Elbow.rotateX | | *3.63* | 3.83 | 4.10 | 4.92 | 4.56 |
| R_Wrist.rotateX | | *25.71* | 2.46 | 1.86 | 0.75 | 5.05 |
| R_Wrist.rotateY | | *27.54* | 8.35 | 4.08 | 4.60 | 4.26 |
| R_Wrist.rotateZ | | *44.34* | 7.44 | 6.71 | 10.83 | 9.81 |
| **Average:** | | ***18.07*** | **4.16** | **3.44** | **3.89** | **4.28** |
| TRANSITION 4 | *ka0404* | | | | | |
| transition selected: | *da01071* | | | | | |
| R_Humerus.rotateX | | *5.48* | 9.37 | 6.67 | 13.06 | 12.74 |
| R_Humerus.rotateY | | *2.77* | 4.80 | 5.19 | 3.94 | 2.80 |
| R_Humerus.rotateZ | | *14.24* | 5.19 | 4.24 | 1.90 | 1.86 |
| R_Elbow.rotateX | | *29.57* | 13.63 | 9.31 | 5.24 | 8.32 |
| R_Wrist.rotateX | | *171.19* | 10.77 | 21.91 | 34.15 | 32.83 |
| R_Wrist.rotateY | | *91.62* | 18.74 | 11.57 | 31.56 | 31.12 |
| R_Wrist.rotateZ | | *153.48* | 15.05 | 27.34 | 62.00 | 59.46 |
| **Average:** | | ***66.91*** | **11.08** | **12.32** | **21.69** | **21.30** |
| TRANSITION 5 | *ka0405* | | | | | |
| transition selected: | *ka04061* | | | | | |
| R_Humerus.rotateX | | *12.37* | 8.79 | 8.95 | 5.67 | 4.54 |
| R_Humerus.rotateY | | *7.93* | 7.24 | 7.94 | 9.78 | 7.59 |
| R_Humerus.rotateZ | | *32.05* | 5.73 | 6.48 | 7.25 | 8.74 |
| R_Elbow.rotateX | | *6.89* | 2.62 | 1.88 | 0.54 | 3.71 |
| R_Wrist.rotateX | | *44.91* | 4.93 | 7.33 | 8.79 | 14.30 |
| R_Wrist.rotateY | | *107.80* | 29.14 | 23.94 | 31.99 | 42.22 |
| R_Wrist.rotateZ | | *74.93* | 16.08 | 9.75 | 4.69 | 28.69 |
| **Average:** | | ***40.98*** | **10.65** | **9.47** | **9.82** | **15.69** |
| TRANSITION 6 | *ka0406* | | | | | |
| transition selected: | *ka04051* | | | | | |
| R_Humerus.rotateX | | *29.67* | 3.97 | 4.64 | 6.30 | 6.05 |
| R_Humerus.rotateY | | *36.70* | 5.44 | 4.96 | 14.03 | 13.48 |
| R_Humerus.rotateZ | | *8.38* | 3.57 | 2.28 | 10.46 | 10.96 |
| R_Elbow.rotateX | | *86.28* | 5.23 | 1.48 | 2.02 | 6.17 |
| R_Wrist.rotateX | | *7.51* | 6.68 | 1.03 | 6.99 | 7.55 |
| R_Wrist.rotateY | | *55.85* | 8.25 | 4.63 | 30.86 | 32.72 |
| R_Wrist.rotateZ | | *51.13* | 10.46 | 3.00 | 24.50 | 30.72 |
| **Average:** | | ***39.36*** | **6.23** | **3.14** | **13.60** | **15.38** |

## APPENDIX C - CONSENT TO USE OF IMAGE AND CAPTURED MATERIAL

Pelo presente instrumento particular, eu, ________________________________________________________________________,
de nacionalidade __________________, R.G. n.________________________, profissão _________________ e residente à ________________________________________________________________________
_________________ na cidade de ______________ - ___ **AUTORIZO** de forma gratuita e sem qualquer ônus, ao pesquisador Leandro Martin Guertzenstein Angare, a captação de materiais gravados (fotos e/ou vídeos) e capturados utilizando técnicas de captura de movimento (*motion capture*) para o projeto de mestrado (CNPq Processo 160941/2013-0) da Faculdade de Engenharia Elétrica e de Computação (FEEC) da Universidade Estadual de Campinas (UNICAMP). Ao mesmo tempo libero a utilização de tais materiais para artigos, palestras, congressos, paineis e outros materiais para fins científicos e educacionais em território nacional e no exterior.

Campinas, __ de ________ de 2014.

______________________________________
Declarante

______________________________________
Pesquisador responsável pelo projeto

# APPENDIX D – ANIMATION

In this appendix we present a more in-depth exploration into the topics raised in Chapter 2. This appendix is divided into 2 sections: Section D.1 which discuss the foundations of terms in traditional animation and Section D.2 with a more focused discussion on the computer animation field.

## D.1 ANIMATION

Animation is the illusion of life, quoting the title of the illustrious book by THOMAS and JOHNSTON (1995). Moreover, Norman McLaren, a famous Canadian animator, stated that "animation is not the art of drawings that move but the art of movements that are drawn" (SÁENZ VALIENTE, 2011, p. 30). In a broader sense, animation is change, variation of an attribute in time; this attribute could be a character's pose resulting in movement, or could be a blush in such character's face, resulting in an apparent shy reaction to the audience. For the purpose of this research, animation will be restricted to change in position and orientation, i.e. movement, although the principles and ideas presented in the text could apply to other types of changes. In our context, an animator is the professional who is tasked with imbuing life in materials as varied as sand, puppets, paper drawings or computer-generated images, that is perceived as (illusion of) movement.

This section explores some significant aspects of animation history and techniques, focusing on how the illusion of movement was developed and what is the usual process to create it, relating these topics with the developments described in this dissertation. For a more comprehensive study on this we refer to the works of BENDAZZI (1995) and BECK (2004).

It is considered that movement representation dates as far back as the prehistoric cave painting of Altamira where a bison is depicted with eight legs (MACHADO, 2007). Supposedly the intent of the painter was to represent in a single image a running movement. Since then, throughout millenniums, Men has tried to devise a way to represent, register, and visually transmit the sensation of movement, but real breakthroughs began with the invention of optical toys:

> [...] in 1825, a famous English physician, John A. Paris, created the prototype of optical toys, the thaumatrope: a disc with a complementary image on each side, and strings attached at each end of its horizontal axis. When the disc is spun on the strings, the two complementary images appear to merge. In 1832, the Belgian scientist Joseph Plateau invented the prototype of optical toys, the phenakistiscope. This was a device made of a pivot and a cardboard disc, along the edges of which successive images of an object in motion had been drawn. By focusing on one drawing while viewing the rotating disk through a slit, one had the illusion that the drawing moved. (BENDAZZI, 1995, p. 3)

These two optical toys illustrate fundamental animation and cinematic concepts: persistence of vision and phi phenomenon. Persistence of vision theory states that when light reaches the eyes the image it creates persists for some moment in the retina. This can be seen in the classic thaumatrope with one side of a disc a drawing of a bird and the other a cage, when this disc is spun it merges both images because the speed by which images change is fast enough so the bird image persists within the retina when the new image (cage) is presented, thus creating the illusion of a trapped bird (Figure D.1). Max Wertheimer developed this theory further, stating that we perceive continuous motion when shown fast successions of static movement images (frames). He named it the Phi phenomenon and MACHADO explains it as so:

> Today we could say that the eye, as a rule, does not distinguish between a directly perceived movement and an apparent, artificial or mechanically produced movement, as it was demonstrated by Wertheimer in his pioneer study about the phi phenomenon. (MACHADO, 2007, p. 22)

Persistence of vision and the phi phenomenon aren't sufficient to explain the illusion of movement we have in a film. Imagine a multifaceted object such as a cube, where in each face there is a depiction of an instant (or frame) of a jump. If the multifaceted object was spun on its axis there would be no movement perceived, only a blur of probably indistinguishable forms. A pause is needed between images. That is what the slit on the phenakistoscope (Figure D.2) is for. When strobes of images displaying a minimal variation of a stimulus are shown to a human brain, those stimuli are merged together not only in space as in the thaumatrope but also in time, creating the illusion of movement. That is the Beta phenomenon.

It is noteworthy to digress on the time discrete nature of movement. Although for over a century the predominant form of space representation for images was continuous in the form of analog film stock frames, time representation was discrete since its conception. In computer animation, both space and time are discrete in their final results but are stored and manipulated as continuous parameters.

**Figure D. 1. Thaumatrope with drawing of bird and cage. When fast flipped the two images are perceived as one due to the Phi effect.**

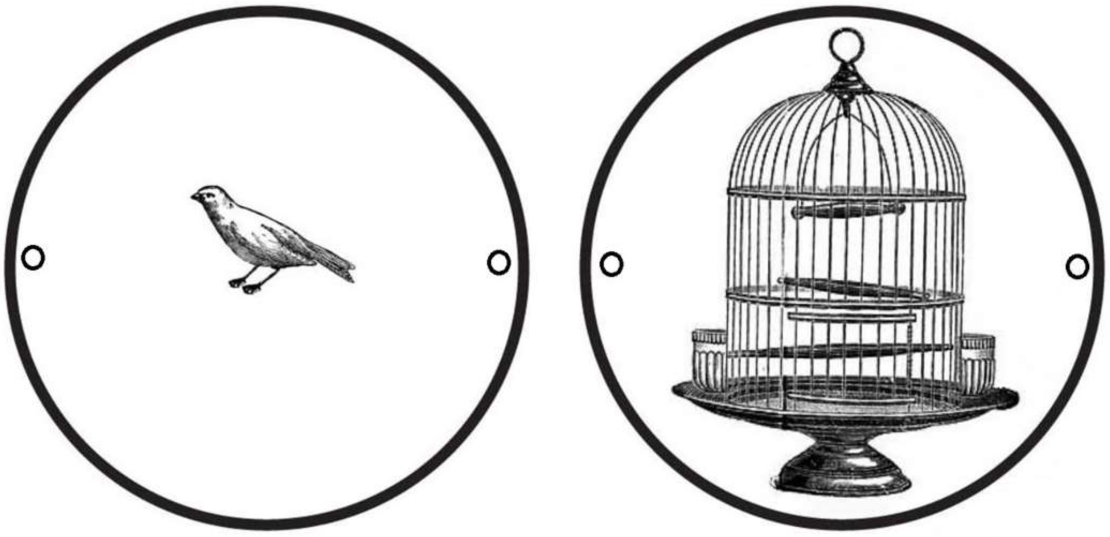


Image from U. S. NATIONAL PARK SERVICE (2022).

**Figure D. 2. Phenakistoscope with a horse galloping. When spun and viewed through the slits in front of a mirror the various images of a horse gallop are perceived by our brain as one continuous movement.**

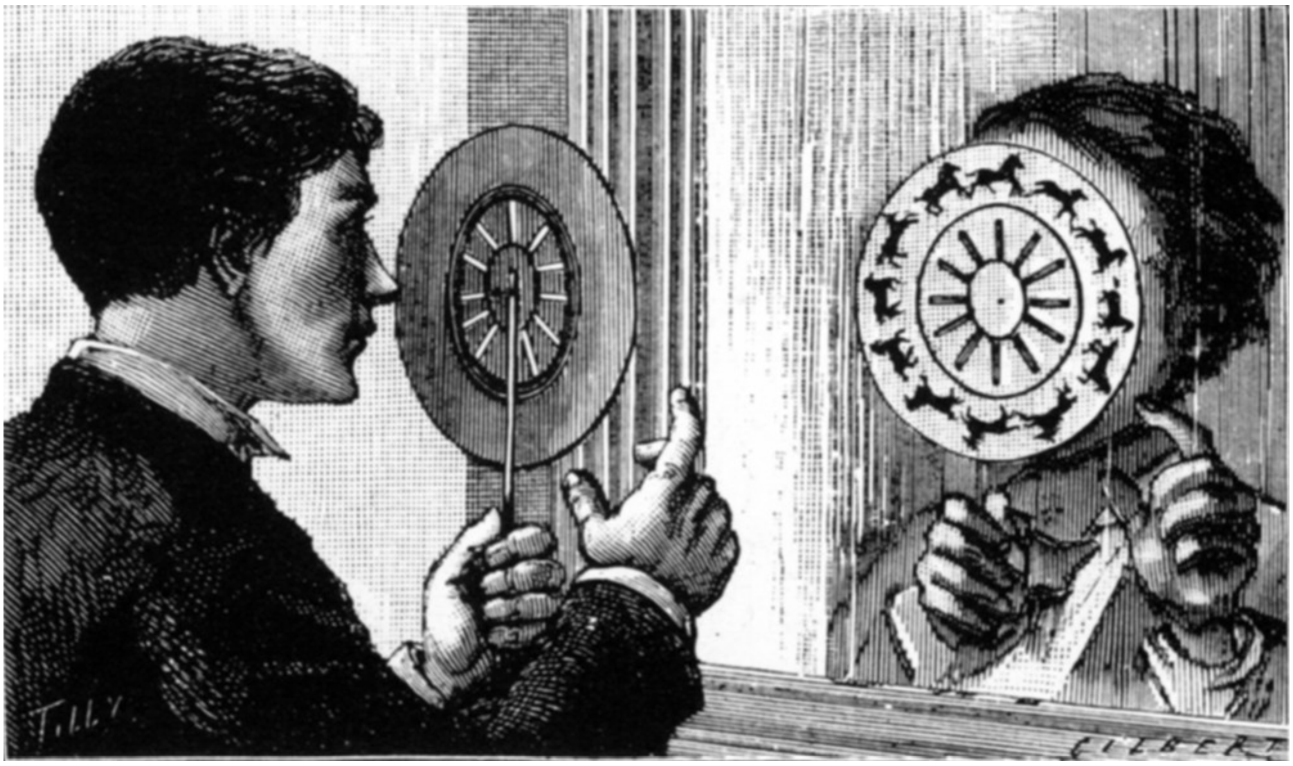
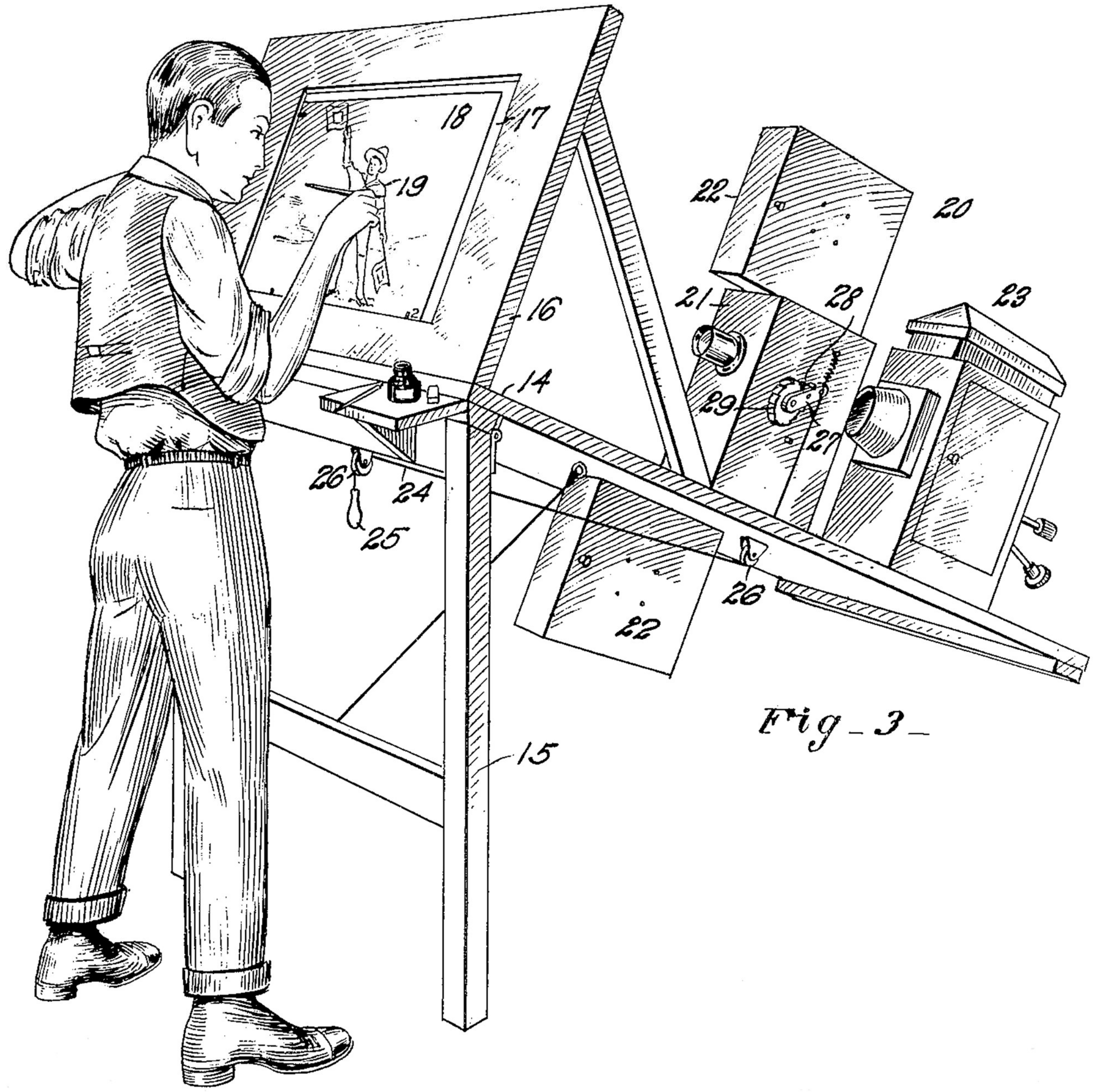


Image from JOBSON, C. (2015).

Movement reproduction and perception were solid concepts by the birth of the 20th century and new animation tools were developed almost yearly, as is the case of the 1910 Earl Hurd invention of celluloid animation which originated the layer system used in image and video editing software. Artistic aspects of animation were also thriving with studies of human and animal motion, and one invention made a remarkable contribution to generating natural movement for artificial characters. In 1917, Max Fleischer, a renowned animator who contributed to series such as *Out of the Inkwell* and *Superman*, under the patent US 1242674 A, disclosed to the world the Rotoscope (Figure D.3).

> An object of my invention is to provide a method by which improved cartoon films may be produced, depicting the figures or other objects in a life-like manner, characteristic of the regular animated photo pictures.
>
> In producing cartoon films by my improved method, scenes are enacted by the aid of living actors depicting the subjects to be displayed by the cartoons, and, through the instrumentality of a moving picture camera, pictures of the enacted scenes are taken, and from these pictures, line pictures or cartoons of the characters or objects to be portrayed are made. The series of cartoons are then photographically reproduced on a film or equivalent medium, and the photographs of the cartoons thus obtained are projected on a screen and displayed in the usual manner by any approved moving picture machine. (FLEISCHER, 1917)

**Figure D. 3. Rotoscope. Max Fleischer's diagram for the US patent.**



Public domain image.

Rotoscoping has been extensively used since its invention. The goal, to achieve life-like movement to artificial characters, was so successful that even Disney studios used rotoscoping to create complicated movements such as the dance sequence in *Snow White and the Seven Dwarfs*. A modern version of rotoscoping is the motion capture system, which also allows life-like movement for virtual characters using real actors as basis. This life-like movement is particularly needed when dealing with sign language, since small modifications in a sign might change its meaning. Motion capture of signing interpreters is explained in

Section 3.2.

Animated film-making has also undergone significant changes, going from a “one man does all craft” to a production line industry with specialized jobs involving technicians and artists. Although there isn't only a single animation studio organization, there are some key elements common to most of them, such as a director, producer, writer, storyboard artist, and animator. According to SHAMUS, C. 1990 and THOMAS, F et JOHNSTON, O. 1995, in a studio that still uses traditional cel animation, there are three professionals (Figure D.4) directly involved in the creation of movement:

- The Animator, responsible for creating the main or key poses of the character (also called *keyframes*). The animator is a highly experienced professional, skilled in acting theory and draftsmanship. The animator defines ***what*** a character does by creating the key moments of a movement. For instance, to outline a character reaching for a glass of water on top of a table, the animator can define a first key pose of the arm near the torso and a final key pose of the arm grasping the cup.
- The Assistant Animator or Break Down Man, working under the animator's supervision, is responsible for roughing out the poses made by the animator and creating the breakdown poses between key poses to help the inbetweener's work (explained below). The assistant animator defines ***how*** a character acts in its action. For instance, in the previous reaching example, there are several arcs the hand can perform, it could go up as in a very aggressive manner or could go in a downward arc with a more sneaky acting.
- The Inbetweener works under the assistant animator's and the animator's supervisions. Usually, the Inbetweener is not very experienced in animation although drawing skills are necessary. The inbetweening executed by this professional is a repetitive and laborious activity aimed at creating the drawings between the key and break down poses made by the animator and assistant animator, respectively. The Inbetweener does not need to have a deep understanding of the movement since this movement has been created by the animator and further refined with the extreme poses by the assistant animator.

**Figure D. 4. Animation department diagram. The animation department (central blocks) receives layout, which is the background and the different layers and rough positions of characters, from the Layout Deparment (left block) and delivers the drawn frames for the movie for the Ink and Paint Department (right block) which is responsible for transferring the drawings to cel and coloring them. Inside the animation department, we can see the different animation professionals (animators, assistant animator and inbetweener) and what they are responsible for (key poses, break down poses, and in-betweens, respectively).**

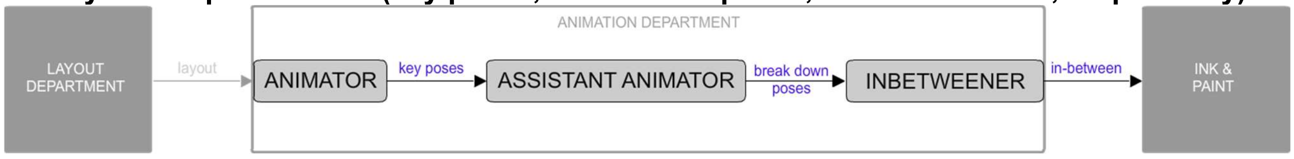


Image from the author.

Computers aid in animation in almost every aspect of the endeavor, substituting typewriters for script writers to composing the final movie in what would be a very expensive analog film compositing and revelation process. For movement creation, most pipelines rely solely on animators for character posing, but inbetweening can be easily automatically executed by a computer, especially in 3D computer animation, as the activity can be reduced to a mathematical interpolation process. The Assistant Animator's task still needs human interference, since most of the time the associated activity requires previous experience not available in any simple mathematical interpolation. The method proposed in this document intends to replace the assistant animator’s activity by extending interpolation algorithms with complex curves learned from motion capture data.

## D.2 COMPUTER ANIMATION

To understand where this research fits into the development of a computer animation avatar we will go over some of the basic principles and methodologies in the area.

As with many other human activities, computers play nowadays an important role in animation as a tool not only for decreasing production time and cost but also, as a new media, providing artists with a new way of exploring artificial movement as pointed by DARLEY (2011) in his analysis of the Pixar short movie *Red's Dream* (1987).

In his analysis, Andy Darley states that:

> [T]he new technique of digital imaging does not produce these new forms of image [...] all by itself. What is being attempted with this new means involves particular kinds of contact with already established aesthetic conventions and forms. (DARLEY, 2011).

It is interesting to note that the same studio, Pixar, has released a short animated movie (*Piper*, 2016) in which the same "continued refinement of a particular tradition of cinematic realism" (DARLEY, 2011) can be seen. A somewhat opposite view is defended in MANOVICH (1991), wherein the author portrays the role of a "professional designer/animator" in terms of their compliance with realistic renderings of images and movements.

Computer Animation history is more than 60 years old with its first development being done by research laboratories and individuals in the 1960s, and it has evolved from rudimentary black and white renders of a hand (CATMULL et PARKE, 1972) to the almost indistinguishable to life images popular in today's movies.

History and aesthetics of computer animation are extensive and interesting topics but will not be further explored in this text for the sake of brevity and focus on the project's subject of animation synthesis. Instead, this section explores the technical aspects of animation used as base to this project, focusing on the basic of computer animation pipeline to virtual agents and the different methods of animation synthesis. We will also refer to computer animation herein as three-dimensional computer-generated motion, excluding two-dimensional animations and manipulations of other attributes not related to the deformation of a virtual model by means of translation or rotation of internal controllers, i.e. joints of a virtual skeleton.

A traditional computer animation project consists of three main stages: Pre-Production, Production, and Post-Production (BEAN, 2012), each stage dealing with different issues and preparing the necessary material for the next stage.

In the Pre-Production stage, the story and the visual elements (scripts, storyboards, characters, locations, and prop designs) are created, these elements are drawings and writings which draft the main concepts of the animation guide the production in later stages. This stage usually does not yet involve any final product that would be seen by users or audiences, but instead focuses on developing ideas so that the next stage will have a clear path for the development.

In the Production stage, the bulk of work is done. This involves the creation of virtual characters, scenarios, and animations and rendering the 3D information into 2D images. Usually, the characters are polygonal meshes with textures applied to them to give greater detail and realism associated with a virtual skeleton and control system, known as rig, that enables the animation of such characters. Also, combining the animated characters and scenarios, artists use virtual lights to render each shot and create sequences of 2D images.

In the Post-Production stage, sequences of 2D images are edited together, combined with audio and enhanced with special effects and color correction to generate the final movie.

When dealing with interactive 3D graphics and not movie production the post-production stage is usually non-existent or limited to adjustments in camera placement and on-camera effects, since the final product is real-time generation of an animation controlled by the user. For the generation of interactive avatars, one usually does not work with scripts and, therefore, the pre-production stage is predominantly focused on developing the visual for the models. Production, therefore, concentrates almost the entirety of the workload.

Usually, the production stage is subdivided into sub-stages, called departments by PARENT (2012), shown in Figure D.5. BEAN (2012) proposes a more streamlined linear pipeline comprised of the following sub-stages: Layout, Research & Development, Modeling, Texturing, Rigging/Setup, Animation, 3D Visual Effects, Lighting Rendering. Animation pipelines are not fixed and usually change from company to company and even from project to project but, as it is possible to see with the examples provided in PARENT (2012) and BEAN (2012), there are some sub-stages that recur and are important for this dissertation. Below, we elaborate further on these sub-stages and show some of them in our avatar (Figure D.6).

**Figure D. 5. 3D Computer Animation Pipeline.**

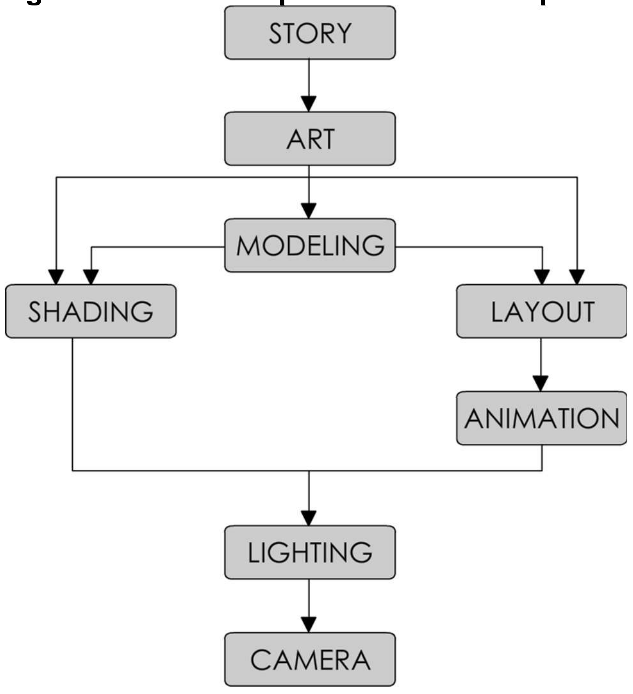
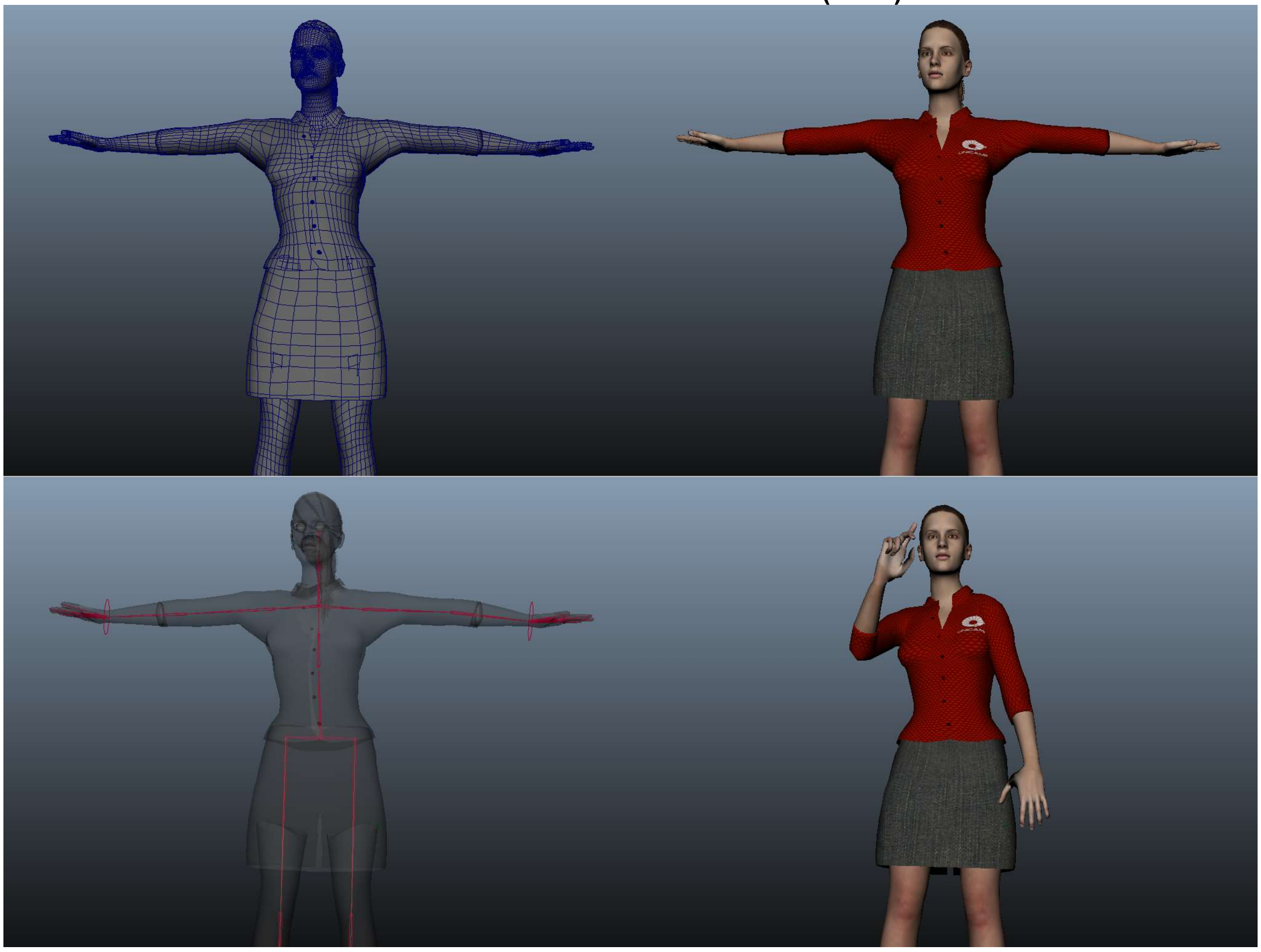


Image from PARENT (2012)

- Modeling: This is where the shape of virtual objects is created. These virtual objects are usually comprised of thousands of three-dimensional polygons. The model, also called "mesh", is used to represent the shape of the object in a 3D environment. There are several approaches to represent the geometry of virtual objects, such as polygons and NURBS (Non-Uniform Rational B-Splines), but the former is much more usual, especially because most rendering engines (both real-time and offline) work exclusively with polygons. A 3D polygonal mesh can be defined as a series of vertices, which are points with 3 coordinates (x, y, z), usually represented as a vector in the form $v = [x, y, z]$. A mesh consists of a series of vertices, $v_i$ that are linked together by straight lines called edges, *e*. Each vertex can be associated with one or more edges, but edges have only two vertices, one in its beginning, $v_0$, and one in its end, $v_1$, and can be represented as $e = [v_0, v_1]$. Both are usually not rendered into the final image. What is visible for the final user/audience are polygons, *f*, which are defined by a close loop of edges. Since edges are well defined as 2 consecutive vertices, a polygon is usually just represented as a vector of *n* vertices, and the edges are inferred by 2 consecutive vertices as such $f = [v_0, v_1, \ldots, v_n]$. This vector also presents an order by which the loop is closed, and determines the polygon's normal, which is a 3-dimensional vector perpendicular to the polygon that is used to control the interaction of the polygon with virtual light rays. Operations on 3D meshes are usually done on vertices, and their changes are passed on to the respective edges and polygons. Although in theory each vertex of each polygon can be animated, this process would demand a lot of time from animators.

- Texturing (Shading): Textures are usually applied to the model's mesh to create a variety of effects. In general, textures are two-dimensional images that can contain color information (such as the shades of a character's skin); three-dimensional deformations known as bumps, normal or displacement maps (such as the fine details on a character's face as pores and wrinkles); reflection maps (which can be seen as the different oily parts of a human face, usually making a forehead more shiny/reflective than the rest); and many other attributes.
- Rigging / Setup: Since it is not feasible to animate each vertex, a virtual skeleton is created mimicking a real human skeleton in this stage. It can be a straightforward skeleton (which is the case in this project), that consists of only virtual joints connected to the polygonal mesh, or can be a more complicated structure simulating muscles and tendons. In this sub-stage controls are also added to facilitate the animator's job, providing an intuitive interface.
- Animation: In this stage, the movement is created. The virtual skeleton has, for each of its joints, different attributes that can be transformed, usually the joint's rotations on x, y, and z angles. Each of these attributes is denominated a *degree of freedom* (DOF) and varies from 1 to 3. A joint with 1 degree of freedom can only rotate on one axis, such as the human elbow. A joint with 3 degrees of freedom can rotate on three axes, such as the human neck that can make the head rotate up/down, left/right, and tilt sideways.

**Figure D. 6. Examples of different sub-stages of a 3D animation production stage - modeled mesh (top left) with polygon edges in blue; textured model (top right); rigged character (bottom left) with the skeleton joints shown inside the polygonal mesh and the hands' controls as circles on the wrists; and animated/posed/signaling character (bottom right). Model derived from MAKEHUMAN (2016).**



Images generated in Autodesk Maya.

In order to animate three-dimensional models' joints, its values must change over time. An *animation curve* is responsible for storing information on how each DOF changes its values (*y*) over time (*x*), it is a function of (usually) a joint's angle value given at a specific time: *f(x)* = *y*. It is important to note that the property of a function is that for each given value of *x* (time or frame) there is only 1 value of *y* (rotation or attribute value). To create an animation curve we have several techniques which can be grouped, according to PARENT (2012), into three different categories: Artistic, Data-Driven, and Procedural animation. Artistic Animation is where an artist is responsible for creating the exact poses of each object and guide the system on how to transition between those poses, it's the standard technique used in most computer animation films; Data-Driven Animation is when real-world data is the input to the movement, such as in motion capture; and Procedural Animation is where a mathematical model is created and the user is responsible on feeding different values to it, such as in rigid bodies simulations. These categories are further detailed in the next sections D.2.1.Artistic Animation, D.2.2 Data-driven Animation and D.2.3.Procedural Animation.

### D.2.1 Artistic Animation

This type of animation relies heavily on the animator's skills. While technically simple, it requires a keen perception of animation principles since the task to create the poses relies solely on the animator's talent and the computer is only used as a tool to create intermediate frames between these poses, substituting the assistant animator in the pipeline (Section D.1).

In each animation frame, the animator poses the model by changing its controllers' attributes and creating *keyframes*. It is important to emphasize the difference of keyframe definition in this context from the key pose definition used in traditional animation. In digital animation keyframes denote an instance in time where one or more attributes' values have been fixed. In traditional animation, a keyframe is defined as a specific drawing (or key pose) that defines the action. This traditional animation keyframe (henceforth in this work's context called key pose) in digital animation requires the animator to specify the controller's attributes values with a keyframe in the digital environment. In a single frame, a key pose usually contains multiple keyframes, one for each animation curve of a controller's attribute.

A single keyframe, $k$, can be thought of as a two-dimensional point in an animation curve, describing an attribute's value, $y$, in a specific time (frame), $x$, yielding a representation in the form $k = [y, x]$. From the previous section, an animation curve can be defined as a function $f(x) = y$ that maps an attribute's value $y$ to a unique time value $x$, as such a keyframe works as a direct mapping of the function $f$. Usually, an animation comprises of a series of keyframes, and thus a compact form to store and present keyframes is a list of values, $Y = [y_0, y_1, \dots, y_n]$ with its associated list of time, $X = [x_0, x_1, \dots, x_n]$.

Information between keyframes needs to be filled out. While in traditional animation this would be done by the animation assistant or the inbetweener, in digital animation this can be done manually with the animator changing the values of each animation curve frame by frame or, most commonly, by the software interpolating the values between each digital keyframe. This interpolation requires a mathematical curve model that can range from linear interpolation to complex spline curves where usually user interfaces are provided to the animator enabling him to increase or decrease acceleration between keyframes (O'ROURKE, 1998). The Hermite curve is an example of a curve that can provide parameters for the user to change keyframes' tangents, and it is further explored in Chapter 3 and Appendix A.

QIANG and XINGFEN (2012) propose a new approach to the interpolation process. Instead of using spline curves directly on the keyframes, a velocity curve is extracted from the curve's parametric representation. From analyzing the first and second derivatives (velocity and acceleration, respectively) intermediate keyframes are created, resulting in more complex interpolations than those possible with other curves.

Artistic computer animation system has had significantly less advances compared to other fields (modeling, rendering, etc.) or computer animation categories (shown below), patent in the fact that almost 20 years later modern software animation system still use the same methods proposed in O'ROURKE (1998).

An exception to it lies in the search for a yet to be found good general interpolation approach to 2D shape animation. Examples of this are the works of MELIKHOV et al (2004), WHITED et al (2010), and OLIVEIRA (2016).

### D.2.2 Data-driven Animation

While artistic animation relies heavily on the animator operating the system, data-driven animation is created by capturing information from the real world and applying it to computer graphics. Most research on this field dwells on the problem of retrieving information, such as on passive motion capture systems estimation of three-dimensional joint rotations based on a set of bi-dimensional image sequences, thus these are questions closely akin to computer vision and not within this dissertation's scope - computer graphics. That said, there are some interesting works that use data-driven models to generate or augment existing animations.

PULLEN and BREGLER (2002) propose a method that combines the flexibility of artistic animation with the nuances and naturalness of data-driven animation from motion capture. The method's goal is to use a user-created small keyframe set that defines key poses on a virtual skeleton and uses motion capture to augment the animation and create more natural movement. This is done by applying high-frequency noise from a set of banked motion capture data to the user-created key poses. When the virtual skeleton transitions from one key pose to another it does so with a very simple curve interpolation, resulting in a mechanic movement that lacks nuances found in real life. The high-frequency noise from motion capture is intended to bring these nuances in the movement. They call this process texturing. Their proposed method is divided into 4 steps:

1. Frequency Analysis: both user-created key poses and motion capture data are decomposed into frequency bands using Laplacian Pyramid decomposition. This is a process usually used in image processing described in BURT et ADELSON (1983) by which the signal $g_0$ with size $2^N +1$ is sub-sampled in $g_i$ signals with sizes $2^{N-i} + 1$ and each one has an associated band $l_i$ that is the difference between $g_{i-1}$ and an expanded $g_0$, i.e. $l_i = g_{i-1} – EXPANDED(g_i)$. This expansion does not produce $g_{i-1}$, but instead produces a lower resolution of $g_{i-1}$ because it has been sub-sampled. Therefore, on each band $l_i$ we have the high-frequency difference between the signal $g_i$ and its predecessor $g_{i-1}$.
2. Matching: a low-frequency band from the Laplacian Pyramid decomposition of a joint's angle is selected and broken down into fragments (in their work the authors used the left hip x angle). "The results are not highly dependent upon which frequency band is chosen, as long as it is low enough to provide information about the overall motion" (PULLEN et BREGLER, 2002). This low-frequency choice means that for a chosen joint angle animation curve we have both in the user-created and motion capture data a similar “overall” movement. The fragmentation is done both in the user-created and motion capture bands of the same joint angle. The point where the sign of the first derivative of each band changes is used to break down the band into fragments. This first derivative sign change can be roughly thought of as the change in orientation of the chosen joint angle. Now, for each fragment from the user-created band, we find *K* closer segments from the motion capture data.
3. Path Finding: since the motion has been segmented to combine them back a cost matrix based on the motion captured segments proximity is used to find the best closest matches between segments. As the authors point out:

For each join between fragments, we create a cost matrix, the *i*, $j^{th}$ component of which gives the cost for joining fragment *i* with fragment *j*. A score of zero is given if the fragments were consecutive in the original data, and one if they were not. We find all of the possible combinations of fragments that go through the points of zero cost.
This technique is easiest to explain using an example, which is diagrammed in [Figure D.7]. Suppose we had 4 fragments of synthetic data to match, and saved 3 nearest matches. In the illustration we show that for fragment 1 of the keyframed [user-created] data, the best matches were to fragments 4, 1, and 3 of the real data, and for fragment 2 of the keyframed [user-created] data the closest matches were to fragments 5, 7,and 2 of the real data [motion capture], and so on. We have drawn lines between fragments to indicate paths of zero cost. Here there are three best choices. One is fragment 4, 5, 6, and 2 from the real data. In this case we choose fragment 2 of the real data to match the fourth fragment of keyframed data rather than 8 or 5 because it was originally the closest match. A second possible path would be 4, 5, 4, and 5,and a third would be 1, 2, 4, 5. All three would yield two instances of zero cost. (PULLEN et BREGLER, 2002)

**Figure D. 7. Cost matrix image from PULLEN et BREGLER (2002). to find the best fragments of motion capture (Matching Data) to user-created (Keyframed) animation. Here K = 3. Blue lines indicate cost zero, that is the segments were consecutive.**

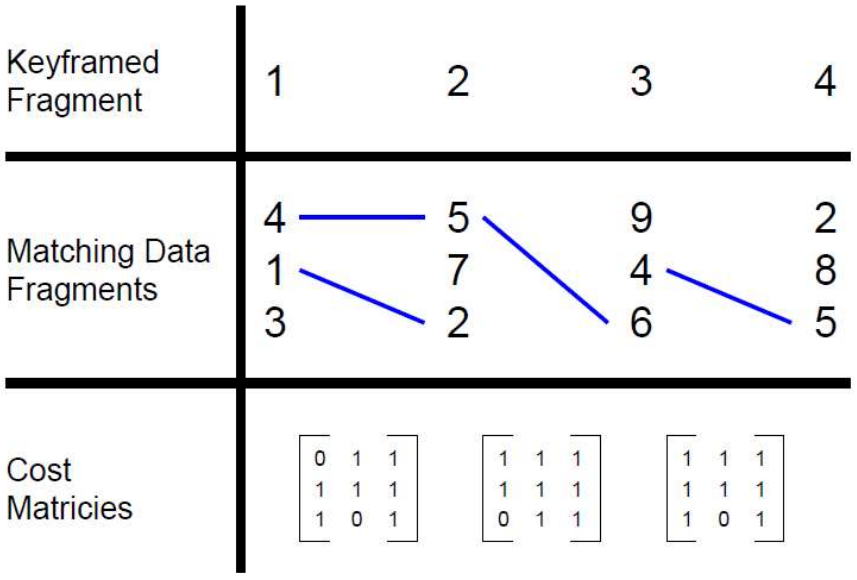


Image from PULLEN et BREGLER (2002).

4. Joining: since fragments' first and final values can not be guaranteed to coincide with those from their neighboring fragments, a joint algorithm is used. The mean value between the first segment's final value and the next segment's first value is used to skew each segment. “To achieve this warping, we define two lines, one that passes through the old endpoints, and one that passes through the new endpoints. We subtract the line that passes through the old endpoints and add the line that passes through the new endpoints to yield the shifted fragment” (PULLEN et BREGLER,

2002). Since this joining does not guarantee continuity between segments a smoothing on the join points is produced using a smoothing quadratic function on the data around the join points, the equations for this smoothing function are presented in their work.

JAIN et al (2009) propose a method in which motion capture data is used to create key poses based on an animator's drawing. They propose this system to speed up the process of creating a pose in a 3D character. Instead of the animator having to change each joint's rotation to achieve a desired key pose he simply draws the pose and an assistant helps translate it to a virtual environment. The system's input is a series of drawings that are annotated by assistants with virtual markers correlated to an already modeled and rigged 3D character. With this, the assistant indicates to the system where are specific points of the 3D character in the drawing (for instance, where is the left humerus in the drawing). Also, the assistants provide an estimated virtual camera position and orientation. These are used to create pose descriptors that retrieve a closed match motion capture pose. This is achieved by estimating the joints rotations and angles in three-dimensional space (based on the annotations and camera information provided by the assistant). Besides unburdening the animator with the process of creating the pose in the 3D environment, the assistant also needs only to provide information on 8 joints (left and right humerus, radius, femur and tibia) of the total 21 used to create the whole skeleton, that is, the system manages to not only estimate the position and rotation of the joints given by the assistant but also pose the whole set of 21 joints of the virtual character in the pose the animator intended with their drawings.

It is interesting to note that both methods use only a small set of information - just one frequency band on the first and just 8 joints for pose descriptors on the second - to retrieve motion capture data and generate a full, complex movement.

### D.2.3 Procedural Animations

In terms of user control, this type of animation can be seen as an intermediate between artistic and data-driven animation creating movement automatically but still enabling the animator to modify parameters that significantly alter the final result. Procedural animation can require a lot of processing power to create animation curves. It uses mathematical models that simulate real-world situations, and with increasing model complexity, such as movement affected by friction and air turbulence, the number of variables and constraints represented as mathematical functions needed to be evaluated for each frame requires more processing power.

As discussed in PARENT (2012), procedural animation is widely used to simulate physical phenomenons - cloth, water, soft and rigid bodies. This section focuses on the use of procedural animation on virtual humans.

Before artistic animation became the standard in the entertainment industry a lot of research was done aiming at systems that could animate characters based solely on the mathematical formulation of its movements.

COHEN (1992), and later LIU et al (1994), propose a model to "provide the ability to interactively design the motion of simulated creatures which appears correct, and satisfies goals and objectives set by the user" (COHEN, 1994). This model presents the user the possibility to create a set of constraints on DOFs as a function of time, similar to a keyframe, which then is processed spanning the entire length of the animation as cubic B-Spline curves which should satisfy such constraints. To solve the problem, a combination of Lagrange's method and Newton's method is used. The solution is then presented to the user who can then interactively change the animation by creating, deleting, or modifying constraints. To test the system, COHEN (1992) used three situations:

- a three-linked arm and ball, in which a stationary arm with DOFs in shoulders, elbow and wrist has to throw a ball (point mass representation) forward, to calculate this animation physical constrain (gravity and friction) is applied to the ball, initial and final position constrains to the hand and minimization of the arm's torque throughout the entire animation are considered, in a first iteration of the system it did not produce the expected movement, but with new user constrains the desired effect was created;
- a basketball player, which was similar to the previous example, but the arm (shoulder) now could move in space;
- acrobat, a very simple model of a character with just the foot and waist movable was required to jump forward similarly to the Pixar's Luxo Jr. animation.

Although this method requires a somewhat complex understanding of mathematics and physics not usually mastered by animators (the last example took 158 constrains), it does provide an interesting approach to motion synthesis in which combinations of artistic and procedural animations are used.

KIM, CORDIER et MAGNENAT-THALMANN (2000) created a virtual hand that, using a neural network, played the violin according to correct physical constraints and positions following recommended violin pedagogy. A score containing each string touch's start time, end time, and intensity is used as input to the neural network, which has been previously trained with correct fingering, and a best-first search is used to determine first the *active fingers* (those that are touching the strings of the violin), then wrist state and later *passive finger* (those that are not contributing to produce the desired note). "The advantage of this fingering determination method over the rule-based method is adaptability to different factors, ability to consider global effect, and scalability to polyphony." (KIM, CORDIER et MAGNENAT-THALMANN, 2000) This adaptability is an interesting asset that was considered when designing the method presented in this dissertation (Chapter 3).