2020 IEEE International Conference on Robotics and Automation (ICRA)
31 May - 31 August, 2020. Paris, France

Robotic needle insertion in moving soft tissues using constraint-based inverse Finite Element simulation

Paul Baksic[1], Hadrien Courtecuisse[1], Christian Duriez[2] and Bernard Bayle[1],

Abstract— **This paper introduces a method for robotic steering of a flexible needle inside moving and deformable tissues. The method relies on a set of objective functions allowing to automatically steer the needle along a predefined path. In order to follow the desired trajectory, an inverse problem linking the motion of the robot end effector with the objective functions is solved using a Finite Element simulation. The main contribution of the article is the new constraint-based formulation of the objective functions allowing to: 1) significantly reduce the computation time; 2) increase the accuracy and stability of the simulation-guided needle insertion. The method is illustrated, and its performances are characterized in a realistic framework, using a direct simulation of the respiratory motion generated from in vivo data of a pig. Despite the highly non-linear behavior of the numerical simulation and the significant deformations occurring during the insertion, the obtained performances enable the possibility to follow the trajectory with the desired accuracy for medical purpose.**

I. INTRODUCTION

Radio-Frequency Ablation (RFA) is a percutaneous therapy that uses heat to destroy cancer cells. Such treatments provide alternative therapeutic options for the management of tumors or metastasis that are considered unresectable with traditional approaches (concerns about the age, the extent or localization of the disease). RFA has proven to be a viable option with limited complications for patients with hepatic diseases of small volume [1].

Despite their benefits, such needle-based approaches raise significant difficulties. The required accuracy for RFA of liver tumors is typically around 2 to 3 mm [2], which is particularly challenging as needles are manipulated from outside the patient using intraoperative images offering poor visibility of internal structures. Besides, tumors can be located deeply inside the liver volume with poor access conditions (for instance in posterior segments VI or VII), reducing the ability to control the needle tip close to the target. In addition, the liver is a very soft organ that tends to deform in contact with the diaphragm during the respiratory motion resulting in non-uniform displacements (up to 5 cm).

The previous difficulties explain why interventional radiologists usually need several attempts to reach the targeted zone or to operate under apnea [3], which increases the intervention time and the risks for the patient. Robotic systems have the potential to assist percutaneous needle insertion to overcome limitations due to human factors and increase the accuracy of tool positioning. In the last decade, numerous solutions have been proposed for robotic needle steering. Most of them are image-guided [4], [5] since they rely on data coming from imaging devices (MR, CT, US or a combination of these modalities).

The most common strategy to control a needle with a robotic system is to manipulate the needle base position in order to modify the needle tip path inside tissues [4], [6]. In this process, the six degrees of freedom (DOF) of the needle base are controlled by the robot end effector to which the needle base is attached. Relating these 6 DOF to the needle tip motion is a difficult inverse problem, especially if this problem has to be solved in real time. Other approaches exploit the possibility of changing the curvature of beveled-tip flexible needles. The method called *duty-cycling* consists in spinning the needle along its insertion axis in order to modify the path taken by the beveled tip [7], [8], [9]. However, the applicability of duty-cycling approaches in hepatic tissues remains an open question due to the risk of damaging the organ. Other authors [10] propose to deform the tissue to move the target on the needle's trajectory. However, such methods have only been applied for breast and prostate surgery.

Recent surveys are of particular interest regarding robotic needle guidance systems [11], [12]. While some patient-mounted robots partially compensate for physiological motions [6], the motions of internal organs are by far too complex and different from skin motions to be compensated for using purely passive solutions. Robotized insertions in liver tissues are usually performed during apnea [6], [13] in order to reduce organs motion. Yet, the complete procedure is generally too long for a single apnea. Therefore, the deformation of both the organs and surgical needles remains an open problem that limits the development of automatic tasks performed by the robots in the operating room [12].

In the context focused on deformations, the control of needle insertions evokes the field of soft robotics. Recently, original works propose to rely on Quadratic Programming (QP) to solve equations of motion and control soft robots in real time [14]. However, the optimization problem is defined using the actuators' space of the considered robot which has not yet been applied to needle insertion where the motion of the tissue is completely independent from the robotic system.

The coauthors of the present article recently proposed to rely on a Finite Element simulation to steer a needle in deformable environment [15], [16]. The robot-assisted needle insertion relies on the actuation of the 6 DOF of the base of the needle. Assuming a reachable trajectory defined by the radiologist at the planning step, the method provides

[1]ICube Laboratory, UMR 7357, CNRS - University of Strasbourg, Strasbourg, France.
[2]Defrost team, CRIStAL laboratory, UMR 9189, Inria - University of Lille, Lille, France.

displacements of the needle base in order to simultaneously follow the trajectory and take into account needles and tissue deformations during the insertion. Yet, the method is currently limited to insertions where deformations only result from the needle penetration and displacements of the robot.

In this paper, the problem of needle insertion during respiratory motions is addressed specifically. Numerical optimizations and stabilization techniques are proposed to significantly reduce the computation time of inverse steps while increasing the stability of the system. In the proposed scenario, the performances achieved by this novel inverse FE simulation method are crucial to enforce the accuracy and the convergence during the insertion. The method is illustrated and characterized in a realistic simulation where the motion of the liver during breathing cycles was generated from in vivo data of a pig under anesthesia.

II. BACKGROUND

In this section the necessary FE simulation background is summarized to ease the understanding of the present article.

A. Real time finite element methods

Following Courtecuisse et al. [17], an implicit integration of Newton's second law of two deformable bodies $n \in \{1,2\}$ (needle and tissue) interacting through Lagrangian Multipliers requires to solve the Karush-Kuhn-Tucker (KKT) system:

$$\begin{cases} \mathbf{A}_1 \Delta\mathbf{v}_1 + \mathbf{H}_1^T \boldsymbol{\lambda} = \mathbf{b}_1 & (1) \\ \mathbf{A}_2 \Delta\mathbf{v}_2 + \mathbf{H}_2^T \boldsymbol{\lambda} = \mathbf{b}_2 & (2) \\ \mathbf{H}_1 \Delta\mathbf{v}_1 + \mathbf{H}_2 \Delta\mathbf{v}_2 = \boldsymbol{\delta} & (3) \end{cases}$$

with $\mathbf{A}_n$ the so-called *stiffness matrix* encoding the linearized elastic properties of the material (inertial, stiffness and damping) $\mathbf{b}_n$ the forces exerted on the model (external and internal) at the end of the time step and $\Delta\mathbf{v}_n$ the variation of velocities during the time step. $\mathbf{H}_n$ stands for the *Jacobian of the constraints* linking the motion's space to the constraint's space. $\mathbf{H}_n$ gathers three types of constraints:

1) Bilateral constraints used to attach the needle's base to the robot end effector.
2) Needle insertion constraints coupling displacements of the needle with the tetrahedral mesh [18].
3) Registration constraints used to impose displacements in order to maintain the consistency of the models.

Lagrangian multipliers $\boldsymbol{\lambda}$ corresponding to the models response forces enforce the constraints and $\boldsymbol{\delta}$ are their respective violations.

The KKT system is solved with a four-step method [17]:
1- Free motion: Isolating $\Delta\mathbf{v}_n$ in equations (1) and (2) and putting $\boldsymbol{\lambda} = \mathbf{0}$ allows decomposing the problem. We first consider the internal and external forces applied to the models without considering constraints:

$$\begin{cases} \Delta\mathbf{v}_1^{\text{free}} = \mathbf{A}_1^{-1}\mathbf{b}_1 \\ \Delta\mathbf{v}_2^{\text{free}} = \mathbf{A}_2^{-1}\mathbf{b}_2 \end{cases} \quad (4)$$

These equations can be solved efficiently using a Conjugate-Gradient method on GPU [19]. Yet, in the context of inverse simulations, this step remains expensive since $\mathbf{A}_n$ are large matrices whose dimensions are proportional to the number of nodes of the meshes.
2- Constraints definitions: Matrices $\mathbf{H}_n$ are defined at the beginning of the time step either performing a collision detection or associating nodes in order to perform the registration of the models. By substituting equations (4) in (3) we obtain the following equation :

$$\underbrace{\sum_{n\in\{1,2\}} \mathbf{H}_n \mathbf{A}_n^{-1} \mathbf{H}_n^T}_{\mathbf{W}} \boldsymbol{\lambda} = \underbrace{\sum_{n\in\{1,2\}} \mathbf{H}_n \Delta\mathbf{v}_n^{\text{free}}}_{\boldsymbol{\delta}^{\text{free}}} - \boldsymbol{\delta} \quad (5)$$

where $\boldsymbol{\delta}^{\text{free}}$ is directly evaluated using the *free motion* computed in equation (4). $\mathbf{W}$ is known as the *compliance matrix*, encoding the mechanical coupling between constraints through the mechanics of the models. Its formulation is equivalent to the computation of the Schür complement of the system. Although it could be computed on GPU [17], it remains the most expensive task of the simulation. Yet, the size of $\mathbf{W}$ is equal to the number of constraints, which is usually much smaller than the number of nodes.
3- Constraints solving: Equation (5) forms a *Non Linear Complementary Problem* (NLCP) where $\boldsymbol{\lambda}$ and $\boldsymbol{\delta}$ are unknown. Indeed, Coulomb friction is simulated along the shaft of the needle and unilateral constraints are used to simulate contacts based on Signorini's law. This equation is solved using a modified Gauss-Seidel algorithm allowing for the iterative computation of $\boldsymbol{\lambda}$ and $\boldsymbol{\delta}$ (see [16] for details).
4- Corrective motion: Once the constraint forces $\boldsymbol{\lambda}$ are known in the constraint space, they are re-projected in the motion space using the following equation:

$$\begin{cases} \Delta\mathbf{v}_1 = \Delta\mathbf{v}_1^{\text{free}} - \mathbf{A}_1^{-1}\mathbf{H}_1^T\boldsymbol{\lambda} \\ \Delta\mathbf{v}_2 = \Delta\mathbf{v}_2^{\text{free}} - \mathbf{A}_2^{-1}\mathbf{H}_2^T\boldsymbol{\lambda} \end{cases} \quad (6)$$

B. Robotic Control based on inverse simulations

In [15], the robotic needle steering is formulated as a minimization problem. The cost function is composed of several *objective functions* chosen to insert the needle along the predefined path. Due to the non-linearity of FE models and constraints, the above simulation is used to numerically derive the *Jacobian of the simulation* $\mathbf{J}$ linking the motion of the needle base to the objective functions minimization. Six simulation steps are performed where each DOF of the needle base are successively disturbed, allowing for the evaluation of the variations of objective functions in the numerical models.

To reach computation times compatible with robotic control, the method computes the Stiffness matrices, the Jacobian of the constraints and the Compliance matrix only once for all the inverse steps. Indeed, $\mathbf{H}_n$ and $\mathbf{A}_n$ are linearized around the model positions at the beginning of each simulation step. Therefore they do not depend on the needle's base variations. It allows to solve most of the operations of the inverse steps in the constraint space which is much smaller than the motion space (i.e. only solving

equations (5) and (6) corresponding to steps 3 and 4 of the above methods).

The method was validated performing the insertion of a flexible needle inside a deformable polyurethane foam [16]. A set of markers $\mathbf{m}^{(i)}$ located on the surface of the model were used to register the models and enforce their consistency with real structures (see Fig. 1). The references for the robot end effector position and orientation are derived solving the following equation:

$$\mathbf{T}^{(i+1)} = \mathcal{X}^{(i)} - \mathbf{J}^{+(i)}(\mathbf{k} \odot \mathbf{e}^{(i)}) \tag{7}$$

with $\mathcal{X}^{(i)}$ the current position (i.e. at time i) of the robot and $\mathbf{T}^{(i+1)}$ the next desired position. $\mathbf{e}^{(i)}$ are the values of the objective function $E(\mathbf{q}, \mathcal{X}, \mathbf{m})$ that depend on the positions of the FE meshes $\mathbf{q}$, the position of the robot $\mathcal{X}$ and the markers $\mathbf{m}$. The Hadamard product $\odot$ is used to scale the values of the objective function $\mathbf{e}^{(i)}$ with the gain $\mathbf{k}$.

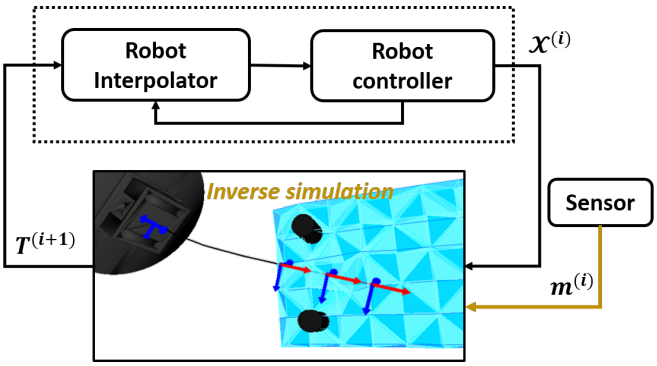


Fig. 1. Control loop using the inverse FE simulation of a needle insertion. The inverse problem is solved for each simulation step i providing the next desired position $\mathbf{T}^{(i+1)}$. The motion of the robot is then asynchronously interpolated by the low-level controller.

During the insertion, vertical and lateral deformations were generated leading to significant modifications of the undeformed trajectory and important bending of the needle. Despite these strong modifications, the method was able to maintain the tip of the needle within the 1 cm thick foam following the desired path with errors smaller than 1 mm. However, the method is not efficient to take into account respiratory motions in the same way.

III. METHODOLOGY

In this section we introduce the objective functions needed to steer the needle inside moving organs. Then, we propose a constraint-based formulation of the objective functions allowing to significantly reduce the computation time. And finally we describe our stabilization and regularization strategies needed for the control.

Following [16], errors of the model are corrected in real time thanks to a non-rigid registration. Although this step is necessary, the non-rigid registration of deformable bodies is not addressed here as it goes far beyond the scope of this paper. More importantly, contributions of the present article are generic and do not depend on the registration procedure as long as a set of points $\mathbf{m}$ are available to impose displacements and enforce the consistency of the models. In addition, to steer the needle outside the patient, the proposed solution relies on the tracking of the insertion point $\mathbf{p}$ located on the skin. Finally, the low-level control of the robot will not be addressed in this paper. Instead, our goal is to provide Cartesian positions $\mathbf{T}^{(i+1)}$ as fast as possible to the robotic system that will perform the interpolation of the robot motion assuming a constant velocity (see Fig. 2).

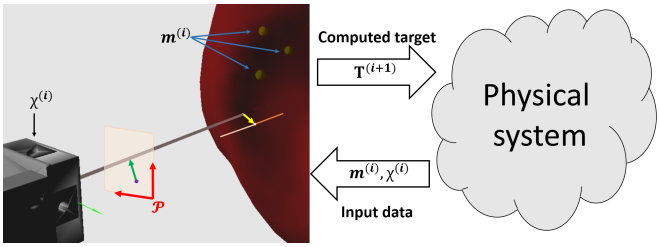


Fig. 2. Overview of communications and objective functions.

A. Objective functions

To steer the needle in the dynamic deformable environment, we propose to minimize the absolute value of the 3 following objective functions at each simulation step:

1) **The needle position along the predefined path:** This objective function is a three-dimensional vector linking the needle tip to a target point that moves along the path. Its expression is given by $\mathbf{e}_p = \mathbf{q}_{\text{target}} - \mathbf{q}_{\text{tip}}$ and is represented by the yellow vector in the figure 2.
2) **The needle orientation outside of the patient:** The second one is a one-dimensional vector. Its expression is given by $\mathbf{e}_a = \arccos(\mathbf{n}_{\text{tip}} \cdot \mathbf{n}_{\text{traj}})$, with $\mathbf{n}_{\text{tip}}$ the normalized direction at the needle tip and $\mathbf{n}_{\text{traj}}$ the tangent to the trajectory at the entry point of the skin.
3) **The motion of the entry point:** The last one is defined as the two-dimensional vector linking the current position of the penetration point $\mathbf{p}_c$ and the position $\mathbf{p}_p$ stored at the moment of the puncture. Its expression is given by $\mathbf{e}_e = \text{proj}_{\mathcal{P}}(\mathbf{p}_c - \mathbf{p}_p)$ which corresponds to the distance between insertion points projected on the tangential plane $\mathcal{P}$ at the surface of the skin. This is represented by the green vector on the figure 2 and with the normal plane in red. The purpose of the function is to avoid tearing the skin during the insertion.

The objective functions are evaluated using the inverse simulation to compute the Jacobian of the simulation whose dimension depends on the position of the needle tip. Outside the patient the size is (4×6) whereas inside the size is (5×6).

The inverse simulation involves the mechanical modeling of the needle, the liver constrained by the aforementioned tracked points and the interaction between those two models. The desired path is defined inside the liver model, and follows its motion. This allows computing the values of all the considered objective functions described above.

B. Constraint Objective functions

All the physical interactions used in the inverse simulation are expressed through Lagrangian multipliers (needle/tissue

constraints, attach of the needle with the robotic arm and registration constraints), allowing this way to obtain the complete mechanical coupling between constraints in the condensed system matrix $\mathbf{W}$. This choice is necessary to compute the Compliance matrix once for all the inverse steps as proposed in [16]. Yet, the computation of the $\mathbf{J}$ requires the evaluation of the objective functions in the motion's space. In other words, the linear system of equation (6) must be solved for each variation of the needle's base in order to derive robotic commands in equation (7). Although this operation is usually not the bottleneck of direct simulations, it raises computation time issues when it's performed several times in a close robotic loop.

In order to get rid of the re-projection operation, we introduce the *constraint objective functions*. We propose to formulate the objective functions as virtual constraints and integrate them in the $\mathbf{H}$ matrix. The $\mathbf{H}$ matrix is therefore augmented with 4 or 5 lines depending on the position of the needle. The objective functions $\bar{\mathbf{e}}$ are projected in the constraint space according to the lines introduced in $\mathbf{H}$. These additional constraints do not have a physical meaning but instead they allow to compute the variation of the objective functions directly in the constraint's space according to the needle's base perturbations. For this purpose, the Compliance matrix is modified as follows:

$$\left[\begin{array}{ccc|c} w_{1,1} & \dots & w_{1,k} & \\ \vdots & \ddots & \vdots & \\ w_{k,1} & \dots & w_{k,k} & \mathbf{0}_{(k+p,p)} \\ \hline w_{k+1,1} & \dots & w_{k+1,k} & \\ \vdots & \ddots & \vdots & \\ w_{k+p,1} & \dots & w_{k+p,k} & \end{array}\right] \left[\begin{array}{c} \lambda_1 \\ \vdots \\ \lambda_k \\ \hline 0 \\ \vdots \\ 0 \end{array}\right] = \left[\begin{array}{c} \delta_1 \\ \vdots \\ \delta_k \\ \hline \bar{e}_1 \\ \vdots \\ \bar{e}_p \end{array}\right] \quad (8)$$

Lines 1 to k correspond to the original Compliance matrix used to impose constraints in the inverse simulations, whereas lines $k+1$ to $k+p$ correspond to the constraint objective functions. The mechanical influence of the constraint objective functions on all the other constraints is removed by setting columns $k+1$ to $k+p$ to $\mathbf{0}$. This enforces that mechanical constraints are not impacted by the constraint objective functions. Instead, the mechanical influence of all the other constraints is conserved (lines $k+1$ to $k+p$) allowing retrieving the value of the objective functions $\bar{\mathbf{e}}$ without any need for re-projection in motion's space.

It is important to note that the proposed solution does not modify the Jacobian $\mathbf{J}$ for linear objective functions (position-based, such as $\mathbf{e}_p$). However, since linearizations are performed at the construction of $\mathbf{H}$, our method does not provide the same values for nonlinear functions (such as angular functions as used in [16]). Although the evaluation of nonlinear functions may be possible with additional re-projections of only the concerned DOF, we used directly the linearized version which is sufficient to build the Jacobian. Indeed, the sign of objective functions are conserved and their values are later scaled with the gain $\mathbf{k}$.

The convergence criteria of the Gauss-Seidel is defined as the norm of the variation of the $\boldsymbol{\lambda}$ vector between two successive iterations. To enforce that the convergence is not modified by the constraint objective functions, the associated forces are set to 0 at each iteration of the Gauss-Seidel.

C. Stabilization

We propose to compute the centered Jacobian where each line j is obtained with:

$$\mathbf{J}^j = \frac{E(\mathbf{q}, \mathcal{X} + \Delta\mathcal{X}^j, \mathbf{m}) - E(\bar{\mathbf{q}}, \bar{\mathcal{X}} - \Delta\mathcal{X}^j, \mathbf{m})}{2||\Delta\mathcal{X}^j||} \quad (9)$$

with $\Delta\mathcal{X}^j$ the perturbation of the j^{th} DOF of the needle's base. With regards to [16], this formulation requires to compute 6 additional inverse simulation steps (12 steps) to construct the centered Jacobian. However due to the nature of the constraints (complementarity, friction), it allows to increase the spatial validity domain of the Jacobian.

In addition, the problem may become ill conditioned during the insertion. Therefore, we use a Tikhonov regularization while inverting the Jacobian. The pseudo-inverse $\mathbf{J}^+_\alpha$ of the Jacobian is computed using the following equation:

$$\mathbf{J}^+_\alpha = (\mathbf{J}^T\mathbf{J} + \alpha\mathbf{U}\Delta_{\sigma(\mathbf{J}^T\mathbf{J})<\alpha}\mathbf{V}^T)\mathbf{J}^T \quad (10)$$

where $\mathbf{UDV}^T$ is the singular value decomposition of $\mathbf{J}^T\mathbf{J}$, and $\Delta_{\sigma(\mathbf{J}^T\mathbf{J})<\alpha}$ a diagonal matrix with the diagonal element $\Delta_{k,k}$ defined by :

$$\begin{cases} \Delta_{k,k} = d_{k,k} & \text{if } d_{k,k} > \alpha \\ \Delta_{k,k} = \alpha & \text{otherwise} \end{cases} \quad (11)$$

With $d_{k,k}$ the elements of the matrix $\mathbf{D}$ and α the regularization parameter.

Finally, the pseudo-code of one step of the inverse simulation is shown in the algorithm 1. With regards to [16], this formulation requires to compute 6 additional inverse simulation steps (12 steps) to construct the centered Jacobian.

Algorithm 1: Optimized inverse simulation loop

1 **Get input data:** $\mathcal{X}^{(i)}, \mathbf{m}^{(i)}$;
2 **Free Motion:** $\Delta\mathbf{v}^{\text{free}} = \mathbf{A}^{-1}\mathbf{b}$;
3 **Constraint and Objective Definition:** $\mathbf{H}$;
4 **Compute Compliance:** $\mathbf{W} = \sum \mathbf{H}\mathbf{A}^{-1}\mathbf{H}^T$;
5 **Compute error:** $\bar{\mathbf{e}}_0 = E(\mathbf{q}, \mathcal{X}, \mathbf{m})$;
6 **for** *j= 0 to 6* **do**
7 **for** *k = {-1,1}* **do**
8 **Compute Violation:** $\boldsymbol{\delta}^{j,k}$;
9 **Compute objective:** $\bar{\mathbf{e}}^{j,k} = E(\mathbf{q}, \mathcal{X} + k\Delta\mathcal{X}^j, \mathbf{m})$;
10 **Solve Constraints:** $\mathbf{W}\begin{bmatrix} \boldsymbol{\lambda}^{j,k} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\delta}^{j,k} \\ \bar{\mathbf{e}}^{j,k} \end{bmatrix}$;
11 **Compute Jacobian:** $\mathbf{J}^j = \frac{\bar{\mathbf{e}}^{j,1} - \bar{\mathbf{e}}^{j,-1}}{2\|\Delta\mathcal{X}^j\|}$;
12 **Solve the inverse problem:** $\Delta\mathcal{X} = -\mathbf{J}^+_\alpha \cdot \bar{\mathbf{e}}_0$;
13 **Send the target:** $\mathbf{T}^{(i+1)} = \mathcal{X} + \Delta\mathcal{X}$;
14 **Compute Violation:** $\boldsymbol{\delta}$;
15 **Solve Constraints:** $\mathbf{W}\begin{bmatrix} \boldsymbol{\lambda} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\delta} \\ \mathbf{0} \end{bmatrix}$
16 **Corrective Motion:** $\mathbf{q} = \Delta\mathbf{v}^{\text{free}} - \mathbf{A}^{-1}\mathbf{H}^T\boldsymbol{\lambda}$;

Line 1 consists in the acquisition of the data from the environment. Line $2-4$ concerns the computation of the mechanical matrices (performed only once for each simulation

step). Line $6-7$ stands for the computation of the centered Jacobian $\mathbf{J}$. Line $8-11$ provide successively the line j of the Jacobian. Line $12-13$ solve the inverse problem and send the next position to the robotic system. Line $14-16$ compute the positions of the FE models for the next simulation steps.

IV. EXPERIMENTAL FRAMEWORK

The method is evaluated using a realistic simulation (called *direct simulation*) of a pig during the respiratory motion that has been generated from in-vivo data. A set of 5 metallic fiducials has been inserted percutaneously inside the pig liver. Two CT scans have been acquired providing the 3D positions of the markers in the extreme positions (inspiration and expiration). In addition, the 2D displacements of markers have been recorded and manually segmented in fluoroscopic images during the breathing cycles. These data have been extrapolated in 3D solving the 2D/3D registration problem. Although the acquisition of these data may raise technical issues in a medical context, it is used as a realistic data set allowing for the evaluation of the method independently on the registration procedure/modality.

The direct simulation involves a co-rotational model of the liver composed of 2660 nodes (12328 tetrahedra), that was generated from the segmentation of the CT scans. The model includes the desired trajectory (generated manually by the interventional radiologist) and the markers. The young modulus $E = 5.5$ kPa and the Poisson ratio $\nu = 0.45$ are chosen according to the literature [20]. Interaction forces with the diaphragm are simulated. A minimization problem is solved (offline) to provide displacements of the diaphragm minimizing the distance between markers of the FE model and the previous data set. Collisions between the skin and the liver are also taken into account using the GPU-based method introduced in [21]. The skin is composed of 144 points and 450 tetrahedra and parameterized with $E = 10$ kPa and $\nu = 0.3$. The model of the needle relies on the Timoshenko formulation of the beam theory. It is composed of 13 edges, and parameterized with a Young's modulus $E = 200$ GPa, and $\nu = 0.3$ with a radius of $0,723$ mm. Needle interactions are simulated allowing for the penetration of the needle in both the liver and the skin.

The direct simulation has been optimized in order to enforce a fixed frame rate of $40Hz$ for any insertion path. The time step of the simulation has been chosen to 0.025 ms to enforce a real-time simulation of the respiratory motion. Input robotic commands (here considered as the displacement of the needle's base) are provided by a second simulation called *inverse simulation* that runs asynchronously on a separate computer. The inverse simulation provides the next position $\mathbf{T}^{(i+1)}$ at various frequencies, while the direct simulation performs a Cartesian interpolation of the motion at fixed velocity of 10 mm/s. Once the needle has reached the insertion point in the liver (i.e., after the penetration of the skin), the desired trajectory point $\mathbf{q}_{\text{target}}$ is advanced along the trajectory at fixed velocity. The trajectory is 7 cm long; it takes 1 minute to move the target position from the first to the last point of the trajectory.

The inverse simulation receives positions of markers $\mathbf{m}$ and the current position of the robot from the direct simulation. This simulation involves the FE model of the liver and the needle. Since the needle is not patient specific, the same mechanical parameters are used in both the direct and inverse simulations because it can be estimated offline as done in [16]. However, the young modulus of the liver is randomly chosen within the range of 13% variability of hepatic tissues (see [20] for justification). In addition, due to the asynchronous nature of our work, all the following results are averaged over 22 simulations.

In terms of registration accuracy, although the inverse simulation is only driven by 5 markers (without taking into account the complex surrounding interactions modeled in the direct simulation), the mean Hausdorff distance between the surface of the liver in the direct and inverse simulations is on average 0.64 mm $\pm$ 0.19 when the needle is outside and 2.55 mm $\pm$ 1.55 when the needle is inserted.

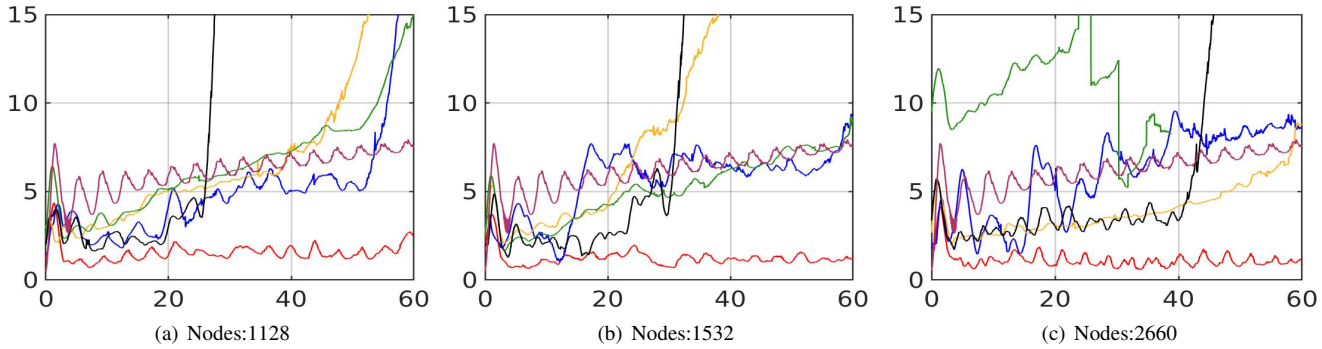
V. RESULTS

In the following paragraphs, we compare the accuracy and performances of 4 control strategies. **Static** denotes a method where both the trajectory and the needle are assumed rigid in the inverse simulation. **Full** stands for the complete resolution of the FE resolution for each inverse step. **Ada.** is the method introduced in [16] (re-projection in motion's space) with the objective functions introduced in the section III-A. **Us** is the method introduced in the present article. The suffix **-opt** means that the stabilization strategies introduced in the section III-C are used to build the Jacobian $\mathbf{J}$. For all the simulations, the gain $\mathbf{k}$ is chosen to favor the objective function of the entry point with a factor $2\times$ with respect to the other objective functions.

Figure 3 summarizes the performances obtained with an Intel core i7-6700 CPU running at 3.40GHz and a Nvidia GTX 980. In terms of computation time, **Ada.** already provide a speedup of $3.5\times$ compared to **Full**. Although the method provides the same range of computation time for small meshes, **Us** brings an additional speedup of more than $2\times$ for large meshes. More importantly, the overhead of inverse steps is independent from the mesh resolution (representing this way a smaller percentage of the computation time when the size of the mesh increase) whereas the cost of inverse steps with **Ada.** increase more than 400% between the smaller and larger mesh.

	Simulation Steps (%)						Computation Time (ms)		
	MM		IL		FE				
Mesh Size	Ada. -opt	Us -opt	Ada. -opt.	Us -opt	Ada. -opt.	Us -opt	Full -opt	Ada. -opt	Us -opt
1128	35.4	59.4	55	13.6	9.6	27	75.7	19.3	12.2
1532	31.2	63.5	59.9	7.4	8.9	29.1	166.6	47.3	24.5
2660	29.9	65.5	61.6	6	8.5	28.5	251	71.1	34.2

Fig. 3. Computation time of different parts of the algorithm 1 using various mesh resolution. MM: Mechanichal matrices definition (Lines 1-4), IL: Inverse loop (Lines 7-12), FE: FE resolution (Lines 15-17)



(a) Nodes:1128 (b) Nodes:1532 (c) Nodes:2660

Fig. 4. Distance in mm (in the direct simulation) between the needle tip and the desired position on the trajectory (corresponding to the norm of the objective function $\| \bar{\mathbf{e}}_p \|$) according to the insertion time (s), averaged over 22 simulations. The following methods with the associated colors are compared for various resolutions of the mesh liver used in the inverse simulation: **Static**: purple, **Full**: black **Ada.**: green, **Us**: yellow, **Ada.-opt**: blue, **Us-opt**: red.

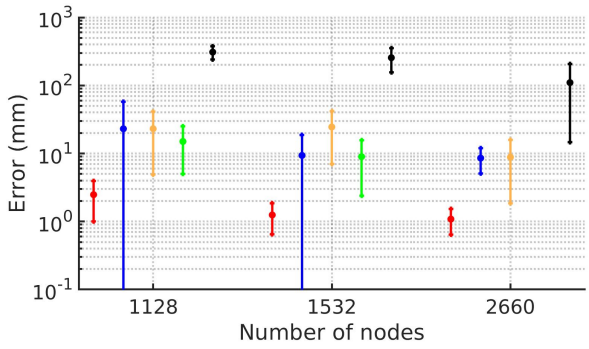


Fig. 5. Mean error and standard deviation between the needle tip and the last point of the trajectory at the end insertion for various mesh resolution.

The accuracy of the method is now evaluated and compared. The figure 4 shows the distance between the needle tip and the desired point on the trajectory during the insertion. The figure 5 shows the distance between the needle tip and the last point of the trajectory at the end of the insertion. The **static** method does not depend on the mesh resolution but for the ease of reading, it is displayed with the other methods in figure 4. **static** cannot account for deformations and quickly diverge to a final error of 7 mm. The oscillations are the consequence of the breathing motion that periodically moves closer or further away the target on the trajectory.

Although the method **Full-opt** relies on complex biomechanical models to steer the needle, the significant delay introduced by solving the complete FE simulation for each inverse step prevents from anticipating deformations of the respiratory motion. Indeed, less than 3 FPS are obtained for the mesh composed of 2660 nodes, resulting in the divergence of the method for all the tested scenarios. The main reason is the fact that interaction constraints between the needle and the liver are created at different locations in the direct and inverse simulations (because of the delay). The resulting motion estimated by the Jacobian in the inverse simulation is therefore significantly different from the motion in the direct simulation which results in over compensations, significant positioning errors and numerical instabilities due to the large deformations generated in the inverse simulation.

The optimizations **-opt** significantly improve the stability and the accuracy of the system. Indeed, without optimizations both methods **Ada.** and **Us** provide either large positioning errors or are subject to instabilities preventing to perform the complete insertion. However, optimizations allow the methods to converge, except for **Ada.-opt** with the smaller mesh which tends to create over-constrained problem preventing to generate the necessary deformations to reduce errors during the insertion. Maintaining 20 FPS in the inverse simulation, **Us-opt** provides an average error of 1.08 mm $\pm$ 0.44 with the finer mesh, resulting in reproducible results (despite the randomized Young modulus and the asynchronous computations) that fulfill the recommendation for RFA procedures (i.e., lower than 3 mm).

Finally, the chosen maximal velocity achieved by the robot in the direct simulation does not influence the results since the robotic trajectories of the end effector is generated to follow the respiratory motion at the same speed. Indeed, another important benefit of the proposed solution lies in the fact that the average *Von Mises Stress* measured in the direct simulation with **Us-opt** is 2.32 kPa, which corresponds to a reduction of 91.7% compared to **Ada.-opt**, allowing this way to automatically compensate for the deformations induced by the breathing motion.

VI. CONCLUSION

A numerical method has been introduced allowing for the computation of complex FE simulations at sufficient frame rate to steer a needle in deformable and moving tissues. The main contribution is the introduction of the constraint objective functions used to build the Jacobian of the simulation in the constraint space, reducing this way the computation time while increasing the accuracy and stability of the system. The method was evaluated in a realistic framework generated from in vivo data. The proposed solution fulfills recommendations for RFA surgery. Future works will aim to apply the method to a robotic system. This might help to better anticipate external motions, and reduce the stress generated by the needle during the insertion.

Acknowledgement: This work was supported by French National Research Agency (ANR) within the project SPERRY ANR-18-CE33-0007 and the Investissements dAvenir program (ANR-11-LABX-0004, Labex CAMI).

REFERENCES

[1] S. McDermott and D. A. Gervais, "Radiofrequency ablation of liver tumors." *Seminars in interventional radiology*, vol. 30, no. 1, pp. 49–55, mar 2013.

[2] T. L. De Jong, N. J. van de Berg, L. Tas, A. Moelker, J. Dankelman, and J. J. van den Dobbelsteen, "Needle placement errors: Do we need steerable needles in interventional radiology?" *Medical Devices: Evidence and Research*, vol. 11, pp. 259–265, 2018.

[3] R. Poon, K. Ng, C. Lam, V. Ai, J. Yuen, S. Fan, and J. Wong, "Learning curve for radiofrequency ablation of liver tumors: prospective analysis of initial 100 patients in a tertiary institution," *Annals of surgery*, vol. 239, pp. 441–9, 05 2004.

[4] D. R. Kaye, D. Stoianovici, and M. Han, "Robotic ultrasound and needle guidance for prostate cancer management: review of the contemporary literature." *Current opinion in urology*, vol. 24, no. 1, pp. 75–80, 2014.

[5] P. Li, Z. Yang, and S. Jiang, "Needle-tissue interactive mechanism and steering control in image-guided robot-assisted minimally invasive surgery: a review," *Medical and Biological Engineering and Computing*, vol. 56, no. 6, pp. 931–949, 2018.

[6] O. Piccin, L. Barbé, B. Bayle, M. De Mathelin, and A. Gangi, "A Force Feedback Teleoperated Needle Insertion Device for Percutaneous Procedures," *The International Journal of Robotics Research*, vol. 28, no. 9, pp. 1154–1168, 2009.

[7] D. C. Rucker, J. Das, H. B. Gilbert, P. J. Swaney, M. I. Miga, N. Sarkar, and R. J. Webster, "Sliding mode control of steerable needles," *IEEE Transactions on Robotics*, vol. 29, no. 5, pp. 1289–1299, 2013.

[8] A. Krupa, "A new duty-cycling approach for 3D needle steering allowing the use of the classical visual servoing framework for targeting tasks," *International Conference on Biomedical Robotics and Biomechatronics (EMBS)*, pp. 301–307, 2014.

[9] R. Secoli and F. Rodriguez y Baena, "Adaptive path-following control for bio-inspired steerable needles," in *IEEE BioRob*. IEEE, jun 2016.

[10] V. G. Mallapragada, N. Sarkar, and T. K. Podder, "Robot-assisted real-time tumor manipulation for breast biopsy," *IEEE Transactions on Robotics*, vol. 25, no. 2, pp. 316–324, 2009.

[11] I. Elgezua, Y. Kobayashi, and M. G. Fujie, "Survey on current state-of-the-art in needle insertion robots: Open challenges for application in real surgery," *Procedia CIRP*, vol. 5, pp. 94–99, 2013.

[12] P. Kulkarni, S. Sikander, P. Biswas, S. Frawley, and S.-E. Song, "Review of Robotic Needle Guide Systems for Percutaneous Intervention," *Annals of Biomedical Engineering*, 2019.

[13] G. Widmann, P. Schullian, M. Haidu, F. Wiedermann, and R. Bale, "Respiratory motion control for stereotactic and robotic liver interventions," *The international journal of medical robotics + computer assisted surgery : MRCAS*, vol. 6, pp. 343–9, 09 2010.

[14] E. Coevoet, A. Escande, and C. Duriez, "Soft robots locomotion and manipulation control using FEM simulation and quadratic programming," *RoboSoft 2019 - 2019 IEEE International Conference on Soft Robotics*, pp. 739–745, 2019.

[15] Y. Adagolodjo, L. Goffin, M. De Mathelin, and H. Courtecuisse, "Inverse real-time Finite Element simulation for robotic control of flexible needle insertion in deformable tissues," *IROS*, pp. 1–6, 2016.

[16] Y. Adagolodjo, L. Goffin, M. D. Mathelin, H. Courtecuisse, M. De Mathelin, and H. Courtecuisse, "Robotic insertion of flexible needle in deformable structures using inverse Finite Element simulation," *IEEE Transactions on Robotics*, pp. 1–12, 2019.

[17] H. Courtecuisse, J. Allard, P. Kerfriden, S. P. A. Bordas, S. Cotin, and C. Duriez, "Real-time simulation of contact and cutting of heterogeneous soft-tissues," *Medical Image Analysis*, vol. 18, no. 2, pp. 394–410, 2014.

[18] C. Duriez, C. Guébert, M. Marchal, S. Cotin, and L. Grisoni, "Interactive simulation of flexible needle insertions based on constraint models," *Lecture Notes in Computer Science*, pp. 291–299, 2009.

[19] J. Allard, H. Courtecuisse, and F. Faure, "Implicit FEM Solver on GPU for Interactive Deformation Simulation," in *GPU Computing Gems Jade Edition*. Elsevier, 2011, pp. 281–294.

[20] J. M. Hudson, L. Milot, C. Parry, R. Williams, and P. N. Burns, "Inter- and Intra-Operator Reliability and Repeatability of Shear Wave Elastography in the Liver: A Study in Healthy Volunteers," *Ultrasound in Medicine and Biology*, vol. 39, no. 6, pp. 950–955, 2013.

[21] J. Allard, F. Faure, H. Courtecuisse, F. Falipou, C. Duriez, and P. G. Kry, "Volume Contact Constraints at Arbitrary Resolution," *ACM Transactions on Graphics*, vol. C, no. 3, pp. 1–10, jul 2010.