Fast and Reliable Autonomous Surgical Debridement with Cable-Driven Robots Using a Two-Phase Calibration Procedure

Daniel Seita, Sanjay Krishnan, Roy Fox, Stephen McKinley, John Canny, Ken Goldberg

Abstract—Surgical debridement is the process of removing dead or damaged tissue to allow the remaining parts to heal. Automating this procedure could reduce surgical fatigue and facilitate teleoperation, but doing so is challenging for Robotic Surgical Assistants (RSAs) such as the da Vinci Research Kit (dVRK) due to inherent non-linearities in cable-driven systems. Consequently, we propose and evaluate a two-phase calibration process. In Phase I, the robot performs a set of open-loop trajectories to obtain abundant, cheap but coarse data of camera pixels and internal robot joint values, and learns a nonlinear transformation. In Phase II, the robot uses Phase I to move systematically to target points in a printed array. Each time, a human operator manually adjusts the end-effector position by direct contact (not through teleoperation), resulting in a small, high-quality dataset. Experiments suggest that without calibration, position errors are 4.55mm. Phase I alone can reduce average error to 2.14mm, but the combination of Phase I and Phase II reduces average error to 1.08mm. We apply this combination to an approximation of debridement with randomly oriented raisins and pumpkin seeds. Using an endoscopic stereo camera with standard edge detection, experiments with 120 trials achieved success rates of 91.7% to 99.2%, slightly exceeding prior results (89.4%) and more than 2.1x faster, decreasing the time per fragment from 15.8 seconds to 7.3 seconds. Source code, data, and videos are available at https://sites.google.com/view/calib-icra/.

I. INTRODUCTION

Surgical robots [13], such as Intuitive Surgical’s da Vinci [12, 14], are widely used in laparoscopic procedures worldwide [1, 4, 24]. Existing surgical robotics platforms focus on teleoperation, often through a master-slave interface where the surgeon fully controls the motions of the robot’s tools. Such interfaces are tedious to use in long or repetitive procedures, and recent work has highlighted several opportunities for autonomous execution of surgical subtasks [37] including debridement [15], suturing [25, 32, 34], and palpation for tumor detection [6].

For limited autonomy to be viable, the execution must approach the reliability and speed of a human operator, which is challenging in existing platforms. The da Vinci and other Robotic Surgical Assistants (RSAs) such as Applied Dexterity’s Raven II [9] are cable-driven and designed to be compliant as not to damage anatomy. Cable-driven robots are susceptible to issues such as cable-stretch, hysteresis, and other problems described in [8, 22, 23, 30].

Consider the dVRK surgical robot [12, 14]. The robotic arm consists of a 3 DoF manipulator to which a modular 4 DoF instrument is attached. The arm manipulates the attached instrument about a mechanically-fixed fulcrum point called the remote center that is invariant to the configuration of the joints. The tool is commanded to some position and orientation from a camera and its internal joint sensors. After data cleaning, we train a Deep Neural Network (DNN) to map from world position and robot orientation (extracted from the images) to tool positions. Phase II: we apply the DNN on a calibration grid and a human directly corrects the resulting end-effector positions. The residuals are stored and used as training data for Random Forests (RFs). We apply the combined DNN and RF pipeline to debridement.
shown in Figure 1. In Phase I, we automatically explore trajectories in the workspace with random targets, using red tape tied to the end-effector to track its location. We form training data consisting of end-effector world positions and rotations as input, and tool positions as output. As detailed in Section III using rotations as input is useful in tasks such as debridement where we can determine the orientation of fragments in advance using OpenCV ellipse fitting on the camera image of the workspace. A Deep Neural Network (DNN) is trained on this data. In Phase II, to correct for residual errors, we send the dVRK’s end-effector to each target location in a printed calibration grid and then have a human directly correct the position. We train a Random Forest (RF) to predict these residuals.

Our approach is based on four prior works on automated surgical debridement and/or calibration [15, 22, 26, 30], and improves on them with four novel contributions:

1) An alternative calibration method compared to [23, 30], using automatic data collection with DNNs and manual movements with RFs. We evaluate three calibration algorithms (rigid body, DNN, DNN+RF) and test various DNN settings; see Sections V and VI.

2) We use colored tape for tracking, which is cheaper than the external 3D tracking systems in [15, 22].

3) We use more challenging fragments: pumpkin seeds and raisins. Sizes are shown in Table I the fragments from [15, 22] were substantially larger (about 10mm each in width, length, and thickness) and made of foam rubber to be forgiving of gripper orientation error. Similarly, fragments from [26] were glued onto a highly elastic surface to tolerate errors. In contrast, pumpkin seeds easily slide out of the gripper.

4) Motivated by the observation that autonomous execution was typically 2-3x slower than a human [15], we optimize speed without decreasing success rate, and obtain 2.1x speedups over the fastest prior work [22].

II. RELATED WORK

We use the da Vinci Research Kit (dVRK) [12, 14] as our RSA, which is a research platform based on Intuitive Surgical’s da Vinci surgical system [3] and which has been frequently used in surgical robotics research [18, 26, 34, 36]. Calibration is critical for the dVRK and other robots to reliably perform fine-grained manipulation tasks. The majority of work in calibration of kinematic parameters, reviewed in [11], has focused on modeling linearities in kinematic chains. Examples include work on autonomously calibrating both a stereo vision system and robot manipulators [5] and jointly calibrating multiple cameras [19]. Since the dVRK has inherent non-linearities, we must incorporate non-linear models in our calibration procedure.

To account for actuator imprecision, one could use visual servoing [17, 20] as part of an automation pipeline. For instance, [21] trained a large convolutional neural network to predict the probability that specific motions would lead to successful grasps. To servo, they sampled a set of candidate commands and evaluated the likelihood of grasp success based on their trained network. In [19], the authors applied visual servoing of the dVRK for needle insertion and picking tasks, but required large amounts of task-specific expert demonstrations to train a visual servoing policy. Learning a compensation rather than a policy is far more sample efficient.

Surgical debridement [2, 7, 28] is the process of removing dead or damaged tissue to allow the remaining part to heal. Automated surgical debridement was first demonstrated in [15]. Using the Raven II, the authors debrided at a rate of 91.8 and 60.3 seconds per fragment for the autonomous single-arm and two-arm cases, respectively, and had success rates of 89.4% for the single-arm. Mahler et al. [22] used calibration to avoid replanning steps and sped up debridement by 3.8x with the two-arm setup to get a rate of 15.8 seconds per fragment with comparable success rates, but they did not report benchmarks on the slower one-arm scenario. Murali et al. [26] used more realistic fragments but had them glued onto highly elastic tissue to tolerate several millimeters of calibration error, and they did not achieve notable speed-ups over [22] nor did they discuss their calibration procedure.

We use a different debridement model by using small fragments with inclusion sizes that are close to the maximum width of the gripper (10mm). One of our fragment types (pumpkin seeds) is smooth and lacks friction with our workstation surface. Moreover, inspired by [15]’s timing comparisons involving human execution, we focus on optimizing speed jointly with accuracy. These factors lead to higher required precision in control.

III. PROBLEM STATEMENT

A. Definitions and Notation

Throughout the paper, we refer to the following:

- The base or world position \( x_b = (b_x, b_y, b_z) \in \mathbb{R}^3 \) is the end-effector position with respect to the base frame. These are determined from left and right camera pixels and their disparity.
- The tool position \( x_t = (t_x, t_y, t_z) \in \mathbb{R}^3 \) is the end-effector position with respect to the tool frame.
- \( \phi = (\phi_y, \phi_p, \phi_r) \in \mathbb{R}^3 \) is the yaw, pitch, and roll of the end-effector in the base frame. See Figure 2 for
two separate lenses and is located 7.5 inches above the workspace.

Fig. 3: Overall view of our setup. We use the dVRK arm labeled above for our experiments. The endoscopic stereo camera contains two separate lenses and is located 7.5 inches above the 7.5cm x 7.5cm workspace.

B. Assumptions

Setup of dVRK. The setup is shown in Figure 3. We use one dVRK arm with a gripper end-effector, called a large needle driver (see Figure 2), that can be opened up to 75°, or a gripper width of 10mm. We assume access only to the perception available in current da Vinci deployments: a 1920x1080 stereo endoscopic camera. These are located above a printed grid with black circles overlaid on our silicone tissue. We do not assume access to a motion capture system as in [22] or force sensing as in [29]. All grasps during debridement are pinch grasps. Finally, we assume that the axes are oriented for our dVRK instrument (the orientation is non-standard).

To compare calibration algorithms, we use a printed grid with black circles overlaid on our silicone tissue. For each circle, we get the left and right camera pixels of its center via OpenCV contour detection, and with the disparity, we can determine the location of the end-effector in pixel space. The distance in pixels between the circle center and the end-effector tip (i.e., its lowest point) is the calibration error, and the goal is to minimize this quantity. See Figure 5 for details.

Debridement. Our objective is to debride as many fragments as possible while minimizing the time per fragment. A fragment is successfully debrided when the dVRK is able to grasp it and move it above a receptacle outside of the workspace without collisions with other fragments, while using both grippers. We defer a detailed discussion on the debridement setup to Section VI-A.

IV. METHODOLOGY

Our method has two data collection phases. The first phase obtains a large amount of cheap, automatically generated coarse data. The second phase involves human intervention to generate a small amount of precious, high quality data.

A. Phase I: Automatic Collection of Coarse Data

The goal in this phase is to collect the dataset

$$\mathcal{X}_{DNN} = \{(x_b, \phi)^{(1)}, x_e^{(1)}, \ldots, (x_b, \phi)^{(N)}, x_e^{(N)}\}, \quad (1)$$

where each $$(x_b, \phi)^{(i)}$$ is the concatenation of a base position and a rotation vector and $$x_e^{(i)}$$ is a tool position. The rotation is part of the input, which follows from our rotation assumptions in Section III since the $$\phi_y$$ values (and consequently, $$\phi_p$$ and $$\phi_r$$) are determined using OpenCV.

To build $$\mathcal{X}_{DNN}$$, we automatically execute trajectories of the dVRK. For each trajectory, its starting location is one of the four workspace corners. Then, its target position $$(t_x, t_y, t_z)$$ is randomly chosen, though each coordinate is discretized into five choices, -90°, -45°, 0°, 45°, and 90°. Each value corresponds to a distinct training data for non-linear regressors, which we describe in Section IV-B.1 During debridement applications, each elliptical angle is mapped to the nearest yaw value, and then pitch and roll are determined from that (see Figure 4).

C. Evaluation

Calibration. To compare calibration algorithms, we use a printed grid with black circles overlaid on our silicone tissue. For each circle, we get the left and right camera pixels of its center via OpenCV contour detection, and with the disparity, we can determine $$x_b$$. The calibration provides us with $$x_t$$ and the dVRK’s closed end-effector moves to $$x_e$$. A human then identifies the location of the end-effector in pixel space. The distance in pixels between the circle center and the end-effector tip (i.e., its lowest point) is the calibration error, and the goal is to minimize this quantity. See Figure 5 for details.

Reliability of debridement depends substantially on having the two gripper tips touch the surface simultaneously while surrounding the target. We observed that a small range of pitch and roll values enables these tips to touch the surface in this manner. The yaw values, however, need the full range $$[-90^\circ, 90^\circ]$$ so that the gripper can reliably grasp arbitrarily-oriented fragments in their minor elliptical axis. Thus, we make the simplifying assumption that the pitch and roll can be known conditioned on a yaw.

We additionally assume that before debridement, we can use OpenCV to fit an ellipse to all fragments to get an angle of orientation along the major axis. This angle can be translated into a yaw value for the dVRK. We furthermore discretize the yaw values into five choices, -90°, -45°, 0°, 45°, and 90°. Each value corresponds to a distinct training data for non-linear regressors, which we describe in Section IV-B.1 During debridement applications, each elliptical angle is mapped to the nearest yaw value, and then pitch and roll are determined from that (see Figure 4).

| Fragment | Length | Width | Thickness |
|----------|--------|-------|-----------|
| Pumpkin  | 12.4 ± 0.8 | 6.8 ± 0.3 | 2.4 ± 0.3 |
| Raisins  | 12.3 ± 1.5 | 5.9 ± 1.1 | 4.2 ± 0.5 |

TABLE I: FRAGMENT DIMENSIONS. In millimeters, the average length (longer axis), width (shorter axis), and height of ten instances of each type, ± one standard deviation.
from view. As we show in Section V-A, we need to perform workspace, and the dVRK's wrist can block the red tape
points, so before executing the trajectories we apply red

\( f \) train a Deep Neural Network

\( \phi \) the same robot position, with

for

\( X \) \((892,160)) and the

center of the target circle (colored blue, pixel location (888,192)).

Evaluating calibration methods. Left: the end-effector after

discretization: \(-90^\circ, -45^\circ, 0^\circ, 45^\circ, 90^\circ\). Each yields hand-tuned pitch and roll values that we observed empirically allowed both tips to touch the workspace simultaneously when lowering the end-effector to the surface. Above, the pumpkin seed is rotated counter-clockwise across consecutive images, and the resulting yaw, along with our chosen pitch and roll values, are listed below (see Figure 2). For all five, moving downwards (adjusting \( t_z \) only) a few millimeters is sufficient to get the gripper in a spot where it can reliably grasp the seed.

In our second phase, we collect five datasets: \( \{x_1, b_1, y_1, \phi_1, r_1\} \) to map

\( \phi \) for each \( \phi \in \{-90^\circ, -45^\circ, 0^\circ, 45^\circ, 90^\circ\} \), where each \( x_i \) and \( \epsilon \) refer to a predicted end-effector tool position and a residual error vector, respectively.

The rationale for this second phase is that, after obtaining

\( f_{DNN} \), the calibration may still be imperfect due to noisy data (e.g. slight perception mismatches when locating the red tape among the two cameras). Since our debridement scenarios require fine-grained calibration, as in [22, 30], we model the residuals using a non-linear function.

Figure 6 visualizes a step in this process. We use the same printed calibration grid of black circles, with the centers known via OpenCV contour detection. From our Section III assumptions, we have five discretized yaws and thus five rotation vectors \( \phi \) (see Figure 4). For each \( \phi \), we command the dVRK’s closed end-effector to touch the center of each circle in the grid while maintaining that rotation. All rotations use the same calibration function \( f_{DNN} \) to get \( x_i \) from \( (x_i, \phi) \).

In this process, the dVRK’s end-effector tip may not reach the exact center of the circles, as judged by a human. Consequently, for each circle, once the dVRK’s end-effector has stopped moving, a human directly (not through teleoperation) adjusts it to the center of the circle. For the ith circle of yaw \( \phi_y \), we record the world-space transformation and store it in the data \( \chi_{RF}^{(\phi_y)} \) as \( \epsilon \).

With these datasets, it is then possible to train five random forests \( f_{RF}^{(\phi_y)} : \mathbb{R}^3 \rightarrow \mathbb{R}^3 \), each of which maps a predicted \( (t_x, t_y, t_z) \) from \( f_{DNN} \) to a residual vector \( (\epsilon_x, \epsilon_y, \epsilon_z) \).

For a given fragment in the workspace, we can detect \( (b_x, b_y, b_z) \) and its orientation angle. The orientation angle then maps to the nearest yaw value in

\[
\chi_{RF}^{(\phi_y)} = \{(x_i^{(1)}, \epsilon^{(1)}), \ldots, (x_i^{(N)}, \epsilon^{(N)})\},
\]
\(-90^\circ, -45^\circ, 0^\circ, 45^\circ, 90^\circ\), and thus the pre-computed pitch and roll values. The result from calibration is the mapping from \((x_b, \phi) = (b_x, b_y, b_z, \phi_y, \phi_y, \phi_r)\) to \(x_t\) as:

\[
f_{\text{DNN}}(x_b, \phi) + f_{\text{RF}}^{(\phi_y)}(f_{\text{DNN}}(x_b, \phi)) = x_t. \tag{3}
\]

\section{V. Calibration Experiments}

\subsection{A. Training a Deep Neural Network}

We ran the dVRK arm for 57 trajectories, which collectively provided 5256 initial data points \(\{(x_b, \phi), x_t\}_{i=1}^{5256}\). Different trajectories have different lengths, and they contribute to this dataset in proportion to their length — the longer the trajectory, the more data. We cleaned the data by keeping those points where the left and right cameras both detected a pixel center (of the red tape) corresponding to the same workspace location, resulting in 1939 examples.

To determine the architecture for \(f_{\text{DNN}}\), we ran a preliminary search over several hyperparameter settings for fully connected networks, and settled on three hidden layers with 300 units each, and with the ReLU activation \([27]\). After running experiments with this network, we later ran more extensive hyperparameter searches by varying the number of layers (1-4), number of hidden units in each layer (30 each or 300 each) and the nonlinearity (ReLU, sigmoid, or tanh). All training used Adam \([16]\] on the squared \(L_2\) loss.

Figure 7 shows the performance of the top 12 of these 24 hyperparameter settings, reported as average squared loss. After running experiments with this network, we later ran more extensive hyperparameter searches by varying the number of layers (1-4), number of hidden units in each layer (30 each or 300 each) and the nonlinearity (ReLU, sigmoid, or tanh). All training used Adam \([16]\] on the squared \(L_2\) loss.

\section{VI. Debridement Experiments

\subsection{A. Debridement Setup and Failure Modes}

We benchmark the calibration performance of various methods by following the procedure in Section III-C: commanding the dVRK end-effector to reach the center of each circle in the calibration grid and measuring how far the result is from the true center. We test the following:

- **Rigid Body Transformation (RBT).** Following \([22]\), we initially test a rigid body transformation where we map the base to tool positions by means of a rotation and a translation. We determine the rigid body by solving an orthogonally constrained least squares problem with singular value decomposition \([10]\).

- **Deep Neural Network (DNN).** Using only the function \(f_{\text{DNN}}\) (see Section IV-A).

- **DNN and Random Forest (DNN+RF).** Using a DNN and then a Random Forest, i.e. Equation 3. We used 100 trees for each, with no specified depth, the default behavior of scikit-learn \([31]\).

We test using all yaws in our discretization. In addition, we add a sixth “yaw setting” where, for each circle, we randomly choose a yaw value in \([-90, 90]\], so that the dVRK maps each random yaw into the closest value in the discretization.

Table II lists the errors (in pixels) based on 35 circle centers of the above algorithms. The RBT exhibits substantial calibration errors, which is likely due to coarse data. The average pixel error among the six settings is 51.45. The DNN performs substantially better with improvements across the entire set of \(\phi_y\) choices, roughly halving the error in pixel space (24.13 pixels). Adding Random Forests halves the errors again (12.20 pixels). The standard error of the mean implies that the differences are statistically significant.

We measured 1mm in the workspace in terms of pixels and found the correspondence to be roughly 1mm to 11.3 pixels. This is distance in \((t_x, t_y)\) only, not including \(t_z\) since the vertical distance can’t be captured using a single camera image. The RBT, DNN, and DNN+RF workspace errors are roughly 4.55mm, 2.14mm, and 1.08mm, respectively.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|}
\hline
Mapping & Yaw \(\phi_y\) & Mean ± SE Med. & (Min, Max) \\
\hline
RBT & -90 & 39.9 ± 1.8 & 30.8 & (26.4, 60.0) \\
RBT & -45 & 53.6 ± 1.5 & 50.7 & (43.2, 78.1) \\
RBT & 0 & 52.0 ± 2.6 & 53.7 & (29.0, 85.7) \\
RBT & 45 & 72.0 ± 5.3 & 66.4 & (32.0, 167.2) \\
RBT & 90 & 44.1 ± 1.4 & 43.1 & (29.1, 67.2) \\
RBT & [-90,90] & 47.1 ± 2.1 & 48.3 & (19.7, 75.1) \\
DNN & -90 & 26.7 ± 1.5 & 22.0 & (12.7, 47.8) \\
DNN & -45 & 28.3 ± 0.8 & 29.4 & (18.3, 38.9) \\
DNN & 0 & 22.7 ± 1.0 & 22.6 & (11.4, 39.4) \\
DNN & 45 & 21.5 ± 1.3 & 21.0 & (7.0, 39.1) \\
DNN & 90 & 22.5 ± 2.3 & 25.0 & (2.2, 53.3) \\
DNN & [-90,90] & 23.1 ± 1.6 & 21.0 & (3.6, 48.8) \\
DNN+RF & -90 & 10.5 ± 1.0 & 8.0 & (1.0, 25.0) \\
DNN+RF & -45 & 14.2 ± 1.1 & 14.1 & (1.0, 29.1) \\
DNN+RF & 0 & 12.8 ± 1.2 & 12.0 & (1.4, 27.2) \\
DNN+RF & 45 & 12.1 ± 1.2 & 11.6 & (2.0, 30.0) \\
DNN+RF & 90 & 8.9 ± 1.0 & 8.4 & (1.0, 28.0) \\
DNN+RF & [-90,90] & 14.7 ± 1.4 & 13.0 & (2.2, 34.9) \\
\hline
\end{tabular}
\caption{Calibration Error Results (in Pixels). The first six are RBT estimators. The next six combine the DNN with RFs. These are conditioned on yaw values. For each mapping, we evaluated its accuracy by measuring the pixel-wise error distance from the dVRK’s closed image.
\end{table}
Fig. 7: Average squared $L_2$ loss of different algorithms on the data from Phase I (see Section IV-A), each taken over a 10-fold cross validation with one standard deviation shown with an error bar. Left: Random Forest regressors. Key: $t_A dB$ where $A$ is the number of trees and $B$ is the depth (or none in the case of $N$, which is the default setting of scikit-learn [31]). Middle: linear regression. Key: $E A$ and $Q$ means angles are Euler angles or quaternions, respectively. Right: neural networks (top 12 out of 24). Key: $uA_hB_C$ where $A$ is the number of units in each hidden layer, $B$ is the number of hidden layers, and $C$ is the activation.

Fig. 8: Properties of pumpkin seeds during debridement. These seeds slide and rotate on the paper surface. Type A errors frequently result in the seed getting pushed out of the gripper’s control (first two images). This can also happen even if calibration is accurate and the dVRK ends up in a reasonable location. The third image shows a gripper initially in an excellent spot to grab the seed, but one tip can rotate the seed before it gets gripped (fourth image) which can cause it to snap out of control (fifth image). This is a type B error.

Fig. 9: Properties of raisins during debridement. Raisins are sticky and elastic, and a gripper that grips it by the edge often pulls it successfully out of the workspace (first two images). A type C error is when the gripper pulls it out using one tip (third image).

Prior to each trial, we inspected the contours of all eight fragments to check that perception was accurate (i.e., that their centers were correctly identified). For each fragment, a human judge classified its debridement outcome into one of four possibilities:

- **Success**: When a fragment is pulled out of the workspace using both grippers.
- **Error Type “A”**: True calibration errors. The gripper ends up too far from the center of a fragment to successfully grasp it. These are the most critical sources of error, as they suggest flaws in the calibration procedure.
- **Error Type “B”**: A fragment was not pulled out of the workspace despite accurate calibration. These happen with pumpkin seeds, which are smooth and lack friction with the paper surface. They can therefore slide out of control of the gripper (see Figure 8).
- **Error Type “C”**: When a gripper “grabs” a raisin with just one tip, implying that the fragment is pulled out of the workspace despite inaccurate calibration. See Figure 9 for details. We do not consider the case when the dVRK grabs a raisin off-center using both tips, as we consider this a more realistic debridement since it does not involve puncturing a fragment.

The rationale for splitting outcomes into these cases is that the “Success” metric provides a clear number as to how many fragments were pulled out of the workspace without getting punctured (i.e., excluding type C). In addition, distinguishing between type A and B errors makes it clearer to identify if failures are due to calibration or properties of the fragment, gripper, and/or setup.

For each trial, the same human judge labeled the result of all eight seeds. Certain failure cases were ambiguous (e.g., may have been A or B), so to be conservative, all ambiguous cases involving type A were labeled as type A.

One limitation of this setup is that evaluating the debridement outcome is subjective. To partially mitigate this, we ran our trials in a blind fashion: the choice of whether DNN or DNN+RF was used for calibration was unknown to the human judge until after labeling the outcome (success, A, B, or C) of all eight seeds in a trial.
TABLE III: Debridement Using Pumpkin Seeds. Results of 15 trials with eight pumpkin seeds each, for the DNN and DNN+RF cases following the setup in Section VI-A. The columns indicate the outcome of each fragment, separated by commas. A dash (−) indicates a fragment that was successfully pulled out of the workspace and the “Success” row indicates the total across 15 trials for each setting. Any other occurrence (A, B, or C) follows the labeling as described in Section VI-A. The average running time for these 30 trials was 57.62 ± 1.77 seconds.

| Trial | DNN | DNN+RF |
|-------|-----|--------|
| 1     | A,B,A,B,A,B,-,-,-,B | A,B,A,B,A,B,-,-,-,B |
| 2     | A,B,A,-,-,-,-,-,-,- | A,B,A,-,-,-,-,-,-,- |
| 3     | A,B,-,-,-,-,-,-,-,- | A,B,-,-,-,-,-,-,-,- |
| 4     | -,-,-,-,-,-,-,-,-,- | -,-,-,-,-,-,-,-,-,- |
| 5     | -,-,-,-,-,-,-,-,-,- | -,-,-,-,-,-,-,-,-,- |
| 6     | -,-,-,-,-,-,-,-,-,- | -,-,-,-,-,-,-,-,-,- |
| 7     | -,-,-,-,-,-,-,-,-,- | -,-,-,-,-,-,-,-,-,- |
| 8     | -,-,-,-,-,-,-,-,-,- | -,-,-,-,-,-,-,-,-,- |
| 9     | -,-,-,-,-,-,-,-,-,- | -,-,-,-,-,-,-,-,-,- |
| 10    | -,-,-,-,-,-,-,-,-,- | -,-,-,-,-,-,-,-,-,- |
| 11    | -,-,-,-,-,-,-,-,-,- | -,-,-,-,-,-,-,-,-,- |
| 12    | -,-,-,-,-,-,-,-,-,- | -,-,-,-,-,-,-,-,-,- |
| 13    | -,-,-,-,-,-,-,-,-,- | -,-,-,-,-,-,-,-,-,- |
| 14    | -,-,-,-,-,-,-,-,-,- | -,-,-,-,-,-,-,-,-,- |
| 15    | -,-,-,-,-,-,-,-,-,- | -,-,-,-,-,-,-,-,-,- |

Count: A:34, B:13, C:0; A:1, B:9, C:0
Success: 73/120

TABLE IV: Debridement Using Raisins. Results following the same setup and description as in Table III except that these use raisins instead of pumpkin seeds as fragments. The average running time for these 30 trials was 58.57 ± 0.80 seconds.

| Trial | DNN | DNN+RF |
|-------|-----|--------|
| 1     | A,B,A,B,A,B,-,-,-,B | A,B,A,B,A,B,-,-,-,B |
| 2     | A,B,A,-,-,-,-,-,-,- | A,B,A,-,-,-,-,-,-,- |
| 3     | A,B,-,-,-,-,-,-,-,- | A,B,-,-,-,-,-,-,-,- |
| 4     | -,-,-,-,-,-,-,-,-,- | -,-,-,-,-,-,-,-,-,- |
| 5     | -,-,-,-,-,-,-,-,-,- | -,-,-,-,-,-,-,-,-,- |
| 6     | -,-,-,-,-,-,-,-,-,- | -,-,-,-,-,-,-,-,-,- |
| 7     | -,-,-,-,-,-,-,-,-,- | -,-,-,-,-,-,-,-,-,- |
| 8     | -,-,-,-,-,-,-,-,-,- | -,-,-,-,-,-,-,-,-,- |
| 9     | -,-,-,-,-,-,-,-,-,- | -,-,-,-,-,-,-,-,-,- |
| 10    | -,-,-,-,-,-,-,-,-,- | -,-,-,-,-,-,-,-,-,- |
| 11    | -,-,-,-,-,-,-,-,-,- | -,-,-,-,-,-,-,-,-,- |
| 12    | -,-,-,-,-,-,-,-,-,- | -,-,-,-,-,-,-,-,-,- |
| 13    | -,-,-,-,-,-,-,-,-,- | -,-,-,-,-,-,-,-,-,- |
| 14    | -,-,-,-,-,-,-,-,-,- | -,-,-,-,-,-,-,-,-,- |
| 15    | -,-,-,-,-,-,-,-,-,- | -,-,-,-,-,-,-,-,-,- |

Count: A:4, B:0, C:3; A:1, B:0, C:0
Success: 113/120

B. Debridement Results

We ran 15 trials for each fragment. Tables III and IV list the debridement results for pumpkin seeds and raisins, respectively. In particular, they contain the exact outcome of each fragment and a tally of the error cases.

The results suggest that our DNN+RF calibration is highly reliable. Overall success rates are 110/120 (91.67%) and 119/120 (99.17%) for pumpkin seeds and raisins, respectively. These exceed the 89.4% success rate in the single-arm scenario reported in [15]. [22] did not report exact success rates but said results were similar to those in [15].

Failure cases due to calibration errors (type A) occur just once out of 120 instances for both pumpkin seeds and raisins (0.83% of the time). These are lower than the type A error rates with the DNN only, which are 34/120 (28.33%) for pumpkin seeds and 4/120 (3.33%) for raisins. The difference in type A error rates in the DNN case for the pumpkin seeds versus raisins is due to how raisins are elastic and can be pulled successfully despite slightly inaccurate calibration.

We observe that type B errors only occur with pumpkin seeds: 13 cases for DNN and 9 cases for DNN+RF, but zero times for both raisin scenarios. Again, this follows from properties of the fragments, workspace, and gripper. In particular, most of the type B errors occurred when the seeds had not been part of previous trials and thus retained their natural smoothness. When the gripper grabs seeds, it slightly deforms their edges, making them coarser and less likely to slip out of control in future trials. To avoid this from excessively biasing our results, we replaced seeds every 5 trials or after they experienced any significant damage.

Error type C, for raisins, occurred three times for DNN but zero times for DNN+RF, providing some additional evidence that the extra RF correction is beneficial.

Average runtimes for each trial were 57.62 seconds for pumpkin seeds and 58.57 seconds for raisins. These correspond to debridement rates of 7.20 seconds for pumpkin seeds and 7.32 seconds for raisins, more than an order of magnitude faster than the 91.8 seconds reported in [15] and more than 2.1x faster compared to the 15.8 seconds from [22] who furthermore tested only the faster two-arm scenario.

To investigate runtime in more detail, we ran a frame-by-frame analysis of a 25fps video of a pumpkin seed trial. We considered six phases for each fragment: to seed, lower, grip, raise, to receptacle, and drop & rotate. The average frame count for each phase across the eight seeds was, respectively, 26.0, 16.1, 37.5, 24.0, 42.1, and 38.5. Moving back to the receptacle, without loss in speed.

VII. CONCLUSIONS

In this paper, we demonstrate the ability for our two-step pipeline to calibrate the dVRK so that it can quickly and reliably debride pumpkin seeds and raisins. Future work should prioritize two limitations of this setup: reducing the influence of human bias when collecting data and evaluating results, and avoiding the need to hand-tune and discretize rotations. Other possibilities for future work include testing alternatives to our DNN and RF design, using more elaborate debridement setups with clutter, and conducting different autonomous surgical robotics tasks such as suturing.

ACKNOWLEDGMENTS

This research was performed at the AUTOLAB at UC Berkeley in affiliation with the RISE Lab, BAIR, and the CITRIS "People and Robots" (CPAR) Initiative.
