Generic Initialization for Motion Capture from 3D Shape

Benjamin Raynal, Michel Couprie, and Vincent Nozick

Université Paris-Est
Laboratoire d'Informatique Gaspard Monge, Equipe A3SI

Abstract. Real time and markerless motion capture is an active research area, due to applications in human-computer interactions, for example. A large part of the existing markerless motion capture methods require an initialization step, consisting in finding the initial position of the different limbs of the subject. In this paper, we propose a new method for interactive time initialization step, only based on morphological and topological information and which can be easily adapted to any kind of model (full human body or only hand, animals, for example).

Key words: motion capture, initialization, markerless, skeleton, tree matching

1 Introduction

Motion capture without marker is a highly active research area, as shown by Moeslund and al. [1]: between 2000 and 2006, more than 350 papers on this topic have been published. Markerless motion capture approaches can be classified in two categories: those which detect the pose of the subject independently at each frame, and those which start by an initialization step in order to find the initial pose of the subject, then use tracking to find the pose in the following frames. In most of these methods, the initialization step uses an a priori model, which can be of several kinds, describing different information: kinematic skeleton, shape, color priors. Most of real-time methods for full body pose estimation purpose [2–4] use both color priors and shape fitting for the initialization. The method proposed in [5], close to interactive time (about one iteration per second), uses kinematic skeleton fitting.

In a context of generic motion capture, where the subject can be a full body, the hand, or the the upper part of the body for example, some of this information cannot be retained, as the color information (hands have homogeneous color). Furthermore, the shape of the subject can differ from person to person. In addition, from our point of view, the a priori model in generic motion capture must be as simple as possible.

Our goal is to propose a motion capture initialization method which has the following properties:

interactive runtime: our method must be fast enough to be usable in online context (more than one iteration per second).

markerless: our method does not require any kind of marker.
generic: our method has to be compatible with any subject.

Our method is based on 3D shape obtained by visual hull reconstruction. In our method, we use a voxel representation of the visual hull, in opposition to the polygonal representation. This choice is guided by the fact that we use some discrete treatments which are easy and fast to perform in a voxel grid. As we use a small number of cameras, the 3D shape can contain some deformities, as variations of limb thickness, noisy surface, or ghost limbs (i.e. parts which not exist in the subject). Thus only the global topology of the shape and the length of the different limbs are well preserved.

The method starts by extracting this information by skeletonizing the shape (Sec.2). Then a data tree representation of the skeleton is extracted (Sec.3), containing all the information we need: positions of ending and intersection points on vertices, and distances between them on edges.

Then, according to the edge information, we proceed to a matching between an a priori model and the data tree (Sec.4). Finally, we discriminate similar limbs if necessary, using "between" constraints of the model (Sec.5). Figure 1 show the complete pipeline of our method.

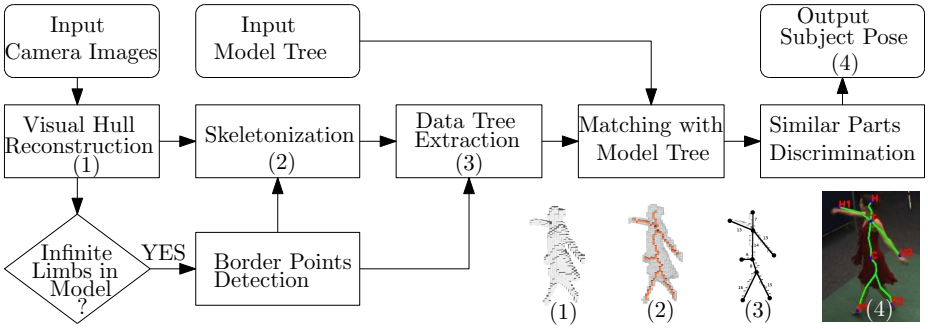

Fig. 1. Pipeline of our method.

1.1 A Priori Model Definition

Our a priori model is very simple. It consists of a tree, where vertices represent the different parts of the subject, and the edges contain information on distance between the different parts. Two kinds of models can be considered (see Fig.2 for examples):

– the *incomplete models*, which are part of a biggest shape, as in the case of hand pose estimation. In this case, a part of the shape intersects the border of the 3D acquisition space, and we represent this part in the model as an *infinite limb*.

– the *complete models*, for subjects fully contained in the 3D acquisition space, as in the case of full human body motion capture, for example.

In addition of the tree representation, two kinds of constraints can be added:

"Between" constraints specify the position of a part between two others, in order to discriminate similar limbs. In the case of hand model, we require e.g. that the index is between the thumb and the middle finger. More details about "between" constraints are given in Sec.5.

"Coordinate" constraints require a particular spatial position of a part in regard of the spatial position of one of its neighbor. In the case of full body model for example, we require that the head is above the torso. Constraints of this kind improve the matching robustness.

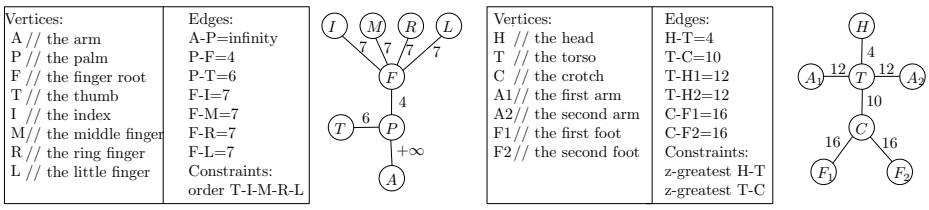

Fig. 2. From left to right: description of the hand model, model tree for the hand, description of the full body model, and model tree for the full body.

2 Skeletonization

Topology-preserving operators, like homotopic skeletonization, are used to transform an object while leaving unchanged its topological characteristics. In discrete grids ($\mathbb{Z}^n$, with $n = 2, 3$), such a transformation can be defined thanks to the notion of simple point [6–8]: intuitively, a point of an object (*i.e.*, a subset of $\mathbb{Z}^n$) is called simple if it can be deleted from this object without changing its topological characteristics.

The most "natural" way to thin an object consists of removing some of its border points in parallel, in a symmetrical manner. By repeating such a procedure until stability, one can obtain a well-centered "skeleton" of the original object. However, parallel deletion of simple points does not, in general, guarantee topology preservation. In fact, such a guarantee is not obvious to obtain, even for the 2D case. To check whether a point is simple or not, it is sufficient to examine its $3 \times 3 \times 3$ neighborhood (3×3 in 2D), but such a local criterion does not allow to check whether a simple point may be safely removed together with other ones.

In [9], G. Bertrand introduces a general framework for the study of parallel thinning in any dimension. The most fundamental result proved in [9] is that, if a subset Y of an object X contains the so-called critical kernel of X, then Y has the same topological characteristics as X. In [10], several new parallel algorithms to compute curvilinear skeletons are proposed, in which topological and geometrical conditions are clearly separated, unlike in many previous works. The topological soundness of these algorithms is proved thanks to the aforementioned property of critical kernels. Furthermore, these algorithms may be expressed by the way of masks and are relatively simple to implement.

The skeletonization algorithm that we use in this study is named ACK^3 in [10]. We choose this algorithm for its computational speed, the possibility of parallelization, and for the quality of the resulting skeleton (very low amount of noise, and guaranty that branch thickness is always of one voxel (asymmetrical skeleton). Since the complete presentation of this algorithm is beyond the scope of this paper, we give here a sketch of its main lines.

Let us describe one step of algorithm ACK^3. Let X be the current object. The set S of all simple points of X is computed, as well as the set I of all $1D$ isthmuses of X (points of which the removal would break locally X into several components). Then a subset Y of X is computed, that verifies the following conditions: i) Y is a superset of $X \setminus S$, ii) Y contains the critical kernel of X, and iii) Y contains I. If $Y = X$ then the algorithm stops, otherwise X is set to Y and the algorithm continues.

In the case of incomplete models, we have to take into consideration the part of the shape which is on the border of grid. Preliminary to the iterative process, the set B of the border points contained in X is computed. For each connected component of B, we compute the centroid, which will be preserved during the skeletonization process. Figure 3 show iterations results, for both kinds of model.

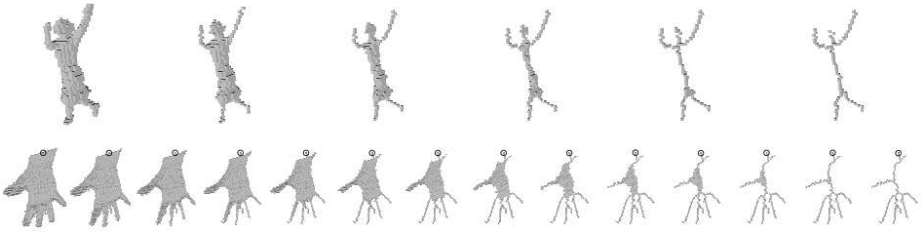

Fig. 3. From left to right, results of successive iterations of skeletonization. Top: human model. Bottom: hand model. The circled points represent the border points used to constraint the skeletonization.

3 Extraction of Data Tree Representation

We extract the data tree representation from the skeleton obtained in the previous step. The skeleton points can be classified into three classes, in regard of their number of neighbors included in the skeleton: ending points (exactly one neighbor included in the skeleton), linking points (exactly two neighbors), and intersection points (strictly more than two neighbors).

The points of interest are the ending points and the intersection points. For each of them, we create a vertex in the data tree. If several intersection points are neighbors, we merge their associated vertices. Intersection points and ending points are connected by sequences of linking points. For each sequence, we create an edge between vertices associated to its extremities, weighted by the length of the sequence, incremented by one if an extremity is an ending point. See Fig.4 for an example with a 2D skeleton.

In the case of an incomplete model, the skeleton is tied to contain at least one border point. It implies that at least one intersection point is a border point. We consider that all the edges having a bounding vertex associated with a border point have infinite weight. See Fig.4 for an example with a 2D skeleton.

A skeleton can contain cycles, for example if a sequence of linking points has its two extremities in the same point. In this case, the extracted graph is not a tree, and we stop the pipeline for the current frame.

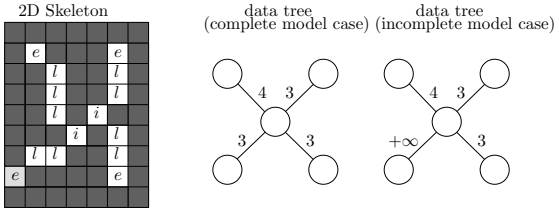

Fig. 4. Example of a data tree extraction from a 2D skeleton. On the left, the 2D skeleton, with pixels labeled by i,e and l representing respectively intersection, ending, and linking points. The light gray pixel represents a border point. In the middle, the data tree extracted from the skeleton, with the edge weights represented by numbers. On the right, the data tree extracted from the skeleton, in the case of an incomplete model.

4 Matching with A Priori Model

The data tree can be affected by different kinds of noise, which must be taken into consideration during the matching: due to the irregularities of the shape surface, skeleton branches without important topological signification can appear. These branches are not difficult to remove, but the problem is that it generates new

vertices in the tree. These vertices, after the removing of branches, uselessly split an edge (and its weight) into two parts, making difficult a good matching. The second kind of noise is due to the skeletonization: a cluster of vertices linked by weakly weighted edges in the data tree can correspond to a vertex with more than three neighbors in the model tree. In order to match the data tree with the model tree, we use the optimal homeomorphic alignment method [11], especially designed to be robust in regard of these kinds of noise.

4.1 Preliminary Definitions and Notations

In order to present the homeomorphic alignment, some definitions and notations are necessary. We denote a weighted tree as a triplet $T = (V, E, \omega)$, V is a finite set (called the *vertex* set), A is a subset of $V \times V$ (called the *edge* set), and ω is a mapping from A to $\mathbb{R}^+$, corresponding to the weights.

An homeomorphic alignment is based on edit operations: *deletion* consists of removing an edge in the tree, *resizement* consists of changing the weight of an edge, and *merging* consists of replacing two edges (a, b) and (b, c), where b has exactly two neighbors, by an unique edge (a, c) weighted by $\omega((a, b)) + \omega((b, c))$, and removing b from V. The *merging kernel* of a tree T is obtained by applying iteratively all possible mergings on T.

The *cost* of an operation is equal to the variation of weights in the tree before and after the application of the involved operation. Then, the cost of a deletion is equal to the deleted edge weight and is denoted by $\gamma(w, 0)$, where w is the weight of the deleted edge. In the same way, the cost of a resizement $\gamma(w, w')$ is equal to the difference between the former weight w and the new weight w' of the resized edge. A merging has a null cost, since the total weight of the tree is preserved by the operation.

In order to match trees with infinite weights, we have to take the convention that $\gamma(+\infty, +\infty) = 0$.

Two weighted trees $T = (V_T, E_T, \omega_T)$ and $T' = (V'_T, E'_T, \omega'_T)$ are said to be *isomorphic* if there exists a bijection $f : V_T \to V'_T$, such as for any pair $(x, y) \in V_T \times V_T$, $(x, y) \in E_T$ if and only if $(f(x), f(y)) \in E'_G$.

Two weighted graphs $T = (V_T, E_T, \omega_T)$ and $T' = (V'_T, E'_T, \omega'_T)$ are *homeomorphic* if there exists an isomorphism between the merging kernel of T and the merging kernel of T'.

4.2 Homeomorphic Alignment Definition

Let $T_1 = (V_1, E_1, \omega_1)$ and $T_2 = (V_2, E_2, \omega_2)$ be two weighted trees. Let $T'_1 = (V'_1, E'_1, \omega'_1)$ and $T'_2 = (V'_2, E'_2, \omega'_2)$ be weighted graphs obtained by deleting edges in T_1 and T_2, such that there exists an homeomorphism between T'_1 and T'_2 (not necessarily unique). Let $T''_1 = (V''_1, E''_1, \omega''_1)$ and $T''_2 = (V''_2, E''_2, \omega''_2)$ be the merging kernel of T'_1 and T'_2, respectively. By definition, there exists an isomorphism $\mathcal{I}$ between T''_1 and T''_2. The set of all couples of arcs $\mathcal{H} = \{(e, e'); e \in E''_1, e' \in E''_2, e' = \mathcal{I}(e)\}$ is called an *homeomorphic alignment* of T_1 with T_2 (see figure 5).

The *cost* $C_{\mathcal{H}}$ of $\mathcal{H}$ is the sum of the costs of all operations used to homeomorphically align T_1 and T_2: the deletion of edges in T_1 and T_2, to obtain T_1' and T_2' respectively, and the resizement for each edge $e_1 \in E_1''$ to the weight of $\mathcal{H}(e_1)$. An homeomorphic alignment with minimal cost is said to be optimal. The cost of an optimal homeomorphic alignment is called the homeomorphic alignment distance.

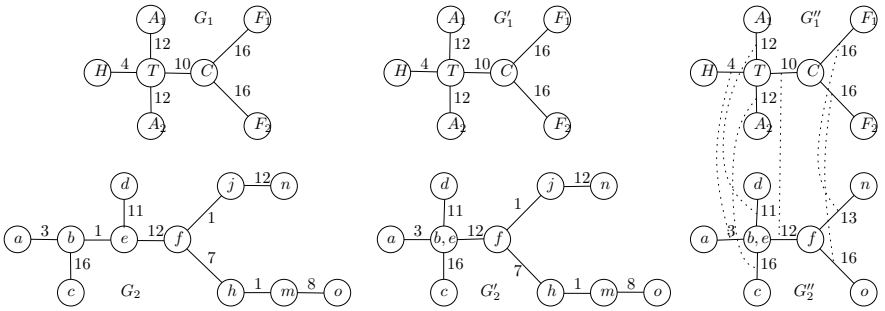

Fig. 5. Left: two trees G_1 and G_2. Middle: G_1' and G_2', obtained respectively from G_1 and G_2 by deletion of edges, and which are homeomorphic. Right: G_1'' and G_2'', which are respectively the merging kernels of G_1' and G_2', and which are isomorphic. Dotted lines represent an optimal homeomorphic alignment.

The "coordinate" constraints of the model are applied during the optimal homeomorphic alignment: the cost of the resizement of a model edge (v_M, v_M'), weighted by w_M, to the weight w_D of a data edge (v_D, v_D') is equal to $+\infty$ if there is a constraint C associated to (v_M, v_M'), and the 3D point associated to v_D is in a spatial position relative to the one associated to v_D' which is not compatible with C.

For the purpose of scale invariance, we start our method by normalizing the weights of the two trees, so that the sum of all non-infinite weights in a tree is equal to 1. Then, we compute the homeomorphic alignment distance between the trees, using the algorithm described in [11]. If the distance is greater than a given threshold T_d, which is defined by the user, we assume that the data tree is not enough similar to the model tree to provide a good matching, and we stop the pipeline for this frame. Otherwise, we use the optimal homeomorphic alignment to match the labels associated to the model vertices, with the 3D positions associated to the data vertices.

The number of non-aborted matchings, and their quality, obviously depend on the choice of the threshold value T_d : as both trees are normalized, $T_d = 2$ means that none matching will be aborted (both trees can be deleted), but the resulting matching can be null, or poor. In the other hand, a lower value yields a lower amount of non-aborted matchings, but with better probability of good matching.

5 Discrimination of Similar Limbs Using Model "Between" Constraints

In case of similar limbs, the matching with the model tree can generate multiple solutions. In the case of our model of hand, as the descriptions of index, middle finger, ring finger and little finger are identical, the matching will give a set of four possible positions for each finger. To solve this problem, we introduce the "between" constraints. It consists of constraining the position of some limb to lie between two other ones. A usual ternary relation "between" definition is based on collinearity [12] : a point is said to be between two other points P_1 and P_2 if it belongs to the segment P_1P_2. However, this definition is too restrictive. On the other hand, we could say that a point is between P_1 and P_2 if its projection on the line (P_1P_2) is between P_1 and P_2. In this case, the problem is that there exist triplets (P_1, P_2, P_3) such that each point is between the two others (for example, the three vertices of an equilateral triangle).

We propose a definition which is not too restrictive, and which gives at most one possibility of "betweenness" for three points P_1, P_2, P_3 : consider $\mathcal{B}$ the unique ball with diameter P_1P_2 which contains P_1 and P_2. The point P_3 is between P_1 and P_2 iff $P_3 \in \mathcal{B}$. An other formulation is that P_3 is between two points P_1 and P_2 iff the angle between $\overrightarrow{P_3P_1}$ and $\overrightarrow{P_3P_2}$ is greater than $\frac{\pi}{2}$. Then, if there exists a "between" constraint on model vertices, defined by a sequence $m_0, ..., m_n$, the corresponding data vertices $d_0, ..., d_n$, with associated positions $P_0, ..., P_n$ must be chosen such as, for each $i \in [1, n-1]$, $\overrightarrow{P_iP_{i-1}} \cdot \overrightarrow{P_iP_{i+1}} < 0$.

6 Implementation and Results

Our method has been tested on a computer with a processor Intel(R) Core(TM) 2 Quad Q8200 at 2.33 GHz, a GPU Nvidia(R) Geforce(TM) 9800 GT and 3 Go of RAM. Our implementation is implemented in C++, and the visual hull is computed on GPU, using GLSL. For the tests, we use two data sets: a set for hand pose estimation, produced by our team, and the dancer set produced by the INRIA Perception Group [1], for full body body pose estimation. We have tested our method for different voxel grid resolutions, in order to estimate the computation speed, the number of matchings and their quality.

Figure 6 shows the results of speed measurement. Since image data must be loaded from a data base instead of being captured online, shape acquisition cost is overestimated. It can be observed that our implementation reaches interactive time. However, our program is still a prototype, and can be widely optimized, e.g. by parallelizing the skeletonization step. The difference of initialization speed for both models can be explained by taking into consideration the time complexities of the two costly steps: the skeletonization time complexity is in $O(S_G + S_S * T_S)$, where S_G and S_S are respectively the size of the voxel grid and the size of the (in voxels), and T_S the thickness of the shape. The tree matching time complexity

[1] http://4drepository.inrialpes.fr/

is in $O(S^2 * (D * 2^{3*D} + S^2 * D))$, where S and D are respectively the maximal size and the maximal degree of the both trees. The values of S_S, T_S, S and D being higher for the hand model, the initialization speed for this model is slower than for the other one.

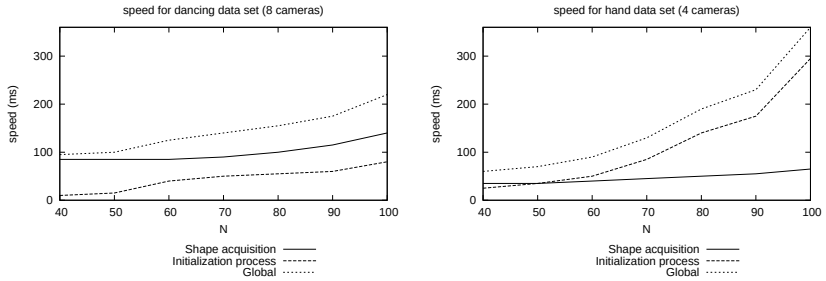

Fig. 6. Speed results for two kinds of model (Left: human body model; Right: hand model) for different sizes of grid. (The complete grid size is N^3 voxels).

Figure 7 shows some results of pose initialization. The accuracy of our method obviously depends on the size of the voxel grid. It is also the case for the proportion of non-aborted matchings, and for their probability to give robust matching, as shown in Table 6. The reasons of high rate of aborted matching are different for the two sets. In the case of the dancer set, it is due to the fact that the pose of the subject does not always allow the initialization : an arm can be too close to the torso, cycles can appear, or other cases of the same kind. In the case of the hand set, even if the hand always has a good pose for initialization (spread fingers), the positioning of the four cameras is not efficient enough to provide a good shape reconstruction.

grid side	threshold T_d	matching	FP
40	0.4	52%	16.3%
40	2.0	87.5%	15.4%
100	0.4	58%	11.2%
100	2.0	71.5%	9%

grid side	threshold T_d	matching	FP
40	0.4	40.6%	40%
40	2.0	75.6%	49.8%
100	0.4	18.1%	0.0%
100	2.0	61.5%	0.0%

Table 1. Matching count. Left: dancer data set. Right: hand data set. A matching is considered as a false positive (FP) if at least one part is not in a correct position.

7 Conclusion

In this paper, we have presented a new method for generic pose initialization, for markerless motion capture purpose. The performances of our method allow the initialization in interactive time, for an online usage. Our future works will focus on the detection of 2-degree joints, as elbows or shoulders, which can be detected by the presented method, and on the optimization of our prototype, in the aim to reach real time initialization.

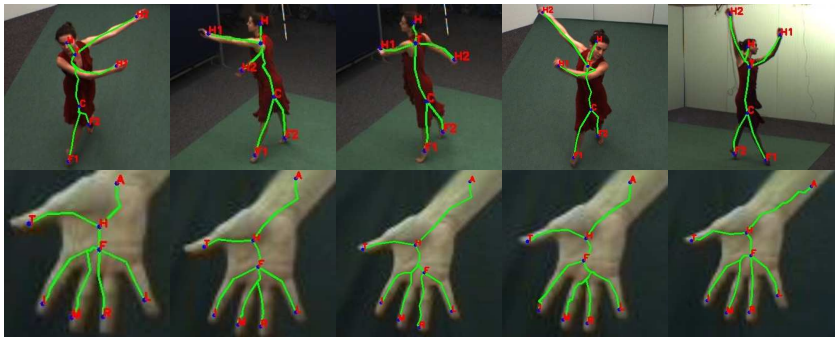

Fig. 7. Some examples of initial pose estimation results. In green, the skeleton, in blue the matched points, and in red, the corresponding labels.

References

1. Moeslund, T., Hilton, A., Krüger, V.: A survey of advances in vision-based human motion capture and analysis. Computer vision and image understanding **104**(2-3) (2006) 90–126
2. Alisi, T., Del Bimbo, A., Pucci, F., Valli, A.: Motion capture based on color error maps in a distributed collaborative environment. In: Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 4, IEEE Computer Society Washington, DC, USA (2004) 953–956
3. Michoud, B., Guillou, E., Briceno, H., Bouakaz, S.: Real-Time Marker-free Motion Capture from multiple cameras In: IEEE 11th International Conference on Computer Vision (2007). pp. 1–7.
4. Colombo, C., Del Bimbo, A., Valli, A.: A real-time full body tracking and humanoid animation system. Parallel Computing **34**(12) (2008) 718–726
5. 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, IEEE Computer Society (2006) pp. 389–396.
6. Kong, T.Y., Rosenfeld, A.: Digital topology: Introduction and survey. Computer Vision, Graphics, and Image Processing **48**(3) (1989) 357–393
7. Bertrand, G., Malandain, G.: A new characterization of three-dimensional simple points. Pattern Recognition Letters **15**(2) (1994) 169–175
8. Couprie, M., Bertrand, G.: New characterizations of simple points in 2D, 3D and 4D discrete spaces. IEEE Transactions on Pattern Analysis and Machine Intelligence **31**(4) (2009) 637–648
9. Bertrand, G.: On critical kernels. Comptes Rendus de l'Académie des Sciences, Série Mathématiques **1**(345) (2007) 363–367
10. Bertrand, G., Couprie, M.: Three-dimensional parallel thinning algorithms based on critical kernels. Technical report (2010)
11. Raynal, B., Couprie, M., Biri, V.: Homeomorphic Alignment of Edge-Weighted Trees. In: Proceedings of the 7th IAPR-TC-15 International Workshop on Graph-Based Representations in Pattern Recognition, Springer (2009) 134–143
12. Greenberg, M.: Euclidean and non-Euclidean geometries: Development and history. WH Freeman (1993)