

# Self-Supervised Learning for Interactive Perception of Surgical Thread for Autonomous Suture Tail-Shortening

Vincent Schorp<sup>1,2</sup>, Will Panitch<sup>1</sup>, Kaushik Shivakumar<sup>1</sup>, Vainavi Viswanath<sup>1</sup>, Justin Kerr<sup>1</sup>  
Yahav Avigal<sup>1</sup>, Danyal M Fer<sup>3</sup>, Lionel Ott<sup>2</sup>, Ken Goldberg<sup>1</sup>

**Abstract**—Accurate 3D sensing of suturing thread is a challenging problem in automated surgical suturing because of the high state-space complexity, thinness and deformability of the thread, and possibility of occlusion by the grippers and tissue. In this work we present a method for tracking surgical thread in 3D which is robust to occlusions and complex thread configurations, and apply it to autonomously perform the surgical suture “tail-shortening” task: pulling thread through tissue until a desired “tail” length remains exposed. The method utilizes a learned 2D surgical thread detection network to segment suturing thread in RGB images. It then identifies the thread path in 2D and reconstructs the thread in 3D as a NURBS spline by triangulating the detections from two stereo cameras. Once a 3D thread model is initialized, the method tracks the thread across subsequent frames. Experiments suggest the method achieves a 1.33 pixel average reprojection error on challenging single-frame 3D thread reconstructions, and an 0.84 pixel average reprojection error on two tracking sequences. On the tail-shortening task, it accomplishes a 90% success rate across 20 trials. Supplemental materials are available at: <https://sites.google.com/berkeley.edu/autolab-surgical-thread/>

## I. INTRODUCTION

Many steps in suturing, such as tail-shortening (where a thread is pulled through a suture to a desired length) and knot tying, require accurate thread tracking, which is particularly challenging due to the thin and flexible nature of suturing thread, as well as its propensity for self-intersections and partial occlusions.

In this paper, we propose a novel interactive perception system for tracking suture thread in 3D, which we apply to track thread in the autonomous tail-shortening task described in Figure 1, requiring precise tracking to avoid pulling the thread too far out of the suture.

The learned 2D suturing thread detection model is trained using the Labels from UltraViolet (LUV) method [1] for self-supervised data collection, which has previously been shown to be effective for detecting cables and surgical needles. We extend it to surgical threads and combine the detection network with a 3D tracking method for temporal stability. Modeling the thread in 3D is a non-trivial task due to its complex shape and unclear endpoints. To address this issue, we model the thread as a 3D Non-Uniform Rational B-Spline

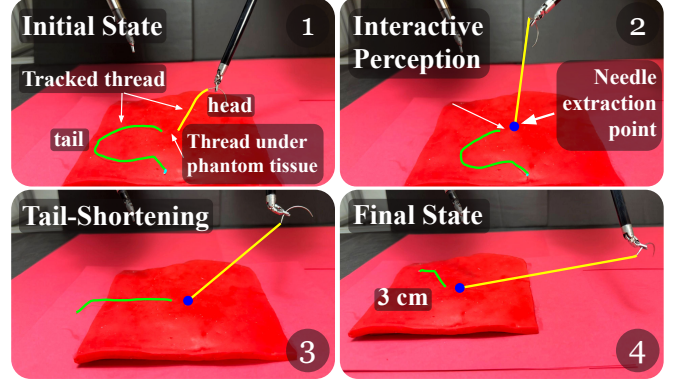


Fig. 1. **Surgical Suture Tail-shortening with 3D Thread Tracking.** We present a thread tracking method which we use for automating surgical suture “tail-shortening”, i.e. pulling thread through tissue until a desired length of thread remains exposed. Initially, the robot grasps the needle close to the wound. It uses interactive perception to determine which portions of the reconstructed thread are on the needle side (head) vs slack side (tail). The system accomplishes tail-shortening by visually servoing the needle driver until a desired tail length remains.

(NURBS) [2] based on stereo images of the scene. We adapt the thread model across frames by optimizing spline control points to minimize the error between the current detections and the reprojection of the 3D spline into the images.

Using NURBS optimization alone can break down in complex thread configurations because of false-positive detections or self-intersections. To address this, we develop an analytic 2D tracing approach based on prior work for cable untangling [3], which is used as a prior to prevent the NURBS optimization from collapsing in the presence of distracting false detections or challenging thread configurations.

Experiments conducted on a physical Intuitive Surgical da Vinci Research Kit (dVRK) RSA demonstrate that the system utilizing our thread tracker achieves 18/20 successful trials on the tail-shortening task (shown in Figure 1).

This paper makes the following contributions:

- 1) A 3D surgical thread tracking algorithm, described in Figure 2, that combines a learned thread detection module trained on data collected in a self-supervised fashion with a NURBS spline optimization.
- 2) An interactive perception approach to suture thread tail-shortening which utilizes the thread tracker with visibility-maximizing manipulation to estimate the length of remaining thread tail.
- 3) Data from experiments evaluating the perception components individually and applied to suture tail-shortening, achieving a 90% success rate.

The AUTOLab at the University of California, Berkeley (automation.berkeley.edu), {vschorp, goldberg}@berkeley.edu

<sup>1</sup> AUTOLAB at University of California, Berkeley

<sup>2</sup> Autonomous Systems Lab, ETH Zurich

<sup>3</sup> MD, Department of Surgery, University of California San Francisco East Bay

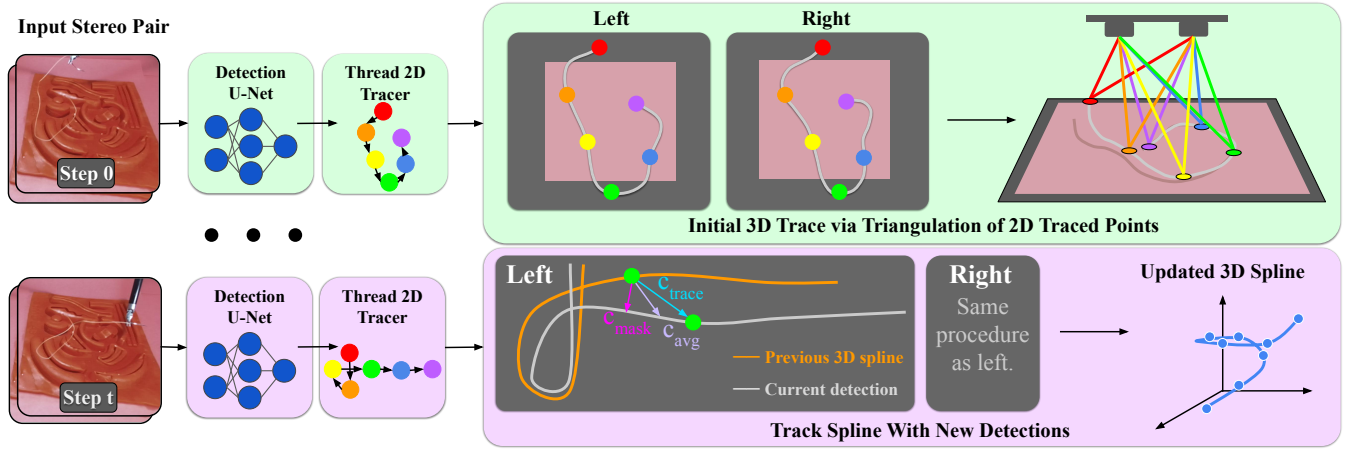


Fig. 2. **Overview of the first 4 modules: 2D Surgical Thread Detection, 2D Tracing, 3D Tracing, and 3D Tracking.** *Left:* For every stereo pair of images, we predict thread segmentation masks, then run a 2D tracer to compute the sequence of pixels along the thread. *Top right:* To initialize the 3D spline of the thread, we match points meeting both stereo image and tracer topology constraints and triangulate their positions in 3D. We initialize the 3D trace by fitting a 3D spline to these points. *Bottom right:* To update the 3D spline with new frames, we compute correction vectors in 2D as an average of vectors which push the projected 3D spline onto the new detection and push each projected 3D point to its corresponding point on the 2D trace. We then triangulate the correction vectors across both images and apply them to the 3D spline to perform an update.

## II. RELATED WORK

### A. Surgical Thread Detection

Detecting surgical thread from an RGB image has been previously explored in a number of different settings. Early approaches relying on analytic curvilinear detectors [4, 2] work well when the thread is isolated and clearly visible, but fail in realistic scenes with shadows and occlusions. Similarly, Joglekar *et al.* [5] assume that thread detections can be obtained from color segmentation; however, this may fail due to light glare, sensor noise, materials covering the thread (e.g., blood), and varying lighting conditions. Learning-based approaches generalize better to different backgrounds and lighting conditions, but require manual collection of large datasets. Lu *et al.* [6] train a U-Net [7] using semi-supervised learning leveraging hand-labeled images for supervision which are time consuming to obtain. We use a self-supervised data collection method that extracts labels autonomously using UV light [1], allowing the system to collect 10 labeled images per minute.

### B. Surgical Thread Reconstruction

Lu *et al.* [6] propose using a 3D graph to represent the triangulated 3D candidate thread points. The method then computes a minimum energy path through the graph and uses it as the 3D model of the thread. Joglekar *et al.* [5] propose using a minimum variation spline to represent the suture. This results in a smooth reconstruction with less tight curvature and yields a confidence value along the spline model which is useful to choose a grasp point along the thread. Both methods mentioned above fully reconstruct the model on each frame, ignoring prior frames, making them more susceptible to one-off missing or false detections. Padoy *et al.* [4] assume the 3D spline has been initialized in advance, and focus on tracking the spline across frames. However, this work assumes that the length of the thread is constant, which limits its applicability to certain applications like tail shortening or knot tying. Jackson *et al.* [2] propose an

approach to jointly trace and reconstruct a 3D spline from stereo images as well as a tracking method using pixel-space error minimization. However, their approach assumes a known initial tracing point, manually defined using a space mouse. In contrast, we leverage tracing of 2D splines to address missing or occluded parts of the thread and use an approach which does not rely on a user-defined seed point. Furthermore, our method tracks the spline across frames, increasing its robustness to noisy detections.

### C. Interactive Perception

Goldberg *et al.* [8] investigate how a robot can use active perception to recognize the shape of an object by moving a touch sensor to trace its contours. Bajcsy [9] defines *active perception* as the search for models and control strategies for perception which can vary depending on the sensor and the task goal, such as adjusting camera parameters [10] or moving a tactile sensor in response to haptic input [8].

Similarly, *interactive perception*, as explored by Bohg *et al.* [11], utilizes robot interactions to enhance perception. Interactive perception has been used in robotic manipulation to extract kinematic and dynamic models from physical interactions with the environment [12] and to improve the understanding of a scene in the presence of occlusions and perception uncertainty [11, 13, 14]. Murali *et al.* [15] leverage feedback from visual and tactile sensors to estimate the pose of partially occluded objects in cluttered environments. Danielczuk *et al.* [13] propose the mechanical search problem, where a robot retrieves an occluded target object from a cluttered bin through a series of targeted parallel jaw grasps, suction grasps, and pushes. Novkovic *et al.* [14] use a robot to move a camera and interact with the environment in order to find a hidden target cube in a pile of cubes, while Shivakumar *et al.* [16] use interactive perception to reduce perception uncertainty when untangling long cables.

In this work we propose an interactive perception-based approach to surgical suture tail-shortening. The robot tensions the thread to create a sharp angle between the taut

thread on the extraction side of the suture and the slack thread on the insertion side. This forces the thread into a linear configuration to facilitate perception. The robot then pulls the thread through the suture until the desired length of slack thread is detected at the tail.

### III. PROBLEM STATEMENT

Using stereo RGB images, we want to accurately track the state of a surgical thread and use these state estimates to automate the task of surgical tail-shortening.

#### A. Workspace and Assumptions

We define the workspace using a cartesian  $(x, y, z)$  coordinate system. The workspace consists of a bimanual dVRK robot [17]; a Simulab TSP-10 human organ phantom<sup>1</sup>); and a fixed ZEDm RGB stereo camera, which outputs images at 1280x720 pixel resolution. The camera is angled at the robot and phantom such that the whole reachable workspace of the robot is captured in the field of view. We work with undyed (beige) Polysorb<sup>TM</sup> surgical suture thread from Covidien. The sutures are of variable length between 10 and 40 cm, with 2-0 USP Size (0.35-0.399 mm in diameter) and are attached to a GS-21 needle or similar. The length, diameter and needle size of the suture are unknown to the algorithm.

We make the following assumptions:

- 1) The robot-to-camera transform is known.
- 2) During test time, the thread and phantom configurations lie within the training data distribution. However, their pose does not necessarily correspond exactly to any pose seen in training.

### IV. METHODS

We decompose the problem of thread modeling and autonomous robot suture tail-shortening into five modules:

- 1) *Learned 2D Surgical Thread Detection*: uses a convolutional neural network to segment the surgical thread in a physical mockup of a surgical environment. This module takes as input an RGB image of dimension 1280 x 720 and returns a pixel-wise probability mask of the thread's location.
- 2) *2D Surgical Thread Tracing*: given the detection probability masks with potential gaps in the detections and false positives, identifies the sequence of image points along the thread.
- 3) *3D Surgical Thread Tracing*: computes a 3D representation of the suture thread based on the traced thread. This algorithm takes the traces from 2 rectified stereo images as input and returns a 3D NURBS spline.
- 4) *3D Surgical Thread Tracking*: adapts the 3D spline model to the current view of the scene. This module takes the traces from the current pair of rectified stereo images as well as the previous 3D spline as input and outputs an updated 3D spline.
- 5) *Surgical Suture Tail-Shortening*: This module performs the surgical tail-shortening task using an interactive

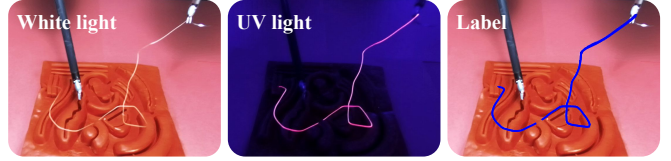


Fig. 3. **2D Surgical Thread Detection Data Collection.** The left image shows the dVRK gripper holding a suture thread under white light. The middle image depicts the same scene under UV light. The thread painted with UV fluorescent color lights up and can be segmented via color thresholding. The right image displays the extracted label used for training.

perception approach which leverages the 3D spline model computed by the previous modules.

An overview of how these modules are combined is shown in Figure 2.

#### A. Module 1: Learned 2D Surgical Thread Detection

We train a neural network to segment surgical thread from scenes using the self-supervised training approach proposed in Thananjeyan *et al.* [1]. To collect labels automatically, we paint the surgical thread with a UV-fluorescent paint. This paint is invisible under visible light but shines when illuminated with UV light. For each scene, the robot arms are moved to a new random position in the workspace, changing the thread configuration and RGB stereo images are recorded under both visible and UV light.

The label masks are extracted from the UV images using color segmentation. Train, validation, and test sets are split from disjoint subsets of scenes to ensure no cross-contamination of the sets from the same state. Using this self-supervised data collection technique, we are able to acquire 10 labeled images per minute (visible light stereo images with corresponding labels extracted under UV light). The image size of 1280 x 720 pixels is chosen to maximize the tradeoff between resolution (which aids in segmenting the thin thread) and inference speed (2.5 FPS on our test computer using an NVIDIA 2080 GPU).

We train a U-Net [18] to detect surgical thread from a single RGB image. We train our model on 1320 images of size 1280 x 720 for 400 epochs. We specifically choose not to upweight false negatives, as would be expected from the ratio of background pixels to thread pixels, as this yields predictions that are biased towards precision over recall. This is desirable because the 2D tracer is able to bridge missing thread detections but can get confused by false positives.

#### B. Module 2: 2D Surgical Thread Tracing

We adapt the analytic cable tracing method from Shivakumar *et al.* [3] to trace the path segments from the 2D thread detection masks. However, instead of generating all possible global paths, this work leverages heuristic scoring rules similar to those proposed by Viswanath *et al.* [19] and Keipour *et al.* [20] to generate a single global trace. In contrast to the learning-based method proposed in [19], which detects and traces cables simultaneously, we propose an analytical method. The method proposed in [20] is similar in the sense that it uses scoring functions that prioritize traces which cover more of the cable and have lesser changes in

<sup>1</sup><https://simulab.com/collections/suturing-skills-training/products/tissue-suture-pad>

angle. However, they model the thread as a chain of cylinders whereas we fit a 2D spline onto the traced detections to bridge gaps. The analytic thread tracer locally traces contiguous segments and greedily stitches them together, as described in Algorithm 1.

---

**Algorithm 1** 2D Surgical Thread Tracing Algorithm

---

**Require:**  $D \leftarrow$  pixelwise thread detection  
 $\text{mask} \leftarrow D > \text{thresh}_d$   
 $\text{mask} \leftarrow \text{mask} - (\text{conn components with area} < \text{thresh}_a)$   
 $\text{path\_segs} \leftarrow []$   
**while**  $\text{sum}(\text{mask}) > \text{thresh}_s$  **do**  
     $\text{start\_point} \leftarrow \arg \max(D)$   
     $\text{paths} \leftarrow \text{sgtm2tracer}(\text{mask}, \text{start\_point})$   
     $\text{best\_path} \leftarrow \arg \max_{p \in \text{paths}} \text{score}(\text{path})$   
    On mask, set points along best\_path to 0.  
    Append best\_path to path\_segs.  
**end while**  
**while**  $\text{length of path\_segs} > 1$  **do**  
    find  $i, j$  within path\_segs with lowest matching cost  
     $\text{new\_seg} \leftarrow \text{merge of path\_segs}[i] \text{ and path\_segs}[j]$   
    add new\_seg to path\_segs  
**end while**  
**return** path\_segs[0]

---

### C. Module 3: 3D Surgical Thread Tracing

As in prior work [2], we model the suturing thread as a 3D NURBS parametric curve. Instead of jointly tracing and reconstructing the thread, we use a dedicated 2D tracer to compute the sequence of thread pixels in both images before reconstructing the 3D thread model. The spline parameter  $t \in [0, 1]$  describes the normalized distance along the spline.

To start the 3D tracing method, a 2D NURBS spline defined by 32 control points is fitted to the traces in both images using a least squares approximation. The number of control points is chosen to allow a sufficient amount of flexibility to the spline so that it can approximate tight curves common in suturing thread.

Next, we triangulate these 2D splines into 3D to estimate the thread state. We therefore propose the following stereo matching approach: The left trace spline point  $p_i^L$  is located at spline parameter  $t_i^L$  along the spline and has pixel coordinates  $[u_i^L, v_i^L]$  for width and height respectively, starting from the top left corner. For each point along the left spline  $p_i^L$ , a corresponding point on the right spline  $p_{j(i)}^R$  is found which minimizes the difference between spline parameters  $t_i^L$  and  $t_{j(i)}^R$  and satisfies rectified stereo image properties. Specifically, the right image point should have the same vertical coordinate than the left image point except for a tolerance of up to  $\alpha = 5$  pixels (condition a).  $p_{j(i)}^R$  must be further left within the image than  $p_i^L$  (condition b). The right spline candidates must be further along the spline than the last matched right spline point (condition c).  $t_i^L$  and  $t_{j(i)}^R$  must be within a distance  $\beta = 0.05$  (condition d). For a

given value of  $i$ , we seek to solve

$$j(i) = \underset{j}{\operatorname{argmin}} |t_j^R - t_i^L|$$

such that a)  $|v_i^L - v_j^R| \leq \alpha$ , b)  $u_j^R \leq u_i^L$ , c)  $t_j^R > t_{j(i-1)}^R \forall i$ , d)  $|t_j^R - t_i^L| \leq \beta$ .

The matched points are then triangulated using the camera intrinsics to obtain their 3D position. A 3D NURBS spline model is then fitted to the triangulated points using least-squares optimization. The values for  $\alpha$ ,  $\beta$  and a rejection threshold for bad reconstructions were set empirically as a trade-off between reconstruction quality and number of discarded frames.

### D. Module 4: 3D Surgical Thread Tracking

Inspired by Jackson *et al.* [2], we compute 200 correction vectors to update the coordinates of the 3D spline control points between frames. The number of correction vectors was set as a trade-off between tracking accuracy and computation speed. The thread tracking computation time is under 2.5 FPS which is the frame rate of the 2D learned surgical thread detection module as described in Section IV-A. Instead of an energy minimization approach to compute correction vectors, we compute correction vectors using the 2D splines fitted on the current stereo traces. The 2D correction vectors are obtained as a sum of two vectors,  $c_{\text{mask}}$  and  $c_{\text{trace}}$ .  $c_{\text{mask}}$  is a vector in image space pointing towards the closest point on the prediction mask.  $c_{\text{trace}}$  matches the point of the 2D spline fitted on the 2D trace at parameter  $t$  with the point at parameter  $t$  of the projected 3D spline. The 2D correction vectors from both stereo images are triangulated to find 3D correction vectors. Both 3D correction vector terms are averaged to obtain the final set of correction vectors.

Using only the distance correction  $c_{\text{mask}}$ , the 3D spline tends to collapse as the segmentation mask of the thread does not constrain the 3D spline along the length of the thread. This is mitigated by the second correction vector,  $c_{\text{trace}}$ , which assigns a fully constrained pixel location to each point along the projected 3D spline.

Given the correction vectors, an updated set of control points is computed using the least square control point update described by Jackson *et al.* [2].

### E. Module 5: Surgical Suture Tail-Shortening

Initially, the thread passes through the phantom at one suture, with the needle held by one dVRK gripper. Excess thread of between 12-16 cm exists on the needle insertion side of the suture and needs to be pulled through while an unknown amount of thread has already been pulled through the suture. First, the robot uses interactive perception to estimate the needle extraction point by pulling the needle side of the thread taut. This is achieved by moving the needle upwards in positive  $z$  direction. The algorithm detects the needle end of the spline to be the one that has the highest  $z$ -coordinate. The system detects the taut segment of string by computing the tangent along the 3D spline and identifying a constant tangent direction segment. The angle between the



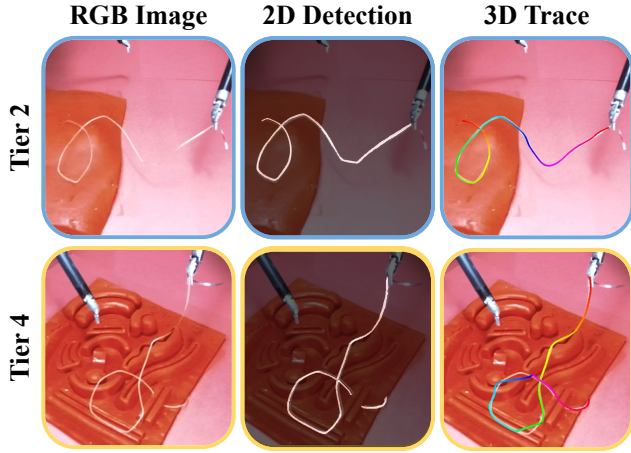


Fig. 4. **Example 2D Thread Detections and 3D Traces.** 2 example executions of the 3D thread tracing method. *Left* shows the left camera’s input RGB image, *Middle* shows the 2D thread detection prediction from the neural network, and *Right* shows the resulting reconstructed 3D spline reprojected into the camera image. The color indicates the path from red to orange.

TABLE I  
2D SURGICAL THREAD DETECTION RESULTS

	Tier 1	Tier 2	Tier 3	Tier 4	Overall
Recall (%)					
Color thresholding	32	32	14	40	30
<b>Learned segment. (ours)</b>	<b>85</b>	<b>79</b>	<b>80</b>	<b>86</b>	<b>83</b>
Precision (%)					
Color thresholding	24	27	5	10	17
<b>Learned segment. (ours)</b>	<b>93</b>	<b>93</b>	<b>87</b>	<b>90</b>	<b>91</b>
IoU (%)					
Color thresholding	16	17	4	9	12
<b>Learned segment. (ours)</b>	<b>80</b>	<b>75</b>	<b>72</b>	<b>79</b>	<b>77</b>

TABLE II  
3D SURGICAL THREAD TRACING RESULTS

	Tier 1	Tier 2	Tier 3	Tier 4	Overall
Mean Reproj. Err. (pix)	0.58	1.14	2.50	1.10	1.33
Max Reproj. Err. (pix)	13.34	20.61	62.16	43.17	62.16

taut segment tangent and the following thread tangent is computed. Pulling the thread upwards leads to a sharp angle in the thread which is identified as the needle extraction point. The 3D thread spline is split into a slack-side and a needle-side at the extraction point and the length of the slack-side thread is computed.

The robot conducts the actual tail-shortening by performing a horizontal motion away from the computed wound location (i.e. the extraction point). We continuously run the 3D thread tracking module during this motion, terminating when the thread tail is less than 3cm.

## V. EXPERIMENTS

### A. Modules 1-3: 2D Thread Detection and 2D & 3D Tracing

1) *Setup*: We test the first 3 modules by using the workspace described in Sec III-A. The experiments begin with the needle in the right end-effector and the tip of the thread going through the phantom. We collect test examples from 4 difficulty tiers:

- Tier 1**: No self-intersection in thread, reversed phantom.
- Tier 2**:  $\geq 1$  self-intersection in thread, reversed phantom.
- Tier 3**: No self-intersection in thread, phantom facing up.

**Tier 4**:  $\geq 1$  self-intersection in thread, phantom facing up. We collect and label stereo images for 5 scenes per tier for a total of 10 images per tier.

2) *Metrics*: In this experiment, we evaluate the IoU, precision, and recall metrics of the segmentation mask with respect to human-labeled ground truth segmentation masks. To evaluate the 3D model of the thread, we report the reprojection error between the human-labeled ground truth thread segmentations and the projection of the 3D model of the thread into both stereo images.

3) *Results and Failure Modes*: Results for 2D thread detection and 3D thread tracing are presented in Table I and Table II respectively. Example detection masks and reconstructions are shown in Figure 4. Comparison with the color thresholding baseline clearly shows that the learned method is able to detect thread in low contrast scenes and in the presence of light reflections on the phantom. Note that while detections can miss segments of highly difficult thread (recall of 83%), the precision of predictions is 90%. Our detection model shows slightly better recall and IoU performance in the more difficult Tier 4 with respect to Tier 2. This difference is due to a scene in Tier 2 in which the thread has a particularly low contrast with the background and is thus not detected. The discrepancy lies in the error bars of this experiment. Lu *et al.* [6] report an average IoU of 85% with a recall of 93%. While these are impressive results, all their scenes have suturing threads lying on the ground plane without any tools that can cast shadows or reflective configurations that make segmentation more challenging.

The 3D surgical thread traces have an overall mean error of 1.33 pixels, showing that the reconstruction approximates the spline well in general. The 3D tracing fails on parts of the thread in 3 scenes, resulting in the high maximum reprojection error. These errors are mainly due to reflective bright edges on the phantom which lead to erroneous detections and 3D traces. Joglekar *et al.* [5], report reprojection errors between mean 0.4 and 1.1 pixels on 10 real scenes with printed surgical backgrounds. These results seem comparable to ours even though performance remains highly dependant on the particular scenes, making results hard to compare objectively.

### B. Module 4: 3D Surgical Thread Tracking

1) *Setup*: Using the workspace setup described in Section III-A, we thread the needle through the phantom and place it in the right gripper of the dVRK. We evaluate the thread tracking system on two trajectories: the “no loop” trajectory, a line in the image plane, and the “one loop”, an elliptical track above the phantom. The first trajectory avoids any occlusion or self-intersection of the thread, while the second incurs a challenging, self-crossing configuration.

2) *Metrics*: We report the same metrics as in the single-frame 3D tracing (Section V-A.2) experiments. We manually label ground-truth segmentation masks in stereo images taken every 1 cm along the trajectory and evaluate the 3D thread model against them. This leads to 10 evaluation frames for the “No loop” trajectory and 9 for the “One loop.”

TABLE III  
3D SURGICAL THREAD TRACKING RESULTS

	Mean Reproj. Err.	Max Reproj. Err.
No loop		
No tracking	<b>0.30 pix</b>	9.22 pix
With tracking	0.39 pix	<b>5.39 pix</b>
One loop		
No tracking	5.37 pix	94.92 pix
With tracking	<b>1.28 pix</b>	<b>16.26 pix</b>

*No tracking* refers to computing a new 3D trace for every frame.

*With tracking* refers to using Module 4 and leads to a significantly better mean reprojection error in the more difficult “One loop” case.

3) *Results and Failure Modes*: The results in Table III suggest that the method is able to track the thread reliably in both configurations. The “One loop” trajectory sees higher reprojection errors, as it presents a more challenging thread configuration for the tracer. The full tracking pipeline achieves a mean reprojection error of 0.39 pixels on the intersection-free trajectory and a 1.28 pixels on the loop-forming trajectory. Padoy *et al.* [4], report a mean reprojection error of 1.21 pixels. However, they use only short threads in configurations which present no self-intersections.

### C. Module 5: Surgical Suture Tail-Shortening

1) *Setup*: We additionally test automated suture tail shortening using the workspace setup described in Section III-A, with the needle threaded fully through the reversed phantom and held by the right end-effector. The tail of the thread is then placed arbitrarily in the workspace such that the entire suture thread is within the view of both the right and left stereo cameras.

2) *Metrics*: We define a successful tail-shortening maneuver as a termination in which the final tail length lies within 1 cm of the desired value of 3 cm. We report the success rate of our pipeline on the tail-shortening task, as well as the mean absolute error between the achieved and desired length and the average time to completion for this task.

3) *Results and Failure Modes*: The proposed method achieves 18 successes out of 20 trials with a mean absolute tail error of 0.53cm. The mean time to completion is 106.8 seconds. The method achieves a success rate of 90%, indicating that our pipeline provides high-confidence 3D traces using interactive perception. The main failure case occurs during the 3D tracing of the spline due to false detections on the human organ phantom.

## VI. LIMITATIONS & FUTURE WORK

The primary limitation is the uncertainty about the ability of the learned 2D surgical thread detection to generalize to new thread or phantoms, which will be addressed in future work. Also, the learned thread segmentation method remains vulnerable to false positive detections of light reflections from the edges of the human organ phantom. This could be mitigated in future work by adapting the lighting setup of the workspace.

## ACKNOWLEDGEMENT

This research was performed at the AUTOLAB at UC Berkeley in affiliation with the Berkeley AI Research (BAIR) Lab, and the CITRIS

“People and Robots” (CPAR) Initiative. The da Vinci Research Kit is supported by the National Science Foundation, via the National Robotics Initiative (NRI), as part of the collaborative research project “Software Framework for Research in Semi-Autonomous Teleoperation” between The Johns Hopkins University (IIS1637789), Worcester Polytechnic Institute (IIS 1637759), and the University of Washington (IIS 1637444).

## REFERENCES

- [1] B. Thananjeyan, J. Kerr, H. Huang, J. E. Gonzalez, and K. Goldberg, “All you need is luv: Unsupervised collection of labeled images using uv-fluorescent markings,” vol. 2022-October, IEEE, 2022.
- [2] R. C. Jackson, R. Yuan, D. L. Chow, W. S. Newman, and M. C. Çavuşoğlu, “Real-time visual tracking of dynamic surgical suture threads,” *IEEE T-ASE*, vol. 15, 2018.
- [3] K. Shivakumar *et al.*, “Sgtm 2.0: Autonomously untangling long cables using interactive perception,” Sep. 2022.
- [4] N. Padoy and G. D. Hager, “3d thread tracking for robotic assistance in tele-surgery,” IEEE, Dec. 2011.
- [5] N. Joglekar, F. Liu, R. Orosco, and M. Yip, “Suture thread spline reconstruction from endoscopic images for robotic surgery with reliability-driven keypoint detection,” 2022.
- [6] B. Lu *et al.*, “Toward image-guided automated suture grasping under complex environments: A learning-enabled and optimization-based holistic framework,” *IEEE T-ASE*, vol. 19, 2022.
- [7] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *MICCAI 2015, Munich, Germany, 2015*.
- [8] K. Y. Goldberg and R. Bajcsy, “Active touch and robot perception,” *Cognition and Brain Theory*, vol. 7 (2), 1984.
- [9] R. Bajcsy, “Active perception,” *IEEE*, vol. 76 (8), 1988.
- [10] R. Bajcsy, Y. Aloimonos, and J. K. Tsotsos, “Revisiting active perception,” *Autonomous Robots*, vol. 42(2), 2018.
- [11] J. Bohg *et al.*, “Interactive perception: Leveraging action in perception and perception in action,” *IEEE T-RO*, vol. 33, no. 6, 2017.
- [12] R. Martin-Martin and O. Brock, “Coupled recursive estimation for online interactive perception of articulated objects,” *IJRR*, vol. 41 (8), 2022.
- [13] M. Danielczuk *et al.*, “Mechanical search: Multi-step retrieval of a target object occluded by clutter,” in *ICRA*, IEEE, 2019.
- [14] T. Novkovic, R. Pautrat, F. Furrer, M. Breyer, R. Siegwart, and J. Nieto, “Object finding in cluttered scenes using interactive perception,” in *ICRA*, IEEE, 2020.
- [15] P. K. Murali, A. Dutta, M. Gentner, E. Burdet, R. Dahiya, and M. Kaboli, “Active visuo-tactile interactive robotic perception for accurate object pose estimation in dense clutter,” *IEEE RAL*, vol. 7, no. 2, 2022.
- [16] K. Shivakumar *et al.*, “Sgtm 2.0: Autonomously untangling long cables using interactive perception,” *arXiv preprint arXiv:2209.13706*, 2022.
- [17] P. Kazanzides, Z. Chen, A. Deguet, G. S. Fischer, R. H. Taylor, and S. P. Dimaio, “An open-source research kit for the da vinci® surgical system,” *ICRA*, 2014.
- [18] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” vol. 9351, Springer Verlag, 2015.
- [19] V. Viswanath *et al.*, *Learning to trace and untangle semi-planar knots (tusk)*, 2023. arXiv: 2303.08975 [cs.RO].
- [20] A. Keipour, M. Bandari, and S. Schaal, “Deformable one-dimensional object detection for routing and manipulation,” *CoRR*, vol. abs/2201.06775, 2022. arXiv: 2201.06775.

## VII. APPENDIX

### A. Workspace

Figure 5, shows an image of the workspace used for the experiments. It contains two dVRK robot arms, a ZEDm stereo camera, a red tissue phantom placed on a red background and a surgical suture composed of a needle and white thread.

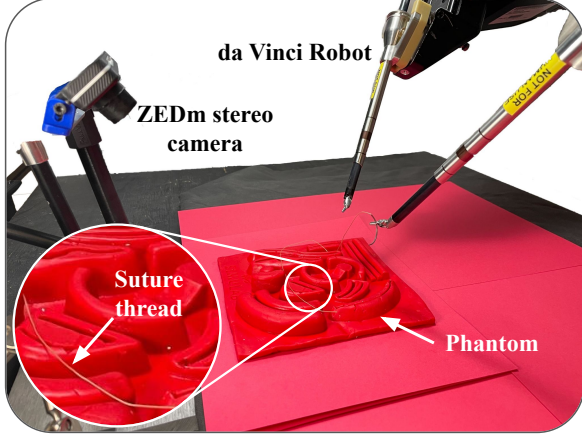


Fig. 5. **Workspace.** The da Vinci Research Kit with two PSRs (Patient Side Manipulators) holds a thread above the phantom tissue placed on a red background. The suture thread is thin, making it difficult to perceive. The ZEDm stereo camera pair faces the workspace on the opposite side to the robot at a steep downwards angle.

### B. Module 4 Experiment Extension: 3D Surgical Thread Tracking

We show an ablation study for the experiment described in Module 3 V-B. It shows the value of both components of the 3D surgical thread tracking correction vector.

1) *Setup*: The setup is the same as in the main experiment described in Section V-B. We evaluate the performance on the same trajectories we call ‘No Loop’ and ‘One Loop’. Both trajectories are represented in Figure 6.

2) *Metrics*: We use the same metrics as in the main experiment, i.e. mean and maximum reprojection error in pixels. Also, we add the ground truth coverage metric which measure how much of the ground truth segmentation mask is covered by the reprojected 3D surgical thread model spline.

3) *Results and Failure Modes*: We present three ablations of the 3D surgical thread tracking pipeline and compare them to the full pipeline performance. Results can be seen in Table IV. *No tracking* refers to performing a new 3D reconstruction every frame. It can be seen that especially in the more difficult ‘One Loop’ trajectory, not using the knowledge from prior frames leads to higher reprojection errors. The *Mask track* ablation refers to tracking with only the correction vector  $c_{\text{mask}}$ . While it leads to lower reprojection errors, looking at the median ground truth coverage shows that the spline actually shortens along the detection mask as the spline points are not fixed along this degree of freedom. This is clearly visible when looking at Figure 7. The *Trace track* ablation refers to tracking with only the correction vector  $c_{\text{trace}}$ . This correction vector addresses the spline collapse which arises when only using  $c_{\text{mask}}$  as it fully constrains

the 3D position of the spline evaluation points. However, it can be seen that using only this term leads to higher reprojection errors as the two traces on the left and right image can be of different length due to perspective effects, thus leading to a mismatch of correction vectors between both images. This imprecision in turn is addressed by the first correction vector. Using the *Full tracking* method leads to the best mean reprojection error while achieving a high ground truth coverage value which shows that it is reconstructing the whole thread.

TABLE IV  
3D SURGICAL THREAD TRACKING RESULTS

	Mean Reproj. Err.	Max Reproj. Err.	GT Coverage
No loop			
No tracking	0.30 pix	9.22 pix	93.22 %
Mask track.	<b>0.20 pix</b>	<b>4.12 pix</b>	88.56 %
Trace track.	0.77 pix	7.00 pix	<b>96.37 %</b>
Full tracking	0.39 pix	5.39 pix	96.10 %
One loop			
No tracking	5.37 pix	94.92 pix	<b>95.90 %</b>
Mask track.	<b>0.15 pix</b>	<b>5.0 pix</b>	82.79 %
Trace track.	1.91 pix	17.03 pix	94.53 %
Full tracking	1.28 pix	16.26 pix	95.26 %

*GT Coverage* is the ground truth coverage metric which measures how much of the ground truth mask overlaps with the current reprojected thread model in pixel space. *No tracking* refers to performing a new 3D reconstruction for every frame. The *Mask track* ablation refers to tracking with only the correction vector  $c_{\text{mask}}$ . The *Trace track* ablation refers to tracking with only the correction vector  $c_{\text{trace}}$ . Using the *Full tracking* method leads to the best mean reprojection error while reconstructing the whole thread.

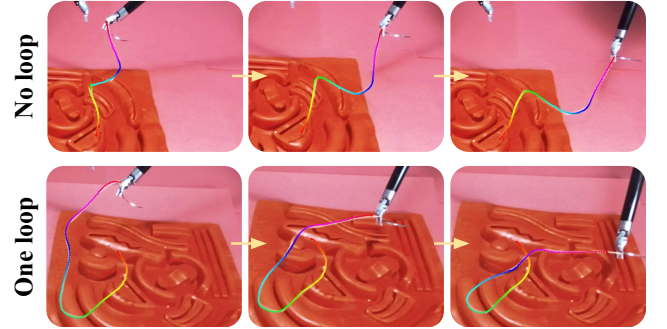


Fig. 6. **Tracking Experiment Trajectories**: these panels illustrate the two trajectories used for computing tracking performance. The rainbow colored line shows the projected 3D spline while the colors depicts the sequence of the spline. *Top* shows the first “No loop” trajectory where the thread has no self-intersections (moving from left to right). *Bottom* shows the second “One loop” trajectory where a self-intersecting thread configuration is created.



Fig. 7. **Tracking Ablation Study**: This figure shows a scene from the *One loop* trajectory with the 3D spline reprojected into the left camera image computed with four tracking ablations for which experimental results are in Table IV. Image a shows the spline obtained through reconstruction only, image b shows the spline obtained when only the mask error  $c_{\text{mask}}$  is used for tracking. Image c shows the spline obtained when only the trace error  $c_{\text{trace}}$  is used for tracking and image d shows the result obtained with the full tracking method. It can be seen in image b that the spline collapses along the detection mask when the trace correction vector is not used as well.