Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

Just-in-Time Modelling of Real-Time Tissue Deformation in Ultrasound-Guided Percutaneous Surgery

Kian Zalzalah | **Carlos Rossa**

[1]Department of Systems and Computer Engineering, Carleton University, 1125 Colonel By Drive, Ottawa, K1S 5B6, Ontario, Canada

Correspondence
Corresponding author Kian Zalzalah, Department of Systems and Computer Engineering, Carleton University, 1125 Colonel By Drive, Ottawa, Ontario, K1S 5B6, Canada.
Email: kianzalzalah@cmail.carleton.ca

Abstract

Simulators for training ultrasound-guided percutaneous surgery (usPS) demand rapid updates of tissue and image deformation, yet high-fidelity physics models exceed real-time budgets on standard hardware. This paper presents a Just-in-Time (JIT) modelling framework that predicts ultrasound image deformation from needle insertions in real-time. First, an offline historical database of needle locations and resulting image deformations is created using a simulation engine. For each new query, the model selects a local neighbourhood and computes weights that minimize a mean-square error. The resulting weighted prediction of the full nodal deformation is reshaped for direct simulation or visualization. The framework is evaluated against mesh deformations calculated using thin plate spline (TPS), five mesh resolutions, and multiple database sizes. On the finest mesh, the mean relative error is 0.2% while the median per-query runtime is 1.1 ms, providing a speedup of 31,900 times over the simulation engine used to generate the offline database. These results indicate that JIT closely reproduces the TPS reference deformation while satisfying stringent real-time requirements. The proposed framework is suitable for real-time, embedded, and portable usPS training simulators.

KEYWORDS
just-in-time modelling, tissue deformation, ultrasound-guided needle insertion, surrogate modelling, surgical simulation, real-time prediction

PRACTITIONER POINTS

- A Just-in-Time modelling framework is proposed for real-time prediction of tissue deformation during ultrasound-guided percutaneous surgery.
- The method interpolates an offline database of physics-based simulations to achieve millisecond-scale prediction while maintaining close agreement with the reference deformation engine.
- The framework is well suited to portable and simulation-based training systems that require fast and realistic deformation updates.

1 | INTRODUCTION

Ultrasound-guided percutaneous surgery (usPS) uses thin instruments to reach targets in the tissue through small punctures in the skin. Procedures such as biopsy, brachytherapy, nephrolithotomy, and tumour ablation commonly rely on ultrasound as the primary guiding imaging modality. Compared to CT and MRI, ultrasound is portable, radiation-free, widely accessible, and safer for vulnerable patients as is the case with pediatrics[1]. Despite its benefits over other types of interventions, usPS has a steep learning curve and requires specialized training to achieve surgical proficiency. This is often attributed to the high level of coordination and dexterity needed to simultaneously image the tool while steering it towards a target. In nephrolithotomy, for example, the learning curve only plateaus after 60 interventions and surgical excellence requires 115 trials[2].

During the intervention, the clinical practitioner visualizes the tool with longitudinal or transverse ultrasound images. In longitudinal imaging, they insert the tool so that its shaft travels parallel to the imaging plane, appearing as a line in the 2D image. In transverse imaging, the tool shaft crosses the imaging plane orthogonally, appearing as a bright spot in the image. To visualize and guide the tool tip toward the target, the interventionist must continuously sweep the ultrasound probe back and forth. Coordinating one hand holding the probe and one hand holding the tool is challenging. In addition, the tool deforms the tissue and pushes the target away from its original location, while also shifting anatomical structures in the ultrasound images[3]. Accurate spatial coordination between the tool and real-time ultrasound feedback is critical for success, yet the procedure's steep learning curve limits widespread

 © Copyright Holder Name

adoption and increases the risk of complications. Other challenges, including limitations in image quality and poorly resolved targets, add to the learning curve thereby making the procedure outcomes highly operator-dependent[4].

To overcome this learning curve, usPS can strongly benefit from simulation training. Nowadays physical, virtual, or cyber-physical simulators are a fundamental component of surgical training programs, allowing interventionists to practice and hone their skills in a risk-free environment[5,6] emphasized the growing role of simulation and feedback frameworks in interventional radiology and image-guided surgery, where haptic and visual feedback improve targeting precision and reduce procedural errors. These developments highlight the importance of real-time, physics-aware models for safe and efficient needle guidance. Real-time tissue deformation is also relevant in robotic ultrasound image acquisition, where probe-tissue interaction must be regulated during scanning. Recent work has combined smooth robotic scanning-path planning with dynamic contact force regulation that accounts for tissue deformation and ultrasound-image feedback[7,3]. In usPS, tissue and image deformation models can teach the interventionist how to minimize tissue damage, interpret ultrasound images, and ensure accurate tissue targeting[8,9]. State-of-the-art simulators include dry labs, virtual reality (VR), cadaver-based wet labs, and animal tissue training[10]. Robotic-assisted surgical systems, such as the widely adopted da Vinci Surgical System, primarily use dry lab simulations and built-in scoring systems to evaluate trainee performance[11]. Despite their effectiveness in procedural training and skill assessment, current robotic simulators frequently lack comprehensive modules addressing tissue handling under image guidance while also incorporating realistic force feedback[11,12].

Virtual or cyber-physical simulators are more versatile and allow the interventionists to practice a wider range of procedures[13,14]. However, such simulators often require accurate modelling of the tool (like a needle) and tissue interaction to update the resulting ultrasound images in real-time[15]. Traditionally, these models involve detailed biomechanical and dynamic models of tool-tissue interactions that often exceed real-time computational constraints on standard hardware, thereby limiting their usability in portable or remote medical training scenarios[16,17].

The biomechanical response of soft tissue to external forces, such as those caused by needle insertion and tissue cutting, has primarily been simulated using Finite Element Models (FEM). This method relies on solving complex partial differential equations describing tissue deformation. FEM has been extensively used to simulate needle insertion, including for needle guidance and tissue trauma assessment in the liver[18,19], for needle steering in brachytherapy[20,15], and general 2D and 3D image deformation[21]. While accurate, FEM models are computationally expensive and large models often require specialized hardware to run in real-time[22,23,24]. Furthermore, FEM models often struggle with non-linear tissue response, particularly with large deformations commonly seen in needle-based procedures[20]. Despite advancements in computer power and optimized solvers, the use of real-time standard FEM simulation in a clinical setting and surgical simulators remains limited[25,26].

As a result, researchers have developed new methods that use FEM as a baseline combined with other modalities to approach real-time usage for surgical simulators.[22] reformulated deformation by using a Kalman filter that was built on a linear-elastic Finite Element (FE) model and constrained so that estimated reaction forces match the applied mechanical load at each step.[27] built a 3D FEM model for bevel-tip insertion in gel phantoms that matched measured needle forces and deflections and resolved time-resolved shear-stress fields around the tip—evidence of great physical detail for planning and design. FEM has also been coupled with simplified one-dimensional inextensible elastic rods models representing the needle mechanics[28]. For all of these novel methods, the trade-off for high fidelity is the need for computational power which, is difficult with the strict per-frame budgets of portable ultrasound trainers. Aside from FEM, several modifications and alternative physics-based methods have emerged for simulating needle-tissue interactions and the resulting images. An example uses a meshless total Lagrangian explicit dynamics (MTLED) framework that discretizes the tissues into a cloud of points rather than traditional FE, enabling a rapid, mesh-free setup[29,30] integrated distance, volume, shape matching, and a neo-Hookean constitutive constraints within a 160 Hz physics engine to simulate real-time liver deformations on GPU hardware. However, the dependence on GPU acceleration and specialized setups still poses challenges in resource-limited portable trainers.

In our previous work[31], we showed that Thin Plate Spline (TPS) is an accurate and faster alternative to FEM for simulating ultrasound image deformation during usPS. TPS models needle insertion, tissue cutting, and friction as localized mesh displacements that propagate through a mesh using a non-rigid transformation. We showed that TPS can be 51.8% faster than an equivalent FEM model while achieving equivalent simulation accuracy. However, TPS still does not satisfy the real-time computational requirements for large mesh models. Real-time ultrasound-guided needle simulation has also been investigated using alternative rendering and interaction strategies. F. Vidal et al. developed a patient-specific simulation environment combining virtual ultrasound imaging with three-dimensional anatomical data[32]. Barnouin et al. more recently proposed real-time ultrasound rendering coupled with model-based tissue deformation during needle insertion, with deformation computed using GPU-based displacement functions[33].

An alternative to the discussed methods above is to construct a real-time data-driven model that interpolates simulation results from high-fidelity simulations performed offline. Pre-computed data can significantly reduce computational demands and allow for rapid approximations of tissue deformations. In this paper, we propose to use Just-in-Time (JIT) modelling to interpolate offline created data and estimate tissue deformation for configurations that are not captured in the database. First, a given needle-tissue interaction model is selected and used to create a database containing a variety of different needle insertion locations in the tissue and the resulting tissue deformation (modelled as a mesh). During real-time simulation, the current position

of the needle in the mesh (query) is an input to the JIT model. The model then compares the query with every other entry in the offline database. From this, it computes weights for every entry based on the distance between the position of the needle in the query and in the entry. It then applies a quadratic Taylor surrogate via Hessian-based local linear regression to weigh the meshes stored in the database accordingly, and estimate the final mesh deformation that matches the query. Since the model is reduced to a simple regression, it is computationally efficient with high realism and suitable for real-time, embedded, and portable training systems.

The proposed JIT framework is not specific to TPS and can be applied to any deformation engine capable of generating consistent input-output pairs of needle configurations and corresponding deformation fields. In the present study, TPS is selected as the reference deformation engine and evaluated against both an equivalent FEM model and ex vivo porcine tissue deformation measured from ultrasound images using speckle tracking[31]. The paper is structured as follows: Section 2.1 introduces the JIT model applied to the needle-tissue interaction problem, Section 2.2 details how a variable-size database is created. Our evaluation criteria are defined in 2.3 followed by the results reported in Section 3 across five mesh sizes and five different database sizes; showing that JIT can accurately predict the mesh deformation with an error that does not exceed 0.2% while achieving speedups ranging from 1,140 times to 31,900 times relative to the needle-tissue interaction model. Section 4 discusses the acquired results and Section 5 summarizes the study and outlines directions for future work. To the best of our knowledge, this is the first implementation of JIT to simulate tissue displacement in usPS. This proposed framework provides a foundation to develop computationally efficient models that can be embedded in portable simulators.

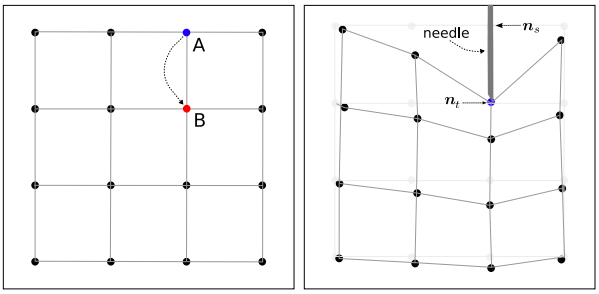


FIGURE 1 Illustration of mesh deformation during needle insertion. Left: initial undeformed mesh with mesh points *A* (entry, blue) and *B* (target, red). Right: Deformed mesh after a needle is inserted in *A* and pushed point *A* to the initial position of *B*. TPS is used to calculate the position of all other points in the mesh.

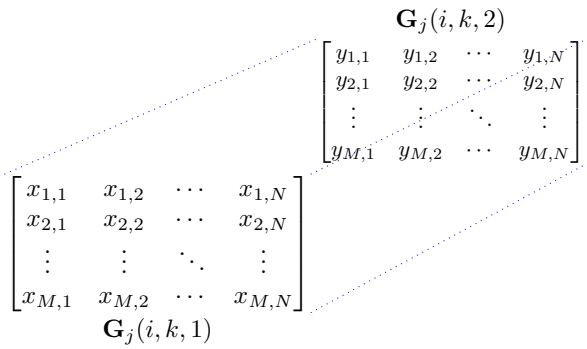


FIGURE 2 Illustration of the tensor layout: $\mathbf{G}_j(:,:,1)$ holds *x*-coordinates, $\mathbf{G}_j(:,:,2)$ holds *y*-coordinates.

2 | MATERIALS AND METHODS

2.1 | Just-in-Time Model

The proposed JIT framework has two stages. First, a simulation engine generates examples of needle insertions and their resulting tissue mesh deformations to build an offline database. Then, for a new needle insertion (query), the JIT model compares it with similar cases in the database, weights and interpolates them to output the predicted mesh that corresponds to the current needle insertion in real-time.

2.1.1 | Step 1: Offline Database

Assume that the tissue is represented as a mesh $\mathbf{G}_j$ of $M \times N$ interconnected nodes as shown in Fig. 1. Further, assume that the needle is inserted in the mesh at a position described by matrix $\mathbf{X}_j$, where *j* is one entry in the database. A simulation engine determines the position of each node in the mesh $\mathbf{G}_j$ that results from needle insertion $\mathbf{X}_j$. Each point within the mesh corresponds to a pixel in an ultrasound image.

For each entry *j* in the database, the position of the needle is described by recording the location of the needle tip and the entry point in the mesh as:

$$\mathbf{X}_j = \begin{bmatrix} x_s & y_s \\ x_t & y_t \end{bmatrix} \in \mathbb{R}^{2\times 2}, \tag{1}$$

where (x_s, y_s) denotes the entry point of the needle in the mesh and (x_t, y_t) denotes the position of the needle tip in the mesh in the horizontal and vertical directions *x* and *y*, respectively.

The mesh that results from the j^{th} needle insertion $\mathbf{X}_j$ is described by the tensor $\mathbf{G}_j(i,k,p) \in \mathbb{R}^{M\times N\times 2}$, where the *i* and *k* are the column and row indexes of $\mathbf{G}_j$, with $1 \leq i \leq M$ and $1 \leq k \leq N$. Each node *i*, *k* has coordinates $x_{i,k}$ and $y_{i,k}$ along the *x* and *y* directions. The third dimension of the tensor, $p \in \{1, 2\}$, stores the position of every node in *x* ($p = 1$) and *y* ($p = 2$), i.e.,

As illustrated in Fig. 2, this format mirrors the spatial location of nodes in the mesh while providing direct access to absolute node positions in the *x* and *y* directions.

$$\mathbf{x}_j = \text{vec}(\mathbf{X}_j) = \begin{bmatrix} x_s & y_s & x_t & y_t \end{bmatrix}^T \in \mathbb{R}^{n\times 1}, \tag{2}$$

and

$$\begin{aligned}\mathbf{g}_j &= \text{vec}(\mathbf{G}_j) \in \mathbb{R}^{2MN\times 1} \\ &= \begin{bmatrix} x_{1,1} & y_{1,1} & x_{2,1} & y_{2,1} & \cdots & y_{M,1} & \cdots & y_{M,N} \end{bmatrix}^T\end{aligned} \tag{3}$$

For the present formulation, the query contains four coordinates, so $n = 4$.

Flattening is a necessary step in the JIT framework, as the algorithm performs vector-based regression and therefore requires the input–output pairs to lie in a common high-dimensional Euclidean space. This representation enables efficient distance computation, neighbour selection, and local model fitting using standard linear algebra operations. The transformation does not discard structural information, since the predicted deformation vector can be reshaped back into its original tensor form for simulation or visualization.

Using any needle–tissue interaction simulation engine, we compute offline a collection $\mathcal{S}$ of N_h pairs consisting of a needle configuration and the corresponding deformed mesh to construct the historical database:

$$\mathcal{S} = \left\{(\mathbf{x}_j, \mathbf{g}_j)\right\}_{j=1}^{N_\text{h}}. \tag{4}$$

Once the database is created, given a query $\mathbf{x}_0$, the JIT model returns a prediction $\hat{\mathbf{g}}(\mathbf{x}_0)$, which is then reshaped back into a tensor $\hat{\mathbf{G}}(\mathbf{x}_0)$ representing the final positions of the mesh nodes.

2.1.2 | Step 2: Neighbourhood Selection and Weighting

Given query $\mathbf{x}_0$, JIT compares it with the closest N_q samples in the database. It then assigns weights to every N_q samples and defines the predicted mesh as a weighted local mean of outputs:

$$\hat{\mathbf{g}}(\mathbf{x}_0) = \sum_{q=1}^{N_q} w_q\, \mathbf{g}_q, \tag{5}$$

where w_q are the weights assigned to every neighbour point, to be calculated later, and $1 \le q \le N_q$.

To find the weights w_q, we first quantify the similarity between the query and every entry in the database. To this end, we calculate the Euclidean distance between them as

$$\delta_q = \|\mathbf{x}_q - \mathbf{x}_0\|_2. \tag{6}$$

before sorting these distances from lowest to highest Euclidean distance, that is:

$$\omega_{N_q}(\mathbf{x}_0) = \left\{\delta_q \text{ is among the } N_q \text{ smallest}\right\}, \tag{7}$$

where $\omega_{N_q}(\mathbf{x}_0)$ contains the N_q closest samples in the dataset to $\mathbf{x}_0$. By varying N_q, the bias–variance trade-off can be controlled: A smaller N_q yields a more localized but potentially noisy estimate, while a larger N_q produces a smoother but potentially biased fit. We therefore select N_q by minimizing a local prediction criterion over a range $N_\text{min} \le N_q \le N_\text{max}$ where the maximum and minimum bounds are defined by the user. To determine the optimal neighbourhood size, we first take the N_q nearest inputs to $\mathbf{x}_0$ based on the scalar Euclidean distances $\delta_q = \|\mathbf{x}_q - \mathbf{x}_0\|_2$. For the local quadratic model, we use the vector offset $\boldsymbol{\delta}_q = \mathbf{x}_q - \mathbf{x}_0$:

$$y_q \approx \alpha + \mathbf{J}^T\boldsymbol{\delta}_q + \tfrac{1}{2}\boldsymbol{\delta}_q^T\mathbf{H}\,\boldsymbol{\delta}_q. \tag{8}$$

where the Jacobian $\mathbf{J}$ represents the first-order sensitivity of the local response to changes in the query coordinates, while the Hessian $\mathbf{H}$ captures the corresponding second-order curvature. In other words, $\mathbf{J}$ and $\mathbf{H}$ represent the first- and second-order derivatives of the local response with respect to the query coordinates, evaluated at $\mathbf{x}_0$. The scalar α represents the predicted value at $\mathbf{x}_0$ (intercept). The parameters of this local quadratic model are estimated for each candidate neighbourhood size N_q, which is subsequently evaluated using the prediction-error criterion defined below. Equation (8) is the simplest local model that can capture both slope and curvature; if the relationship is nearly linear, the fit will return $\hat{\mathbf{H}} \approx 0$. Estimating the intercept, gradient, and Hessian from the N_q nearest samples allows for curvature-aware weighting and provides a parameter-aware error score (FPE) to choose N_opt that balances bias and variance. If the query has n dimensions, $\mathbf{J}$ will have n unknowns, $\mathbf{H}$ will have $n(n+1)/2$ unknowns. Combined with α, there are d variables in (8):

$$d = 1 + n + \frac{n(n+1)}{2}. \tag{9}$$

which can be calculated through a least-squares regression if $N_q \ge d$. For $n = 4$, the local quadratic model contains $d = 15$ parameters.

To fit this regression, the feature vector ψ can be constructed for each neighbour as

$$\psi_q = \begin{bmatrix} 1 \\ \boldsymbol{\delta}_q \\ \text{vech}\left(\boldsymbol{\delta}_q\boldsymbol{\delta}_q^T - \frac{1}{2}\,\text{diag}(\boldsymbol{\delta}_q\boldsymbol{\delta}_q^T)\right) \end{bmatrix}. \in \mathbb{R}^d, \tag{10}$$

where vech is the half vectorization operator applied to δ_q. These feature vectors are then assembled into the design matrix $\Psi_q \in \mathbb{R}^{N_q\times d}$, and the recorded entries into the vector $\mathbf{y}_q \in \mathbb{R}^{Nq}$:

$$\boldsymbol{\Psi}_q = \begin{bmatrix}\psi_1 & \psi_2 & \cdots & \psi_{N_q}\end{bmatrix}^T \in \mathbb{R}^{N_q\times d}, \tag{11}$$

$$\mathbf{G}_q = \begin{bmatrix}\mathbf{g}_1 & \mathbf{g}_2 & \cdots & \mathbf{g}_{N_q}\end{bmatrix}^T \in \mathbb{R}^{N_q\times 2MN}. \tag{12}$$

The design matrix stacks the constant α, linear, and unique quadratic terms (from $\mathbf{J}$ and $\mathbf{H}$), turning the quadratic model into a linear regression. Solving the ordinary least-squares (OLS) problem yields the least–squares estimate

$$\boldsymbol{\Theta} = (\Psi^T\Psi)^{-1}\Psi^T\mathbf{G}_\mathbf{q} \in \mathbb{R}^{d\times 2MN}, \tag{13}$$

which is the same as:

$$\boldsymbol{\Theta} = \begin{bmatrix} \alpha \\ \mathbf{J} \\ \text{vech}(\mathbf{H}) \end{bmatrix} \in \mathbb{R}^{d\times 2MN}, \tag{14}$$

The residual sum of squares between the actual position of each node in mesh $\mathbf{y}_q$ and the corresponding estimation $\boldsymbol{\psi}_q^T \boldsymbol{\theta}$ is:

$$\epsilon(N_q) = \sum_{q=1}^{N_q} \left\| \mathbf{g}_q - \boldsymbol{\psi}_q^T \boldsymbol{\Theta} \right\|_2^2, \tag{15}$$

The optimal neighbourhood size is then selected using Akaike's Final Prediction Error (FPE). For each candidate N_q, FPE accounts for both the regression error and the number of model parameters relative to the neighbourhood size:

$$\mathrm{FPE}(N_q) = \frac{\|\mathbf{G}_q - \boldsymbol{\Psi}_q \boldsymbol{\Theta}\|_F^2}{\left(1 - \frac{d}{N_q}\right)\left(1 + \frac{d}{N_q}\right)}, \tag{16}$$

Finally, the optimal neighbourhood size is the one with the lowest *FPE* for $N_{min} \leq N_q \leq N_{max}$. This penalizes small neighbourhoods (high variance) and large ones (high bias), yielding a balanced N_{opt}. Since all meshes share uniform stiffness and identical boundary conditions, the optimal neighbourhood size N_{opt} is expected to vary little across different insertion configurations. To avoid re-computing N_{opt} for every query, we pre-compute a global constant offline during database construction. This is performed once per database construction, and the resulting N_{opt} is subsequently used for all queries during online prediction for the given database, eliminating the need for per-query optimization while preserving accuracy. Given the chosen N_{opt}, we define an effective Hessian $\mathbf{H}_{\text{eff}}$ by averaging across all output dimensions, i.e.,

$$\mathbf{H}_{\text{eff}} = \frac{1}{2MN} \sum_{j=1}^{2MN} \mathbf{H}_j. \tag{17}$$

This aggregation produces a single $n \times n$ matrix that captures the average curvature of the mapping between needle parameters and the full mesh deformation, which will be used in our online prediction.

The offline database construction and neighbourhood-selection procedure described above is summarized in Algorithm 1.

Algorithm 1 Offline Database and Neighbourhood Selection

```
Inputs: Mesh of size M × N
for j = 1 to N_h do
    X_j ← Define needle entry point and tip position (1)
    G_j ← Calculate resulting mesh using TPS
    Flatten X_j and G_j into x_j and g_j with (2) and (3)
    S = (x_j, g_j) ← store x_j and g_j in database S.
end for
Given anchor query x_o
δ_j ← Compute the Euclidean distance ∀q with (6).
Sort δ_j from lowest to highest
for N_q ∈ {N_min, ..., N_max} do
    Take N_q nearest neighbours {x_q, g_q}.
    Ψ ∈ R^{N_q × d} ← Form features and the design matrix (11).
    Θ ← Estimate local parameters by OLS using (13):
    ε(N_q) ← Compute residual sum of squares using (15):
    FPE(N_q) ← Score the global neighbourhood with (16):
end for
Select N_opt = arg min FPE(N_q)
The global N_opt nearest neighbours have the smallest δ_j
H_eff ← determine effective Hessian for use in online prediction using (17)
```

2.1.3 | Step 3: Online Prediction with Optimal Neighbourhood Size

Once the optimal neighbourhood size N_{opt} is calculated, we redefine the previous range of q to $1 \leq q \leq N_{opt}$. To compute the final weighted-average prediction, we select the nearest neighbours as:

$$\omega_{N_{opt}}(\mathbf{x}_0) = \{\delta_q \text{ is among the } N_{opt} \text{ smallest}\}, \tag{18}$$

then use the previously calculated Hessian $\mathbf{H}_{\text{eff}}$ to calculate the second order bias term for each neighbour $q \in \omega_{N_{opt}}$:

$$v_q = \tfrac{1}{2} (\mathbf{x}_q - \mathbf{x}_0)^T \mathbf{H}_{\text{eff}} (\mathbf{x}_q - \mathbf{x}_0). \tag{19}$$

These bias terms can now form the vector $\mathbf{v} = [v_1, v_2, \ldots, v_{N_{\text{opt}}}]^T$. Next, the matrices required for the constrained mean square error minimization are defined. The matrix

$$\mathbf{V} = \mathbf{v}\,\mathbf{v}^T \in \mathbb{R}^{N_{opt} \times N_{opt}}, \tag{20}$$

captures the curvature bias (through $\mathbf{v}\,\mathbf{v}^T$). The matrix

$$\mathbf{D} = \begin{bmatrix} (\mathbf{x}_1 - \mathbf{x}_0)^T \\ \vdots \\ (\mathbf{x}_{N_{\text{opt}}} - \mathbf{x}_0)^T \end{bmatrix} \in \mathbb{R}^{N_{\text{opt}} \times n}, \tag{21}$$

contains the differences between the query $\mathbf{x}_0$ and each neighbour $\mathbf{x}_q$. Also, let:

$$\mathbf{o_1} = (1, \ldots, 1)^T \in \mathbb{R}^{N_{opt}} \tag{22}$$

To obtain the optimal weights w_q under the unbiased constraints $\sum_q w_q = 1$ and $\sum_q w_q(\mathbf{x}_q - \mathbf{x}_0) = 0$, the Karush–Kuhn–Tucker system can now be formed to compute the weights w_q as:

$$\begin{bmatrix} 2\mathbf{V} & \mathbf{o_1} & \mathbf{D} \\ \mathbf{o_1^T} & 0 & 0 \\ \mathbf{D}^T & 0 & 0 \end{bmatrix} \begin{pmatrix} \mathbf{w} \\ \lambda_0 \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ \mathbf{0} \end{pmatrix}. \tag{23}$$

The Lagrange multipliers λ_0 and $\boldsymbol{\lambda}$ are used in the above to convert the least-squares error problem into a single linear system. The multipliers act as penalties that push the solution toward satisfying the constraints. The system is solved jointly for $(\mathbf{w}, \lambda_0, \boldsymbol{\lambda})$, but only the weights w_q are used to form the final prediction. Finally, the mesh corresponding to $\mathbf{x}_0$ can be predicted by the weighted average of the selected neighbour outputs as

$$\hat{\mathbf{g}}(\mathbf{x}_0) = \sum_{q=1}^{N_{opt}} w_q\, \mathbf{g}_q. \tag{24}$$

The workflow of the JIT algorithm is summarized visually in Fig. 3. The complete online JIT prediction procedure is summarized in Algorithm 2.

Algorithm 2 Online JIT Prediction with FPE-Selected Neighbourhood and KKT Weights

`Input: Query` $\mathbf{x}_0$`, database` $\mathcal{S}$`, optimal neighbourhood size` N_{opt}`, effective Hessian` $\mathbf{H}_{\text{eff}}$

`Output: Predicted mesh` $\hat{\mathbf{G}}(\mathbf{x}_0)$ `and weights` $\mathbf{w}$

$\{\mathbf{x}_q, \mathbf{g}_q\} \leftarrow$ `Select` N_{opt} `nearest neighbours of` $\mathbf{x}_0$

$v_q \leftarrow$ `use the Hessian` $\mathbf{H}_{\text{eff}}$ `to calculate the second order bias term for each neighbour (19)`

$\mathbf{V}, \mathbf{D}, \mathbf{o_1} \leftarrow$ `Construct using (20), (21), (22)`

$\mathbf{w} \leftarrow$ `Solve KKT system (23)`

$$\hat{\mathbf{g}}(\mathbf{x}_0) \leftarrow \sum_{q=1}^{N_{\text{opt}}} w_q \mathbf{g}_q \tag{24}$$

$\hat{\mathbf{G}}(\mathbf{x}_0) \leftarrow$ `Reshape` $\hat{\mathbf{g}}(\mathbf{x}_0)$ `into` $\mathbb{R}^{M\times N\times 2}$

2.2 | Historical Database Construction

The algorithm described earlier requires a database computed offline, using any simulation engine, containing N_h pairs of needle position $\mathbf{x}_j$ in a mesh and the resulting position of each mesh node $\mathbf{g}_j$. While any simulation engine can be used, we employ here a thin plate spline (TPS) algorithm previously presented in [31], shown to produce similar results as an equivalent finite element model. First, an $M \times N$ mesh with rectangular elements is created, and the needle is inserted from the top of the mesh along a straight line until it reaches a node at the bottom margin of the mesh. At the boundaries of the mesh, all nodes are clamped so that the tissue edges remain fixed for any deformation. This ensures realistic, localized displacement fields and prevents global drift of the mesh. Databases of different sizes are then created by varying the number of needle insertions in the mesh, contrasted and given by the following parameters:

- E needle insertion points along the top boundary of the mesh, evenly spaced by a distance r;
- U different insertion angles per entry point, limited to a range that ensures the needle reaches the bottom line of the mesh without grazing the corners;
- K sampling points between the entry point and the bottom of the mesh, producing $K+1$ entries in the database.

Let the undeformed mesh have nodes along its top boundary

$$\{(x_i, y_1), \quad \forall \quad i = 1, \ldots, M\}.$$

Entry points $\mathbf{n}_e$ are selected at regular intervals from the points above with stride r, yielding a set of entry points

$$\mathbf{E} = \{\, \mathbf{n}_e = (x_e, y_1)\,, \forall\, e = 1 + (e-1)r,\ e = 1, \ldots, E\,\}.$$

At each mesh entry point $\mathbf{n_e}$ in $\mathbf{E}$, we insert the needle along U insertion angles $\theta_u \in [\theta_{\min}, \theta_{\max}]$, i.e., along the direction of unit vector

$$\mathbf{d}(\theta_u) = \begin{bmatrix} \cos\theta_u \\ \sin\theta_u \end{bmatrix}, \tag{25}$$

until the needle tip reaches the bottom of the mesh. These limits correspond to the angles of two lines that connect the entry point to the left and right bottom corners of the mesh, with a small angular margin to avoid grazing. Let $\ell_{e,u}$ be the resulting distance from the entry point to the bottom of the mesh for entry point $\mathbf{n}_e$ and angle θ_u. Along this path, we record $K+1$ snapshots, each containing the needle tip position and the mesh deformation, with corresponding tip locations

$$\mathbf{n}_{t_{e,u,m}} \;=\; \mathbf{n}_e + (s_m)\,(\ell_{e,u})\mathbf{d}(\theta_u). \tag{26}$$

with

$$s_m = \frac{m}{K}, \qquad m = 0, \ldots, K, \tag{27}$$

At each snapshot, the TPS engine computes the deformed mesh $\mathbf{G}$ given the entry $\mathbf{n}_s$ and tip position $\mathbf{n}_{t_{e,u,m}}$. Since each of the E entry points contributes U angles, and each angle produces $K+1$ snapshots, the total number of database entries is

$$N_h = E \cdot U \cdot (K+1). \tag{28}$$

The complete database is then given by (4), which serves as the offline dataset for the JIT model.

2.3 | Evaluation Criteria

To validate the proposed JIT framework, a testing dataset is generated separately from the historical database $\mathcal{S}$. The testing dataset has a size

$$N_t \approx \sigma N_{\text{h}}, \tag{29}$$

where $\sigma = 0.1$ and N_t is rounded to the nearest whole number. The test configurations are generated independently and are not contained in $\mathcal{S}$. From each sample s in the testing dataset, with $1 \le s \le N_t$, the needle entry point and tip position in the mesh $\mathbf{x}_0$ is input into the JIT model, which predicts the mesh deformation $\hat{g}_j^s$ of each of the $2MN$ mesh nodes j. The predicted position of each node is then compared against the TPS reference mesh displacement g_j^s in the testing dataset. The mean and maximum errors, ϵ_{me} and $\epsilon_{\max}$ respectively, are defined as:

$$\epsilon_{\text{me}} = \frac{1}{N_t}\sum_{s=1}^{N_t}\frac{1}{2MN}\sum_{j=1}^{2MN}\left|\hat{g}_j^s - g_j^s\right|, \tag{30}$$

$$\epsilon_{\max} = \frac{1}{N_t}\sum_{s=1}^{N_t}\max\left|\hat{g}_j^s - g_j^s\right|, \forall j = 1, \ldots, 2MN. \tag{31}$$

Additionally, we compute the 95^{th} percentile error ϵ_{95}, that is, the threshold under which 95% of the mesh deformation error falls. The smaller the error, the closer the JIT-predicted deformation matches the

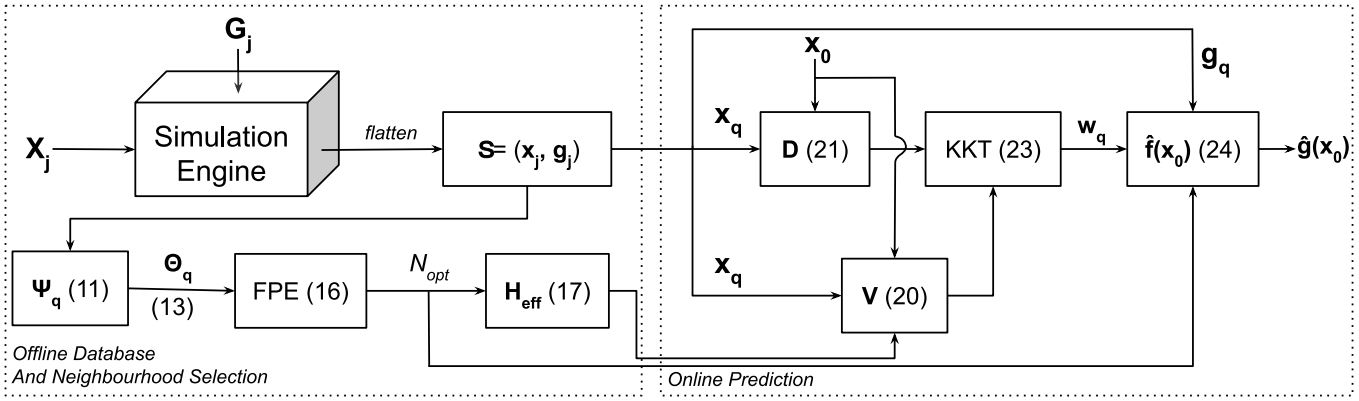


FIGURE 3 Representation of the offline database construction (TPS-based simulation to build $\mathcal{S}$) and online JIT prediction.

TPS reference deformation. The model can then be evaluated using different database sizes N_h, downsampling factor σ, and mesh resolutions $(M \times N)$.

3 | RESULTS

To validate the proposed framework, we conducted two sets of simulations. In the first set, we tested different mesh sizes with constant database sizes. In the second set, we tested different database sizes with a constant mesh size. For all simulations, the height and width of the mesh is kept constant at 1 unit; hence, the more mesh nodes in the mesh, the smaller the distance between them. For both sets of simulations, a series of queries are fed as inputs into both the JIT and TPS models. The JIT algorithm selects an optimal neighbourhood size N_{opt} offline via the final prediction error criterion. For each evaluation configuration, an independently generated testing dataset is used, with the number of test queries defined as approximately $N_t = \sigma N_h$, where $\sigma = 0.1$ and N_t is rounded to the nearest whole number. The test configurations are generated separately from and are not contained in the historical database. The position of each mesh node is then compared to evaluate the difference between the reference model (TPS) and the JIT prediction. Median execution time per query and the median speedup ratio between TPS and JIT time per query are recorded. The simulations are performed in MATLAB using a computer with a 12th generation 3.2 GHz i9 Intel Processor, 128 GB of RAM and an NVIDIA GeForce RTX 3060 GPU.

Varying mesh resolution with constant database size: A database of 504 TPS simulations are generated offline for 5 different mesh resolutions, and subsequently used for online prediction with the JIT model. Errors are reported relative to the larger of the mesh width and height, whichever is greater.

Table 1 and the first two panels of Fig. 4 summarize the results across meshes ranging from 30×40 to 160×200 nodes. As the resolution increases, JIT predictions become more accurate, with mean relative error dropping around 0.2% at the finest grid. This trend indicates that higher discretization of the tissue mesh provides richer geometric information,

TABLE 1 Modelling error between JIT and TPS reference across different mesh sizes, and processing time t_{JIT} and t_{TPS} of JIT and TPS, respectively. Errors are reported relative to the domain size. Time is reported per query in milliseconds.

Mesh	ϵ_{me}	ϵ_{95}	ϵ_{max}	t_{JIT}	t_{TPS}	Speedup
30×40	6.80×10^{-3}	4.33×10^{-2}	8.63×10^{-2}	0.142	160	1140
60×80	4.20×10^{-3}	1.60×10^{-2}	4.00×10^{-2}	0.229	1200	5280
90×120	3.00×10^{-3}	1.34×10^{-2}	3.46×10^{-2}	0.541	6850	12700
120×160	2.50×10^{-3}	1.08×10^{-2}	2.75×10^{-2}	0.836	16060	19200
160×200	2.00×10^{-3}	1.04×10^{-2}	2.49×10^{-2}	1.10	36240	31900

which the regression model can exploit to approximate deformations more faithfully. Meanwhile, TPS runtimes scale poorly with resolution, growing from tenths of a second to tens of seconds, whereas JIT runtimes remain below 2 ms across all grids. The combined effect is a dramatic increase in speedup, ranging from 1,140 times at 30×40 to 31,900 times at 160×200, with the largest speedup occurring at the finest tested mesh resolution. The online JIT runtime can be separated into two main components. Neighbour selection and weight computation operate on the low-dimensional needle configuration vectors ($n = 4$) and are therefore primarily affected by the database and neighbourhood sizes rather than the mesh resolution. In contrast, reconstruction of the predicted deformation requires the weighted combination of the $2MN$-dimensional neighbour outputs and therefore scales with the mesh resolution. This explains the moderate increase in JIT runtime as the number of mesh nodes increases.

The 160×200 mesh is the largest resolution evaluated in this study. Higher-resolution meshes can be accommodated, but increasing the mesh resolution increases both the size of the stored deformation fields and the computational cost of reconstructing the predicted mesh.

Varying database size with constant mesh resolution: The second group of simulations looked into the effect of increasing the database size while keeping the mesh resolution fixed at 90×120. Databases ranging from $N_h = 378$ to 1134 offline TPS simulations were generated, and the JIT predictor was evaluated on N_t novel test queries in each case. The same relative error metrics ($\epsilon_{\text{me}}, \epsilon_{95}, \epsilon_{max}$) were computed, along

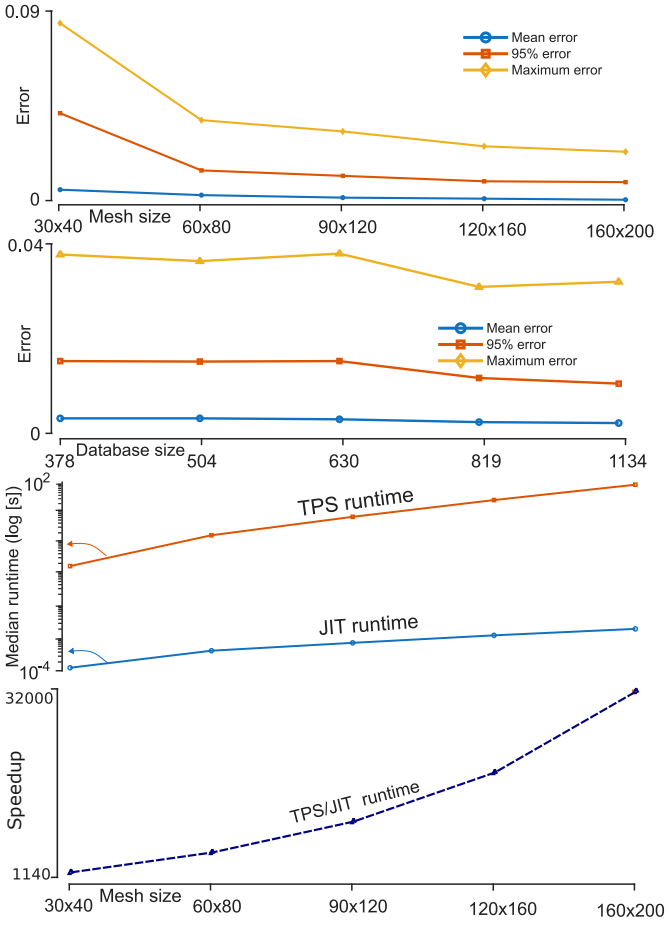


FIGURE 4 JIT prediction error and prediction error for different mesh sizes and different database sizes on a 90×120 mesh (top two panels), and runtime per query compared to TPS reference deformation across different mesh resolutions for a database of 504 over 50 test queries (bottom two panel).

with the median runtimes of JIT and TPS. Table 2 summarizes these results. Increasing the database size slightly reduces the error. This indicates that larger databases improve the robustness of the predictor across the majority of nodes, but occasional outlier errors remain comparable. Meanwhile, the runtime of the JIT predictor remains nearly constant at 0.41–0.48 ms per query, whereas TPS run-times are on the order of 5–7 s. As a result, the JIT speedup remains within 12,200–15,700 times across the tested database sizes, as the database size grows.

Fig. 4 summarizes these trends: mean, 95th percentile, and maximum errors decrease monotonically with database size. The diminishing returns beyond roughly 800 database entries suggest a practical balance between offline computation/storage and online accuracy, guiding database design for portable simulators.

The JIT model predicts the deformed coordinates of the mesh nodes rather than ultrasound pixel intensities directly. To generate the deformed ultrasound images shown in Fig. 5, the nodal displacement field is interpolated onto the ultrasound image grid using linear interpolation,

TABLE 2 Relative error between JIT predictions and TPS reference across different database sizes on a fixed 90×120 grid using independently generated testing datasets with $N_t \approx 0.1N_h$, and runtime t_{JIT} and t_{TPS}.

DB Size	ϵ_{me}	ϵ_{95}	ϵ_{max}	t_{JIT}	t_{TPS}	Speedup
378	3.50×10^{-3}	1.57×10^{-2}	3.84×10^{-2}	0.449	5460	12200
504	3.50×10^{-3}	1.56×10^{-2}	3.70×10^{-2}	0.474	7420	15700
630	3.30×10^{-3}	1.57×10^{-2}	3.86×10^{-2}	0.430	6590	15300
819	2.70×10^{-3}	1.21×10^{-2}	3.15×10^{-2}	0.427	6010	14100
1134	2.50×10^{-3}	1.09×10^{-2}	3.26×10^{-2}	0.413	5270	12800

and the resulting dense displacement field is used to remap the original image. The quantitative errors reported in this study are computed from the differences between the JIT and TPS mesh-node coordinates and therefore represent mesh deformation errors.

Fig. 5 shows the output of the JIT and TPS algorithms and the difference between their predictions for two sample queries using a mesh size of 90×120 and a database size of 504. The red line represents the needle. The corresponding deformed ultrasound images are shown in the bottom panel, illustrating how differences in the predicted mesh deformation translate to differences in the warped image. For the 90×120 mesh, the diagonal distance between adjacent nodes is approximately 0.03, meaning that the maximum observed nodal error is less than the diagonal spacing of two mesh nodes. While these discrepancies are visible locally, the proposed algorithm substantially reduces computation time relative to the TPS reference model.

To determine the benefit of the proposed JIT weighting strategy, two additional database-interpolation methods were considered and compared: nearest-neighbour (NN) retrieval and inverse-distance weighting (IDW). Both methods operate on the same historical database $\mathcal{S}$ and use the same Euclidean distance in needle-configuration space as the proposed JIT model. The IDW method provides a distance-only counterpart to JIT in which no Hessian information is used to describe the nonlinearity between nearby poses.

For nearest-neighbour interpolation, the database sample whose needle configuration is closest to the query $\mathbf{x}_0$ is selected as

$$j^* = \underset{j=1,\ldots,N_h}{\operatorname{argmin}} \|\mathbf{x}_j - \mathbf{x}_0\|_2 , \tag{32}$$

and its corresponding deformation field is returned directly:

$$\hat{\mathbf{g}}_{\mathrm{NN}}(\mathbf{x}_0) = \mathbf{g}_{j^*} . \tag{33}$$

For IDW, the N_{opt} database samples closest to the query are selected. Let

$$d_q = \|\mathbf{x}_q - \mathbf{x}_0\|_2 , \qquad q = 1, \ldots, N_{\mathrm{opt}}, \tag{34}$$

denote the distance of the q^{th} selected neighbour. The normalized IDW weights are

$$w_q^{\mathrm{IDW}} = \frac{(d_q + \varepsilon)^{-p}}{\sum_{r=1}^{N_{\mathrm{opt}}} (d_r + \varepsilon)^{-p}}, \tag{35}$$

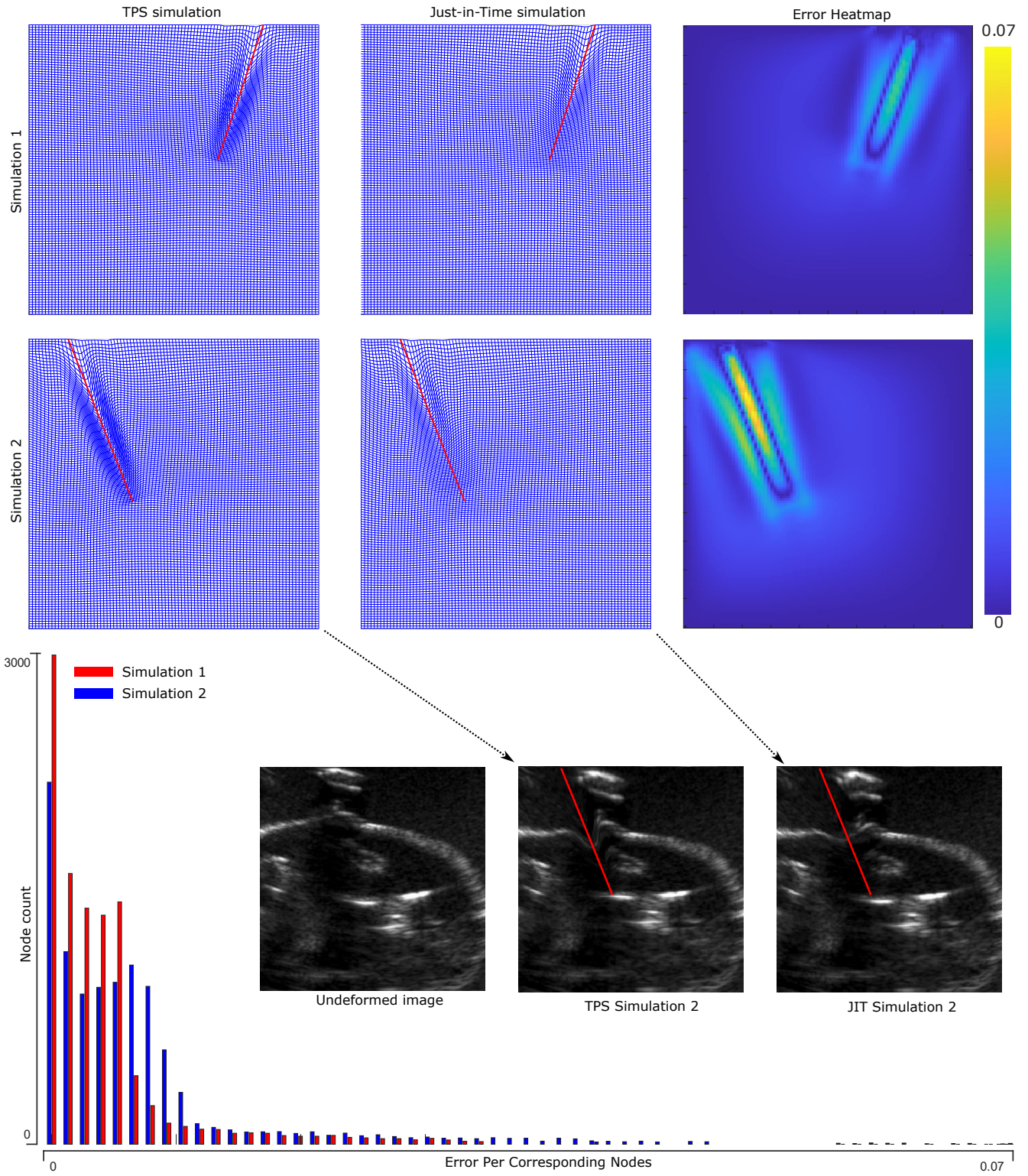


FIGURE 5 Sample mesh and image deformation for two different needle insertion simulations using both TPS (reference, first column) and JIT model (centre column). The third column shows the per-node error heatmap. The bottom histogram shows the absolute error for a representative query along with corresponding ultrasound images. The needle is indicated as the red line in the image (not modelled in this paper).

where $p = 2$ is the inverse-distance exponent and ε is a small positive constant used to avoid division by zero. The predicted deformation is then

$$\hat{\mathbf{g}}_{\mathrm{IDW}}(\mathbf{x}_0) = \sum_{q=1}^{N_{\mathrm{opt}}} w_q^{\mathrm{IDW}} \mathbf{g}_q. \tag{36}$$

TABLE 3 Comparison of the proposed JIT method with nearest-neighbour (NN) and inverse-distance weighting (IDW) baselines for a 90×120 mesh. All errors are computed relative to the TPS reference deformation on the normalized 1×1 domain.

Method	ϵ_{me}	ϵ_{95}	ϵ_{max}
JIT	3.00×10^{-3}	1.34×10^{-2}	3.46×10^{-2}
IDW (Non Hessian JIT)	4.02×10^{-3}	2.02×10^{-2}	4.61×10^{-2}
NN	8.42×10^{-3}	4.93×10^{-2}	8.86×10^{-2}

The comparison was performed using the representative 90×120 mesh, and the same $N_t = 50$ held-out TPS reference queries for all three methods.

As shown in Table 3, the proposed JIT weighting strategy produced lower errors than both lightweight baselines. Relative to IDW, JIT reduced the mean, 95^{th}-percentile, and maximum errors by a small amount. The improvement over NN was larger, indicating that interpolation between multiple relevant historical deformations is beneficial, while the additional first-order consistency and curvature-aware weighting used by JIT further improves accuracy over distance-only interpolation.

4 | DISCUSSION

The proposed Just-in-Time (JIT) modelling framework demonstrates that data-driven interpolation of precomputed biomechanical simulations can reproduce full tissue deformation fields with sub-percent accuracy and millisecond latency suitable for portable usPS trainers: across mesh resolutions (Table 1), mean relative error drops from 0.68% at 30×40 to $\approx 0.20\%$ at 160×200, while median per-query runtime remains ≤ 1.10 ms; TPS, by contrast, scales from 0.16 s to 36.24 s, yielding speedups ranging from 1,140 times to 31,900 times across the tested mesh resolutions. On a fixed 90×120 grid, enlarging the offline database from N_h=378 to 1134 improves mean and 95th-percentile relative errors from $(0.35\%, 1.57\%)$ to $(0.25\%, 1.09\%)$ while keeping JIT latency essentially flat at 0.41–0.48 ms (Table 2), indicating diminishing returns beyond 800 histories and offering a practical target for database sizing. This runtime stability makes JIT suitable for interactive use, including real-time surgical training systems where update rates on the order of milliseconds are required.

Database storage also increases with mesh resolution and the number of historical samples because each entry is a full $2MN$-dimensional deformation field. Assuming double-precision storage, the deformation fields in the largest database of the mesh-resolution study (160×200, $N_h = 504$) require approximately 258 MB, while those in the largest database-size configuration (90×120, $N_h = 1134$) require approximately 196 MB. These memory requirements are practical for many modern portable computing platforms, although the database footprint becomes an important design consideration for more resource-constrained embedded hardware.

Compared with existing real-time deformation strategies the proposed JIT method eliminates the need for explicit physical integration during runtime. Instead, it learns a local surrogate from an offline database and reconstructs new deformations through curvature-aware regression and KKT-constrained weighting. This design preserves the spatial continuity and smoothness characteristic of physics-based models while achieving deterministic latency on standard CPUs. The resulting update rates are well within the visual refresh cycle of ultrasound displays (30–60 Hz).

Residual discrepancies between JIT and TPS predictions occur mainly near the needle tip and along high-curvature regions of the shaft, where local deformations exhibit strong non-linearities. Although the effective Hessian reduces these residuals by capturing average curvature, the accuracy of the JIT prediction ultimately depends on the deformation behaviour represented in the historical database. Furthermore, model accuracy ultimately depends on the representativeness of the offline database: sparse coverage or unmodelled deformation modes can introduce bias. These factors suggest that adaptive database expansion and anisotropic distance metrics could further improve performance. Incorporating temporal coherence into the local regression would also enable dynamic predictions for continuous tool motion rather than static query points.

The database considered in this study represents a homogeneous two-dimensional tissue domain and does not explicitly distinguish between different tissue types or other anatomical structures. Tissue heterogeneity, out-of-plane rigid structures, and contact with structures along the needle path could alter the resulting deformation field and are not explicitly represented in the present database.

These assumptions define the scope of the TPS-based database used in the present evaluation rather than an inherent limitation of the JIT framework. If the database contains deformation fields generated in the presence of heterogeneous tissue stiffness or rigid anatomical structures, the corresponding deformation behaviour can be incorporated into the JIT prediction. Recent studies have shown that physics-based soft-tissue models can be informed by visual observations and multimodal force–vision measurements to improve deformation estimation and parameter calibration[34,35]. Such experimentally informed models could therefore be used to construct higher-fidelity heterogeneous or three-dimensional deformation databases for future JIT implementations.

The accuracy reported should be interpreted as fidelity to the reference deformation model rather than as direct experimental validation of the JIT model. The TPS deformation engine used to construct the historical database was previously compared with an equivalent FEM model and evaluated against ex vivo porcine tissue deformation measured from ultrasound images using speckle tracking[31]. These prior comparisons provide supporting validation for the deformation behaviour represented by the TPS model; however, the JIT predictions presented here have not been independently compared with FEM or experimental tissue measurements. Direct validation of the JIT predictions against FEM and experimental measurements will be considered in future work.

5 | CONCLUSION

Beyond algorithmic improvements, the JIT framework has broader implications for surgical simulation and computer-assisted interventions. Its millisecond-scale prediction time and deterministic behaviour enable integration into embedded systems, haptic devices, and augmented-reality displays without dedicated GPU hardware. This capability complements recent advances in ultrasound-based navigation and deformation compensation, providing a computational link between offline biomechanical realism and real-time interactivity. In future work, we plan to couple the JIT model with ultrasound rendering engines and force-feedback interfaces to create a complete, cyber-physical training environment for ultrasound-guided percutaneous procedures. To incorporate haptic feedback, the current JIT input–output descriptors could be extended to include dynamic information. For example, the input vector could include the needle position together with its insertion velocity, while each historical database entry could store both the resulting mesh deformation and the corresponding needle–tissue interaction force. During online prediction, neighbouring configurations could then be selected using the augmented input descriptor, and both the deformation field and interaction force could be interpolated from the corresponding database outputs. This would allow the same JIT framework to provide synchronized visual deformation and haptic-force feedback in future training simulators. The present validation primarily evaluates the fidelity of the JIT predictions relative to a two-dimensional TPS-based reference model. Extension and validation of the framework for heterogeneous three-dimensional tissues and more realistic surgical environments will require further investigation.

Overall, this study demonstrates that Just-in-Time modelling can bridge the long-standing gap between the accuracy of biomechanical simulation and the responsiveness required for clinical realism. By achieving millisecond-scale predictions from an offline database of physics-based simulations, JIT enables practical, portable, and scalable simulation platforms for training, planning, and potentially intraoperative guidance in ultrasound-guided interventions.

AUTHOR CONTRIBUTIONS

Kian Zalzalah conceived the study, developed the methodology, implemented the models, performed the simulations, analyzed the results, and drafted the manuscript. Carlos Rossa supervised the research, contributed to the methodological development, and revised the manuscript. Both authors reviewed and approved the final manuscript.

FINANCIAL DISCLOSURE

This research was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

CONFLICT OF INTEREST

The authors declare no competing interests.

ETHICS STATEMENT

None.

References

1. Miller DL, Abo A, Abramowicz JS, et al. Diagnostic Ultrasound Safety Review for Point-of-Care Ultrasound Practitioners. *Journal of Ultrasound in Medicine*. 2020;39(6):1069–1084. doi: 10.1002/jum.15202

2. Song Y, Ma Y, Song Y, Fei X. Evaluating the Learning Curve for Percutaneous Nephrolithotomy under Total Ultrasound Guidance. *PLOS ONE*. 2015;10(8):e0132986. doi: 10.1371/journal.pone.0132986

3. Mazdarani H, Cotton A, Rossa C. 2D Ultrasound-Guided Visual Servoing for In-Plane Needle Tracking in Robot-Assisted Percutaneous Nephrolithotomy. In: IEEE. 2023:1786–1791

4. Trojak M, Stanuch M, Kurzyna M, Darocha S, Skalski A. Mixed Reality Biopsy Navigation System Utilizing Markerless Needle Tracking and Imaging Data Superimposition. *Cancers*. 2024;16(10):1894. doi: 10.3390/cancers16101894

5. Lelevé A, McDaniel T, Rossa C. Haptic Training Simulation. *Frontiers in Virtual Reality*. 2020;1. doi: 10.3389/frvir.2020.00003

6. Selim M, Eisenburger L, Dijkhuis T, et al. Experimental evaluation of virtual needle insertion framework with enhanced haptic feedback. *International Journal of Computer Assisted Radiology and Surgery*. 2025;20(8):1643–1652. doi: 10.1007/s11548-025-03420-2

7. Zhang L, Yang D, Niu B, et al. Smooth Path Planning and Dynamic Contact Force Regulation for Robotic Ultrasound Scanning. *IEEE Robotics and Automation Letters*. 2025;10(10):10570-10577. doi: 10.1109/LRA.2025.3604746

8. Adagolodjo Y, Goffin L, Mathelin dM, Courtecuisse H. Inverse real-time Finite Element simulation for robotic control of flexible needle insertion in deformable tissues. In: IEEE. 2016:2717–2722

9. Gjerald SU, Brekken R, Bø LE, Hergum T, Nagelhus Hernes TA. Interactive development of a CT-based tissue model for ultrasound simulation. *Computers in Biology and Medicine*. 2012;42(5):607–613. doi: 10.1016/j.compbiomed.2012.02.006

10. Choi KS, Soo S, Chung FL. A virtual training simulator for learning cataract surgery with phacoemulsification. *Computers in Biology and Medicine*. 2009;39(11):1020–1031. doi: 10.1016/j.compbiomed.2009.08.003

11. Rahimi AM, Uluç E, Hardon SF, Bonjer HJ, Peet v. dDL, Daams F. Training in robotic-assisted surgery: a systematic review of training modalities and objective and subjective assessment methods. *Surgical Endoscopy*. 2024;38(7):3547–3555. doi: 10.1007/s00464-024-10915-7

12. Gani A, Pickering O, Ellis C, Sabri O, Pucher P. Impact of haptic feedback on surgical training outcomes: A Randomised Controlled Trial of haptic versus non-haptic immersive virtual reality training. *Annals of Medicine and Surgery*. 2022;83:104734. doi: 10.1016/j.amsu.2022.104734

13. Sainsbury B, Łącki M, Shahait M, et al. Evaluation of a Virtual Reality Percutaneous Nephrolithotomy (PCNL) Surgical Simulator. *Frontiers in Robotics and AI*. 2020;6. doi: 10.3389/frobt.2019.00145

14. Wu L, Seibold M, Cavalcanti NA, et al. A novel augmented reality-based simulator for enhancing orthopedic surgical training. *Computers in Biology and Medicine*. 2025;185:109536. doi: 10.1016/j.compbiomed.2024.109536

15. Rossa C, Tavakoli M. Issues in closed-loop needle steering. *Control Engineering Practice*. 2017;62:55–69. doi: 10.1016/j.conengprac.2017.03.004

16. Tahmasebi AM, Hashtrudi-Zaad K, Thompson D, Abolmaesumi P. A Framework for the Design of a Novel Haptic-Based Medical Training Simulator. *IEEE Transactions on Information Technology in Biomedicine*. 2008;12(5):658–666. doi: 10.1109/TITB.2008.926496

17. Chorney HV, Forbes JR, Driscoll M. System identification and simulation of soft tissue force feedback in a spine surgical simulator. *Computers in Biology and Medicine*. 2023;164:107267. doi: 10.1016/j.compbiomed.2023.107267

18. Mohammadi H, Ebrahimian A, Maftoon N. Finite-Element Modelling of Needle-Tissue Interactions. *Archives of Computational Methods in Engineering*. 2024;31(3):1363–1404. doi: 10.1007/s11831-023-10020-3

19. Kobayashi Y, Onishi A, Watanabe H, et al. Development of an integrated needle insertion system with image guidance and deformation simulation. *Computerized Medical Imaging and Graphics*. 2010;34(1):9–18. doi: 10.1016/j.compmedimag.2009.08.008

20. Khadem M, Rossa C, Sloboda RS, Usmani N, Tavakoli M. Mechanics of Tissue Cutting During Needle Insertion in Biological Tissue. *IEEE Robotics and Automation Letters*. 2016;1(2):800–807. doi: 10.1109/LRA.2016.2528301

21. Burger B, Bettinghausen S, Radle M, Hesser J. Real-Time GPU-Based Ultrasound Simulation Using Deformable Mesh Models. *IEEE Transactions on Medical Imaging*. 2013;32(3):609–618. doi: 10.1109/TMI.2012.2234474

22. Xie H, Song J, Gao B, Zhong Y, Gu C, Choi KS. Finite-element kalman filter with state constraint for dynamic soft tissue modelling. *Computers in Biology and Medicine*. 2021;135:104594. doi: 10.1016/j.compbiomed.2021.104594

23. Hou W, Liu PX, Zheng M. A new model of soft tissue with constraints for interactive surgical simulation. *Computer Methods and Programs in Biomedicine*. 2019;175:35–43. doi: 10.1016/j.cmpb.2019.03.018

24. Smit JN, Kuhlmann KFD, Thomson BR, Kok NFM, Ruers TJM, Fusaglia M. Ultrasound guidance in navigated liver surgery: toward deep-learning enhanced compensation of deformation and organ motion. *International Journal of Computer Assisted Radiology and Surgery*. 2024;19(1):1–9. doi: 10.1007/s11548-023-02942-x

25. Kaya M, Senel E, Ahmad A, Bebek O. Visual tracking of multiple moving targets in 2D ultrasound guided robotic percutaneous interventions. In: IEEE. 2017:1996–2002

26. Frisken S, Luo M, Juvekar P, et al. A comparison of thin-plate spline deformation and finite element modeling to compensate for brain shift during tumor resection. *International Journal of Computer Assisted Radiology and Surgery*. 2020;15(1):75–85. doi: 10.1007/s11548-019-02057-2

27. Yamaguchi S, Tsutsui K, Satake K, Morikawa S, Shirai Y, Tanaka HT. Dynamic analysis of a needle insertion for soft materials: Arbitrary Lagrangian–Eulerian-based three-dimensional finite element analysis. *Computers in Biology and Medicine*. 2014;53:42–47. doi: 10.1016/j.compbiomed.2014.07.012

28. Chentanez N, Alterovitz R, Ritchie D, et al. Interactive simulation of surgical needle insertion and steering. In: ACM. 2009:1–10

29. Wittek A, Bourantas G, Zwick BF, Joldes G, Esteban L, Miller K. Mathematical modeling and computer simulation of needle insertion into soft tissue. *PLOS ONE*. 2020;15(12):e0242704. doi: 10.1371/journal.pone.0242704

30. Guo Y, Kou D, Zhang X, Zhang L, Cao K. A real-time virtual liver simulator using position based dynamics with compliant constraints for medical applications. *Scientific Reports*. 2025;15(1):26557. doi: 10.1038/s41598-025-11519-4

31. Zalzalah K, Selladurai S, Rossa C. Real-Time Simulation of Ultrasound Image Deformation Using Thin Plate Spline. In: IEEE. 2025:1–6

32. Vidal FP, John NW, Healey AE, Gould DA. Simulation of ultrasound guided needle puncture using patient specific data with 3D textures and volume haptics. *Computer Animation and Virtual Worlds*. 2008;19(2):111-127. doi: https://doi.org/10.1002/cav.217

33. Barnouin C, Zara F, Jaillet F. A Real-time Ultrasound Rendering with Model-based Tissue Deformation for Needle Insertion. In: INSTICC. SciTePress 2020:235-246

34. Zhang L, Yang D, Yang H, Liu Z, Liu Y, Jiang L. Physics-Visual Fusion for Cross-Posture Deformation Mapping in Robotic Breast Ultrasound. *IEEE Robotics and Automation Letters*. 2026;11(9):10138-10145. doi: 10.1109/LRA.2026.3711251

35. Zhang L, Liu Y, Liang Z, Yang D. Force–Vision Multimodal Measurement of Model-Equivalent Elastic Parameters for Soft Tissue Modeling. *Biomimetics*. 2026;11(10). doi: 10.3390/biomimetics11100691