Laplacian Mesh Editing for Interaction Learning

Laplace'sche Polygonnetz Editierung für Interaktions Lernen
Bachelor-Thesis von Hong Linh Thai
März 2014

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Laplacian Mesh Editing for Interaction Learning
Laplace'sche Polygonnetz Editierung für Interaktions Lernen

Vorgelegte Bachelor-Thesis von Hong Linh Thai

1. Gutachten: Prof. Dr. Jan Peters
2. Gutachten: Dr. Heni Ben Amor
3. Gutachten: M.Eng. Oliver Kroemer

Tag der Einreichung:

Erklärung zur Bachelor-Thesis

Hiermit versichere ich, die vorliegende Bachelor-Thesis ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.
In der abgegebenen Thesis stimmen die schriftliche und elektronische Fassung überein.

Darmstadt, den 4. März 2014

(Hong Linh Thai)

Thesis Statement

I herewith formally declare that I have written the submitted thesis independently. I did not use any outside support except for the quoted literature and other sources mentioned in the paper. I clearly marked and separately listed all of the literature and all of the other sources which I employed when producing this academic work, either literally or in content. This thesis has not been handed in or published before in the same or similar form.
In the submitted thesis the written copies and the electronic version are identical in content.

Darmstadt, March 4, 2014

(Hong Linh Thai)

Abstract

For motion editing of close interactions between one or multiple people, such as handing over objects or punching, it is necessary to preserve the implicit spatial and temporal relationships between the different joints in order to keep the edited motion as close as possible to the original movement. These relationships can be encoded by an interaction mesh and the Laplacian representation of the mesh. Using this representation retargeting and synthesis of a motion can be formulated as a combination of several optimizations. In this work, distinctive movements are analysed with respect to how they and their spatial and temporal relationships are encoded in the Laplacian space. Furthermore, we investigated in this work, if it is possible to adapt the weights of the Laplacian energy or the interaction meshes to a specific movement in order to improve the reconstructed results. Finally, we show how this method could be applied to temporal sequences.

Zusammenfassung

Bei der Bearbeitung von Bewegungen, die enge Interaktionen zwischen einer oder mehreren Personen beschreiben, wie die Übergabe eines Gegenstands oder einem Fausthieb, ist Erhaltung der räumlichen und zeitlichen Abhängigkeiten zwischen den jeweiligen Gelenken der interagierenden Personen entscheidend, um die bearbeitete Bewegung möglichst nah am Original zu halten. Diese Abhängigkeiten können mit Hilfe eines Interaktionsnetz und der Laplace'schen Darstellung eines Polygonnetzes einkodiert werden, so dass die Synthese von neuen Bewegungen und das Übertragen der Bewegung auf neue Proportionen durch diverse Optimierungen formuliert werden kann. In dieser Arbeit wird die Darstellung von verschiedenen Bewegungen im Laplace'schen Raum analysiert und anhand dessen die Möglichkeit zur Verbesserung der synthetisierten Ergebnisse durch die Anpassung der Gewichte der Laplace'schen Energie oder der Interaktionsnetze untersucht.

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

"In the long history of humankind (and animal kind, too) those who learned to collaborate and improvise most effectively have prevailed.", Charles Darwin

Since the stone age, humans have been cooperating and collaborating with each other, as cooperation and collaboration are basic interactions between human beings. In the process of cooperation and collaboration close interactions between two persons are often needed as a common way to communicate and work together. With the birth of artificial intelligence effective collaboration and close interactions between humans and robots are becoming progressively more important, considering the continuous growth in the usage of robots in industrial manufacturing and possible future usage of robots in different fields of our daily life (e.g. household, elderly care). In order to achieve close interactions between humans and robots, robots need to learn from observing actions between two humans and generalize them to new robot-human interactions.

As the semantic of a close interaction is not only represented by the position of the joints of the interacting agents but also in the spatial relationship between different body parts, it is necessary to incorporate both aspect when learning a specific movement. One way to encode this information is using *interaction meshes* [HKT10] or *3d+t graphs* [NCG13] and their *Laplacian representation* [Ale03], which were originally used in animation editing and computer graphics. *Interaction mesh* and *3d+t graph* are volumetric meshes defined by their vertices and edges. The vertices are defined by the joints of the interacting agents and the shape of the objects with which the agents are interacting. The edges are defined by the spatial and temporal relationships between the joints of the agents and the objects. In order to adapt to different skeleton topologies or to new situations, it is necessary to keep the high-level semantics of the interaction by preserving the local details and global shape of the *interaction mesh* or the *3d+t graph*. *Laplacian mesh editing*, originally proposed by Alexa [Ale03] and Sorkine [SCOL$^+$04], is ideal for this task, as it aims to deform a mesh in a specified local region while preserving the global shape of the mesh as close as possible to the original.

Furthermore, these interaction meshes have also been successfully used by Zarubin et al. [ZIT$^+$12] for motion planing of a reaching task on the KUKA LWR 4 robotic arm in a dynamic environment as well as by Ho et al. [HS13] for synthesizing full body motions on a humanoid robot capable of adapting in constraint environments. Therefore, this work's goal is to learn the spatial and temporal relationship of a distinctive movement between two humans and use this knowledge to adapt interaction meshes to the movement to be learned. The learning process is achieved by using *steepest ascent hill climbing*, adapting an initial generic *interaction mesh* by adding additional and removing existing edges, as the edges represent the spatial and temporal relationship between specific joints of the movement. Once an *interaction mesh* is adapted to a specific movement between two humans, it could be used by robots to take one part of the interaction and to interact with the human counter part.

The thesis is structured in the following way: To being with, in second chapter, we will give an introduction to meshes and explain *Laplacian mesh editing* as mesh deformation technique. In the third chapter, we will show different possibilities to generate meshes from a given temporal sequence and explain how *Laplacian mesh editing* could be applied on these meshes to modify or synthesise similar motions. Moreover, in the third chapter our own approach to adapt *interaction meshes* to specific movements will be presented. Results of this method are shown and discussed in chapter 4 and 5.

2 Laplacian Mesh Editing

Laplacian Mesh Editing is a free form editing technique widely used in computer vision. It was proposed in 2003 by Marc Alexa [Ale03] and in 2004 by Sorkine et al. [SCOL+04]. The goal of Laplacian Mesh Editing is to deform a mesh in a specified local region while preserving the global shape of the mesh. Therefore the mesh is represented in *differential coordinates*, which describe the local properties of the mesh rather the absolute position in space. This chapter should give an introduction into Laplacian Mesh Editing and give an intuitive understanding.

2.1 Formal Definition of a Mesh

A mesh $\mathbf{M}$ can be formally seen as a undirected Graph, defined as a pair $(\mathbf{V}, \mathbf{E})$. $\mathbf{V} = \{v_1, v_2, \ldots, v_{|\mathbf{V}|}\}$ is a set containing the vertices of the graph and $\mathbf{E} = \{e_1, e_2, \ldots, e_{|\mathbf{E}|}\}$ with $e_i \in \mathbf{V} \times \mathbf{V}$ is a set containing the edges of the graph, describing the connectivity of the vertices. In the following the vertices of a mesh will be represented by their geometrical position with $v_i \in \mathbb{R}^d$, $1 \leq i \leq |\mathbf{V}|$ and d as the dimension of the mesh.

2.2 Laplacian Coordinates

Laplacian coordinates are *differential coordinates*, which can be theoretically derived from the *Laplace-Beltrami operator*. They can be seen as a discretization of the continuous *Laplace-Beltrami operator*, if one sees a mesh as a piecewise-linear approximation of a smooth surface [Sor05] [BS08].
Let δ_i be the Laplacian coordinate and v_i be the absolute coordinate of a vertex i. Then the Laplacian coordinate δ_i and the *Laplacian operator* L is defined as:

$$\delta_{\mathbf{i}} = L(\mathbf{v_i}) = n_i \sum_{\mathbf{v_j} \in N(\mathbf{v_i})} w_{ij}(\mathbf{v_i} - \mathbf{v_j}) \tag{2.1}$$

where $v_j \in N(\mathbf{v_i})$ are the direct neighbours of v_i, n_i is a per-vertex normalization weight and w_{ij} are the edge weights. Intuitively the Laplacian coordinate is the difference from the vertex to a weighted centroid of its neighbour. Hence one can move the centroid nearer or farer away from specific neighbours, using the weights.

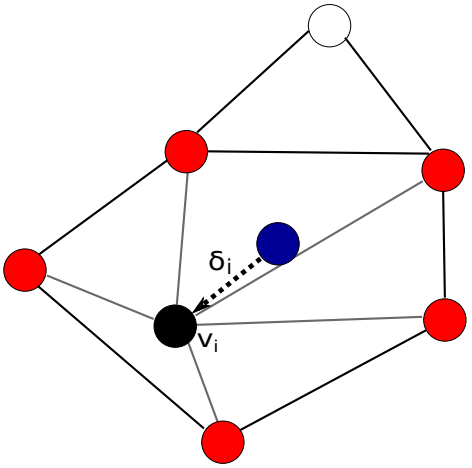


Figure 2.1.: An example of a vertex(black) and its Laplacian coordinate(arrow) calculated from the centroid(blue) of its neighbours(red) in a 2 dimensional Mesh

Thus the discretization, respectively the Laplacian coordinate, depends on n_i and w_{ij} leading to different weighting schemes (see Section 2.3).

2.3 Weighting Schemes

In the following, two standards weighting schemes for the discretization in computer vision are presented and explained. *Uniform weights* [SCOL+04] [BS08] [Ale03] and *cotangent weights*. [BS08] [DMSB99]

2.3.1 Uniform Weights

Uniform weights use $w_{ij} = 1$ and $n_i = \frac{1}{\sum_j w_{ij}}$. Intuitively they define the Laplacian coordinate as the difference from the vertex to the centroid of its neighbour without taking the local geometry (angles or edge lengths) of the mesh in consideration, as shown in Figure 2.2. Therefore they cannot give a good approximation on irregular meshes [BS08].

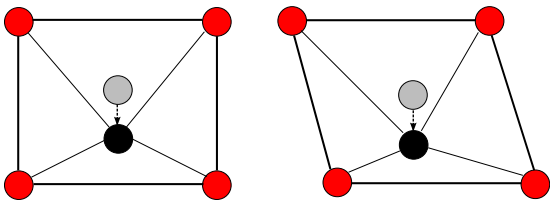


Figure 2.2.: An example of uniform weights of different mesh topologies but same laplacian coordinates

2.3.2 Cotangent Weights

Cotangent weights were proposed first by Desbrun et Al. in 1999 as an improvement of the *uniformed weights* and as a better approximation of the *Laplace-Beltrami operator*, including the information about the local geometry of the mesh [DMSB99]. They set the weights as

$$w_{ij} = \cot \alpha_{ij} + \cot \beta_{ij} \qquad n_i = \frac{1}{\sum_j \cot \alpha_{ij} + \cot \beta_{ij}} \tag{2.2}$$

with α_{ij} and β_{ij} as the angles opposite to the edge of the two triangles having the edge e_{ij} in common. (depicted in figure 2.3) An alternative formulation by Botsch et Al. in 2004 [BK04] is

$$w_{ij} = \frac{1}{2}(\cot \alpha_{ij} + \cot \beta_{ij}) \qquad ni = \frac{1}{A_i} \tag{2.3}$$

where A_i is the *Voronoy area* of the vertex i, defined as the area built from the incident triangles' circumcenters or the edge midpoints for obtuse angles. (depicted in figure 2.3)
According to Botsch and Sorkine [BS08] these weights lead to the best solution. However, in a mesh with near-degenerate triangles the usage of cotangent weight could lead to numerical problems and singular matrices, as the cotangent values would degenerate.

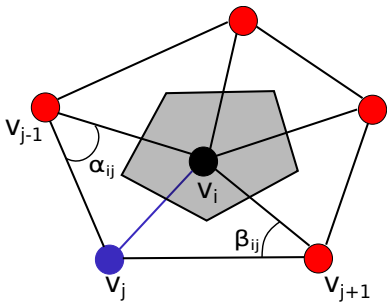


Figure 2.3.: The angles and the voronoi area (dark grey) used in Equation 2.2 and 2.3 for an edge between vertex i (black) and its neighbour j (blue)

2.4 Laplacian System

This section will explain how one can create a linear system of equations and solve it in order to edit a mesh, independent from the chosen weighting scheme.

2.4.1 Laplacian Matrix

The transformation between Euclidean space and Laplacian space (see Section 2.2) can be written for the whole mesh as a matrix operation

$$\begin{pmatrix} \delta_1 \\ \vdots \\ \delta_{|V|} \end{pmatrix} = \underbrace{\boldsymbol{M^{-1}L_s}}_{L} \begin{pmatrix} \mathbf{v_1} \\ \vdots \\ \mathbf{v}_{|V|} \end{pmatrix} \tag{2.4}$$

$$\boldsymbol{M_{ii}} = n_i \qquad \boldsymbol{L_s} = \begin{cases} -\sum_{\mathbf{v_k} \in N(\mathbf{v_i})} w_{ik} & i = j \\ w_{ij} & v_j \in N(\mathbf{v_i}) \\ 0 & otherwise \end{cases}$$

where $\boldsymbol{M}$ is a diagonal matrix containing the normalization weights n_i for each vertex and $\boldsymbol{L_s}$ is a symmetric matrix containing the different edge weights w_{ij} [BS08].
It can be shown that $\boldsymbol{L} = \boldsymbol{M^{-1}L_s} \in \mathbb{R}^{|V|x|V|}$ is symmetric, sparse and has rank $|V| - 1$ [SCOL$^+$04] [Ale03] [Sor05]. Thus $\boldsymbol{L}$ is not invertible and therefore the mesh is not reconstructible without more specified constraints. $\boldsymbol{L}$ is known as the *Laplacian Matrix* or as the *Laplacian Operator* for a mesh.
When using *uniform weights* $\boldsymbol{L}$ can be simplified to

$$\boldsymbol{L} = \boldsymbol{I} - \boldsymbol{DA}$$

where $\boldsymbol{D}$ is a diagonal matrix with $\boldsymbol{D_{ii}} = \frac{1}{|N(v_i)|}$ and $\boldsymbol{A}$ as the *adjacency matrix* of the mesh, defined as :

$$\boldsymbol{A_{ij}} = \begin{cases} 1 & \text{if } (\mathbf{v_i}, \mathbf{v_j}) \in \mathbf{E} \\ 0 & \text{otherwise} \end{cases}$$

2.4.2 Reconstruction of the Mesh from Laplacian Coordinates

As shown in the previous section, it is not uniquely possible to reconstruct the Euclidean coordinates given the Laplacian coordinates and $\boldsymbol{L}$, as $\boldsymbol{L}$ is singular. Therefore the expression

$$\begin{pmatrix} \mathbf{v_i} \\ \vdots \\ \mathbf{v}_{|V|} \end{pmatrix} = L^{-1} \begin{pmatrix} \boldsymbol{\delta_i} \\ \vdots \\ \boldsymbol{\delta}_{|V|} \end{pmatrix} \tag{2.5}$$

is not defined. For that reason, Sorkine and Alexa proposed to fix the position of several vertices, as additional position constraints [SCOL$^+$04] [Sor05] [Ale03].

$$\mathbf{v_j} = \mathbf{c_j} \quad j \in C \tag{2.6}$$

where C is a set containing all the indices of the fixed vertices. By adding the positional constrains one receives the following linear equation system.

$$\underbrace{\begin{pmatrix} & \boldsymbol{L} & \\ \dots & \omega_1 & \dots \\ & \vdots & \\ \dots & \omega_{|C|} & \dots \end{pmatrix}}_{\tilde{L}} \begin{pmatrix} \mathbf{v_i'} \\ \vdots \\ \mathbf{v}'_{|V|} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\delta_i} \\ \vdots \\ \boldsymbol{\delta}_{|V|} \\ \omega_1 \mathbf{c_1} \\ \vdots \\ \omega_{|C|} \mathbf{c}_{|C|} \end{pmatrix} \tag{2.7}$$

The system matrix of Equation 2.7 is referred as $\tilde{\boldsymbol{L}} \in \mathbb{R}^{(|V|+|C|)\times|V|}$ and contains additional rows for the given positional constrains with the weight ω_i at the corresponding ith index column. The right hand side vector is extended in addition by the respective weighted result of the positional constrains. By adding the positional constrains as additional rows, instead of substituting them, the positional constrains are treated in a least-square sense. Thus they may not be fully satisfied by the least-squares solution and are no hard constraints. By increasing the weight ω_i one can enforce the positional constraints. However, this leads to higher condition numbers for $\tilde{\boldsymbol{L}}$ and possibly results in numerical problems [BS08].
Due to the additional positional constraints, $\tilde{\boldsymbol{L}}$ is an overdetermined system (more equations than variables) with full rank, which can be uniquely solved using *linear least squares*. Solving the Equation 2.7 with *linear least squares* is equivalent to minimizing the following energy function

$$E(V') = \sum_{i=1}^{|V|} ||\boldsymbol{L}\mathbf{v}'_\mathbf{i} - \boldsymbol{\delta}_i||^2 + \sum_{j\in C} \omega_j ||\mathbf{v}'_\mathbf{j} - \mathbf{c}_\mathbf{j}||^2 \tag{2.8}$$

Intuitively, this energy describes the difference in the Laplacian coordinates between a modified mesh and the given mesh and the difference between the modified mesh and the given positional constraint. Thus, the first term tries to preserve the local information of the original mesh, while the second term tries to satisfy the given positional constraints. The analytical least-squares solution for Equation 2.7 can be computed by the *pseudo-inverse* of the system matrix $\tilde{\boldsymbol{L}}$.

$$\begin{pmatrix} \mathbf{v}'_\mathbf{i} \\ \vdots \\ \mathbf{v}'_{|\mathbf{V}|} \end{pmatrix} = (\tilde{\boldsymbol{L}}^T\tilde{\boldsymbol{L}})^{-1}\tilde{\boldsymbol{L}}^T \begin{pmatrix} \boldsymbol{\delta}_i \\ \vdots \\ \boldsymbol{\delta}_{|V|} \\ \omega_1\mathbf{c_1} \\ \vdots \\ \omega_{|C|}\mathbf{c}_{|\boldsymbol{C}|} \end{pmatrix} \tag{2.9}$$

To solve the system effectively and accurate Sorkine proposed to use a sparse Cholesky factorization, which can be found in [Sor05]. Advantage of that method is that the factorization is only computed once when the system is build. This is useful when editing is done repeatedly on different targets without changing the model.
She however notes, that the accuracy of the solution is highly dependent on the conditioning of the linear system and that the Laplacian matrix is often ill-conditioned. Though, one can improve the condition numbers and stabilize the factorization by adding more positional constraints.
However, this solution does not include the sensitivity of Laplacian Coordinates to linear transformation. A solution, which covers this aspect, can be found in Section 2.5.2

2.5 Sensitivity to Linear Transformations

The optimization in Section 2.4.2 only works, if the chosen positional constraints of Equation (2.6) do not imply a linear transformation. The reason for this behaviour is that Laplacian coordinates are sensitive to linear transformations. Thus, the local information of the structure can be translated, but not scaled or rotated, as shown in Figure 2.4.

To deal with this problem a local transformation is often used. The idea of this local transformation is to keep the relative orientation and size of the the local features during the deformation and to split the problem into two sequential problems. The first part is to derive the transformation $\boldsymbol{T}_i$, which holds (definition of $\delta_\mathbf{i}$ and $\delta'_\mathbf{i}$ see Figure 2.4).

$$\delta'_\mathbf{i} = \boldsymbol{T}_i\delta_\mathbf{i} \tag{2.10}$$

The second part is to plug these expressions into the previous reconstruction problem (Section 2.4.2) and compute the final solution by minimizing the error function.

$$E(\boldsymbol{V'}) = \sum_{i=1}^{|V|} ||\boldsymbol{L}\mathbf{v}'_\mathbf{i} - \boldsymbol{T}_i\boldsymbol{\delta}_i||^2 + \sum_{j\in C} ||\mathbf{v}'_\mathbf{j} - \boldsymbol{c}_j||^2 \tag{2.11}$$

There are different ways to compute this local transformation [BS08], however common methods are the estimation of the transformation from an initial guess solution [LSCO$^+$04], comparing the original surface and the deformed surface in a neighbourhood of $\mathbf{v}_\mathbf{i}$ using normals or the usage of an implicit transformation [SCOL$^+$04].

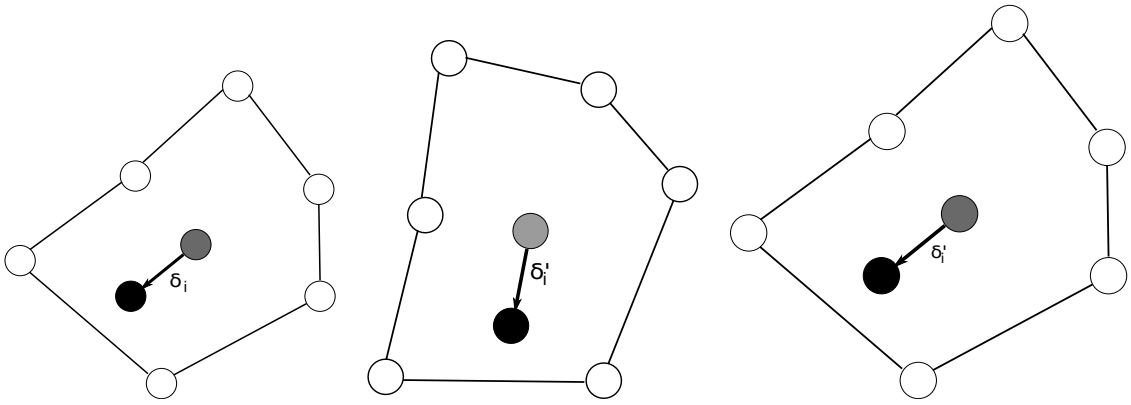


Figure 2.4.: As rotation and scalation of the mesh changes the Laplacian coordinate δ_i, Laplacian coordinates are sensitive to linear transformations

2.5.1 Implicit Transformation

Implicit transformations, proposed by Sokrine [SCOL+04], are one way to deal with the unknown transformation, combining both steps of last section into one. They are based on the assumption that the unknowns of $\boldsymbol{T_i}$ are a linear function of $\boldsymbol{V'}$, modifying Equation (2.11) to:

$$E(\boldsymbol{V'}) = \sum_{i=1}^{|V|} ||\boldsymbol{L}\mathbf{v}'_\mathbf{i} - \boldsymbol{T_i}(\boldsymbol{V'})\boldsymbol{\delta_i}||^2 + \sum_{j \in C} ||\mathbf{v}'_\mathbf{j} - \mathbf{c}_j||^2 \tag{2.12}$$

Thus solving $E(V')$ implies finding $\boldsymbol{T_i}$ implicitly.
Each $\boldsymbol{T_i}$ is derived from the transformation of $\mathbf{v_i}$ and its neighbours to their reconstructed position. As a result Sorkine propose that $\boldsymbol{T_i}$ should hold:

$$\boldsymbol{T_i} = \underset{\boldsymbol{T_i}}{\operatorname{argmin}}(\, ||\boldsymbol{T_i}\mathbf{v_i} - \mathbf{v}'_\mathbf{i}||^2 + \sum_{j \in N(v_i)} ||\boldsymbol{T_i}\mathbf{v_j} - \mathbf{v}'_\mathbf{j}||^2 \,) \tag{2.13}$$

Furthermore, $\boldsymbol{T_i}$ needs to be constrained to only representing rotations, scaling and translations and no other operation. In the 2D case it is easy to define such a linear transformation complying the constraint. However, in the 3D case it is not possible to define a exact linear transformation, as rotation matrices cannot be expressed linearly in 3D. Hence a good linear approximation for small angles is used in this case [SCOL+04]

$$\boldsymbol{T_i} = \begin{pmatrix} s_i & -h_{i,z} & h_{i,y} & t_{i,x} \\ h_{i,z} & s_i & -h_{i,x} & t_{i,y} \\ -h_{i,y} & h_{i,x} & s_i & t_{i,z} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.14}$$

Further details for the derivation of the approximation can be found in [LSA+05]. Given now $\boldsymbol{T_i}$, one can write down the linear dependency of equation (2.13) using *homogeneous coordinates* on $\boldsymbol{V'}$ as:

$$\begin{pmatrix} s_i & -h_{i,z} & h_{i,y} & t_{i,x} \\ h_{i,z} & s_i & -h_{i,x} & t_{i,y} \\ -h_{i,y} & h_{i,x} & s_i & t_{i,z} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} v_{k,x} \\ v_{k,y} \\ v_{k,z} \\ 1 \end{pmatrix} = \begin{pmatrix} v'_{k,x} \\ v'_{k,y} \\ v'_{k,z} \\ 1 \end{pmatrix} \quad k \in \{i\} \cup N(\mathbf{v_i}) \tag{2.15}$$

where $v_{k,x}$ is the x-coordinate of the kth vertex, vice-versa for $v_{k,y}, v_{k,z}$. As one needs $s_i, \boldsymbol{h_i}$ and $\boldsymbol{t_i}$ as expression of $\boldsymbol{V}$ and $\boldsymbol{V'}$, the equation system can be reformulated as

$$\underbrace{\begin{pmatrix} v_{k,x} & 0 & v_{k,z} & v_{k,y} & 1 & 0 & 0 \\ v_{k,y} & -v_{k,z} & 0 & v_{k,y} & 0 & 1 & 0 \\ v_{k,z} & v_{k,y} & -v_{k,x} & 0 & 0 & 0 & 1 \\ & & \vdots & & & & \end{pmatrix}}_{\boldsymbol{A_i}} \begin{pmatrix} s_i \\ h_{i,x} \\ h_{i,y} \\ h_{i,z} \\ t_{i,x} \\ t_{i,y} \\ t_{i,z} \end{pmatrix} = \underbrace{\begin{pmatrix} v'_{k,x} \\ v'_{k,y} \\ v'_{k,z} \\ \vdots \end{pmatrix}}_{\boldsymbol{b_i}} \quad k \in \{i\} \cup N(\mathbf{v_i}) \tag{2.16}$$

As this equation is equivalent to Equation (2.13), one can write down this system as $\boldsymbol{A_i}(s_i, \boldsymbol{h_i}, \boldsymbol{t_i})^T = \boldsymbol{b_i}$ and solve the system in the least-square sense via the *pseudo-inverse* resulting in

$$\begin{pmatrix} s_i \\ h_{i,x} \\ h_{i,y} \\ h_{i,z} \\ t_{i,x} \\ t_{i,y} \\ t_{i,z} \end{pmatrix} = (\boldsymbol{A_i}^T \boldsymbol{A_i})^{-1} \boldsymbol{A_i}^T \begin{pmatrix} v'_{k,x} \\ v'_{k,y} \\ v'_{k,z} \\ \vdots \end{pmatrix} = \boldsymbol{A_i}^+ \begin{pmatrix} v'_{k,x} \\ v'_{k,y} \\ v'_{k,z} \\ \vdots \end{pmatrix} = \begin{pmatrix} \boldsymbol{A^+_{i,11}} v'_{k,x} + \boldsymbol{A^+_{i,12}} v'_{k,y} + \boldsymbol{A^+_{i,13}} v'_{k,z} + \dots \\ \boldsymbol{A^+_{i,21}} v'_{k,x} + \boldsymbol{A^+_{i,22}} v'_{k,y} + \boldsymbol{A^+_{i,23}} v'_{k,z} + \dots \\ \boldsymbol{A^+_{i,31}} v'_{k,x} + \boldsymbol{A^+_{i,32}} v'_{k,y} + \boldsymbol{A^+_{i,33}} v'_{k,z} + \dots \\ \vdots \end{pmatrix} \quad k \in \{i\} \cup N(\mathbf{v_i}) \tag{2.17}$$

Since $\boldsymbol{A_i}$ is already known from the initial mesh, the coefficient of $\boldsymbol{T_i}$ are linearly dependent of $\boldsymbol{V'}$ as required. The calculated coefficients can now be plugged into $\boldsymbol{T_i}$ to calculate the transformation of the Laplacian coordinates $\boldsymbol{\delta_i}$.

$$\begin{pmatrix} \delta'_{i,x} \\ \delta'_{i,y} \\ \delta'_{i,z} \end{pmatrix} = \begin{pmatrix} s_i & -h_{i,z} & h_{i,y} \\ h_{i,z} & s_i & -h_{i,x} \\ -h_{i,y} & h_{i,x} & s_i \end{pmatrix} \begin{pmatrix} \delta_{i,x} \\ \delta_{i,y} \\ \delta_{i,z} \end{pmatrix} \quad k \in \{i\} \cup N(\mathbf{v_i}) \tag{2.18}$$

$$= \begin{pmatrix} \boldsymbol{A^+_{i,11}} v'_{k,x} + \boldsymbol{A^+_{i,12}} v'_{k,y} + \boldsymbol{A^+_{i,13}} v'_{k,z} + \dots & -(\boldsymbol{A^+_{i,41}} v'_{k,x} + \boldsymbol{A^+_{i,42}} v'_{k,y} + \boldsymbol{A^+_{i,43}} v'_{k,z} + \dots) & \boldsymbol{A^+_{i,31}} v'_{k,x} + \boldsymbol{A^+_{i,32}} v'_{k,y} + \boldsymbol{A^+_{i,33}} v'_{k,z} + \dots \\ \boldsymbol{A^+_{i,41}} v'_{k,x} + \boldsymbol{A^+_{i,42}} v'_{k,y} + \boldsymbol{A^+_{i,43}} v'_{k,z} + \dots & \boldsymbol{A^+_{i,11}} v'_{k,x} + \boldsymbol{A^+_{i,12}} v'_{k,y} + \boldsymbol{A^+_{i,13}} v'_{k,z} + \dots & -(\boldsymbol{A^+_{i,21}} v'_{k,x} + \boldsymbol{A^+_{i,22}} v'_{k,y} + \boldsymbol{A^+_{i,23}} v'_{k,z} + \dots) \\ -(\boldsymbol{A^+_{i,31}} v'_{k,x} + \boldsymbol{A^+_{i,32}} v'_{k,y} + \boldsymbol{A^+_{i,33}} v'_{k,z} + \dots) & \boldsymbol{A^+_{i,21}} v'_{k,x} + \boldsymbol{A^+_{i,22}} v'_{k,y} + \boldsymbol{A^+_{i,23}} v'_{k,z} + \dots & \boldsymbol{A^+_{i,11}} v'_{k,x} + \boldsymbol{A^+_{i,12}} v'_{k,y} + \boldsymbol{A^+_{i,13}} v'_{k,z} + \dots \end{pmatrix} \begin{pmatrix} \delta_{i,x} \\ \delta_{i,y} \\ \delta_{i,z} \end{pmatrix}$$

In this case, no *homogeneous coordinates* are used as Laplacian coordinates are invariant to translations.
As the expression $\boldsymbol{T_i}(\boldsymbol{V'})\boldsymbol{\delta_i}$ from equation (2.12) is only dependent on the unknowns $\mathbf{v'_i}$ and $\mathbf{v'_k}$, $k \in N(\mathbf{v_i})$ seen in latter Equation (2.18)(the matrix $\boldsymbol{A_i}$ and the vector $\boldsymbol{\delta_i}$ are given from the initial mesh) , one can build an equation system similar to the *Laplacian System* (Equation (2.7)), which includes the linear transformation $\boldsymbol{T_i}$ and is equivalent to equation (2.12).

2.5.2 Laplacian System including the Implicit Transformation

Using the result from the last section one can reformulate Equation (2.12) as:

$$E(V') = \sum_{i=1}^{|V|} ||\boldsymbol{L} \begin{pmatrix} v'_{i,x} \\ v'_{i,y} \\ v'_{i,z} \end{pmatrix} - \tilde{\boldsymbol{T}}_i \begin{pmatrix} v'_{i,x} \\ v'_{i,y} \\ v'_{i,z} \\ v'_{k,x} \\ v'_{k,y} \\ v'_{k,z} \\ \vdots \end{pmatrix} ||^2 + \sum_{j \in C} \omega_j ||\mathbf{v'_j} - \mathbf{c_j}||^2 \quad k \in N(\mathbf{v_i}) \tag{2.19}$$

$$\text{with} \quad \tilde{\boldsymbol{T}}_i = \begin{pmatrix} \boldsymbol{A^+_{i,11}}\delta_{i,x} - \boldsymbol{A^+_{i,41}}\delta_{i,y} + \boldsymbol{A^+_{i,31}}\delta_{i,z} & \boldsymbol{A^+_{i,12}}\delta_{i,x} - \boldsymbol{A^+_{i,42}}\delta_{i,y} + \boldsymbol{A^+_{i,32}}\delta_{i,z} & \dots \\ \boldsymbol{A^+_{i,41}}\delta_{i,x} + \boldsymbol{A^+_{i,11}}\delta_{i,y} - \boldsymbol{A^+_{i,21}}\delta_{i,z} & \boldsymbol{A^+_{i,42}}\delta_{i,x} + \boldsymbol{A^+_{i,12}}\delta_{i,y} - \boldsymbol{A^+_{i,22}}\delta_{i,z} & \dots \\ -\boldsymbol{A^+_{i,31}}\delta_{i,x} + \boldsymbol{A^+_{i,21}}\delta_{i,y} - \boldsymbol{A^+_{i,11}}\delta_{i,z} & -\boldsymbol{A^+_{i,32}}\delta_{i,x} + \boldsymbol{A^+_{i,22}}\delta_{i,y} + \boldsymbol{A^+_{i,12}}\delta_{i,z} & \dots \end{pmatrix}$$

As the first sum of the equation is now only dependent on the unknown $\boldsymbol{V'}$, one can reformulate this sum into a matrix. The first term of this sum can be described as following:

$$\underbrace{\begin{pmatrix} \boldsymbol{L} & 0 & 0 \\ 0 & \boldsymbol{L} & 0 \\ 0 & 0 & \boldsymbol{L} \end{pmatrix}}_{\underline{\boldsymbol{L}}} \begin{pmatrix} \mathbf{v_x}' \\ \mathbf{v_y}' \\ \mathbf{v_z}' \end{pmatrix} \quad \text{with} \quad \mathbf{v_x}' = \begin{pmatrix} v_{1,x} \\ \vdots \\ v_{|V|,x} \end{pmatrix} \; \mathbf{v_y}' = \begin{pmatrix} v_{1,y} \\ \vdots \\ v_{|V|,y} \end{pmatrix} \; \mathbf{v_z}' = \begin{pmatrix} v_{1,z} \\ \vdots \\ v_{|V|,z} \end{pmatrix} \tag{2.20}$$

The matrix above is abbreviated as $\underline{\boldsymbol{L}} \in \mathbb{R}^{3|V| x 3|V|}$ containing the *Laplacian Matrix* $\boldsymbol{L}$, described in Section 2.4.1. To include the second term into $\underline{\boldsymbol{L}}$, one just needs to subtract the indices of $\tilde{\boldsymbol{T}}_i$ at the corresponding rows and

columns of $\underline{L}$. For example the first index of $\tilde{T}_1$, $\tilde{T}_{1,11} = A^+_{1,11}\delta_{i,x} - A^+_{i,41}\delta_{1,y} + A^+_{1,31}\delta_{i,z}$, needs to be inserted as $-(A^+_{1,11}\delta_{i,x} - A^+_{i,41}\delta_{1,y} + A^+_{1,31}\delta_{i,z})$ into the first row and first column, since the first row and the first column is multiplied with $v_{1,x}$. This results in a system which can be written as:

$$\begin{pmatrix} L - T_{x,x} & -T_{x,y} & -T_{x,z} \\ -T_{y,x} & L - T_{y,y} & -T_{y,z} \\ -T_{z,x} & -T_{z,y} & L - T_{z,z} \end{pmatrix} \begin{pmatrix} \mathbf{v_x}' \\ \mathbf{v_y}' \\ \mathbf{v_z}' \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \tag{2.21}$$

where $T_{x,x}$ is a matrix containing the all indices of $\tilde{T}_k$ with $k \in V$, which are multiplied with $\mathbf{v_x}'$ to receive x-coordinate of δ'_k, and vice-versa for $T_{x,y}, T_{x,z}$ etc. Including of the positional constraints modifies the system to:

$$\underbrace{\begin{pmatrix} L - T_{x,x} & -T_{x,y} & -T_{x,z} \\ -T_{y,x} & L - T_{y,y} & -T_{y,z} \\ -T_{z,x} & -T_{z,y} & L - T_{z,z} \\ C_i & 0 & 0 \\ 0 & C_i & 0 \\ 0 & 0 & C_i \\ & \vdots & \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} \mathbf{v_x}' \\ \mathbf{v_y}' \\ \mathbf{v_z}' \end{pmatrix}}_{V} = \underbrace{\begin{pmatrix} 0 \\ 0 \\ 0 \\ \omega_i c_{i,x} \\ \omega_i c_{i,y} \\ \omega_i c_{i,z} \\ \vdots \end{pmatrix}}_{b} \tag{2.22}$$

where C_i is a row vector containing only zeros except ω_i at the ith index corresponding to the constraint $\mathbf{c_i}$. The size of this system matrix and the right hand side vector are $(3|V| + 3|C|) \times 3|V|$ and $(3|V| + 3|C|)$ respectively. Solving this system in a least-square sense using the *pseudo-inverse* leads to the solution and reconstruction of the edited mesh including the linear transformation and is equivalent to minimizing Equation (2.18). This technique leads to almost rotation-invariant Laplacian coordinates. However, for large rotations artefacts still remain due to the linearisation of the rotations [BS08]. Furthermore, as this system is similar to Laplacian System in Section 2.4.2, it suffers from the same numerical problems to condition numbers.

3 Laplacian Represenations of Temporal Sequences

This chapter explains how one can apply Laplacian Mesh Editing to animations and temporal sequences in order to maintain the spatial and temporal relationships of different interacting joints. Therefore this chapter deals with the problem how one can generate a mesh given the data of a temporal sequence.

3.1 Creation of the Mesh

In the following two different related approaches to create a mesh given an animation will be presented and explained. Ho et al. proposed to create an interaction meshes for every animation frame individually using *Delaunay tetrahetralization* [HKT10], while Naour et al. proposed to use an *3D+t graph* created from the skeleton over time connecting the frames to a large graph [NCG13].

3.1.1 Interaction Mesh

Given a set of animation frames and assuming that the interacting persons are given as rigid skeletons, one can represent the posture of the interacting persons via the position of their joints. The idea for using the joint positions instead of using joint angles is because the Laplacian coordinates are based on absolute coordinates. If one considers the joint positions as a sets of points one can apply *Delaunay tetrahetralization* to create a mesh out of the point cloud.
Delaunay tetrahetraliazation is a method from *'Computational' Geometry* and creates a tetrahetralization $DT(P)$ given a set of points $P \in \mathbb{R}^3$ such that no point $p \in P$ is inside the cirumsphere of any tetrahedron $t \in DT(P)$. The advantage of this method is that the resulting grid is quite regular, as *Delaunay tetrahetralization* maximizes the smallest angle of all angles of each tetrahedron in the grid to hold latter requirements. However, Ho et al. do not only use the joint positions as the input, they also include virtual vertices, sampled from the bones of the skeleton. This approach was inspired by Shi et al., who used so called tetrabones in their approach [SZT+07]. The *Interaction mesh* is the result of the *Delaunay tetrahetralization* of each individual frame, using the position of the joints and the virtual verticies. (depicted in Figure 3.1) Hence each frame possesses a distinct *interaction mesh*, which is used to keep the spatial relationship of the interaction persons by minimizing the Laplacian deformation energy of all individual *interaction meshes*.

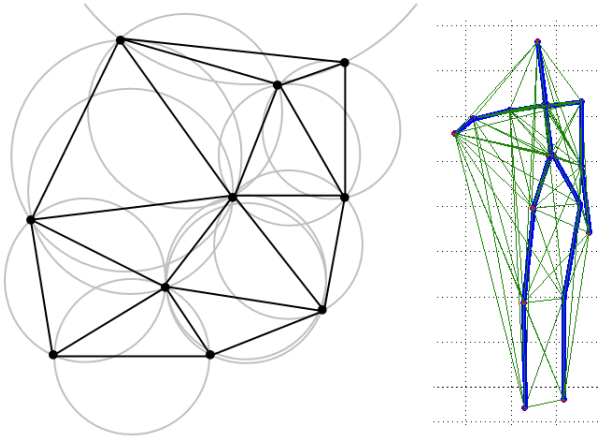


Figure 3.1.: Examples of grids resulting from *Delaunay Tetrahetralization*
left : a *Delaunay Tetrahetralizaion* of a 2D example, shown with the circumcircles of the triangles
right : a *Delaunay Tetrahetralizazion* of a frame from a handing over movement

3.1.2 3D+t Graph

Instead of using *Delaunay Tetrahetralization*, Naour et al. propose a different method to create the mesh given an animation. Let S be a sequence of skeletons with $S = (S_1, \ldots, S_m)$, where each skeleton S_k is defined as the skeleton of the kth frame of the animation by $S_k = (\mathbf{V_k}, \mathbf{E_k})$, with $\mathbf{V_k} = (\mathbf{v_{1,k}}, \ldots, \mathbf{v_{n,k}})$ containing the joint positions of the kth frame and $\mathbf{E_k}$ containing the edges of the kth frame. Naour et al. choose as edges for $\mathbf{E_k}$ the bones of the skeleton structure and the temporal connection of the joints between adjacent skeletons in time. The structure $G = (\mathbf{V}, \mathbf{E})$ with $\mathbf{V} = \{\mathbf{V_k}\}$ and $\mathbf{E} = \{\mathbf{E_k}\}$ containing the spatial and temporal relationships of the skeleton of each frame is called the *3D+t Graph*, as shown in Figure 3.2.
Thus, the Laplacian coordinates of the *3D+t Graph* depend on the spatial and temporal relationships and are calculated as following:

$$\delta_{\mathbf{i,k}} = L(\mathbf{v_{i,k}}) = w_{i,k}\Big(\sum_{\mathbf{v_{j,k}} \in N_k(\mathbf{v_{i,k}})} w_{ij,k}(\mathbf{v_{i,k}} - \mathbf{v_{j,k}}) + w_k^-(\mathbf{v_{i,k}} - \mathbf{v_{i,k-1}}) + w_k^+(\mathbf{v_{i,k}} - \mathbf{v_{i,k-1}})\Big) \tag{3.1}$$

with w_k^- and w_k^+ as special weights for the temporal relationship between the frames and $N_k(\mathbf{v_{i,k}})$ containing all the neighbour joints of joint $v_{i,k}$ of the kth frame.

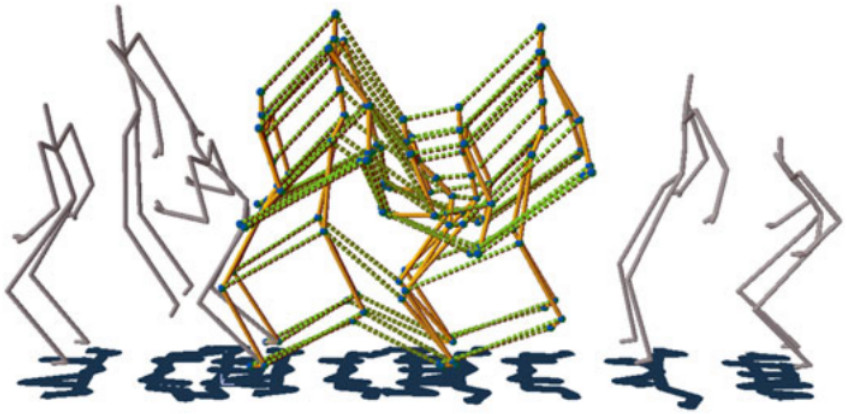


Figure 3.2.: Example of a *3D+t Graph* of a reference movement(grey). Blue points represent the joints / geometrical information. The connectivity of the graph is represented by the green dashed and yellow lines. The yellow edges are extracted from the skeleton and the green dashed line are created from the temporal relationship of the joints. [NCG13]

3.2 Bone-length Constraints

As Laplacian Mesh Editing applied to the meshes described in Section 3.1 do not always preserve the distance between the vertices, it is necessary to introduce bone length constraints. Therefore an additional energy term is needed, which enforces the bone lengths in each time step of the sequence. An other advantage of this bone length energy term is by choosing a different desired bone length than the bone length of the original skeleton, one can retarget the synthesized motion to different skeleton proportions, e.g. longer arm size or shorter leg size. The naive approach for the bone constraints would be following energy :

$$E_b(\mathbf{V'}) = \sum_{e(i,j) \in B} (||\mathbf{v'_i} - \mathbf{v'_j}|| - l_{e(i,j)})^2 \tag{3.2}$$

with $e(i,j)$ as a bone between two adjacent joints $\mathbf{v_i}$ and $\mathbf{v_j}$ of the same time step, B as a set containing all the bones for all time steps and $l_{e(i,j)}$ as the desired bone length for the bone $e(i,j)$. This leads to following energy function for Laplacian Mesh Editing applied to time sequences and animations.

$$\begin{aligned} E(\boldsymbol{V'}) &= w_l E_l(\boldsymbol{V'}) + w_p E_p(\boldsymbol{V'}) + w_b E_b(\boldsymbol{V'}) \\ &= w_l \sum_{i=1}^{|V|} ||\mathbf{L}\mathbf{v'_i} - \mathbf{T_i}(\mathbf{V'})\boldsymbol{\delta_i}||^2 + w_p \sum_{j \in C} ||\mathbf{v'_j} - \mathbf{c_j}||^2 + w_b \sum_{e(i,j) \in B} (||\mathbf{v'_i} - \mathbf{v'_j}|| - l_{e(i,j)})^2 \end{aligned} \tag{3.3}$$

with w_l as the weight for the Laplacian energy, w_p as the weight for the positional constraints and w_b as the weight for the bone length constraints. However, this energy would lead to a non-linear problem, as the bone-length constraints adds a non-linear function to the energy function E, and could not be solved using *linear-least squares* any more. For that reason, it is necessary to modify the bone-length constraints to the following energy function

$$E_b(\mathbf{V}') = \sum_{e(i,j)\in B} ||(\mathbf{v_i'} - \mathbf{v_j'}) - \frac{(\mathbf{v_i} - \mathbf{v_j})}{||(\mathbf{v_i} - \mathbf{v_j})||} l_{e(i,j)}||^2 = \sum_{e(i,j)\in B} ||(\mathbf{v_i'} - \mathbf{v_j'}) - l(v_i, v_j))||^2 \tag{3.4}$$

and solve the minimization of the energy using an iterative method. This energy is a linearisation of the energy defined in Equation (3.2) and was originally proposed by Weng et al. [WXW+06] and was later adapted by Huang et al. [HSL+06] , by Shi et al. [SZT+07] and Ho et al. [HKT10]. Using this bone constraint one can solve the non-linear least squares problem and minimize the energy function iteratively with the Gauss-Newton method [MBT99]. Hence, we reformulate the energy minimization problem to a equation system, which is similar to Equation System (2.22), as follow :

$$\underset{V}{\operatorname{argmin}} ||\boldsymbol{AV} - \boldsymbol{b(V)}||^2 \quad \text{with} \tag{3.5}$$

$$A = \begin{pmatrix} \boldsymbol{L_T} \\ \boldsymbol{C} \\ \boldsymbol{H} \end{pmatrix}, \boldsymbol{b}(V) = \begin{pmatrix} 0 \\ \boldsymbol{U} \\ \boldsymbol{l(V)} \end{pmatrix}$$

where $\boldsymbol{L_T}$ is the Laplacian Matrix combined with the implicit Transformation, $\boldsymbol{C}$ the positional constraint matrix, $\boldsymbol{U}$ the vector containing the desired positional constraints, $\boldsymbol{H}$ the bone constraint matrix and $\boldsymbol{l(V)}$ a vector containing the desired bone lengths and the bone lengths of the actual iteration step. Weng et al. propose to solve the problem utilizing iterative Gauss-Newton method, as it solves the problem in the following way :

$$\underset{V^{k+1}}{\operatorname{argmin}} ||\boldsymbol{AV^{k+1}} - \boldsymbol{b(V^k)}||^2 \tag{3.6}$$

where $\boldsymbol{V^k}$ is the solution containing the reconstructed vertices solved from the k-th iteration [WXW+06]. Since $\boldsymbol{b(V^k)}$ is known from the current iteration, one can solve the minimization using *linear-least squares* :

$$\boldsymbol{V_{k+1}} = (\boldsymbol{A^T A})^{-1} \boldsymbol{A^T b(V^k)} \tag{3.7}$$

During each iteration $\boldsymbol{b(V^k)}$ is calculated from the current solution $\boldsymbol{V^k}$, ergo one needs to calculate $\boldsymbol{l(V^k)}$ respectively its entries, which are defined as:

$$l(v_i^k, v_j^k) = \frac{(\mathbf{v_i^k} - \mathbf{v_j^k})}{||(\mathbf{v_i^k} - \mathbf{v_j^k})||} l_{e(i,j)}, \quad \text{for} \quad e(i,j) \in B \tag{3.8}$$

3.3 Movement Specific Interaction Mesh

In this section our own approach to create a *movement specific interaction mesh*, derived from *interaction meshes* (see Section 3.1.1) and *3D+t graph* (see Section 3.1.2) will be proposed and explained. The main ideas of the *movement specific interaction mesh* are the addition of temporal links to the *interaction meshes* and the adaptation of an generic interaction mesh to only encode the interaction specific spatial and temporal relationships of a given interaction.

Inspired from the *3D+t graph*, the temporal relationship is added to the *movement specific interaction mesh* by linking the individual *interaction meshes* of each frame with each other. To achieve this, for each joint of each time step an edge between the joint and the same joint of the previous frame respectively the next frame is added to the mesh. (e.g. the right hand joint of the kth time step will be linked to the right hand joint of the $k-1$th and $k+1$th time step)

The adaptation process is attained by training, where an initial generic mesh, generated from a reference movement, is adapted to a given sample of a specific movement by removing(pruning) and adding(growing) edges to the mesh. For this purpose pruning and growing is done in an optimization with *steepest ascent hill climbing* using an distance function to evaluate the performance of the modified mesh. Thus the adaptation consist out of two phases : *initialisation phase* and *optimisation phase*.

After a *movement specific interaction mesh* is adapted to a given sample, Laplacian Mesh Editing can be applied on the mesh to synthesise new motions by solving Equation (3.5) given a skeleton topology as bone constraints and a target as positional constraints.

3.3.1 Initialization Phase

In order to create the initial *interaction mesh* from the reference movement, we use two different types of generic *interaction meshes*. The first type, is an *interaction mesh* with initial edges between the two persons depicted in Figure 3.3. This mesh is created by applying *Delaunay tetrahetraliazion* on the joint positions of the first time step of both agents. The second type is an *interaction mesh* without any edges between the two persons depicted in Figure 3.3. This mesh is created by merging two distinct meshes, which are created by *Delaunay tetrahetralization* applied on the joint positions of the first time step of one agent. To incorporate the temporal relationship of the initial mesh in both cases, the *interaction meshes* of each individual time step are linked with each other by connecting each joint to the same joints of the previous time step and the next time step.

Formally the initial generic mesh $M = (\boldsymbol{V}, \boldsymbol{E})$ is defined as follow. Let $\boldsymbol{V_k} = \{\mathbf{v_{1,k}}, \ldots, \mathbf{v_{n,k}}\}$ be a set containing the joint positions of the kth frame, $\mathbf{v_{i,k}} \in \mathbb{R}^3$ be the position of one joint of the kth frame, n be the number of joints of the skeleton, $\boldsymbol{E_d}$ be the edges from the resulting grid of *Delaunay tetrahedralization* and $\boldsymbol{E_{t,k}} = \{e(\mathbf{v_{1,k-1}}, \mathbf{v_{1,k}}), e(\mathbf{v_{1,k}}, \mathbf{v_{1,k+1}}), \ldots, e(\mathbf{v_{n,k-1}}, \mathbf{v_{n,k}}), e(\mathbf{v_{n,k}}, \mathbf{v_{n,k+1}})\}$ be the edges from the temporal relationship of the joints of the kth frame. Then $\boldsymbol{V} = \{\boldsymbol{V_k}\}$ and $\boldsymbol{E} = \{\boldsymbol{E_k}\}$ with $1 \leq k \leq m$, where $\boldsymbol{E_k} = \{\boldsymbol{E_d}, \boldsymbol{E_{t,k}}\}$ and m as the total frame count of the animation.

3.3.2 Optimization Phase

In the optimization phase the initial generic mesh M of latter section is adapted using *steepest ascent hill climbing*. *Hill climbing* is a iterative local search optimization technique. It attempts to maximize or minimize a target function $f(x)$, where x is a vector of continuous or discrete values. At each iteration step *steepest ascent hill climbing* computes all possible neighbours of the current value of x and determines the value which results in the largest or smallest $f(x)$. This x is used for the next iteration step. This process is continued until no better solution is found [RN10].
In our case x is represented by an interaction mesh $M = (V, E)$ and $f(x)$ is a distance function $d : V \times V \rightarrow \mathbb{R}$ evaluating the performance of the reconstructed result $V' = \{\mathbf{v'_1}, \ldots, \mathbf{v'_{|V'|}}\}$ of the modified mesh $M' = (V, E')$ to the given reference sample $V = \{\mathbf{v_1}, \ldots, \mathbf{v_{|V|}}\}$ of the movement to be trained. The distance function d is defined as:

$$d(V', V) = \sum_{i \in V}^{|V|} w_i ||\mathbf{v_i} - \mathbf{v'_i}|| \tag{3.9}$$

where w_i is a vertex specific weight. The neighbour $M' = (V, E')$ to a given $M = (V, E)$ is computed by removing or adding one additional edge between the two agents to E. By restricting the actions to only to growing or pruning, one can create two different variants and reduce on the same time the computation time of the algorithm.

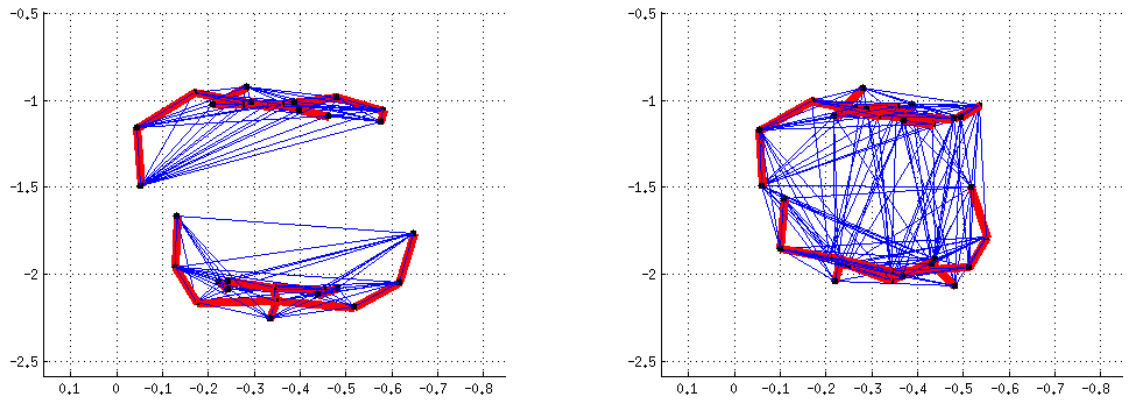


Figure 3.3.: Example of *interaction meshes* used in the initialisation phase, left : interaction mesh with initial *interperson* edges, right : interaction mesh with no initial *interperson* edges

3.3.3 Algorithm

The adaptation process of a interaction mesh to a specific motion can be described by following algorithm :

Algorithm 1 Optimization of a interaction mesh to a specific motion

Input : Skeleton and input motion sequence
Skeleton and motion sequence of reference movement
Output : Optimized specific motion interaction mesh
(**Initialization**)
- Compute vertex positions $\boldsymbol{V}$ out of given reference skeleton and motion sequence
- Compute the initial interaction mesh $\boldsymbol{M}$ (vertex positions, adjacency matrix) of the input motion sequence
(**Optimization**)
- Retarget the input motion sequence to the reference movement using $\boldsymbol{M}$ and compute $\boldsymbol{V'}$ by solving (3.5)
currentMesh = $\boldsymbol{M}$
oldEval = $d(\boldsymbol{V}, \boldsymbol{V'})$
loop
  nextEval = Inf
  nextMesh = NULL
  - Compute list $\boldsymbol{L}$ of all neighbour meshes of currentMesh (growing / pruning edges)
  for all Meshes $\boldsymbol{M'}$ in $\boldsymbol{L}$ **do**
    - Retarget the input motion sequence to the reference movement using $\boldsymbol{M'}$ and compute $\boldsymbol{V'}$ by solving (3.5)
    eval = $d(\boldsymbol{V}, \boldsymbol{V'})$
    if eval < nextEval **then**
      nextEval = eval
      nextMesh = $\boldsymbol{M'}$
    end if
  end for
  if oldEval < nextEval **then**
    // break loop and return currentMesh as there are no better neighbours
    break
  end if
  currentMesh = nextMesh
end loop
return currentMesh

3.3.4 Variants

There are two different variants of the algorithm: *growing* and *pruning*. The idea of the *pruning* is to start with a initial mesh, which contains already numerous *interperson* edges and to remove all interperson edges from the initial mesh, that are not decisive for the given motion. Thus *pruning* uses the first type of the proposed initial interaction meshes and restricts the the computation of the neighbours only to removing interperson egdes. *Growing* is based on the opposite idea, starting with a initial mesh, which contains no *interperson* edges and adding all interperson edges to the initial mesh, that are decisive for the given motion. Thus *growing* uses the second type of the proposed initial interaction meshes and restricts the computation of the neighbours only to adding interperson edges.

4 Experiments and Results

4.1 Data Acquisition

For the experiments it was necessary to capture animations / motion capture data in order to train *interaction meshes* and apply Laplacian Mesh Editing on them. Therefore a low cost Microsoft Kinect camera was used to capture interactions of two human subjects utilizing the OpenNI framework [org11]. As the NITE skeleton tracking system, included in OpenNI [Inc11], is able to track up to 15 different joints the skeleton of the captured data has the following structure and 45 degrees of freedom:

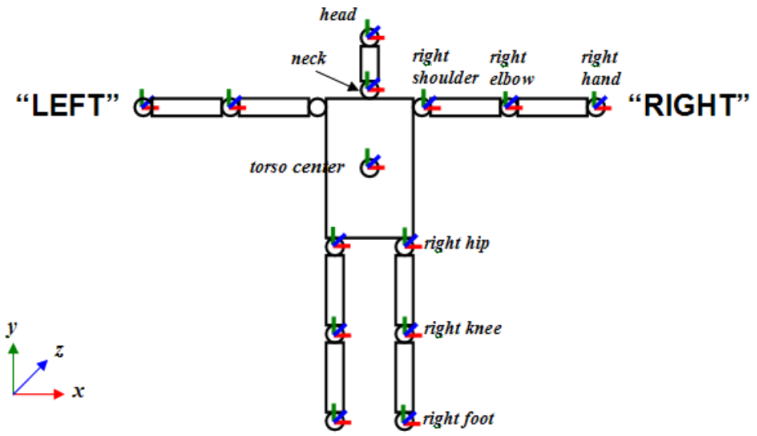


Figure 4.1.: Skeleton structure and joint definition of the motion capture data using OpenNI and NITE skeleton tracking system, note: the skeleton front side is seen here in this figure

For this work two different types of interactions between two person were captured with different variations: boxing and handing over of an object. Different variations are for example handing over the object more to the left or the right or above or below. In total 56 different samples were captured. During the capturing process a smoothing factor of 0.55 and a factor of 0.5 for the minimum confidence of the joints were used as initial parameter in OpenNI, in order to deal with the problem of varying bone lengths received from the NITE skeleton tracking system between different frames. As the NITE skeleton tracking system has large difficulties with tracking persons standing sidewards to the Kinect, the interacting persons had to face the Kinect directly and perform the movement in direction of the camera, instead of performing the movement towards the other person. Hence the recorded movement had to be adapted by moving one of the interacting person in front of the other and mirroring his movement to match the intended interaction.

4.2 Experiments and Results of Motion Specific Interaction Meshes

In this section results of several experiments on motion specific interaction meshes will be presented. In the subsequent experiments motion specific interaction meshes are trained with reference motions to different specific movements and afterwards generalized to different samples of the same movement. Finally, the generalized result is compared to the original samples using the distance function specified in Section 3.3. For the reconstruction of the movements via Laplacian Mesh Editing the parameters $w_b = 3.0$, $w_p = 1.5$, $w_l = 1.0$ were used as weights for the individual energies (see Equation (3.3)). For the distance function the weight $w_i = 1.0$ was used for all vertices.

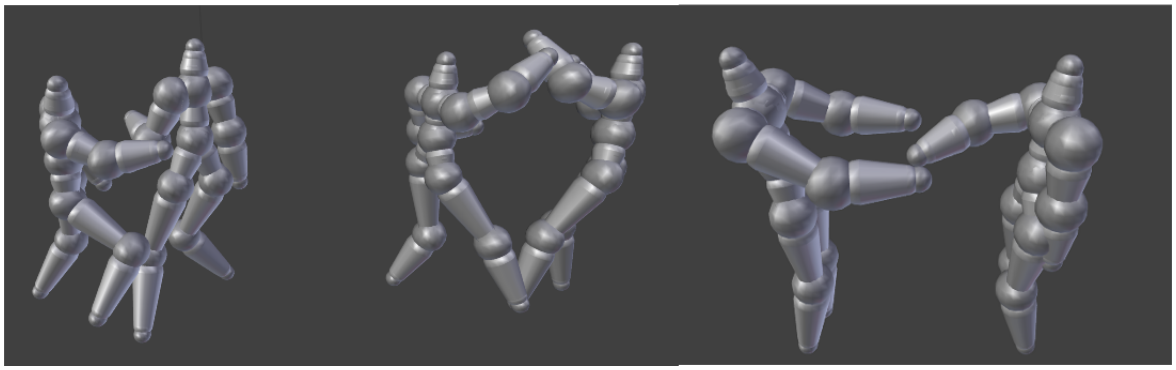
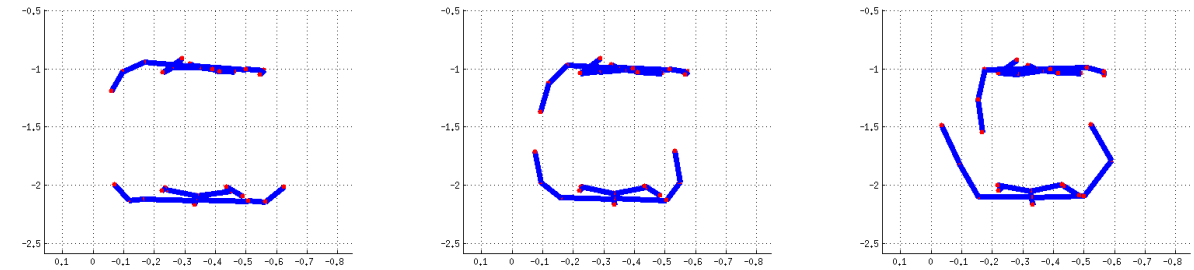


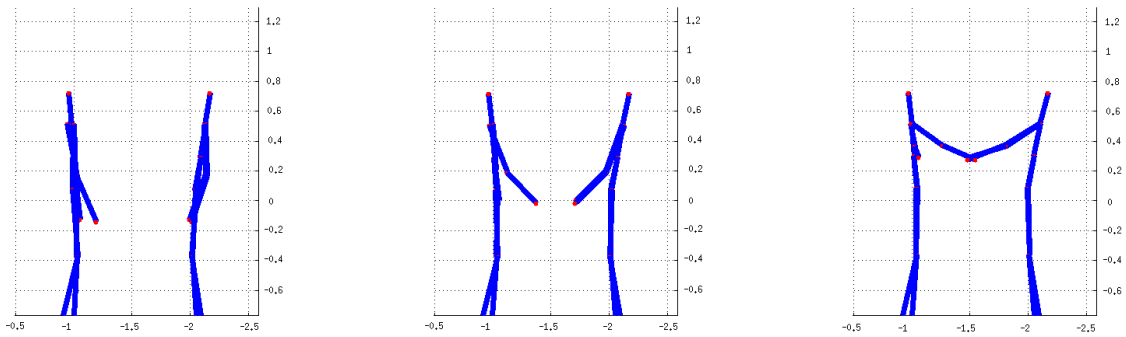
Figure 4.2.: Three different captured samples visualized in blender, left : *punch middle*, middle : *punch up*, right : *hand over*

4.2.1 Scenario : Handing Over

This scenario investigates a scene, in where two persons are interacting with each other by handing over an object. One person, referred as the '*giver*', is handing over an object with the right hand, while the other person, referred the '*receiver*', takes over the object with both hands, as depicted in Figure 4.3. For more detailed visualizations of handing over movements, we refer the reader to appendix part A.
The handing over scenario was chosen for the following experiments, as it is crucial for the semantic and the success of the handing over movement to maintain the spatial and temporal relationships between the hands and different body parts of the interacting persons. Thus this scenario is an sufficient difficult task for the motion specific interaction meshes. Moreover, the scenario is a very common interaction between collaborating humans.



(a) Frame 1 **(b)** Frame 30 **(c)** Frame 60

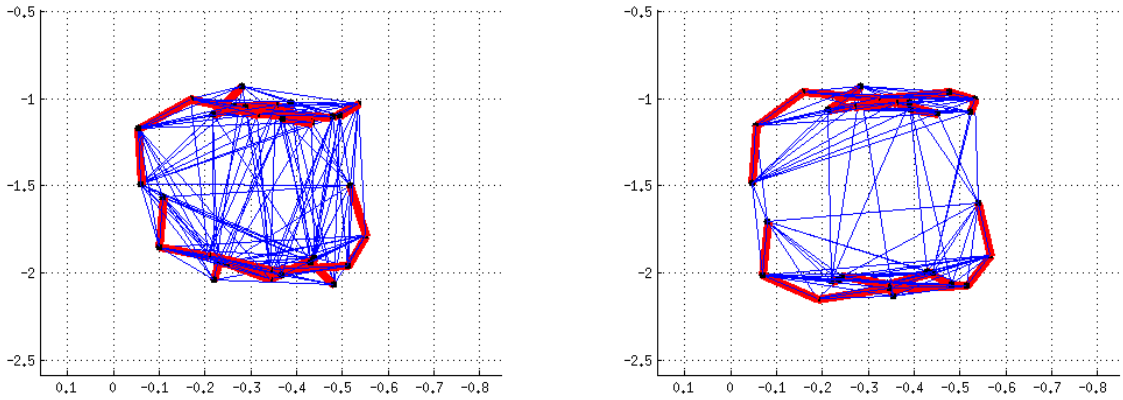


(d) Frame 1 **(e)** Frame 30 **(f)** Frame 60

Figure 4.3.: Decisive frames of a handing over movement (Sample 2), first row : view from above, second row : view from side, blue lines represent the bones of the interacting persons, while red dots represent the joints

4.2.2 Training using Pruning

In this experiment the variant *pruning* of our algorithm 1 of Section 3.3.3 was used to train an initial mesh to a specific handing over movement. Sample 2 was used as the input movement, while sample 1-5 were used as reference motions for different test cases. After training the result was applied on other different samples to reconstruct adapted motions.



(a) before pruning **(b)** after pruning

Figure 4.4.: Comparison between an untrained initial interaction mesh (left) and a trained movement specific interaction mesh using pruning (right), blue lines are the edges of the interaction mesh representing spatial relationship between the joints, shown in black, while the bones of the skeleton are shown in red

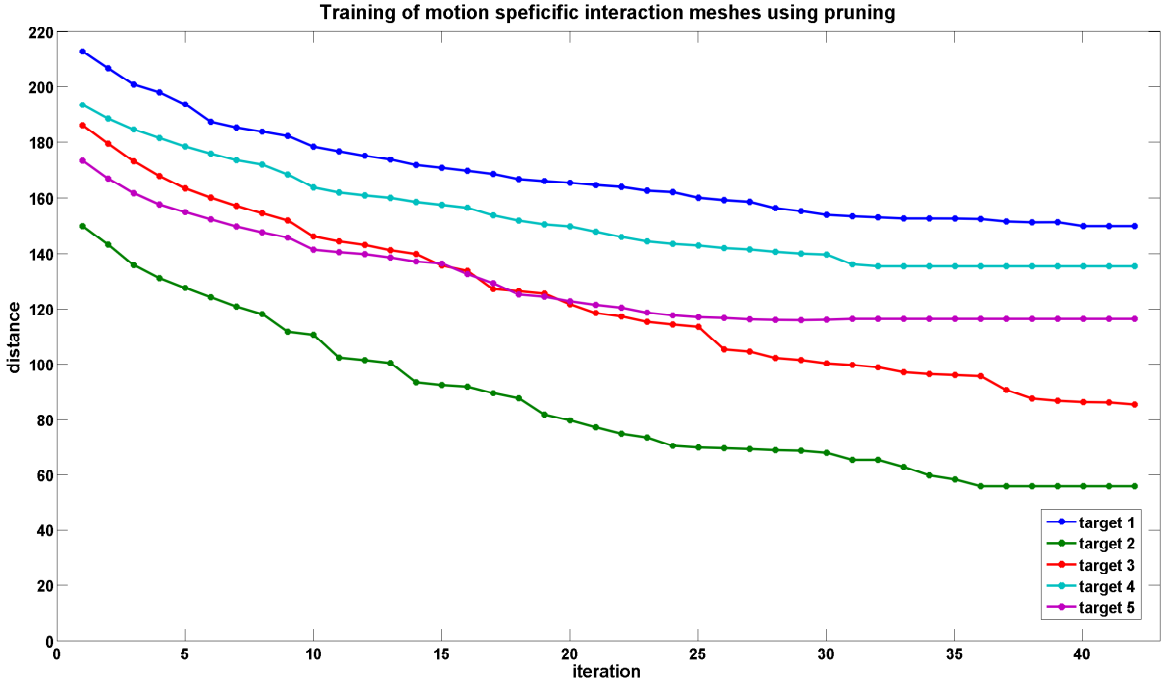


Figure 4.5.: Convergence and improvement of movement specific interaction meshes using pruning per optimization step corresponding to the distance distance between the input motion (sample 2) and the reference motion (target 1 to 5)

Figure 4.4 shows the difference between an initial *interaction mesh* and a trained *movement specific interaction mes*h, resulting from training the latter mesh on sample 1. For more visualisations of other samples we refer the reader to appendix part A. It is noticeable that numerous unnecessary edges are removed while few edges between

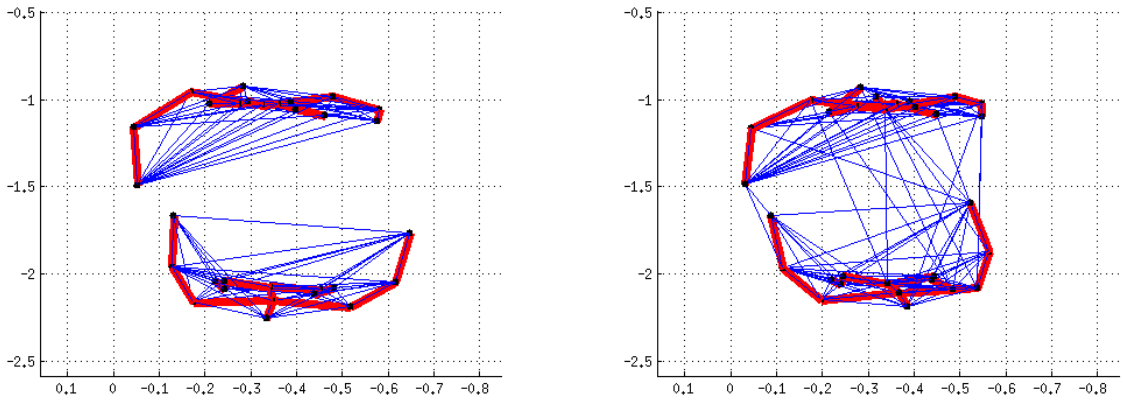
Training	Reconstruction				
	Target 1	Target 2	Target 3	Target 4	Target 5
no training (initialisation)	212.6802	151.9322	186.0163	193.3583	173.1756
Target 1	**149.9261**	62.6319	105.7444	145.6166	115.6867
Target 2	163.1939	**55.9176**	106.0026	148.8157	124.2817
Target 3	177.3767	66.0906	**85.2536**	139.8456	125.9778
Target 4	172.6448	79.9069	107.4567	**135.3754**	121.7705
Target 5	167.0329	74.7379	111.6507	143.1870	**116.3383**

Table 4.1.: Distance between reference sample and reconstructed adapted movements of different samples using movement specific interaction meshes trained to different reference movements by pruning (Sample 2 trained to Target 1-5), bold values are the results from the training process to the specific target, non-bold values are results from generalization using the learned interaction mesh to synthesise movement to other targets

the hands, arms, the torso and legs of the giver and the receiver are retained. If one compares this result to the other results off training on samples 2-5, a common trend in the remaining edges can be seen, as edges between the hands, arms, torso, and the legs of the giver and receiver tend to remain.
Figure 4.5 shows on the one hand the convergence of the method and on the other hand the improvement of the performance in each training step of the *movement specific interaction meshes*. It is observable, that after about 40 iterations the pruning approach converges, as there are almost no changes in the distance any more. Convergence is given in this approach, since at some point only edges will remain, which are important for the movement to be trained. Pruning them would increase the distance instead of reducing it.
Table 4.1 shows the result of generalizing a trained *movement specific interaction mesh* to other targets, denoted by the value of the distance function. Moreover the figure indicates two import aspects: First, trained *movement specific interaction meshes* perform on their target almost always better than all other generalized reconstructed movements. Second, it shows that the distances of the generalized reconstructed movements are always lower than the result synthesized via untrained *interaction meshes* and are all in the same range of the optimal reconstructed movements.

4.2.3 Training using Growing

In this experiment an initial mesh was trained to a specific handing over movement using the growing variant algorithm 1 of Section 3.3.3. Sample 2 was used as the input movement and samples 1-5 were used as reference motions for different test cases. After training the result was applied on other different samples to reconstruct adapted motions.
Similar to the first experiment Figure 4.4 shows the improvement from an untrained initial *interaction mesh* to a trained *movement specific mesh*. In this case it is observable that only a few important edges between the hands, arms and the torso of the giver and the receiver are added to the initial mesh. Looking at the results from training the same mesh on the other samples 2-5, one can see similarities between the added edges. There is a trend to connect the right arm of the receiver with the right arm of the giver and to connect the feet and the torso with each other. For the visualization of the other results we refer the reader to the appendix part 1.
Figure 4.7 shows the convergence of the growing approach, since at some point all edges, which are important for the trained movement, are added to the mesh. Adding more edges at this point would only increase the distance instead of reducing it. It is noticeable that convergence is already reached after about 10 iterations.
Table 4.2 shows the results of generalization of the adapted *interaction meshes* to other targets, denoted in the distance value. The table indicates almost analogous results for growing. However, trained *movement specific interaction meshes* perform on their trained always better than all other generalized reconstructed movements.



(a) before growing **(b)** after growing

Figure 4.6.: Comparison between an untrained initial interaction mesh (left) and a trained movement specific interaction mesh using growing (right), blue lines are the edges of the interaction mesh representing spatial relationship between the joints, shown in black, while the bones of the skeleton are shown in red

Another difference to pruning is that generalized results are closer to the reconstructed results of the untrained *interaction mesh* and differ more from the optimal result.

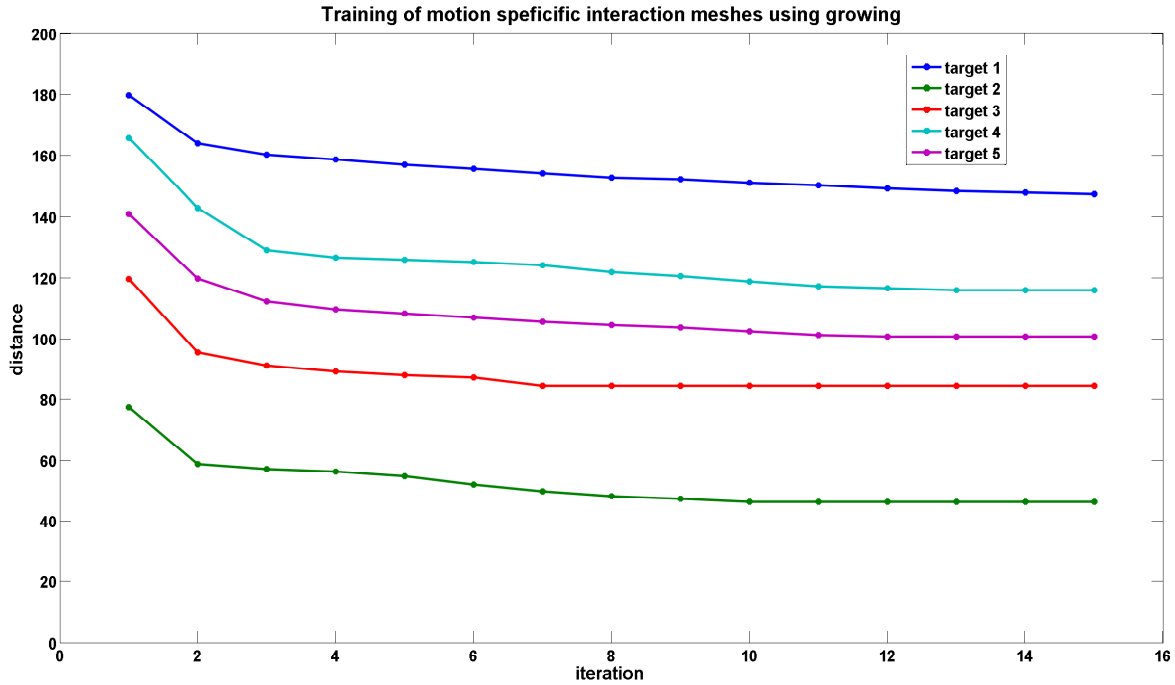


Figure 4.7.: Improvement of the trained movement specific interaction mesh using growing per optimization step corresponding to the distance distance between the input motion (sample 4) and the reference motion (target 1 to 5)

4.2.4 Discussion

Concerning the difference of both approaches, it is noticeable that growing leads to a much faster convergence, as less edges need to be altered than in the pruning approach. However, one iteration step in growing takes more computation time than pruning, because there are much more possibilities to add new edges between the two interacting persons than edges to remove in the pruning approach. Thus, in the end, the computation effort is quite

Training	Reconstruction				
	Target 1	Target 2	Target 3	Target 4	Target 5
no training (initialisation)	179.5994	77.3663	119.5287	165.5458	140.8939
Target 1	**147.2350**	66.5611	122.5619	153.9787	122.6171
Target 2	164.6189	**46.2863**	92.4765	137.2643	117.4041
Target 3	170.4324	61.4900	**84.4851**	132.8890	117.3768
Target 4	178.9507	74.0411	101.5868	**115.8765**	114.0387
Target 5	164.4972	62.6003	99.1190	130.9264	**100.6056**

Table 4.2.: Distance between reference sample and reconstructed adapted movements of different samples using movement specific interaction meshes trained to different reference movements by pruning (Sample 2 trained to Target 1-5), bold values are the results from the training process to the specific target, non-bold values are results from generalization using the learned interaction mesh to synthesise movement to other targets

similar in both approaches. Moreover, it is noticeable that growing leads slightly to smaller distances on the trained samples than pruning. This could indicate that growing is the better approach, as growing computes all possible *interperson* edges, while pruning only uses the internal *interperson* edges, computed by Delaunay tetrahetralization. Ideally would be a combination of both methods though, as pruning can remove interperson edges, which were added by growing in the first place ,but are performing worse after more edges were added.
Looking at the similarities, it is observable that both approaches perform well, as the distance decreases significantly for both approaches when comparing the training result. Furthermore, if we take a look at the order of the targets how good they perform, we can see that the order is the same for both approaches. (e.g. Target 2 shows the best performance, Target 1 the worst, etc.) Hence, it can be assumed, that the difficulty to adapt a movement is not significantly influenced by the pruning or the growing approach.
An other similarity is that *movement specific interaction meshes* perform on their trained target always better than all other generalized reconstructeed movement, which could indicate a case of overfitting. Thus a solution would be training on multiple samples simultaneously to prevent this phenomena.

In general the experimental results showed the effectiveness of *movement specific interaction meshes*, as they outperformed untrained generic *interaction meshes* in each sample. The results indicate the possibility to generalize from the learned *interaction mesh* and to synthesise new similar movements in different scenarios, as the distance of the generalized results tend to be in the same range of the optimal result and usually perform better than the untrained interaction meshes. Furthermore, they indicate the possibility of learning the spatial relationship of a specific movement, as the algorithm leads to quite feasible connections between the different joints of the interacting persons. Thus they are more suitable to encode the spatial and temporal relationship of a given movement than traditional interaction meshes.

However, it seems that there are motions, which can be adapted easier and motions which are more difficult to adapt. For example Target 2 has in both approaches the smallest difference while Target 1 has the largest difference. This can be explained by the fact, that some of the targets are closer to the reference motion, while others are differ more. This shows a limitation of this method. If the new scenario differ to much from the reference sample, the method performs worse, as Laplacian Mesh Editing would try to stick close to the data. A solution for this behaviour would be by proving enough samples for this method.

5 Conclusion and Future Works

In this thesis, an improved version of an *interaction mesh*, which encodes the spatial and temporal relationship of a specific movement by learning, was presented. Futhermore, it was shown how they could applied to synthesise similar interactions between two persons in different scenarios using Laplacian Mesh Editing. The core aspects of this idea are: 1. To use *Delaunay tetrahetralization* only as an initial solution for the spatial relationship between the interacting humans and learn the spatial relationship by pruning and growing the initial mesh. 2. To merge the individual *interaction meshes* to a single one by adding edges between joints of adjacent frames, which allows us to incorporate additionally the temporal relationship. In different experiments the effectiveness of the proposed version was demonstrated, as *movement specific interaction meshes* showed more adapted and realistic synthesized movements on different targets, compared to the traditional approaches of *interaction meshes*. However, the experiments also showed limitations of these *movement specific interaction meshes*, as the method performs worse when the new situation differs too much from the learned reference movement.

As future work the learning process of the *movement specific interaction meshes* could be improved by using simultaneously multiple reference motions in one optimization step or by varying the weights of the distance function, in order to improve the feedback on individual important body parts (feet, arms etc.). For the reconstruction part learning the weights of the Laplacian coordinates or the usage of cotangent weights, would be an attractive topic, as they perform better on irregular meshes than uniform weights. Another possibility would be the consideration of other mesh deformation techniques and investigate if they are usable for reconstruction on *interaction meshes* or not. Last but not least it would be interesting to extend this method to real humanoid robots, to learn interactions between two humans using *movement specific interaction meshes* and evaluate if robots could take over one part and interact with the other human counter part.

Acknowledgements

At this point, I want to thank everybody who supported me writing this thesis.

Especially, to my both supervisors Dr. Heni Ben Amor and M.Eng. Oliver Kroemer for your generous and patient support and the enlightening input I received during our Skype meetings. I am thankful to Heni, that he still continued to supervise me even after he moved to Georgia Tech. I am also grateful to Oli, as he accepted to supervise my thesis after Heni had to move to Georgia Tech.

I am grateful to Prof. Dr. Jan Peters and the Intelligent Autonomous Systems Lab for accepting my thesis and giving me the opportunity to work in such an excellent research group. I really treasure the time spend at the IAS.

I am additionally thankful to my friends and colleagues in the lab for their mental and physical support, giving me the motivation whenever I needed it. Special thanks go to Victora Fehr, for helping me correcting my spelling.

Last but not least I want to thank my parents for all the support they gave me throughout the thesis and my whole life.

6 Bibliography

[Ale03] Marc Alexa. Differential coordinates for local mesh morphing and deformation. *The Visual Computer*, pages 105–114, 2003.

[BK04] Mario Botsch and Leif Kobbelt. A Remeshing Approach to Multiresolution Modeling. *Proceedings of the 2004 Eurographics ACM SIGGRAPH symposium on Geometry processing SGP 04*, page 185, 2004.

[BS08] Mario Botsch and Olga Sorkine. On linear variational surface deformation methods. *IEEE Transactions on Visualization and Computer Graphics*, 14(1):213–230, January 2008.

[DMSB99] Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques - SIGGRAPH '99*, volume 33, pages 317–324, 1999.

[HKT10] Edmond S. L. Ho, Taku Komura, and Chiew-Lan Tai. Spatial relationship preserving character motion adaptation. *ACM Transactions on Graphics*, 29(4):1, 2010.

[HS13] Edmond S. L. Ho and Hubert P. H. Shum. Motion adaptation for humanoid robots in constrained environments. In *ICRA*, pages 3813–3818. IEEE, 2013.

[HSL+06] Jin Huang, Xiaohan Shi, Xinguo Liu, Kun Zhou, Li-Yi Wei, Shang-Hua Teng, Hujun Bao, Baining Guo, and Heung-Yeung Shum. Subspace gradient domain mesh deformation. *ACM Trans. Graph.*, 25(3):1126–1134, July 2006.

[Inc11] PrimeSense Inc. Prime sensor™ NITE 1.5 algorithms notes. `http://www.openni.org/wp-content/uploads/2013/02/NITE-Algorithms.pdf`, 2011. Accessed: 14-02-2014 14:57.

[LSA+05] Yaron Lipman, Olga Sorkine, Marc Alexa, Daniel Cohen-Or, David Levin, Christian Rössl, and Hans-Peter Seidel. Laplacian framework for interactive mesh editing. *International Journal of Shape Modeling*, 11(1):43–62, 2005.

[LSCO+04] Y. Lipman, O. Sorkine, D. Cohen-Or, D. Levin, C. Rossi, and H.P. Seidel. Differential coordinates for interactive mesh editing. *Proceedings Shape Modeling Applications, 2004.*, 2004, 2004.

[MBT99] Kaj Madsen, Hans Bruun, and Ole Tingleff. Methods for non-linear least squares problems, 1999.

[NCG13] Thibaut Le Naour, Nicolas Courty, and Sylvie Gibet. Spatiotemporal coupling with the 3d+t motion laplacian. *Journal of Visualization and Computer Animation*, 24(3-4):419–428, 2013.

[org11] OpenNI organization. Openni user guide. `http://kinectcar.ronsper.com/docs/openni/index.html`, Dezember 2011. Accessed: 14-02-2014 14:57.

[RN10] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*, chapter 4.1.1. Pearson Education, 3. edition, 2010.

[SCOL+04] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, SGP '04, pages 175–184, New York, NY, USA, 2004. ACM.

[Sor05] Olga Sorkine. Laplacian Mesh Processing. *Eurographics - State of the Art Reports*, (Section 4):53–70, 2005.

[SZT+07] Xiaohan Shi, Kun Zhou, Yiying Tong, Mathieu Desbrun, Hujun Bao, and Baining Guo. Mesh puppetry: Cascading optimization of mesh deformation with inverse kinematics. *ACM Trans. Graph.*, 26(3), July 2007.

[WXW+06] Yanlin Weng, Weiwei Xu, Yanchen Wu, Kun Zhou, and Baining Guo. 2D shape deformation using nonlinear least squares optimization. *The Visual Computer*, 22(9-11):653–660, 2006.

[ZIT+12] Dmitry Zarubin, Vladimir Ivan, Marc Toussaint, Taku Komura, and Sethu Vijayakumar. Hierarchical Motion Planning in Topological Representations. In *Proceedings of Robotics: Science and Systems*, Sydney, Australia, July 2012.

A Animations

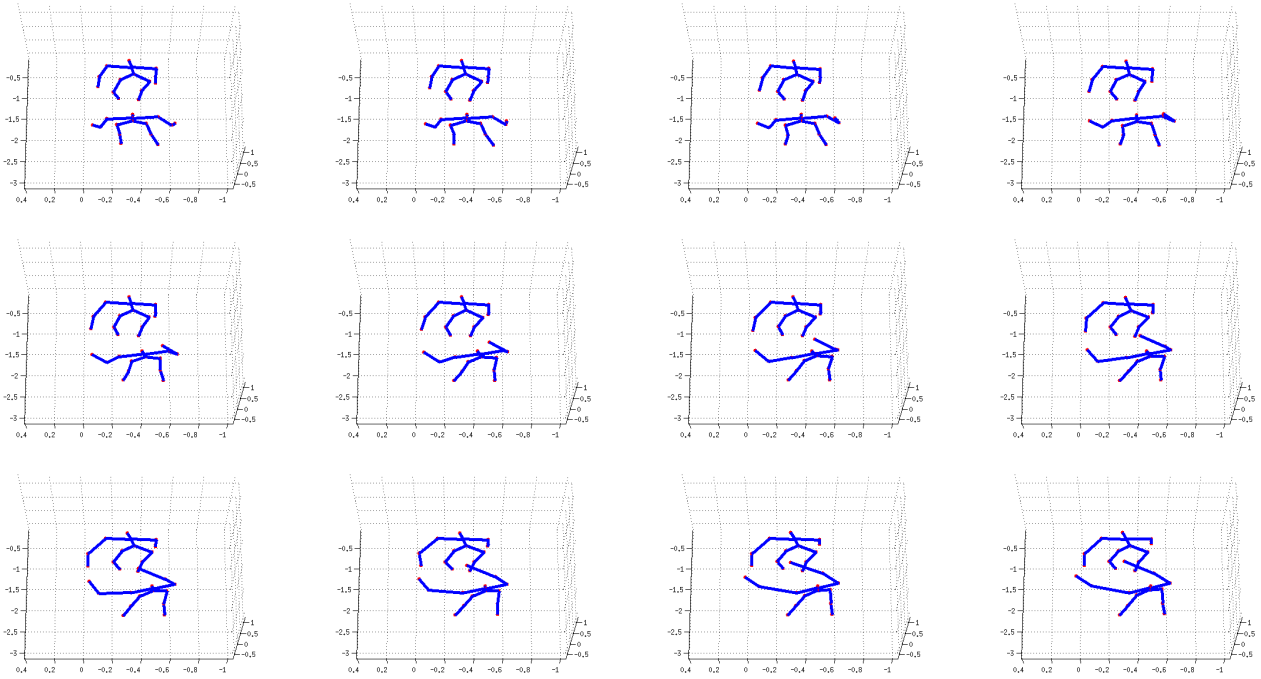


Figure A.1.: Visualisation of sample 1 (handing over movement) , view from above, Frames 1-60, successive image from left to right: 1, 5, 10, 15,...,60

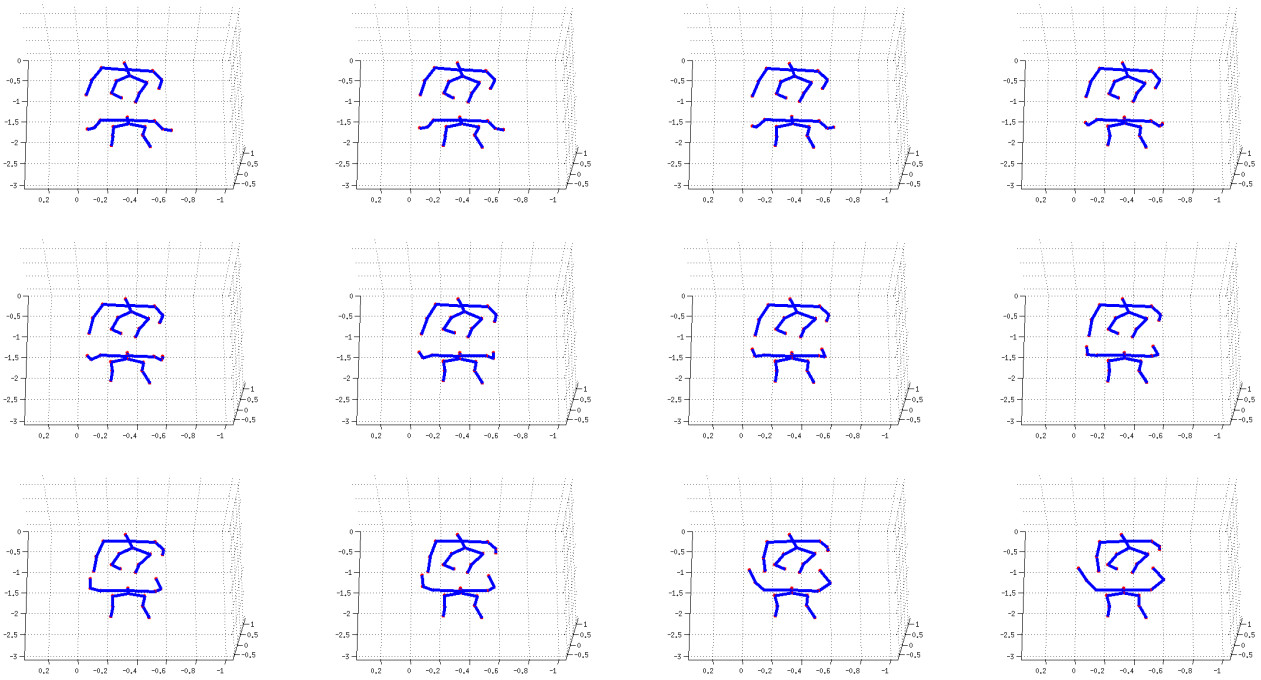
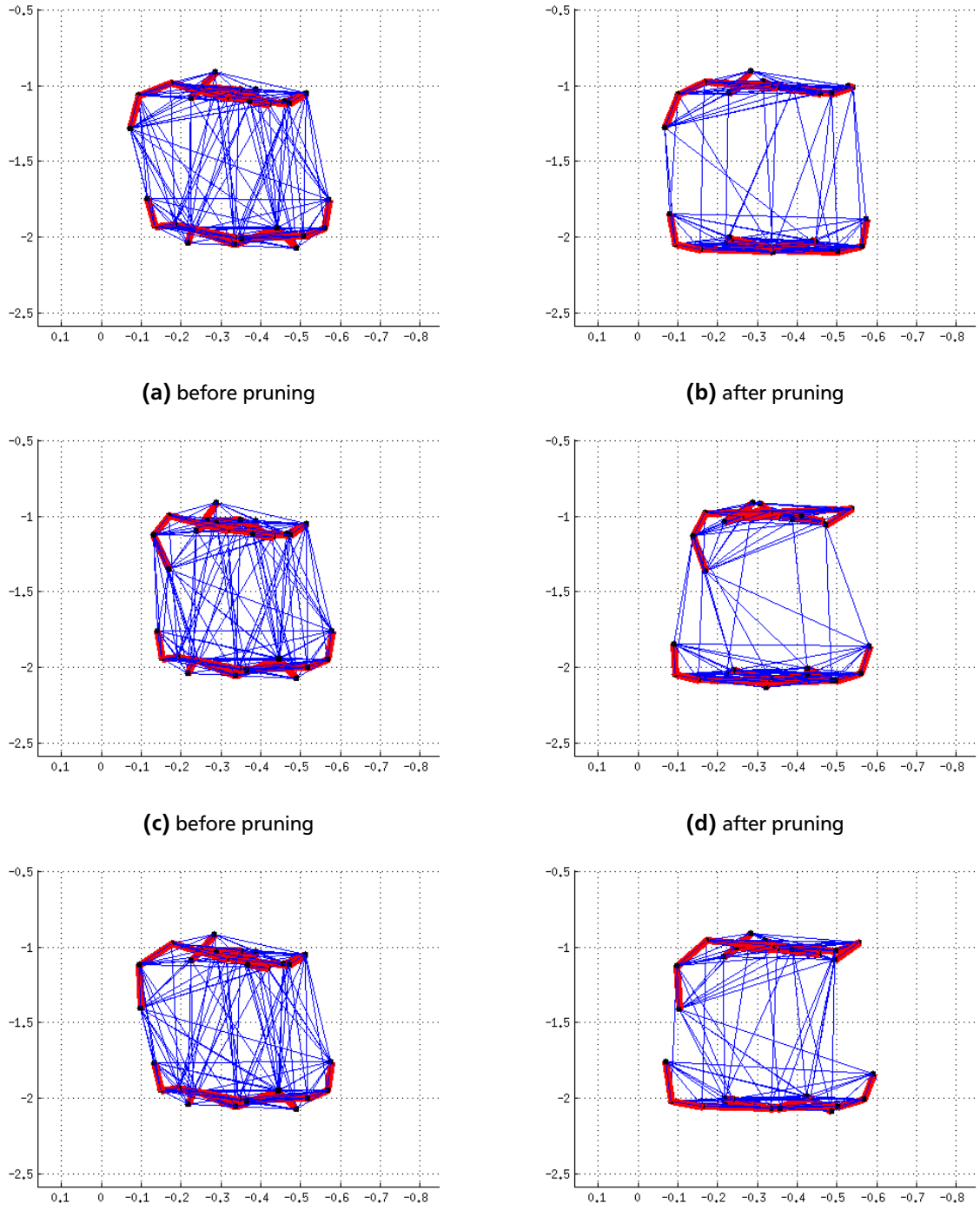


Figure A.2.: Visualisation of sample 2 (handing over movement), view from above, Frames 1-60, successive image from left to right: 1, 5, 10, 15,...,60



(a) before pruning

(b) after pruning

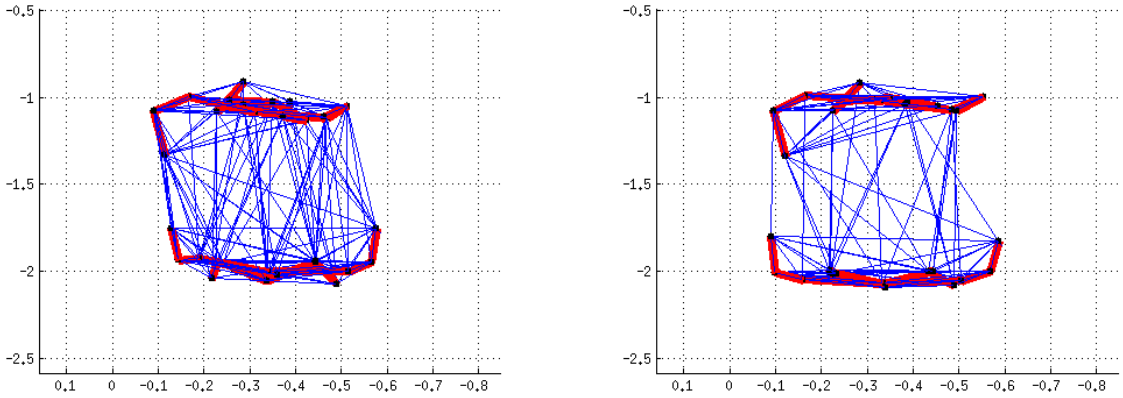
(c) before pruning

(d) after pruning

(e) before pruning

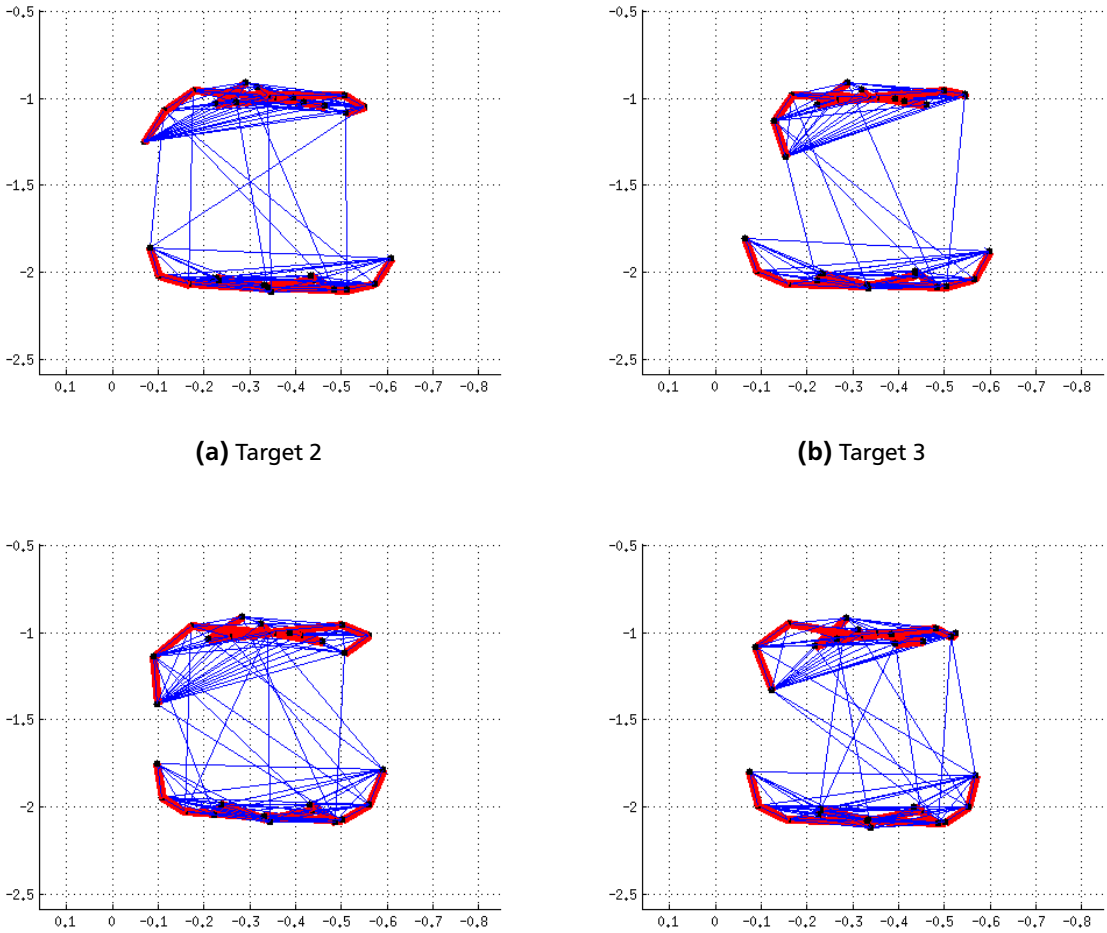
(f) after pruning

Figure A.3.: Target 2-4, Comparison between an untrained initial interaction mesh (left) and a trained movement specific interaction mesh using pruning (right), blue lines are the edges of the interaction mesh representing spatial relationship between the joints, shown in black, while the bones of the skeleton are shown in red



(a) before pruning **(b)** after pruning

Figure A.4.: Target 5, Comparison between an untrained initial interaction mesh (left) and a trained movement specific interaction mesh using pruning (right), blue lines are the edges of the interaction mesh representing spatial relationship between the joints, shown in black, while the bones of the skeleton are shown in red



(a) Target 2 **(b)** Target 3

(c) Target 4 **(d)** Target 5

Figure A.5.: Target 2-5, Trained movement specific interaction mesh using growing (right), blue lines are the edges of the interaction mesh representing spatial relationship between the joints, shown in black, while the bones of the skeleton are shown in red