

# 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. For this purpose, we match the tree representation of the skeleton of the 3D shape to a pre-specified tree model. Unfortunately, the tree representation can contain vertices that split limbs in multiple parts, which do not allow a good match by usual methods. To solve this problem, we propose a new alignment, taking in account the homeomorphism between trees, rather than the isomorphism, as in prior works. Then, we develop several computationally efficient algorithms for reaching real-time motion capture.

**Key words:** Graphs, homeomorphism, alignment, matching algorithm

## 1 Introduction

Motion capture without markers is a highly active research area, and is used in several applications which have not the same needs: 3D models animation, for movies FX or video games for example, requests an highly accurate model, but does not need real-time computation (offline video processing is acceptable). Real-time interaction, for virtual reality applications, requests a fast computation, at the price of a lower accuracy. This paper is placed in the context of real-time interaction.

The first step (called *initialization step*) consists of finding the initial pose of the subject, represented here by a 3d shape (visual hull) constructed using a multi view system with an algorithm of Shape From Silhouette [1].

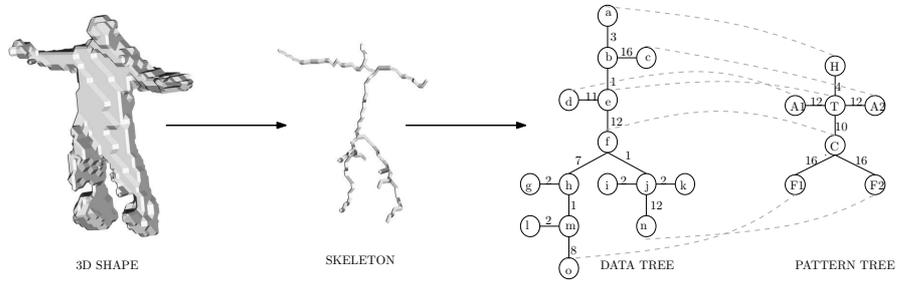
An important part of the algorithms of 3D pose estimation use a manually initialized model, or ask the subject to move succesively the differents parts of his/her body, but several automatic approaches have been developped, using an a priori model. This a priori model can approximate different characteristics of the subject: kinematic structure, shape or appearance. This kind of a priori model have several constraints. It is complex because characteristics of different nature are involved, and needs to be adapted to each subject (specialy in the case of appearance).

### 1.1 Motivation

Our goal is to automatize the initial pose estimation step. To achieve this aim, we use a very 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 each edge represents the link between this parts, associated to a weight, representing the distance between two parts.

Concerning the data, we extract the curve skeleton of the visual hull, 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 the edge beeing the geodesic distance between them (see figure 1.1).



**Fig. 1.** Example of data tree acquisition and expected alignment with model tree

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.

A lot of similar approaches have been developed, using the skeleton of a shape, in motion capture research area [3–5], and in 3D shape matching research area [6, 8]. In the first case, the best time obtained for find the initial pose is some one second [4], which is too slow, even for interactive time interaction. In the second case, the algorithms used give an approximated solution [8], or need a accurate knowledge of the radius distance of the skeleton, in order to compute the associated shock graph [9].

As shown on Fig.1.1, several kinds of noise and deformities can appear in the data tree : spurious branches (edges  $\{g, h\}$ ,  $\{l, m\}$ ,  $\{i, j\}$ ,  $\{j, k\}$ ), useless 2-degree vertices, obtained after spurious branches deletion (in our example, vertices  $j, k, m$ ), and splitted vertices (vertex  $T$  of pattern tree match with vertices  $b$  and  $e$  in data tree).

Approaches found in the literature do not permit to achieve a robust matching, with respect to these perturbations, mainly because they are defined for reach an isomorphism between the trees, instead of an homeomorphism. In the following, after adapting basic notions, we introduce both a new alignment, called

homeomorphic alignment, and a robust tree-matching algorithm which may be used for real-time pose estimation.

## 2 Basics Notions

An *undirected graph* is a pair  $(V, E)$ , where  $V$  is a finite set of *vertices*, and  $E$  a subset of  $\{\{x, y\}, x \in V, y \in V, x \neq y\}$  (called *edge*). The *degree* of  $v \in V$  is denoted by  $\deg(v)$ . 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  $\pi(x, y)$ . A *forest* is a graph with no cycles, each of its connected components being a tree.

A *directed graph* is a pair  $(V, A)$ , where  $V$  is a finite set, and  $A$  a subset of  $V \times V$  (called *arc set*). 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 *antisymmetric* 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 antisymmetric and if the undirected graph associated to  $G$  is a tree. A graph, where each of its connected components is a rooted tree, is called a *rooted forest*. The *parent* of  $x \in V$  is denoted by  $\text{par}(x)$ , the set of the *ancestors* of  $x$  by  $\text{anc}(x)$  and the set of all *children* of  $x$  is denoted by  $\mathcal{C}(x)$ .

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  $f : V_G \rightarrow V_{G'}$ , 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  $(V, A, \omega)$ , where  $V$  is a finite set,  $A$  a subset of  $V \times V$ , and  $\omega$  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.

## 3 Measure of Similarity

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. We assume that  $\gamma$  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. [10] : Edit distance [11], alignment

distance [12, 13], isolated-subtrees distance [14], and top-down distance [15]. 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.

### 3.1 Alignment Distance

In [12], Jiang et 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

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

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.

### 3.2 Cut Operation

In the purpose of removing spurious branches without any cost, we propose to integrate the cut operation in our alignment.

In [16], 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 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.

At this step, we can combine the methods described above [12, 16] as follows : given  $P = (V_P, A_P, \omega_P)$  (the pattern tree) and  $D = (V_D, A_D, \omega_D)$  (the data tree), we define  $\alpha_{cut}(P, D) = \min_{K \subseteq A_D, v \in V_D} \{\alpha(P, Cut(D, K, v))\}$ , which is the minimal alignment distance between  $P$  and a subgraph of  $D$ . The introduction of  $\alpha_{cut}(P, D)$  solves the problems of splitted vertices and spurious branches, but not the problem of useless 2-degree vertices. In the example of Fig.1.1, the vertex  $F1$  in pattern tree will match with the vertex  $h$  in the data tree, instead of the vertex  $o$ , because after cut of  $\{g, h\}$  and  $\{l, m\}$ , edges  $\{f, h\}$ ,  $\{h, m\}$  and  $\{m, o\}$  cannot be merged in only one edge, and then cannot be matched with  $\{C, F1\}$ .

### 3.3 Homeomorphic Alignment Distance

For the purpose of solving the useless vertex problem, we propose a new alignment, which removes 2-degree vertices and search for minimal sequence of operations to reach a homeomorphism, instead of an isomorphism between the trees.

**Homeomorphism.** The *merging* is an operation that can be applied 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 of 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 a graph obtained by mergings on  $G$  and a graph obtained by mergings on  $G'$ .

**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  $G_2 = (V_2, A_2, \omega_2)$  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

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

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

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  $D = (V_D, A_D, \omega_D)$  (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,  $\eta_{cut}(P, D) = \min_{K \subseteq A_D} \{\eta(P, Cut(D, K, r_D))\}$ ), and the associated homeomorphic alignment.

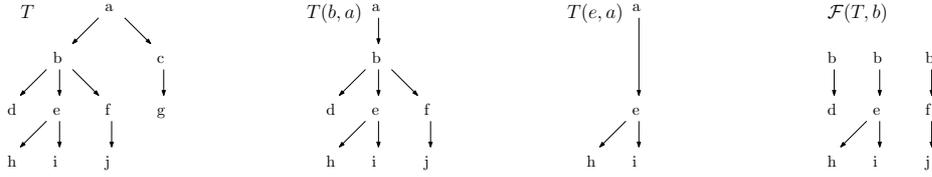
## 4 Algorithms

### 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 [17].

**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  $r_D$ .

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

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

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

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

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

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

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

---

**Algorithm 1:** Homeomorphic Alignment Distance for Rooted Trees

---

**Data:** pattern rooted tree  $P$ , datas rooted tree  $D$

**Result:**  $\eta_{\text{cut}}(P, D) = \eta_{\text{cut}}(\mathcal{F}(P, r_P), \mathcal{F}(D, r_D))$ ; // Prop.2

**begin**

**foreach**  $p \in V_P$ , in suffix order **do**

**foreach**  $A \subseteq \mathcal{F}(P, p)$  **do** Compute  $\eta_{\text{cut}}(A, \emptyset)$ ; // Prop.3

**foreach**  $p_a \in \text{anc}(p) \setminus \{p\}$  **do** Compute  $\eta_{\text{cut}}(P(p, p_a), \emptyset)$

**foreach**  $d \in V_D$ , in suffix order **do**

**foreach**  $B \subseteq \mathcal{F}(D, d)$  **do** Compute  $\eta_{\text{cut}}(\emptyset, B)$ ; // Prop.3

**foreach**  $d_a \in \text{anc}(d) \setminus \{d\}$  **do** Compute  $\eta_{\text{cut}}(\emptyset, D(d, d_a))$

**foreach**  $p \in V_P, d \in V_D$ , both in suffix order **do**

**foreach**  $A \subseteq \mathcal{F}(P, p), B \subseteq \mathcal{F}(D, d)$  **do**

      Compute  $\eta_{\text{cut}}(A, B)$ ; // Prop.5

**foreach**  $p_a \in \text{anc}(p) \setminus \{p\}, d_a \in \text{anc}(d) \setminus \{d\}$  **do**

      Compute  $\eta_{\text{cut}}(P(p, p_a), D(d, d_a))$ ; // Prop.4

**end**

---

The total computation time complexity is in  $O(|V_P| * |V_D| * (2^{d_P} * 2^{d_D} * (d_D * 2^{d_P} + d_P * 2^{d_D}) + h_P * h_D * (d_P + d_D)))$ , 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.*

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

By choosing an adapted order of navigation in the trees, avoiding the redundancy of subtrees alignment computation, we can use the same algorithm than for rooted trees. The total computation time of this algorithm is in  $O(|V_P| * |V_D| * (d_P * 2^{d_P+2*d_D} + d_D * 2^{d_D+2*d_P} + |V_P| * |V_D| * (d_P + d_D)))$  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

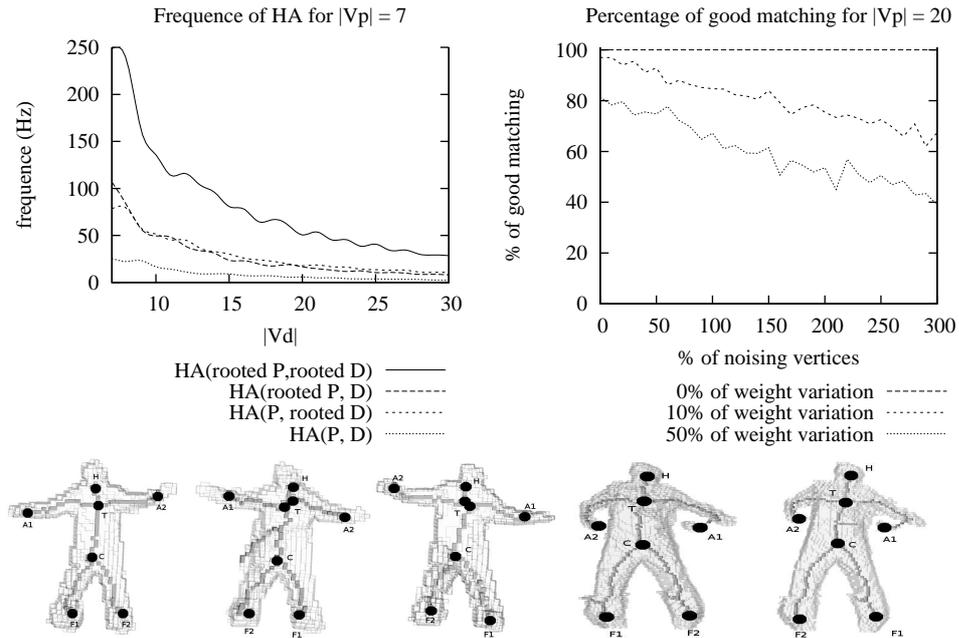
Our model tree contains seven vertices, representing head, torso, crotch, the two hands and the two feet. Experimentally, the data tree obtained from the skeleton of the visual hull has a degree bounded by 4, and its number of vertices is between seven and twenty, with a gaussian probability repartition 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 alignment between a rooted pattern tree and an unrooted data tree (we assume than the torso is always aligned), can be easily computed in real time (frequency superior to 24Hz) and in the worst case ( $|V_D| \simeq 20$ ), we keep an interactive time (frequency superior

to 12Hz). For tracking, if we can use the homeomorphic alignment between two rooted trees, we are widely above 50Hz.

For finding the average precision of our algorithm, we have generate data trees from pattern trees by adding new vertices, by three ways : splitting an existing vertex in twice, adding a new 1-degree vertex, adding a new 2-degree vertex. Then, we modify the weight of each edge in a proportional range. The results are shown on Fig. 3.



**Fig. 3.** Top : frequencies of the different homeomorphic alignments for variable sizes of data tree, and precision for several kind of noises. Bottom : Examples of results obtained on 3D shape : black circles represent the points matching with pattern tree.

## 6 Conclusion

In this paper, we have introduced a new type of alignment between weighted trees, the homeomorphic alignment, taking into account the topology and avoiding the noise induced by spurious branches, splitted and useless 2-degree vertices. This alignment has the particularity to propose graph transformation to reach an homeomorphism between tree, instead of an isomorphism, as usually proposed in the literature.

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. Laurentini, A.: The Visual Hull Concept for Silhouette-based Image Understanding. In: IEEE transactions on Pattern Analysis and Machine Intelligence. vol 16(2) pp.150–162. IEEE Computer Society (1994)
2. 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)
3. 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)
4. 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)
5. Brostow, G., Essa, I., Steedly, D., Kwatra, V.: Novel Skeletal Representation for Articulated Creatures. LNCS, pp. 66–78. Springer, Heidelberg (2006)
6. Sundar, H., Silver, D., Gagvani, N., Dickinson, S.: Skeleton Based Shape Matching and Retrieval. In SMI, pages 130–139, (2003)
7. 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)
8. 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)
9. Siddiqi, K., Shokoufandeh, A., Dickinson, S., Zucker, S.: Shock Graphs and Shape Matching. International Journal of Computer Vision 35(1):13–32 (1999)
10. Wang, J., Zhang, K.: Finding similar consensus between trees: an algorithm and a distance hierarchy. Pattern Recognition 34(1) pp. 127–137. Elsevier (2001)
11. Tai, K.: The Tree-to-Tree Correction Problem. Journal of the ACM 26(3) pp. 422–433. ACM New York, NY, USA (1979)
12. 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)
13. Jansson, J., Lingas, A.: A Fast Algorithm for Optimal Alignment between Similar Ordered Trees. LNCS vol. 2089 pp. 232–240. Springer, Heidelberg (2001)
14. Tanaka, E., Tanaka, K.: The Tree-to-tree Editing Problem. International Journal of Pattern Recognition and Artificial Intelligence. 2(2) pp. 221–240 (1988)
15. Selkow, S.: The Tree-to-Tree Editing Problem. Information Processing Letters 6(6) pp. 184–186 (1977)
16. 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)
17. Raynal, B., Biri, V., Couprie, M.: Homeomorphic Alignment of Weighted Trees. Internal report IGM 2009-01. LIGM, Université Paris-Est (2009)