Uncertainty-Aware Simulation of Ultrasound-Guided Robotic Needle Insertion

Juliane Mercoli, University of Washington

Abstract

We present a complete uncertainty-aware software simulation for ultrasound-guided robotic needle insertion. To implement the full pipeline, we rely on techniques that include MC-Dropout segmentation, connected-component fusion, world-space particle filtering, and a pause-and-re-scan policy, all of which are integrated in a two-link arm simulation. The system seeks to quantify uncertainty to make robotic needle insertion safer.

1. Introduction and Motivation

Over the past four decades, medical robotics has evolved to be able to perform some of the most complex tasks and image-guided systems are more and more used used in hospitals. Yet despite their today's success, every year reports of robot errors, whether due to sensor noise, calibration drift, or control faults, continue to appear in literature, sometimes resulting in patient harm and even fatalities. Such incidents emphasize the fact that, in safety-critical domains like medicine, reliability of perception and decision-making are crucial robotic problems.

Among the newer classes of medical robots are ultrasound-guided needle insertion devices. These robotic arms aim to automate tasks such as regional anesthesia, vascular access, and tissue biopsy with millimeter accuracy. Compared to traditional procedures, robotic arms have been shown to reduce variability in needle angle and depth, potentially lowering complication rates. However, these robots rely heavily on live ultrasound images, which are known to be noisy. Furthermore, patient anatomy varies widely in tissue's response to ultrasounds and nerve visibility, making the generalization of image segmentation a challenge.

In practice, most current ultrasound-guided needle robots treat the output of their segmentation networks as ground truth. A convolutional neural network, for example, may generate a binary mask of the target nerve, and the robot planner simply treats that mask as exact. This approach overlooks the fact that deep learning models can be overconfident on unfamiliar or noisy ultrasound image inputs. Segmentation errors then propagate through the planning pipeline and result in unsafe needle trajectories.

To address this issue, we focus on quantifying uncertainty, which involve relying on techniques such as Monte Carlo Dropout to allow a (potentially pretrained) network to produce not only a mean prediction but also a pixel-wise measure of uncertainty. When uncertainty is explicitly represented, the arm's planners and controllers can make more risk-aware decisions.

We have thus decided to propose a new complete, software-only pipeline that extends from ultrasound image segmentation to uncertainty-aware arm's trajectory planning. This is different from most current research efforts, which emphasizes on hardware improvements, but which are not integrating the uncertainty of the robot's vision model into the control loop.

In this project, we propose a simulation pipeline that treats the segmentation network as a "noisy sensor", explicitly quantifying uncertainty, and using that information to create safer needle insertion behaviors. Our key insight is that by fusing noisy segmentation outputs into a metric estimate and embedding a confidence-based "pause and re-scan" policy, we can demonstrate measurable improvements in insertion safety within a fully simulated environment.

Specifically, our pipeline is made of three main steps:

1. Uncertainty-focus segmentation: We add to the U-Net model a Monte Carlo Dropout at infer-

ence time, producing both a soft nerve mask and a pixel-wise uncertainty map.

2. Metric fusion: We extract the most likely nerve centroid via connected-component analysis, then project it into robot's coordinates and fuse multiple noisy estimates using a particle filter in world coordinates.

3. Uncertainty-aware control: We implement a two-link arm simulation that monitors regional confidence. When confidence falls below a threshold, the system automatically pauses, simulates a "re-scan" of the ultrasound region, and recomputes its estimate nerve location.

2. Related Work

- **"Safe and Efficient Ultrasound-Guided Needle Placement Using Uncertainty-Aware Deep Learning"** [3] This work experiments with different U-NET models for the nerve segmentation problem and concludes that the classic U-NET performs better than all others, including a U-NET with Monte Carlo Dropout included in the training.

- **"An ultrasound visual servoing dual-arm robotics system for needle placement in brachytherapy treatment"** [2] This work describes a dual-arm system that attempts to get the best view to orient the needle for brachytherapy. It tackles the joint control of imaging and insertion to improve placement precision.

3. Methodology

3.1. Segmentation with MC-Dropout

We have used the Kaggle dataset "Ultrasound Nerve Segmentation" [4] which contains ultrasound images of the Brachial Plexus (BP), where the nerves locations had been annotated by doctors. We resized the dataset to 128*128 images. We first created a 20% validation set and then trained a U-NET on the training split of the dataset, using cross-entropy and Dice loss. Once we got good enough performances on the validation set, we loaded the trained weights and augmented the network at test time with 1% dropout after every convolutional block. For a chosen image from the validation set, we then performed $K = 20$ stochastic forward passes, producing a probability maps of where the nerve may be located. This produced K soft probability maps $p_k(x, y)$, which we aggregated as:

$$\bar{p}(x, y) = \frac{1}{K} \sum_{k=1}^{K} p_k(x, y),$$

$$\mathrm{Unc}(x, y) = \sqrt{\frac{1}{K} \sum_{k=1}^{K} \big(p_k(x, y) - \bar{p}(x, y)\big)^2}.$$

Here, $\bar{p}$ is used as the soft segmentation mask and Unc encodes uncertainty.

3.2. Connected-Component Fusion

After obtaining the mean probability map $\bar{p}(x, y)$, we computed two complementary estimates of the nerve location:

1. Mean-Monte Carlo centroid μ_{MC}: We treated $\bar{p}$ as a continuous "mass" distribution and computed its center of mass:

$$\mu_{\mathrm{MC}} = \frac{\sum_{x,y} \bar{p}(x, y)\, [x, y]^\top}{\sum_{x,y} \bar{p}(x, y)}.$$

This gave us a subpixel-accurate estimate $\mu_{\mathrm{MC}} = (u_{\mathrm{MC}}, v_{\mathrm{MC}})$, reflecting the average predicted nerve location.

2. Otsu thresholding and connected components: We applied Otsu's method to $\bar{p}$ to find an optimal threshold T on 5x5 patches to separate probable nerve pixels from background:

$$T = \mathrm{OtsuThreshold}(\bar{p}).$$

We then binarized:

$$M(x, y) = \begin{cases} 1, & \bar{p}(x, y) \geq T, \\ 0, & \text{otherwise}, \end{cases}$$

and labeled each connected component in M, and computed its discrete centroid center with:

$$(u_i, v_i) = \frac{1}{|C_i|} \sum_{(x,y) \in C_i} (x, y).$$

3. Selecting the primary component: To pick the connected component that most likely corresponds to the nerve, we chose the one whose centroid (u_i, v_i) is closest (in Euclidean distance) to the mean Monte-Carlo centroid μ_{MC}:

$$(u_{\mathrm{CC}}, v_{\mathrm{CC}}) = \arg\min_{(u_i,v_i)} \| (u_i, v_i) - \mu_{\mathrm{MC}} \|.$$

This ensures that even if the binarized mask is made of multiple blobs, we consistently selected the blob most aligned with the network's average prediction as our estimated nerve.

4. World-Space Particle Filtering We projected $(u_{\mathrm{CC}}, v_{\mathrm{CC}})$ into millimeters (robot's space world) through the traditional camera model:

$$X = \frac{(u - c_x)Z}{f} \quad Y = \frac{(v - c_y)Z}{f}$$

and general value for Z, the depth, f, the focal length and cx, cy, the camera's center point:

$$Z = 20 \text{ mm}, \ f = 500 \text{ px}, \ c_x = c_y = 128/2 = 64.$$

We drew $N = 200$ samples according to the distribution $p_i \sim \mathcal{N}([X, Y], \sigma_{\mathrm{mm}}^2 I_2)$ (with $\sigma_{\mathrm{mm}} = \sigma_{\mathrm{meas}} Z/f$) and took the mean as our fused 3D nerve estimate `world_mean`.

5. Trajectory Planning We defined on the image an entry point where the robot is initially located $[X_h, Y_h]$. We then constructed a straight line trajectory in task space from $[X_h, Y_h]$ to `world_mean`, which we discretized into M points:

$$[X_i, Y_i] = [X_h, Y_h] + \tfrac{i}{M-1}\big(\texttt{world\_mean} - [X_h, Y_h]\big)$$

$$\text{with } i = 0, \ldots, M-1.$$

Each point (X_i, Y_i) is converted to joint angles (θ_1^i, θ_2^i) via analytic inverse kinematics for a planar two-link arm.

6. Two-Link Arm Simulation We simulated a planar two-link manipulator with link lengths $l_1 = l_2 = 40$px. At each control timestep, we computed position commands to move from $(\theta_1^{i-1}, \theta_2^{i-1})$ to (θ_1^i, θ_2^i), simulating the arms' moves to reach the nerve.

7. Confidence-Based Pause and Re-Scan Policy Throughout the trajectory, we monitored the local confidence by averaging $\bar{p}$ over a small patch centered at current nerve estimate. If this confidence falls below a threshold τ, the system:

1. Pauses motion
2. Simulates a new ultrasound frame
3. Recomputes $\bar{p}$, connected-component fusion, and particle filtering to update `world_mean`
4. Replans the remaining waypoints toward the updated target
5. Resumes execution

4. Experimental Setup and Results

4.1. Segmentation Results

Including Monte-Carlo Dropout in the U-NET training has been shown to lower segmentation performance for this dataset, in different studies. Indeed, when we added a layer of dropout to the U-NET, DICE and IoU dropped significantly (below 0.6), even when the dropout rate was low, which shows that the network was not learning how to recognize the nerve anymore and treated the full image as background noise. So, instead of integrating dropout into training, we first trained the U-NET without it and inserted the dropout layers on the pretrained version where we had loaded the original weights.

Baseline U-Net After experimenting with different settings, this is the U-NET architecture that seems to be performing better:

Input	Input((IMG, IMG, 1))
Down 1	2×Conv2D(64) + MaxPool2D
Down 2	2×Conv2D(128) + MaxPool2D
Down 3	2×Conv2D(256) + MaxPool2D
Down 4	2×Conv2D(512) + MaxPool2D
Bottleneck	2×Conv2D(1024) + Dropout(0.1)
Up 1	Transpose(512) + Concat + 2×Conv2D(512)
Up 2	Transpose(256) + Concat + 2×Conv2D(256)
Up 3	Transpose(128) + Concat + 2×Conv2D(128)
Up 4	Transpose(64) + Concat + 2×Conv2D(64)
Output	Conv2D(1, sigmoid)

Table 1. U-Net architecture overview

Results on the training set: Dice = 0.78, IoU = 0.63 Results on the validation set: Dice = 0.65, IoU = 0.49

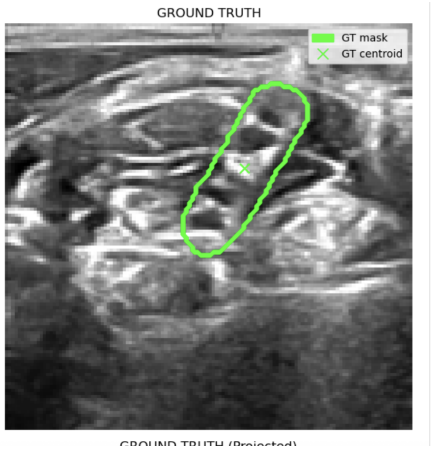

Figure 1. Nerve location - true value

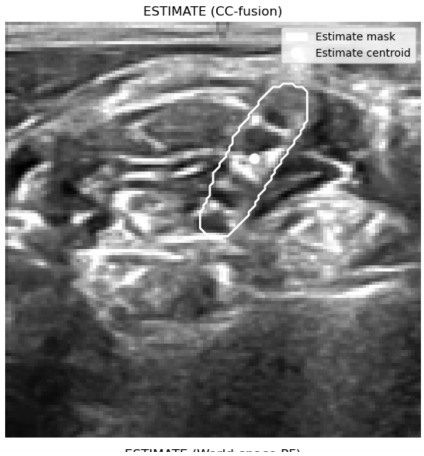

Figure 2. Nerve location - prediction

We achieve a very high accuracy, of over 99%, with many different settings. However, this is likely due to the fact that most (over 90%) of the images are made of ”background noise” and not the actual nerve which means that a network that completely misses the nerve and identifies everything as noise will still have a great accuracy. This is why we decided to train to optimize DICE and IoU metrics, which are much more representative here.

4.2. Measuring Uncertainty - Results

- The average standard deviation of the 20 Monte-Carlo Dropout centroids across over the validation set (230 frames) is about 8.30 px. This is then used as the noise parameter in the particles filter.

- We have an ECE of 0.012, which means that on average our predicted confidence deviates from true accuracy by only 1.2%. We computed a reliability diagram over B=10 bins using the mean MC-dropout soft masks. The uncertainty values appear highly correct.

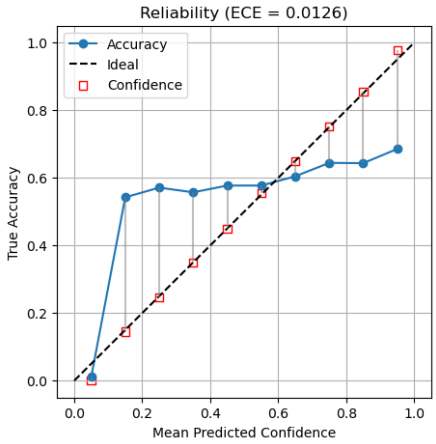

Figure 3. Note: ideal means that if the model says “I’m x% sure,” it really is correct 70% of the time

4.3. Trajectory and Simulation - Results

- The ground-truth nerve location in the example image is centered around [76.12, 40.95]px, obtained by averaging the (x,y) coordinates of all “nerve” pixels on true mask. After applying the Connected Component filter, the estimated nerve location is estimated to be at [76.30, 41.84] px.

- The Particle Filter fused centroid is [0.49, -0.89, 20.]mm in robot coordinates, which converts back to [76.07, 41.82]px. This has been obtained by mapping the fused pixel centroid into 2D via the chosen pixel-to-meter scale (1 px = 1 mm) and applying the camera-to-robot coordinate transform.

- The Particle Filter fused centroid leads to a pixel-error of 3.7px, which corresponds to a world-space error of $\approx 139.3\,\mu$m.

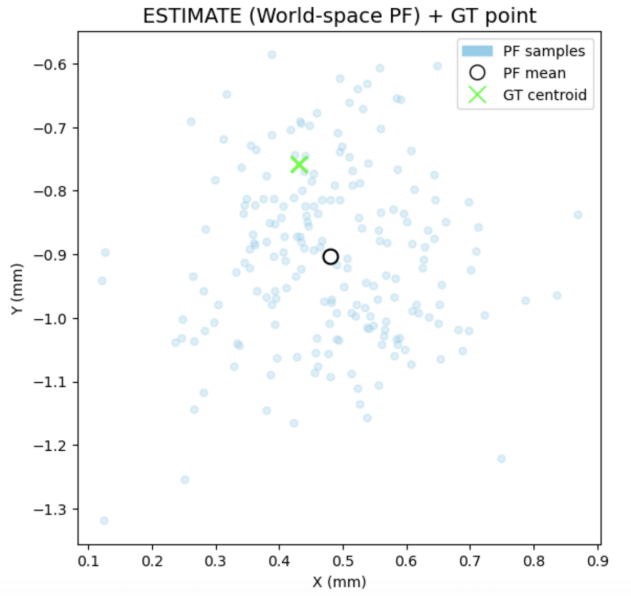

Figure 4. Particle-filter samples in robot world coordinates for the chosen image. The spread of the samples reflects the measurement noise σ_{mm}, while the mean consistently converges near the ground-truth position.

We then simulated the robot arm by creating a planar two-link arm, with a link length of 40 px, positioned on the ultrasound image, with left and right bases at $[34, 20]$ and $[94, 20]$. We also used Matplotlib to demonstrate how the arm finds and reaches the nerve step by step.

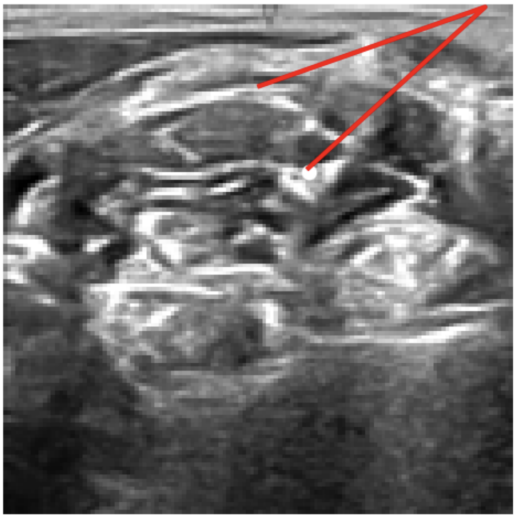

Figure 5. Demo of the simulation of the arm reaching the nerve (at the last step

4.4. Pause and Re-Scan { Results

Since the dataset provides only one static ultrasound slice per nerve, we simulate getting a new ultrasound from the re-scan by applying a small random affine transformation to the original image (such as translations up to ±2px and rotations up to ±2°, and scale ±1).

- With a threshold $\tau = 0.2$ (the perception model is at least 20% sure of its estimate of the nerve location). We tested with other thresholds, but given our static dataset, it does not make much of a difference.
- One pause and re-scan was triggered (at step 87/150)
- Final localization error: 3.5px (vs 3.7px initial error)

5. Discussion

Our experiment shows that explicitly modeling segmentation uncertainty can improve the safety of ultrasound-guided needle insertion in simulation. The Monte-Carlo Dropout step reliably (given the high ECE obtained) highlights low-confidence regions. The connected component and particle–filter fusion produces a quite stable estimate even when the raw segmentation is noisy. Integrating a "pause and re-scan" policy could allow the system to catch potential failures before they translate into unsafe trajectories, reducing worst-case localization error. However, this needs to be much more extensively tested, including with real live ultrasound imaging rather then on just one static image. Indeed, here we rely on a single static slice per nerve, and our re-scan (through small affine transformations) can only very broadly approximate a real robotic arm's perception. Also, our planar two-link simulation is a very simplified version of the real robotic arms used today for needle insertion and omits many degrees of freedom. As a side note, the end-to-end simulation, once the U-NET is trained, is quite efficient, taking about 5 minutes and with the vast majority of the time being used to run the Monte-Carlo Dropout passes.

This work also raises important ethical considerations, including the limitation of the diversity of patients' anatomy in the dataset among others.

6. Future Work

- Most of the filtering work was motivated by the fact that the segmentation U-NET is not performing really well. Testing other perception models can be useful to improve nerve localization.
- Adapt for and test with a stream of ultrasounds so we can test the pause and re-scan policy for real (and not just transform an image). This would be much more meaningful as the network could get a better view of the nerve as it approaches it so it could become surer of the nerve location as it approaches it.
- If we had live imaging, we could integrate a SLAM pipeline to register successive ultrasound frames, build a coherent 3D map of the tissue, and continuously refine the nerve's location as the arm moves. Although our current study is limited to a single static image, future work could explore such multi-view SLAM. SLAM would be particularly interesting to help build a 3D map, as ultrasound only sees 2D but we have tissue that have different volumes that the robotic arm must avoid.
- Test on a 3D robot arms to see if it moves as planned.

7. Conclusions

In conclusion, we have presented a fully software-based pipeline that extends from robotic perception uncertainty to the final motion planning for needle insertion. By treating the segmentation network as a noisy sensor, fusing its outputs with particle filtering, and enforcing a confidence-based safety policy, we demonstrate a new approach to risk-aware needle guidance in simulation. Future work is needed to extend this framework to real-time live ultrasound imaging, account for more complex kinematic models, and ultimately test with a real robot.

References

[1] X. Du, D. Lu, Q. Song, X. He, X. Wang, Y. Li, and X. Yang.
Robotic needle insertion guided by real-time 3D ultrasound imaging: A review.
International Journal of Imaging Systems and Technology, 28(1):3–13, 2018.
https://onlinelibrary.wiley.com/doi/full/10.1155/int/6114741

[2] Y. Li *et al.*, "An ultrasound visual servoing dual-arm robotics system for needle placement in brachytherapy treatment," *Front. Robot. AI*, vol. 12, art. 1558182, 2025. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC11986853/ 2

[3] M. A. M. de Rooij. *Safe and Efficient Ultrasound-Guided Needle Placement Using Uncertainty-Aware Deep Learning*. Master's thesis, Delft University of Technology, 2022.
https://repository.tudelft.nl/record/uuid:9652d083-8159-43da-bf83-679c951f58f5 2

[4] Kaggle Dataset - Ultrasound Nerve Segmentation Available: https://www.kaggle.com/competitions/ultrasound-nerve-segmentation/data?select=test 2