Design of robust vascular tree matching: validation on liver

Arnaud Charnoz[1,3], Vincent Agnus[1], Grégoire Malandain[2], Stéphane Nicolau[1], Mohamed Tajine[3] and Luc Soler[1]

[1] IRCAD R&D, Strasbourg, France
[2] Epidaure Research group, INRIA, Sophia Antipolis, France
[3] LSIIT, CNRS/ULP, Strasbourg, France

Abstract. In this paper, we propose an original and efficient tree matching algorithm for intra-patient hepatic vascular system registration. Vascular systems are segmented from CT-scan images acquired at different times, and then modeled as trees. The goal of this algorithm is to find common bifurcations (nodes) and vessels (edges) in both trees.
Starting from the tree root, edges and nodes are iteratively matched. The algorithm works on a set of match solutions that are updated to keep the best matches thanks to a quality criterion. It is robust against topological modifications due to segmentation failures and against strong deformations.
Finally, this algorithm is validated on a large synthetic database containing cases with various deformation and segmentation problems.

1 Introduction

Motivations: liver tumors follow-up The main purpose of our work is to make an intra-patient follow-up of tumors (see our previous work [3]). This approach is motivated by the fact that the liver is a highly deformable organ and that tumors evolution study needs the determination of this volumic deformation. Now it is well-known that the most reliable landmarks to estimate deformations sustained by the liver are provided by its vascular network [2, 6, 12, 7, 10].

Previous works Related works propose algorithms to match and/or register vascular systems (brain, liver and, in a similar manner, lung airways). Generally, veins are modeled as graphs computed from segmented images and skeletons [11]. Some authors use some tree structure notions in their algorithms to register a tree with an image [2] or two trees [6]. Other approaches really match structures (nodes and vessels), but use general graph matching methods [12, 7, 8] or specific methods like subtree isomorphism [10] which do not take segmentation problems into account.

The oriented tree matching problem is more specific than graph matching because the structure is oriented and the path that connects two nodes is unique.

Moreover, it cannot be considered as a oriented subtree isomorphism problem because of segmentation problems. Indeed, the segmentation process can miss some vessels (edges). This implies a (virtual) pruning on both trees (for example an edge in a tree could be represented by several successive edges on the other tree) and thus the tree topology differs between acquisitions.

In our previous work [3], vascular systems are modeled as a tree and then tree vertices are matched together without taking possible segmentation errors into account. The previous algorithm works well on most branches but suffers from a lack of robustness in complex (but real) cases, especially on small branches where segmentation problems are important.

Proposal The new algorithm proposed in this paper does not to focus on the best solution (given two edge sets to match) like in our previous algorithm but on the most likely solutions which are updated along the process. The remainder of this paper is organized as follows. The first part presents our iterative oriented tree matching. We describe how we generate solutions at each step of the tree according to local criteria, and how we select the most likely ones with a global quality criterion.

In the second part, an evaluation of our algorithm on large database shows that in standard cases (20% of pruning or less), our algorithm matches 90% of nodes and that even in worst, cases 75% of matches are correct.

2 A new iterative tree matching technique

Skeletons computed from segmented vascular systems can be represented as an oriented tree. Thus, the proposed algorithm is a tree matching. The orientation symbolizes blood circulation flow. Nodes represent bifurcations and edges correspond to vessels between two bifurcations. And in our algorithm, some geometric vessel attributes are used (3D positions, radius, vessel path).

Vascular trees segmented for a patient follow-up represent the same vascular system; our goal is to find common bifurcations and registering them. However, their topology and 3D positions may differ due to segmentation errors and deformations applied on them. The main challenge consists in using tree topology to detect deformations, and in parallel, geometric informations to detect topology problems. In the following, we assume that we work on standard patient case, thus that tree roots are known (detection of vascular system entrance) and that tree deformations are small.

In next sections, we explain our tree matching. Firstly, we focus on a global view of the algorithm framework. Then, we detail the solution creations and the quality criteria that select at each step the most likely solutions.

2.1 Notations

In this paper, we use the notions of oriented tree [1]. We work on a tree noted $T = (V, E, r)$ where V represents the set of vertices, $E \subset V \times V$ the set of edges

and r the root. We note $\|T\|$ the number of vertices of T. For a node u in a tree T, $T(u)$ denotes the subtree of T induced from u. For a vertex v, $sons(v)$ denotes the set of their child vertices, $father(v)$ its father vertex, $out(v)$ the set of out-edges of v, and $in(v)$ its in-edge. For an oriented edge $e = (v, u)$, we define $src(e) = v$ and $tgt(e) = u$. For two vertices $v, w \in V$, $P(v, w)$ is the unique path in T linking v to w. Let e be an edge and v its target vertex, $DV_L(e) = \{u/u \in$ vertices of $T(v), \|P(v, u)\| \leq L\}$ denotes the descendant vertex set composed of L-first depth level vertices in subtree induced from e. For a vertex v, $T_+(v)$ denote the subtree $T(v)$ to which $father(v)$ is added to the vertex set and $in(v)$ to the edge set. More generally, A being a finite set, A^i is the i^{th} element of A, and $|A| = card(A)$. So $A = \{A^i : 1 \leq i \leq |A|\}$. We introduce also some notations on functions. $\mathcal{B}^k_{A,B}$ is the set of bijections from subset of k elements of A to subset of k elements of B. For $f \in \mathcal{B}^k_{A,B}$, $D_f(A)$ (resp. $I_f(B)$) is the domain (resp. the image) of f. So $f(D_f(A)) = I_f(B)$ and $|D_f(A))| = |I_f(B)| = k$

2.2 Framework of the algorithm

Our algorithm searches for the best tree matching between $T_1 = (V_1, E_1, r_1)$ and $T_2 = (V_2, E_2, r_2)$ starting from roots (r_1 match with r_2). Since possibilities are high, we propose to generate and select the most likely solutions. The algorithm starts by studying the root match and updates selected solutions when it explores and finds other possible matches in both trees. This means that some solutions selected at a process step can be eliminated later if they become unlikely. The likelihood of solutions is evaluated at each tree depth step with quality match criteria.

Our algorithm studies simultaneously N likely solutions ($S_i^0 \dots S_i^N$), i being the depth step. S_i^0 contains a set of matched vertices which descendant vertices are not studied yet. To continue building the final solution (all nodes analysed in both trees), the algorithm have to explore, one by one, these vertex matches. The exploration of one of them generates new solutions more complete noted S_{i+1}^j. Our algorithm progresses in solution exploration from S_i to S_{i+1}^j when a vertex match is analysed. Solutions are studied and developed simultaneously to be able to compare themselves.

In particular, the figure 1 shows the creation of the most likely solution S_1^i of the first tree depth step of process by exploring the root vertex match of the initial solution S_0. This figure details the constuction of match solutions from an initial vertex in S_0 (root match between r_1 and r_2). The first process step (1.a) consists in generating all out-edge match set hypotheses, noted HE^i (two hypotheses are shown among all: HE^A and HE^B). However, the number of solution is too high to be explored and a local quality criterion ($cost(HE^i)$) is computed to keep only the n best hypotheses (1.b). Then, we study each hypothesis to find next vertex match associated with an out-edge match. The figure shows the vertex match research of one out-edge match set $((a_1, a_2), (b_1, b_2))$ corresponding to the hypothesis HE^A.

The second process step (2.a) consists in generating path match hypotheses, noted ${}^nHP^i$, from the n^{th} out-edge match (for instance, two hypotheses are

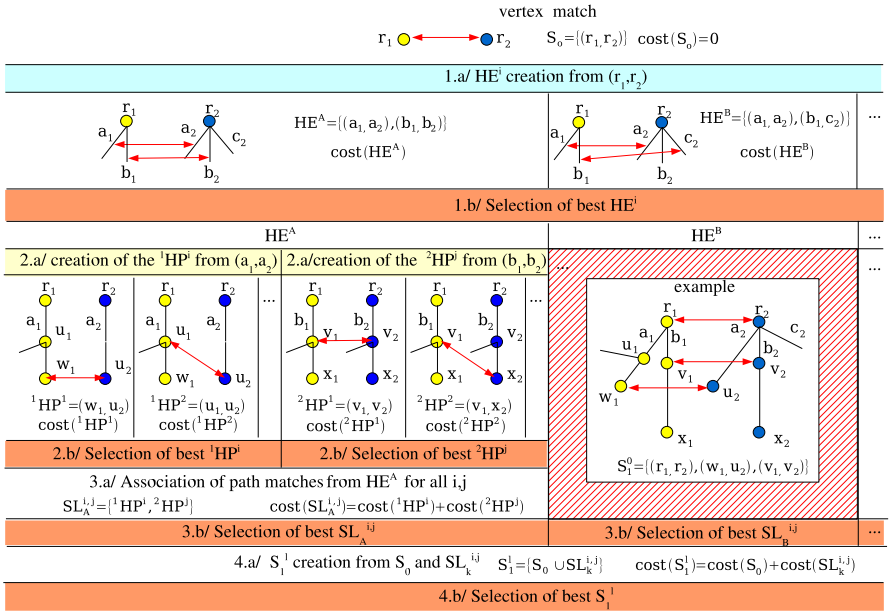

Fig. 1. This figure details the constuction of solutions on a tree depth step.

shown, ${}^1HP^1$ and ${}^1HP^2$ corresponding to the first out-edge match (a_1, a_2)). Once again, the number of solution is too high to be explored and a local quality criterion ($cost({}^1HP^i)$) is computed to keep only the m best hypotheses (2.b). When the algorithm finishes to compute all best path match hypotheses for each out-edge match, we test all possible permutations and reassemble them (3.a) in local solutions noted $SL_k^{i,j}$ where k is the k^{th} out-edge match set hypothesis, i and j the path match chosen for the first out-edge matches. Once again, only the best of them are selected (3.b). In (4.a) and (4.b), we build and select all best solutions resulting of different out-edge matches set hypotheses and path match hypotheses. A global solution example possible is shown on the right in the figure and is noted S_1^0. The algorithm restart this process from one of vertex matches included in different solutions S_1^l.

In next subsections, we detail how we generate out-edge match set hypotheses and path match hypotheses and how we select the best most likely one using either local or global quality criterion.

2.3 Hypothesis generation

Step I: Out-edge match set hypothesis This step consists in generating out-edge match possibilities from a vertex match, noted (v_1, v_2), to continue the match of $T_1(v_1)$ and $T_2(v_2)$. Two possiblilities are shown on figure 2. Let $O_1 = out(v_1)$ and $O_2 = out(v_2)$. An out-edge match set hypothesis is noted $\mathcal{HE}(v_1, v_2)$. An hypothesis $\mathcal{HE}^i(v_1, v_2)$ is represented by an out-edge match set $\mathcal{E}_f(v_1, v_2)$ which characterizes a match between k elements of O_1 with k elements of O_2. $\mathcal{E}_f(v_1, v_2) = \{(e, f(e))/e \in D_f(O_1), f \in \mathcal{B}^k_{O_1,O_2}\}$. This out-edge match set

assumes implicitly that some out-edges of O_1 (respectively O_2) noted $D_f(O_1)^c$ (respectively $I_f(O_2)^c$) have no association. Thus, some subtrees have no match in the other tree. A cost will be attributed to them to be able to compare hypotheses with or without lost subtrees. Thus these lost subtrees must be retained in the solution (this was deliberately omitted in the figure 1 explications for clarity reason). Let $\phi(v, OE) = \{T_+(u)/\forall(v,u) \in OE\}$ be the subtree induced by a vertex and a subset OE of its out-edges. We note $O_{min} = min(|O_1|, |O_2|)$, then possible hypotheses are given by:

$$\mathcal{HE}(v_1, v_2) = \bigcup_{k=0...O_{min}} \bigcup_{f \in \mathcal{B}^k_{O_1,O_2}} \left\{(\mathcal{E}_f(v_1, v_2), \phi(v_1, D_f(O_1)^c), \phi(v_2, I_f(O_2)^c)\right\}$$

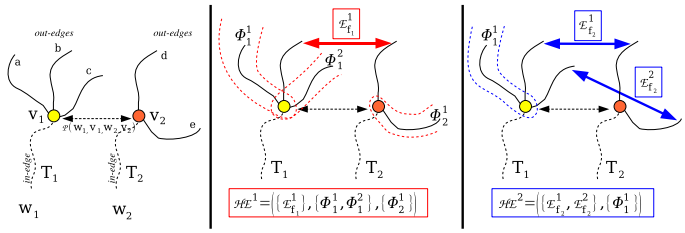

Fig. 2. Illlustration of a creation of out-edge match set hypotheses from a vertex match. The left figure resumes the vertices match between v_1 and v_2. The others show two possible solutions, in which an out-edge match set is chosen for each solution($\{\mathcal{E}^1_{f_1}\}$ for $\mathcal{HE}^1$ and $\{\mathcal{E}^1_{f_2}, \mathcal{E}^2_{f_2}\}$ for $\mathcal{HE}^2$). Hypotheses suppose that some out-edges do not have their equivalent in other trees and thus that the corresponding subtree is not matched ($\{\phi^1_1, \phi^2_1, \phi^1_2\}$ for $\mathcal{HE}^1$ and $\{\phi^1_1\}$ for $\mathcal{HE}^2$).

Step II: Path match hypothesis This step consists in generating path match possiblilities from an out-edge match, noted (e_1, e_2). Tree possiblilities are shown on figure 3. We assume that an edge $e_1 \in O_1$ and an edge $e_2 \in O_2$ match (representing the same starting vessel). the algorithm must find the next common bifurcation in subtrees $T_1(tgt(e_1))$ and $T_2(tgt(e_2))$ closest to v_1 and v_2. Due to segmentation defects, $tgt(e_1)$ and $tgt(e_2)$ do not necessarily represent the same bifurcation (this case happens frequently when branches are small). We search a vertex match in subtrees and not only between $tgt(e_1)$ and $tgt(e_2)$ (Fig. 3).

The research of next vertex match is restricted on the L first level of subtrees $T_1(tgt(e_1))$ and $T_2(tgt(e_2))$. Thus, we search the best vertex match between $DV_L(e_1)$ and $DV_L(e_2)$. In our algorithm, L is empirically choosen and is generally fixed 3.

Now, let (w_1, w_2) be a vertex match with $w_1 \in DV_L(e_1)$ and $w_2 \in DV_L(e_2)$. w_1 does not necessarily equal $tgt(e_1)$ and this vertex match defines a path match $\mathcal{P}(v_1, w_1, v_2, w_2) = (P(v_1, w_1), P(v_2, w_2))$. This match of pathes implies that some subtrees starting from them are not matched. We note this forest of no match subtrees $\psi(v, w) = \{T_+(u), u \in sons(k), k \in V_P, T_+(u) \cap P(v, w) = \{k\}\}$ where $V_P = \{\text{vertices of } P(v, w)\}\backslash\{v, w\}$. if we note $v_1 = src(e_1)$ and $v_2 = src(e_2)$, the set of possible path matches is defined as:

$$\mathcal{HP}(e_1, e_2) = \bigcup_{w_1 \in DV_L(e_1)} \bigcup_{w_2 \in DV_L(e_2)} \left\{\mathcal{P}(v_1, w_1, v_2, w_2), \psi(v_1, w_1), \psi(v_2, w_2)\right\}$$

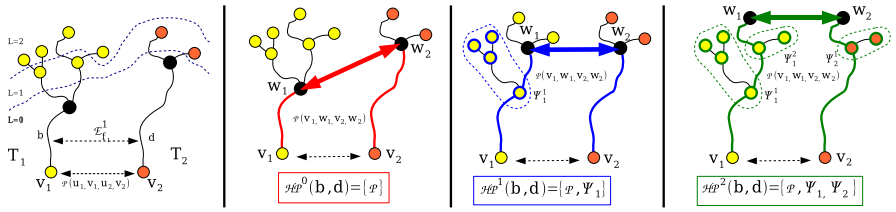

Fig. 3. Figures show the creation of path match hypotheses from an out-edge match. The different depths L are shown on the left figure and three solutions are illustrated.

2.4 Hypotheses selection

In the previous section, we have seen how to generate all out-edge match set hypotheses and path match hypotheses. matches. However, all possible tree matching solutions cannot be explored due to huge combinatorial possibilities : Selections must be made.

Therefore some cost functions are computed to determine the quality of matches. In our algorithm, two types of cost are computed : a global cost to determine the solution quality $S_i(cost(S_i))$, and two local costs to determine the quality of each out-edge match set hypothesis $\mathcal{H}E^i$ and each path match hypotheses $\mathcal{H}P^i$ (Fig. 1).We give here general expression of the cost functions. Each term of these cost functions are detailed in the next paragraph.

We define the two following local costs that select the most likely $\mathcal{H}E^i$ and the most likely $\mathcal{H}P^i$.

$$\begin{aligned} cost(\mathcal{H}E^i(v_1,v_2)) &= \textstyle\sum_{j=1}^{N_1} cost(\mathcal{E}_f^j(v_1,v_2)) + \sum_{j=1}^{N_2} cost(\phi^j(v_1, D_f(O_1)^c)) \\ &\quad + \textstyle\sum_{j=1}^{N_3} cost(\phi^j(v_2, I_f(O_2)^c)) \\ cost(\mathcal{H}P^i(e_1,e_2)) &= cost(\mathcal{P}(v_1,w_1,v_2,,w_2)) \ + \textstyle\sum_{j=1}^{N_4} cost(\psi^j(v_1,w_1)) \\ &\quad + \textstyle\sum_{j=1}^{N_5} cost(\psi^j(v_2,w_2)) \end{aligned}$$

The global cost expression selects the most likely solutions S_i. If algorithm explores a vertex match (v_1, v_2) of a current solution S_i, we obtain new solutions S_{i+1}^l (for example $S_{i+1}^l = S_i \bigcup_{e \in D_f(O_1)} \mathcal{H}P^T(e, f(e))$ where T is different for each out-edge match). S_{i+1}^l is caracterised by an out-edge match set from v_1 and v_2 and a path match for each out-edge match. The value of solution cost is given by :

$$\begin{aligned} cost(S_{i+1}^l) = &\sum_{e \in D_f(O_1)} cost(\mathcal{H}P^T(e, f(e)) + \textstyle\sum_{j=1}^{N_2} cost(\phi^j(v_1, D_f(O_1)^c)) \\ &+ \textstyle\sum_{j=1}^{N_3} cost(\phi^j(v_2, I_f(O_2)^c)) \ + cost(S_i) \end{aligned}$$

In these equations, N_i represents the different set cardinals. Note that there are three kinds of costs: a cost between two out-edges, a cost between two paths and a cost for subtrees which have no correspondence in the other tree.

Physical cost used: We define here the basic functions that allow to compare the geometric properties of match solutions (vertex or edge). The cost C_E represents the distance between extremity edges, C_R represents the radius difference along edges (vessels), C_{OE} represents the difference between the out-edges number from each extremity edge, C_S represents the scale between edges and C_A represents the angle between edges. These costs are normalized thanks to a truncated quadratic robust estimator ρ and its empirically chosen parameter α [5]. The perfect match is symbolised by a zero cost.

We remind that an edge e represents a vessel between two bifurcations. In the following cost formules, $e(t)$ is the 3D parametric curve representation of the vessel, $r(t)$ represents the vessel's radius along the curve and l is the curve's length. Thus by default, $e(t)$ (respectively $r(t)$) is defined between $t \in [0, l]$ where $e(0)$ and $e(l)$ represent the vessel extremities. We note $\boldsymbol{e}$ the vector between two points $e(0)$ and $e(l)$. For each cost comparison between e_1 and e_2, we supposed that $e_1(0) = e_2(0)$.

$$C_E(e_1, e_2) = \rho\big(\|e_1(l_1) - e_2(l_2)\|,\ \alpha_E\big)$$

$$C_R(e_1, e_2) = \rho\Big(\int_0^1 \|r_1(s \times l_1) - r_2(s \times l_2)\| ds,\ \alpha_R\Big)$$

$$C_S(e_1, e_2) = \rho\big(1 - \frac{l_1}{l_2},\ \alpha_S\big), \quad \text{if } l_1 < l_2$$

$$C_A(e_1, e_2) = \rho\big(1 - \tfrac{\|\boldsymbol{e_1}.\boldsymbol{e_2}\|}{\|\boldsymbol{e_1}\|\|\boldsymbol{e_2}\|},\ \alpha_A\big)$$

$$\rho(v, \alpha) = \begin{cases} \big(\frac{v}{\alpha}\big)^2 \text{ if } |v| < |\alpha| \\ 1 \end{cases}$$

Out Edge Match Cost: $cost(\mathcal{E}_f^i(v_1, v_2))$ compare edge orientation and radius.

$$cost(\mathcal{E}_f^i(v_1, v_2)) = \frac{1}{\gamma_1 + \gamma_2}(\gamma_1 C_A(e_1', e_2') + \gamma_2 C_R(e_1', e_2'))$$

with $e_1'(t)$ (respectively $e_2'(t)$) is $e_1(t)$ ($e_2(t)$) defined on $[0, \min(l_1, l_2)]$ and where γ_1 and γ_2 are weights used to favor robust characteristics in the algorithm.

Path Match Cost: In $cost(\mathcal{P}(v_1, w_1, v_2, w_2))$, weights are added to favor path matches with same small length, same orientation and same vessel radius and vessel extremities.

$$cost(\mathcal{P}(v_1, v_2, w_1, w_2)) = \frac{1}{\beta_1 + \ldots + \beta_4}\big(\beta_1 C_A(e_1, e_2) + \beta_2 C_R(e_1, e_2) + \beta_3 C_S(e_1, e_2) + \beta_4 C_E(e_1, e_2)\big) + \min_i\ cost(\mathcal{H}E^i(w_1, w_2))$$

with $e_1 = P(v_1, w_1)$ and $e_2 = P(v_2, w_2)$. This cost is composed of a local cost representing the current edge comparison and the cost of the next best out-edge matches from current extremity edges. This last term allows the algorithm to be more efficient and robust because an information is added on the vessel extremity similarity.

No Match Tree Cost: We have previously considered a cost for no inclusion subtrees in the match solution represented by $\phi^i(u, E)$ and $\psi^j(u, w)$. Each subtree $T_+(v)$ is defined by a vertex v. Cost computation is the same in both cases and is noted $costLost(v)$. We highlight that choosing the weight of this cost is difficult and depends on other match costs. If this cost is too high, then all nodes are matched (we forbid a subtree lost and then the algorithm is not robust against segmentation problem). Conversely, if it is too low, the algorithm does not select matches (the algorithm looses all branches). Hence a minimum cost $cost_{min}$ is introduced.

$$costLost(v) = R'(v) + \sum_{w_k \in sons(v)} costLost(w_k)$$

$$\text{with: } R(v) = \frac{1}{\mu_1 + \mu_2}\left(\mu_1 \frac{\|T(v)\|}{\|T\|} + \mu_2 \int_0^1 \|r(s) - R_{min}\| ds\right)$$
$$R'(v) = \max(R(v), cost_{\min})$$

the constant R_{min} corresponds to minimum radius to detect vessels in images. This cost is composed of two terms, the first one give us an information on the subtree surface to avoid loosing big subtree, the second one is an information on vesssel radius to avoid loosing large vessel (vessels with large radius are not concerned by segmentation problem and thus can be found in the other tree)

3 Experiments and validation

3.1 Virtual patient creations

To test and validate our algorithm, we have worked on a liver and its hepatic vascular system. To work on a complex vascular system (380 nodes), the Visible Man (cf. The Visible Human Project of NLM) has been segmented. The matching is harder (more bifurcations) than for a real patient case. This leads to better tests to evaluate the algorithm robustness.

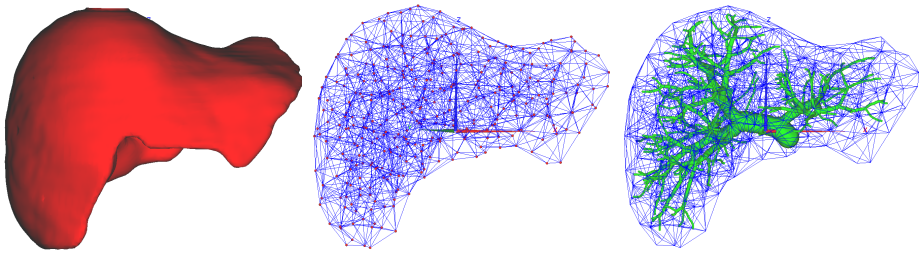

Fig. 4. The surgery simulator prototype is used to simulate liver and vascular system deformations thanks to a volumic model. [**Left**] Surfacic model [**Center**] Volumic model [**Right**] Volumic model and portal vascular system.

To simulate deformations, we have used the minimally invasive hepatic surgery simulator prototype developed at INRIA [9]. The goal of this simulator is to provide a realistic training framework to learn laparoscopic gestures. For this paper,

we used it only to simulate deformations of the liver and its vascular system (Fig. 4). This simulator uses complex biomechanical models, based on linear elasticity and finite element theory, which include anisotropic deformations.

To simulate segmentation errors on our phantom, we have pruned random tree branches. The probability to loose small vessels is greater than to loose large ones (Fig. 5).

To test the algorithm, a database of 600 patient follow-up cases has been generated from 2 types of deformations : a small (mean distance between commun points = 9 mm) and a strong (30mm) and 5 pruning steps (0,10,20,30,40 %) with on each step, 20 randomly generated prunings (Tab. 1 and 2).

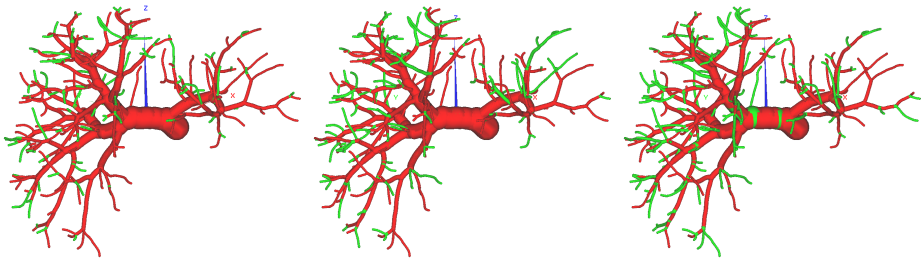

Fig. 5. The visible Man's portal vascular system is randomly pruned to loose approximately 20%, 30% and 40% of length in both trees. Lost branches appear in green.

3.2 Results on a virtual patient

Algorithm parameters have been chosen and fixed empirically to work more efficiently on all these cases. These paramater choices and their different consequences on the algorithm process (error, robustness, procesus time) are not detailed here but in a future journal paper.

The process is fast (about 10 minutes to register 380 nodes on 1GHz PC). Two process results are shown (Fig. 6) for a small and a strong deformation and pruned to loose approximately 20% of surface branches in both trees. Tab. 1 and 2 show that on small deformations the algorithm is very robust (practically all possible matches with a small standard deviation were found in the different cases) even with large pruning. With strong deformations and large pruning, the process is less robust (around 80%).

Results are reported in terms only of node identification. In fact, the consequences of performing an incorrect connection may be much larger in a proximal branch than peripherally. However, we noticed that most of the match errors (incorrect node correspondences and lost branches) are localized on terminal edges. On these nodes, the algorithm suffers from a lack of information (no subtree, dense node concentrations, small vessels). This makes the matching task harder.

To conclude, deformations and prunings (20% or less) used for these tests correspond with standard observed real cases. For this values, experts consider our algorithm efficient (sensitivity and similarity greater than 90%) to find a good approximation of the 3D liver deformation.

% T_1 pruning	% T_2 pruning	common nodes	% sensitivity	% efficiency
0	0	380 ± 0	100 ± 0	100 ± 0
0	10	314 ± 7	98,7 ± 0,8	98,9 ± 0,7
0	20	242 ± 8	96,2 ± 3,1	97,5 ± 0,7
0	30	189 ± 7	92,1 ± 5,2	94,9 ± 1,5
0	40	144 ± 3	84,0 ± 5,8	90,9 ± 2,5
10	10	260 ± 9	98,5 ± 0,7	97,8 ± 1,1
10	20	203 ± 7	97,4 ± 1,0	94,7 ± 1,3
10	30	164 ± 6	94,9 ± 2,5	93,0 ± 1,5
10	40	128 ± 6	90,5 ± 6,4	89,7 ± 4,9
20	20	169 ± 8	96,2 ± 1,8	92,8 ± 1,0
20	30	135 ± 10	96,2 ± 1,7	89,9 ± 1,6
20	40	108 ± 6	90,3 ± 6,9	85,4 ± 3,3
30	30	115 ± 7	94,3 ± 3,6	87,0 ± 2,6
30	40	90 ± 6	90,8 ± 6,5	82,6 ± 3,5
40	40	71 ± 6	93,5 ± 3,2	79,5 ± 3,8

Table 1. Matching results with a small deformation: Each line represents a pruning configuration with 20 randomly computed cases. Each column shows the mean result of these 20 cases with their magnitude. Three results are shown : the number of common nodes (match number in the reference solution) between both pruned trees, the sensitivity which is the number of correct found matches among the number of solutions matches and the efficiency which is the number of correct found matches among the number of found matches (correct and uncorrect).

4 Conclusion

The purpose of this paper was to present the design of our original new robust method to match liver vascular systems between two CT/NRI acquisitions. This method is well adapted, fast and robust on a complex vascular system. Thanks to the virtual database generated by the INRIA simulator, we have tested numerous configurations.

Currently, we are working on the second step of tumor follow-up: the estimation of liver deformation computed from the vascular system matching. In parallel, we have started first tests on a real patient database with very encouraging results (Fig. 7). These results will be detailed in a future paper.

Then, we will validate our works with surgeons on a real patient database with the collaboration of the Strasbourg hospital and also propose a new tool for automatic diagnosis of tumor evolution in the liver.

Acknowledgments We thank the Strasbourg hospital and their surgeons for providing images as well as their advice on "standard" deformations applied on the liver. This work has benefited from the segmentation program of the vascular system developed by the IRCAD R&D team. The realistic liver deformations are provided by the INRIA simulator from the Epidaure project. Many thanks to Clément Forest [4] for his assistance during the use of the simulator.

% T_1 pruning	% T_2 pruning	common nodes	% sensitivity	% efficiency
0	0	380 ± 0	100 ± 0	100 ± 0
0	10	311 ± 9	97,7 ± 1,0	98,7 ± 0,6
0	20	246 ± 6	94,4 ± 0,8	95,8 ± 0,8
0	30	195 ± 10	75,1 ± 37,3	76,9 ± 35,1
0	40	147 ± 6	69,6 ± 34,5	72,3 ± 32,7
10	10	257 ± 6	95,7 ± 1,8	96,0 ± 1,3
10	20	206 ± 5	93,0 ± 2,7	92,5 ± 1,3
10	30	162 ± 7	92,9 ± 2,2	91,4 ± 1,7
10	40	128 ± 7	88,2 ± 4,5	86,5 ± 3,0
20	20	169 ± 7	92,9 ± 4,2	91,3 ± 1,1
20	30	138 ± 7	90,3 ± 4,8	87,6 ± 2,6
20	40	109 ± 6	88,9 ± 5,4	85,6 ± 2,6
30	30	114 ± 7	90,0 ± 7,3	85,6 ± 2,0
30	40	93 ± 6	91,6 ± 2,5	82,9 ± 3,6
40	40	73 ± 4	87,7 ± 5,8	78,0 ± 4,4

Table 2. Matching results with a strong deformation: see description Tab. 1. The standart deviation of cases (0-30%) and (0-40%) is very high. Our algorithm attains its limits when we have a great difference between pruning (topology of trees become very different) associated with a strong deformation. These configuration cases occurre infrequently.

References

1. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.
2. S.R. Aylward, J. Jomier, S. Weeks, and E. Bullitt. Registration and analysis of vascular images. *IJCV*, 55(2-3):123–138, 2003.
3. A. Charnoz, V. Agnus, and L. Soler. Portal vein registration for the follow-up of hepatic tumours. In *MICCAI*, volume 3217 of *LNCS*, pages 878–886, Saint-Malo, France, September 2004. Springer Verlag.
4. C. Forest, H. Delingette, and N. Ayache. Surface contact and reaction force models for laparoscopic simulation. In *International Symposium on Medical Simulation*, June 2004.
5. F. R. Hampel, E. M. Ronchetti, P. J. Rousseeuw, and W. A. Stahel. Robust statistics. In *John Wiley and Sons*, New York, 1986.
6. T. Lange, S. Eulenstein, M. Hunerbein, H. Lamecker, and P.-M Schlag. Augmenting intraoperative 3d ultrasound with preoperative models for navigation in liver surgery. In *MICCAI*, volume 3217 of *LNCS*, pages 534–541, Saint-Malo, France, September 2004. Springer Verlag.
7. Y. Park. *Registration of linear structures in 3-D medical images*. PhD thesis, Osaka University, Japan. Departement of informatics and Mathematical Science, 2002.
8. M. Pelillo, K. Siddiqi, and S.W. Zucker. Matching hierarchical structures using association graphs. *PAMI*, 21:1105–1120, November 1999.
9. G. Picinbono, J-C. Lombardo, H. Delingette, and N. Ayache. Improving realism of a surgery simulator: linear anisotropic elasticity, complex interactions and force extrapolation. *JVCA*, 13(3):147–167, jully 2002.
10. C. Pisupati, L. Wolff, W. Mitzner, and E. Zerhouni. Tracking 3-d pulmonary tree structures. In *MMBIA*, page 160. IEEE Computer Society, 1996.

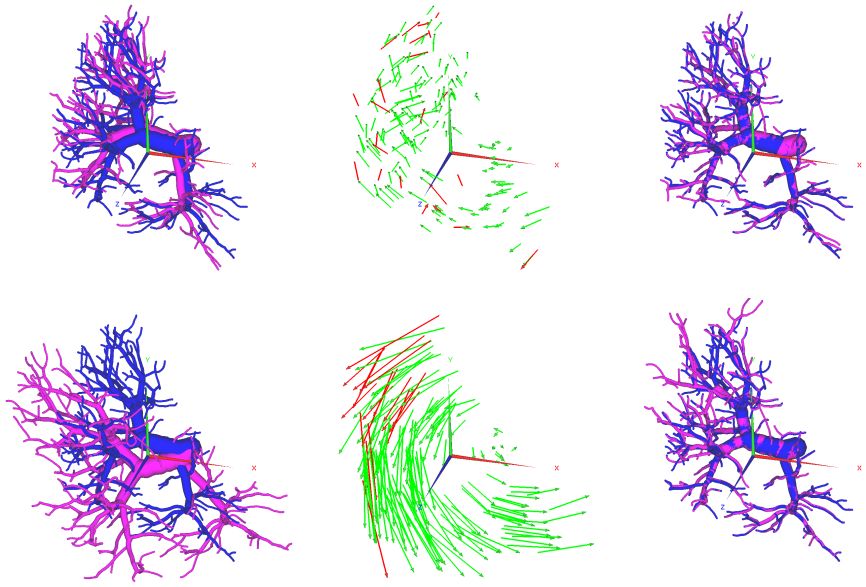

Fig. 6. **[Top]** On the left, small deformation case is pruned at 20%. The center figure shows the result of our oriented tree matching, good matches are represented by green arrows and represent 95% of all nodes and wrong matches by red arrows. The right figure shows the tree registration after the process. **[Bottom]** A strong deformation with an equivalent pruning where the algorithm find 91% of all nodes.

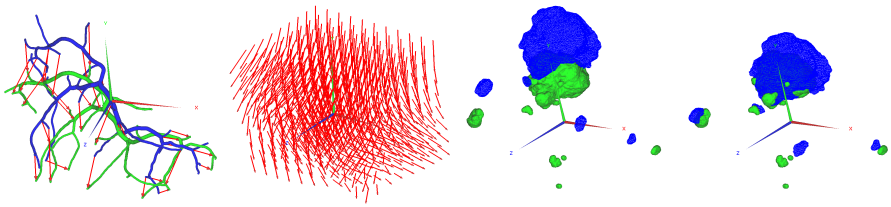

Fig. 7. **[a]**Real patient where the vascular system has been matched where vertex matches are shown in red. **[b]**Deformation field computed from matches. **[c,d]**Tumors before and after registration.

11. L. Soler, H. Delingette, G. Malandain, J. Montagnat, N. Ayache, J.-M. Clément, C. Koehl, O. Dourthe, D. Mutter, and J. Marescaux. A fully automatic anatomical, pathological and fonctionnal segmentation from CT-scans for hepatic surgery. In *Medical Imaging*, SPIE proceedings, pages 246–255, San Diego, February 2000.
12. J. Tschirren, K. Palágyi, J.M. Reinhardt, E.A. Hoffman, and M. Sonka. Segmentation, Skeletonization, and Branchpoint Matching - A Fully Automated Quantitative Evaluation of Human Intrathoracic Airway Trees. In *MICCAI*, volume 2489 of *LNCS*, pages 12–19. Springer-Verlag, 25 September 2002.