HAL open science

Homeomorphic Alignment of Edge-Weighted Trees

Benjamin Raynal, Michel Couprie, Venceslas Biri

To cite this version:

Benjamin Raynal, Michel Couprie, Venceslas Biri. Homeomorphic Alignment of Edge-Weighted Trees. 7th IAPR-TC-15 International Workshop, GbRPR 2009, May 2009, Venise, Italy. pp.134-143, 10.1007/978-3-642-02124-4_14 . hal-01626326

HAL Id: hal-01626326

https://hal.science/hal-01626326v1

Submitted on 30 Oct 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Homeomorphic Alignment of Edge-Weighted Trees

Benjamin Raynal, Michel Couprie, and Venceslas Biri

Université Paris-Est
Laboratoire d'Informatique Gaspard Monge, Equipe A3SI
UMR 8049 UPEMLV/ESIEE/CNRS

Abstract. Motion capture, a currently active research area, needs estimation of the pose of the subject. This requires a match between a model and the 3D shape, constructed using a multiview system. Our purpose is to realize it in real-time, using the tree representation of the skeleton of the 3D shape. In this paper, we propose a new alignment distance between both rooted and unrooted weighted trees, taking in account the different types of noise occuring in the data tree. Then, we develop several algorithms with acceptable time complexity for our purpose.

Key words: Graphs, homeomorphism, aligment, matching algorithm

1 Introduction

Motion capture without markers is a highly active research area, as shown by Moeslund and al. [1]:between 2000 and 2006, more than 350 papers on this topic were published. One difficulty of motion capture consists in finding the initial pose of the subject, represented by a 3D shape and constructed using a multiview system. Therefore, our motivation is to match the different parts of this 3D shape (the data) to a simple a priori model.

The model is an unrooted weighted tree (called the *pattern tree*), where vertices represent the different parts of the shape, and edges model the links between these parts. Edges are associated to a weight representing the distance between the parts. Concerning the data, we extract the curve skeleton of the 3D shape, and compute the associated weighted unrooted tree (called the *data tree*), by considering each multiple point and ending point, and linking them when they are directly connected. The weight of an edge is the geodesic distance between its vertices.

After this step, the main difficulty is to match the pattern tree in the data tree, with a good preservation of both topology and distances.

Several approaches have been developed, using the skeleton of a shape, in motion capture research area [2–4], and in 3D shape matching research area [5–7]. The best time obtained for finding the initial pose is one second [3], which is too slow, even for interactive time.

Moreover, several kinds of noise and deformities can appear in the data tree :

Spurious branches. Due to the skeletonization algorithm and to the amount of noise of the shape surface, branchs of skeleton can appear, but without important topological signification. The method must be robust enough to work on data trees with consequent amount of spurious branches.

Useless vertex. Vertices with exactly two neighbors are not useful to describe the topology of a shape, they uselessly split an edge (and its weight) in two parts, making difficult a good matching. This kind of vertices can appear when removing spurious branches. The method must be able to match two edges joined by this kind of vertex, with a unique edge.

Splitted vertex. Vertices with more than 3 neighbors in the pattern tree can correspond to a cluster of vertices linked by weakly weighted edges in the data tree, due to the skeletonization algorithm. The method must be able to match them.

Approaches found in the literature (see Sect. 3) do not permit to achieve a robust matching, with respect to these pertubations. In the following, after reviewing basic notions, we introduce both a new alignment, called homeomorphic alignment, and a robust tree-matching algorithm which may be used for realtime pose estimation.

2 Basics Notions

Undirected Graphs. An *undirected graph* is a pair (V, E), where V is a finite set, and E a subset of $\{\{x, y\}, x \in V, y \in V, x \neq y\}$. An element of E is called an *edge*, an element of V is called a *vertex*. If $\{x, y\} \in E$, x and y are said to be *adjacent* or *neighbors*. The set of all neighbors of x is denoted by $\mathcal{N}(x)$. The number of vertices adjacent to a vertex v is called the *degree* of v, and is denoted by $deg(v)$. Let $G = (V, E)$ be an undirected graph, and let x, y be in V, a *path* from x to y in G is a sequence of vertices $s_0, ..., s_k$ such that $x = v_0$, $y = v_k$ and $\{v_{i-1}, v_i\} \in E, 1 \leq i \leq k$. The number k is called the *length* of the path. If $k = 0$ the path is called a *trivial* path. A path is *closed* if $x = y$. A path is *simple* when no vertex (except possibly x) occurs more than once in the sequence of vertices of the path. A non-trivial simple closed path in which all edges are distinct is called a *cycle*. A graph is *connected* if for all $\{x, y\} \subset V$, a path from x to y exists in G. A *tree* is a connected graph with no cycles. A simple path from x to y in a tree is unique and is denoted by $\pi(x, y)$. An unconnected graph with no cycles is called a *forest*, each of its connected components being a tree.

Directed Graphs. A *directed graph* is a pair (V, A), where V is a finite set, and A a subset of $V \times V$. An element of A is called an *arc*, an element of V is called a *vertex*. Let $G = (V, A)$ be a directed graph, and let x, y be in V, a *path* from x to y in G is a sequence of vertices $s_0, ..., s_k$ such that $x = v_0$, $y = v_k$ and $(v_{i-1}, v_i) \in A, 1 \leq i \leq k$. The *undirected graph associated* to G is the undirected graph $G' = (V, E)$, such that $\{x, y\} \in E$ if and only if $(x, y) \in A$ or $(y, x) \in A$. A vertex $r \in V$ is a root of G if for all $x \in V \setminus \{r\}$, a path from r to x in G exists. G is *antisymetric* if for all $(x, y) \in A$, $(y, x) \notin A$. The graph G is a *rooted tree*

(with root r) if r is a root of G, G is antisymetric and if the undirected graph associated to G is a tree. An unconnected graph, where each of its connected components is a tree, is called a *rooted forest*.

Let $G = (V, A)$ be a rooted tree. If $(y, x) \in A$, we say that y is the *parent* of x (denoted by $par(x)$), and that x is a *child* of y. The set of all children of y is denoted by $\mathcal{C}(y)$. The maximum length of a path between the root and another node is called the *height* of the tree. The vertices on the path from the root to a vertex x are called the *ancestors* of x. We denote the set of the ancestors of x by $anc(x)$.

Common Definitions. Unless otherwise indicated, all the other definitions and notations in this article are similar for the two kinds of graphs. We will give them for the directed graphs, the versions for undirected graphs can be obtained by replacing arcs by edges.

Two graphs $G = (V_G, A_G)$ and $G' = (V_{G'}, A_{G'})$ are said to be *isomorphic* if there exists a bijection $f : V_G \to V_{G'}$, such that for any pair $(x, y) \in V_G \times V_G$, $(x, y) \in A_G$ if and only if $(f(x), f(y)) \in A_{G'}$.

A *weighted graph* is a triplet (V, A, ω), where V is a finite set, A a subset of $V \times V$, and ω a mapping from A to $\mathbb{R}$. In a weighted tree, the weight of the unique path from x to y, denoted by $\omega(x, y)$ is the sum of the weights of all arcs traversed in the path.

Two weighted graphs (V, E, ω) and (V', E', ω') are isomorphic whenever the graphs (V, E) and (V', E') are isomorphic.

3 Measure of Similarity

The problem of comparing graphs occurs in diverse areas such as computational biology, image analysis and structured databases. However, the graphs considered in these domains are most often with *labeled vertices*. Each notion in this section will be introduced in the case of graphs with weighted edges/arcs.

We present here measure of similarity allowing the comparison of graphs. After adapting basic edit operations for weighted trees, we define a new alignment distance: the homeomorphic alignment distance, preserving topology and handling useless and splited vertices. We then see how we handle spurious branches.

3.1 Edit Operations

An approach widely used to compare two graphs is to search for a sequence of simple primitive operations (called *edit operations*) that transforms a graph into the other and that has a minimal cost, called the *edit distance*.

For a graph $G = (V, A, \omega)$, commonly used operations are :

resize : Change the weight of an arc $a = (u, v) \in A$.
delete : Delete an arc $a = (u, v) \in A$ and merge u and v into one vertex.
insert : Split a vertex in two vertices, and link them by a new arc.

The cost of these edit operations is given by a cost function $\gamma(w, w')$, where w (respectively w') is the total weight of the arcs involved in the operation before (respectively after) its application. As a consequence, the cost of a deletion can be denoted by $\gamma(w, 0)$, where w is the weight of the deleted arc, and the cost of an insertion by $\gamma(0, w)$, where w is the weight of the created arc. Furthermore, we asume that γ is a metric. Typically, $\gamma(w, w') = |w - w'|$ or $(w - w')^2$.

Various edit-based distances have been defined, using different constraints on sequence order and different definitions of operations. These edit-based distances can be classified, as proposed by Wang and al. [8] : Edit distance [9], alignment distance [10, 11], isolated-subtrees distance [12], and top-down distance [13]. Proposed edit distances, isolated-subtrees distances and top-down distances cannot always match all the model tree, but only subparts, most often unconnected. However, we will see in the next subsection that it is not the case for alignment distance.

3.2 Alignment Distance

In [10], Jiang and al. propose a similarity measure between vertex-labeled trees, that we transpose here for edge-weighted graphs.

Let $G_1 = (V_1, A_1, \omega_1)$ and $G_2 = (V_2, A_2, \omega_2)$ be two weighted graphs. Let $G'_1 = (V'_1, A'_1, \omega'_1)$ and $G'_2 = (V'_2, A'_2, \omega'_2)$ be weighted graphs obtained by inserting arcs weighted by 0 in G_1 and G_2, such that there exists an isomorphism $\mathcal{I}$ between G'_1 and G'_2. The set of all couples of arcs $\mathcal{A} = \{(a_1, a_2); a_1 \in A'_1, a_2 \in A'_2, a_2 = \mathcal{I}(a_1)\}$ is called an alignment of G_1 and G_2. The *cost* $C_{\mathcal{A}}$ of $\mathcal{A}$ is given by

$$C_{\mathcal{A}} = \sum_{(a_1,a_2)\in\mathcal{A}} \gamma(\omega'_1(a_1), \omega'_2(a_2)) \ . \tag{1}$$

The minimal cost of all alignments from G_1 and G_2, called the *alignment distance*, is denoted by $\alpha(G_1, G_2)$. Alignment distance is an interesting way in our case for three reasons:it preserves topological relations between trees, it can be computed in polynomial time, and it enables to "remove edges", regardless of the rest of the graph, solving the problem of splitted vertices.

3.3 Homeomorphic Alignment Distance

For the purpose of solving the useless vertex problem, we propose a new alignment, which removes 2-degree vertices.

Homeomorphism. A *subdivision* of an arc (u, v) in a weighted graph $G = (V, A, \omega)$ is an operation which consists in adding a new vertex w in V and two arcs (u, w) and (w, v) in A, removing (u, v) and assigning weights on the new arcs, such as $\omega((u, w)) + \omega((w, v)) = \omega((u, v))$. A *subdivision* of a weighted graph G is a graph obtained by a sequence of subdivisions of arcs of G.

The *merging* is the inverse operation of the subdivision, it applies only on arcs sharing a 2-degree vertex. The merging of two arcs (u, v) and (v, w) in a weighted graph $G = (V, A, \omega)$ consists in removing v in V, replacing (u, v) and (v, w) by (u, w) in A, weighted by $\omega((u, w)) = \omega((u, v)) + \omega((v, w))$.

Two weighted graphs $G = (V_G, A_G, \omega_G)$ and $G' = (V_{G'}, A_{G'}, \omega_{G'})$ are *homeomorphic* if there exists an isomorphism between some subdivision of G and some subdivision of G'.

Merging Kernel. Considering that a merging on a vertex v on the graph $G = (V, A, \omega)$ does not affect the degree of any vertex in $V \setminus \{v\}$ (by definition of merging operation) and therefore the possibility of merging this vertex, the number of possible mergings decreases by one after each merging. In consequence, the maximal size of a sequence of merging operations, transforming G into another graph $G' = (V', A', \omega')$ is equal to the initial number of possible mergings in G. It can be remarked that any sequence of merging operations of maximal size yields the same result. The graph resulting of such a sequence on G is called the *merging kernel* of G, and is denoted by $\mathcal{MK}(G)$. The following proposition is straightforward :

Proposition 1. *Two graphs $G_1 = (V_1, A_1, \omega_1)$ and $G_2 = (V_2, G_2, \omega_2)$ are homeomorphic iff $\mathcal{MK}(G_1)$ and $\mathcal{MK}(G_2)$ are isomorphic.*

Homeomorphic Alignment Distance. Let $G_1 = (V_1, A_1, \omega_1)$ and $G_2 = (V_2, A_2, \omega_2)$ be two weighted graphs. Let $G'_1 = (V'_1, A'_1, \omega'_1)$ and $G'_2 = (V'_2, A'_2, \omega'_2)$ be weighted graphs obtained by deleting arcs in G_1 and G_2, such that there exists an homeomorphism between G'_1 and G'_2(not necessarily unique). Let $G''_1 = (V''_1, A''_1, \omega''_1)$ and $G''_2 = (V''_2, A''_2, \omega''_2)$ be the merging kernels of G'_1 and G'_2, respectively. From proposition 1, there exists an isomorphism $\mathcal{I}$ between G''_1 and G''_2. The set of all couples of arcs $\mathcal{H} = \{(a, a'); a \in A''_1, a' \in A''_2, a' = \mathcal{I}(a)\}$ is called an *homeomorphic alignment* of G_1 with G_2.

The *cost* $C_{\mathcal{H}}$ of $\mathcal{H}$ is defined as

$$C_{\mathcal{H}} = \sum_{(a,a') \in \mathcal{H}} \gamma(\omega''_1(a), \omega''_2(a')) + \sum_{a_d \in A_1 \setminus A'_1} \gamma(\omega_1(a_d), 0) + \sum_{a'_d \in A_2 \setminus A'_2} \gamma(0, \omega_2(a'_d)) \ . \tag{2}$$

This minimal cost of all homeomorphic alignments between G_1 and G_2, called the *homeomorphic alignment distance*, is denoted by $\eta(G_1, G_2)$.

3.4 Cut Operation

The last remaining problem is the presence of spurious branches, which have to be removed without any cost. For this purpose, we propose to integrate the cut operation in our alignment.

In [14], Wang et al. propose a new operation allowing to consider only a part of a tree. Let $G = (V, A, \omega)$ be a weighted tree. *Cutting* G at an arc $a \in A$, means

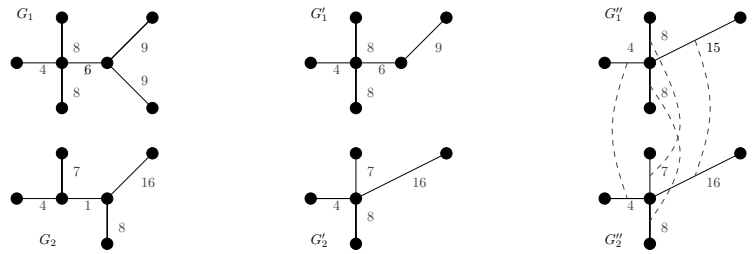

Fig. 1. Example of homeomorphic alignment:G'_1 (resp. G'_2) is obtained from G_1 (resp. G_2) by deletions of edges. $G''_1 = \mathcal{MK}(G'_1)$ and $G''_2 = \mathcal{MK}(G'_2)$. The dotted lines represent a possible homeomorphic alignment of G_1 and G_2, with cost equal to 12, assuming that $\gamma(x, y) = |x - y|$.

removing a, thus dividing G into two subtrees G_1 and G_2. The *cut operation* consists of cutting G at an arc $a \in A$, then considering only one of the two subtrees. Let K a subset of A. We use $Cut(G, K, v)$ to denote the subtree of G containing v and resulting from cutting G at all arcs in K. In the case of a rooted tree, we consider that the root $r_G$of$G$ cannot be removed by the cut operation, and then we can use the notation $Cut(G, K) = Cut(G, K, r_G)$. In the case of a rooted forest, we consider that the root of each rooted tree composing the rooted forest cannot be removed by the cut operation, and then we can use the same notation than above:$Cut(G, K)$.

Our main problem can be stated as follows:Given a weighted tree $P = (V_P, A_P, \omega_P)$ (the pattern tree) and a weighted tree $G_D = (V_D, A_D, \omega_D)$ (the data tree), find $\eta_{cut}(P, D) = min_{K \subseteq A_D, v \in V_D}\{\eta(P, Cut(D, K, v)\}$ (in the case of rooted trees and rooted forests, $\eta_{cut}(P, D) = min_{K \subseteq A_D}\{\eta(P, Cut(D, K))\}$), and the associated homeomorphic alignment.

4 Algorithms

4.1 Algorithm for Rooted Trees

Let $T = (V, A, \omega)$ be a weighted tree rooted in r_T. For each vertex $v \in V \setminus \{r_T\}$, we denote by $\uparrow v$ the arc $(w, v) \in A$, w being the parent of v. We denote by $T(v), v \in V$, the subtree of T rooted in v. We denote by $\Pi(a, b)$ the set of all vertices of the path $\pi(a, b)$. Let v_a be an ancestor of v, we denote by $T_{cut}(v, v_a)$ the subgraph of T defined as follows :

$$T_{cut}(v, v_a) = Cut(T(v_a), \{\uparrow p', p' \in \mathcal{C}(p) \setminus \Pi(v_a, v), p \in \Pi(v_a, par(v))\}) \ . \quad (3)$$

We denote by $T(v, v_a)$ the tree obtained from $T_{cut}(v, v_a)$ by merging on each vertex $n \in \Pi(v_a, v) \setminus \{v_a, v\}$. We denote by $\mathcal{F}(T, v)$ the rooted forest, the connected components of which are the trees $T(p, v)$, for all $p \in \mathcal{C}(v)$. By abuse

of notation we also denote by $\mathcal{F}(T, v)$ the set of all connected components of this forest (that is, as set of trees).

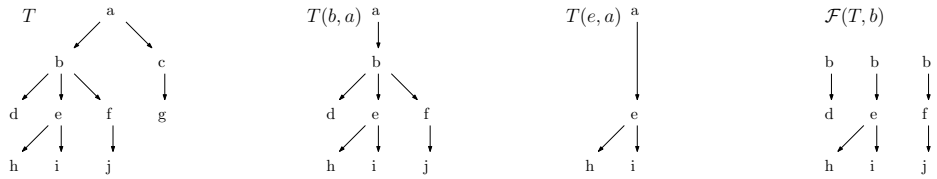

Fig. 2. Examples for a rooted tree T.

Proofs of the following propositions can be found in [15].

Proposition 2. *Let $P = (V_P, E_P, \omega_P)$ and $D = (V_D, E_D, \omega_D)$ be two weighted trees, rooted respectively in r_P and r_D.*

$$\eta_{cut}(P, D) = \eta_{cut}(\mathcal{F}(P, r_P), \mathcal{F}(D, r_D)) \ . \tag{4}$$

Proposition 3. *Let $i \in V_P \setminus \{p\}, j \in V_D \setminus \{d\}, i_a \in anc(i), j_a \in anc(j)$,*

$$\begin{aligned}
&\eta_{cut}(\emptyset, \emptyset) = 0 \\
&\eta_{cut}(P(i, i_a), \emptyset) = \eta_{cut}(\mathcal{F}(P, i), \emptyset) + \gamma(\omega(i_a, i), 0) \\
&\eta_{cut}(\mathcal{F}(P, i_a), \emptyset) = \sum_{i' \in \mathcal{C}(i_a)} \eta_{cut}(P(i', i_a), \emptyset) \\
&\eta_{cut}(\emptyset, D(j, j_a)) = 0 \\
&\eta_{cut}(\emptyset, \mathcal{F}(D, j_a)) = 0 \ .
\end{aligned} \tag{5}$$

Proposition 4. *Let $i \in V_P \setminus \{p\}, j \in V_D \setminus \{d\}, i_a \in anc(i), j_a \in anc(j)$.*

$$\eta_{cut}(P(i, i_a), D(j, j_a)) = \\ min \begin{cases} \eta_{cut}(\mathcal{F}(P, i), \emptyset) + \gamma(\omega(i_a, i), 0) \\ \gamma(\omega(i_a, i), \omega(j_a, j)) + \eta_{cut}(\mathcal{F}(P, i), \mathcal{F}(D, j)) \\ min_{j_c \in \mathcal{C}(j)}\{\eta_{cut}(P(i, i_a), D(j_c, j_a))\} \\ min_{i_c \in \mathcal{C}(i)}\{\eta_{cut}(P(i_c, i_a), D(j, j_a)) + \displaystyle\sum_{i'_c \in \mathcal{C}(i) \setminus i_c} \eta_{cut}(P(i'_c, i), \emptyset)\} \ . \end{cases} \tag{6}$$

Proposition 5. $\forall A \subseteq \mathcal{F}(P, i), B \subseteq \mathcal{F}(D, j)$,

$$\eta_{cut}(A, B) = \\ min \begin{cases} min_{D(j', j) \in B} & \{\eta_{cut}(A, B \setminus \{D(j', j)\})\} \\ min_{P(i', i) \in A} & \{\eta_{cut}(A \setminus \{P(i', i)\}, B) + \eta_{cut}(P(i', i), \emptyset)\} \\ min_{P(i', i) \in A, D(j', j) \in B} & \{\eta_{cut}(A \setminus \{P(i', i)\}, B \setminus \{D(j', j)\}) + \\ & \eta_{cut}(P(i', i), D(j', j))\} \\ min_{P(i', i) \in A, B' \subseteq B} & \{\eta_{cut}(A \setminus \{P(i', i)\}, B \setminus B') \\ & + \eta_{cut}(\mathcal{F}(P, i'), B') + \gamma(\Omega(i'), 0)\} \\ min_{A' \subseteq A, D(j', j) \in B} & \{\eta_{cut}(A \setminus A', B \setminus \{D(j', j)\}) + \\ & \eta_{cut}(A', \mathcal{F}(D, j')j) + \gamma(0, \Omega(j'))\} \ . \end{cases} \tag{7}$$

The algorithm will use a bottom-up approach: for each $(i,j) \in V_P \times V_D$ in suffix order, compute succesively $\eta_{cut}(P(i,i_a),\emptyset)$ and $\eta_{cut}(\mathcal{F}(P,i),\emptyset)$ (using Prop. 3), $\eta_{cut}(A,B)$, $\forall A \subseteq \mathcal{F}(P,i), B \subseteq \mathcal{F}(D,j)$ (using Prop. 5), and $\eta_{cut}(P(i,i_a),D(j,j_a))$, $\forall i_a \in anc(i), j_a \in anc(j)$ (using Prop.4). Then, return $\eta_{cut}(P,D)$, using Prop.2.

The total computation time complexity is in $O(|V_P|*|V_D|*(2^{d_P}*2^{d_D}*(d_D*2^{d_P}+d_P*2^{d_D})+h_P*h_D*({d_P}^2+d_D))$, where d_G and h_G denote, respectively, the maximal degree of a vertex in G and the height of G. If the maximal degree is bounded, the total computation time complexity is in $O(|V_P|*|V_D|*h_P*h_D)$.

4.2 Algorithm for Unrooted Trees

Let $G=(V,E,\omega)$ be a weighted tree, let $r \in V$, we denote by G^r the directed weighted tree rooted in r, such that G is the undirected graph associated to G^r.

Proposition 6. *Let $P=(V_P,E_P,\omega_P)$ and $D=(V_D,E_D,\omega_D)$ be two weighted trees.*

$$\eta_{cut}(P,D) = min_{i\in V_P, j\in V_D}\{\eta_{cut}(P^i,D^j)\} \ . \tag{8}$$

Let $P=(V_P,E_P,\omega_P)$ and $D=(V_D,E_D,\omega_D)$ be two undirected weighted trees. We denote by $\mathcal{F}(P,a,b), a,b \in V_P$ the set of rooted trees P^r, $r \in V_P$, such that b is an ancestor of a in P^r. We denote by $anc(P,a,b)$, $a,b \in V_P$ the set of the ancestors of a in at least one rooted tree in $\mathcal{F}(P,a,b)$. We denote by $\mathcal{C}(P,a,b)$, $a,b \in V_P$ the set of the children of a in at least one rooted tree in $\mathcal{F}(P,a,b)$.

Our algorithm is based on an approach similar to bottom-up. It is easy to see that for computing $\eta_{cut}(P^p(i,i_a),D^d(j,j_a))$ and $\eta_{cut}(\mathcal{F}(P^p,i),\mathcal{F}(D^d,j))$, for any $p \in V_P, d \in V_d$ we need to know $\eta_{cut}(P^p(i_c,i),D^d(j_c,j))$ for all $i_c \in \mathcal{C}(P,i,p)$, $j_c \in \mathcal{C}(D,j,d)$. We start by computing $\eta_{cut}(P^p(i,i_a),D^d(j,j_a))$ and $\eta_{cut}(\mathcal{F}(P^p,i),\mathcal{F}(D^d,j))$, for all i (respectively j) being a leaf of P^p (respectively D^d) and continue iteratively with vertices which have all their children already computed.

The total computation time of this algorithm is in $O(|V_P|*|V_D|*(d_P*2^{d_P+2*d_D}+d_D*2^{d_D+2*d_P}+|V_P|*|V_D|*({d_P}^2+d_D)))$ complexity. If the maximal degree is bounded, the total computation is in $O(|V_P|^2*|V_D|^2)$ time complexity.

5 Experimentation

5.1 Usage of Homeomorphic Alignment

In case of motion capture, we can use homeomorphic alignment in three different ways :

- between the two unrooted trees, if we have no a priori knowledge.
- between two rooted trees, obtained from the unrooted trees by rooting them on vertices we want to match together.
- between a rooted tree and an unrooted tree, if we want to be sure that the root is conserved by the homeomorphic alignment.

5.2 Results

Our model tree contains seven vertices, representing head, torso, crotch, the two hands and the two feet. Experimentally, the data tree obtained from the skeleton of the visual hull has a degree bounded by 4, and its number of vertices is between seven and twenty, with a gaussian probability repartition centred on ten. All the results have been obtained on a computer with a processor Xeon 3 GHz and 1 Go of RAM.

For finding the average time of computation of our algorithm, we have randomly generated 32 pattern trees, and for each pattern tree, we have generated 32 data trees, which yields 1024 pairs of trees. Each pattern tree has seven vertices, one of which has a degree equals to 4. Each data tree has at least one 4-degree vertex. The results of the four kinds of alignments are shown on Fig. 3.

In the average case ($|V_D| \leq 12$), the homeomorphic alignement between a rooted pattern tree and a unrooted data tree (we assume than the torso is always aligned), can be easily computed in real time (frequence superior to 24Hz) and in the worst case ($|V_D| \simeq 20$), we keep an interactive time (frequence superior to 12Hz). For tracking, if we can use the homeomorphic alignment between two rooted trees, we are widely above 50Hz.

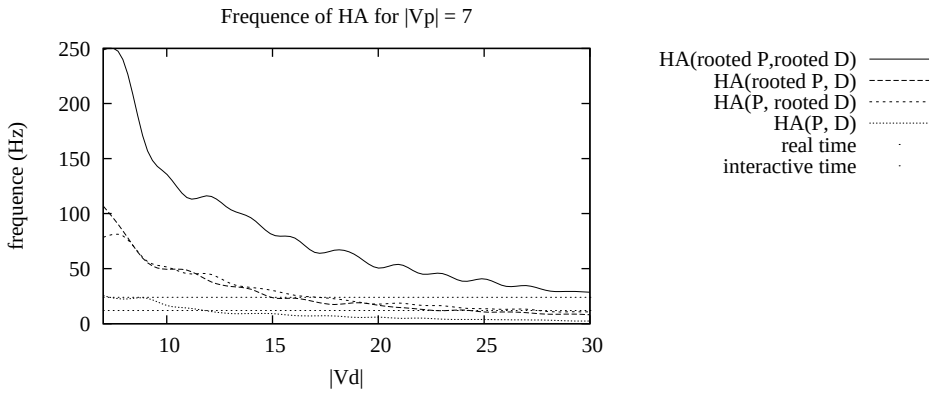

Fig. 3. Frequences of the differents homeomorphic alignments for variable sizes of data tree.

6 Conclusion

In this paper, we have introduced a new type of alignment between weighted trees, the homeomorphic aligment, taking into account the topology and avoiding the noise incluced by spurious branches, splited and useless 2-degree vertices. We have also developed several robust algorithms to compute it with a good complexity, which enable its application in real time for motion capture purpose.

In future works, we will take into account more useful information on the model, such as spatial coordinates of data vertices, and include them in our algorithm, for a better robustness. Finally, using this alignment, we will propose a new fast method of pose initialization for motion capture applications.

References

1. Moeslund, T.B., Hilton, A., Krüger, V.: A Survey of Advances in Vision-based Human Motion Capture and Analysis. In: Computer Vision and Image Understanding vol. 104(2-3), pp. 90–126. Elsevier (2006)
2. Chu, C., Jenkins, O., Mataric, M.: Markerless Kinematic Model and Motion Capture from Volume Sequences. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition. vol 2., IEEE Computer Society (2003)
3. Menier, C., Boyer, E., Raffin, B.: 3d Skeleton-based Body Pose Recovery. In: Proceedings of the 3rd International Symposium on 3D Data Processing, Visualization and Transmission, Chapel Hill (USA). (2006)
4. Brostow, G., Essa, I., Steedly, D., Kwatra, V.: Novel Skeletal Representation for Articulated Creatures. LNCS, pp. 66?78. Springer, Heidelberg (2006)
5. Sundar, H., Silver, D., Gagvani, N., Dickinson, S.: Skeleton Based Shape Matching and Retrieval. In SMI, pages 130–139, (2003)
6. Baran, I., Popović, J.: Automatic rigging and animation of 3D characters. In: International Conference on Computer Graphics and Interactive Techniques, ACM Press New York, NY, USA (2007)
7. Cornea, N., Demirci, M., Silver, D., Shokoufandeh, A., Dickinson, S., Kantor, P.: 3D Object Retrieval using Many-to-many Matching of Curve Skeletons. In: Shape Modeling and Applications. (2005)
8. Wang, J., Zhang, K.: Finding similar consensus between trees: an algorithm and a distance hierarchy. Pattern Recognition 34(1) pp. 127–137. Elsevier (2001)
9. Tai, K.: The Tree-to-Tree Correction Problem. Journal of the ACM 26(3) pp. 422?433. ACM New York, NY, USA (1979)
10. Jiang, T., Wang, L., Zhang, K.: Alignment of Trees - an Alternative to Tree Edit. In: CPM 94: Proceedings of the 5th Annual Symposium on Combinatorial Pattern Matching, London, UK, pp. 75–86. Springer-Verlag (1994)
11. Jansson, J., Lingas, A.: A Fast Algorithm for Optimal Alignment between Similar Ordered Trees. LNCS vol. 2089 pp. 232–??. Springer, Heidelberg (2001)
12. Tanaka, E., Tanaka, K.: The Tree-to-tree Editing Problem. International Journal of Pattern Recognition and Artificial Intelligence. 2(2) pp. 221–240 (1988)
13. Selkow, S.: The Tree-to-Tree Editing Problem. Information Processing Letters 6(6) pp. 184–186 (1977)
14. Wang, J., Zhang, K., Chang, G., Shasha, D.: Finding Approximate Patterns in Undirected Acyclic Graphs. In : Pattern Recognition vol.35(2) pp. 473?483. Elsevier (2002)
15. Raynal, B., Biri, V., Couprie, M.: Homeomorphic Alignment of Weighted Trees. Internal report IGM 2009-01. LIGM, Université Paris-Est (2009)