IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2024.0429000

A Real-Time Markerless Approach for 3D Pose Estimation of Laparoscopic Instruments in Surgical Training Simulators

Dehlela Shabir[1,2,3], Jhasketan Padhan[2], Nihal Abdurahiman[2], Julien Abinahed[2], Zhigang Deng[4], and Nikhil V. Navkar[1,2]

[1]AI Innovation Hub, Hamad Medical Corporation, Doha, Qatar
[2]Itqan Clinical Simulation and Innovation Center, Hamad Medical Corporation, Doha, Qatar
[3]Department of Computer Science and Engineering, Qatar University, Doha, Qatar
[4]Department of Computer Science, University of Houston, Houston, Texas, USA

Corresponding author: Nikhil V. Navkar (e-mail: nnavkar@hamad.qa).

Research reported in this publication was supported by the Qatar Research Development and Innovation Council "QRDI" Academic Research Grant (ARG) award ARG01-0430-230047 and Graduate Sponsorship Research Award (GSRA) GSRA11-L-1-0319-24006. All opinions, findings, conclusions, or recommendations expressed in this work are those of the authors and do not necessarily reflect the views of our sponsors.

ABSTRACT Box trainers are widely used for laparoscopic skills training due to their ease of setup and realistic tool-tissue interaction. Automating skill assessment on these systems can enhance training by reducing analysis time and eliminating subjective evaluations. Achieving this requires accurate real-time estimation of the instrument tooltip's three-dimensional (3D) pose. However, existing methods either depend on additional sensing hardware, which introduces ergonomic challenges, or lack real-time capabilities. A modular architecture has been developed for estimation of an instrument's tooltip pose from video frames acquired during laparoscopic training. Our method incorporates a GPU-based algorithm that takes input from a customized deep learning model (YOLOv7), detects the edges of the surgical instrument, and computes tooltip's 3D pose. Additionally, an automated 3D-pose dataset generation technique is introduced to generate datasets for training the model. Our method was evaluated using a vision-based tracking experiment, achieving 3D positional errors of 0.72 mm, 0.88 mm, and 3.81 mm over the X, Y, and Z-axes, and an orientation error of 5.65° with an end-to-end latency of 144 ms and sustained real-time throughput of 11.4 frames per second (fps). The proposed method enables accurate estimation of laparoscopic instrument tooltip poses without the need for additional hardware or markers. This work represents a significant step toward automated skill assessment in laparoscopic training using box trainers.

INDEX TERMS 3D Pose Estimation, Laparoscopic Simulators, Real-time Instrument Tracking

I. INTRODUCTION

LAPAROSCOPIC surgery is considered an evolution of conventional open surgeries due to several advantages such as reduced invasiveness, lower post-operative pain, minimized blood loss, scarring, infection rates, shorter hospital stays, and quicker recovery times [1]. It is performed within the abdominal cavity using elongated surgical instruments and an intra-corporeal scope camera. An inflated working space is generated within the abdominal cavity by infusing CO_2 gas, and the patient's anatomy (captured by the scope camera) is visualized on an external monitor. The surgical instruments and the camera are introduced through small incisions via cannulas (or trocars), and the miniaturized tooltips allow the surgeon to manipulate the patient's internal anatomy, dissect, ablate and suture tissues [2].

However, such advancements pose new challenges to surgeons. Laparoscopic procedures are limited by reduced field of view due to the single camera, loss of depth perception, inverted motions caused by pivoted rotation around the incision (also called the fulcrum effect), and amplification of normal hand tremor at the tip of long instruments [3]. Moreover, they demand specific laparoscopic psychomotor skills and hand-eye coordination in addition to skills required for open surgery, such as knowledge of anatomy, pathology, and surgical techniques. Hence, to reduce surgical errors and improve outcomes, surgeons must undergo extensive training

in acquiring laparoscopic skills, including the precise motion of surgical instrument tooltips while operating on the tissue at the surgical site.

Training for laparoscopic surgery typically follows either the traditional apprenticeship model [4] or simulation-based methods using box trainers or virtual reality (VR) simulators [5], [6], [7]. Unlike the apprenticeship model, simulation-based training offers several advantages, such as the ability to rehearse outside the operating room (OR), reduced ethical concerns and risk to patients, longer training periods, and exposure to a wider range of surgical tasks [8]. It has been shown that simulation-based training methods improve laparoscopic skills [9], which are transferable to the OR [10]. Among simulation-based methods, box trainers offer a low-cost alternative to VR simulators. A typical box trainer setup consists of a sealed box with entry ports, a tissue phantom, laparoscopic instruments, and an RGB camera (with an integrated light source) connected to a laptop for visualization. Due to ease of setup and realistic tool-tissue interaction, box trainers are widely used in surgical training. However, skill evaluation on box trainers has traditionally been performed manually by analyzing captured video using procedure-specific checklists and global rating scales. This makes skill assessment time-consuming and expensive [11]. Thus, automating skill assessment on box trainers has the potential to reduce the analysis time, eliminate subjectivity in human judgement, and allow expert surgeons to more efficiently assess trainees.

Automated skill assessment using box trainers is typically performed by employing external devices, such as sensors (optical or electro-magnetic) or cameras (RGB or depth), to measure laparoscopic tooltip motion in three-dimension (3D) space. A common approach is to analyze the collected tooltip motion data and calculate descriptive metrics, such as path length, smoothness, and curvature of instrument trajectories, to estimate the surgical skill of the trainee [12], [13]. In general, the motion of the laparoscopic instrument tooltip in 3D space forms the basis for all other derived metrics. To track laparoscopic instrument tooltip movement in 3D space, most approaches require either additional hardware-based systems (such as optical, electro-magnetic, or depth-based tracking systems) or attaching color-based or pattern-based markers to the laparoscopic instrument. This requires additional equipment installation with the box trainer, impose ergonomic challenges, or necessitates modifications to the surgical instruments and operative field. Machine Learning (ML)-based approaches, which rely solely on processing video frames acquired from the scope's RGB camera, have the potential to overcome these challenges. However, they are constrained by the limited availability of training datasets containing 3D position and orientation (pose) information of laparoscopic instruments and estimation techniques capable of predicting 3D poses in real-time.

In this work, we introduce a 3D pose estimation method that mitigates the need for hardware-based or marker-based instrument tooltip tracking. The main contributions of this work are outlined below:

1) A modular architecture for real-time estimation of instrument tooltip poses from video frames acquired during laparoscopic training scenarios.
2) A GPU-based algorithm that takes input from a customized ML model, detects edges of the surgical instrument, and computes the 3D pose of the tooltip in real-time.
3) An automated 3D-pose dataset generation technique. The dataset comprises videos depicting laparoscopic instrument movement in 3D space inside a box trainer and their corresponding poses relative to the scope camera. We will make this research dataset publicly available to the broad research community upon publication.

II. RELATED WORK

Existing 3D pose estimation methods for surgical instrument tooltips can be broadly classified into sensor-based and vision-based techniques. However, these approaches of localizing and estimating 3D poses face several restrictions when translating into practical applications for laparoscopic training. In this section, we discuss existing efforts and the challenges they encounter in 3D pose estimation. In the case of vision-based techniques, we also highlight the lack of much needed training datasets for developing ML models tailored to laparoscopic training environments.

A. SENSOR-BASED TECHNIQUES

Optical tracking uses retroreflective markers to determine an object's pose, but it requires an uninterrupted line of sight, and its accuracy varies across the tracking volume. In laparoscopic training, the instrument tip is often hidden within a cavity, preventing direct tracking. Therefore, markers are placed on the instrument handle [14]. However, this approach requires non-trivially translating the handle pose to the tip pose, and even slight shaft bending significantly reduces the accuracy.

Electromagnetic (EM) tracking determines the 3D pose of a wired sensor using a magnetic field generated by a field generator. These EM sensors are attached to either instrument handle [15] or near the distal end [16] to track the tooltip's movement during laparoscopic training. While EM tracking avoids line-of-sight issues [17], its accuracy is affected by magnetic field distortions caused by nearby ferromagnetic objects, including laparoscopic instruments. Inertial sensors are also used to track the motion of laparoscopic instruments but require extensive calibration steps to be used within a box trainer [18]. Time-of-Flight (ToF) sensors track surgical instrument motion in 3D [19]. However, ToF sensors rely on a flat surface for emitted pulse rebound, necessitating the addition of an external disk to the instrument shaft.

Other sensors have also been explored. LeapMotion uses multiple infrared (IR) cameras and LEDs to project structured IR light patterns onto the laparoscopic instrument, analyzing the deformation in the pattern to determine the 3D pose [20]. Similarly, a combination of the Microsoft Kinect

and colored markers placed on laparoscopic instrument enables 3D pose detection [21]. Another approach increases the number of scope cameras to capture laparoscopic instrument motion from different perspectives [22]. Although precise, these sensor-based methods require installation of additional expensive hardware and pose ergonomic challenges during instrument handling [23].

B. VISION-BASED TECHNIQUES

1) Computation Techniques

Vision-based techniques use computer vision and image processing to compute the 3D poses of surgical instruments from the video frames acquired by the scope camera. Most of these techniques rely on attaching a color-based or pattern-based marker to the distal end of the laparoscopic instrument. Examples of such markers used during laparoscopic training include equally spaced round-strip markers [24], [25], unique solid-colored tapes [26], cylindrical marker with a unique pattern [27], [28]. These techniques typically require manual modifications to instruments, which are not strictly reproducible and may suffer from human errors.

Another category of vision-based techniques involves aligning a 3D virtual model (e.g., a CAD model) of the instrument tooltip with their corresponding video frames to estimate the poses [29], [30]. While effective, such techniques require prior availability of 3D models and knowledge of tools, limiting their versatilities. Additional techniques rely on geometric constraints and gradient-based features to localize tooltip poses [31], [32]. However, these techniques often require knowledge of the insertion point, and the gradient-based features of tools tend to blend with the background, making these image processing methods unreliable for real-time implementation.

2) Learning-based Techniques

Traditional learning-based techniques heavily depended on hand-crafted, color-based, or texture-based feature descriptors of surgical instruments to train computationally intensive algorithms like support vector machines, random forests, and decision trees [33], [34], [35]. However, these techniques were not only limited by their lack of real-time performance but also by poor robustness to variations in colors, lighting, and the presence of complex tissues, restricting their application to pose estimation in 2D space.

To overcome such limitations, the focus has shifted toward deep learning (DL). Most DL techniques utilize Convolutional Neural Networks (CNNs) to automatically extract robust image features, eliminating the need for tedious feature engineering processes [36], [37]. Zhao et. al. [38] proposed a 2D shaft and end-effector tip estimation mechanism based on instrument edge-line extraction using CNNs, followed by 3D pose estimation. However, their work relied on predetermined 3D position of the instrument's insertion point and does not demonstrate real-time capabilities. Hasan et. al. [39] proposed a DL-based technique with one encoder and multiple decoders to detect tool existence, followed by direct regression of the geometric primitives of the laparoscopic shaft from input video frames. The 3D pose was estimated from the extracted geometric primitives. This study achieved good performance on in-vivo data; however, the results were evaluated over a subset of 32 image frames selectively chosen based on acceptable reprojection errors. Although the proposed model was lightweight, it was not evaluated for real-time performance. Gerats et. al. [40] proposed a method using neural field representations of surgical instruments, enhanced by OmniMotion [41] and class weighting for 3D depth estimations of laparoscopic instruments from monocular videos. However, their method does not estimate poses in all 3D dimensions and is based on generated pseudo-depth of the instruments, which are approximations derived from scene reconstructions. Several other attempts at instrument pose tracking based on DL have shown promising results [42], [43], [44], but they are either limited to 2D space or focus on robotic instruments with articulated tooltips and stereo vision.

C. TRAINING DATASETS

For ML-based 3D pose estimation of laparoscopic surgical tools, the requirement is a well-annotated, reliable dataset for training and evaluation. While there is a plethora of datasets for robotic surgeries, limited data are available for manual laparoscopic surgeries. Existing laparoscopic datasets include 2D manually annotated datasets for laparoscopic tool presence classification [39], [45], tool mask segmentation [46], [47], tool type, surgical workflow, phase, and action recognition [37], [45], [48]. Recently published datasets, such as Cholec80-locations [37] and CholecT50 [48], include laparoscopic tool detection data with 2D bounding boxes around the tooltips. Additionally, the ART-Net dataset [39] includes images with precise locations of manually annotated laparoscopic instrument shaft tips. However, these datasets are limited to 2D annotations and do not include 3D ground truth poses of the instruments. To overcome the challenges related to the lack of datasets, this work proposes an automated 3D-pose dataset generation technique, which generates datasets containing 3D pose annotations for laparoscopic instruments used in box trainers.

III. MATERIALS AND METHODS

Estimation of the instrument's tooltip poses from video frames is implemented as a pipeline comprising multiple modules. Section III-A provides an overview of the pose extraction pipeline, while Section III-B explains the functionality of each module in detail. Section III-C describes the implementation of parallelization within the pipeline. The experimental setup used to evaluate the pipeline is presented in Section III-D.

A. MODULAR ARCHITECTURE

The architecture of the proposed pose extraction pipeline is shown in Fig. 1. The pipeline takes video frame $\mathbf{F}(t)$ as input, processes it through five sequential modules, and outputs the estimated tooltip pose. We use the naming convention $\mathbf{X}(t)$

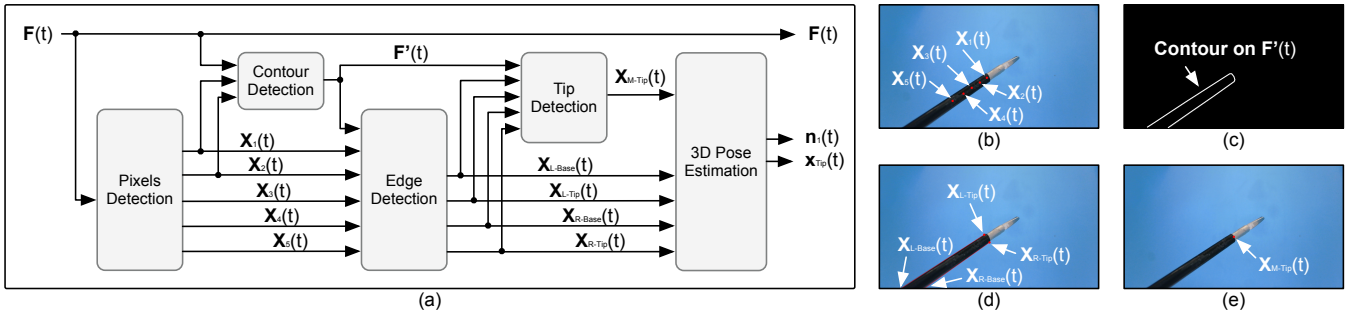

FIGURE 1. **(a) Architecture of the proposed pose extraction pipeline comprising of five modules. (b) Points detected by the pixel detection module on the surgical tool shaft. (c) The contour of the surgical tool shaft generated by the contour detection module. (d) Endpoints of the left and right line segments representing the edges of the surgical tool shaft generated by the edge detection module. (e) The tooltip detected by tip detection module in the image frame.**

$\in \mathbb{R}^3$ where $\mathbf{X}(t)$ is a vector representing a 3D point at time instance t, measured with respect to a camera-frame coordinate system. The Z axis of the coordinate system is orthogonal and pointing towards the video frame plane, while the X and Y axes are same as the video frame plane. Within each module, if required, $\mathbf{X}(t)$ can be projected onto the video image plane and its corresponding pixel can be retrieved on the video image to enable image processing. The tooltip pose is defined by a point $\mathbf{X}_{Tip}(t)$, representing the position of the tooltip shaft, and a vector $\mathbf{n}_1(t)$, indicating its orientation. Both $\mathbf{X}_{Tip}(t)$ and $\mathbf{n}_1(t)$ are measured with respect to the reference frame of the scope camera. Each module of the pipeline is implemented as an independent thread running in parallel. The data transferred from one module to another is stored in a deque, with a constant length. The design ensures that each module processes data as soon as it becomes available and passes it immediately to the next module, ensuring an efficient and uninterrupted flow of data throughout the pipeline.

The *pixel detection module* receives the image frame $\mathbf{F}(t)$ from the scope camera, processes the image using DL techniques (described in Section III-B1), and detects two pixels which are converted to the points $\mathbf{X}_1(t)$ and $\mathbf{X}_2(t)$ on the video image plane. The module also detects up to three additional collinear points, $\mathbf{X}_i(t)$ where $3 \leq i \leq 5$, along the midline of the tool. These points form an approximate midline of the tool (shown in Fig. 1b). Points $\mathbf{X}_1(t)$ and $\mathbf{X}_2(t)$ are sent to the *contour detection module*, while all the points are sent to *edge detection module*. The *contour detection module* processes the image frame $\mathbf{F}(t)$, extracts the pixels outlining the surgical tool shaft, and stores it in a binary image frame $\mathbf{F}'(t)$ (Fig. 1c). The module operates under two modes, as described in Section III-B2. The *edge detection module* uses the binary image frame $\mathbf{F}'(t)$ along with five points $\mathbf{X}_i(t)$ (where $1 \leq i \leq 5$) to compute the edges of the surgical tool shaft in the image frame, represented by two line segments with endpoints $\mathbf{X}_{L\text{-}Tip}(t)$, $\mathbf{X}_{L\text{-}Base}(t)$, and $\mathbf{X}_{R\text{-}Tip}(t)$, $\mathbf{X}_{R\text{-}Base}(t)$ (shown in Fig. 1d). A shift-rotate-search algorithm (described in Section III-B3) is implemented on a GPU to compute these lines segments. The line segments, along with binary image frame $\mathbf{F}'(t)$, is fed to the *tip detection module*, which computes the tip $\mathbf{X}_{M\text{-}Tip}(t)$ of the tool shaft in the image (Fig. 1e). The details of the *tip detection module* are presented in Section III-B4. The tip $\mathbf{X}_{M\text{-}Tip}(t)$, along with the left and right line segment endpoints, are then fed to the last *3D pose estimation module* in the pipeline. This module (described in Section III-B5) computes the pose of the tooltip and returns $\mathbf{X}_{Tip}(t)$ and $\mathbf{n}_1(t)$.

B. MODULES

1) Pixel Detection Module

The pixel detection module detects up to five collinear points on the video frame $\mathbf{F}(t)$, starting from the tip of the tool and extending along its midline. A deep learning-based model was trained on our proposed dataset to detect two points $\mathbf{X}_1(t)$ and $\mathbf{X}_2(t)$, and three additional points $\mathbf{X}_i(t)$, where $3 \leq i \leq 5$, depending on the length of the instrument visible in the frame. We formalize the pixel detection problem as a supervised multi-output regression problem, where the input is the video frame $\mathbf{F}(t)$, and the output includes n pixel coordinates $(x_i, y_i) \in \mathbb{R}^2$ for $i = 1, \ldots, n$, where $2 \leq n \leq 5$. The model was based on YOLOv7 [49] object detection network, where bounding box regression was modeled as an accurate pixel regressor.

The data generation process was automated using a robotic manipulator. A laparoscopic instrument was mounted to the end effector of a UR5e robotic arm (Universal Robots – Denmark) using a 3D-printed instrument adapter (Fig. 2a). Alongside the laparoscopic instrument, three scopes were inserted into a box trainer lined with a monochromatic screen. A tissue phantom was placed inside the box trainer, and a marker (developed by [27]) was attached to the tooltip of the laparoscopic instrument (Fig. 2b). Three different scopes were used to acquire the motion of the laparoscopic instrument with respect to the tissue phantom from different viewpoints. The marker provided the position and orientation (pose) of the tooltip relative to the camera.

A pre-defined motion of the laparoscopic instrument was generated using the ROS platform based on our previous work [50]. The motion enabled the tooltip to traverse a path inside a cube ($6 \times 6 \times 6cm^3$) while maintaining a remote center of motion at the incision point on the box trainer (Fig. 2c). The path was discretized into waypoints for the movement

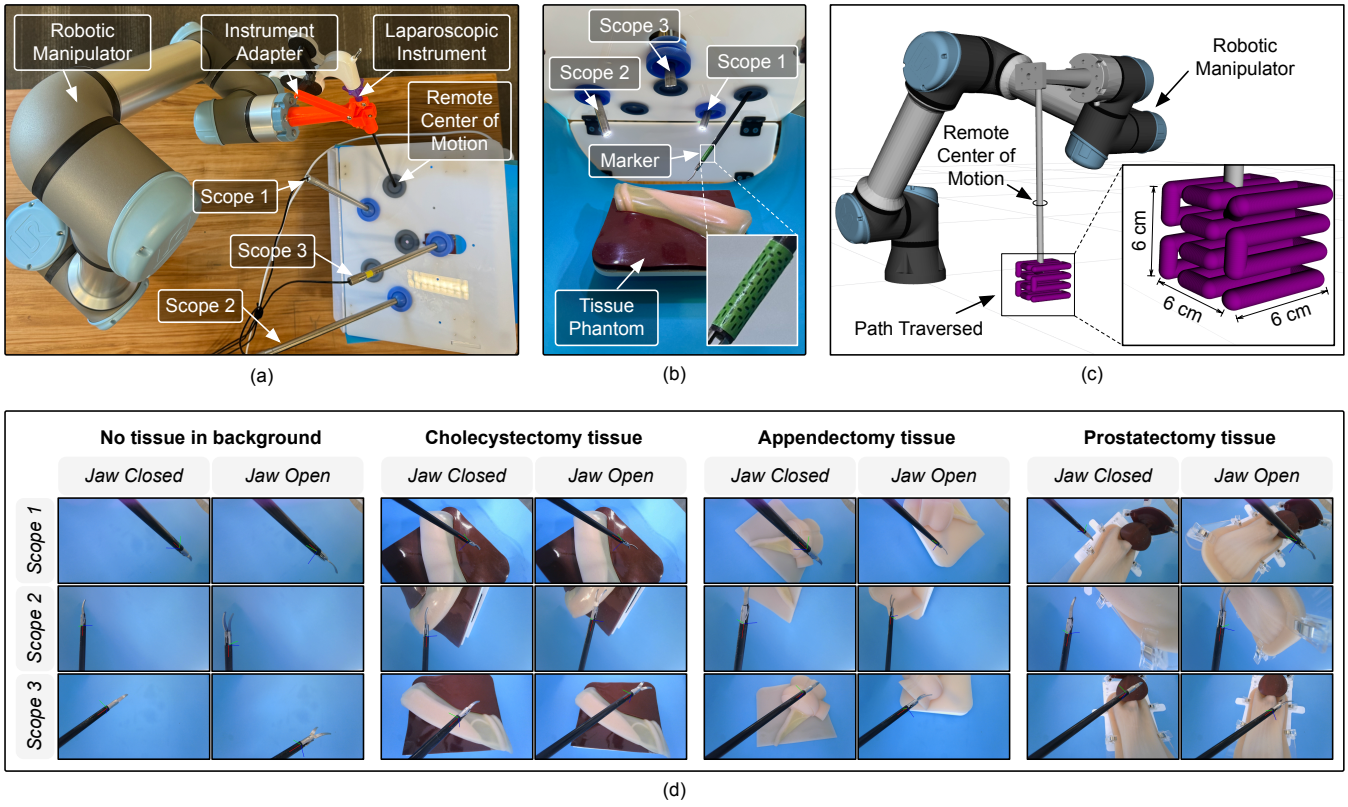

FIGURE 2. **(a) Setup used for data collection process, comprising of laparoscopic instrument attached to a robotic manipulator, three scopes, and a box trainer. (b) Inside of the box trainer with a tissue phantom placed on a monochromatic screen. A visual marker is attached to the tooltip of the instrument. (c) Path traversed by the tooltip while maintaining a remote center of motion at the incision point. (d) Sample of images in the collected dataset comprising four different tissue backgrounds from the three scope cameras. Each background consists of images with tooltip jaw in both open and close state.**

of the tooltip. At each waypoint, three images were acquired using the scope cameras. The robot controller executed the motion twice: first with the marker attached to the tooltip, and then without it. During the first motion, the pose of the tooltip relative to the camera was recorded for each frame. In the second execution, the poses computed from the first motion were mapped to the corresponding images acquired during the second execution. This dual-step execution produced images of the tool without the marker while retaining pose information. The dataset was collected using the three scope cameras, incorporating four distinct tissue backgrounds and capturing the tool's jaw in both open and closed states for each background (Fig. 2d). Frames where the marker was not clearly visible were excluded. The final dataset contained a total of 9,590 frames with tooltip pose annotations (detailed in Section III-C).

For a frame $\mathbf{F}(t)$, the dataset provided the corresponding tooltip's 3D pose. The 3D position of the tip was projected onto frame $\mathbf{F}(t)$ using known camera intrinsics, to obtain $\mathbf{X}_1(t)$. To generate the rest of the points, the 3D tip was translated to four equidistant locations along the orientation of the tool shaft. These 3D points were projected onto frame $\mathbf{F}(t)$ to generate the ground truth for the five points following $\mathbf{X}_1(t)$. Additionally, since the model was based on YOLOv7, each pixel coordinate was assigned a class label and a surrounding bounding box. The class labels were ordered from 1 to 5, with 1 at the distal end of the instrument. The bounding box annotation process was automated based on known camera and tool information in 3D space, eliminating manual efforts.

Since the model was trained to detect multiple points on the shaft, it was susceptible to drifting (non-collinear detections) and increased false positives. To address this, we ensured that each of the bounding box included a portion of the tool's metallic region. This metallic reference provided a spatial anchor, ensuring that detected points were constrained relative to the metallic region. Incorporating this reference enabled our model to produce context-aware detections, reducing reliance on ambiguous features of the shaft. The approach also improved the model's stability compared to modeling this problem as a key-point estimation issue.

A total of 8,267 frames from the dataset were used, split into 80% training ($n = 6613$), 10% validation ($n = 827$) and 10% testing ($n = 827$) subsets. To evaluate the training performance of the model, we focused on minimizing localization loss and additionally monitored precision, recall, and mAP (mean Average Precision). The model utilized transfer learning from pre-trained weights of YOLOv7 trained on the COCO (Common Objects in COntext) dataset [51]. The

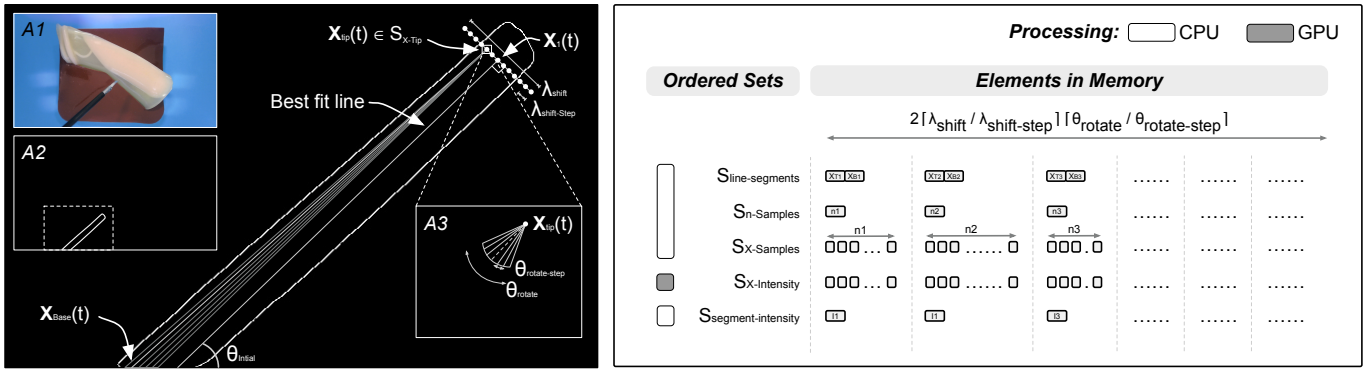

FIGURE 3. **(a) Schematic representation of the processing performed by the edge detection module. Panel A1 shows the original the frame F(*t*), Panel A2 shows the binary image frame F′(*t*) after contour detection, and Panel A3 illustrates the rotation angles for the edge detection algorithm. (b) Memory allocation for the ordered sets to facilitate processing on both CPU and GPU.**

training consisted of 100 epochs and a batch size of 16 on a Quadro RTX 5000 GPU with four dedicated CPU threads. Data augmentations techniques included 20% translation, 50% scaling, 50% flip up & down, 50% flip left & right, and HSV channels adjustments (1.5% Hue, 70% Saturation, 40% value). Additionally, mosaic augmentation was applied to every training sample, which creates a composite image out of four randomly sampled frames. Each of these four images in a mosaic is randomly cropped, placed and scaled to ensure robustness to complex backgrounds, better detection of smaller tool sizes, and improved generalization. The mosaic also ensured inclusion of cases where $\mathbf{F}(t)$ contained no tools, eliminating data imbalance.

2) Contour Detection Module

The *contour detection module* processes the image frame $\mathbf{F}(t)$, generates a contour in the form of pixels outlining the surgical tool shaft on the image, stores it in a binary image frame $\mathbf{F}'(t)$, and passes it to edge detection and tip detection modules. The module runs two variants:

- **Canny variant:** The variant involves several preprocessing steps. First, the video frame $\mathbf{F}(t)$ is smoothed using median blur with kernel size of 25, producing $\mathbf{F}_{blur}(t)$. Next, to adjust brightness, $\mathbf{F}_{blur}(t)$ undergoes linear blending with weight of 1.2. The resultant image is then converted to greyscale, followed by Canny edge detection [52] with a hysteresis threshold range of 120 to 130. To improve robustness to lighting variations, an additional image of higher brightness is formed using the same image processing steps. This is achieved by blending $\mathbf{F}_{blur}(t)$ with a weight of 3, followed by greyscale conversion and Canny edge detection with the same threshold range. The two Canny-generated binary images are then combined to produce the binary image frame $\mathbf{F}'(t)$.
- **FasterSAM variant:** The variant receives video frame $\mathbf{F}(t)$ along with points $\mathbf{X}_1(t)$ and $\mathbf{X}_2(t)$ and extracts the contour in three steps. First, an object segmentation model, Faster Segment Anything Model (FasterSAM), generates a separate mask for every detectable object within $\mathbf{F}(t)$. Second, each mask is validated to determine if it contains the points $\mathbf{X}_1(t)$ and $\mathbf{X}_2(t)$ that specify the location of the tool. Additionally, the bounding box of the tool shaft is also generated using the points $\mathbf{X}_1(t)$ and $\mathbf{X}_2(t)$ to specify the region of interest. These location and region prompts are combined to select the frame containing the mask of the tool shaft. In the third step, the tool mask is processed using Canny edge detection to extract the tool contour $\mathbf{F}'(t)$. If no mask is found, the system defaults to *Canny variant* of contour detection.

3) Edge Detection Module

The *edge detection module* takes as input: (a) the binary image frame $\mathbf{F}'(t)$ (Fig. 3a, Panel A2) generated from video frame $\mathbf{F}(t)$ (Fig. 3a, Panel A1), (b) five points $\mathbf{X}_i(t)$ where $1 \leq i \leq 5$, and (c) four parameters θ_{rotate}, $\theta_{rotate\text{-}step}$, λ_{shift}, $\lambda_{shift\text{-}step}$. It outputs two line segments: (a) $\mathbf{X}_{L\text{-}Tip}(t)$ and $\mathbf{X}_{L\text{-}Base}(t)$, representing the left side of surgical tool shaft, and (b) $\mathbf{X}_{R\text{-}Tip}(t)$ and $\mathbf{X}_{R\text{-}Base}(t)$, representing the right side of the surgical tool shaft on the image frame. The *edge detection module* runs an algorithm (Algorithm 1) to search for optimal edges on both left and right sides from multiple line segments representing the edges. The multiple line segments are generated by adjusting the slope and shifting the position from an initial line segment.

An initial slope $\theta_{initial}$ of a line passing through the five points $\mathbf{X}_i(t)$ (where $1 \leq i \leq 5$) is calculated (Fig. 3). An ordered set $S_{\theta\text{-}tip}$ of rotation angles is computed for searching. The range varies from $\theta_{initial} - (\theta_{rotate}/2)$ to $\theta_{initial} + (\theta_{rotate}/2)$ with a step size of $\theta_{rotate\text{-}step}$ (Fig. 3, Panel A3). A set of points $S_{X\text{-}Left}$ and $S_{X\text{-}Right}$ are computed on both left and right sides of a line passing through $\mathbf{X}_1(t)$ and orthogonal to $\theta_{initial}$. This is done by shifting the point $\mathbf{X}_1(t)$ by a step size of $\lambda_{shift\text{-}step}$ until the distance λ_{shift} is covered. Both sets $S_{X\text{-}Left}$ and $S_{X\text{-}Right}$ are combined into $S_{X\text{-}Tip}$.

After storing the range of possible values for the slope and

Algorithm 1 Edge Detection Module

Input: $\mathbf{X}_1(t), \mathbf{X}_2(t), \mathbf{X}_3(t), \mathbf{X}_4(t), \mathbf{X}_5(t), \mathbf{F}'(t), \theta_{rotate}, \theta_{rotate\text{-}step}, \lambda_{shift}, \lambda_{shift\text{-}step}$
Output: $\mathbf{X}_{L\text{-}Tip}(t), \mathbf{X}_{L\text{-}Base}(t), \mathbf{X}_{R\text{-}Tip}(t), \mathbf{X}_{R\text{-}Base}(t)$

$\theta_{initial} \leftarrow getBestFitLineSlope(\mathbf{X}_1(t), \mathbf{X}_2(t), \mathbf{X}_3(t), \mathbf{X}_4(t), \mathbf{X}_5(t))$
$S_{\theta\text{-}Tip} = \{\theta_{initial} - (\theta_{rotate}/2) + k \cdot \theta_{rotate\text{-}step} \mid k = 0, 1, \ldots, \lceil \theta_{rotate}/\theta_{rotate\text{-}step} \rceil\}$
$S_{X\text{-}Left} = \left\{\mathbf{X}_1(t) - k \cdot \lambda_{shift\text{-}step} \cdot (\cos(\theta_{initial} + 90°), \sin(\theta_{initial} + 90°)) \mid k = 0, 1, \ldots, \lceil \lambda_{shift}/\lambda_{shift\text{-}step} \rceil\right\}$
$S_{X\text{-}Right} = \left\{\mathbf{X}_1(t) + k \cdot \lambda_{shift\text{-}step} \cdot (\cos(\theta_{initial} + 90°), \sin(\theta_{initial} + 90°)) \mid k = 0, 1, \ldots, \lceil \lambda_{shift}/\lambda_{shift\text{-}step} \rceil\right\}$
$S_{X\text{-}Tip} = S_{X\text{-}Left} \cup S_{X\text{-}Right}$
for each $(\mathbf{X}_{tip}, \theta_{tip})$ *where* $(\mathbf{X}_{tip}, \theta_{tip}) \in S_{X\text{-}Tip} \times S_{\theta - Tip}$ ***do***
    $\mathbf{X}_{base} \leftarrow getBasePoint(\mathbf{X}_{tip}, \theta_{tip})$
    $S_{line\text{-}segments} = S_{line\text{-}segments} \cup \{(\mathbf{X}_{tip}, \mathbf{X}_{base})\}$
    $S_{n\text{-}samples} = S_{n\text{-}samples} \cup \{\|\mathbf{X}_{tip} - \mathbf{X}_{base}\|_2\}$
    $S_{X\text{-}samples} = S_{X\text{-}samples} \cup \{\mathbf{X}_{tip} + (k \cdot (\mathbf{X}_{tip} - \mathbf{X}_{base})/\|\mathbf{X}_{tip} - \mathbf{X}_{base}\|_2) \mid k = 0, 1, \ldots, \lceil \|\mathbf{X}_{tip} - \mathbf{X}_{base}\|_2 \rceil\}$
end for
for each $\mathbf{X}$ *where* $\mathbf{X} \in S_{X\text{-}samples}$ ***in parallel do***
    $S_{X\text{-}Intensity} = S_{X\text{-}Intensity} \cup \{getIntensity(\mathbf{X}, \mathbf{F}'(t))\}$
end for
$c_n = 0$
for each n *where* $n \in S_{n\text{-}samples}$ ***do***
    $S_{segment\text{-}intensity} = S_{segment\text{-}intensity} \cup \{\sum s_k/n \mid s_k \in S_{X\text{-}Intensity},\ k = c_n + 1, \ldots, c_n + n\}$
    $c_n = c_n + n$
end for
$c_{segments} = \lceil \theta_{rotate}/\theta_{rotate\text{-}step} \rceil \cdot \lceil \lambda_{shift}/\lambda_{shift\text{-}step} \rceil$
$S_{segments\text{-}left} \leftarrow sort(S_{segment\text{-}intensity}, 1, c_{segments})$
$S_{segments\text{-}right} \leftarrow sort(S_{segment\text{-}intensity}, c_{segments}, 2 \cdot c_{segments})$
$(\mathbf{X}_{L\text{-}Tip}(t), \mathbf{X}_{L\text{-}Base}(t)) \leftarrow getBestFitLine(S_{segments\text{-}left}, 5)$
$(\mathbf{X}_{R\text{-}Tip}(t), \mathbf{X}_{R\text{-}Base}(t)) \leftarrow getBestFitLine(S_{segments\text{-}right}, 5)$

position in $S_{\theta\text{-}tip}$ and $S_{X\text{-}Tip}$, three additional sets, $S_{line\text{-}segments}$, $S_{n\text{-}samples}$, and $S_{X\text{-}samples}$, are computed. These sets store the endpoints of all possible line segments, the number of samples on each line segment, and the position of the samples (as shown in Fig. 3b), respectively. For each sample, an intensity value I is computed and stored in $S_{X\text{-}Intensity}$. The intensity computation is performed in parallel using the *getIntensity* function, which applies a 9×9 kernel as follows:

$$I = \frac{\sum_{j=\lfloor y \rfloor - 4}^{\lfloor y \rfloor + 5} \sum_{i=\lfloor x \rfloor - 4}^{\lfloor x \rfloor + 5} w(i,j) \cdot I(\mathbf{X}_{i,j})}{\sum_{j=\lfloor y \rfloor - 4}^{\lfloor y \rfloor + 5} \sum_{i=\lfloor x \rfloor - 4}^{\lfloor x \rfloor + 5} w(i,j)},$$

where

$$w(i,j) = \begin{cases} \frac{1}{\|\mathbf{X} - \mathbf{X}_{i,j}\|_2} & \text{if } \mathbf{X}_{i,j} \neq \mathbf{X} \\ 0 & \text{if } \mathbf{X}_{i,j} = \mathbf{X} \text{ or } \mathbf{X}_{i,j} \text{ is outside } \mathbf{F}(t). \end{cases}$$

Here, $\mathbf{X}_{(i,j)}$ represents the pixel at i-th row and j-th column of the frame $\mathbf{F}'(t)$. $I(\mathbf{X}_{(i,j)})$ denotes the intensity of the pixel at the i-th row and j-th column of the frame $\mathbf{F}'(t)$. The intesity values for each sample within a line segment are combined to compute the overall intensity of the entire line segment. The higher the intensity values, the closer the segment is to the edges in $\mathbf{F}'(t)$. These intensity values for the line segments are then sorted, and the top five line segments with the highest intensity values are selected to determine the best-fit line segment. The tip and base of the best-fit line segment on the left side are represented by the endpoints $\mathbf{X}_{L\text{-}Tip}(t)$ and $\mathbf{X}_{L\text{-}Base}(t)$. Similarly, the tip and base of the best-fit line segment on the right side are represented by the endpoints $\mathbf{X}_{R\text{-}Tip}(t)$ and $\mathbf{X}_{R\text{-}Base}(t)$. These endpoints are returned by Algorithm 1 and passed on to the 3D pose estimation module.

4) Tip Detection Module

The *tip detection module* utilizes the points $\mathbf{X}_{L\text{-}Tip}(t)$, $\mathbf{X}_{L\text{-}Base}(t)$, $\mathbf{X}_{R\text{-}Tip}(t)$ and $\mathbf{X}_{R\text{-}Base}(t)$ received from the edge detection module to compute the point $\mathbf{X}_{M\text{-}Tip}(t)$. A midline is computed from the left and right line segments, and $\mathbf{X}_{M\text{-}Tip}(t)$ is located along this midline at the point where there is a significant change in pixel intensity, corresponding to the region where the metallic portion of the tool connects to the shaft.

5) 3D Pose Estimation Module

The *3D pose estimation module* computes $\mathbf{X}_{Tip}(t)$ and $\mathbf{n}_1(t)$ based on points $\mathbf{X}_{L\text{-}Tip}(t)$, $\mathbf{X}_{L\text{-}Base}(t)$, $\mathbf{X}_{R\text{-}Tip}(t)$, and $\mathbf{X}_{R\text{-}Base}(t)$ from the *edge detection module* and $\mathbf{X}_{M\text{-}Tip}(t)$ from the *tip detection module*. The radius of the shaft $\lambda_{ToolRadius}$ and the position of the surgical scope camera $\mathbf{X}_{Camera}(t)$ are known. Two unit vectors $\mathbf{n}_L(t)$ and $\mathbf{n}_R(t)$ are defined such that $\mathbf{n}_L(t)$ is orthogonal to the plane consisting of points $\mathbf{X}_{L\text{-}Base}(t)$, $\mathbf{X}_{L\text{-}Tip}(t)$, and $\mathbf{X}_{Camera}(t)$, and $\mathbf{n}_R(t)$ is orthogonal to the plane consisting of points $\mathbf{X}_{R\text{-}Base}(t)$, $\mathbf{X}_{R\text{-}Tip}(t)$, and

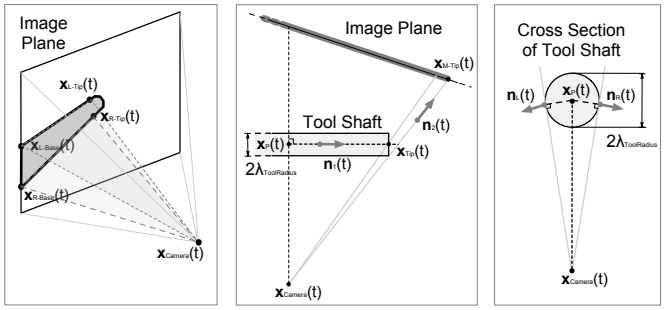

FIGURE 4. (a) Two planes enclosing the surgical tool shaft, formed by two sets of points ($\mathbf{X}_{L\text{-}Tip}(t)$, $\mathbf{X}_{L\text{-}Base}(t)$, and $\mathbf{X}_{Camera}(t)$) and ($\mathbf{X}_{R\text{-}Tip}(t)$, $\mathbf{X}_{R\text{-}Base}(t)$, and $\mathbf{X}_{Camera}(t)$). (b) Top view of the image plane and tool shaft, showing the point $\mathbf{X}_{M\text{-}Tip}(t)$ re-projected to find point $\mathbf{X}_{Tip}(t)$. (c) Cross-sectional view of the tool-shaft $\mathbf{X}_{M\text{-}Tip}(t)$, with $\mathbf{n}_L(t)$ and $\mathbf{n}_R(t)$ representing the normal vectors of the planes enclosing the tool shaft.

$\mathbf{X}_{Camera}(t)$ (Fig. 4a). In addition, two unit vectors $\mathbf{n}_1(t)$ and $\mathbf{n}_2(t)$ are computed as follows:

$$\mathbf{n}_1(t) = \frac{\mathbf{n}_L(t) \times \mathbf{n}_R(t)}{\|\mathbf{n}_L(t) \times \mathbf{n}_R(t)\|},$$

$$\mathbf{n}_2(t) = \frac{\mathbf{X}_{M\text{-}Tip}(t) - \mathbf{X}_{Camera}(t)}{\|\mathbf{X}_{M\text{-}Tip}(t) - \mathbf{X}_{Camera}(t)\|},$$

$\mathbf{n}_1(t)$ represents the direction of the tool shaft, whereas $\mathbf{n}_2(t)$ represents the direction from $\mathbf{X}_{Camera}(t)$ to $\mathbf{X}_{M\text{-}Tip}(t)$ (Fig. 4b). The tip $\mathbf{X}_{Tip}(t)$ of the tool shaft is computed as:

$$\mathbf{X}_{Tip}(t) = \mathbf{X}_{M\text{-}Tip}(t) + \mathbf{n}_1(t) \left(\frac{((\mathbf{X}_P(t) - \mathbf{X}_{M\text{-}Tip}(t)) \times \mathbf{n}_2(t)) \cdot (\mathbf{n}_1(t) \times \mathbf{n}_2(t))}{\|\mathbf{n}_1(t) \times \mathbf{n}_2(t)\|^2} \right) - \mathbf{n}_2(t) \left(\frac{\lambda_{ToolRadius}}{\tan(\theta_{12}(t))} \right),$$

$$\mathbf{X}_P(t) = \mathbf{X}_{Camera}(t) + \left(\frac{\mathbf{n}_L(t) + \mathbf{n}_R(t)}{\|\mathbf{n}_L(t) + \mathbf{n}_R(t)\|} \right) \left(\frac{\lambda_{ToolRadius}}{\cos(\theta_{LR}(t)/2)} \right),$$

where $\theta_{LR}(t)$ is the angle between vectors $\mathbf{n}_L(t)$ and $\mathbf{n}_R(t)$, and $\theta_{12}(t)$ is the angle between $\mathbf{n}_1(t)$ and $\mathbf{n}_2(t)$ (Fig. 4c). The point $\mathbf{X}_P(t)$ is computed such that the direction along the points $\mathbf{X}_P(t)$ and $\mathbf{X}_{Camera}(t)$ is orthogonal to $\mathbf{n}_1(t)$.

C. PARALLELIZED IMPLEMENTATION

In our work, we define real-time as the ability of the pipeline to process input frames at a specified target frame rate (30 fps) with bounded end-to-end latency. All timings were obtained from a representative workstation with NVIDIA Quadro RTX 5000 GPU, Intel Xeon Silver 4216 CPU (32 cores, 2.10 GHz), 128 GB RAM, after 50 warm-up frames. The pipeline was implemented in C++ on the Windows operating system and compiled using MSVC (Microsoft Visual C++). All machine learning models were converted to the ONNX format using the ONNX Runtime library with CUDA support for GPU acceleration and compatibility within the C++ environment.

To achieve parallelism, the std::thread library was utilized, which provides a high-level, portable interface for creating and managing threads in C++. In MSVC, each std::thread creates a native Windows thread managed by the Windows kernel scheduler. Windows uses a preemptive, priority-based, time-sliced scheduler. The scheduler allocates CPU time based on thread priorities, preempting lower-priority threads when higher-priority ones become runnable. Within the same priority level, threads share CPU time in a round-robin manner. The Windows scheduler aims for fairness but does not guarantee deterministic ordering or equal CPU time — it depends on load, core count, and thread priority. When executing our application, it does not manage or assign thread priority explicitly and relies solely on the Windows OS scheduler for thread scheduling. However, we run the application in a controlled environment, without any other heavy resource-consuming (CPU time or memory) applications in the background, to ensure that our application receives more consistent CPU time and runs smoothly.

D. EXPERIMENTAL SETUP

1) Assessing Performance of the Pose Detection Pipeline

The experiment assessed the performance of the proposed pose detection pipeline in terms of positional error, orientation error, and computation time while varying the input parameters. A new experimental dataset was collected (different from the training dataset detailed in Section III-B1). In the setup, a computer vision-based technique was used to compute the ground truth. The experimental setup was similar to the data generation setup, which utilized a robotic manipulator (Fig. 2a). The robotic manipulator maneuvered a laparoscopic instrument (curved scissors) inside the box trainer along a path. The instrument was maneuvered over three different tissue background structures (suturing pad, cholecystectomy, and abdomen, as shown in Fig. 5a), and for each background structure, three different paths were used (circular = 891 frames, sinusoidal = 1854 frames, and spiral = 1836 frames, as shown in Fig. 5b) totalling 4581 frames. For each path, the instrument's movement was recorded twice by a single scope camera: one with the marker and the other without it. A frame $\mathbf{F}(t)$ of size 1920×1080 was acquired from the scope camera. The tooltip poses computed from video frames with the visual marker served as the ground truth and were compared to those obtained using the proposed pose detection pipeline from frames without the marker. This marker-based pose estimation technique was chosen due to their proven work exceeding state-of-the-art, and code availability. The marker has a reported mean absolute 3D error of 0.28 mm and 0.45°on all axes, for a working distance between 50 and 100 mm. The comparison enabled the computation of positional and orientational errors for the detected poses. In addition, the computational time for Algorithm 1 was also recorded. The computational load of the pose detection pipeline was varied

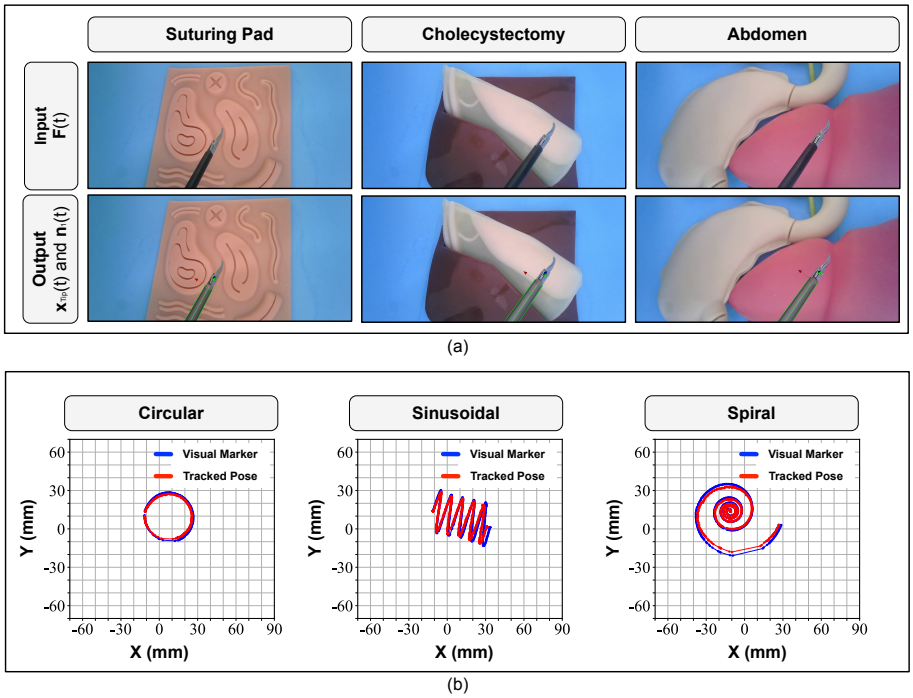

FIGURE 5. **(a) Background tissues used in the experimental setup to compute position and orientation error. The pipeline takes video frame F(t) as input and outputs the position $\mathbf{X}_{Tip}(t)$ of the tooltip shaft and a vector $\mathbf{n}_1(t)$ indicating its orientation. (b) Paths traversed by the laparoscopic instrument tooltip, tracked using a visual marker and the proposed pose detection pipeline.**

by altering its parameters to assess its performance under different settings. In particular, $\lambda_{shift\text{-}step}$ parameter values varied between 1, 2, and 3; $\theta_{rotate\text{-}step}$ parameter was varied among 0.1, 0.5, and 1.0; and both variants (*FasterSAM* and *Canny*) of the contour detection module were used.

2) Evaluating Performance of Pixel Detection Module

In the proposed pose detection pipeline, a machine learning model was trained and used in the pixel detection module. The performance of this module was evaluated using two metrics: (a) Mean Euclidean distance, computed during the validation (827 frames) and testing (827 frames) phases using the training dataset detailed in Section III-B1 (Fig. 2d), and (b) the detection rate, assessed using the experimental dataset detailed in Section III-D1 (Fig. 5a). The Mean Euclidean distance measured the average pixel-wise distance between the ground truth and the predicted positions of $\mathbf{X}_i(t)$, where $1 \leq i \leq 5$. The detection rate was defined as the percentage of trials in which at least two points $\mathbf{X}_i(t)$ were successfully detected by the module when tested on the experimental dataset.

IV. RESULTS AND DISCUSSION

A. EXPERIMENTAL RESULTS

The results of the studies conducted using the experimental setup described in Sections III-D1 and III-D2 are presented in following sections: IV-A1 and IV-A2, respectively.

1) Performance of the Pose Detection Pipeline

Table 1 and Fig. 7, Fig. 8, and Fig. 9 present the performance of the proposed pose detection pipeline across different tissue backgrounds and the paths traversed by the tooltips. The performance is measured in terms of average positional errors (mm), orientation errors (°), and computation time for Algorithm 1 by varying parameters $\lambda_{shift\text{-}step}$, $\theta_{rotate\text{-}step}$, and the variants used in the contour detection module. The following paragraphs provide a summary of the results.

- **Positional Errors:** A slight increase in positional error was observed for the proposed pose detection pipeline as $\lambda_{shift\text{-}step}$ and $\theta_{rotate\text{-}step}$ increased along the X, Y, and Z directions. Additionally, the *FasterSAM variant* had slightly higher positional errors (X-direction range: 0.85 mm to 1.05 mm; Y-direction range: 1.05 mm to 1.19 mm; and Z-direction range: 4.28 mm to 5.55 mm) compared to the Canny variant (X-direction range: 0.67 mm to 1.00 mm; Y-direction range: 0.85 mm to 1.20 mm; and Z-direction range: 3.58 mm to 5.70 mm). The range of positional errors was higher in the Z-direction. This occurred because the Z-coordinate of $\mathbf{X}_{Tip}(t)$ depends on the line segments representing the left and right sides of the surgical tool shaft in $\mathbf{F}'(t)$. Due to the dark color of the instrument shaft, shadows may mistakenly be included as a part of the shaft, leading to an overestimation of its width. As a result, the estimated line segments appeared wider than they actually were, causing deviations in the Z-coordinate during 3D pose estimation.

TABLE 1. Performance of the proposed pose detection pipeline with different parameter settings

Contour Detection	$\lambda_{shift\text{-}step}$	$\theta_{rotate\text{-}step}$	X (mm)	Y (mm)	Z (mm)	Orient. Error (°)	Alg. 1 Time (ms)
Faster SAM Variant (Time = 91ms)	1	0.1	0.85 ± 0.27	1.05 ± 0.25	4.28 ± 0.89	3.38 ± 1.55	413 ± 36
		0.5	0.87 ± 0.28	1.07 ± 0.25	4.44 ± 0.97	3.49 ± 1.50	100 ± 9
		1.0	0.91 ± 0.29	1.08 ± 0.25	4.60 ± 1.02	3.89 ± 1.45	55 ± 5
	2	0.1	0.86 ± 0.27	1.06 ± 0.26	4.33 ± 0.91	3.42 ± 1.50	226 ± 16
		0.5	0.90 ± 0.28	1.09 ± 0.26	4.63 ± 0.99	3.67 ± 1.42	54 ± 3
		1.0	0.97 ± 0.31	1.15 ± 0.26	5.12 ± 1.12	4.16 ± 1.35	29 ± 2
	3	0.1	0.87 ± 0.27	1.07 ± 0.27	4.40 ± 0.90	3.50 ± 1.49	156 ± 17
		0.5	0.94 ± 0.29	1.12 ± 0.26	4.85 ± 1.02	3.77 ± 1.34	39 ± 3
		1.0	1.05 ± 0.31	1.19 ± 0.26	5.55 ± 1.19	4.61 ± 1.31	20 ± 2
Canny Variant (Time = 24ms)	1	0.1	0.67 ± 0.35	0.85 ± 0.32	3.58 ± 1.68	5.17 ± 3.39	386 ± 25
		0.5	0.72 ± 0.36	0.88 ± 0.32	3.81 ± 1.61	5.65 ± 3.19	88 ± 7
		1.0	0.82 ± 0.40	0.98 ± 0.36	4.33 ± 1.73	7.14 ± 3.23	45 ± 3
	2	0.1	0.68 ± 0.35	0.86 ± 0.32	3.64 ± 1.62	5.24 ± 3.28	206 ± 15
		0.5	0.80 ± 0.39	0.97 ± 0.36	4.29 ± 1.69	6.80 ± 3.30	45 ± 3
		1.0	0.94 ± 0.43	1.12 ± 0.41	5.19 ± 1.93	9.00 ± 4.40	26 ± 2
	3	0.1	0.69 ± 0.35	0.87 ± 0.32	3.69 ± 1.60	5.40 ± 3.24	137 ± 11
		0.5	0.87 ± 0.42	1.06 ± 0.40	4.78 ± 1.86	7.99 ± 3.97	32 ± 3
		1.0	1.00 ± 0.45	1.20 ± 0.41	5.70 ± 1.72	10.58 ± 5.05	17 ± 2

TABLE 2. Mean Euclidean distance (in pixels) between predicted and ground truth position of $\mathbf{x}_i(t)$

Phase	Platform	$\mathbf{X}_1(t)$	$\mathbf{X}_2(t)$	$\mathbf{X}_3(t)$	$\mathbf{X}_4(t)$	$\mathbf{X}_5(t)$
Validation	PyTorch	1.23 ± 1.17	1.35 ± 1.26	1.43 ± 1.39	1.62 ± 1.66	1.75 ± 1.71
	ONNX	2.75 ± 1.56	3.16 ± 1.99	3.58 ± 2.95	3.76 ± 2.00	3.53 ± 2.15
Testing	PyTorch	1.18 ± 1.26	1.29 ± 1.31	1.30 ± 1.17	1.48 ± 1.36	1.68 ± 1.65
	ONNX	2.81 ± 1.56	3.19 ± 2.07	3.43 ± 2.99	3.46 ± 1.74	3.31 ± 1.83

- **Orientation Errors:** The orientation errors of the proposed pose detection pipeline varied between 3.38° to 4.61° for the *FasterSAM variant*, and 5.17° to 10.58° for the *Canny variant*. The *FasterSAM variant* exhibited lower orientation errors than the *Canny variant*, suggesting a better accuracy for orientation detection. The orientation errors depend on the accurate computation of $\mathbf{n}_1(t)$ from the unit vectors $\mathbf{n}_L(t)$ and $\mathbf{n}_R(t)$, which are in turn dependent on accurate computation of the line segments representing the left and right sides of the surgical tool shaft. The *FasterSAM variant* generates a single contour around the tool shaft, while the *Canny variant* often produces contours due to the similar colors of the background tissue or shadows casted by the tool shaft. Consequently, the orientation errors are lower for the *FasterSAM variant*.
- **Algorithm 1 Computational Time:** The computational time of Algorithm 1 in the *edge detection module* varied, depending on the selection of values for the parameters $\lambda_{shift\text{-}step}$ and $\theta_{rotate\text{-}step}$. As $\lambda_{shift\text{-}step}$ increases from 1 to 3 and $\theta_{rotate\text{-}step}$ increases from 0.1 to 1.0, the search space gets reduced, leading to a reduction in computational time. This was observed irrespective of the *contour detection module* variant (*Canny* or *FasterSAM*). For a given set of parameter values for $\lambda_{shift\text{-}step}$ and $\theta_{rotate\text{-}step}$, the *Canny variant* of the *contour detection module* took less computational time than the *FasterSAM variant*. This is because *Canny* uses small convolutional kernels with a fixed sequence of operations, while *FasterSAM*, utilizing visual transformers, performs intensive matrix multiplications.

The average computational times for the *pixel detection module* was 32±1.9 ms, *contour detection module* was 24.11±0.8 ms, *edge detection module* was 88±7 ms, and both *tip detection module* and *3D pose estimation module* was less than 1 ms. Fig. 6 presents a timeline diagram illustrating the processing performed by each module. The diagram is not an absolute representation of the processing time, but assists in understanding how the threaded modules communicate, and the delays are introduced. In the pipeline, the *edge detection module* takes the maximum time. The cumulative end-to-end latency for the pipeline to process an input frame and generate output is ∼144 ms. Some of the intermediate input frames get overwritten, and thus, only a selected few input frames are processed. Once the pipeline reaches steady operation, its end-to-end computation time is bounded by its slowest module (∼88 ms), hence maintaining a sustained frame rate of ∼11.4 fps.

Based on the aforementioned results, we choose our pipeline configurations as *Canny* variant of *contour detection module*, at $\lambda_{shift\text{-}step}$=1 and $\lambda_{rotate\text{-}step}$=0.5, as this configuration yielded the lowest 3D pose estimation errors. Although the *FasterSAM* variant showed lower orientation errors, its prediction latency (91 ms) is much higher than *Canny* variant

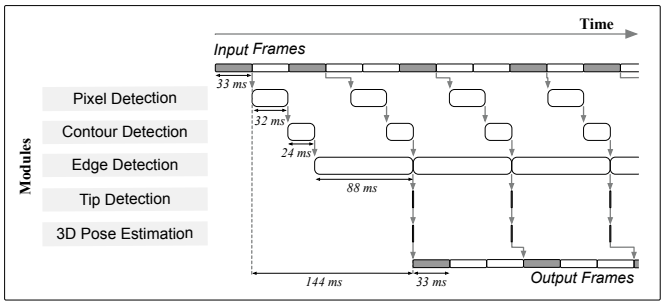

FIGURE 6. **A timeline diagram depicting timings and processing performed by each module within the proposed pipeline.**

(24 ms) creating an undesirable bottleneck for 3D pose estimation.

Apart from average 3D pose errors, we also report 95th percentile (P95) errors to further evaluate the pipeline's worst-case behavior. Under the best configuration, X and Y directions showed P95 errors of 1.29 mm and 1.28 mm, respectively, indicating that 95% of estimation errors are less than 2 mm on these axes. The Z-direction P95 error was 6.98 mm, indicating that depth-estimation can reach approximately 7 mm in the worst-case.

2) Performance of the Pixel Detection Module

The trained model was deployed on both PyTorch and ONNX platforms, and the Mean Euclidean distance was computed for each using the training dataset (Table 2). A mean Euclidean distance of smaller than 2 pixels on PyTorch and smaller than 4 pixels on ONNX was recorded. This indicates that the model was effectively trained and maintained a low error rate even on unseen data. The average inference time for the pixel detection module was 7.26 ± 0.76 ms using the PyTorch implementation and 18 ± 1 ms using the ONNX implementation. PyTorch consistently outperformed ONNX across all positions and both phases. The ONNX platform, although optimized for real-time performance, showed a higher Mean Euclidean distance due to minor approximation introduced during the conversion from the original PyTorch implementation.

TABLE 3. **Detection rate (in percentage) of the pixel detection module**

Tissue Background	Circular	Sinusoidal	Spiral
Suturing pad	95%	99%	93%
Cholecystectomy	92%	99%	87%
Abdomen	84%	77%	69%

High detection rates were observed when the Suturing Pad and Cholecystectomy phantom were used as backgrounds (Table 3). This is likely because these backgrounds closely resembled the conditions present in the training dataset. In contrast, the detection rate was lower when the Abdomen phantom was used as the background, suggesting reduced model performance in less familiar visual contexts. Overall, an average detection rate of 90% on unseen data demonstrates that our model is robust to unseen tissues and tool motions.

When trained on videos from two scopes and evaluated on a held-out scope, the model exhibits higher pixel-detection errors ($\sim$6.3 px) while maintaining a high detection rate (99%). This emphasizes the importance of all three scopes in improved pixel detection, while suggesting that cross-scope generalizability may be limited using this model. Nevertheless, the remaining modules in our pipeline are designed to accommodate detection errors and deviations, by considering the detections only as an initialization, followed by further processing to mitigate its effects on the final 3D pose estimation.

With five points detected by the *pixel detection module*, our pipeline estimates 3D pose with average errors of 0.72 mm, 0.88 mm and 3.81 mm on the X, Y and Z directions, respectively, and an average orientation error of 5.65° about the tool's shaft axis. For the same configuration, when the detected points were reduced to two, the average errors increased by 0.38 mm (X), 0.034 mm (Y), and 2.56° (orientation), while the Z-direction positional error reduced slightly by 0.25 mm. These changes indicate that the default design with five midline points improves 3D pose estimation of the laparoscopic instrument. As another variation of the *pixel detection module*, we considered using a YOLO-pose model instead of YOLOv7. YOLO-pose yields only two keypoints for the laparoscopic shaft: (a) a point where the metallic portion of the tool-tip connects to the shaft, and (b) a pivot point about which the jaw-tips rotate. These two points proved insufficient for robust 3D pose estimation compared with the five points predicted by YOLOv7.

B. COMPARISON WITH EXISTING ALGORITHMS

The comparison of our work with existing works in similar directions is summarized in Table 4. Allan et al. [33] utilized Random Forests and 3D model alignment to estimate the 3D poses of laparoscopic instruments, but reported more than 4 seconds of computational time per frame. In their subsequent work [53], they improved the results, but the computational time per frame remained high, on average over 1 second per frame. A further improved version of their work [34] reduced the processing time to 830 ms with an improved accuracy, but still lacked real-time capability. Moreover, their work focused on robotic articulated instruments and required prior availability of their 3D CAD models. Augustinos et al. [54] employed image processing techniques along with known information about the 3D insertion point for 3D pose estimation, but the errors along the Z-direction were high (about 7.24 mm), and the latency of their system was not reported. Zhao et al. [38] used CNNs combined with the known 3D insertion point to estimate the 3D pose of laparoscopic instruments, reporting very low estimation errors along the Z-direction (about 0.35 mm). However, measuring the 3D position of the insertion point required additional sensors. The work most relevant to ours, by Hasan et al. [39], estimates 3D poses of both robotic and laparoscopic instruments using

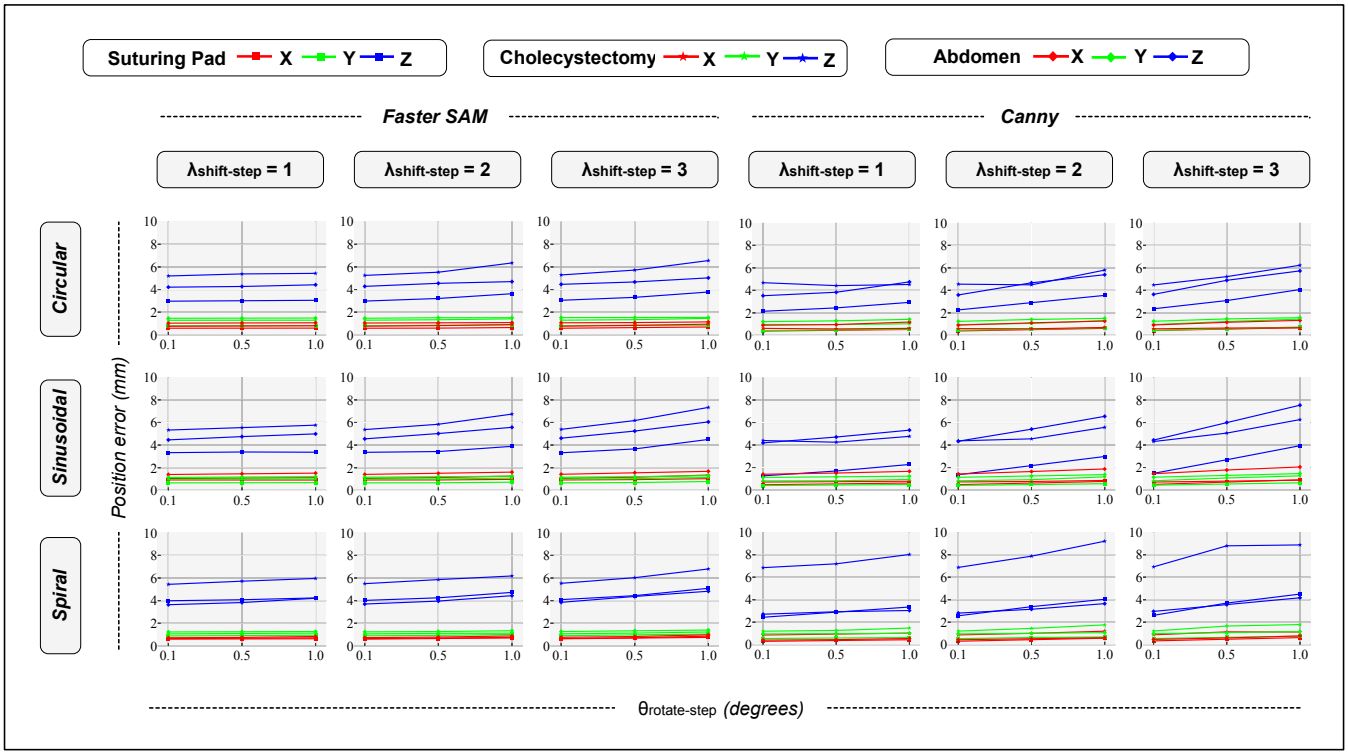

FIGURE 7. Variation in positional error (millimeters) by altering the parameters $\lambda_{\text{shift-step}}$, $\theta_{\text{rotate-step}}$, and contour detection module variant (*FasterSAM* or *Canny*) in the proposed pose detection pipeline. The analysis was performed across the three paths (spiral, sinusoidal, circular) traversed by the tooltips over three different tissue backgrounds (suturing pad, cholecystectomy, abdomen).

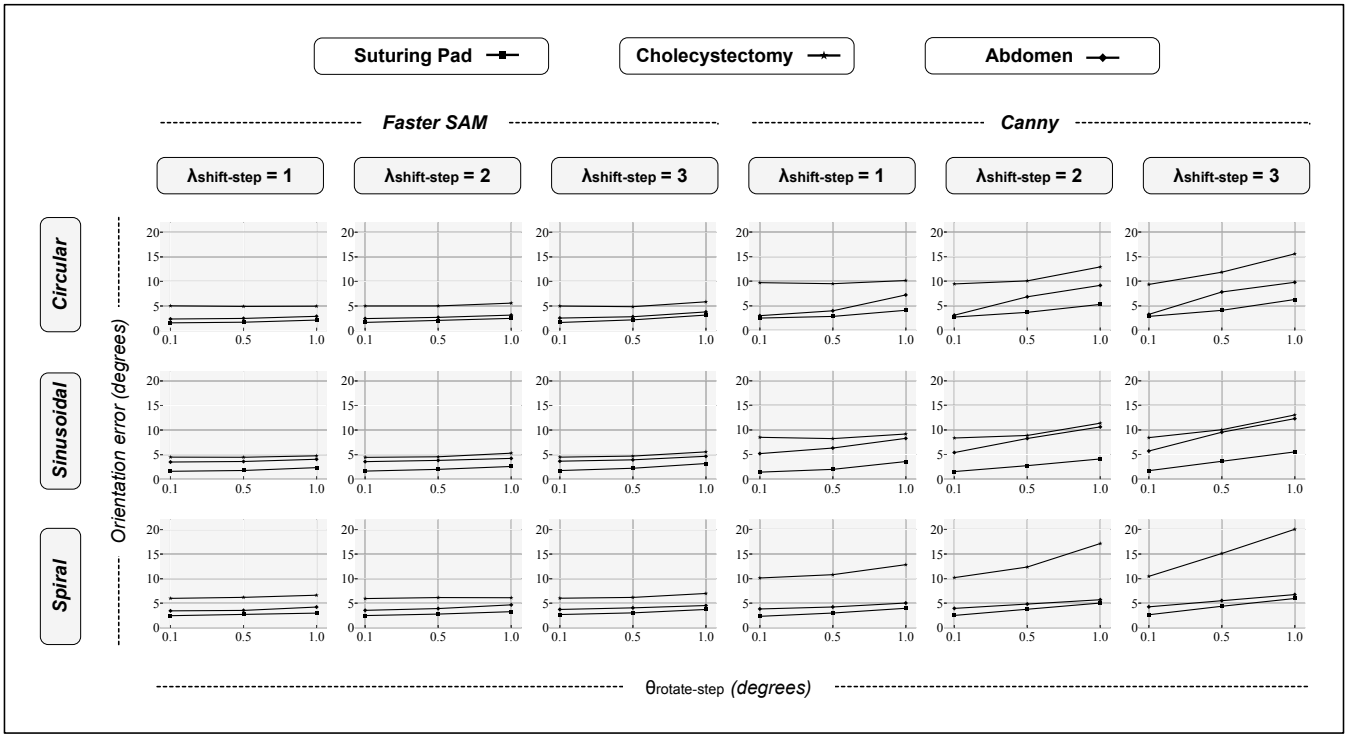

FIGURE 8. Variation in orientation error (degrees) by altering the parameters $\lambda_{\text{shift-step}}$, $\theta_{\text{rotate-step}}$, and contour detection module variant (*FasterSAM* or *Canny*) in the proposed pose detection pipeline. The analysis was performed across the three paths (spiral, sinusoidal, circular) traversed by the tooltips over three different tissue backgrounds (suturing pad, cholecystectomy, abdomen).

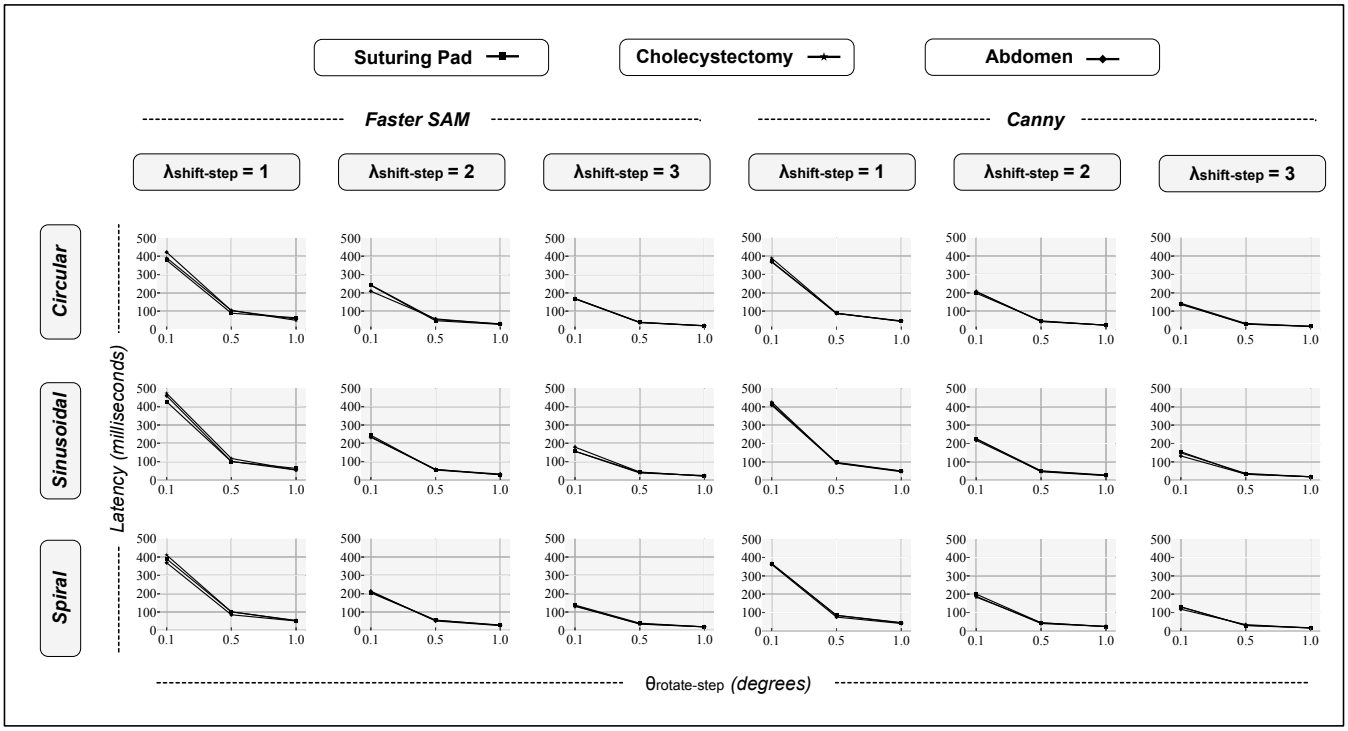

FIGURE 9. Variation in computation time (milliseconds) of Algorithm-1 by altering the parameters $\lambda_{\text{shift-step}}$, $\theta_{\text{rotate-step}}$, and contour detection module variant (*FasterSAM* or *Canny*) in the proposed pose detection pipeline. The analysis was performed across the three paths (spiral, sinusoidal, circular) traversed by the tooltips over three different tissue backgrounds (suturing pad, cholecystectomy, abdomen).

TABLE 4. 3D Pose Estimation Methods

Existing Work	L	R	I	Techniques	X (mm)	Y (mm)	Z (mm)	Orientation Error (°)	Time (ms)
Allan et al. [33]	✓			Random Forests	1.73	1.89	9.86	–	>4000
Allan et al. [53]	✓	✓		Random Forests	1.09	0.59	7.48	10.0	>1000
Allan et al. [34]		✓		Random Forests	0.70	0.50	4.09	4.58	830
Augustinos et al. [54]	✓		✓	NA (Image processing)	1.79	2.42	7.24	–	–
Zhao et al. [38]	✓		✓	CNN	2.57	1.97	0.35	–	–
Hasan et al. [39]	✓	✓		CNN	1.87	0.70	4.80	5.94	–
Proposed	✓			YOLO	0.72	0.88	3.81	5.65	144

L: Laparoscopy Surgery, R: Robotic Surgery, I: Insertion point needed

DL techniques. However, the reported values were based on a subset of 32 in-vivo frames, and they did not evaluate real-time performance of the system. In contrast, our work evaluates 3D pose estimation on 4,581 frames, demonstrating the lowest reported latency of 144 ms, lowest error of 3.81 mm in the Z-direction and an improved orientation error of 5.65°.

Since the proposed pipeline focuses on estimating the 3D pose of laparoscopic instrument tooltip primarily from the shaft, several existing works in the field of robotic surgery can also be considered for comparison. In robotic surgery, the error is reported as Root Mean Squared Error (RMSE) by measuring the difference between predicted and actual values along a series of points on the articulated robotic tooltip. Ye et al. [55] reported an extremely low 3D Pose estimation RMSE of 2.83 mm for translation and 7.4° for orientation of robotic instruments, with demonstrated real-time performance of around 40 ms. However, their work requires the 3D model of the tool and kinematic data from the instrument's robotic manipulator, limiting their application to laparoscopic procedures. Additionally, their system relies on features extracted from the instrument's appearance, specific to robotic articulated instruments, which is not applicable to laparoscopic instruments. Another work by Allan et al. [56] utilized Random Forests and a 3D model of the instrument to estimate its 3D pose, reporting an RMSE of 5.07 mm translation and 24° for orientation, but with a latency higher than 3 seconds per frame. Moreover, their system utilized region-based estimation, where both metallic and shaft portions of the instrument contributed to the estimation, limiting its applicability to laparoscopic instruments. Recently, Fan et al. [30] used DL and reinforcement learning techniques to estimate the 3D pose of articulated robotic instruments, reporting the lowest RMSE of 2.81 mm and low latency. However, their work requires the 3D model of the instru-

ment, and the RMSE results were reported on synthetic data generated using the dVRK simulator [57]. Additionally, the 2D key points detected by their system depend on features surrounding the metallic joints of the articulated instrument tip, limiting comparability to laparoscopic instruments.

C. LIMITATIONS

The background tissues in our dataset are limited to synthetic phantoms, which means the pose estimation pipeline's performance on real surgical videos cannot be assessed. As our system was designed and tested for box-training and simulated environments, its applicability to live surgeries is currently not feasible. However, in the future, the pipeline can be trained by chroma-keying the monochromatic backgrounds in video frames to substitute actual intra-operative video footage. Such surgical videos can be recorded during the procedure by withdrawing the laparoscopic instruments and manoeuvring only the scope to capture the operative field from various perspectives. The pipeline can also be extended to other training environments with live animals or cadavers. In such cases, the robotic setup is still applicable for collecting training data. The modules trained on such dataset may have the potential to address challenging conditions that are prevalent in in-vivo scenes, like real blood, tissue, and smoke not present in box-trainer environments.

Our dataset did not account for common challenging conditions that occur within laparoscopic videos, like motion blur, tool occlusion, smoke, and uneven illumination. Most of the challenges could be addressed by ensuring availability of such data during ML training. Motion blur could be induced by varying the robotic manipulator's end-effector speed, and occlusion of instrument tooltips could be introduced by modifying the setup to partially or heavily occlude the tooltips. To further ensure that such conditions do not impact the subsequent modules in our pipeline, a pre-processing stage can be introduced that detects distortions and helps the pipeline to adapt dynamically. A lightweight distortion classification model similar to [58] could be used to detect whether input frames are distorted, followed by recognition of the type of distortion, and a ranking on its severity [59]. This would allow the pipeline to filter input frames and either perform targeted corrections or reject severely distorted images. Laparoscopic training environments do not typically encounter distortions pertaining to heavy smoke or fog, however, occurance of light smoke and noise can be rectified using desmoking techniques utilizing CycleGAN [60], probabilistic graphical models [61], or texture-preserving temporal smoothing [62]. Although our work has introduced rigorous augmentations to overcome illumination challenges, severe cases of low-illuminance could be detected using metrics based on no-reference background illuminance [63] and can be corrected using edge-preserving illumination equalization and color correction techniques [64]. Furthermore, a combination of distortion detection techniques and confidence dips within our ML models can be used to trigger uncertainty-aware thresholding for Canny edge-detector. In severe cases where input frames are rejected, the last reliable output can be used in a hold-last-good manner to maintain a stable and adaptive pipeline.

Our current work does not incorporate errors from the cylindrical marker used to extract ground-truth. As our results are referenced to this marker, they may include positive or negative deviations from the reported 3D errors. Moreover, our work does not estimate the instrument's jaw angle or its rotation around the shaft axis. As part of future work, we plan to use the same robotic setup to generate a dataset that includes these parameters and subsequently extend the pipeline to estimate both jaw angle and axial rotation.

V. CONCLUSION

In this work, we present a modular pipeline for real-time, marker-less 3D pose estimation of laparoscopic instruments in simulated training scenarios. To enable training of the DL modules within the pipeline, an automated dataset generation method is introduced. Additionally, to ensure real-time performance, the pipeline incorporates a parallelized GPU-based processing module. The pipeline extracts geometric features of the instrument using context-aware DL techniques, followed by 3D-pose computation. The system estimates the 3D-pose of instrument tooltip with low errors of 0.72 mm, 0.88 mm, and 3.81 mm along the X, Y, and Z directions, and 5.65°of instrument-axis orientation error with 144 ms end-to-end latency and a sustained frame rate of 11.4 fps. Thus, the work addresses the lack of non-hardware-based, marker-less 3D pose estimation techniques for laparoscopic instruments, primarily due to the limited availability of ground-truth data in 3D space. Moreover, our method achieves real-time performance with accuracy comparable to state-of-the-art approaches. Our work forms a significant step toward automated skill assessment with the potential to improve laparoscopic training.

REFERENCES

[1] W. Lee, C. Chan, and B. Wang, "Recent advances in laparoscopic surgery," *Asian J. Endosc. Surg.*, vol. 6, no. 1, pp. 1–8, 2013.
[2] D. Bouget, M. Allan, D. Stoyanov, and P. Jannin, "Vision-based and marker-less surgical tool detection and tracking: a review of the literature," *Med. Image Anal.*, vol. 35, pp. 633–654, 2017.
[3] B. Levy and M. Mobasheri, "Principles of safe laparoscopic surgery," *Surgery*, vol. 35, no. 4, pp. 216–219, 2017.
[4] D. T. Harrington, G. Roye, B. A. Ryder, T. J. Miner, P. Richardson, and W. G. Cioffi, "A time-cost analysis of teaching a laparoscopic entero-enterostomy," *J. Surg. Educ.*, vol. 64, no. 6, pp. 342–345, 2007.
[5] R. M. Viglialoro, N. Esposito, S. Condino, F. Cutolo, S. Guadagni, M. Gesi, M. Ferrari, and V. Ferrari, "Augmented reality to improve surgical simulation: Lessons learned towards the design of a hybrid laparoscopic simulator for cholecystectomy," *IEEE Trans. Biomed. Eng.*, vol. 66, no. 7, pp. 2091–2104, 2018.
[6] B. Zendejas, R. Brydges, S. J. Hamstra, and D. A. Cook, "State of the evidence on simulation-based training for laparoscopic surgery: a systematic review," *Ann. Surg.*, vol. 257, no. 4, pp. 586–593, 2013.
[7] E. Yiannakopoulou, N. Nikiteas, D. Perrea, and C. Tsigris, "Virtual reality simulators and training in laparoscopic surgery," *Int. J. Surg.*, vol. 13, pp. 60–64, 2015.
[8] R. Campo, A. Wattiez, R. L. De Wilde, and C. R. M. Sanabria, "Training in laparoscopic surgery: from the lab to the or," *Slovenian Journal of Public Health*, vol. 51, no. 4, pp. 285–298, 2012.

[9] A. Supe, R. Prabhu, I. Harris, S. Downing, and A. Tekian, "Structured training on box trainers for first year surgical residents: does it improve retention of laparoscopic skills? a randomized controlled study," *J. Surg. Educ.*, vol. 69, no. 5, pp. 624–632, 2012.

[10] S. M. Lundberg, B. Nair, M. S. Vavilala, M. Horibe, M. J. Eisses, T. Adams, D. E. Liston, D. K.-W. Low, S.-F. Newman, J. Kim, and S.-I. Lee, "Explainable machine-learning predictions for the prevention of hypoxaemia during surgery," *Nature Biomed. Eng.*, vol. 2, no. 10, p. 749–760, 2018.

[11] K. Ahmed, D. Miskovic, A. Darzi, T. Athanasiou, and G. B. Hanna, "Observational tools for assessment of procedural skills: a systematic review," *Amer. J. Surg.*, vol. 202, no. 4, pp. 469–480. e6, 2011.

[12] M. J. Fard, S. Ameri, R. Darin Ellis, R. B. Chinnam, A. K. Pandya, and M. D. Klein, "Automated robot-assisted surgical skill evaluation: Predictive analytics approach," *Int. J. Med. Robot. Comput. Assist. Surg.*, vol. 14, no. 1, p. e1850, 2018.

[13] A. Zia, Y. Sharma, V. Bettadapura, E. L. Sarin, and I. Essa, "Video and accelerometer-based motion analysis for automated surgical skills assessment," *Int. J. Comput. Assist. Radiol. Surg.*, vol. 13, pp. 443–455, 2018.

[14] K. Ebina, T. Abe, K. Hotta, M. Higuchi, J. Furumido, N. Iwahara, M. Kon, K. Miyaji, S. Shibuya, Y. Lingbo, S. Komizunai, Y. Kurashima, H. Kikuchi, R. Matsumoto, T. Osawa, S. Murai, T. Tsujita, K. Sase, X. Chen, ..., and N. Shinohara, "Objective evaluation of laparoscopic surgical skills in wet lab training based on motion analysis and machine learning," *Langenbeck's Arch. Surg.*, vol. 407, no. 5, pp. 2123–2132, 2022.

[15] V. Lahanas, C. Loukas, and E. Georgiou, "A simple sensor calibration technique for estimating the 3d pose of endoscopic instruments," *Surg. Endosc.*, vol. 30, pp. 1198–1204, 2016.

[16] S. Yamaguchi, D. Yoshida, H. Kenmotsu, T. Yasunaga, K. Konishi, S. Ieiri, H. Nakashima, K. Tanoue, and M. Hashizume, "Objective assessment of laparoscopic suturing skills using a motion-tracking system," *Surg. Endosc.*, vol. 25, pp. 771–775, 2011.

[17] G. Xiao, E. Bonmati, S. Thompson, J. Evans, J. Hipwell, D. Nikitichev, K. Gurusamy, S. Ourselin, D. J. Hawkes, and B. Davidson, "Electromagnetic tracking in image-guided laparoscopic surgery: comparison with optical tracking and feasibility study of a combined laparoscope and laparoscopic ultrasound system," *Med. Phys.*, vol. 45, no. 11, pp. 5094–5104, 2018.

[18] C. Heiliger, D. Andrade, C. Geister, A. Winkler, K. Ahmed, A. Deodati, V. H. E. v. Treuenstätt, J. Werner, A. Eursch, and K. Karcz, "Tracking and evaluating motion skills in laparoscopy with inertial sensors," *Surg. Endosc.*, vol. 37, no. 7, pp. 5274–5284, 2023.

[19] L. H. Olivas-Alanis, R. A. Calzada-Briseño, V. Segura-Ibarra, E. V. Vázquez, J. A. Diaz-Elizondo, E. Flores-Villalba, and C. A. Rodriguez, "Lapkaans: tool-motion tracking and gripping force-sensing modular smart laparoscopic training system," *Sensors*, vol. 20, no. 23, p. 6937, 2020.

[20] I. Oropesa, T. de Jong, P. Sánchez-González, J. Dankelman, and E. Gómez, "Feasibility of tracking laparoscopic instruments in a box trainer using a leap motion controller," *Measurement*, vol. 80, pp. 115–124, 2016.

[21] M. Owlia, M. Khabbazan, M. M. Mirbagheri, and A. Mirbagheri, "Real-time tracking of laparoscopic instruments using kinect for training in virtual reality," in *Annu. Int. Conf. IEEE Eng. Med. Biol. Soc.* IEEE, 2016, Conference Proceedings, pp. 3945–3948.

[22] K. R. Van Sickle, D. M. Iii, A. Gallagher, and C. Smith, "Construct validation of the promis simulator using a novel laparoscopic suturing task," *Surg. Endosc.*, vol. 19, pp. 1227–1231, 2005.

[23] N. Bandari, J. Dargahi, and M. Packirisamy, "Tactile sensors for minimally invasive surgery: A review of the state-of-the-art, applications, and perspectives," *IEEE Access*, vol. 8, pp. 7682–7708, 2020.

[24] Z. Zhao, "Real-time 3d visual tracking of laparoscopic instruments for robotized endoscope holder," *Biomed. Mater. Eng.*, vol. 24, no. 6, pp. 2665–2672, 2014.

[25] M. Carletti, D. Zerbato, A. Calanca, and P. Fiorini, "Robust 3d pose estimation of a laparoscopic instrument with three landmarks," in *Proceedings of the Smart Tools & Apps for Graphics (STAG)*, 2015, Conference Proceedings, pp. 7–14.

[26] B. Gautier, H. Tugal, B. Tang, G. Nabi, and M. S. Erden, "Real-time 3d tracking of laparoscopy training instruments for assessment and feedback," *Front. Robot. AI*, vol. 8, p. 751741, 2021.

[27] J. Cartucho, C. Wang, B. Huang, D. S. Elson, A. Darzi, and S. Giannarou, "An enhanced marker pattern that achieves improved accuracy in surgical tool tracking," *Comput. Methods Biomech. Biomed. Eng. Imaging Vis.*, vol. 10, no. 4, pp. 400–408, 2022.

[28] L. Zhang, M. Ye, P.-L. Chan, and G.-Z. Yang, "Real-time surgical tool tracking and pose estimation using a hybrid cylindrical marker," *Int. J. Comput. Assist. Radiol. Surg.*, vol. 12, pp. 921–930, 2017.

[29] F. Bagheri, R. Daneshmand, B. Taghibeyglou, and H. Azarnoush, "Semi-automatic 3-d pose estimation of laparoscopic tools to generate 3-d labeled database by developing a graphical user interface," in *Iran. Conf. Biomed. Eng.* IEEE, 2020, Conference Proceedings, pp. 226–233.

[30] K. Fan, Z. Chen, Q. Liu, G. Ferrigno, and E. De Momi, "A reinforcement learning approach for real-time articulated surgical instrument 3d pose reconstruction," *IEEE Trans. Med. Robot. Bion.*, 2024.

[31] R. Dockter, R. Sweet, and T. Kowalewski, "A fast, low-cost, computer vision approach for tracking surgical tools," in *IEEE Int. Conf. Intell. Robots Syst.* IEEE, 2014, Conference Proceedings, pp. 1984–1989.

[32] I. Oropesa, P. Sánchez-González, M. K. Chmarra, P. Lamata, A. Fernández, J. A. Sánchez-Margallo, F. W. Jansen, J. Dankelman, F. M. Sánchez-Margallo, and E. J. Gómez, "Eva: laparoscopic instrument tracking based on endoscopic video analysis for psychomotor skills assessment," *Surg. Endosc.*, vol. 27, pp. 1029–1039, 2013.

[33] M. Allan, S. Ourselin, S. Thompson, D. J. Hawkes, J. Kelly, and D. Stoyanov, "Toward detection and localization of instruments in minimally invasive surgery," *IEEE Trans. Biomed. Eng.*, vol. 60, no. 4, pp. 1050–1058, 2012.

[34] M. Allan, P.-L. Chang, S. Ourselin, D. J. Hawkes, A. Sridhar, J. Kelly, and D. Stoyanov, "Image based surgical instrument pose estimation with multi-class labelling and optical flow," in *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Springer International Publishing, Cham, 2015, Conference Proceedings, p. 331–338.

[35] I. Rivas-Blanco, C. J. Pérez-Del-Pulgar, I. García-Morales, and V. F. Muñoz, "A review on deep learning in minimally invasive surgery," *IEEE Access*, vol. 9, pp. 48 658–48 678, 2021.

[36] D. Sarikaya, J. J. Corso, and K. A. Guru, "Detection and localization of robotic tools in robot-assisted surgery videos using deep neural networks for region proposal and detection," *IEEE Trans. Med. Imag.*, vol. 36, no. 7, pp. 1542–1549, 2017.

[37] Y. Liu, Z. Zhao, P. Shi, and F. Li, "Towards surgical tools detection and operative skill assessment based on deep learning," *IEEE Trans. Med. Robot. Bion.*, vol. 4, no. 1, pp. 62–71, 2022.

[38] Z. Zhao, S. Voros, Y. Weng, F. Chang, and R. Li, "Tracking-by-detection of surgical instruments in minimally invasive surgery via the convolutional neural network deep learning-based method," *Comput. Assist. Surg.*, vol. 22, no. sup1, pp. 26–35, 2017.

[39] M. K. Hasan, L. Calvet, N. Rabbani, and A. Bartoli, "Detection, segmentation, and 3d pose estimation of surgical tools using convolutional neural networks and algebraic geometry," *Med. Image Anal.*, vol. 70, p. 101994, 2021.

[40] B. G. Gerats, J. M. Wolterink, S. P. Mol, and I. A. Broeders, "Neural fields for 3d tracking of anatomy and surgical instruments in monocular laparoscopic video clips," *Healthcare Technol. Lett.*, vol. 11, no. 6, pp. 411–417, 2024.

[41] Q. Wang, Y.-Y. Chang, R. Cai, Z. Li, B. Hariharan, A. Holynski, and N. Snavely, "Tracking everything everywhere all at once," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2023, Conference Proceedings, pp. 19 795–19 806.

[42] T. Kurmann, P. Marquez Neila, X. Du, P. Fua, D. Stoyanov, S. Wolf, and R. Sznitman, "Simultaneous recognition and pose estimation of instruments in minimally invasive surgery," in *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Springer, 2017, Conference Proceedings, pp. 505–513.

[43] A. Jin, S. Yeung, J. Jopling, J. Krause, D. Azagury, A. Milstein, and L. Fei-Fei, "Tool detection and operative skill assessment in surgical videos using region-based convolutional neural networks," in *IEEE winter conference on applications of computer vision (WACV)*. IEEE, 2018, Conference Proceedings, pp. 691–699.

[44] C. I. Nwoye, D. Mutter, J. Marescaux, and N. Padoy, "Weakly supervised convolutional lstm approach for tool tracking in laparoscopic videos," *Int. J. Comput. Assist. Radiol. Surg.*, vol. 14, pp. 1059–1067, 2019.

[45] A. P. Twinanda, S. Shehata, D. Mutter, J. Marescaux, M. De Mathelin, and N. Padoy, "Endonet: a deep architecture for recognition tasks on laparoscopic videos," *IEEE Trans. Med. Imag.*, vol. 36, no. 1, pp. 86–97, 2016.

[46] W.-Y. Hong, C.-L. Kao, Y.-H. Kuo, J.-R. Wang, W.-L. Chang, and C.-S. Shih, "Cholecseg8k: a semantic segmentation dataset for laparoscopic cholecystectomy based on cholec80," *arXiv preprint arXiv:2012.12453*, 2020.

[47] S. Bodenstedt, S. Speidel, M. Wagner, J. Chen, A. Kisilenko, B. Müller, L. Maier-Hein, B. Oliveira, S. Hong, and J. Zamora-Anaya, "Heichole surgical workflow analysis and full scene segmentation (heisurf)," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2021, Conference Proceedings.
[48] C. I. Nwoye, T. Yu, C. Gonzalez, B. Seeliger, P. Mascagni, D. Mutter, J. Marescaux, and N. Padoy, "Rendezvous: Attention mechanisms for the recognition of surgical action triplets in endoscopic videos," *Med. Image Anal.*, vol. 78, p. 102433, 2022.
[49] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, Conference Proceedings, pp. 7464–7475.
[50] N. Abdurahiman, J. Padhan, M. Khorasani, H. Zhao, V. M. Baez, A. Al-Ansari, A. T. Becker, and N. V. Navkar, "Interfacing mechanism for actuated maneuvering of articulated laparoscopes using head motion," in *IEEE International Conference on Mechatronics and Automation (ICMA)*. IEEE, 2023, Conference Proceedings, pp. 2289–2296.
[51] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European Conference on Computer Vision (ECCV)*. Springer International Publishing, Cham, 2014, Conference Proceedings, pp. 740–755.
[52] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, no. 6, pp. 679–698, 1986.
[53] M. Allan, S. Thompson, M. J. Clarkson, S. Ourselin, D. J. Hawkes, J. Kelly, and D. Stoyanov, "2d-3d pose tracking of rigid instruments in minimally invasive surgery," in *Information Processing in Computer-Assisted Interventions: 5th International Conference (IPCAI)*. Springer, 2014, Conference Proceedings, pp. 1–10.
[54] A. Agustinos and S. Voros, "2d/3d real-time tracking of surgical instruments based on endoscopic image processing," in *International Workshop on Computer-Assisted and Robotic Endoscopy*, vol. 9515. Springer International Publishing, Cham, 2015, Conference Proceedings, pp. 90–100.
[55] M. Ye, L. Zhang, S. Giannarou, and G.-Z. Yang, "Real-time 3d tracking of articulated tools for robotic surgery," in *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Springer International Publishing, Cham, 2016, Conference Proceedings, pp. 386–394.
[56] M. Allan, S. Ourselin, D. J. Hawkes, J. D. Kelly, and D. Stoyanov, "3-d pose estimation of articulated instruments in robotic minimally invasive surgery," *IEEE Trans. Med. Imaging*, vol. 37, no. 5, pp. 1204–1213, 2018.
[57] K. Fan, A. Marzullo, N. Pasini, A. Rota, M. Pecorella, G. Ferrigno, and E. De Momi, "A unity-based da vinci robot simulator for surgical training," in *Proc. IEEE RAS EMBS Int. Conf. Biomed. Robot. Biomechatron*. IEEE, 2022, Conference Proceedings, pp. 1–6.
[58] N. Aldahoul, H. A. Karim, M. J. T. Tan, and J. L. Fermin, "Transfer learning and decision fusion for real time distortion classification in laparoscopic videos," *IEEE Access*, vol. 9, pp. 115 006–115 018, 2021.
[59] N. AlDahoul, H. A. Karim, M. A. Momo, M. J. T. Tan, and J. L. Fermin, "Encoding laparoscopic image to words using vision transformer for distortion classification and ranking in laparoscopic videos," *Multimedia Tools and Applications*, vol. 84, no. 10, pp. 7159–7181, 2025.
[60] V. Venkatesh, N. Sharma, V. Srivastava, and M. Singh, "Unsupervised smoke to desmoked laparoscopic surgery images using contrast driven cyclic-desmokegan," *Computers in Biology and Medicine*, vol. 123, p. 103873, 2020.
[61] A. Kotwal, R. Bhalodia, and S. P. Awate, "Joint desmoking and denoising of laparoscopy images," in *2016 IEEE 13th International Symposium on Biomedical Imaging (ISBI)*. IEEE, 2016, pp. 1050–1054.
[62] A. Baid, A. Kotwal, R. Bhalodia, S. Merchant, and S. P. Awate, "Joint desmoking, specularity removal, and denoising of laparoscopy images via graphical models and bayesian inference," in *2017 IEEE 14th international symposium on Biomedical imaging (ISBI 2017)*. IEEE, 2017, pp. 732–736.
[63] T.-S. Nguyen, J. Chaussard, M. Luong, H. Zaag, and A. Beghdadi, "A no-reference measure for uneven illumination assessment on laparoscopic images," in *2022 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2022, pp. 4103–4107.
[64] L. Wang, Q. Li, H. Yang, J. Huang, and K. Xu, "A sample-based color correction method for laparoscopic images," in *2020 IEEE International Conference on Real-time Computing and Robotics (RCAR)*. IEEE, 2020, pp. 446–451.

DEHLELA SHABIR received her M.Sc. in artificial intelligence from the University of St. Andrews, Scotland, United Kingdom, in 2020. She is currently pursuing a Ph.D. at Qatar University, Doha, Qatar.

She is also a Research Assistant with the AI Innovation Hub and Itqan Clinical Simulation & Innovation Center in Hamad Medical Corporation, Doha, Qatar. Her research interests and work focus on applications of computer vision, extended reality and artificial intelligence in the field of computer-assisted surgical technologies.

JHASKETAN PADHAN received his Master's degree in computer applications from Biju Patnaik University of Technology, India in 2004.

He has over 15 years of experience in design and development of medical devices and over 3 years of experience in medical technology research. He was previously affiliated with General Electric (GE) as a System Architect. He is currently working as a Clinical Research Specialist at Itqan Clinical Simulation and Innovation Center in Hamad Medical Corporation, Qatar.

NIHAL ABDURAHIMAN received his Master's degree in computer science from Hamad Bin Khalifa University (HBKU), Doha, Qatar, and his Bachelor's degree in computer science from Qatar University in 2022 & 2018, respectively.

He is currently a Research Assistant at Itqan Clinical Simulation and Innovation Center in Hamad Medical Corporation, Qatar. His work focuses on development of medical robots for surgical interventions.

JULIEN ABINAHED has a degree in Electrical & Mechanical engineering in 2003 from USJ University in Lebanon. He pursued a PhD in medical image computing at Imperial College London, UK in 2009.

He is currently a Senior Research Scientist at Itqan Clinical Simulation and Innovation Center, Hamad Medical Corporation, Qatar. He has research experience in biomedical applications at Ecole Centrale Paris, France, at Siemens Corporate Research in the US and at the Qatar Robotic Surgery Center, Qatar Foundation prior to joining HMC. His research has resulted in 40+ research publications and patents, and R&D output.

ZHIGANG DENG (Senior member, IEEE) earned his Ph.D. in computer science from the University of Southern California in 2006. Prior to that, he received his B.S. degree in mathematics from Xiamen University (China), and M.S. in computer science from Peking University (China).

He is a Moores Professor of Computer Science at University of Houston, Texas, USA. His research interests include computer graphics, computer animation, virtual humans, human computer conversation, and robotics. Besides serving as the conference general or program co-chairs for CASA 201, SCA 2015, MIG 2022, and PG 2023, Prof. Zhigang Deng has been an Associate Editor for IEEE Transactions on Visualization and Computer Graphics, Computer Graphics Forum, Computer Animation and Virtual Worlds Journal, etc. He is a Distinguished member of ACM and a Senior member of IEEE.

NIKHIL NAVKAR received his Ph.D. and Master's degree in computer science from University of Houston, Texas, United States, in 2013.

He was formerly a Research Executive in medical software engineering with Qatar Foundation's Qatar Robotic Surgery Center at Qatar Science & Technology Park (QSTP) where he worked on medical robotics projects aligned with the national research and health strategies. He is currently a Senior Research Scientist with the AI Innovation Hub and Itqan Clinical Simulation & Innovation Center at Hamad Medical Corporation, Qatar. His work focuses on the research, development, and innovation of medical technologies to solve unmet clinical needs. He has initiated, planned, and executed multi-institutional, multidisciplinary R&D projects across engineering and clinical teams. His research interests include human-computer-interaction for medical devices, computer simulations for preoperative planning and training, and algorithms for image-guided robotic interventions.