UNIVERSITY OF CALIFORNIA SAN DIEGO

Autonomous Maintenance of Hemostasis in Robotic Surgery

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy

in

Electrical Engineering (Intelligent Systems, Robotics and Control)

by

Florian Richter

2022

University of California San Diego

2022

DEDICATION

I dedicate this work to my family. My parents, Kerstin and Stefan Richter, who instilled intellectual curiosity and hard work in me and supply endless amounts of delicious food. My sister, Nora Richter, who assisted me with my writing and inspires me to travel. My brother, Felix Richter, who encouraged me develop medical technologies to help people and is my StarCraft companion. My partner, Yi (Flora) Liu, who supported me during my academic journey through discussions about my work, vacations when I needed a break, and providing variety that kept our day-to-day life exciting.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I thank my advisor, Dr. Michael C. Yip, for his continuous support and mentorship. He gave me the creative space and the necessary resources to succeed as a PhD student. I also learned a great deal from his guidance including technical and professional skills.

I appreciate the help and teachings from Dr. Nikolay Atanasov whose courses gave me a foundation for my PhD research.

I also appreciate the helpfullness of my committee members, Dr. Ryan K. Orosco, Dr. Henrik Christensen, and Dr. Behrouz Touri.

Furthemore, I thank Dr. Ryan K. Orosco for his medical guidance, the resources he supplied, and mentorship. He is a co-author of ten of my papers and supported me on countless experiments. I also express my deepest gratitude to Dr. Amir Degani, Dr. Won Seo Park, Dr. Philip Weissbrod, and Dr. Jianke Zhu. They are each co-authors of one of my papers.

Lastly, I thank all the members of ARCLab with whom I had the pleasure of working with, including (alphabetically): Andrew Bilan, Soumyaraj Bose, Zih-Yun (Sarah) Chiu, Dr. Nikhil Das, Dr. Emily K. Funk, Peter V. Gavrilov, Aaron Gun, Jingbin Huang, Ambareesh Jayakumari, Jacob Johnson, Bjorn Johnson, Calvin Joyce, Hoi Man (Kevin) Lam, Dr. Yang Li, Albert Liao, Jason Lim, Dr. Shan Lin, Dr. Fei Liu, Jingpei Lu, Dr. Tokio Matsuzaki, Parth Natekar, Elizabeth Peiros, Casey H. Price, Dr. Ahmed Qureshi, Andrew Saad, Dimitri Schreiber, Shihao (Kelvin) Shen, Nikhil U. Shinde, Harleen Singh, Mrinal Verghese, Mingwei Yeoh, Fei Yin, Zhaowei Yu, Yifei Zhang, and Yuheng Zhi.

Chapter 1, in part, is a reprint of material from F. Richter, S. Shen, F. Liu, J. Huang, E. K. Funk, R. K. Orosco, M. C. Yip "Autonomous Robotic Suction to Clear the Surgical Field for Hemostasis using Image-based Blood Flow Detection," in *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1383-1390, 2021. The dissertation author is the primary author of this papers.

Chapter 2, in part, is a reprint of material from F. Richter, J. Lu, R. K. Orosco, M.

C. Yip "Robotic tool tracking under partially visible kinematic chain: A unified approach," in *IEEE Transactions on Robotics*, vol. 38, no. 3, pp. 1653-1670, 2022. The dissertation author is the primary author of this paper.

Chapter 3, in part, is a reprint of material from Y. Li*, F. Richter*, J. Lu, E. K. Funk, R. K. Orosco, J. Zhu, M. C. Yip "Super: A Surgical Perception Framework for Endoscopic Tissue Manipulation with Surgical Robotics," in *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2294-2301, 2020. (* indicates equal contribution) The dissertation author is one of the primary author of this paper. Chapter 3, in part, is also a reprint of material from J. Lu, A. Jayakumari, F. Richter, Y. Li, M. C. Yip "Super Deep: A Surgical Perception Framework for Robotic Tissue Manipulation using Deep Learning for Feature Extraction," in *IEEE International Conference on Robotics and Automation*, pp. 4783-4789, 2021.

Chapter 4, in part, is a reprint of material from F. Richter, S. Shen, F. Liu, J. Huang, E. K. Funk, R. K. Orosco, M. C. Yip "Autonomous Robotic Suction to Clear the Surgical Field for Hemostasis using Image-based Blood Flow Detection," in *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1383-1390, 2021 and to appear in publication as F. Richter, R. K. Orosco, M. C. Yip "Image Based Reconstruction of Liquids from 2D Surface Detections," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. The dissertation author is the primary author of both papers.

Chapter 5, in part, is a reprint of material from F. Richter, S. Shen, F. Liu, J. Huang, E. K. Funk, R. K. Orosco, M. C. Yip "Autonomous Robotic Suction to Clear the Surgical Field for Hemostasis using Image-based Blood Flow Detection," in *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1383-1390, 2021. The dissertation author is the primary author of this paper. Chapter 5, in part, is also a reprint of material from J. Huang, F. Liu, F. Richter, M. C. Yip "Model-predictive control of blood suction for surgical hemostasis using differentiable fluid simulations," in *IEEE International Conference on Robotics and Automation*, pp. 12380-12386, 2021.

VITA

2017	Bachelor of Science in Electrical Engineering, University of Illinois Chicago
2020	Masters of Science in Electrical Engineering, University of California San Diego
2022	Doctor of Philosophy in Electrical Engineering, University of California San Diego

PUBLICATIONS

F. Richter, R. K. Orosco, and M. C. Yip, "Image Based Reconstruction of Liquids from 2D Surface Detections," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2022.

F. Richter, J. Lu, R. K. Orosco, and M. C. Yip, "Robotic Tool Tracking under Partially Visible Kinematic Chain: A Unified Approach," in *IEEE Transactions on Robotics*, vol. 38, no. 3, pp. 1653-1670, 2022.

F. Richter*, P. V. Gavrilov*, H. M. Lam*, A. Degani and M. C. Yip, "ARCSnake: Reconfigurable Snake-Like Robot with Archimedean Screw Propulsion for Multi-Domain Mobility," in *IEEE Transactions on Robotics*, vol. 38, no. 2, pp. 797-809, 2022. (* indicates equal contribution)

J. Lu, **F. Richter**, and M. C. Yip, "Pose Estimation for Robot Manipulators via Keypoint Optimization and Sim-to-Real Transfer," in *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4622–4629, 2022.

P. Barba, J. Stramiello, E. K. Funk, **F. Richter**, M. C. Yip, and R. K. Orosco, "Remote Telesurgery in Humans: a Systematic Review," in *Surgical Endoscopy*, vol. 36, pp. 2771–2777, 2022.

F. Richter, S. Shen, F. Liu, J. Huang, E. K. Funk, R. K. Orosco, and M. C. Yip, "Autonomous Robotic Suction to Clear the Surgical Field for Hemostasis using Image-based Blood Flow Detection," in *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1383-1390, 2021.

F. Richter, E. K. Funk, W. S. Park, R. K. Orosco, and M. C. Yip, "From Bench to Bedside: The First Live Robotic Surgery on the dVRK to Enable Remote Telesurgery with Motion Scaling," in *IEEE International Symposium on Medical Robotics*, pp. 1-7, 2021.

J. Huang, F. Liu, **F. Richter**, and M. C. Yip, ”Model-Predictive Control of Blood Suction for Surgical Hemostasis using Differentiable Fluid Simulations,” in *IEEE International Conference on Robotics and Automation*, pp. 12380-12386, 2021.

Z. Y. Chiu, **F. Richter**, E. K. Funk, R. K. Orosco, and M. C. Yip, “Bimanual Regrasping for Suture Needles using Reinforcement Learning for Rapid Motion Planning,” in *IEEE International Conference on Robotics and Automation*, pp. 7737-7743, 2021.

J. Lu, A. Jayakumari, **F. Richter**, Y. Li, and M. C. Yip, “SuPer Deep: A Surgical Perception Framework for Robotic Tissue Manipulation using Deep Learning for Feature Extraction,” *IEEE International Conference on Robotics and Automation*, pp. 4783-4789, 2021.

R. K. Orosco, B. Lurie, T. Matsusaki, E. K. Funk, V. Divi, F. C. Holsinger, C. Holsinger, S. Hong, **F. Richter**, N. Das, and M. C. Yip, “Compensatory motion scaling for time-delayed robotic surgery,” in *Surgical Endoscopy*, vol. 35, no. 6, 2020.

Y. Li*, **F. Richter***, J. Lu, E. K. Funk, R. K. Orosco, J. Zhu, and M. C. Yip, “SuPer: A Surgical Perception Framework for Endoscopic Tissue Manipulation with Surgical Robotics,” in *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2294-2301, 2020. (* indicates equal contribution)

D. A. Schreiber*, **F. Richter***, A. Bilan, P. V. Gavrilov, H. L. Man, C. H. Price, K. C. Carpenter, and M. C. Yip, “ARCSnake: An Archimedes’ Screw-Propelled, Reconfigurable Robot Snake for Complex Environments,” in *IEEE International Conference on Robotics and Automation*, pp. 7029-7034, 2020.(* indicates equal contribution)

M. Verghese, **F. Richter**, A. Gunn, P. Weissbrod, and M. C. Yip, “Model-free Visual Control for Continuum Robot Manipulators via Orientation Adaptation,” in *International Symposium of Robotics Research*, 2019.

F. Richter, R. K. Orosco, and M. C. Yip, “Motion scaling solutions for improved performance in high delay surgical teleoperation,” in *IEEE International Conference on Robotics and Automation*, pp. 1590-1595, 2019.

F. Richter, Y. Zhang, Y. Zhi, R. K. Orosco, and M. C. Yip, “Augmented reality predictive displays to help mitigate the effects of delayed telesurgery,” in *IEEE International Conference on Robotics and Automation*, pp. 444-450, 2019.

ABSTRACT OF THE DISSERTATION

Autonomous Maintenance of Hemostasis in Robotic Surgery

by

Florian Richter

Doctor of Philosophy in Electrical Engineering (Intelligent Systems, Robotics and Control)

University of California San Diego, 2022

Professor Michael C. Yip, Chair
Professor Nikolay Atanasov, Co-Chair

Surgical robots are becoming increasingly common in operating rooms, which provides the opportunity to deploy automation algorithms for surgery. Surgical task automation aims to improve patient throughput, reduce quality-of-care variance among surgeries, and potentially deliver complete automated surgery in the future. While progress in developing autonomous surgical tasks has leaped forward, reactive maneuvers to traumatic events, such as hemostasis, represent a critical area that has attracted little attention. Hemostasis describes a state of the surgical field that is achieved when there is no site of active bleeding and the tissues are unobstructed by blood. Unlike previously automated tasks that occur in a more predictable cadence within a procedure, bleeding can be unpredictable, which necessitates hemostatic maneuvers at any time during surgery.

In this dissertation, all the necessary perception, motion planning, and control strategies are presented to autonomously control a robotic suction tool to clear the surgical field from blood. First, a surgical tool tracking technique is proposed that localizes the robotic agent, which will clear the surgical field, in the endoscopic camera frame. The surgical tool tracking is combined with a deformable tissue tracker to completely track a surgical scene before a vessel rupture occurs. The combination of the two trackers is coined SuPer, the Surgical Perception framework. Next, the blood from a vessel rupture scenario is perceived by detecting and reconstructing the flowing blood from the endoscopic camera data. Finally, a controller and a motion planner for the robotic suction tool to clear the surgical field of blood are presented.

Chapter 1

Introduction

Surgery is an invasive medical treatment that requires physical intervention. In open surgery, large incisions are made to access the surgical site where the treatment would be applied. Through modern sensing technology (e.g., endoscopic cameras), smaller incisions, and sometimes even no incisions, can be achieved. Surgeons visualize the surgical site through small endoscopic cameras while guiding and controlling low-profile surgical instruments (e.g., laparoscopic instruments). Making fewer and smaller incisions is associated with less scarring and improved recovery time for patients, ultimately allowing them to get back to normal life sooner. The use of less invasive surgical procedures is defined as minimally invasive surgery.

Minimally invasive surgery is a motivating factor for developing novel surgical robotic systems that aim to improve surgical precision, increase dexterity, and decrease the instrument profiles in the confined workspace of a small surgical site. The da Vinci Surgical System [23] from Intuitive is a representative example of such a system and has been through multiple iterations (S, Si, Xi). Competitors of similar designs have been developed, such as the Versius from CMR Surgical and the Ottava from Johnson and Johnson. Single port robotic systems (e.g., da Vinci SP from Intuitive and Sport Surgical from Titan Medical) are emerging with the promise of only requiring a single entry point (i.e., one incision).

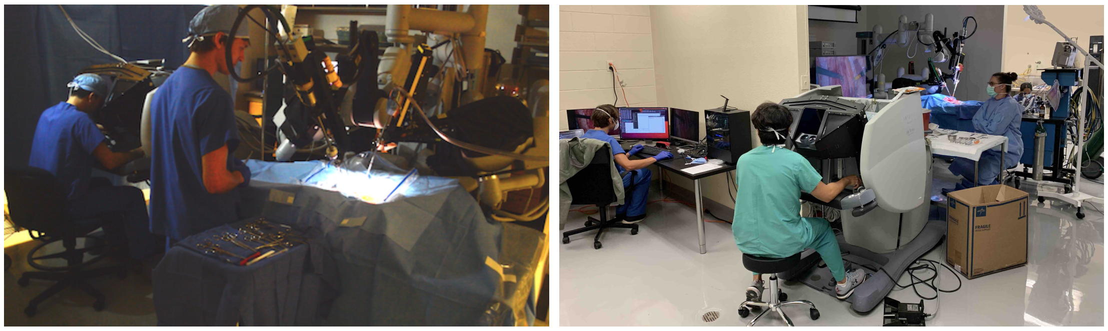


Figure 1.1. The dVRK [65] being used with a phantom abdomen on the left [135] and a live surgery on the right [133].

More specialized surgical robotic systems have also been developed for other surgical procedures. The Monarch Platform from Auris Health is a robotic catheter designed to access hard-to-reach lung nodules. Mako from Stryker and ROSA from Zimmer Biomet were developed for orthopedic surgeries (e.g., knee arthroplasty). CorPath GRX from Cornidus supports percutaneous coronary and vascular procedures.

To date, surgical robots require direct control from an operating surgeon. Surgeons send commands to the surgical robot from a console with a joystick or a back-drivable manipulator, and the robot will execute the exact commands that it received. While this utilization of surgical robotics is helping patients today, it is not the full potential of what surgical robots can do in the operating room. To this end, a variety of research for surgical robotics has emerged, such as performance metrics [56, 55, 75], remote surgeries [137, 134, 115, 133], and surgical task automation [171, 31]. Surgical task automation is an increasingly expanding research field in an effort to improve patient throughput, reduce quality-of-care variance among surgeries, and potentially deliver complete automated surgery in the future.

1.1 Surgical Task Automation

Success in realizing surgical automation has accelerated in recent years, with improvements in available open-source platforms such as the da Vinci Research Toolkit (dVRK) [65], RAVEN [92, 49], and simulators [135, 167, 107]. The dVRK, as shown in Fig. 1.1, is used by over 40 robotic research groups and represents an open-sourced version of the da Vinci® Surgical System [22]. It is the ideal candidate for translating research to practice on over 5,000 da Vinci® Surgical Systems used in hospitals around the world. Furthermore, the dVRK has been tested with animal studies for more realistic experimentation of the developed automation technology [133].

Tissue manipulation is a challenging research area for surgical automation due to the inherent complexity in shape from deformations. General techniques for perceiving tissue deformations have been proposed using a model-free, fusion approach [81, 88, 87]. From the perceived tissue, real-to-sim registration was performed such that control of the tissue could be derived from the simulation [84]. Reinforcement learning has also been applied to control the tissue deformations caused by a surgical robotic manipulator [149, 125]. Coarse approaches along with online Jacobian estimation techniques have been proposed for modeling the deformations [1].

Specific surgical tasks have also been automated, such as resection [108, 158] and debridement removal [67]. The task of suturing has engrossed the community [146, 178], and there is a good reason for this, as suturing for robot-assisted minimally invasive surgery has been cited as being significantly more challenging and time-consuming than manual suturing [39, 54]. Suturing requires a wide range of considerations for effective automation, including perception [15], needle manipulation [156, 16], knot tying [101], identification of entry and exit points for the suture throw [86], and interfacing the automation with the surgeon for effective deployment [165].

1.2 Automation of Maintaining Hemostasis

While progress in developing autonomous surgical tasks has leaped forward, reactive measures to traumatic events, such as hemostasis, represent a critical area that has

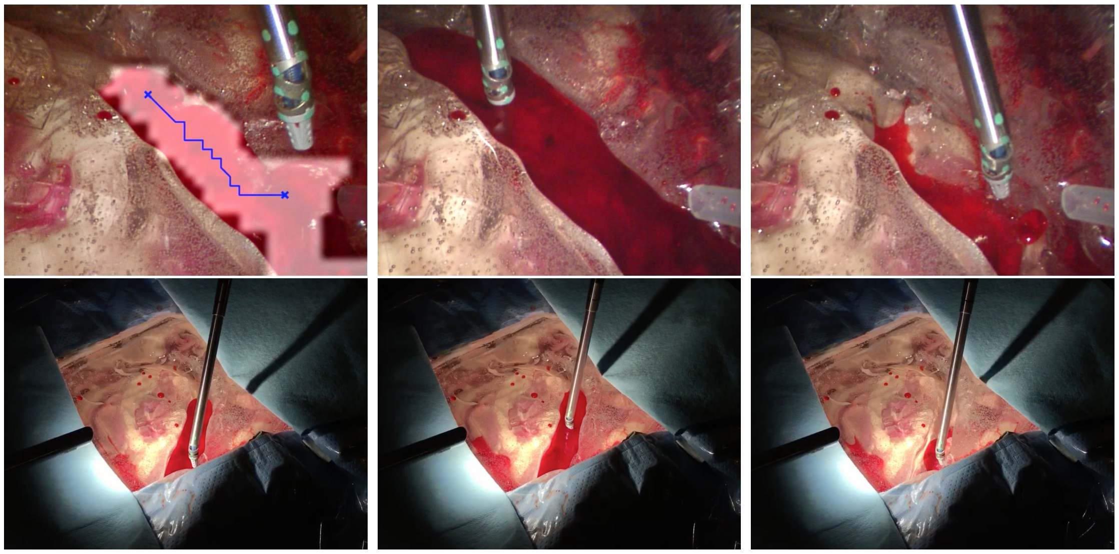


Figure 1.2. Autonomous blood suction for cases of trauma in surgery. A robotic suction tool clears the surgical field of flowing blood.

attracted little attention. Hemostasis describes a state of the surgical field that is achieved when there is no site of active bleeding and the tissues are unobstructed by blood. The bleeding can originate from either a visible or macroscopic blood vessel (artery or vein) or from the microvasculature and capillary network within tissues. Unlike previously automated tasks that occur in a more predictable cadence within a procedure, bleeding can be unpredictable, which necessitates hemostatic maneuvers at any time during any surgery. Typically, surgical manipulation of any type–suction, grasping, retraction, cutting, and dissection–can immediately cause bleeding. However, the onset of bleeding can also be delayed. For example, if a vessel is not definitively sealed, it can rupture spontaneously without direct contact. If surgical automation is ever to be deployed, it will require the implementation of reactive strategies to manage these traumatic scenarios.

This dissertation specifically addresses the problem of small and medium vessel ruptures. Overall, there are four distinct stages in hemostasis of this scenario: (1) clearing the surgical field of blood; (2) identification of the bleeding source (vessel rupture); (3) grasping the identified vessel to temporarily stop bleeding; (4) closing the vessel rupture, usually with an energy-based device, clip, or suture. Each stage requires complex manipulation skills and perception algorithms, which represent non-trivial hurdles to automation.

In this dissertation, all the necessary perception and additional automation components for clearing the surgical field are presented as illustrated in Fig. 1.2. First, a surgical tool tracking technique is proposed that can localize the robotic agent, which is deployed to clear the surgical field, in the endoscopic camera frame. The surgical tool tracking function is combined with a deformable tissue tracker to completely track a surgical scene before a vessel rupture occurs. The combination of the two trackers is coined SuPer, the Surgical Perception framework. Next, the perception of blood from a vessel rupture scenario is achieved by detecting and reconstructing the flowing blood from the endoscopic camera data. Finally, a controller and motion planner for the robotic suction tool to clear the surgical field of blood are presented.

1.3 Acknowledgements

Chapter 1, in part, is a reprint of material from F. Richter, S. Shen, F. Liu, J. Huang, E. K. Funk, R. K. Orosco, M. C. Yip "Autonomous Robotic Suction to Clear the Surgical Field for Hemostasis using Image-based Blood Flow Detection," in *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1383-1390, 2021. The dissertation author is the primary author of this paper.

Chapter 2

Surgical Robotic Tool Tracking

Controlling a robot manipulator via visual feedback depends on an important coordinate transform: the orientation and translation between the robot and the camera frame. This coordinate frame transforms the positions and velocities of the robot based on its kinematics, such as its end-effector position, into the camera's reference frame, where the feedback and trajectories are often defined. Typically, this relative transform is calibrated by placing markers, such as ArUco [40], on the robot, identifying them in the image frame, and solving the homogeneous linear system for the *base-to-camera* transform [28]. However, when the camera can only observe a portion of the robot manipulator, the calibration of the base-to-camera transform is more susceptible to errors due to the limited range of motions the robot can process during data collection [161]. This situation arises when the camera is positioned to perceive the robotic tool (e.g., gripper) along with the environment or the objects targeted for manipulation rather than the entire kinematic chain. An example scenario is shown in Fig. 2.1. This type of scenario comes up frequently in object grasping and manipulation tasks [79, 95] and small-scale manipulations, such as robotic surgery with the da Vinci® Surgical System.

A secondary source of uncertainty affecting robotic control is the inaccuracy of the joint angle measurements. The errors in joint angle measurements are caused by biases in positioning, drifting in readings, and complex transmission effects, such as cable stretch

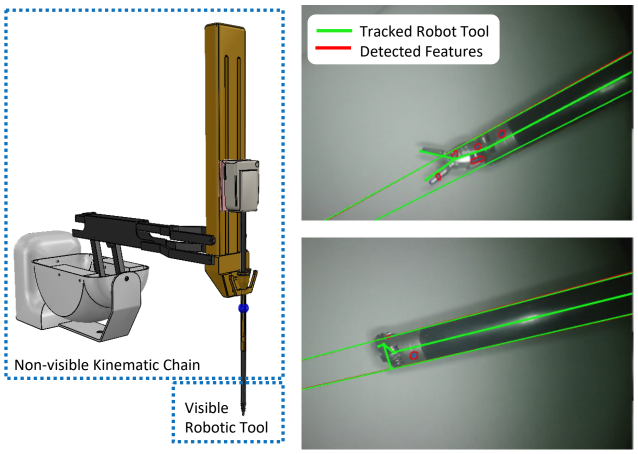


Figure 2.1. Precise robotic manipulation, such as surgery, utilizes visual information with the sensor positioned to observe the environment and objects of interest rather than the entire kinematic chain. In this work, we track the robotic tool even with a partially visible kinematic chain.

and backlash. Similar to finding the base-to-camera transform, this issue is typically solved through calibration with a separate sensor, such as a camera, that collects ground truth measurements and compares them against the joint angle readings [126]. The issue of non-constant errors, such as the cable stretch, has been addressed by explicit dynamic modeling [102] and data-driven approaches with neural networks [57]. These methods, however, are challenging to apply outside of a laboratory setting due to the need for additional sensors or calibration steps. Furthermore, the calibration parameters can degrade over time through irreversible effects from transmission wear-and-tear and mechanical creep.

Thus, surgical robotic-endoscopic platforms [47, 23, 160] will have both challenging base-to-camera calibration and errors in joint angle measurements. The endoscopes are designed to only capture a small working space for higher operational precision. Surgical robotic platforms also typically use cable drives to enable low-profile robotic tools, which are the source of the joint angle measurement error. Furthermore, the bases of the surgical manipulators are adjusted regularly depending on the type of procedure and to fit each patient's anatomy. There is a significant amount of previous literature from the surgical robotics community tackling these two problems separately. Here we unify these two problems and present a solution that also generalizes to other robotic manipulators.

In this work, we demonstrate the ability to track robotic tools from visual observations that only show part of the kinematic chain under the conditions of uncertainty in base-to-camera transform and joint angle measurements. To this end, we present the following novel contributions:

1. a novel problem formulation that proves that the direct estimation of all the described parameters is infeasible since they are non-identifiable,
2. the first approach that unifies these uncertainties into a smaller parameter set that is identifiable and compensates for all the uncertainties,
3. an extension of tracking under simultaneously moving robotic tools and cameras.

We coin the reduced parameter set as *Lumped Error*. A tracking algorithm based on a particle filter was created to track the Lumped Error. The particle filter uses visual features from the tracked robotic tool to continuously update the belief of the Lumped Error. In our implementation, the visual features were detected using markers, edge detectors for geometric primitives such as cylinders, and learned point features. For experimentation, the presented particle filter was evaluated both in simulation and on real-world robotic data using the da Vinci Research Kit (dVRK) [65], a widely used surgical robotics research platform with a total of 10 Degrees of Freedom (DoF) and a gripper across both the endoscopic (i.e., surgical robotic camera arm) and robotic manipulator kinematic chains. The method was also applied to a non-surgical robot, Rethink Robotic's Baxter robot, in Appendix A. In these experiments, the joint angle disturbances included simulated noise, cable stretch from the dVRK, and backlash from the Baxter arm. Overall, these results show that the estimation of the Lumped Error is efficient and yields precise and accurate robotic tool tracking.

2.1 Related Work

Integration of visual observations with robotic manipulators and handling inaccurate joint angle measurements is not a new concept. Therefore, this related work section is split into different categories to cover the wide range of robotic tracking solutions presented by various research groups. A special focus is given to surgical robotics as the challenge of surgical robotic tool tracking represents a direct application of the presented work.

Base-to-camera estimation

A common approach to calibrating the base-to-camera transform is rigidly attaching a marker whose pose can be directly estimated from visual data (e.g., ArUco [40], ARTag [30], AprilTag [114], and STag [5]), collecting multiple images of the marker, and solving the homogeneous linear system [28, 120]. To relieve the heavy reliance on 3D pose reconstruction, which can often be inaccurate from 2D images, markers were attached to robot manipulators to collect 2D keypoints, and the camera-to-base transform was estimated with Solve-PnP [78]. Deep learning approaches have been applied to detect 2D keypoints on robotic manipulators to remove the need for markers [73, 72, 77, 89]. However, these calibration methods do not consider errors in joint angle measurements and instead make the assumption that the robot kinematic chain is located exactly at the joint angle readings.

Zhong et al. proposed an interactive method to maximize the accuracy for calibrating remote center of motion (RCM) robots [179], which are typically used as laparoscopic robots. Similarly, Zhao et al. defined a kinematic remote-center coordinate system (KCS) that absorbs all the errors in the transform from the camera frame to the base of an RCM robot [177]. Tracking the KCS is a common technique in surgical robotics, where the updates come from learned features [131, 132], silhouette matching [50], or online template matching [169]. However, these estimation methods do not explicitly consider the effects of joint angle errors. In fact, we show that the Lumped Error is mathematically equivalent to tracking the KCS, implying that these methods can compensate for both the base-to-camera transform and the joint angle errors. Furthermore, in this work, we generalize the Lumped Error to other robotic manipulators.

Joint measurement error estimation

Using fiducial markers to collect data, Pastor et al. applied a data-driven approach to estimating the joint angle error [121]. Meanwhile, Wang et al. used markers to estimate the joint angle offsets in real-time via inverse kinematics [163]. From the perspective of surgical robotics, the errors of joint angle readings due to transmission effects have largely been studied in the context of cable drives. Miyasaka et al. explicitly modeled the physical effects of cable transmissions, such as friction and hysteresis [102]. Learning-based approaches have been applied based on neural networks for direct estimation of cable stretch [57, 123] and Gaussian processes for compensation [94]. Using visual data, the Unscented Kalman Filter [46] and neural networks [145] were applied to estimate the effects of cable stretch. These techniques, however, are impractical to apply outside of laboratory settings due to the need for additional sensors or calibration steps. In addition, calibration parameters can degrade over time due to mechanical effects, such as cable creep, which occurs when cable stretch varies irreversibly through usage.

Combined base-to-camera and joint estimation

Joint calibration techniques have been proposed which optimize for both joint angle offsets and base-to-camera transformations [126, 76]. To handle dynamic uncertainties, such as the non-constant joint angle errors, a real-time estimation strategy based on combining the iterative closest point from depth sensing and Kalman Filtering has been proposed [70]. A probabilistic approach has also been proposed based on observation models that are grounded in physical parameters, which facilitates the process of tuning the hyperparameters [18]. These works largely focus on the integration of sensors into real-time estimation. Instead of this approach, we assessed the parameter reductions in the case of partially visible kinematic chains. In this context, the proposed Lumped Error parameter reduction can aid these efforts by reducing the total number of parameters that need to be estimated in the case of a partially visible kinematic chain. Nonetheless, we propose a particle filter to estimate the Lumped Error, which relies only on image data. In contrast, the efforts above rely on depth sensing, which is not as readily available on all robotic platforms, such as the da Vinci® Surgical System.

Another widely used approach to controlling a robot from visual feedback without the base-to-camera transform is through online Jacobian estimation [148]. These visual servoing techniques can even compensate for kinematic inaccuracies and joint angle measurement errors [14]. While these techniques are sufficient to control the end-effector in the camera frame, they do not describe the remainder of the kinematic chain in the camera frame.

Eye-in-hand configuration

Another consideration in the case of a robotic camera arm is the problem of the *eye-in-hand* calibration [150]. This particular visual-robotic challenge is not considered here but included to ensure completeness. Zhang et al. developed a computationally efficient method using dual quaternions [176]. Adjoint transformations from twist motions have also been applied to converge to solutions with high accuracy [116, 117]. In the case of RCM robots, a reduction of the computational complexity has been achieved [118, 164]. Similar to the previously described visual servoing techniques, the robot camera arm can also be controlled through online Jacobian estimation [124, 162].

2.2 Problem Formulation

The 3D geometry of a robotic tool can be fully described in the stationary camera frame through a base-to-camera transform and forward kinematics. A single point, $\mathbf{o}^j \in \mathbb{R}^3$, on the j-th link of a robotic tool, can be transformed to the stationary camera frame by

$$\overline{\mathbf{o}}^c_t = \mathbf{T}^c_b \prod_{i=1}^{j} \mathbf{T}^{i-1}_i(q^i_t)\overline{\mathbf{o}}^j \tag{2.1}$$

at time t where $\mathbf{T}^c_b \in SE(3)$ is the base-to-camera transform and $\mathbf{T}^{i-1}_i(q^i_t) \in SE(3)$ is the i-th joint transform with the joint angle q^i_t. The overline operator $(\bar{\cdot})$ defines the homogeneous representation of a 3D point (e.g $\overline{\mathbf{o}} = [\mathbf{o}\ 1]^\top$). Therefore, the base-to-camera transform and the joint angles are the only parameters necessary to describe a robotic manipulator in the camera frame. Typically, calibration can be performed for the base-to-camera transform, and the joint angles can be derived from encoder readings. The issue with applying this approach directly to scenarios where the camera only captures images with a portion of the kinematic chain is that small errors in calibration or joint angles will be exacerbated in the image frame.

Therefore, let $\tilde{q}^1_t, \dots, \tilde{q}^{n_j}_t$ be the joint angle measurements with the respective measurement errors $e^1_t, \dots, e^{n_j}_t$, such that

$$q^i_t = \tilde{q}^i_t + e^i_t \tag{2.2}$$

for all $i = 1, \dots, n_j$. No distribution is assumed for the errors, e^i_t. For example, the error could be a constant bias from the absolute position error or non-constant with hysteresis effects from the cable stretch. In combination with (2.1), the robotic tool can be

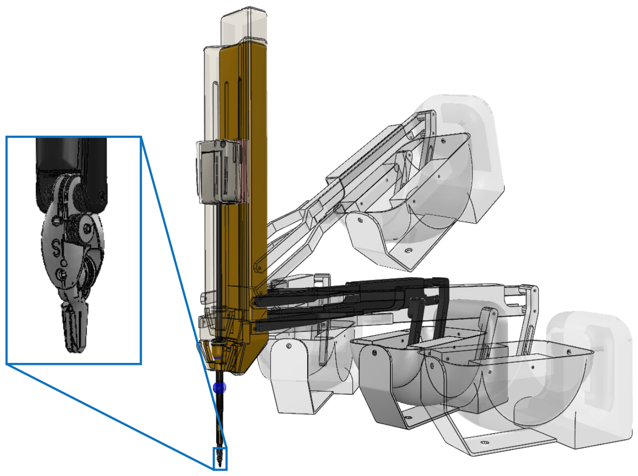


Figure 2.2. Based on an image of the robotic tool (marked in blue) and not the whole kinematic chain, multiple solutions exist for joint angles and base-to-camera transform errors. Examples are presented as transparent kinematic chains.

described in the camera frame by

$$\overline{\mathbf{o}}^c_t = \mathbf{T}^c_{b-}\mathbf{T}^{b-}_b \prod_{i=1}^{j} \mathbf{T}^{i-1}_i(\tilde{q}^i_t + e^i_t)\overline{\mathbf{o}}^j \tag{2.3}$$

where the true base-to-camera transform is broken into $\mathbf{T}^c_{b-} \in SE(3)$ and $\mathbf{T}^{b-}_b \in SE(3)$, which are measured from an initial calibration and the error in the calibration, respectively. Therefore, in order to correctly describe the robotic tool in the camera frame, both the joint angle errors, e^i_t, and the error in the base-to-camera transform, $\mathbf{T}^{b-}_b$, should be estimated. Let n_j be the total number of joint angles and the $SE(3)$ error transform, $\mathbf{T}^{b-}_b$, an estimate based on an axis-angle and a translation vector, yielding a total of $n_j + 6$ parameters to be estimated.

Explicit estimation for the joint angles and the base-to-camera transform is not possible when only a portion of the kinematic chain is visible in the camera frame. This limitation exists because it is not possible to identify the source of the error, either the joint angles or the base-to-camera transform calibration. For example, a surgical tool is considered partially visible based on the endoscopic camera data. The endoscope's narrow field only has visual information of the tool-tip and not the base or the joints preceding the articulated wrist resulting in multiple viable solutions to estimating $\mathbf{T}_b^{b-}$ and e_t^i. One example is shown in Fig. 2.2, where the joint parts of the RCM are not visible to the endoscopic camera, whereas the joints on the gripper are.

This problem relates to the concept of *identifiability* [139]. Identifiability is concerned with the existence of a unique inverse association with regard to the parameters estimated from observations. Fig. 2.2 presents examples that do not show a unique association from the image of the surgical tool from an endoscope to the base-to-camera transform and the joint angle errors. These instances are denoted as *observationally equivalent*. The parameters are only considered identifiable if there are no observational equivalences.

Claim 1. *When only using the camera data for observations, then the error in base-to-camera transform,* $\mathbf{T}_b^{b-}$*, and errors in the first* n_b *joint angles,* $e_t^1, \dots, e_t^{n_b}$*, are not identifiable if all the kinematic links preceding joint* n_b *are out of the camera frame.*

Proof. Let the *Modified Denavit-Hartenberg Parameters* be used to define each forward kinematic joint transform. Therefore: $\mathbf{T}_i^{i-1}(q_t^i) = \mathbf{T}_x(\alpha^i, a^i)\mathbf{T}_z(\theta^i, d^i)$ where

$$\mathbf{T}_x(\alpha^i, a^i) = \begin{bmatrix} 1 & 0 & 0 & a^i \\ 0 & cos(\alpha^i) & -sin(\alpha^i) & 0 \\ 0 & sin(\alpha^i) & cos(\alpha^i) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{T}_z(\theta^i, d^i) = \begin{bmatrix} cos(\theta^i) & -sin(\theta^i) & 0 & 0 \\ sin(\theta^i) & cos(\theta^i) & 0 & 0 \\ 0 & 0 & 1 & d^i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

and q_t^i is plugged into θ^i or d^i for a revolute and prismatic joint, respectively. A revolute joint transform with a joint angle of $\omega + \psi \in \mathbb{R}$, $\mathbf{T}_i^{i-1}(\omega + \psi)$ can be expanded using the

Modified Denavit-Hartenberg Parameters,

$$\mathbf{T}_x(\alpha^i, a^i)\mathbf{T}_z(\omega + \psi, d^i) \tag{2.5}$$

$$\mathbf{T}_x(\alpha^i, a^i)\mathbf{T}_z(\omega, 0)\mathbf{T}_x^{-1}(\alpha^i, a^i)\mathbf{T}_x(\alpha^i, a^i)\mathbf{T}_z(\psi, d^i) \tag{2.6}$$

$$\mathbf{T}_i(\omega)\mathbf{T}_i^{i-1}(\psi) \tag{2.7}$$

where $\mathbf{T}_i(\omega) = \mathbf{T}_x(\alpha^i, a^i)\mathbf{T}_z(\omega, 0)\mathbf{T}_x^{-1}(\alpha^i, a^i)$. Likewise for a prismatic joint transform,

$$\mathbf{T}_x(\alpha^i, a^i)\mathbf{T}_z(\theta^i, \omega + \psi) \tag{2.8}$$

$$\mathbf{T}_x(\alpha^i, a^i)\mathbf{T}_z(0, \omega)\mathbf{T}_x^{-1}(\alpha^i, a^i)\mathbf{T}_x(\alpha^i, a^i)\mathbf{T}_z(\theta^i, \psi) \tag{2.9}$$

$$\mathbf{T}_i(\omega)\mathbf{T}_i^{i-1}(\psi) \tag{2.10}$$

where $\mathbf{T}_i(\omega) = \mathbf{T}_x(\alpha^i, a^i)\mathbf{T}_z(0, \omega)\mathbf{T}_x^{-1}(\alpha^i, a^i)$. Note that the same notation, $\mathbf{T}_i(\omega)$, is used for rotational and prismatic joints as simplified notation in the coming equations.

Based on these expansions, we will use induction to show that portions of the joint angle errors can be expanded out as follows

$$\prod_{i=1}^{n_b} \mathbf{T}_i^{i-1}(\tilde{q}_t^i + e_t^i) = \mathbf{T}^{n_b} \prod_{i=1}^{n_b} \mathbf{T}_i^{i-1}(\tilde{q}_t^i + \beta_i e_t^i) \tag{2.11}$$

where

$$\mathbf{T}^{n_b} = \prod_{k=1}^{n_b} \Big(\prod_{i=1}^{k-1} \mathbf{T}_i^{i-1}(\tilde{q}_t^i + \beta_i e_t^i)\Big)\mathbf{T}_k((1-\beta_i)e_t^k)\Big(\prod_{i=1}^{k-1} \mathbf{T}_i^{i-1}(\tilde{q}_t^i + \beta_i e_t^i)\Big)^{-1} \tag{2.12}$$

and $\beta_i \in \mathbb{R}$ for $i = 1, \ldots, n_b$ is an arbitrary portion of the joint angle error not to be lumped into $\mathbf{T}^{n_b}$. For the base case of $n_b = 1$ in (2.11), the error of the joint angle can be pulled out

$$\mathbf{T}_1^0(\tilde{q}_t^i + e_t^i) = \mathbf{T}_1((1-\beta_1)e_t^i)\mathbf{T}_1^0(\tilde{q}_t^i + \beta_1 e_t^i) \tag{2.13}$$

using (2.5) - (2.10).

If (2.11) holds true for $n_b = m$, the left-hand-expression from (2.11) can be rewritten for $n_b = m + 1$ as follows

$$\mathbf{T}^m \prod_{i=1}^{m} \mathbf{T}_i^{i-1}(\tilde{q}_t^i + \beta_i e_t^i) \mathbf{T}_{m+1}^m(\tilde{q}_t^{m+1} + e_t^{m+1}) \tag{2.14}$$

which expands to

$$\mathbf{T}^m \prod_{i=1}^{m} \mathbf{T}_i^{i-1}(\tilde{q}_t^i + \beta_i e_t^i) \mathbf{T}_{m+1}((1-\beta_{m+1})e_t^{m+1}) \mathbf{T}_{m+1}^m(\tilde{q}_t^{m+1} + \beta_{m+1} e_t^{m+1}) \tag{2.15}$$

using (2.5) - (2.10). Then, the expression is expanded one more time

$$\mathbf{T}^m \prod_{i=1}^{m} \mathbf{T}_i^{i-1}(\tilde{q}_t^i + \beta_i e_t^i) \mathbf{T}_{m+1}((1-\beta_{m+1})e_t^{m+1}) \Big(\prod_{i=1}^{m} \mathbf{T}_i^{i-1}(\tilde{q}_t^i + \beta_i e_t^i) \Big)^{-1} \prod_{i=1}^{m} \mathbf{T}_i^{i-1}(\tilde{q}_t^i + \beta_i e_t^i) \\ \mathbf{T}_{m+1}^m(\tilde{q}_t^{m+1} + \beta_{m+1} e_t^{m+1}) \tag{2.16}$$

which is equivalent to (2.11). Therefore (2.11) holds for $n_b = 1, 2, \ldots$ by mathematical induction.

Let $P(\mathbf{y}_t | \mathbf{T}_b^{b-}, e_t^1, \ldots, e_t^{n_j})$ be a proper probability distribution and the observation model of some feature $\mathbf{y}$ from the surgical tool in the camera frame parameterized by all the unknowns in the kinematic chain described in (2.3). Since $P(\mathbf{y}_t|\cdot)$ cannot describe a feature from the kinematic links that precede joint n_b, the observation model for some feature $\mathbf{y}$ can be re-parameterized to

$$P\Big(\mathbf{y}_t | \mathbf{T}_{b-}^c \mathbf{T}_b^{b-} \prod_{i=1}^{n_b} \mathbf{T}_i^{i-1}(\tilde{q}_t^i + e_t^i), e_t^{n_b+1}, \ldots, e_t^{n_j}\Big) \tag{2.17}$$

The equality in (2.11) implies that the observation is not a one-to-one mapping from the parameter space $(\mathbf{T}_b^{b-}, e_t^1, \ldots, e_t^{n_j})$ to the camera observation (output of $P(\mathbf{y}_t|\cdot)$). In fact, for each observation generated by $P(\mathbf{y}_t|\cdot)$, there are an infinite number of solutions for the inverse mapping that are spanned by $\beta_1, \dots, \beta_{n_b}$. Since there are infinite observational equivalencies, the parameters are not identifiable.

$\square$

The equality in (2.11), which causes the lack of identifiability, can be interpreted as moving the joint errors from the kinematic chain to the base transform of the robot. A lack of identifiability implies undesirable properties for parameter estimation, such as rank deficiency in the Fischer Information Matrix [139]. Furthermore, it shows the inability to estimate both errors in joint angles and the base-to-camera transform.

2.3 Lumped Error Derivation and Tracking

Based on Claim 1, it is infeasible to estimate all error parameters described in (2.3) using camera data alone. Therefore, we propose a parameter reduction technique where all errors of the first n_b joints are lumped together with the error in the base-to-camera transform. Hence, we call it the *Lumped Error* Transform.

Using (2.11), (2.3) can be re-written as

$$\overline{\mathbf{o}}_t^c = \mathbf{T}_{b-}^c \mathbf{T}_{n_b}^{b-}(\mathbf{w}_t, \mathbf{b}_t) \prod_{i=1}^{n_b} \mathbf{T}_i^{i-1}(\tilde{q}_t^i) \prod_{i=n_b+1}^{j} \mathbf{T}_i^{i-1}(\tilde{q}_t^i + e_t^i) \overline{\mathbf{o}}^j \tag{2.18}$$

where $\mathbf{T}_{n_b}^{b-}(\mathbf{w}_t, \mathbf{b}_t) \in SE(3)$ is the Lumped Error transform of all the first n_b joint errors and the base-to-camera transform calibration error, $\mathbf{T}_b^{b-}$, and it is parameterized by an orientation, $\mathbf{w}_t \in \mathbb{R}^3$, and translation, $\mathbf{b}_t \in \mathbb{R}^3$. The Lumped Error analytical solution from the joint angle errors and the error in base-to-camera transform is
$\mathbf{T}_{n_b}^{b-}(\mathbf{w}_t, \mathbf{b}_t) = \mathbf{T}_b^{b-}\mathbf{T}^{n_b}$, where $\mathbf{T}^{n_b}$ is defined in (2.12) with $\beta_i = 0$ for $i = 1, \dots, n_b$.

Intuitively, the Lumped Error transform is virtually adjusting the base of the kinematic chain for the robot in the camera frame. The virtual adjustments are made to fit the error of the first n_b joint angles and any error in base-to-camera transform. The Lumped Error transform removes the many-to-one mapping shown in (2.17). Furthermore, it significantly reduces the number of parameters that should be estimated for robotic tool tracking. Using a total of n_j joints and the $SE(3)$ error transforms, $\mathbf{T}_{b-}^{b}$ and $\mathbf{T}_{n_b}^{b-}(\mathbf{w}_t, \mathbf{b}_t)$, that are estimated based on the axis-angle and a translation vector, (2.3) has $n_j + 6$ parameters to estimate, whereas (2.18) has $n_j - n_b + 6$ parameters.

Even with this parameter reduction, it can still be challenging to constrain all of the parameters with image observations. For example, from a single image frame, four pixel point detections are required to constrain the Lumped Error transform [38], and additional point detections would be needed for the joint errors $e_t^{n_b+1}, \ldots, e_t^{n_j}$. Therefore, we propose the following simplification to (2.18) if there is not an abundance of features,

$$e_t^i \approx 0 \text{ for } i = n_b + 1, \ldots, n_j \tag{2.19}$$

With this simplification, only the Lumped Error transform needs to be estimated. This simplification can be applied in situations where the error from joints $n_b + 1, \ldots, n_j$ does not propagate through the kinematic chain dramatically. In cases where the camera focuses on an articulated wrist or gripper, as shown in Fig. 2.2, this assumption is acceptable because their link lengths are short, which reduces their sensitivity to error.

The expression obtained by combining the simplification in (2.19) with (2.18) is equivalent to what previous literature in robotic surgical tool tracking described as the KCS, which was developed for RCM-based robots [177]. Therefore, the KCS tracking formulation corrects not only for the base-to-camera transform error but also for the joint angle errors.

The Lumped Error can also be moved to the right-hand side of the first n_b joint transforms in (2.18), giving the following expression

$$\mathbf{T}^{c}_{b-} \prod_{i=1}^{n_b} \mathbf{T}^{i-1}_{i}(\tilde{q}^{i}_{t}) \mathbf{T}^{n_b,b-}_{n_b+1}(\mathbf{w}_t, \mathbf{b}_t) \prod_{i=n_b+1}^{j} \mathbf{T}^{i-1}_{i}(\tilde{q}^{i}_{t} + e^{i}_{t}) \overline{\mathbf{o}}^{j} \tag{2.20}$$

where the right hand-side Lumped Error, $\mathbf{T}^{n_b,b-}_{n_b+1}(\mathbf{w}_t, \mathbf{b}_t)$, is

$$\Big(\prod_{i=1}^{n_b} \mathbf{T}^{i-1}_{i}(\tilde{q}^{i}_{t}) \Big)^{-1} \mathbf{T}^{b-}_{n_b}(\mathbf{w}_t, \mathbf{b}_t) \prod_{i=1}^{n_b} \mathbf{T}^{i-1}_{i}(\tilde{q}^{i}_{t}) \tag{2.21}$$

which is equivalent to the tracking method proposed by Hao et al. [50] and shows that their method compensates for both base-to-camera transform errors and joint angle errors.

2.3.1 Extension to Robotic Camera Arm

In the case of the eye-in-hand configuration, the constant true base-to-camera transform, $\mathbf{T}^{c}_{b} = \mathbf{T}^{c}_{b-}\mathbf{T}^{b-}_{b}$, described in (2.3), is replaced with a kinematic chain as follows

$$\overline{\mathbf{o}}^{c}_{t} = \mathbf{T}^{c}_{c_n} \Big(\prod_{i=1}^{n} \mathbf{T}^{c_{i-1}}_{c_i}(q^{c_i}_{t}) \Big)^{-1} \mathbf{T}^{c_b}_{b} \prod_{i=1}^{j} \mathbf{T}^{i-1}_{i}(\tilde{q}^{i}_{t} + e^{i}_{t}) \overline{\mathbf{o}}^{j} \tag{2.22}$$

where $\mathbf{T}^{c}_{c_n} \in SE(3)$ is the static transform from the final joint to the camera frame, $\mathbf{T}^{c_{i-1}}_{c_i}(q^{c_i}_{t}) \in SE(3)$ is the i-th joint transform of the camera arm with joint angle $q^{c_i}_{t}$, and $\mathbf{T}^{c_b}_{b} \in SE(3)$ is the base-to-base transform (i.e., transform from the base of the robotic tool to the base of the robotic camera arm).

Calibration of the base-to-base transform is even more challenging than calibrating the base-to-camera transform since the kinematic chain is extended by the camera arm. Joint angle errors are also still assumed. Let $\tilde{q}^{c_i}_{t}$ and $e^{c_i}_{t}$ be the joint angle measurement and measurement error, respectively, for the joint angle c_i on the camera arm. The base-to-base transform $\mathbf{T}^{c_b}_{b}$ is split into the calibrated base-to-base transform, $\mathbf{T}^{c_b}_{b-}$, and the error in calibration, $\mathbf{T}_{b}^{b-}$. Therefore (2.22) is rewritten as

$$\overline{\mathbf{o}}_t^c = \mathbf{T}_{c_n}^{c}\Big(\prod_{i=1}^{n}\mathbf{T}_{c_i}^{c_{i-1}}(\tilde{q}_t^{c_i}+e_t^{c_i})\Big)^{-1}\mathbf{T}_{b-}^{c_b}\mathbf{T}_{b}^{b-}\prod_{i=1}^{j}\mathbf{T}_{i}^{i-1}(\tilde{q}_t^{i}+e_t^{i})\overline{\mathbf{o}}^j \tag{2.23}$$

The kinematic links from the camera arm are typically not visible in the camera frame. Therefore, the same non-identifiability issue from Claim 1 extends to joint angle errors $e_t^{c_i}$ for $i = 1, \dots c_n$. To resolve this issue, the Lumped Error from (2.18) is applied to the camera arm's kinematic chain. This modification results in

$$\overline{\mathbf{o}}_t^c = \mathbf{T}_{c_n}^{c}\Big(\prod_{i=1}^{n}\mathbf{T}_{c_i}^{c_{i-1}}(\tilde{q}_t^{c_i})\Big)^{-1}\mathbf{T}_{c_n}^{c_b}(\mathbf{w}_t^c,\mathbf{b}_t^c)^{-1}\mathbf{T}_{b-}^{c_b}\mathbf{T}_{n_b}^{b-}(\mathbf{w}_t,\mathbf{b}_t)\prod_{i=1}^{n_b}\mathbf{T}_{i}^{i-1}(\tilde{q}_t^{i}) \prod_{i=n_b+1}^{j}\mathbf{T}_{i}^{i-1}(\tilde{q}_t^{i}+e_t^{i})\overline{\mathbf{o}}^j \tag{2.24}$$

where the $\mathbf{T}_{c_n}^{c_b}(\mathbf{w}_t^c,\mathbf{b}_t^c)$ analytical expression from joint angles is described in (2.11) with $\beta_i = 0$ for $i = 1, \dots, n$. Continuing further, (2.24) can be reduced to a single unknown pose parameterized by orientation and translation vectors $\mathbf{w}_t^l, \mathbf{b}_t^l \in \mathbb{R}^3$, respectively, and the unknown joint errors e_t^i for $i = n_b + 1, \dots, n_j$. The new expression is

$$\overline{\mathbf{o}}_t^c = \mathbf{T}_{c_n}^{c}\Big(\prod_{i=1}^{n}\mathbf{T}_{c_i}^{c_{i-1}}(\tilde{q}_t^{c_i})\Big)^{-1}\mathbf{T}_{b-}^{c_b}\mathbf{T}_{n_b}^{c_n}(\mathbf{w}_t^l,\mathbf{b}_t^l)\prod_{i=1}^{n_b}\mathbf{T}_{i}^{i-1}(\tilde{q}_t^{i})\prod_{i=n_b+1}^{j}\mathbf{T}_{i}^{i-1}(\tilde{q}_t^{i}+e_t^{i})\overline{\mathbf{o}}^j \tag{2.25}$$

where

$$\mathbf{T}_{n_b}^{c_n}(\mathbf{w}_t^l,\mathbf{b}_t^l) = \big(\mathbf{T}_{b-}^{c_b}\big)^{-1}\mathbf{T}_{c_n}^{c_b}(\mathbf{w}_t^c,\mathbf{b}_t^c)^{-1}\mathbf{T}_{b-}^{c_b}\mathbf{T}_{n_b}^{b-}(\mathbf{w}_t,\mathbf{b}_t) \tag{2.26}$$

The Lumped Error estimated for this configuration, $\mathbf{T}_{n_b}^{c_n}(\mathbf{w}_t^l,\mathbf{b}_t^l)$, and the stationary camera Lumped Error described earlier have similar properties. The base of the robotic manipulator relative to the camera arm base is virtually adjusted to compensate for the error in the first n_b joint readings in it and all the joint readings in the robotic camera arm. This Lumped Error also reduces the number of parameters that should be estimated to $n_j - n_b + 6$, whereas in (2.23), there are $n_j + n + 6$ unknown parameters. To even further reduce the number of parameters to be estimated, the simplification in (2.19) can be applied, resulting in only 6 parameters.

2.3.2 Particle Filter

The result in (2.18) reduced the number of parameters that are required to be estimated for the Lumped Error transform, $\mathbf{T}_{n_b}^{b-}(\hat{\mathbf{w}}_t, \hat{\mathbf{b}}_t)$, and the joint errors: $\hat{\mathbf{e}}_t := [\hat{e}_t^{n_b+1}, \dots, \hat{e}_t^{n_j}]^\top$. These reductions make it possible to use previously developed parameter estimation methods for tracking, such as the Extended Kalman Filter, the Unscented Kalman Filter, or a particle filter, all of which can use updates from camera images. For our approach, we utilized a particle filter because of its flexibility to model the posterior probability density function with a finite number of samples [159] rather than using a parametric model, such as the Kalman Filter. Recently, particle filters have been successfully applied to estimate poses [17], which is especially needed here for the Lumped Error transform. The next sections describe the tracking of the parameters by defining the motion models and observation models. The last section covers the few modifications necessary for the eye-in-hand case. An outline of the proposed particle filter is shown in Algorithm 1.

Motion Model

The joint angle errors are initialized from a uniform distribution and have a motion model of additive zero mean Gaussian noise

$$\hat{\mathbf{e}}_0 \sim \mathcal{U}(-\mathbf{a}_{\hat{\mathbf{e}}}, \mathbf{a}_{\hat{\mathbf{e}}}) \qquad\qquad \hat{\mathbf{e}}_{t+1} \sim \mathcal{N}(\hat{\mathbf{e}}_t, \mathbf{\Sigma}_{\hat{\mathbf{e}},t+1}) \tag{2.27}$$

where $\mathbf{a}_{\hat{\mathbf{e}}} \in \mathbb{R}^{n_j - n_b}$ describes the bounds of constant joint angle error, and $\mathbf{\Sigma}_{\hat{\mathbf{e}},t+1} \in \mathbb{R}^{(n_j - n_b)\times(n_j - n_b)}$ is a covariance matrix. The initialization is done to capture the joint angle biases, and a Weiner Process is chosen for the motion model due to its ability

Algorithm 1: Particle Filter to Track Lumped Error

```
    Input  : Initial base-to-camera transform $\mathbf{T}^c_{b-}$
    Output : Estimated Lumped Error and Observable Joint Errors $\hat{\mathbf{w}}_t, \hat{\mathbf{b}}_t, \hat{\mathbf{e}}_t$
 1  Initialize particles $P_{0|0} = \{\alpha^{(p)}_{0|0}, \hat{\mathbf{w}}^{(p)}_{0|0}, \hat{\mathbf{b}}^{(p)}_{0|0}, \hat{\mathbf{e}}^{(p)}_{0|0}\}^N_{p=1}$
 2  for particle $p \in P_{0|0}$ do
 3      $\left[\hat{\mathbf{w}}^{(p)}_{0|0}, \hat{\mathbf{b}}^{(p)}_{0|0}\right]^\top \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_{\mathbf{w},\mathbf{b},0})$
 4      $\alpha^{(p)}_{0|0} \leftarrow \mathcal{G}\left(\left[\hat{\mathbf{w}}^{(p)}_{0|0}, \hat{\mathbf{b}}^{(p)}_{0|0}\right]^\top, \mathbf{\Sigma}_{\mathbf{w},\mathbf{b},0}\right)$
 5      $\hat{\mathbf{e}}^{(p)}_{0|0} \sim \mathcal{U}(-\mathbf{a}_{\hat{\mathbf{e}}}, \mathbf{a}_{\hat{\mathbf{e}}})$
 6  $\{\alpha^{(p)}_{0|0}\}^N_{p=1} \leftarrow normalizeWeights\left(\{\alpha^{(p)}_{0|0}\}^N_{k=1}\right)$
 7  while new image and joint readings, $(\mathbb{I}_t,\ \tilde{\mathbf{q}}_t)$, arrive do
        // Predict Particles
 8      Initialize new particles $P_{t|t-1} = \{\alpha^{(p)}_{t|t-1}, \hat{\mathbf{w}}^{(p)}_{t|t-1}, \hat{\mathbf{b}}^{(p)}_{t|t-1}, \hat{\mathbf{e}}^{(p)}_{t|t-1}\}^N_{p=1}$
 9      for particle $p \in P_{t|t-1}$ do
10          $q \sim P_{t-1|t-1}$ weights $\{\alpha^{(1)}_{t-1|t-1}, \dots, \alpha^{(N)}_{t-1|t-1}\}$
11          $\left[\hat{\mathbf{w}}^{(p)}_{t|t-1}, \hat{\mathbf{b}}^{(p)}_{t|t-1}\right]^\top \sim \mathcal{N}\left(\left[\hat{\mathbf{w}}^{(q)}_{t-1|t-1}, \hat{\mathbf{b}}^{(q)}_{t-1|t-1}\right]^\top, \mathbf{\Sigma}_{\mathbf{w},\mathbf{b},t}\right)$
12          $\hat{\mathbf{e}}^{(p)}_{t|t-1} \sim \mathcal{N}\left(\hat{\mathbf{e}}^{(q)}_{t-1|t-1}, \mathbf{\Sigma}_{\hat{\mathbf{e}},t}\right)$
13          $\alpha^{(p)}_{t|t-1} \leftarrow \mathcal{G}\left(\left[\hat{\mathbf{w}}^{(p)}_{t|t-1}, \hat{\mathbf{b}}^{(p)}_{t|t-1}\right]^\top, \mathbf{\Sigma}_{\mathbf{w},\mathbf{b},t}\right) \cdot \mathcal{G}\left(\hat{\mathbf{e}}^{(p)}_{t|t-1}, \mathbf{\Sigma}_{\hat{\mathbf{e}},t}\right)$
14      $\mathbf{m}_t \leftarrow detectRobotPointFeatures(\mathbb{I}_t)$
15      $\boldsymbol{\rho}_t, \boldsymbol{\phi}_t \leftarrow detectRobotEdgeFeatures(\mathbb{I}_t)$
        // Update Particles
16      for particle $p \in P_{t|t-1}$ do
17          $\hat{\mathbf{m}}_t \leftarrow projPoints(\hat{\mathbf{w}}^{(p)}_{t|t-1}, \hat{\mathbf{b}}^{(p)}_{t|t-1}, \hat{\mathbf{e}}^{(p)}_{t|t-1}, \tilde{\mathbf{q}}_t)$
18          $A_m, \mathbf{C}^m \leftarrow associatePoints(\mathbf{m}_t, \hat{\mathbf{m}}_t)$
19          $\alpha^{(p)}_{t|t-1} \leftarrow \alpha^{(p)}_{t|t-1} \cdot pointObsModel(A_m, \mathbf{C}^m)$
20          $\hat{\boldsymbol{\rho}}_t, \hat{\boldsymbol{\phi}}_t \leftarrow projEdges(\hat{\mathbf{w}}^{(p)}_{t|t-1}, \hat{\mathbf{b}}^{(p)}_{t|t-1}, \hat{\mathbf{e}}^{(p)}_{t|t-1}, \tilde{\mathbf{q}}_t)$
21          $A_l, \mathbf{C}^l \leftarrow associateEdges([\boldsymbol{\rho}_t, \boldsymbol{\phi}_t], [\hat{\boldsymbol{\rho}}_t, \hat{\boldsymbol{\phi}}_t])$
22          $\alpha^{(p)}_{t|t-1} \leftarrow \alpha^{(p)}_{t|t-1} \cdot edgeObsModel(A_l, \mathbf{C}^l)$
23      $P_{t|t} \leftarrow P_{t|t-1}$
24      $\{\alpha^{(p)}_{t|t}\}^N_{k=1} \leftarrow normalizeWeights\left(\{\alpha^{(p)}_{t|t}\}^N_{k=1}\right)$
25      $\left[\hat{\mathbf{w}}_t, \hat{\mathbf{b}}_t, \hat{\mathbf{e}}_t\right]^\top = \sum_{p=1}^{N} \alpha^{(p)}_{t|t} \left[\hat{\mathbf{w}}^{(p)}_{t|t}, \hat{\mathbf{b}}^{(p)}_{t|t}, \hat{\mathbf{e}}^{(p)}_{t|t}\right]^\top$
```

to generalize over a large number of random processes.

Let the tracked Lumped Error, $\mathbf{T}^{b-}_{n_b}(\hat{\mathbf{w}}_t, \hat{\mathbf{b}}_t)$, be represented by an axis angle vector, $\hat{\mathbf{w}}_t \in \mathbb{R}^3$, and a translation vector, $\hat{\mathbf{b}}_t \in \mathbb{R}^3$. Their initialization and motions are defined as

$$\begin{bmatrix}\hat{\mathbf{w}}_0, \hat{\mathbf{b}}_0\end{bmatrix}^\top \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_{\mathbf{w},\mathbf{b},0}) \qquad \begin{bmatrix}\hat{\mathbf{w}}_{t+1}, \hat{\mathbf{b}}_{t+1}\end{bmatrix}^\top \sim \mathcal{N}(\begin{bmatrix}\hat{\mathbf{w}}_t, \hat{\mathbf{b}}_t\end{bmatrix}^\top, \mathbf{\Sigma}_{\mathbf{w},\mathbf{b},t+1}) \tag{2.28}$$

where $\mathbf{\Sigma}_{\mathbf{w},\mathbf{b},t} \in \mathbb{R}^{6\times 6}$ is a covariance matrix. A Weiner Process is once again applied based on the same reason described above for the joint angle error motion model. Integration of the initial distribution and the motion model in the particle filter is shown in lines 1– 6 and 8– 13, respectively, in Algorithm 1.

Observation Model

To update the Lumped Error from images, features need to be detected, and a corresponding observation model must be defined for them. The described observation models can generalize for any point or edge features. Let $\mathbf{m}_t$ be a list of detected point features in the image frame from the projected robot tool. By following the standard camera pin-hole model combined with (2.18), the camera projection equation for the k-th point is

$$\hat{\mathbf{m}}_k(\hat{\mathbf{w}}_t, \hat{\mathbf{b}}_t, \hat{\mathbf{e}}_t) = \frac{1}{s}\mathbf{K}\mathbf{T}^c_{b-}\mathbf{T}^{b-}_{n_b}(\hat{\mathbf{w}}_t, \hat{\mathbf{b}}_t)\prod_{i=1}^{n_b}\mathbf{T}^{i-1}_i(\tilde{q}^i_t)\prod_{i=n_b+1}^{j_k}\mathbf{T}^{i-1}_i(\tilde{q}^i_t + \hat{e}^i_t)\overline{\mathbf{p}}^{j_k} \tag{2.29}$$

where $\frac{1}{s}\mathbf{K}$ is the camera projection operator with intrinsic matrix $\mathbf{K}$ and known location $\mathbf{p}^{j_k}$ on joint link j_k.

Similarly, let the paired lists $\boldsymbol{\rho}_t, \boldsymbol{\phi}_t$ be the parameters describing the detected edges in the image from the projected robot tool. The parameters describe an edge in the image frame using the Hough Transform [99], based on which the k-th pair, ρ^k_t and ϕ^k_t is used to parameterize the k-th detected edge with the following equation,

$$\rho_t^k = u \cos(\phi_t^k) + v \sin(\phi_t^k) \tag{2.30}$$

where (u, v) are pixel coordinates. Let the projection equations for the i-th edge be $\big(\hat{\rho}^i(\hat{\mathbf{w}}_t, \hat{\mathbf{b}}_t, \hat{\mathbf{e}}_t), \hat{\phi}^i(\hat{\mathbf{w}}_t, \hat{\mathbf{b}}_t, \hat{\mathbf{e}}_t)\big)$. These projection equations should be defined based on the geometry of the robot. An example of a cylindrical shape is shown in Appendix A.1. Furthermore, Chaumette derived the projection equations for a multitude of geometric primitives and can be referred to for additional shapes [13]. The point and the edge projections are computed on lines 17 and 20, respectively, in Algorithm 1.

The items on the lists of detected features, which may also contain false detections, should be associated with the correct point position ($\mathbf{p}^{j_i}$) or edge on the robot. This association can be established by generating a cost matrix, $\mathbf{C}^m$, between the detected and projected features. For the k-th detected point feature and i-th projected point, the cost is

$$C_{k,i}^m = \gamma_m ||\mathbf{m}_t^k - \hat{\mathbf{m}}_i(\hat{\mathbf{w}}_t, \hat{\mathbf{b}}_t, \hat{\mathbf{e}}_t)||^2 \tag{2.31}$$

where γ_m is a tuned parameter. Likewise, a cost matrix, $\mathbf{C}^l$, is computed for the edges, and for the k-th detected edge and the i-th projected edge, the associated cost is

$$C_{k,i}^l = \gamma_\rho |\rho_t^k - \hat{\rho}_i(\hat{\mathbf{w}}_t, \hat{\mathbf{b}}_t, \hat{\mathbf{e}}_t)| + \gamma_\phi |\phi_t^k - \hat{\phi}_i(\hat{\mathbf{w}}_t, \hat{\mathbf{b}}_t, \hat{\mathbf{e}}_t)| \tag{2.32}$$

where γ_ρ and γ_ϕ are tuned parameters.

The associations between the detected and projected features are established with a greedy matching technique, which is used because of its computational efficiency. The costs are sorted from lowest to highest, and the first (k, i) pair is matched and added to the set A_m or A_l for points and edges, respectively. All subsequent costs associated with either detection k or projection i are removed from the sorted list. This is repeated until a maximum cost of C^m_{max} or C^l_{max} is reached for points and edges, respectively. False detections can be filtered out by limiting the maximum cost for association. This association technique is conducted on lines 18 and 21 in Algorithm 1 for points and edges, respectively.

The observation model wraps the associations and their costs into a probability function dependent on the state, which ensures that the filter can properly update the states. For the list of point features, the probability is

$$P(\mathbf{m}_t|\hat{\mathbf{w}}_t, \hat{\mathbf{b}}_t, \hat{\mathbf{e}}_t) \propto (n_m - |A_m|)e^{-C^m_{max}} + \sum_{k,i \in A_m} e^{-C^m_{k,i}} \tag{2.33}$$

where there are a total of n_m detectable point features on the robot. Similarly, the probability of the list of detected edges is

$$P(\boldsymbol{\rho}_t, \boldsymbol{\phi}_t|\hat{\mathbf{w}}_t, \hat{\mathbf{b}}_t, \hat{\mathbf{e}}_t) \propto (n_l - |A_l|)e^{-C^l_{max}} + \sum_{k,i \in A_l} e^{-C^l_{k,i}} \tag{2.34}$$

where there are a total of n_l detectable edge features on the robot. The probability distributions can be viewed as a summation of Gaussians centered on the projected features. The individual Gaussian probabilities are bounded and clipped by the maximum cost for association. Clipping the Gaussians is preferred since in the cases of missed feature detections, the posterior probability from the filter does not go to zero. An additional advantage of using a particle filter for tracking the Lumped Error is that these observation models do not need to be normalized. In this case, finding the normalization factor would be challenging due to the matching complexity and clipping of Gaussians. These observation models update the particle filter on lines 19 and 22 in Algorithm 1 for points and edges, respectively.

Modifications for Eye-in-Hand Configuration

The explicitly tracked joint errors, $\hat{\mathbf{e}}_t$, remain the same since they are still the only joints visible in the camera frame. However, the tracked pose is now $\mathbf{T}^{c_n}_{n_b}(\hat{\mathbf{w}}^l_t, \hat{\mathbf{b}}^l_t)$, as described in (2.25). The tracked parameters $\hat{\mathbf{w}}^l_t, \hat{\mathbf{b}}^l_t \in \mathbb{R}^3$ represent the Lumped Error as axis-angle and translation vectors, respectively, and they have the same additive zero mean Gaussian noise, as described in (2.28). The feature detection, association, and observation models all remain unchanged. The only change required is modifying the camera projection equations. The camera projection equation for the i-th marker is changed from (2.29) to

$$\hat{\mathbf{m}}_k(\hat{\mathbf{w}}^l_t, \hat{\mathbf{b}}^l_t, \hat{\mathbf{e}}_t) = \frac{1}{s}\mathbf{K}\mathbf{T}^{c}_{c_n}\Big(\prod_{i=1}^{n}\mathbf{T}^{c_i}_{c_{i-1}}(\tilde{q}^{c_i}_t)\Big)^{-1}\mathbf{T}^{c_b}_{b-}\mathbf{T}^{c_n}_{n_b}(\hat{\mathbf{w}}^l_t, \hat{\mathbf{b}}^l_t)\prod_{i=1}^{n_b}\mathbf{T}^{i-1}_{i}(\tilde{q}^i_t) \prod_{i=n_b+1}^{j_k}\mathbf{T}^{i-1}_{i}(\tilde{q}^i_t + \hat{e}^i_t)\overline{\mathbf{p}}^{j_k} \quad (2.35)$$

by combining (2.25) with the camera pin-hole model. A similarly simple modification is required for the projected edges $\big(\hat{\rho}^i(\hat{\mathbf{w}}_t, \hat{\mathbf{b}}_t, \hat{\mathbf{e}}_t), \hat{\phi}^i(\hat{\mathbf{w}}_t, \hat{\mathbf{b}}_t, \hat{\mathbf{e}}_t)\big)$. The example shown in Appendix A.1 for cylindrical shapes includes the necessary modifications.

2.3.3 Experimentation

Implementation Details

The proposed particle filter in all of the experiments shown here ran on a Intel® Core™ i9-7940X Processor and NVIDIA's GeForce RTX 2080 with a loop rate of 24FPS and was used to track the Lumped Error in a simulated scene of a da Vinci® Surgical System and on a real-world dVRK [65]. The uncertainties of joint angles on the dVRK system are so prevalent that results relying only on base-to-camera calibration and not accounting for the joint angle error were intentionally omitted from a previous work due to the poor results [145]. The parameters used to describe the motion models and observation models are as follows:

- $\mathbf{a}_{\hat{\mathbf{e}}} = \begin{bmatrix} 0.004 & 0.004 & 2 & 0.004 & 0.004 & 0.004 & 0.01 \end{bmatrix}$
- $\mathbf{\Sigma}_{\hat{\mathbf{e}},t} = \text{diag}\big(\begin{bmatrix} 0.0025 & 0.0025 & 1 & 0.0025 & 0.0025 & 0.0025 & 0.005 \end{bmatrix}\big)$
- $\mathbf{a}_{\hat{\mathbf{e}}^c} = \begin{bmatrix} 0.004 & 0.004 & 2 & 0.004 \end{bmatrix}$
- $\mathbf{\Sigma}_{\hat{\mathbf{e}}^c,t}\text{diag}\big(\begin{bmatrix} 0.01 & 0.01 & 2.5 & 0.01 \end{bmatrix}\big)$
- $\mathbf{\Sigma}_{\mathbf{w},\mathbf{b},t} = \text{diag}\big(\begin{bmatrix} \mathbf{\Sigma}_{\mathbf{w},t} & \mathbf{\Sigma}_{\mathbf{b},t} \end{bmatrix}\big)$
- $\mathbf{\Sigma}_{\mathbf{b},t} = \text{diag}\big(\begin{bmatrix} 0.25 & 0.25 & 0.25 \end{bmatrix}\big)$
- $\mathbf{\Sigma}_{\mathbf{w},\mathbf{b},0} = 10(\mathbf{\Sigma}_{\mathbf{w},\mathbf{b},t})$
- $\begin{bmatrix} \gamma_m & \gamma_\phi & \gamma_\rho \end{bmatrix} = \begin{bmatrix} 0.15 & 40.0 & 0.1 \end{bmatrix}$
- $\begin{bmatrix} C^m_{max} & C^l_{max} \end{bmatrix} = \begin{bmatrix} 25\gamma_m & 0.1\gamma_\phi + 25\gamma_\rho \end{bmatrix}$
- Number of particles is set to 1000

The only modification in the non-stationary robotic endoscope case was applied to the covariances of the Lumped Errors, $\mathbf{\Sigma}_{\mathbf{w},t}$ and $\mathbf{\Sigma}_{\mathbf{b},t}$, which were scaled by 2. Note that the relationship of $\mathbf{\Sigma}_{\mathbf{w},\mathbf{b},0} = 10(\mathbf{\Sigma}_{\mathbf{w},\mathbf{b},t})$ was still intact after the scaling. The markers are located in similar locations as the detected features used in the previous work by Ye et al. [169]. All marker locations relative to the joint coordinate frames, $\mathbf{p}^{j_i}$ from (2.29), were measured using calipers on the dVRK.

Datasets

Simulation: A simulated scene in V-REP [138] was developed based on the da Vinci robot model constructed by Fontanelli et al. [32]. The simulated robotic tool and camera arm were a Patient Side Manipulator (PSM) with a Large Needle Driver and an Endoscopic Camera Manipulator (ECM), respectively, from the da Vinci® Surgical

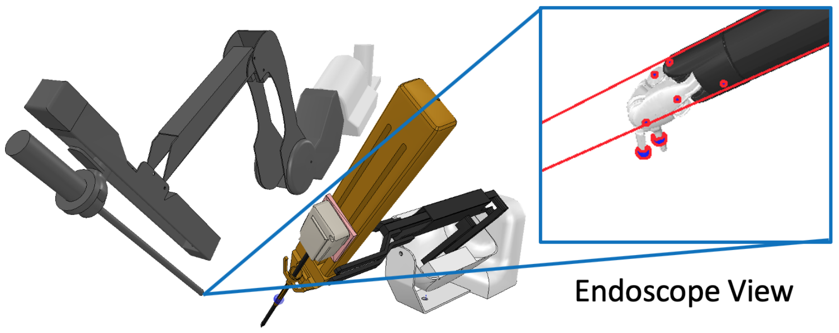


Figure 2.3. A simulated scene in V-REP [32] of a Patient Side Manipulator (PSM) and Endoscopic Camera Manipulator (ECM) from a da Vinci® Surgical System.

System. The PSM has 6 DoF and an additional gripper joint. The ECM is stereoscopic and has 4 DoF. The first joint link visible in the endoscopic camera frame was after the $n_b = 4$ joint, as expected when operating with a da Vinci® Surgical System. The stereoscopic endoscope's virtual cameras were set to render 540 by 432 images with a field of view of 60 degrees. The baseline distance for the stereo cameras was set to 5 mm, which simulated depth challenges that resemble those in real stereoscopic endoscopes.

Small blue spheres were placed as markers along the kinematic links near the gripper to be used as point features to update the particle filter. The blue markers were detected using the standard color segmentation from OpenCV [9]. Each camera image was initially converted to the Hue, Saturation, and Value (HSV) color space. Hand-tuned lower and upper bounds for each HSV channel were then applied to the image resulting in a segmented binary image. The segmented binary image was then clustered into distinct contours from which the centroids were estimated. The listed centroids, $\mathbf{m}_t$, were considered detected pixel coordinate features potentially derived from projected points on the surgical tool. The edges of the projected cylindrical insertion shaft of the PSM tool were also used to update the particle filter and detected using standard OpenCV functionality [9]. All pixels potentially associated with the edges were detected using the Canny edge detector [11]. The pixels were further classified into distinct edges using the Hough Transform [99] with parameters ρ_t^k and ϕ_t^k to fit (2.30). The simulated scene and a corresponding camera image with the detected features is shown in Fig. 2.3.

The error in calibration, $\mathbf{T}_b^{b-}$, was obtained by sampling from zero mean Gaussian in its axis angle and translation vector representations

$$\left[\mathbf{w}^{b-}, \mathbf{b}^{b-}\right]^\top \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_{\mathbf{w},\mathbf{b}}^{b-}) \tag{2.36}$$

Therefore, the initial calibration applied to the filter was set to

$$\begin{aligned} \mathbf{T}_{b-}^{c} &= \mathbf{T}_b^c\Big(\mathbf{T}_b^{b-}(\mathbf{w}^{b-}, \mathbf{b}^{b-})\Big)^{-1} \\ \mathbf{T}_{b-}^{c_b} &= \mathbf{T}_b^{c_b}\Big(\mathbf{T}_b^{b-}(\mathbf{w}^{b-}, \mathbf{b}^{b-})\Big)^{-1} \end{aligned} \tag{2.37}$$

for the stationary camera and the eye-in-hand cases, respectively, where $\mathbf{T}_b^c$ and $\mathbf{T}_b^{c_b}$ were derived by the simulator. The joint error for the PSM was simulated as a summation between a uniformly sampled bias at the start of each trial and a linear cable stretch. Written explicitly, the error for joint angle i was defined as

$$e_t^i = e_b^i + e_c^i q_t^i \tag{2.38}$$

where $e_b^i \sim \mathcal{U}(-a_e^{i,b}, a_e^{i,b})$, e_c^i was the linear cable stretch coefficient, and q_t^i was the correct joint angle from the PSM. Similarly, the joint error c_i for the ECM was defined as

$$e_t^{c_i} = e_b^{c_i} + e_{l,t}^{c_i} \tag{2.39}$$

where $e_b^{c_i} \sim \mathcal{U}(-a_e^{c_i,b}, a_e^{c_i,b})$ was sampled once at the start of each trial and $e_{l,t}^{c_i} \sim \mathcal{N}(0, \sigma_{c_i,l}^2)$ sampled at every time step to simulate the uncertainties associated with the joint angles of the robotic endoscope. The values for the noise parameters are set to as follows:

- $\mathbf{\Sigma}_{\mathbf{w},\mathbf{b}}^{b-} = \mathrm{diag}\big(\begin{bmatrix}0.005 & 0.005 & 0.005 & 5 & 5 & 5\end{bmatrix}\big)$

- $\mathbf{a}_e^b = \begin{bmatrix}0.004 & 0.004 & 2 & 0.004 & 0.004 & 0.004 & 0.01\end{bmatrix}$
- $\mathbf{e}_c = \begin{bmatrix}0.02 & 0.02 & 0.0025 & 0.02 & 0.02 & 0.02 & 0.05\end{bmatrix}$
- $\mathbf{a}_e^c = \begin{bmatrix}0.004 & 0.004 & 2 & 0.004\end{bmatrix}$
- $\sigma_{c,l} = \begin{bmatrix}0.0075 & 0.0075 & 0.75 & 0.0075\end{bmatrix}$

The PSM arms configuration was set via V-REP's inverse kinematics. Its position moved along a preset cyclical trajectory added with a small, random sample from a zero mean Gaussian with a standard deviation of 1 mm. The gripper joint opened and closed at a similar cyclical rate. Likewise, the four joint angles of the ECM were set to move in a cyclical pattern in the eye-in-hand case. The orientation of the PSM end-effector instead performed a random walk, starting at a preset value by rotating an additional, uniformly sampled rotation at every time step. The random walk performed via the orientation, $\mathbf{w}_t^e$ a quaternion vector, of the end-effector was defined as

$$\mathbf{w}_{t+1}^e = \mathbf{w}_t^e \mathbf{w}_t^n \tag{2.40}$$

where $\mathbf{w}_t^n$ was the quaternion representation of the axis-angle vector whose angle was sampled from $\mathcal{U}(0, 0.07)$ radians, and the axis was uniformly sampled in spherical coordinates

$$\begin{bmatrix}\sin(\phi_t^n)\cos(\theta_t^n) & \sin(\phi_t^n)\sin(\theta_t^n) & \cos(\phi_t^n)\end{bmatrix}^\top \tag{2.41}$$

where $\theta_t^n = \arccos(u_t)$, $u_t \sim \mathcal{U}(-1, 1)$, and $\phi_t^n \sim \mathcal{U}(0, 2\pi)$. The trajectory per trial was ran for 140 time steps. This simulation represented the most complex scenario where all robotic components (manipulator, gripper, camera) were continuously moving on independent paths, which allowed the testing of the proposed tracking method in a larger variety of scenarios, including the occlusion of features.

To evaluate the effectiveness of pose or transform estimation, the error was calculated at time t as

$$\epsilon_{\mathbf{b}} = ||\mathbf{b}_t - \hat{\mathbf{b}}_t|| \qquad\qquad \epsilon_{\mathbf{w}} = ||\mathbf{w}_t^r|| \tag{2.42}$$

where $\mathbf{w}_t^r$ is the axis angle representation of $\mathbf{R}_t(\hat{\mathbf{R}}_t)^{-1}$, $\mathbf{b}_t \in \mathbb{R}^3$ and $\mathbf{R}_t \in SO(3)$ are the ground truth translation vector and rotation matrix, respectively, and $\hat{\mathbf{b}}_t \in \mathbb{R}^3$ and $\hat{\mathbf{R}}_t \in SO(3)$ are the tracked translation vector and rotation matrix, respectively. The ith joint angle error was computed as

$$\epsilon_{q^i} = |\hat{q}_t^i - q_t^i| \tag{2.43}$$

The simulation was repeated 50 times for both the stationary and moving endoscope cases.

dVRK: Two one-minute segments of encoder readings and stereoscopic data from the endoscope on an ECM were captured from a dVRK [65]. The stereoscopic camera system used the standard dVRK endoscopic lens and had a resolution of 1920 by 1080 pixels at 30FPS. In both sequences, a single PSM arm was teloperated with the gripper in full view of the stereoscopic camera. In the first sequence, the PSM arm traveled a total distance of 48 mm, and the ECM was stationary. In the second sequence, the PSM arm traveled a total distance of 49 mm, and the ECM arm joint angles were set to sinusoidal patterns similar to the previous simulation experiment resulting in 35 mm for a total distance traveled. The PSM arm had blue-colored markers in the same positions as the simulated scene. The markers and edges of the projected cylindrical insertion shaft were detected in the same manner as the simulated scene. The initial calibrations, $\mathbf{T}_{b-}^{c}$ and $\mathbf{T}_{b-}^{c_b}$, were computed using OpenCV's solvePnP [9] with manually set associations of the markers. When deploying a fully assembled da Vinci® Surgical System, we envision that these initial transformations are computed from the set-up joints that connect the ECM with the PSM arm. However, the dVRK, by default, is not set up with joints, which is why the

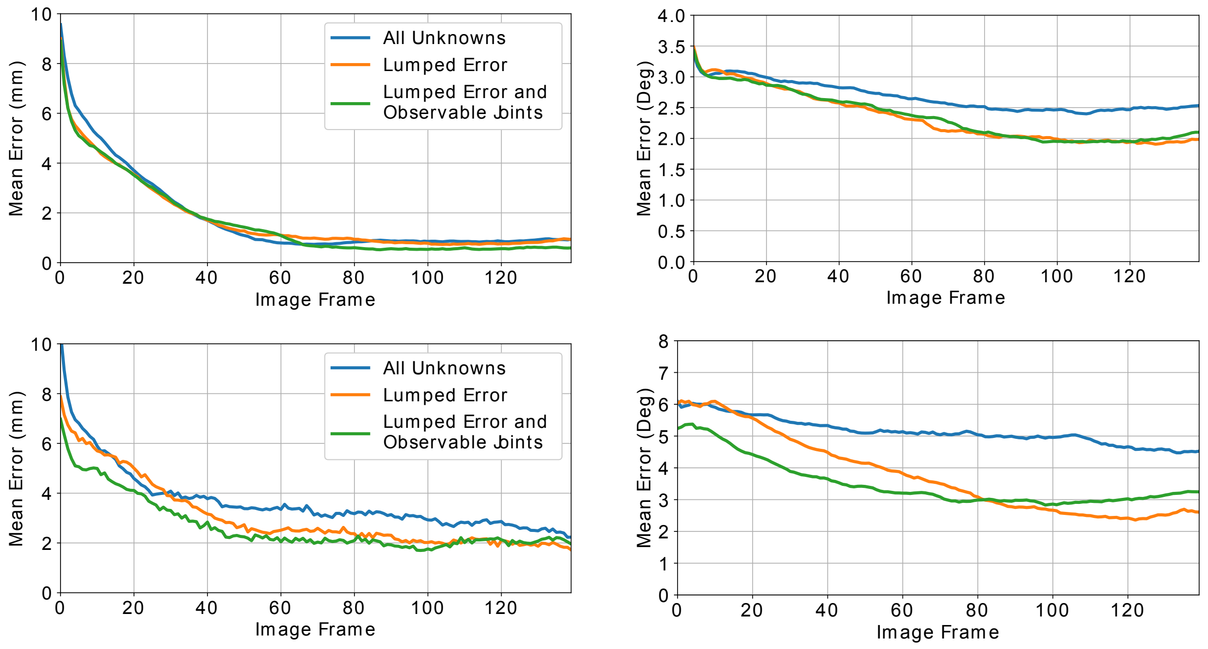


Figure 2.4. Mean end-effector pose error in the camera frame over time under various tracking configurations from simulated da Vinci scene. The top and bottom row of plots were measured using the stationary and the eye-in-hand setup, respectively.

sovlePnP was used with manually set associations for initialization.

From both sequences, 20 evenly distributed images were manually annotated using the VGG labeler [25]. These labels, $\mathbb{I}_G$, were considered ground truth, and IoU was used as the metric in this experiment,

$$\text{IoU}_{2D} = \frac{\mathbb{I}_R \cap \mathbb{I}_G}{\mathbb{I}_R \cup \mathbb{I}_G} \tag{2.44}$$

where $\mathbb{I}_R$ was a generated mask using the rendering procedure described in [137]. The generated mask was rendered using the tracked parameters.

Comparison Study

The evaluated particle filter configurations were:

- *All Unknowns*: tracking all joint angle errors and base-to-camera transform or base-to-base in the stationary and eye-in-hand case, respectively. Done by setting

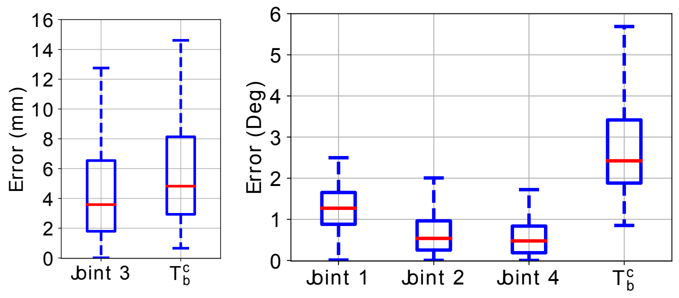


Figure 2.5. Distribution of the first 4 joint angle errors, whose preceding kinematic links were never in the camera frame, and the stationary camera-to-base transform $\mathbf{T}_b^c$ error when explicitly estimating them in the simulated da Vinci scene.

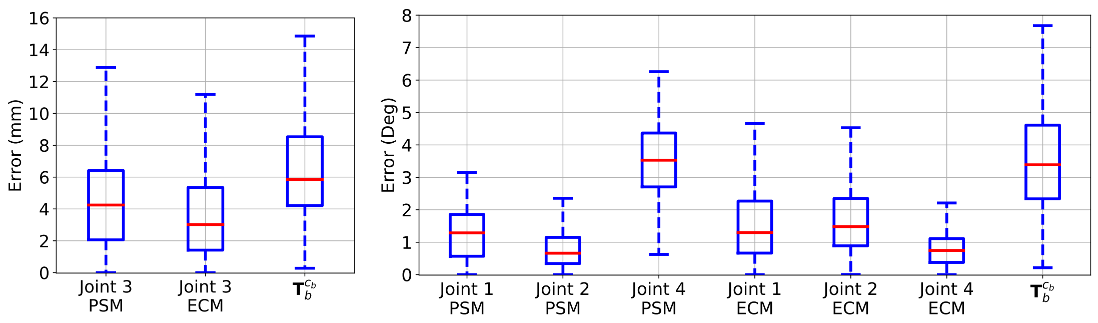


Figure 2.6. Box plots of the tracked joint angle errors, whose preceding kinematic links were never in the camera frame, and the base of the camera arm to the base of the robotic tool transform $\mathbf{T}_b^{c_b}$ error when explicitly estimating all unknowns in the simulated da Vinci scene.

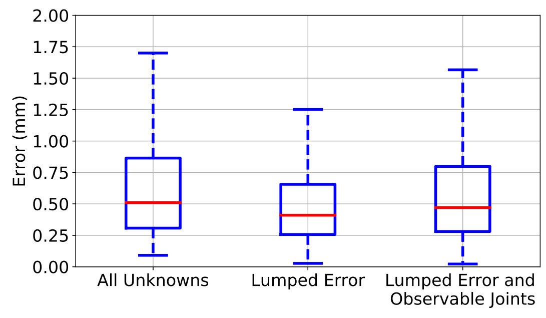
$n_b = 0$ in the particle filter.

- *Lumped Error*: applying (2.19) to the particle filter
- *Lumped Error and Observable Joints*: no modifications to the described particle filter

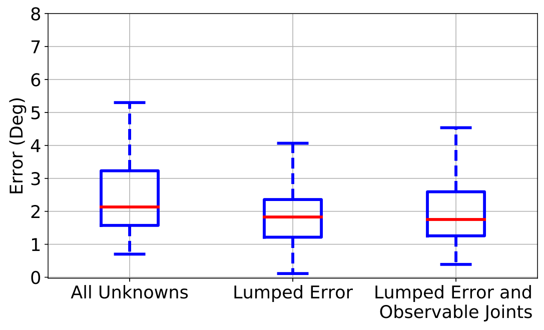
Both stationary camera and moving camera arm scenarios were tested for all three comparisons.

Results

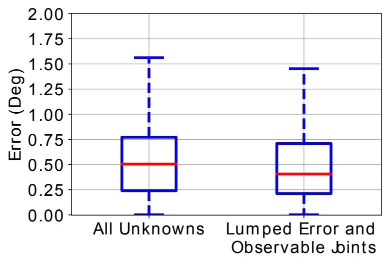
The mean end-effector pose error plots are shown in Fig. 2.4 for both the stationary camera and the eye-in-hand cases in the simulation. These mean error trends were



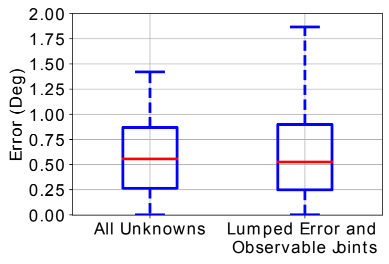
(a) End-effector position error



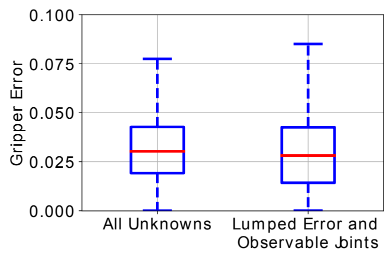
(b) End-effector orientation error



(c) Joint 5 error



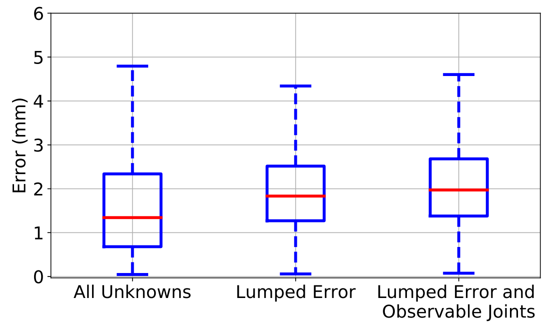
(d) Joint 6 error



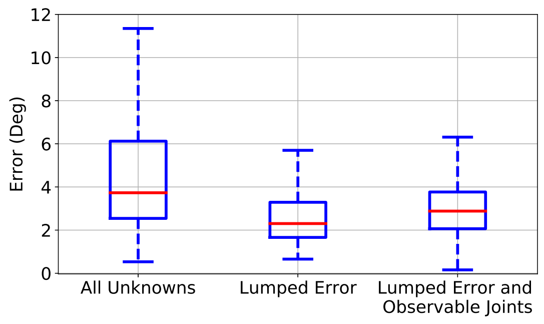
(e) Joint 7 error

Figure 2.7. Box plots of converged tracking performance under various configurations for the particle filter in simulated da Vinci for the stationary camera.

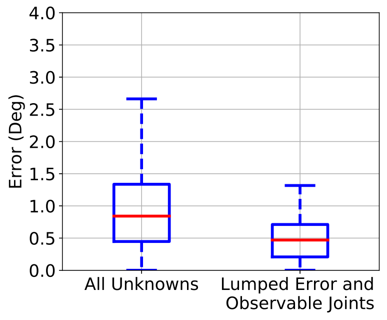
calculated including all 50 trials, which demonstrated that tracking the Lumped Error lowered end-effector orientation error value compared to that error value obtained when tracking all unknowns. Fig. 2.5 and 2.6 show the distributions of errors for the non-identifiable joint angles and the base-to-camera transform in the stationary and eye-in-hand cases, respectively, when explicitly tracking all unknown parameters in the simulation. These values were calculated across 40 time steps from all 50 trials after performing 100 time steps to give enough time for the particle filter to converge. Errors of up to 14 mm and 7 degrees highlight the inability to estimate these unknown values explicitly due to the parameters being non-identifiable, as shown in Claim 1. These errors displayed a large spread, although the particle filter was still able to sufficiently track the end-effector as seen in the mean end-effector pose error plots in Fig. 2.4. This observation supports Claim 1 by showing that it is infeasible to explicitly track all the unknown parameters from robotic tools with limited visibility since they cannot converge to their



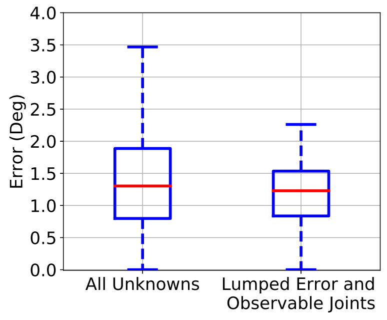
(a) End-effector position error



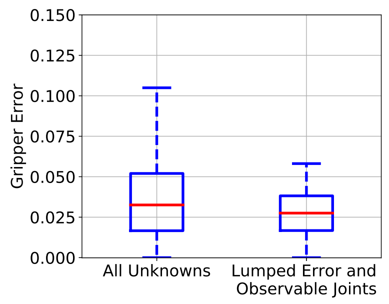
(b) End-effector orientation error



(c) Joint 5 error



(d) Joint 6 error



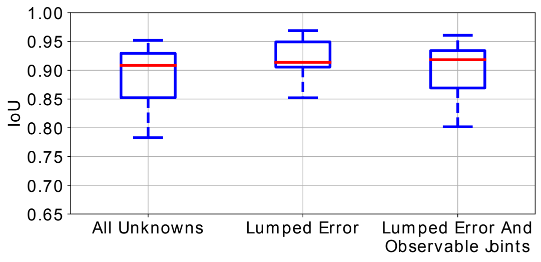
(e) Joint 7 error

Figure 2.8. Box plots of converged tracking performance under various scenarios for the particle filter in a simulated da Vinci scene for the eye-in-hand configuration.

true values.

The distribution of end-effector tracking errors after giving the particle filter enough time to converge in the same manner as previously described are shown in Figs. 2.7 and 2.8 for simulation scenarios with a stationary camera and a moving camera arm, respectively. The end-effector positional error does not substantially differ between the converged error distributions. Importantly, the end-effector orientation error was clearly improved by using the Lumped Error estimation for both the stationary camera and the robotic camera arm cases. For the observable joints, i.e., joints $5, 6, 7$, the Lumped Error tracking method showed no significant difference in error between the stationary camera and the eye-in-hand cases. Meanwhile, when explicitly tracking all unknowns, the error from the observable joints was significantly worse for the eye-in-hand configuration.

The distributions of IoU are shown in Fig. 2.9 for both the stationary and the



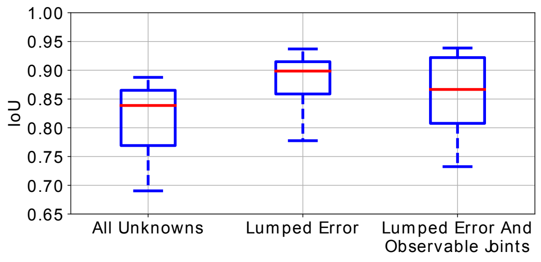
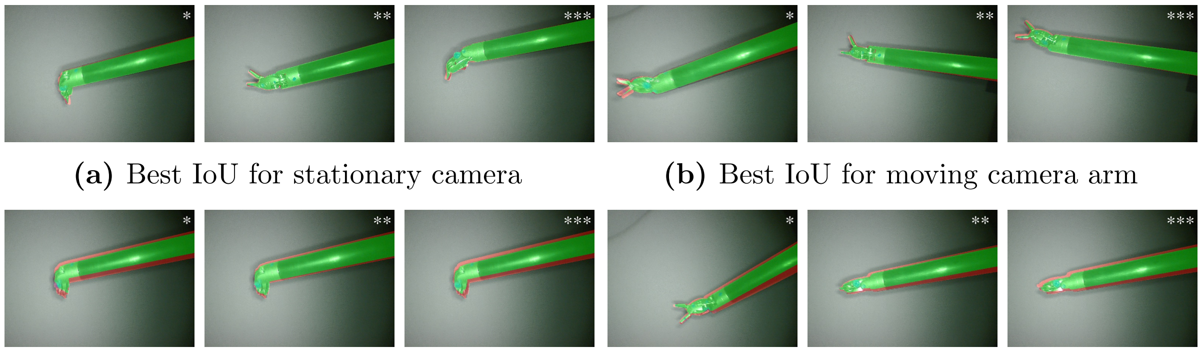


Figure 2.9. The left and right plots show the distribution of the Intersection over Union (IoU) between manual annotations and reprojected renderings from the tracked values under various particle filter configurations on the dVRK [65].



(a) Best IoU for stationary camera **(b)** Best IoU for moving camera arm

(c) Worst IoU for stationary camera **(d)** Worst IoU for moving camera arm

Figure 2.10. Images from the best and worst IoU from tracking the dVRK surgical tool [65] when using All Unknowns*, Lumped Error**, and Lumped Error plus Observable Joints***. The green and red regions are the intersection and the union minus intersection, respectively.

moving camera arm cases on the dVRK. Examples of surgical tool renderings on the top of the image feed are shown in Fig. 2.10. Similar to the simulation results, the Lumped Error clearly performed the best. Furthermore, the performance of applying the simplification in (2.19) is seen in both the stationary and moving camera cases.

2.4 Discussion

The experimental results and their respective metrics clearly indicate that using Lumped Error yields almost always better tool tracking than explicitly estimating all unknowns for both the stationary camera and the eye-in-hand configurations. Only one experimental metric, shown in Fig. 2.8a, performed marginally better when tracking all unknowns. Nevertheless, these metrics cannot be viewed in isolation, and the respectively paired metrics, as presented in Fig. 2.8b-e, were significantly improved when tracking the Lumped Error compared with tracking all unknowns. Moreover, the non-identifiable values estimated when tracking all unknowns yielded non-realistic results, as shown in Figs. 2.5 and 2.6. Thus, it is a clear improvement to derive a single solution from the parameters being estimated when tracking the Lumped Error rather than the infinite set of solutions when tracking all unknowns, as shown in Claim 1. Furthermore, tracking all unknowns, which generated the infinite set of solutions, resulted in large distributions of the parameters being estimated, which is shown in Figs. 2.5 and 2.6. From the perspective of the particle filter, this is an inefficient usage of particles and detrimental to tracking because the particle filter estimates the posterior probability using a finite number of samples.

When using the Lumped Error, the end-effector accuracy varied very little between tracking the observable joints and not tracking them in the da Vinci simulation. This result supports the validity of applying the simplification in (2.19) to the da Vinci robot because the link lengths for the observable joints, a dexterous robotic gripper, are short. However, not tracking the observable joints on the real-world dVRK performed better. We believe this occurred due to the features not being detected as consistently in the real world as in the simulation, which highlights the usefulness of the simplification in (2.19).

The proposed tracking method to estimate the Lumped Error, which used a Weiner Process to model the uncertainty, was experimentally efficient. In our dVRK experiment, the Weiner process compensated for the joint angle errors due to cable stretch. In Appendix A.2, we applied the same approach to the Baxter Robot, where only the gripper is visible in the camera view. The Baxter robot had a significant backlash which was successfully compensated by using the proposed tracking method.

2.5 Acknowledgements

Chapter 2, in part, is a reprint of material from F. Richter, J. Lu, R. K. Orosco, M. C. Yip "Robotic tool tracking under partially visible kinematic chain: A unified approach," in *IEEE Transactions on Robotics*, vol. 38, no. 3, pp. 1653-1670, 2022. The dissertation author is the primary author of this paper.

Chapter 3

SuPer: A Surgical Perception Framework

Significant advances have been made in control and task automation for surgical robots. However, the integration of perception into these controllers is deficient despite the remarkable progress in expanding the capabilities of surgical tool and tissue tracking technologies over the past decade. Without properly integrating perception, control algorithms will never be successful in non-structured environments, such as those under surgical conditions. To overcome the perception challenges, we propose a novel Surgical Perception framework, SuPer, which integrates visual perception from endoscopic image data for a surgical robotic control loop to achieve tissue manipulation. A vision-based tracking system is carefully designed to track both the surgical environment and robotic agents, e.g., the tissue and a surgical tool, as shown in Fig. 3.1. Endoscopic procedures have limited sensory information derived from endoscopic images and take place in a constantly deforming environment. Therefore, we split the tracking system into two methodologies: model-based tracking proposed in Chapter 2 to leverage the available kinematic prior of the agent and model-free tracking for the unstructured physical world. Equipped with the proposed 3D visual perception framework, surgical robotic controllers can manipulate the environment in a closed-loop fashion as the framework maps the environment, tracks the tissue deformation, and localizes the agent continuously and

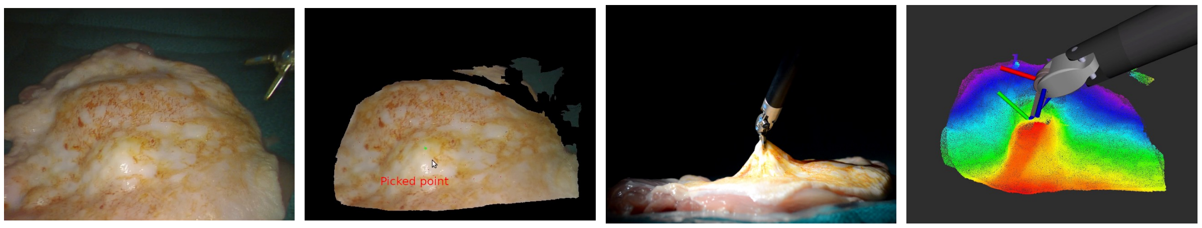


Figure 3.1. A demonstration of the proposed surgical perception framework. A green point on the reconstructed deformable tissue, shown in the second image from the left, is selected by the user. The surgical robot autonomously grasps the tissue at that location.

simultaneously. In the experimental section, we also demonstrate an efficient implementation of the proposed framework on a dVRK in which we successfully manipulate tissue.

To the best of our knowledge, the proposed perception framework is the first work to combine 3D visual perception algorithms for the general control of a surgical robot in an unstructured, deforming environment. More specifically, our contributions can be summarized as

1. a perception framework with both model-based tracking and model-free tracking components to track the tissue and localize the robot simultaneously,
2. deformable environment tracking to track tissue from stereo-endoscopic image data,
3. a released dataset of tissue manipulation with the da Vinci Surgical® System.[1]

The framework was implemented on a da Vinci Surgical® System, and multiple tissue manipulation experiments were conducted to highlight its accuracy and precision. We believe that the proposed framework is a fundamental step toward endoscopic surgical autonomy in unstructured environments. With a uniform perception framework in the control loop, more advanced surgical task automation can be achieved.

[1]Website: https://www.sites.google.com/ucsd.edu/super-framework

3.1 Related Works

Deformable Reconstruction

The first group of related works covers 3D reconstruction and motion capture [112, 140, 182, 181]. Newcombe et al. [111] proposed a real-time method for the reconstruction of a static 3D model using a consumer-level depth camera based on volumes for their internal data structure, while Keller et al. [68] used surfel points rather than volumes. The rigidness assumption was then removed to capture the motion of a deforming scene [110]. To enhance the robustness of the reconstruction, a keypoint alignment function was added to the original cost function of the deformable reconstruction [59]. In addition, multiple-sensor approaches have been shown to further improve accuracy [24]. Guo et al. [37] achieved similar results for deformable object reconstruction with surfel points.

Endoscopic Tissue Tracking

Tissue tracking is a specific area of visual tracking that often utilizes 3D reconstruction techniques. A comprehensive evaluation of different optical techniques for geometry estimation of tissue surfaces concluded that stereoscopic imaging is the only feasible and practical approach to tissue reconstruction and tracking during surgery [97]. For image-guided surgery, Yip et al. [172] proposed a tissue tracking method with keypoint feature detection and registration. 3D dynamic reconstruction was introduced by Song et al. [151] to track in-vivo deformations. Moreover, dense SLAM methods [96, 98] were applied to track and localize the endoscope in the surgical scene using image features. In contrast to the algorithms mentioned above, our proposed framework not only tracks the surgical environment through deformable reconstruction but also integrates the control loop of the surgical robotic arm for automation.

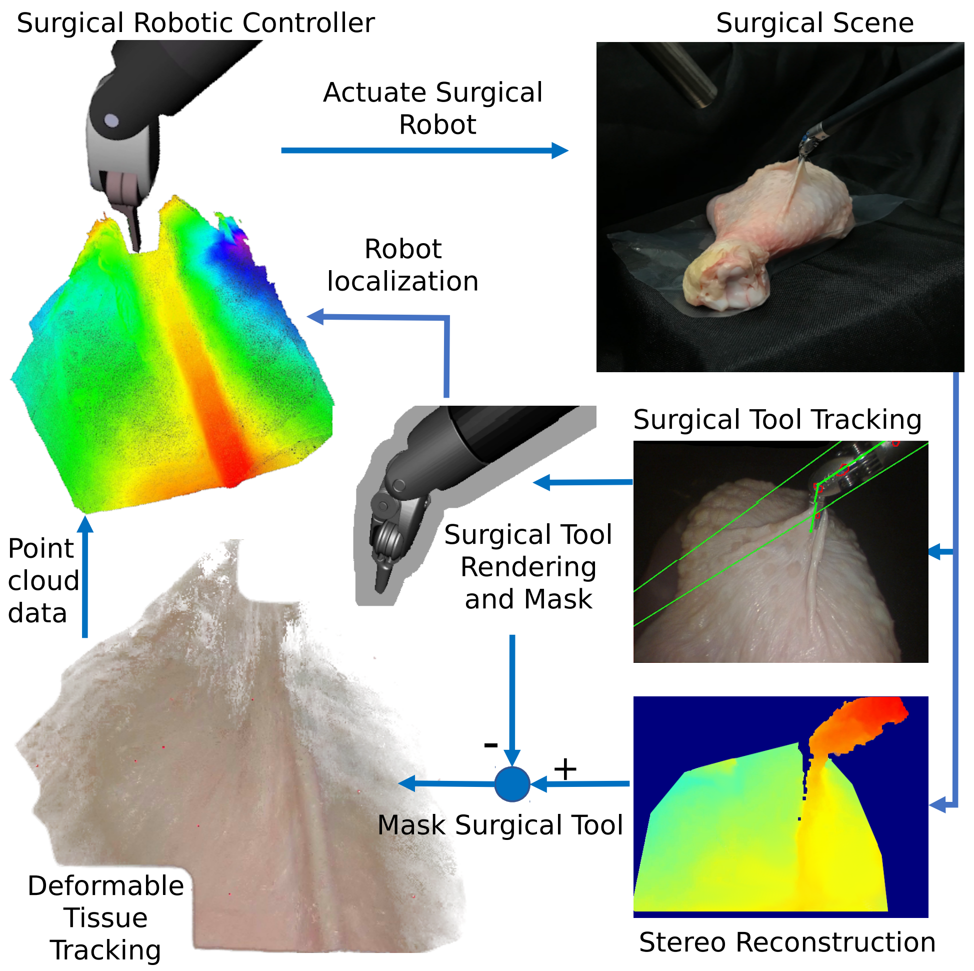


Figure 3.2. The flow chart of the proposed SuPer framework illustrates the integration of perception for localization and environment mapping into the surgical robotic control.

3.2 Deformable Tissue Tracker

The goal of the SuPer framework, as shown in Fig. 3.2, is to provide geometric information about the entire surgical scene, including the robotic agent and the deforming environment. A model-based tracker via a particle filter, as described in Chapter 2, is deployed to localize the surgical robotic tool by utilizing a kinematic prior and fusing the encoder readings and endoscopic image data. For the surgical environment, a model-free deformable tracker is employed since the surgical environment is unstructured and constantly deforming. The model-free tracker uses the stereo-endoscopic data as an

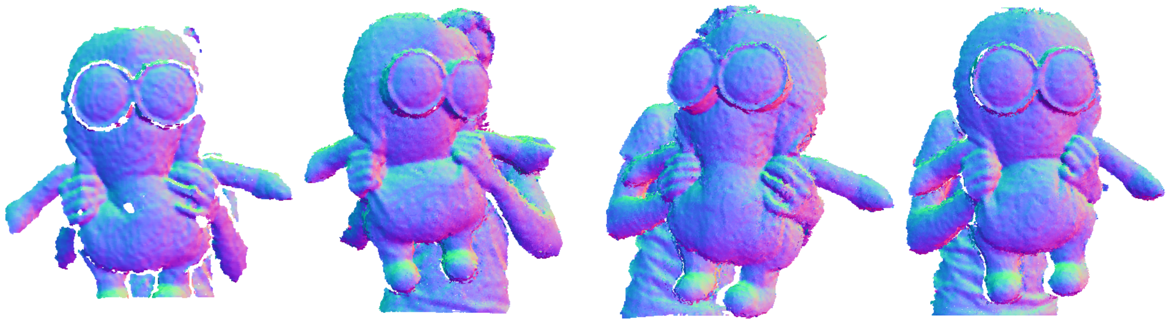


Figure 3.3. Deformable tracking results with testing dataset [59]. The color represents the normal of our surfel data. As the model fuses with more data from left to right, the normal becomes smooth, and the deformations are captured.

observation to reconstruct the deformable scene. To efficiently combine the two separate trackers, a mask of the surgical tool is generated based on the surgical tool tracker and removed from the observation presented to the model-free tracking component. Since the trackers are both perceived in the same camera coordinate frame, a surgical robotic controller can be used in our SuPer framework to manipulate the unstructured surgical scene.

3.2.1 Depth Map from Stereo Images

The depth map from the stereoscopic image is generated using the Library for Efficient Large-Scale Stereo Matching (LIBELAS) [41]. To fully exploit the prior and enhance the robustness of our system, the surgical tool portion of the image and the depth data are not passed to the deformable tissue tracker since the surgical tool is already being tracked. Therefore, a mask of the surgical tool is generated using the same OpenGL rendering pipeline previously developed [137] and applied to the depth and image data that are passed to the deformable tissue tracker. To ensure that the mask covers the entire tool, it is dilated before being applied.

3.2.2 Deformable Surfel Model

To represent the environment, we choose surfel [68] as our data structure due to the direct conversion to point cloud, which is a standard data type for the robotics community. A surfel $\mathcal{S}$ represents a region of an observed surface and is parameterized by the tuple $(\mathbf{p}, \mathbf{n}, \mathbf{c}, \mathbb{r}, \mathbb{c}, \mathbb{t})$, where $\mathbf{p}, \mathbf{n}, \mathbf{c} \in \mathbb{R}^3$ are the expected position, normal, and color, respectively, and the scalars $\mathbb{r}, \mathbb{c}, \mathbb{t}$ are the radius, confidence score, and time stamp of last update, respectively. Alongside the geometric structure that the surfel data provides, it also generates the confidence and timestamp of the last update, both of which can be exploited to further optimize a controller working in the tracked environment. For adding/deleting and fusing surfels, refer to work done by Keller et al. [68] and Gao et al. [37].

The number of surfels grows proportionally to the number of image pixels provided to the deformable tracker, which makes it infeasible to track the entire surfel set individually. Inspired by the work about Embedded Deform (ED) [155], we process our surfel set with a less-dense ED graph, $\mathcal{G}_{ED} = \{\mathcal{V}, \mathcal{E}, \mathcal{P}\}$, where $\mathcal{V}$ is the vertex index set, $\mathcal{E}$ is the edge set, and $\mathcal{P}$ is the parameters set. With a uniform sampling from the surfel, the number of ED nodes, N_{ED}, is much lower than the number of surfels, N_{surfel}. Thus, the ED graph has significantly fewer parameters to track compared with the entire surfel model. Moreover, the ED graph can be thought of as an embedded sub-graph representing the skeletonization of the surfels to capture their deformations.

An ED node consists of a parameter tuple $(\mathbf{g}_i, \mathbf{q}_i, \mathbf{b}_i) \in \mathcal{P}$, where $\mathbf{g}_i \in \mathbb{R}^3$ is the position of the ED node, and $\mathbf{q}_i \in SO(3)$ and $\mathbf{b}_i \in \mathbb{R}^3$ are the quaternion and translation parameters, respectively, that are converted to a homogeneous transform matrix with $\mathbf{T}(\mathbf{q}_i, \mathbf{b}_i) \in SE(3)$. The transformation of every surfel is modeled as follows

$$T(\overline{\mathbf{p}}) = \mathbf{T}_g \sum_{i \in \mathrm{KNN}(\overline{\mathbf{p}})} \alpha_i [\mathbf{T}(\mathbf{q}_i, \mathbf{b}_i)(\overline{\mathbf{p}} - \vec{\mathbf{g}}_i) + \vec{\mathbf{g}}_i] \tag{3.1}$$

where $\mathbf{T}_g \in SE(3)$ is the global homogeneous transformation (e.g., common motion shared with all surfel), α_i is a normalized weight, and $\text{KNN}(\overline{\mathbf{p}})$ is an index set that contains k-nearest neighbors of $\overline{\mathbf{p}}$ in $\mathcal{G}_{ED}$. Note that $\vec{\cdot}$ is a vector in homogeneous representation (e.g.,$\vec{\mathbf{g}} = [\mathbf{g}, 0]^T$). Both α_i and $\text{KNN}(\overline{\mathbf{p}})$ are generated using the same method proposed by Sumner et al. [155]. The normal transformation is similarly defined as

$$T_n(\mathbf{n}) = \mathbf{T}_g \sum_{i \in \text{KNN}(\overline{\mathbf{p}})} \alpha_i [\mathbf{T}(\mathbf{q}_i, \mathbf{0})\vec{\mathbf{n}}] \tag{3.2}$$

When implementing the ED graph, the $\mathbf{q}_i$ and $\mathbf{b}_i$ for node i are the current frames for estimating the deformation. After every frame, the deformations are committed to $\mathbf{g}_i$ and the surfels based on (3.1) and (3.2). Therefore, with an ED graph of n nodes, the whole surfel model can be estimated with $7 \times (n + 1)$ parameters. Note that the extra 7 parameters come from $\mathbf{T}_g$, which is also estimated with a quaternion and translational vector. An example of using this model to track deformations is shown in Fig. 3.3.

3.2.3 Cost Function

To track the visual scene with the parameterized surfel model, a cost function is defined to represent the distance between an observation and the estimated model. It is defined as follows

$$E = E_{data} + \lambda_a E_{Arap} + \lambda_r E_{Rot} + \lambda_c E_{Corr} \tag{3.3}$$

where E_{data} is the error between the depth observation and the estimated model, E_{ARAP} is a rigidness cost ensuring that ED nodes nearby one another have a similar deformation, E_{Rot} is a normalization term for the quaternions to satisfy a rotation in $SO(3)$ space, and E_{Corr} is a visual feature corresponding to the cost to ensure texture consistency. More specifically, the traditional point-plane error metric [111] is used for the depth data cost.

When minimized, the model is aligned with the observed depth image. The expression is

$$E_{data} = \sum_i (T_n(\vec{\mathbf{n}}_i)^T (T(\overline{\mathbf{p}}_i) - \overline{\mathbf{o}}_i))^2 \tag{3.4}$$

where $\mathbf{o}_i = D(u,v)\mathbf{K}^{-1}[u,v,1]^T$ is the observed position from the depth map with D at pixel coordinate (u,v), and $\mathbf{p}_i$ and $\mathbf{n}_i$ are the associated surfel position and normal from the most up-to-date model. This cost term, however, is highly curved and not easy to solve. To simplify the optimization, the normal is fixed at every iteration during optimization. This results in the following expression at iteration j

$$E_{data}^{(j)} = \sum_i (\hat{\mathbf{n}}_i^{(i)T} (T(\overline{\mathbf{p}}_i; O^{(j)}) - \overline{\mathbf{o}}_i))^2 \tag{3.5}$$

where $\hat{\mathbf{n}}_i^{(j)} = T_n(\vec{\mathbf{n}}_{i-1}; O^{(j-1)})$ and $O^{(j)}$ is the set of ED nodes at iteration j. This is a normal-difference cost term similar to the Iterative Closest Point [111].

The rigid term is constructed by l_2 norm of the difference between the positions of an ED node transformed by two nearby transformations. The cost expression is

$$E_{ARAP} = \sum_i \sum_{k \in \mathbf{e}_i} ||T(\mathbf{q}_k, \mathbf{b}_k)(\overline{\mathbf{g}}_i - \vec{\mathbf{g}}_k) + \vec{\mathbf{g}}_k - \vec{\mathbf{g}}_i - \vec{\mathbf{b}}_i||^2 \tag{3.6}$$

where $\mathbf{e}_i \in \mathcal{E}$ is the edge set of ED nodes neighboring node i. The edge set $\mathcal{E}$ is generated by the k-nearest neighbor algorithm based on the ED node positions. This cost term forces the model to have consistent motion among the nearby ED nodes. Intuitively, it provides feedback to the model when a portion of the ED nodes does not receive enough data from the observation in the current frame.

To have a rigid-like transformation, the normalizing term in the cost function is set to

$$E_{Rot} = \sum_k ||1 - \mathbf{q}_k^T \mathbf{q}_k||^2 \tag{3.7}$$

since quaternions hold $||\mathbf{q}||^2 = 1$. Both E_{Rot} and E_{ASAP} are critical to ensuring that all ED nodes move as rigid as possible. This specification is needed since $7 \times n$ is a very large space that needs to be optimized relative to the observed data. For example, in cases of obstruction, the optimization problem is ill-defined without these terms.

The final cost term is required to ensure visual feature correspondence, which forces visual texture consistency between the model and the observed data. The expression for the cost is

$$E_{Corr} = \sum_{(\mathbf{m},\mathbf{c}) \in Feat} ||T(\overline{\mathbf{p}}_{\mathbf{m}}) - \overline{\mathbf{o}}_{\mathbf{c}}||^2 \tag{3.8}$$

where $Feat$ is a set of associated pairs of matched feature points $\mathbf{m}, \mathbf{c} \in \mathbb{R}^2$ between the rendered color image of our model and the observed color image data, respectively. The observed point is obtained using the same expression as before: $\mathbf{o}_c = D(\mathbf{c})\mathbf{K}^{-1}\overline{\mathbf{c}}$. The feature matching gives a sparse but strong hint for the model to fit the current data.

To solve the non-linear least square problem proposed in (3.3), the Levenberg Marquardt (LM) algorithm is implemented to efficiently obtain the solution for the model. The LM algorithm requires the cost function to be in the form of a sum of squared residuals. Therefore, all the parameters from O are stacked into a vector, $\mathbf{x}$, and all cost terms are reorganized into vector form such that $||\mathbf{f}(\mathbf{x})||^2 = \mathbf{f}(\mathbf{x})^T\mathbf{f}(\mathbf{x}) = E$. In this form, the function is linearized with a Taylor expansion

$$\delta = \arg\min_{\delta} ||\mathbf{f}(\mathbf{x}) + \mathbf{J}\delta||^2 \tag{3.9}$$

where $\mathbf{J}$ is the Jacobian matrix of $\mathbf{f}(\mathbf{x})$. Following the LM algorithm, the function is solved for δ by using

$$(\mathbf{J}^T\mathbf{J} + \mu\mathbf{I})\delta = \mathbf{J}^T\mathbf{f}(\mathbf{x}) \tag{3.10}$$

where μ is a damping factor. The LM algorithm accepts the δ by setting $\mathbf{x} \leftarrow \mathbf{x} + \delta$ when the cost function decreases: $||\mathbf{f}(\mathbf{x})||^2 > ||\mathbf{f}(\mathbf{x} + \delta)||^2$. Otherwise, it increases the damping factor. Intuitively, the LM algorithm aims at finding a balance between the Gaussian-Newton method and the gradient descent solver.

3.2.4 Experiments

Implementation Details

To measure the effectiveness of the proposed framework, our implementation was deployed on a da Vinci Surgical® System. The stereo camera is the standard 1080p laparoscopic camera running at 30fps. The Open Source dVRK [65] is used to send end-effector commands and get joint angles and the end-effector location in the base frame of a single surgical robotic arm with a gripper, also known as Patient Side Manipulator (PSM). The data submission for the PSM occurs at a rate of 100Hz. The communication between subsystems of the code relies on the Robot Operating System (ROS), and everything is processed on two identical computers with an Intel® Core™ i9-7940X Processor and NVIDIA's GeForce RTX 2080.

The surgical tool tracking is set to the same implementation as presented in Chapter 2. The endoscopic image data is resized to 640 by 480 before processing to generate the depth map from the stereo images. The LIBELAS parameters are applied using the default settings from their open-sourced repository [41]. After computing this dataset, D, the resulting depth map is masked by the rendered surgical tool. The mask is dilated by 9 pixels before being applied. The depth map is then smoothed spatially with a bilateral filter and temporally with a median filter of four frames to decrease the noise. The surfel radius is set to $\mathbb{r} = \sqrt{2}D(u, v)/(f|n_z|)$, and the confidence score is calculated with $\mathbb{c} = exp(-d_c^2/0.72)$ at pixel coordinate (u, v), where n_z is the z component of camera frame normal, f is the cameras focal length, and d_c is the normalized distance from the pixel coordinate to the center of the image [111][68]. Whenever new surfels are added to the model, ED nodes are randomly sampled from them [37]. This process typically results in 300 ED nodes and, therefore, roughly 2K parameters to estimate. Very similar surfels, both temporally and spatially, are merged to each other when we fuse the observed map to the model to keep the model concise as described in [37]. OpenCV's implementation of SURF is used for feature extraction and matching in the cost functions visual correspondence term. To solve the linear system for the LM algorithm in (3.10), we implemented a GPU version of the preconditioned conjugate gradient and set the max iteration number to 10. For the cost function, the parameters $[\lambda_a, \lambda_r, \lambda_c]$ are set to $[10, 100, 10]$.

Datasets

Repeated Tissue Manipulation: To test the effectiveness of the proposed framework, a simple controller was implemented to grasp and tug on the tissue at the same tracked point repeatedly. At the beginning of the experiment, a small cluster of surfels was selected on the tissue in the deformable tracker, and their resulting averaged position, $\mathbf{p}^c_g$, and normal, $\mathbf{n}^c_g$, represented the tracked point to be grasped. The following steps were then repeated five times or until failure on the PSM gripper.

1. Align above surface: move to $\mathbf{p}^c_g + d\mathbf{n}^c_g$ where $d = 2$ cm and the orientation $\mathbf{q}^c_g$ is adjusted that the opening of the gripper is pointed towards the surface normal, $\mathbf{n}^c_g$

2. Move to the tissue: stop updating $\mathbf{p}^c_g$ and $\mathbf{n}^c_g$ from the deformable tracker and move to $\mathbf{p}^c_g + d\mathbf{n}^c_g$ where $d = 0.5$ cm and orientation $\mathbf{q}^c_g$

3. Grasp and stretch the tissue: close the gripper to grasp the tissue and move to $\mathbf{p}^c_g + d\mathbf{n}^c_g$ where $d = 2$ cm and orientation $\mathbf{q}^c_g$

4. Place back the tissue: move to $\mathbf{p}^c_g + d\mathbf{n}^c_g$ where $d = 0.5$ cm and orientation $\mathbf{q}^c_g$ and open the gripper

5. Continue updating $\mathbf{p}^c_g$ and $\mathbf{n}^c_g$ from the deformable tracker.

It should be noted that the end-effector on the PSM gripper is defined on the link preceding the jaws from the gripper, which are approximately 1 cm long.

To move the PSM to the target end-effector position, $\mathbf{p}_g^c + d\mathbf{n}_g^c$, and orientation, $\mathbf{q}_g^c$, the trajectories were generated using linear and spherical linear interpolation, respectively. The trajectories were re-generated after every update to $\mathbf{p}_g^c$ and $\mathbf{n}_g^c$ from the deformable tracker and generated in the camera frame from the current end-effector pose. The current end-effector pose was calculated by transforming the PSM end-effector pose from dVRK with the Lumped Error transform from the surgical tool tracker. Finally, to follow the trajectory, the end-effector poses were transformed back to the base frame of the PSM using the surgical tool tracker and set via dVRK.

This experiment was repeated using the following configurations:

- The complete proposed framework.
- The framework without deformable tracking, i.e., just static reconstruction, by setting the number of ED nodes to 0.
- The framework without surgical tool masking.
- The framework without surgical tool tracking, and, instead, relying on calibrated hand-eye.

The tissue used was the skin of a chicken leg.

Reprojection Error for Tracking Accuracy: To evaluate our proposed approach quantitatively, we manually annotated 20 points on the tissue through time on the raw image data from the repeated tissue manipulation experimentation. The 20 points were chosen from the highest confidence points of SURF in the first frame. This time series of 2D image positions was compared against the reprojection from the deformable tissue tracker. We also evaluated the result of an off-the-shelf SURF approach from OpenCV which matched the keypoints in every frame with the description in the first frame.

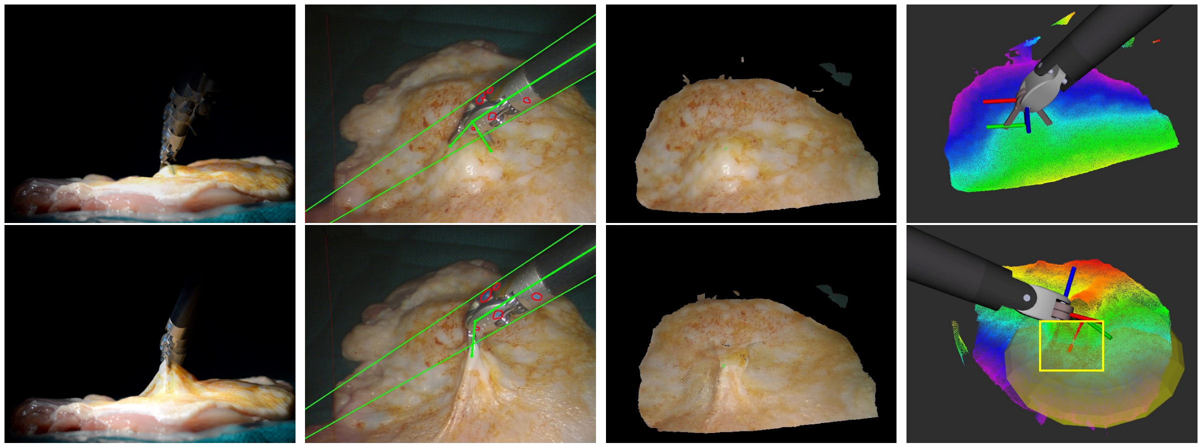


Figure 3.4. Tissue manipulation with the SuPer framework was implemented on the da Vinci® Surgical System in real time. Images (from left to right): the real scene, tool tracking, deformable reconstruction, and the entire SuPer Framework in RViz.

Moreover, the surgical tool accuracy was evaluated by comparing 50 manually segmented images that were selected at random from the repeated tissue experiment and compared against the reprojected/rendering of the surgical tool tracking. The experiment was conducted with different numbers of particles to highlight the trade-off between the accuracy of modeling the posterior probability and computational cost in real-time tracking.

Results

The separate components of the framework ran at 30fps, 30fps, 8fps, and 3fps for the surgical tool tracking, surgical tool rendering, depth map generation, and deformable tissue tracker, respectively. An example of the procedure used for the repeated tissue manipulation experiment is shown in Fig. 3.1. When using the complete framework, the PSM arm successfully grasped the same location of the tissue all five times after repeated deformations. As indicated by the yellow rectangle in Fig. 3.4, the deformable tracker even managed to capture the structure of the tissue that was not visible to the endoscopic camera during the stretching.

As Fig. 3.5a illustrates, without our SuPer framework, the direct input was very

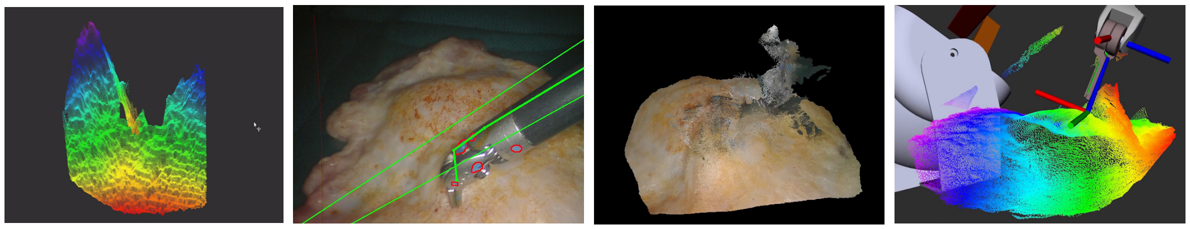

(a) Raw depth map **(b)** Without tool tracking **(c)** Without mask **(d)** without deformable tracking

Figure 3.5. Results from the repeated tissue manipulation experiment without using the complete proposed SuPer framework. None of these results are ideal since they do not properly capture the real surgical scene through failed robotic localization or improper environmental mapping.

Table 3.1. Reprojection error of surgical tool for tracking accuracy

Num. of Particles	Mean IoU	Perc. above 80%	Fps
100	80.8%	68%	30
500	82.4%	71%	30
1000	81.7%	73%	26
5000	82.8%	77%	8

noisy and could not provide consistent geometrical information about the scene. When not using the deformable tracker, the computer crashed due to memory overflow after three grasps, and the reconstruction was not at all representative of the real environment. Without the mask, the reconstructed scene in the deformable tracker did not converge properly and failed after three grasps. Finally, when not using the surgical tool tracking function, no successful attempt could be made because the grasper missed the tissue. All three failure cases are presented in Fig. 3.5.

A comparison between the reprojected rendering from the surgical tool and the manual segmentation results is presented in Table. 3.1. The results demonstrate that more particles generally result in better performance. However, to ensure efficiency, we set the particle number to 500 to keep the method in real time. The comparison provided in Fig. 3.6 demonstrates that our SuPer framework is much more stable than the SURF feature matching because our method is designed to reconstruct the dynamic scene

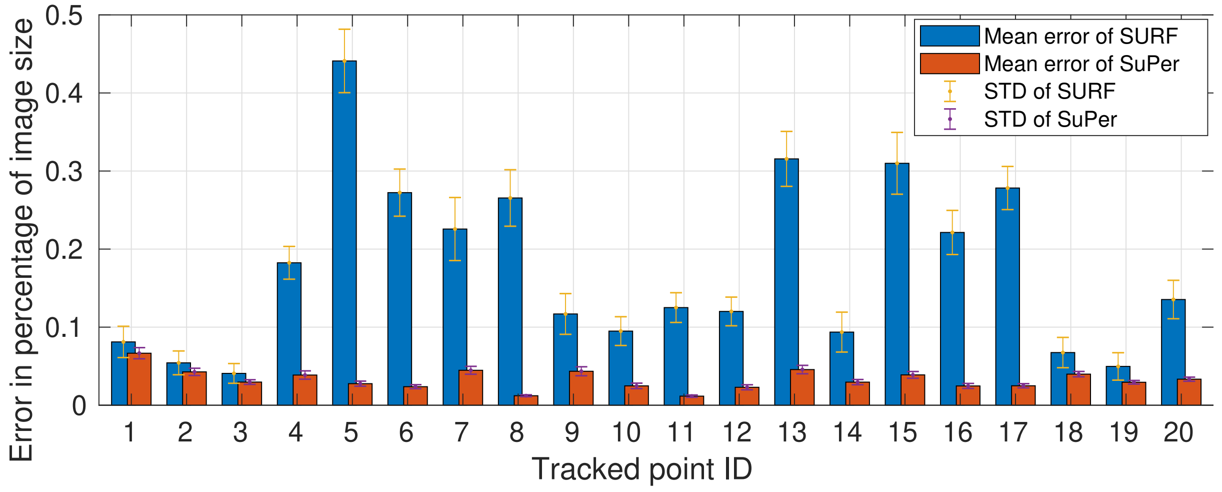


Figure 3.6. The reprojection error comparison of 20 labeled points in our dataset between our SuPer and the native SURF keypoint tracking.

entirely, whereas SURF only finds the local minimal matching position. Moreover, our method has a higher accuracy because our error is smaller than the error derived from SURF, even with the tracked point features No.2 and No.3 that had the best performance of SURF, as shown in Fig. 3.6.

3.2.5 Discussion

The ability to continuously and accurately track the tissue during manipulation enables control algorithms to be successful in the unstructured environment. Currently, we believe that the first limiting factor of our frameworks is the noise from the depth map reconstructed by the stereo-endoscopic camera as shown in Fig. 3.5a. A second experimental limitation is related to the features used to update the surgical tool tracker. The markers were manually painted and inherently inaccurate in terms of position, $\mathbf{p}^{j_k}$ from (2.29). We believe this is the main cause of the inconsistencies in the surgical tool tracking, and other methods, such as as [100] would be viable to use in place of the color tracking. Improving these components would be simple as other strategies for more recent and effective depth reconstruction and instrument feature tracking could be substituted at no additional effort. Furthermore, the certainty of the perception could be used for optimal control algorithms, endoscopic camera control to maximize certainty, and other advanced control techniques. Handling blood and topological changes, such as cutting, are the next big challenges to overcome to make our proposed framework even more suitable for real clinical scenarios.

In conclusion, we propose a surgical perception framework, SuPer, to localize the surgical tool and track the deformable tissue. SuPer was experimentally evaluated on a da Vinci® System to show its ability to track during ongoing manipulation tasks where instrument occlusions, significant tissue deformations, and tissue tracking had to be addressed. In addition, a deformable tissue tracking dataset was released for further community research.

3.3 SuPer Deep

As highlighted in Section 3.2.5, there are two major challenges to providing perception for surgical automation efforts: tracking of the surgical tool to control and localize it in the camera frame, and tracking of the deformable environment for the surgical tool to plan and interact with. While these two problems have been solved outside of surgical robotics [82, 37], the domain-specific challenges are the narrow field of view provided by endoscopes, poor lighting conditions, and the requirement of very high accuracy [8].

The surgical tool tracking community has largely focused on developing feature detection algorithms to update the pose of the surgical tool [8]. The algorithms need to be robust to the poor lighting conditions and the highly reflective tool surfaces. Examples of recent work include using the Canny-edge detector for silhouette extraction [50], online template matching [169], and categorized features using classical image features, such as the spatial derivatives [131], [132]. Deep neural networks have also achieved promising results in feature tracking for surgical tools [71], [19], but utilizing them for full 3D pose estimation still remains unexamined.

Simultaneously, efforts in tissue tracking have focused mainly on adaptions of 3D reconstruction techniques, such as SurfelWarp for deformable tracking [37]. The lack of directly measurable depth information in endoscopes is a significant challenge in the adaptation. Hence, the common approach is to work with stereoscopic endoscopes and use stereo reconstruction techniques, such as Efficient Large-Scale Stereo Matching (ELAS), to generate depth images [41]. This depth estimation allows the application of deformable tracking techniques [151, 152]. Other tissue tracking techniques include tracking keypoint features and registration [172],as well as dense SLAM methods, which use image features to localize the endoscope [96, 98].

A common theme across these two challenges is the need for high-quality image features. Surgical tool tracking mainly focuses on developing detectors for tool features, and recent studies in tissue tracking have highlighted depth reconstruction from stereo matching as the most significant bottleneck [151]. Deep learning has the advantage of learning features, which will eliminate the need for feature engineering. However, deep learning previously has not been a front runner in surgical perception due to the lack of large quantities of high-quality medical and surgical data [63].

We use state-of-the-art deep neural networks (DNNs) that require minimal training data to explore its application in surgical perception. Our contributions can be summarized as follows:

1. Using deep learning for high quality and robust surgical tool feature extraction,
2. Investigative study of popular deep learning and traditional stereo matching algorithms to improve deformable tissue tracking,
3. Complete integration of deep neural networks into the Surgical Perception (SuPer) framework to fully perceive the entire surgical scene - *SuPer Deep.*

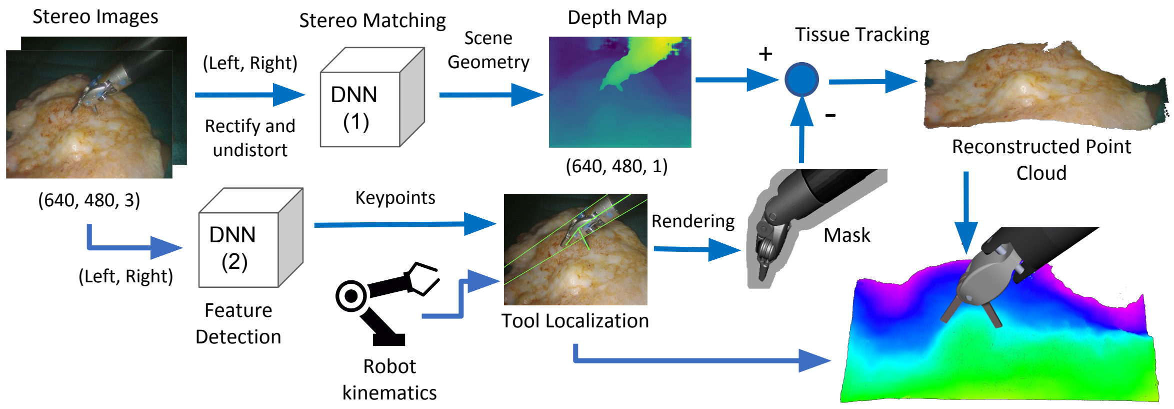


Figure 3.7. The complete workflow of the proposed SuPer Deep framework, which was built on our original framework with DNNs. The DNNs were used to match the features from stereo images to generate the depth map and detect point features for estimating the pose of the surgical tool.

Experiments were ran using a tissue manipulation dataset collected in Section 3.2.4, along with two other publicly available datasets collected from the da Vinci® Surgical System, to evaluate the tool tracking and tissue tracking performances individually. SuPer Deep framework advances on SuPer, its predecessor, by eliminating the requirement for painted markers and the keypoint association process in tool tracking. Moreover, our framework generates more realistic tissue reconstruction by improving the depth estimation. Finally, our work includes the first comparative study on deformable tissue tracking, whereas previous studies have been hampered by a paucity of standard benchmarking datasets for surgical perception.

Our surgical perception framework, as shown in Fig. 3.7, combines a deformable tissue tracker and a surgical tool tracker to perceive the entire surgical scene, including the the deforming environment and the robotic agent. Two deep neural networks are embedded into our framework for specific feature extractions: DNN(1) finds and matches features from stereo images to generate a depth map for tissue tracking, and DNN(2) extracts point features for surgical tool tracking. Meanwhile, the other components (e.g., tissue tracking) remain unchanged as in the original SuPer presented in Section 3.2.

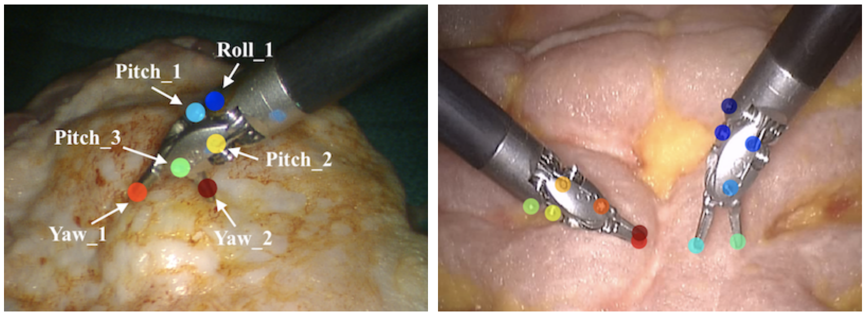


Figure 3.8. Illustration of the point features to detect on surgical tools. The left figure shows the features detected on a single instrument, and the right figure shows the features detected and associated appropriately with two instruments.

3.3.1 Keypoint Detection for Surgical Tool Tracking

To localize the surgical tools on image frames, we use DeepLabCut [100], which employs DeeperCut [60] as the backbone for point feature detection. The DNN consists of variations of Deep Residual Neural Networks (ResNet) [52] for feature extraction and deconvolutional layers to up-sample the feature maps and produce spatial probability densities. The output estimation for each point feature is represented as a tuple $(\mathbf{m}^i, \eta^i)$, where $\mathbf{m}^i \in \mathbb{R}^2$ is the image coordinate of the i-th feature and $\eta^i \in \mathbb{R}$ is the corresponding confidence score. The DNN was fine-tuned with few training samples to adapt to surgical tool tracking by minimizing the cross-entropy loss. The samples were hand-labeled using the open-source DLC toolbox [109]. Fig. 3.8 shows examples of point features that were detected on surgical instruments.

To estimate the pose of the surgical tool in 3D space, the 2D detections are combined with the encoder readings from the surgical robot, and a particle filter is applied for estimation as detailed in Chapter 2. To integrate this with the Lumped Error tracking, the point feature observation model in (2.33) was modified to

$$P(\mathbf{m}_t|\hat{\mathbf{w}}_t, \hat{\mathbf{b}}_t, \hat{\mathbf{e}}_t) \propto \sum_{i\in A_m} \eta_t^i e^{-\gamma_m ||\mathbf{m}_t^i - \hat{\mathbf{m}}_i(\hat{\mathbf{w}}_t, \hat{\mathbf{b}}_t, \hat{\mathbf{e}}_t)||} \tag{3.11}$$

which removes the association step in line 18 from Algorithm 1. It is important to include the DNN's confidence, η_t^k, in the model because sometimes the detections can be poor and the corresponding update needs to be weighted lower.

3.3.2 Depth Estimation for Deformable Tissue Tracking

Deformable tissue tracking relies heavily on the quality of depth estimation, as the deformable tracker uses the depth maps as the observation [151]. To estimate the depth, a stereo matching algorithm is used to compute the disparity, and then inverted to obtain pixel-wise depth. Traditional stereo matching algorithms, like [41] and [53], typically take a pair of rectified stereo images I_l and I_r as input, and estimate the disparity by matching image patches or features between I_l and I_r. Due to the complexity of the surgical setting, the image quality is not the sharpest, which makes finding pixel-level correspondence extremely challenging. Deep-learning-based algorithms, such as [69], [12], and [175], use a weight-sharing feature extractor to obtain feature maps F_l and F_r from I_l and I_r, respectively. Then a 4D matching cost volume C_d is formed by concatenating the F_l and F_r, such that $C_d(i, j, d, :)$ is the concatenation of $F_l(i, j)$ and $F_r(i, j + d)$, where (i, j) is the pixel location and d is the disparity. The cost volume is regularized using 3D convolutional layers and reducing the dimension of the 4-th channel to 1. Finally, the resulting 3D tensor S_d is used to estimate the disparity for each pixel as

$$\hat{d}(i, j) = \sum_{d=0}^{D_{max}} \sigma(-S_d(i, j, d))d \tag{3.12}$$

where D_{max} is the max disparity and $\sigma(\cdot)$ denotes the softmax function. An investigation between these stereo matching algorithms is presented in the upcoming Section 3.3.3.

After estimating the disparity $\hat{d}$, the depth value z is obtained by the following transform

$$z(i, j) = \frac{bf}{\hat{d}(i, j)} \tag{3.13}$$

where b is the horizontal offset (i.e., baseline) between the two cameras, and f is the focal length, which can be obtained from camera calibration.

Various stereo-matching algorithms can be substituted in for DNN(1) of our framework shown in Fig. 1. To find the best one, we investigated several stereo matching algorithms by combining with the deformable tissue tracker presented in 3.2. The state-of-the-art algorithm, Guided Aggregation Network (GA-Net) [175], was finally chosen for our framework. In comparison to previous works on deformable tissue tracking, which require a substantial amount of spatial and temporal filtering on the depth image [151], our method employs a deep stereo matching network for accurate and dense depth estimation, which requires no post-processing on the depth image.

3.3.3 Experimentation

Implementation Details

We evaluated the proposed framework on three open-source datasets for multiple tasks addressing the performance of the surgical tool tracking and deformable tissue tracking. The experiments were conducted on two identical computers, each containing an Intel® Core™ i9-7940X Processor and NVIDIA's GeForce RTX 2080. The weights of DeepLabCut to detect the keypoints were pre-trained on ImageNet and fine-tuned by training on only 50 hand-labeled images for 7100 iterations with stochastic gradient descent of batch size 1 and learning rate $lr = 0.005$. The surgical tool tracking is set to the same implementation as Chapter 2 and $\gamma_m = 0.1$. For depth map estimation, the raw stereo images were rectified, undistorted, and resized to (640, 480) before being passed into the stereo matching algorithm. Due to the lack of task-specific datasets for surgical environments, the pretrained weights of GA-Net were utilized, which was trained on the Scene Flow dataset from scratch for 10 epochs and fine-tuned on the KITTI2015 dataset for 640 epochs. After inverting the disparity, the resulting depth map was fused into the tissue model after subtracting the rendered tool mask, which is dilated by 5 pixels.

Datasets

SuPer: The dataset consists of a raw stereo endoscopic video stream and encoder readings from the surgical robot with ground-truth labels for the tool tracking tasks, which consists of 50 hand-labeled surgical tool masks. The tool tracking performance was evaluated by calculating the Intersection-Over-Union (IoU, Jaccard Index) for the rendered tool masks, which are based on estimated tool poses.

Hamlyn Centre Video Dataset: [154] was used to evaluate the performance of deformable tissue tracking. It includes two video sequences of silicone heart phantom deforming with cardiac motion and consists of ex-vivo endoscopic stereo videos (resolution: 360×288) with depth information generated from CT scans. The re-projected depth maps of the reconstructed tissue model are evaluated, which is the projection of the entire reconstructed point cloud to the image plane, with each pixel containing a depth value. We calculated the per-pixel root-mean-square (RMS) error of the depth map for every image,

$$\sqrt{\frac{1}{N_p}\sum_{i,j}(\hat{d}_{i,j} - d_{i,j})^2} \tag{3.14}$$

where i, j is the pixel position, $\hat{d}$ is the estimated depth value, d is the ground truth depth value, and N_p is the total number of pixels for each image. We also reported the percentage of the valid (non-zero) pixels of the depth map.

da Vinci Tool Tracking: [169] consists of a stereo video stream and the corresponding kinematic information of the da Vinci® surgical robot. The dataset is used to evaluate surgical tool feature detection and pose estimation. Note that the SuPer dataset has painted markers, and hence this additional experiment ensures that surgical tool feature detector learns surgical tool point features and is not dependent on colored markers. The performance of feature detection is evaluated by calculating the L_2 norm of the error in

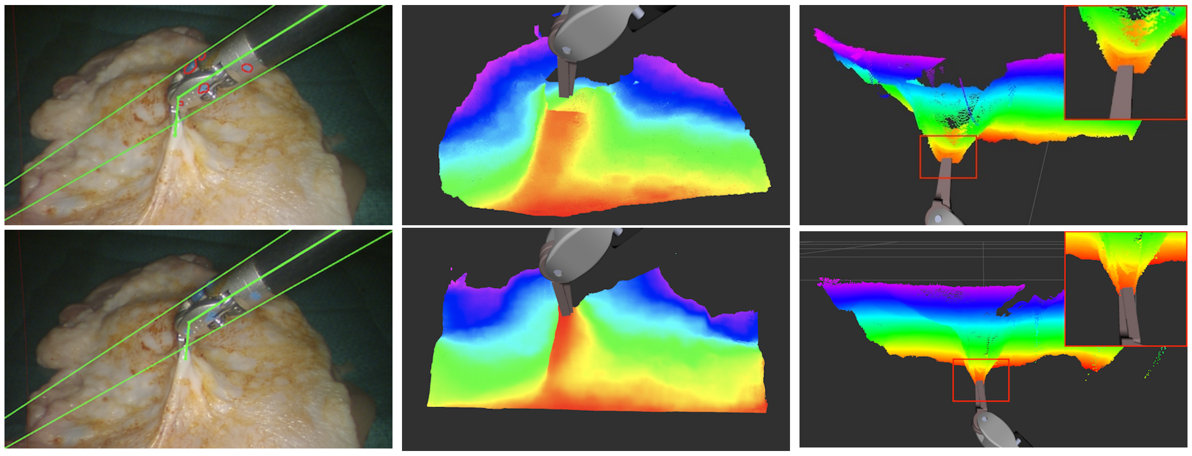


Figure 3.9. Qualitative results from real-time environmental mapping on the SuPer dataset. Top and bottom row figures are results from the original SuPer framework and the proposed SuPer Deep framework, respectively.

pixels which is explicitly computed as follows

$$\frac{1}{N}\sum_{n=1}^{N}||\mathbf{m}_n^i - \mathbf{m}_n^{i*}||_2 \tag{3.15}$$

for the i-th feature point where N is the total number of test images and $\mathbf{m}_n^{i*}$ is the ground truth feature point location in the n-th image. We experiment with varying amounts of hand-labeled training data to illustrate the data efficiency of the proposed surgical tool feature detection method. Due to lack of ground-truth data for pose estimation, we only provide qualitative results for surgical robotic tool tracking on this dataset.

Results

Qualitative results of the environment mapping on the SuPer dataset are presented in Fig. 3.9. As highlighted in the figures, SuPer Deep provides a larger field of view of the unstructured environment while preserving better details on the reconstruction. In comparison to the SuPer framework, more detailed information is captured due to the lack of filtering and smoothing applied on the stereo reconstruction process, which was required in previous implementations of tissue tracking in Section 3.2.

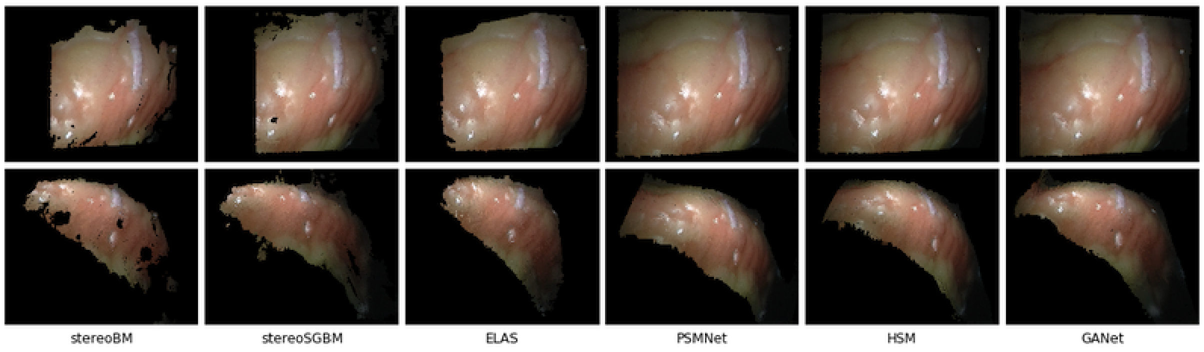


Figure 3.10. A comparison of tissue reconstruction by fusing the depth maps from different stereo matching algorithms. The first row shows the front view and the second row shows the side view of the reconstruction.

Table 3.2. Comparative study for deformable tissue tracking with varying stereo-reconstruction algorithms done on the Hamlyn Cantre heart phantom dataset.

Method	Video 1		Video 2	
	RMSE	Perc. valid	RMSE	Perc. valid
stereoBM + deformable tissue tracker	23.24 ± 2.18	0.565 ± 0.037	34.02 ± 2.08	0.523 ± 0.033
stereoSGBM + deformable tissue tracker	16.84 ± 1.99	0.713 ± 0.038	24.68 ± 1.84	0.683 ± 0.032
ELAS + deformable tissue tracker	16.12 ± 2.22	0.716 ± 0.044	22.03 ± 2.89	0.719 ± 0.049
PSMNet + deformable tissue tracker	5.64 ± 1.48	0.940 ± 0.023	8.23 ± 1.32	0.939 ± 0.014
HSM + deformable tissue tracker	5.33 ± 1.36	0.938 ± 0.023	6.73 ± 1.28	0.946 ± 0.020
GA-Net + deformable tissue tracker	**4.87 ± 1.55**	**0.947 ± 0.026**	**6.20 ± 1.57**	**0.957 ± 0.024**

Using the Hamyln Centre Video Dataset, the deformable tissue tracking results were compared by combining popular stereo matching algorithms with the deformable tissue tracker. We visualize the reconstruction results by fusing the first 10 estimated depth maps from each algorithm in Fig. 3.10. Deep-learning-based algorithms generally provide dense and consistent matches and result in realistic tissue reconstructions. We calculated the average per-pixel RMS error on the re-projected depth maps, and the quantitative results are shown in Table 3.2. Deep-learning-based approaches achieve much lower per-pixel RMS error with the higher percentage of the valid pixel, which confirms the observations from the environment mapping results in Fig. 3.9 and Fig. 3.10.

Fig. 3.11 shows the feature detection performance of the DeepLabCut with varying numbers of training samples. By leveraging transfer learning, the feature detector is able to achieve high performance on detecting surgical tool features using few training samples.

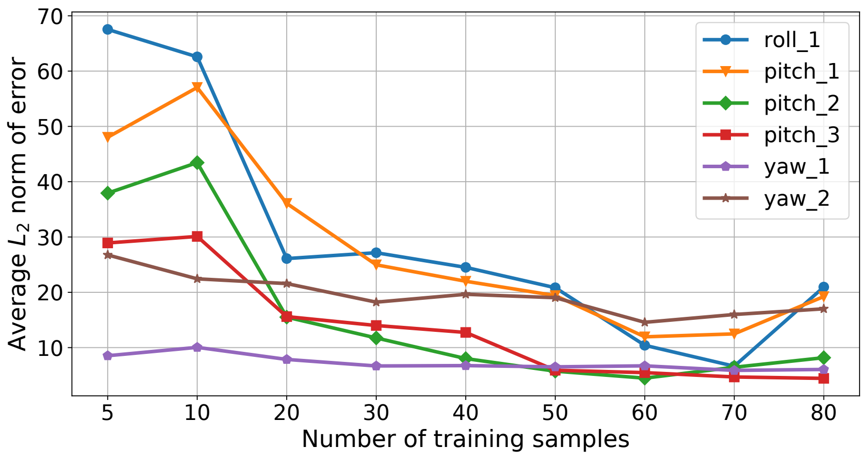


Figure 3.11. Plot of average detection error of features on surgical tool against the amount of training data. As is evident by the plot, only 60 to 80 training samples are required to train an accurate model.

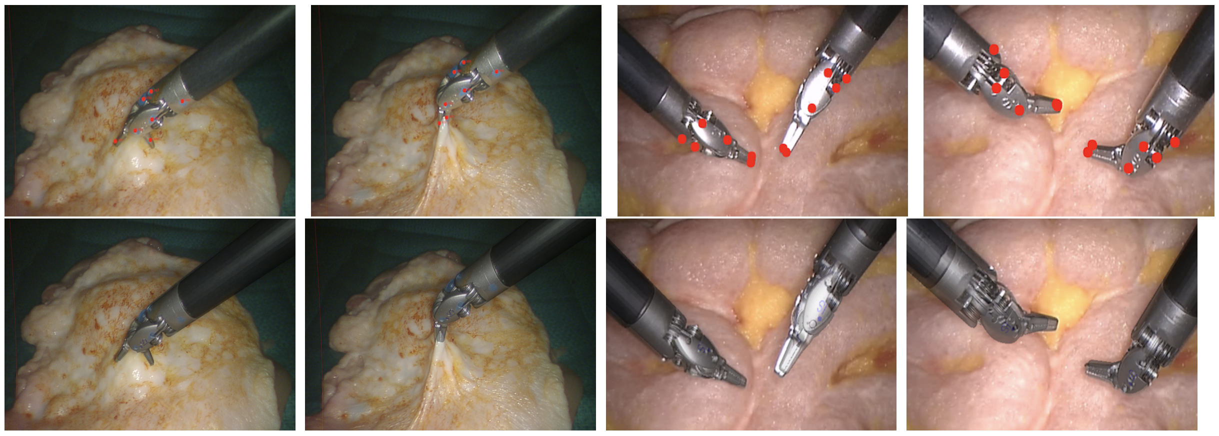


Figure 3.12. Qualitative results of surgical tool tracking. The top row shows the DeepLabCut prediction overlaid on the real images. The bottom row shows an Augmented Reality rendering of the surgical tool [137] on top of the real images. The renderings are best viewed in color.

For the tool tracking task, SuPer Deep achieved **91.0**% mean IoU on the SuPer tool segmentation task, which is a significant improvement on the original method (SuPer: 82.8%). Notably, SuPer Deep does marker-less tool tracking while the former utilizes painted markers. Qualitative results of the tool tracking are presented in Fig. 3.12, where we experimented with our tool tracker on both the SuPer dataset and the da Vinci tool tracking dataset. In the visualization, the Augmented Reality rendering from the

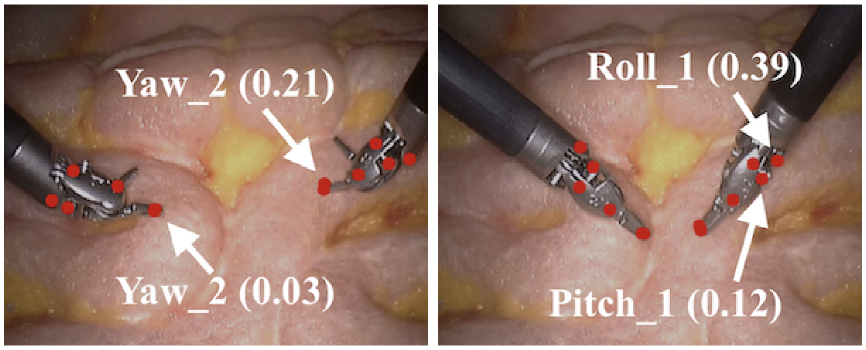


Figure 3.13. Failure cases in tool point feature detection. In parenthesis are the corresponding confidence scores. Note that when a failure occurs the DNN outputs low confidence hence making the missed detection not as detrimental to our tracking algorithm.

estimated tool pose produces a near-perfect overlap with the tool on real images.

3.3.4 Discussion

The experimental results show SuPer Deep achieves excellent performances in both surgical tool tracking and deformable tissue tracking. By utilizing deep neural networks, SuPer Deep produces more consistent depth maps and achieves accurate tool pose estimation. The latter also helps to reduce the dilation of the tool mask, which reduces the amount of information lost. As the visualizations shows in Fig. 3.9, SuPer Deep's reconstruction has the surgical tool *touching* the point cloud (as opposed to just being above the point cloud).

There are occasional failures in feature detection on the surgical tool, owing mainly to the symmetry of the tool parts, for example, the *Roll_1*, *Pitch_1* and *Yaw_2* features. As shown in Fig. 3.11, detecting those features is more challenging compared to other ones. The missed detections are, however, of low confidence. Hence they are handled by the probability weighting of the detected points in the observation model of the particle filter. In Fig. 3.13, for instance, one of the grippers of the tools is miss detected, but has substantially lower confidence; meanwhile the correctly detected points have confidence scores higher than 70%. Similarly, in the second case, two features are detected on the wrong side of the shaft due to the surgical tools symmetry, but again with low confidence and hence is not detrimental to the pose estimation. Overall, the feature detection is robust and results in accurate perception.

Deep learning has not been utilized as a major tool in surgical robotic perception, with a lack of training data as the primary bottleneck. The SuPer Deep framework, incorporating two deep neural networks as major components, shows that the challenge of insufficient data is surmountable. Using transfer learning, even on limited training data, the framework accomplishes excellent feature detection for surgical scene perception. Our comparative study on deformable tissue tracking, which utilizes the deep neural networks with only pretrained weights, shows that deep learning techniques can be applied for stereo reconstruction and gives a performance evaluation on these techniques applied to a surgical context.

3.4 Acknowledgements

Chapter 3, in part, is a reprint of material from Y. Li*, F. Richter*, J. Lu, E. K. Funk, R. K. Orosco, J. Zhu, M. C. Yip "Super: A Surgical Perception Framework for Endoscopic Tissue Manipulation with Surgical Robotics," in *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2294-2301, 2020. (* indicates equal contribution) The dissertation author is one of the primary author of this paper. Chapter 3, in part, is also a reprint of material from J. Lu, A. Jayakumari, F. Richter, Y. Li, M. C. Yip "Super Deep: A Surgical Perception Framework for Robotic Tissue Manipulation using Deep Learning for Feature Extraction," in *IEEE International Conference on Robotics and Automation*, pp. 4783-4789, 2021.

Chapter 4

Perception of Blood from Vessel Ruptures

Hemostasis describes the surgical scene when there is no active bleeding, and the tissues are unobstructed by blood, making maintenance of hemostasis a crucial surgical task. However, surgical manipulation of any type (e.g., suction, grasping, retraction, cutting, dissection) can immediately cause bleeding. Bleeding can also occur in a delayed manner. For example, if a vessel is not definitively sealed, it can rupture spontaneously without direct contact. Previous work in perception algorithms for blood has not investigated surgical scenarios and largely from the context of Wireless Capsule Endsocopy (WCE) [83]. Toward this end, we present perception algorithms for vessel ruptures in a surgical scenario in this work. Furthermore, the algorithms are designed for downstream automation techniques for hemostasis maintenance. In order to trigger reactive hemostasis maneuvers, vessel ruptures must be detected. Once detected, 3D information about the liquid is required such that a robotic manipulator can begin initiating the steps required to achieve hemostasis. Therefore, we present the following novel contributions in perception for vessel ruptures in surgery:

1. a blood flow segmentation algorithm for endoscopic images to identify and track a vessel rupture in the image frame and
2. a liquid reconstruction technique which only requires 2D image detections of liquids.

The approaches are evaluated in both simulated and real-life scenes, including an in vivo surgery, to show their efficiency. Furthermore, the liquid reconstruction technique can be applied more generally to any liquid and tested on a pouring milk example.

4.1 Related Works

Fluid Reconstruction

Historically, specialized sensor modalities have been used to capture fluid flow in science and engineering. Schlieren imaging [21, 3, 2], Particle Image Velocimetry [43], laser scanners [51], and structured light [44] have all been developed for capturing fluids. These specialized sensors however are not common and often expensive, making them less ideal than visible spectrum sensors which are the only imaging sensors available in the current workflow of endoscopic surgeries. This situation led to a wide range of developments in the field of visible light tomography where a combination of 2D image projections of a fluid are used to reconstruct it in 3D. Recent developments in the field have effectively registered fluids with simulation based fluid dynamics [26] and require only a few camera perspectives for effective reconstruction [174, 34]. These approaches however do not consider incompressible fluids and only focus on gasses in free space (i.e., no collision).

General Liquid Detection and Simulation Registration

While direct reconstruction of liquids has not been done before, there has been work in detecting liquids in the image frame and registering with a simulation. Pools of water have been detected for unmanned ground vehicles [128, 129], and flowing blood has been detected during surgeries for autonomous, robotic suction [136]. Liquids during a pouring task have also been detected using optical flow [168] and Deep Neural Networks [143]. Mottaghi et al. were able to estimate a liquid's volume in a container from images directly [104]. Registration of a liquid simulation with the real world has also been conducted for robot pouring [45, 142, 144]. However, these techniques require prior information about the liquid being reconstructed, such as knowing the volume of the liquid before hand. Meanwhile in this work, we only assume prior knowledge of the gravity direction and collision environment and use a novel branching strategy to dynamically adjust the volume of the reconstructed liquid. Nevertheless, we integrated Schenck and Fox's most recent simulation registration work [144] to the best of our ability into our reconstruction approach for comparison.

Blood Detection

Previous work on blood detection is largely from the context of WCE where image processing for detections is used to speed up clinician workflow [83]. The typical approach to blood detection in WCE is to classify either on a pixel level or using patch-based methods [36]. The feature space used for classification is either direct Red, Green, Blue [85] channels or the transformed HSV channels [62]. To efficiently process these color spaces, techniques such as support vector machines [113], chromium moments combined with binary pattern textures [80], and neural networks [119, 35] have been proposed. While these methods are robust to small individual lesions, in a surgical scene there can be stains from previous ruptures and larger amounts of blood flow that make the problem of blood detection and tracking its dynamic motion a more challenging and complex problem.

4.2 Segmenting Blood Flow from Images

An outline of the complete approach for detecting blood in the image frame is shown in Alg. 2. At a high level, the blood region is estimated by updating a probability map on the scene, which describes the probability of each pixel in the image frame being blood or not. The probability map is updated using a novel bayesian filtering strategy. From the probability map, the blood region is extracted.

Algorithm 2: Blood Flow Segmentation from Images

```
1 Initialize image frame probability map of blood: $\mathbf{P}_0$
  // Note: Each pixel $p_t \in \mathbf{P}_t$ holds the posterior probability of it
     being blood (i.e. $p_t = P(p_t = \mathtt{b}|z^p_{1:t})$)
2 Initialize optical flow CNN with $l$ image frame inputs
3 while new image, $\mathbb{I}_t$, arrives do
4     $\mathbf{O}_t \leftarrow getOpticalFlowCNN(\mathbb{I}_t, \ldots, \mathbb{I}_{t-l-1})$
5     for pixel $p_t \in \mathbf{P}_t$ do
          // Get Detections from Image
6         if $||\mathbf{O}_t(p)|| > \gamma_o$ then
7             $z^p_t \leftarrow \mathrm{b}$
8         else
9             $z^p_t \leftarrow \overline{\mathrm{b}}$
          // Predict Step
10        $p'_t \leftarrow P(p_{t+1} = b|p_t = b) \cdot p_t$
11        $p'_t \leftarrow p'_t + P(p_{t+1} = b \mid p_t = \overline{\mathrm{b}}, k^p_t = b) \cdot P(k^p_t = b \mid z^p_{1:t}) \cdot (1 - p_t)$
12        $p'_t \leftarrow p'_t + P(p_{t+1} = b \mid p_t = \overline{\mathrm{b}}, k^p_t = \overline{b}) \cdot (1 - P(k^p_t = b \mid z^p_{1:t})) \cdot (1 - p_t)$
13        $p_t \leftarrow p'_t$
          // Update Step
14        $p_t \leftarrow P(z^p_t|p_t = b) \cdot p_t / \left(P(z^p_{t+1} \mid p_{t+1} = \mathrm{b}) + P(z^p_{t+1} \mid p_{t+1} = \overline{\mathrm{b}})\right)$
      // Get detected blood flow
15    $\mathbf{B}_t \leftarrow \mathbf{P}_t > 0.5$
16    $\mathbf{B}_t \leftarrow errodeAndDilate(\mathbf{B}_t)$
17    $\mathbf{B}_t \leftarrow largestConnectedRegion(\mathbf{B}_t)$
```

4.2.1 Detection in the Image Frame

Optical flow is chosen to detect flowing blood because it extracts information about all moving objects in the scene. In a surgical scene, the main motion comes from surgical tools and flowing fluids. Another source of motion in robotic surgery comes from a moving camera, but for robotic surgery the camera remains stationary when work is being done in a scene and its position is reset only to change the field of view. Therefore only stationary scenes are considered. To mask instrument motion from the scene, previously developed methods can be applied to effectively segment and remove pixels attributed to surgical tools from image as done in Chapter 3.

To estimate optical flow, a pretrained convolutional neural network (CNN) is used

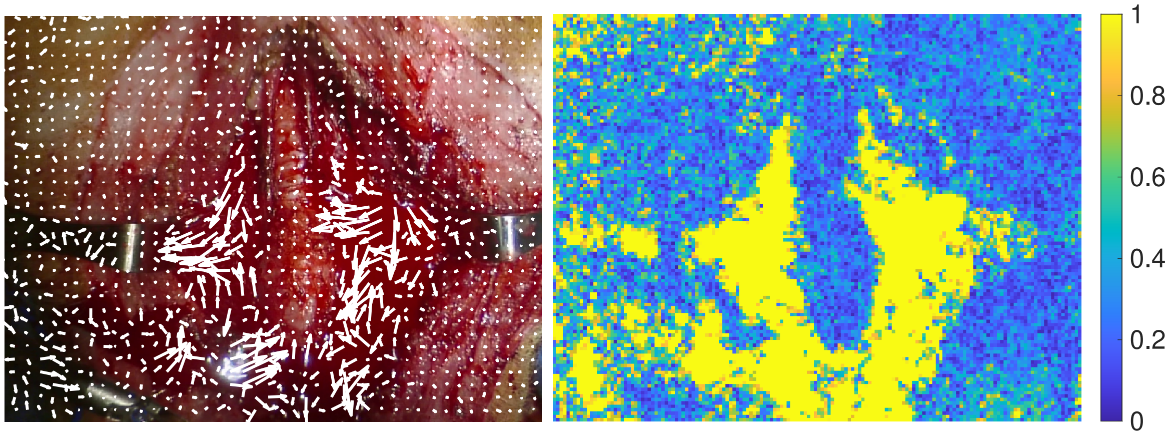


Figure 4.1. The left and right images show the estimated and corresponding magnitude of optical flow, respectively, from a vessel rupture during an in-vivo surgery.

[157]. A deep learning strategy is used instead of traditional methods such as Lucas-Kanade [90] (used in previous work for robot pouring [168]) as traditional optical flow approaches utilize brightness constancy constraint assumption, and this assumption is not valid in endoscopic procedures due to irregular lighting. Meanwhile, the proposed architecture by Teney and Herbert [157] is able to extract motion from learned features that are invariant to textures, brightness, and contrast, which is ideal for detecting flowing blood from an endoscope. The step of estimating optical flow is done on every new image frame from the endoscopic camera as shown in Line 4 of Alg. 2.

Similar to previous work in robot pouring [168], experimental data showed that the magnitude of optical flow is a good signal for detecting fluid motion while the orientation is not. An example of the processed image is shown in Fig. 4.1 comparing the RGB image to the amplitude map for optical flow. The magnitude of the estimated optical flow is considered an observation, and they are fused temporally. Consider the magnitude of optical flow at pixel p. Let z_t^p be the random variable describing the detection of blood at pixel p_t at time t. The detection is modelled where blood is detected, $z_t^p = \text{b}$, if the magnitude of optical flow at pixel p_t is greater than a threshold, γ_o. The inverse is also set, so no blood is detected, $z_t^p = \overline{\text{b}}$, if the magnitude at pixel p_t is less than γ_o. The threshold detection scheme is reflected in Lines 6-9 in Alg. 2. Hence the probability model for these detections can be simply written as

$$P(z_t^p = \mathrm{b} \mid p_t = \mathrm{b}) \qquad\qquad P(z_t^p = \mathrm{b} \mid p_t = \overline{\mathrm{b}}) \tag{4.1}$$

which describes an observation model for the hidden state $p_t \in \{\mathrm{b}, \overline{\mathrm{b}}\}$ which represents the true state of if a pixel is blood or not.

4.2.2 Temporal Filtering for Blood Region Detection

Although the magnitudes of optical flow provide a good initial estimate for blood detection, they are nevertheless noisy and require filtering. Therefore, a temporal filter is based on a Hidden Markov Model (HMM) is proposed to fuse independent measurements of the pixels hidden state over time. The HMM tracks the discrete states for p_t using the observation models in (4.1). Let the following be a transition probability for a pixel p_t be

$$P(p_{t+1} = \mathrm{b} \mid p_t = \mathrm{b}), \tag{4.2}$$

which models the probability that if a pixel is already blood it will continue being blood. In the case of blood vessel ruptures, this should be set close to 1 since the vessel rupture will not stop emitting blood until it has been closed. For the transition probabilities where the pixel is not blood at time t, an additional parameter, k_t^p, is introduced to the model

$$P(p_{t+1} = \mathrm{b} \mid p_t = \overline{\mathrm{b}}, k_t^p = \mathrm{b}) \qquad\qquad P(p_{t+1} = \mathrm{b} \mid p_t = \overline{\mathrm{b}}, k_t^p = \overline{\mathrm{b}}), \tag{4.3}$$

where k_t^p describes the state of the neighboring pixels of p_t. This is modelled as the resulting Boolean-OR operation ($\vee$) on the states of the neighboring pixels

$$k_t^p = \bigvee_{q^i \in \mathbf{A}_p} q_t^i \tag{4.4}$$

where $\mathbf{A}_p$ is the set of neighboring pixels to p_t. Therefore, the first and second models from (4.3) is capturing the flow of the blood and is describing the probability a blood source starts at pixel p_t, respectively. To appropriately describe these processes in this case of blood vessel ruptures, the first model in (4.3) should be set less than (4.2) and the second model in (4.3) should be set close to 0.

The temporal filter is designed to estimate the posterior probability of the state p_t using transition probabilities and observation models. This is done using a predict and update step after every detection. The predict step can be calculated as

$$\begin{aligned} P(p_{t+1} \mid z_{1:t}^p) = P(p_{t+1} \mid p_t = \mathrm{b}) P(p_t = \mathrm{b} \mid z_{1:t}^p) + \\ \sum_{k_t^p \in \{\mathrm{b}, \overline{\mathrm{b}}\}} P(p_{t+1} \mid p_t = \overline{\mathrm{b}}, k_t^p) P(p_t = \overline{\mathrm{b}}, k_t^p \mid z_{1:t}^p) \end{aligned} \tag{4.5}$$

and the update step is computed

$$P(p_{t+1} \mid z_{1:t+1}^p) = \frac{P(z_{t+1}^p \mid p_{t+1}) P(p_{t+1} \mid z_{1:t}^p)}{P(z_{t+1}^p)} \tag{4.6}$$

$$P(p_{t+1} \mid z_{1:t+1}^p) = \frac{P(z_{t+1}^p \mid p_{t+1}) P(p_{t+1} \mid z_{1:t}^p)}{P(z_{t+1}^p \mid p_{t+1} = \mathrm{b}) + P(z_{t+1}^p \mid p_{t+1} = \overline{\mathrm{b}})} \quad . \tag{4.7}$$

However, the predict expression has the joint probability of p_t and k_t^p. Explicit estimation for this joint probability would be computationally intractable, so each pixel's probability of being blood is approximated to be independent of all others at time t. With this simplification, the predict step can be rewritten as

$$P(p_{t+1} \mid z^p_{1:t}) = P(p_{t+1} \mid p_t = \mathrm{b})P(p_t = \mathrm{b} \mid z_{1:t}) + \sum_{k^p_t \in \{\mathrm{b}, \overline{\mathrm{b}}\}} P(p_{t+1} \mid p_t = \overline{\mathrm{b}}, k^p_t)P(k^p_t \mid z^p_{1:t})P(p_t = \overline{\mathrm{b}} \mid z^p_{1:t}) \quad (4.8)$$

and an expression can be found for $P(k^p_t \mid z^p_{1:t})$ using the inclusion-exclusion principle which results in

$$P(k^p_t \mid z^p_{1:t}) = \sum_{j=1}^{|\mathbf{A}_p|} (-1)^{j-1} \sum_{\substack{\mathbf{J} \subseteq \mathbf{A}_p \\ |\mathbf{J}| = j}} \prod_{q^i \in \mathbf{J}} P(q^i_t \mid z^p_{1:t}). \quad (4.9)$$

This results in the ability to compute and track the probabilities of each pixel being blood using (4.8) and (4.7) after every detection as shown in Lines 10-13 and Line 14 in Alg. 2 for the predict and update steps, respectively.

To find the region of blood on the image frame, a mask is generated where all pixels with a posterior probability greater than 0.5 is set to 1, and the rest are set to 0. Then dilation and erosion morphological operations are applied once to reduce noise on the mask. Finally, the largest connected region of the mask is considered the region with blood flowing. This final processing step is reflected in Line 15-17 of Alg. 14.

4.2.3 Experimentation

Implementation Details

All subsequent experiments in this section were ran on a computer with Intel® Core™ i9-7940X Processor and NVIDIA's GeForce RTX 2080. The blood flow detection and trajectory generation algorithms were implemented in MATLAB. The CNN for optical flow estimation [157] is pre-trained on the Middlebury dataset [4], uses $l = 3$ image frames for input, and the resolution of the optical flow estimation is 1/4 of the input frame resolution. These are the default values of the original CNN implementation. The size of the probability map is set to the optical flow resolution. The threshold for detection, γ_o,

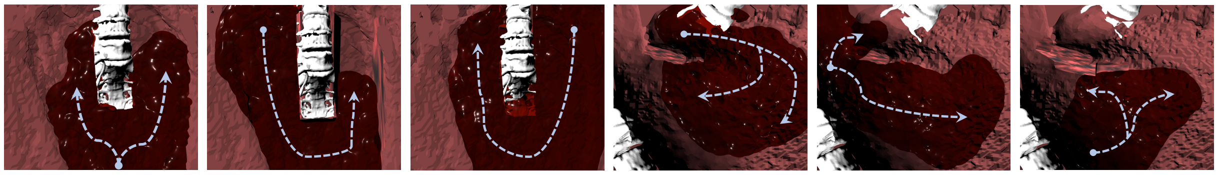


Figure 4.2. Simulated scenes used to evaluate the proposed blood flow segmentation algorithm.

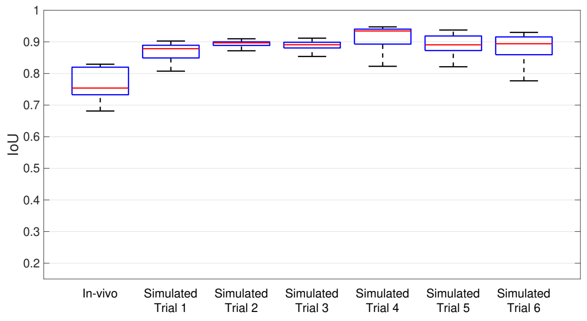
is set to 0.45. The detection probability, $P(z_t^p = \mathrm{b}|p_t = \mathrm{b})$, $P(z_t^p = \mathrm{b}|p_t = \overline{\mathrm{b}})$, are set to 0.95 and 0.2, respectively because experimentally we found the true positive rate and false negative rate to be very accurate and noisy, respectively. The initial probability of a pixel being blood, $P(p_0 = \mathrm{b})$ and transition probabilities of a pixel being blood, $P(p_{t+1} = \mathrm{b}|p_t = \mathrm{b})$, $P(p_{t+1} = \mathrm{b}|p_t = \overline{\mathrm{b}}, k_t^p = \mathrm{b})$, $P(p_{t+1} = \mathrm{b}|p_t = \overline{\mathrm{b}}, k_t^p = \overline{\mathrm{b}})$, are set to 0.1, 0.98, 0.85, and 0.01, respectively. The neighbors for a pixel, $\mathbf{A}_p$, are set to just up, down, left, and right so the algorithm can be ran in real-time.

Datasets

Two separate datasets were generated for this work to evaluate the proposed blood region detection algorithm. Both datasets have labelled ground-truth masks, $\mathbf{G}_t$, of the blood region. Performance is evaluated from these datasets using the IoU metric between $\mathbf{B}_t$, a binary mask of the detected blood region, and $\mathbf{G}_t$.

Simulation: six simulated scenes of flowing blood are generated using Unity3D's particle-based fluid dynamics (PBDs). The scenes are shown in Fig. 4.2. A total of 61 image frames were extracted per scene. The ground-truth mask, $\mathbf{G}_t$, of the blood region is generated by projecting the fluid particles onto a virtual camera's image plane and applying Gaussian smoothing. The rendered image is set to 1095×1284 pixels.

In-Vivo: the second dataset is from an in-vivo surgery. After the completion of a thyroidectomy conducted on a pig (UCSD IACUC S19130), a rupture occurred on the carotid artery. The 8s video data from this incident is used to evaluate the blood flow detection and tracking algorithm in a similar manner to the simulated scenes. For



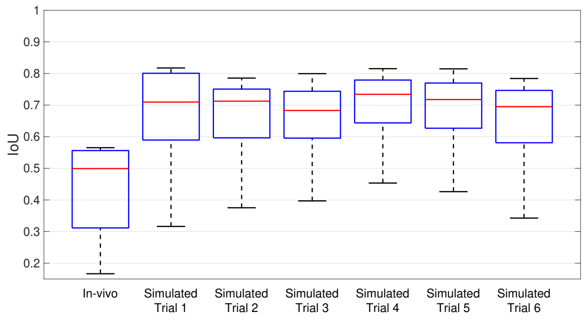


Figure 4.3. The left and right plots show Intersection over Union (IoU) results of blood flow segmentation when using the proposed method and only the blood detection from optical flow thresholding, respectively.

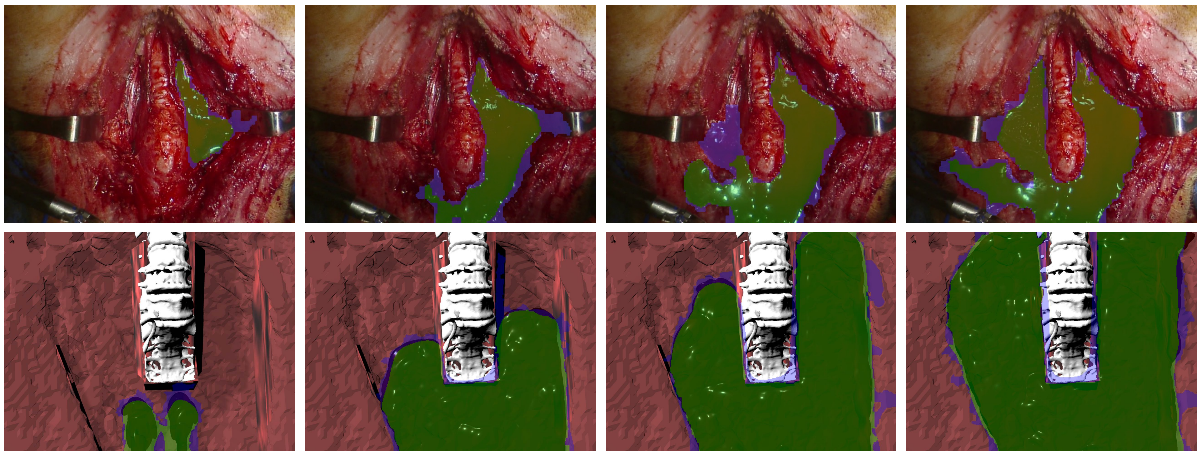


Figure 4.4. Example figures of blood flow segmentation where the green and blue regions indicate the intersection and union minus intersection between the ground truth and blood flow segmentation, respectively.

ground-truth masks of the blood region, $\mathbf{G}_t$, 10 evenly distributed frames were manually annotated. The recorded image size was 640 by 480 pixels.

Results

To show the effectiveness of the tracking algorithm, a comparison experiment was conducted where the blood flow region was simply set to be the pixels with optical flow magnitude greater than γ_O. The distribution of IoU results are shown in Fig. 4.3 for blood flow region detection with and without the tracking algorithm on the simulated scenes and in-vivo dataset. There is a clear difference in performance of the blood flow detection and tracking between the simulated scenes and in-vivo rupture. We believe this is due to the poorer lighting conditions and the reflective surgical clamps used in the in-vivo scene as seen in Fig. 4.4. Nonetheless, the blood region is successfully estimated when using the tracking algorithm despite the many red stains, hence highlighting the importance of using temporal information for detection rather than color features. For additional comparison, Lucas-Kanade's [90] and Farnebeck's [27] optical flow estimation techniques were replaced for the CNN based optical flow estimation [157]. Note that Lucas-Kanade's optical flow estimation is the proposed detection method for fluids by Yamaguchi and Atkeson used for robot pouring [168]. The resulting IoU for the in-vivo and simulated scenes was measured to be consistently below 0.50 in both cases, which is substantially lower than our proposed detection technique.

4.2.4 Discussion

In this section, a novel blood flow detection algorithm for endoscopic image data is presented. To ensure robustness against blood stains, the algorithm relies on temporal information for detection. The novel blood flow detection and tracking algorithm presented offers a unique, probabilistic solution to tracking liquids over 3D cavities and channels, under noisy and harsh visibility conditions, and is a critical perceptual element. This estimation and tracking can help inform a trajectory generation technique to act upon the detected blood from a vessel rupture and clear the surgical field.

4.3 Reconstructing Liquids

From segmented images, denoted as $\mathbf{B}_t$ in the previous section, a 3D understanding of the blood must be derived in order to give sufficient information for a robotic controller to suction the blood. This is accomplished through a novel optimization problem where the liquid is represented with particles. The optimization problem accounts for liquid constraints such as constant density and is seeded with a dynamics prediction based on

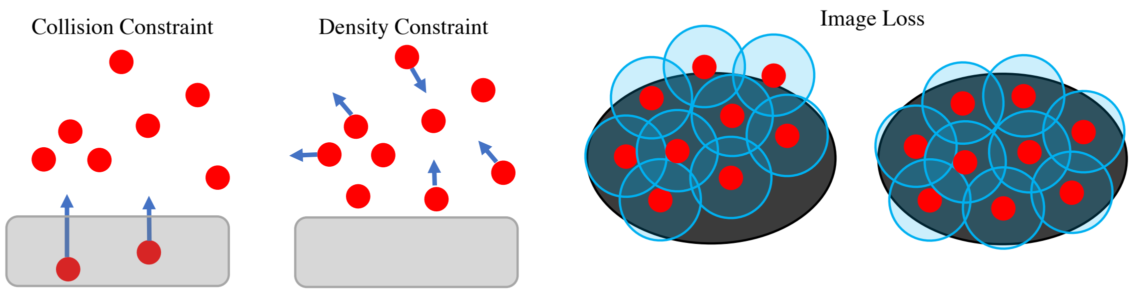


Figure 4.5. A visualization of solving (4.10) in order of left to right where the particle locations are drawn in red. The collision constraint pushes particles out of collision, the density maintains a constant density, and the image loss between the detected and rendered surface is minimized.

the previous time-step. Finally, a branching strategy is presented that dynamically adjust the number of particles in the reconstructed liquid

4.3.1 Problem Formulation

Let $\mathbf{x}_t = \{\mathbf{x}_t^i\}_{i=1}^N$ be the set of particles in $\mathbb{R}^3$ representing the reconstructed liquid at time t. To estimate the particle locations, and hence reconstruct the liquid, we assume only knowledge of a binary masked image which identified the liquids surface, $\mathbf{B}_t$. The estimation for the particles is done by minimizing a loss between the detected surface and a reconstruction of the liquid surface from the particles, $\hat{\mathbf{B}}(\cdot)$. Written explicitly, the optimization problem is

$$\arg\min_{\mathbf{x}_t} \mathcal{L}\left(\mathbf{B}_t, \hat{\mathbf{B}}(\mathbf{x}_t)\right) \quad \text{s.t.} \;\; \mathbf{C}(\mathbf{x}_t) = 0 \tag{4.10}$$

where liquid constraints, $\mathbf{C}(\cdot)$, are applied to the particles positions so they behave like a liquid. The position constraints considered here are density and collision, and a visual explanation is shown in Fig. 4.5. Solving position constraints and deriving velocities from them has produced stable, particle based simulations for large time-step sizes [106, 93]. Similarly, we leverage the liquid-like dynamics induced by position constraints for effective liquid reconstruction from video sequences (i.e. going from t to $t + 1$).

```
Algorithm 3: Reconstruct Liquid at time $t$
Input  : Previous liquid particle positions and velocities, $\mathbf{x}_{t-1}, \dot{\mathbf{x}}_{t-1}$, and
         image, $\mathbf{B}_t$
Output : Updated liquid particle positions and velocities $\mathbf{x}_t, \dot{\mathbf{x}}_t$
// Particle Prediction
1  $\mathbf{x}_t \leftarrow \mathbf{x}_{t-1} + \dot{\mathbf{x}}_{t-1}\Delta t + \frac{1}{2}\mathbf{g}\Delta t^2$
2  for $n_o$ iterations do
3      for $n_j$ iterations do
           // Apply Position Constraints
4          for $n_c$ iterations do
5              $\Delta\mathbf{x}_c \leftarrow solveCollision(\mathbf{x}_t)$
6              $\mathbf{x}_t \leftarrow \mathbf{x}_t + \Delta\mathbf{x}_c$
7              $\Delta\mathbf{x}_\rho \leftarrow solveDensity(\mathbf{x}_t)$
8              $\mathbf{x}_t \leftarrow \mathbf{x}_t + \Delta\mathbf{x}_\rho$
           // Minimize Image Loss
9          for $n_i$ iterations do
10             $\hat{\mathbf{B}}(\mathbf{x}_t) \leftarrow renderSurface(\mathbf{x}_t)$
11             $\mathbf{x}_t \leftarrow \mathbf{x}_t + \alpha_{\mathbf{B}} \left(\partial\mathcal{L}\left(\mathbf{B}_t, \hat{\mathbf{B}}(\mathbf{x}_t)\right)/\partial\mathbf{x}_t\right)$
       // Adjust Particle Count
12     if local_minima_conditions then
13         $\mathbf{x}_t \leftarrow duplicateOrRemoveParticle(\mathbf{x}_t)$
// Update Particle Velocities
14 $\dot{\mathbf{x}}_t \leftarrow (\mathbf{x}_t - \mathbf{x}_{t-1})/\Delta t$
15 $\dot{\mathbf{x}}_t \leftarrow dampVelocityAndApplyViscocity(\mathbf{x}_t, \dot{\mathbf{x}}_t)$
16 return $\mathbf{x}_t, \dot{\mathbf{x}}_t$
```

The following methods detail our solution to the optimization problem shown in (4.10) and an outline is shown in Algorithm 3. First, the position constraints, $\mathbf{C}(\cdot)$, and their respective solvers are described. Second, the rendered surface, $\hat{\mathbf{B}}(\cdot)$ and its gradient with respective to the particle positions to minimize the loss is explained. The constraint solvers and surface loss gradient are applied in a projective gradient descent scheme to solve (4.10) as shown in lines 4 to 11 of Algorithm 3. Third, finding the number of particles, N, to reconstruct the liquid and a strategy of where to add or remove the particles is detailed. Lastly, prediction of the particles from time-step t to $t+1$ is defined to reconstruct from videos of detected liquids, $\mathbf{B}_1, \ldots, \mathbf{B}_T$.

4.3.2 Position Constraints for Liquid Particles

The two position constraints used to reconstruct the liquid when optimizing (4.10) are collision and density. The collision constraint ensures that none of the particles representing the reconstructed liquid are in collision with the scene. Let $C_c(\cdot)$ be the collision constraint for a particle, and it is expressed as

$$C_c(\mathbf{x}^i) = \text{relu}(-SDF(\mathbf{x}^i)) \tag{4.11}$$

where relu$(\cdot)$ is the rectified linear unit function and $SDF(\cdot)$ is the signed distance function of the scene. The collision constraint is satisfied when it is at 0, which occurs by having all of the particles out of collision (i.e. no more negative SDF values at the particle positions).

To push the particles out of collision and satisfy the collision constraint, finite difference is used to approximate a gradient of (4.11) and the particles are moved along the gradient step. This is computed for particle $\mathbf{x}^i$ as follows

$$\Delta\mathbf{x}_c^i = C_c(\mathbf{x}^i) \sum_{\mathbf{k} \in K} w_k SDF(\mathbf{x}^i + d\mathbf{k}) \tag{4.12}$$

where K is the set of finite sample directions (e.g. [±1, 0, 0], [0, ±1, 0], [0, 0, ±1]), w_k is the finite difference weight, and d is the steps size for the sample directions. The finite difference weights are computed optimally [33] and scaled such that the resulting vector from the summation is normalized. The normalization is done so the particles are moved up to the current collision depth, $C_c(\mathbf{x}^i)$, and not in collision free space. The collision constraint is iteratively solved and applied to the particles as shown in lines 5 and 6 in Algorithm 3.

The second constraint, density, ensures that the liquid is in-compressible. The density of particle based representations for liquids can be expressed using the same technique as Smoothed Particle Hydrodynamics (SPH) [42, 91]. SPH simulations compute physical properties from hydrodynamics, such as density, using interpolation techniques with kernel operators centered about the particle locations. Similarly, we compute the density at particle $\mathbf{x}^i$

$$\rho^i(\mathbf{x}) = \sum_{j=1}^{N} W(||\mathbf{x}^i - \mathbf{x}^j||, h) \tag{4.13}$$

where $W(\cdot, h)$ is a smoothing kernel operator with radius h. This is the same as SPH simulations except without the mass term because each particle is set to represent an equal amount of mass in the reconstructed liquid. A density constraint for the i-th particle using (4.14) can be written as

$$C_\rho^i(\mathbf{x}) = \frac{\rho_i(\mathbf{x})}{\rho_0} - 1 \tag{4.14}$$

where ρ_0 is the resting density of the liquid being reconstructed [7]. This density constraint is satisfied at 0 which occurs when the reconstructed liquid is achieves resting density at each of the particle locations. Since each particle represents an equal amount of liquid, the resting density, ρ_0, can be interpreted as the resolution of the reconstruction (e.g. increasing ρ_0 implies more particles are needed to represent the same liquid).

Newton steps along the constraint's gradient are iteratively taken to satisfy the density constraint in (4.14). Each Newton step, $\Delta\mathbf{x}_\rho$, is calculated as

$$\Delta\mathbf{x}_\rho = -\nabla\mathbf{C}_\rho(\mathbf{x}) \left(\nabla\mathbf{C}_\rho^\top(\mathbf{x})\nabla\mathbf{C}_\rho(\mathbf{x}) + \epsilon_\rho\mathbf{I}\right)^{-1} \mathbf{C}_\rho(\mathbf{x}) \tag{4.15}$$

where $\mathbf{C}_\rho(\cdot) = [C_\rho^1(\cdot), \ldots, C_\rho^N(\cdot)]^\top$, the partials are

$$\begin{aligned} \frac{\partial C_\rho^i(\mathbf{x})}{\partial \mathbf{x}^i} &= \frac{1}{\rho_0} \sum_{j=1}^{N} \frac{(\mathbf{x}^i - \mathbf{x}^j)}{||\mathbf{x}^i - \mathbf{x}^j||} W'(||\mathbf{x}^i - \mathbf{x}^j||, h) \\ \frac{\partial C_\rho^i(\mathbf{x})}{\partial \mathbf{x}^j} &= \frac{(\mathbf{x}^i - \mathbf{x}^j)}{\rho_0||\mathbf{x}^i - \mathbf{x}^j||} W'(||\mathbf{x}^i - \mathbf{x}^j||, h) \end{aligned} \tag{4.16}$$

where $W'(\cdot, h)$ is the derivative of smoothing kernel operator in (4.13), and $\epsilon_\rho \mathbf{I} \in \mathbb{R}^{N \times N}$ stabilizes the inversion with a damping factor ϵ_ρ. Enforcing incompressibility in SPH simulations, similar to the proposed density constraint here, when particles have a small number of neighbors is known to cause particle clustering [103]. Therefore, we use Monaghan's solution by adding the following artificial pressure term to $\Delta \mathbf{x}_\rho$:

$$\mathbf{s}^i_{corr} = -\frac{\lambda_s}{\rho_0} \sum_{j=1}^{N} \left(\frac{W(||\mathbf{x}^i - \mathbf{x}^j||, h)}{W(\lambda_\mathbf{x}, h)} \right)^{\lambda_n} \frac{\partial C^i_\rho(\mathbf{x})}{\partial \mathbf{x}^j} \tag{4.17}$$

for the i-th particle where $\lambda_s, \lambda_\mathbf{x}, \lambda_n$ are set according to the original work [103]. The density constraint is iteratively solved with the artificial pressure term and applied to the particles as shown in lines 7 and 8 in Algorithm 3.

4.3.3 Differentiable Liquid Surface Rendering

The loss being minimized in (4.10) to reconstruct the liquid is between the detected surface, $\mathbf{B}$, and the reconstructed surface, $\hat{\mathbf{B}}(\cdot)$. This is equivalent to the differentiable rendering problem formulation, which multiple solutions have been proposed for [64]. The differentiable renderer we employ is Pulsar which renders each particle as a sphere [74] because it is currently state-of-the-art for point-based geometry rendering and requires no training data to get a gradient of the rendered image with respect to the particle locations when not using its shader. The loss used to minimize the difference between the detected surface and rendered surface is the Symmetric Mean Absolute Percentage Error (SMAPE) which is computed as

$$\mathcal{L}\left(\mathbf{B}, \hat{\mathbf{B}}(\mathbf{x})\right) = \frac{1}{N_p} \sum_{u,v \in \mathbf{B}} \frac{|\mathbf{B}_{u,v} - \hat{\mathbf{B}}_{u,v}(\mathbf{x})|}{|\mathbf{B}_{u,v}| + |\hat{\mathbf{B}}_{u,v}(\mathbf{x})| + \epsilon_s} \tag{4.18}$$

where N_p is the number of pixels on the image and ϵ_s is used to stabilize the division. SMAPE was chosen because the ℓ-1 loss was used in the original Pulsar work [74] and

SMAPE is a symmetric version of an ℓ-1 loss. In Algorithm 3, lines 10 and 11 are where the differentiable renderer is integrated into our reconstruction technique with a gradient step size of $\alpha_{\mathbf{B}}$.

4.3.4 Adding and Removing Particles

The number of particles N must be found to solve (4.10), hence making this a mixed-integer optimization problem. To solve for N, we use a branching strategy based on the following heuristic: if the rendered surface area is smaller than the detected surface area, duplicate a particle, $N+1$, and vice-versa to remove a particle, $N-1$. The branching strategy is enabled after confirming a local-minima has been reached with the current number of particles. This is determined by taking the mean image loss gradient and checking if it less than a threshold:

$$\frac{1}{N}\sum_{k=1}^{N}\left\|\frac{\partial\mathcal{L}\left(\mathbf{B},\hat{\mathbf{B}}(\mathbf{x})\right)}{\partial\mathbf{x}^{k}}\right\| \leq \gamma_{s} \tag{4.19}$$

where γ_s is the threshold and if the Intersection over Union (IoU) is less than a threshold:

$$\frac{\mathbf{B}\cup\hat{\mathbf{B}}(\mathbf{x})}{\mathbf{B}\cap\hat{\mathbf{B}}(\mathbf{x})} \leq \gamma_{B} \tag{4.20}$$

where γ_B is the threshold. IoU is chosen over the SMAPE loss because Pulsar renders each sphere with a blending value so the rendered image will have values from $[0,1]$ hence increasing the SMAPE loss as more spheres are rendered even when the spheres make a perfect silhouette fit. Meanwhile IoU directly measures silhouette fit which is in line with our heuristic for the branching strategy. If these two criteria are satisfied, a local-minima due to the number of particles is assumed, and the branching decision of duplicating or removing a particle is triggered. This branching logic is handled in lines 12 and 13 in Algorithm 3.

When duplicating or removing a particle, the collision constraint will remain unchanged and the density constraint will be increased when duplicating a particle and decreased when removing a particle. Therefore, the particle selected to duplicate or remove is chosen to best satisfy the density constraint so the initial particle locations when solving (4.10) after adjusting the particle count remains closest to the density constraint manifold. Written explicitly and using the ℓ-1 loss to describe closeness to the constraint manifold, the index of the particle to duplicate or remove is found by solving

$$\arg\min_i \sum_{k=1}^{N} |C_\rho^{k+}(\mathbf{x})| + |C_\rho^{i+}(\mathbf{x})| \qquad \arg\min_i \sum_{k\neq i}^{N} |C_\rho^{k-}(\mathbf{x})| \tag{4.21}$$

for duplication and removal, respectively, and $C_\rho^{k+}(\cdot)$, $C_\rho^{k-}(\cdot)$ are the density constraint evaluated at particle k after duplicating and removing the i-th particle, respectively. The new density constraints are evaluated as

$$C_\rho^{k+}(\mathbf{x}) = \frac{1}{\rho_0}\sum_{j=1}^{N+1} W(||\mathbf{x}^k - \mathbf{x}^j||, h) - 1 \tag{4.22}$$

$$C_\rho^{k+}(\mathbf{x}) = C_\rho^{k}(\mathbf{x}) + \frac{1}{\rho_0} W(||\mathbf{x}^k - \mathbf{x}^{N+1}||, h) \tag{4.23}$$

for duplicating the i-th particle (so $\mathbf{x}^{N+1} = \mathbf{x}^i$) and

$$C_\rho^{k-}(\mathbf{x}) = \frac{1}{\rho_0}\sum_{j\neq i}^{N} W(||\mathbf{x}^k - \mathbf{x}^j||, h) - 1 \tag{4.24}$$

$$C_\rho^{k-}(\mathbf{x}) = C_\rho^{k}(\mathbf{x}) - \frac{1}{\rho_0} W(||\mathbf{x}^k - \mathbf{x}^{i}||, h) \tag{4.25}$$

for removing the i-th particle where $C_\rho^k(\mathbf{x})$ is the density constraint evaluated at particle k before duplicating or removing a particle. Finally, (4.21) is solved by explicitly computing the loss for every potential i (i.e. computing loss after duplicating or removing every particle) and choosing i that yields the smallest loss, hence duplicating or removing particles that best satisfy the density constraint. Note that this can be efficiently computed due to the simplifications derived in (4.23) and (4.25).

4.3.5 Predicting Liquid

In Position Based Fluid (PBF) simulations, the constraints at every time step update the positions of the particles which in turn induces a velocity for the particles [93]. These constraint induced velocities combined with other external forces such as gravity are used to move the particles forward in time for liquid-like motion of the particles. A similar approach is used here to recreate the liquid-like motion through time and hence enable reconstruction from a video of observations, $\mathbf{B}_1 \ldots, \mathbf{B}_T$. Let $\mathbf{x}_t^{i*}$ and $\mathbf{x}_{t-1}^{i*}$ be the optimized particles from solving (4.10) at time t and $t-1$, respectively. Then the induced velocity for time t is

$$\dot{\mathbf{x}}_t^i = (1-\lambda_d)(\mathbf{x}_t^{i*} - \mathbf{x}_{t-1}^{i*})/\Delta t \tag{4.26}$$

where $\lambda_d \in [0,1]$ is the velocity dampening factor and Δt is the time-step size. For consistent motion, XSPH viscosity [141] is applied

$$\bar{\dot{\mathbf{x}}}_t^i = \dot{\mathbf{x}}_t^i + \lambda_v \sum_{j=1}^{N} \frac{\dot{\mathbf{x}}_t^j - \dot{\mathbf{x}}_t^i}{\rho^j(\mathbf{x})} W(||\mathbf{x}_t^{i*} - \mathbf{x}_t^{j*}||, h) \tag{4.27}$$

where λ_v dictates the amount of viscosity applied. The induced velocity is computed after every timestep of liquid reconstruction as shown in lines 14 and 15 in Algorithm 3. The induced velocity and gravity are used to forward predicts the particles to $t+1$ using equations of motion

$$\mathbf{x}_{t+1}^i = \mathbf{x}_t^{i*} + \bar{\dot{\mathbf{x}}}_t^i \Delta t + \frac{1}{2}\mathbf{g}\Delta t^2 \tag{4.28}$$

where $\mathbf{g}$ is the gravity vector. This forward prediction is done in line 3 of Algorithm 3. The dampening and viscosity not only represent physical properties, but also provide tuning parameters to stabilize the initialization for the next timestep. Dampening, λ_d, dictates how much to rely on the prediction and viscosity, λ_v, adjusts the consistency of the velocity.

4.3.6 Experimentation

Implementation Details

All the arithmetic, e.g. Newton's density constraint step in (4.15), are implemented with PyTorch for its GPU integration [122]. The collision constraint, (4.11), and its solution, (4.12), are implemented with SPNet's *ConvSP* operator and its PyTorch wrapper [144], Kernel $K = \{[\pm 1, 0, 0], [0, \pm 1, 0], [0, 0, \pm 1]\}$, and step size d is equal to the resolution of the $SDF(\cdot)$. The resting density ρ_0 is generated by setting a resting distance between particles because that is more intuitive to adjust. The resting distance between particles is converted to the resting density by packing 1000 particles in a sphere and computing the particle density of the sphere. Then the density constraint parameters to solve (4.14) and (4.15) are set to a resting distance of $0.6h$, $\epsilon_p = 10^2$, and $W(\cdot)$ is set to Poly6 and Spiky Kernels for density estimation and gradient steps, respectively [105]. Differentiable rendering is done with the PyTorch3D framework [130] and $\epsilon_s = 10^{-2}$. The thresholds for adding/removing particles are $\gamma_s = 10^{-3}$ and $\gamma_B = 0.9$, respectively. Velocity dampening and viscosity coefficients are set to $\lambda_d = 0.2$ and $\lambda_v = 0.75$, respectively. The parameters in Algorithm (3) are set to $n_o = 30$, $n_j = 2$, $n_c = 5$, $n_i = 5$, and $\alpha_{\mathbf{B}} = 0.02$. All datasets are stereo so an initial four particles can be placed at a stereo-computed, 3D location from the first liquid detections. The last parameters, $SDF(\cdot)$ resolution and particle interaction radius, h, are set depending on the dataset as they need to be adjusted depending on the scale of the reconstruction.

Datasets

Simulated Fountain: The first dataset is generated on a three step fountain, shown in Fig. 4.6, with Blender [20]. Take note how the first step fills in a consistent shape, but

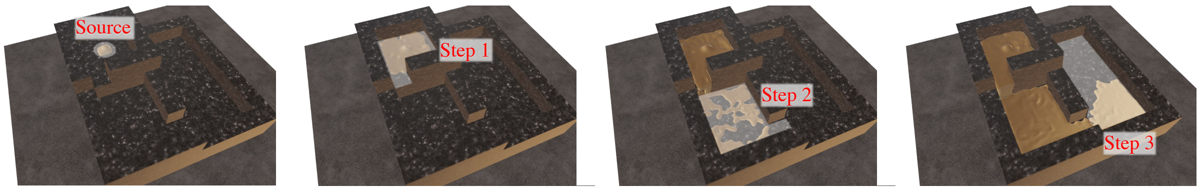


Figure 4.6. The sequence of figures from left to right shows how liquid fills the simulated fountain.

significant turbulence occurs when dropping to the second step making this a challenging component of the scene. Another challenge is by the time the third and final step of the fountain fills, a significant number of particles must be used for reconstruction due to the large volume, hence testing the scalability of the reconstruction method. The liquid simulation uses all default values except the viscosity is set to 0.001. The $SDF(\cdot)$ is generated from the fountain with a resolution of 1cm and the particle interaction radius, h, is set to 1cm. The scene is rendered with 1080p at 24fps stereo cameras, and a mask of the rendered liquid is directly outputted from Blender. For this simulated dataset, the ground-truth liquid mesh is available to evaluate our recontruction with. The metric of 3D IoU is used to capture the shape accuracy of our reconstruction and computed as

$$\text{IoU}_{3D} = \frac{\mathbb{V} \cup \hat{\mathbb{V}}(\mathbf{x})}{\mathbb{V} \cap \hat{\mathbb{V}}(\mathbf{x})} \tag{4.29}$$

where $\mathbb{V}$ and $\hat{\mathbb{V}}(\mathbf{x})$ are voxel representation of the simulated and reconstructed liquid, respectively. The reconstructed liquid in voxel representation, $\hat{\mathbb{V}}(\mathbf{x})$, is generated with the color field, shown in equation (B.1) in Appendix B. The voxel grid is computed at a resolution of 3cm.

Endoscopic Liquid: The second dataset is closer to a vessel rupture scenario and uses a custom silicon cavity that was molded with a 3D printed negative so a $SDF(\cdot)$ for it can be generated. The cavity is 11cm by 9.5cm, $SDF(\cdot)$ resolution is set to 1mm, and particle interaction radius, h, is set to 5mm. To transform the $SDF(\cdot)$ to the camera frame, which is the coordinate frame the particles are being optimized in, an ArUco Marker [40] is

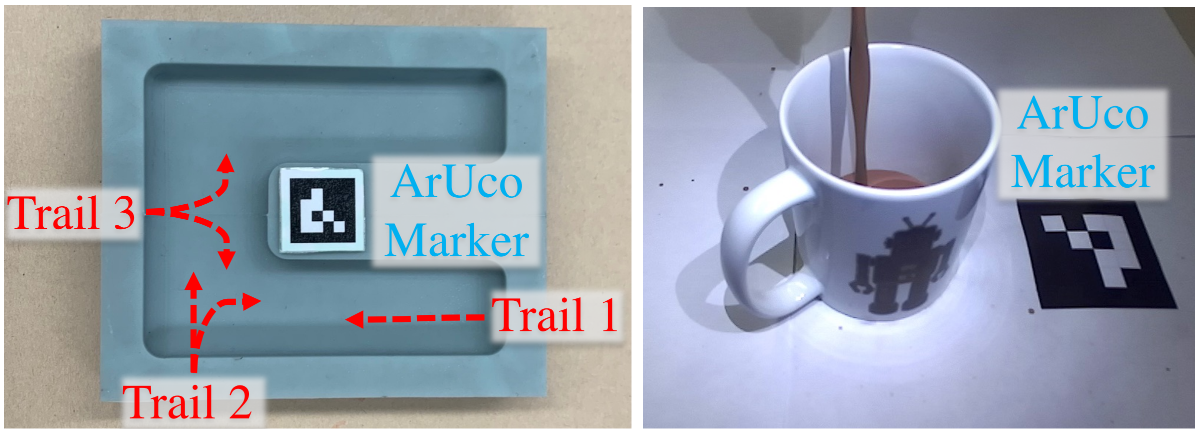


Figure 4.7. The left figure shows a top down view of the silicon cavity used for the Endoscopic Liquid dataset, and the liquid is injected with a syringe at the labelled points for three trials. The right figure shows a camera image from our Pouring Milk experiment set up.

placed on the cavity in a known location. Roughly 50ml of water is injected with a syringe at three different locations for three trails as depicted in Fig. 4.7. The water is mixed with red-coloring dye so color segmentation can be applied to detect the liquid surface. The liquid video is recorded using a dVRK stereo-endoscope which is 1080p at 30fps [65].

Pouring Milk: The third dataset is pouring chocolate milk by a human into a mug as shown in Fig. 4.7. Notice that the milk is partially blocked by the mug, hence testing the reconstructions ability to handle occlusions. The mug is 9cm high and has a 7cm diameter, the $SDF(\cdot)$ resolution is set to 1mm, and particle interaction radius, h, is set to 6.5mm. The mug is placed on a sheet of paper with an ArUco Marker [40] in a marked location. The Aruco Marker and known geometry of the paper provides the transformation to take the $SDF(\cdot)$ to the camera frame. Color segmentation is used to detect the chocolate milk's liquid surface. The liquid video is recorded at 720p 15fps using a ZED Stereo Camera from Stereo Labs.

Comparative Study

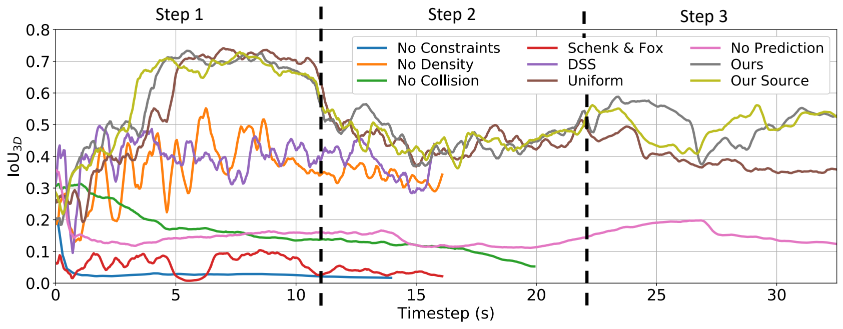
We show the effectiveness of our proposed method through a comparative study. The configurations being compared are:

- *No Constraints* [74] (i.e. no density or collision constraints) and only image loss
- *No Density* constraint
- *No Collision* constraint
- *Schenck & Fox* [144] constraints instead of the density constraint we presented
- *DSS* [170] for rendering gradients rather than Pulsar
- *Uniform* random selection for duplication or removal of particles instead of solving (4.21)
- *No Prediction* of particles (line 3 in Algorithm 3)
- *Our* complete approach
- *Our Source* estimation which adds particles at a source location and detailed in the next sub-section

When removing the collision constraint (i.e. No Constraints and No Collision comparisions), the particle prediction also had to be turned off otherwise the particles will fall forever due to gravity. Without the density or collision constraint, the method is equivalent to differentiable rendering [74] thus giving a baseline comparison. Schenck & Fox proposed their own position-based liquid constraints for constant density [144] (i.e. replacing $\mathbf{C}_\rho$) which were integrated into this method for comparison. We also compared with another recently developed differentiable renderer for point-based geometry called Differentiable Surface Splatting (DSS) [170]. Lastly, a source estimation technique is implemented to highlight how the proposed method can be extended. Implementation details for the Schenk & Fox, DSS, and Our Source comparisons are given in Appendix B.

Results

Quantitative results from the Simulated Fountain dataset are shown in Fig. 4.8, and it shows how effective our proposed approach is in a turbulent, long scene. Note how our



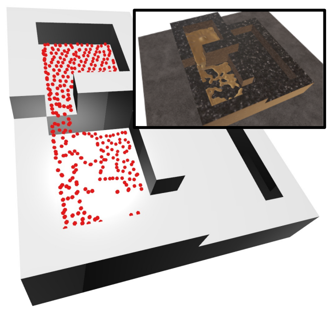


Figure 4.8. The plot on the left shows IoU_{3D} results from the Simulated Fountain datasets along with time-marked points when the liquid reaches different steps in the scene. An example of our reconstruction approach during the turbulent period of the scene is shown on the right figure.

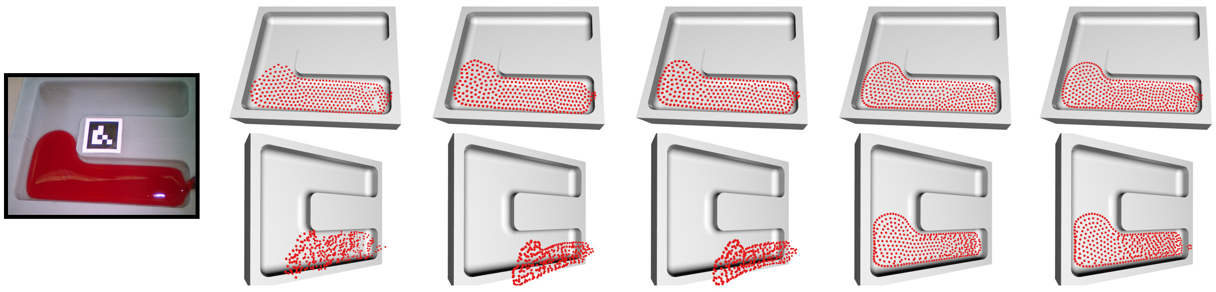


Figure 4.9. From left to right the image columns are an endoscopic image of liquid being reconstruction with: no constraints, no collision constraint, no prediction, our approach, and our source estimation technique.

proposed methods and the uniform comparison are able to reach 70% IoU_{3D} in the first step, and retain a good reconstruction as the very long, and turbulent simulation continues. Meanwhile the compared approaches ran into memory limitations and crashed (required greater than 24GB of memory) or were unable to converge effectively.

Fig. 4.9 shows results from the Endoscopic Trails and how our proposed methods leverage liquid dynamics to fit the cavity shape correctly. The first row of renderings have the virtual camera positioned similar to the real endoscope showing how from that perspective, the particles in red line up with the real image of the liquid. The second row shows another rendered perspective and how our proposed approaches properly reconstruct the liquid in 3D. The first three comparisons are unable to properly reconstruct because they do not leverage a liquids dynamics (i.e. falling to gravity and

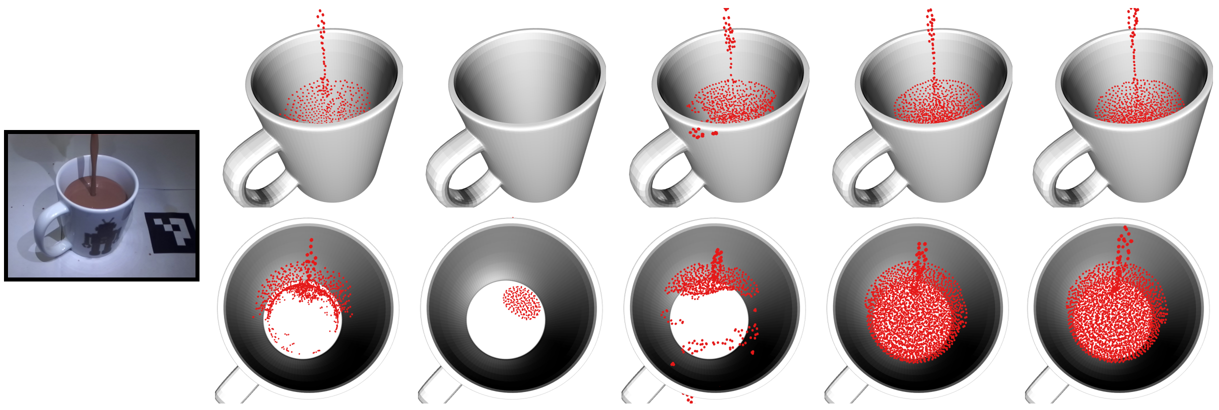


Figure 4.10. From left to right the image columns are an image from the Pouring Milk dataset being reconstructed with: no density constraint, Schenck & Fox constraints [144], DSS [170], our approach, and our source estimation technique.

colliding with the cavity).

From the Milk Pouring experiments, results are shown in Fig. 4.10 which shows that the proposed method infers liquid in occluded regions (i.e. blocked by the mug). The first row of renderings have the virtual camera positioned similar to the raw image showing how from that perspective, the particles in red line up with the real image of the liquid. The second row shows a birds-eye-view perspective and how our proposed approaches properly reconstruct the liquid in 3D. The no density constraint and DSS [170] comparisons are unable to properly reconstruct due to over-fitting on the image loss and fail to make inferences in the occluded region. Meanwhile Schenck & Fox constraints [144] constraints went unstable and splashed particles outside the mug. An additional comparision result is shown in Fig. 4.11 which indicates that our proposed method is able to reconstruct the falling stream, unlike the uniform comparison, due to the novel particle insertion and removal approach.

Performance details on convergence for real-life experiments are shown in Table 4.1 and 4.2. The density constraint ensures incompressibility for the reconstructed liquid and should be 0 when the constraint is satisfied. These results in Table 4.1 show that when applying our constraint solver, the incompressibility property is met. Meanwhile Schenck

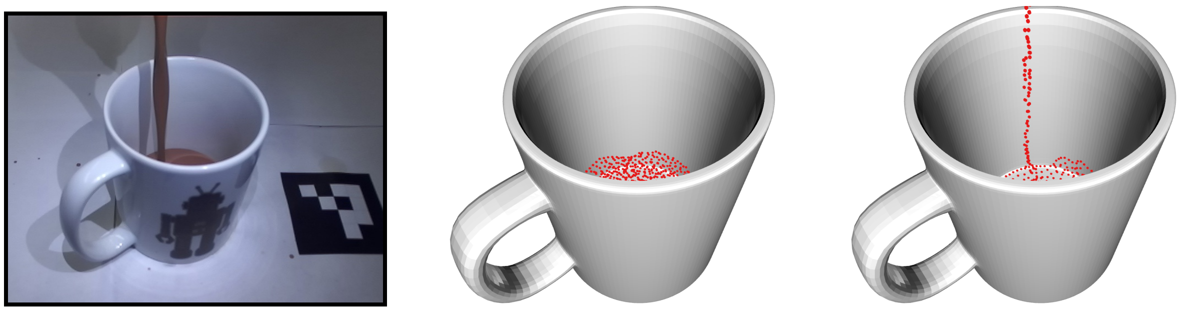


Figure 4.11. The left-most image is from the Pouring Milk dataset being reconstructed with the uniform comparison (middle) and our method (right).

Table 4.1. Mean and standard deviation of the density constraint, defined in (4.14), for the real life experiments.

Method	Simulation	Endo Trail 1	Endo Trial 2	Endo Trail 3	Pouring
No Constraints [74]	-0.03 ± 0.12	-0.25 ± 0.20	-0.09 ± 0.23	-0.01 ± 0.29	-0.08 ± 0.14
No Density	0.89 ± 3.7	0.48 ± 0.65	1.0 ± 0.94	0.91 ± 1.0	66 ± 85
No Collision	$(-5.26\pm61)10^{-5}$	$(-3.2\pm4.9)10^{-3}$	$(-1.4\pm3.3)10^{-3}$	$(-3.7\pm8.7)10^{-4}$	$(-0.33\pm11)10^{-3}$
Schenck & Fox [144]	$(-2.3\pm83)10^{-2}$	-0.11 ± 0.04	-0.12 ± 0.05	-0.11 ± 0.04	0.21 ± 0.37
DSS [170]	$(-1.3\pm25)10^{-3}$	$(-4.6\pm8.4)10^{-3}$	$(-4.9\pm6.6)10^{-3}$	$(-6.4\pm6.4)10^{-3}$	$(-5.2\pm32)10^{-3}$
Uniform	$(-0.05\pm14)10^{-3}$	$(-2.1\pm4.5)10^{-3}$	$(-2.2\pm3.4)10^{-3}$	$(-1.4\pm3.8)10^{-3}$	$(1.3\pm7.3)10^{-3}$
No Prediction	$(-0.51\pm9.9)10^{-3}$	$(-3.1\pm4.6)10^{-3}$	$(-1.5\pm3.4)10^{-3}$	$(-3.5\pm5.0)10^{-3}$	$(-4.6\pm14)10^{-3}$
Ours	$(1.2\pm13)10^{-2}$	$(-1.7\pm3.8)10^{-3}$	$(-2.3\pm4.1)10^{-3}$	$(-1.2\pm3.0)10^{-3}$	$(-0.08\pm4.4)10^{-3}$
Our Source	$(-1.6\pm15)10^{-2}$	$(-1.2\pm3.5)10^{-3}$	$(-1.9\pm4.5)10^{-3}$	$(-1.6\pm8.5)10^{-3}$	$(-0.24\pm7.3)10^{-3}$

Table 4.2. Mean and standard deviation of IoU for the real life experiments.

Method	Simulation	Endo Trail 1	Endo Trial 2	Endo Trail 3	Pouring
No Constraints [74]	0.469 ± 0.133	0.907 ± 0.034	0.904 ± 0.058	0.871 ± 0.063	0.798 ± 0.167
No Density	0.822 ± 0.096	0.915 ± 0.055	0.874 ± 0.214	0.919 ± 0.025	0.867 ± 0.054
No Collision	0.410 ± 0.241	0.904 ± 0.026	0.899 ± 0.048	0.761 ± 0.177	0.869 ± 0.078
Schenck & Fox [144]	0.217 ± 0.113	0.437 ± 0.319	0.882 ± 0.052	0.830 ± 0.100	0.039 ± 0.221
DSS [170]	0.759 ± 0.123	0.916 ± 0.042	0.909 ± 0.081	0.917 ± 0.032	0.815 ± 0.079
Uniform	0.826 ± 0.127	0.900 ± 0.056	0.891 ± 0.061	0.891 ± 0.071	0.576 ± 0.205
No Prediction	0.896 ± 0.049	0.904 ± 0.031	0.899 ± 0.046	0.905 ± 0.020	0.890 ± 0.084
Ours	0.889 ± 0.049	0.902 ± 0.054	0.905 ± 0.051	0.910 ± 0.026	0.849 ± 0.071
Our Source	0.891 ± 0.041	0.911 ± 0.034	0.908 ± 0.055	0.913 ± 0.026	0.843 ± 0.061

& Fox's constraints were unable to reach similar performance. The results in Table 4.2 show that our reconstruction approach is able to achieve comparable image loss performance as the best from No Constraints, No Density, No Collision and No Prediction comparisons. This implies that our approach is effective at converging in image loss with additional constraints (density and collision) and prediction.

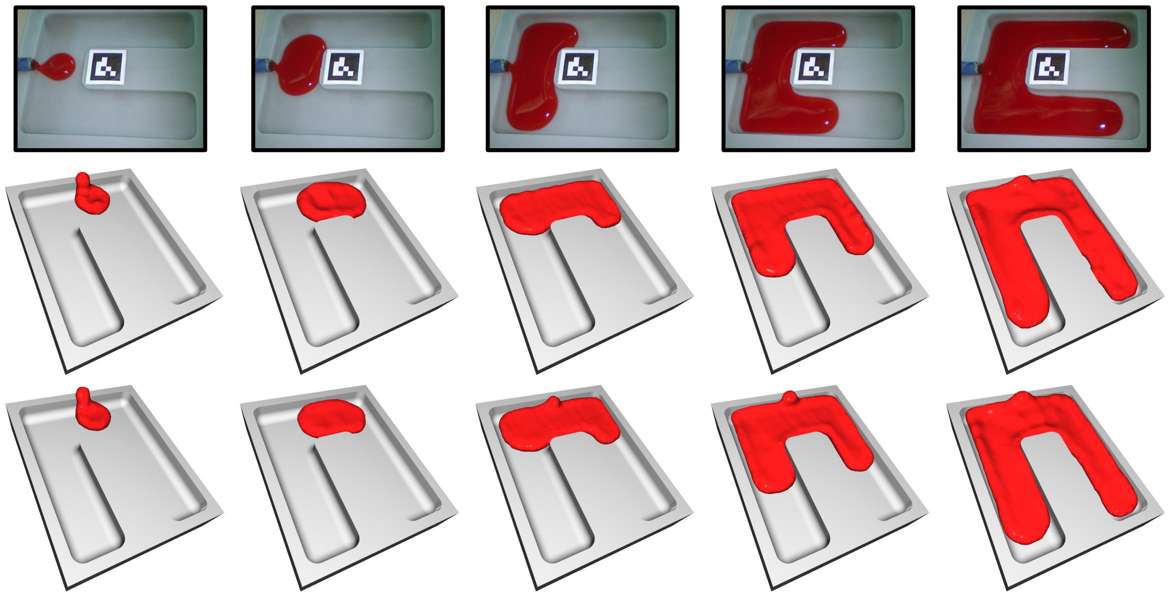


Figure 4.12. Sequence of reconstruction results from Endoscopic Trail 3 where the rows from top to bottom show: endoscopic image, our complete approach, and our source approach.

4.3.7 Discussion

Our experiments highlight the generalizability of our approach through the wide range of liquids (simulated, water, and milk) and cameras (narrow & wide field of view and 15, 24 & 30 fps). In the supplementary material video, consistent particle flow is observed when using the source estimation extension. We envision that the source estimation extension will be beneficial in downstream robotic automation applications such as robotic bar tending [166] and managing hemostasis in surgery where prediction of the liquid is required.

We found that the density constraint, collision constraint, and prediction are crucial to inferring beyond the 2D image loss as seen in Fig. 4.9 and 4.10. Furthermore in longer and more turbulent scenes, the lack of liquid properties can cause instabilities and blow up the mixed-integer optimizer (greater than 10,000 particles). The density constraint can be switched with other constraints that reflect a liquid incompressibility and other liquid properties, such as Schenck & Fox's constraints [144]. However, we were unable to

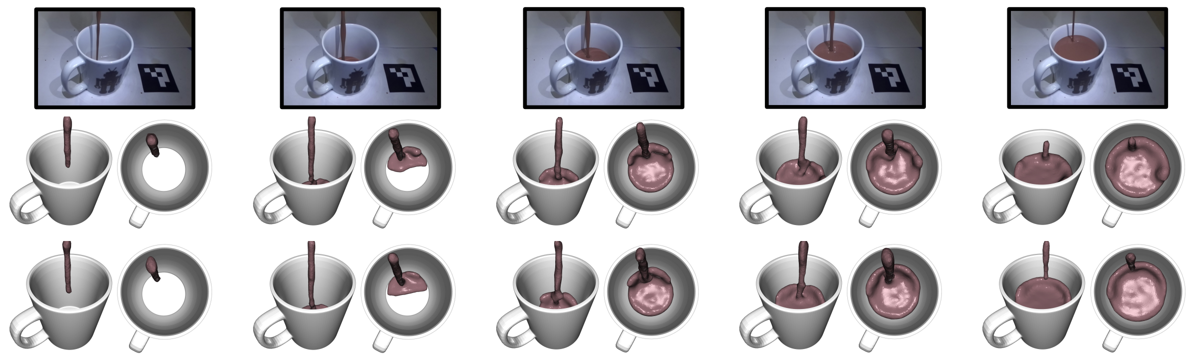


Figure 4.13. Sequence of reconstruction results from Pouring Milk dataset where the rows from top to bottom show: image, our complete approach, and our source approach.

stabilize Schenck & Fox's constraints and found the constraint in (4.14) and its solver to be stable on all of the datasets. Similar is true for the differentiable rendering, and we found Pulsar to be more robust than DSS in our application since DSS requires normals which we observed are not consistently generated. The proposed particle insertion and removal strategy was effective and even able to insert particles to reconstruct a falling stream as seen in Fig. 4.11.

There is a large quantity of hyper-parameters in our method, but this is expected when solving a mixed-integer, optimization problem. Nevertheless, we found a set that generalizes over our diverse datasets, and the interaction radius, h, adjusts the effective resolution of our reconstruction (i.e. smaller h gives a denser reconstruction). An artifact that we observed in our reconstruction approach is the ambiguity of depth due to our observations being limited to only 2D surface detections of the liquid. The incorporation of liquid dynamics does help overcome this challenge in the Endoscopic Liquid experiment as seen in Fig. 4.9. However, in the Pouring Milk experiment the top layer of milk is slightly lopsided which is best seen in Fig. B.1 in Appendix B.

4.4 Acknowledgements

Chapter 4, in part, is a reprint of material from F. Richter, S. Shen, F. Liu, J. Huang, E. K. Funk, R. K. Orosco, M. C. Yip "Autonomous Robotic Suction to Clear the Surgical Field for Hemostasis using Image-based Blood Flow Detection," in *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1383-1390, 2021 and to appear in publication as F. Richter, R. K. Orosco, M. C. Yip "Image Based Reconstruction of Liquids from 2D Surface Detections," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. The dissertation author is the primary author of both papers.

Chapter 5

Robotic Suction Control to Clear the Surgical Field of Blood

Perception techniques for the surgical scene before and during a vessel rupture are proposed in Chapter 3 and Chapter 4, respectively. Ultimately, these methods provide a complete 3D understanding about the surgical scene that is required to maintain hemostasis autonomously. In this work, we propose control and motion plan algorithms that leverage these perception techniques to autonomously direct a robotic suction tool to clear the surgical field. To this end, we propose the following contributions:

1. a visual based controller of robotic end-effectors that utilizes the Lumped Error term proposed in Chapter 2,
2. a naive end-to-emit style motion plan for the robotic suction tool to follow when it clears the surgical field,
3. andw an optimized motion plan based on differentiable simulation for the robotic suction tool to follow when it clears the surgical field.

The Lumped Error controller is used to direct the suction tool along both proposed motion plans. The methods are tested in both a simulated and a real-world environment in a phantom that emulated a vessel rupture scenario.

5.1 Lumped Error Controller

The Lumped Error localizes robots in the camera frame and compensates for joint angle errors when the kinematic chain is only partially visible as shown in Chapter 2. Surgical robots, such as the da Vinci ® Surgical System, often use cable drive systems to keep the surgical tools at a low profile. Therefore, the joint angle errors are non-constant [145], which implies that the Lumped Error is always changing to compensate for the non-constant joint angle errors that are attributed to the cable stretch. In order to obtain a reliable visual controller based on the Lumped Error tracking approach, the Lumped Error transform needs to be iteratively applied at every timestep. The following subsections detail this iterative approach for position and orientation control on the end-effector where the goals are given in the camera frame.

5.1.1 Position Control

Let $\mathbf{p}^c_g \in \mathbb{R}^3$ be the position goal in the camera frame to which the controller directs the end-effector. Let $\mathbf{b}^e_t \in \mathbb{R}^3$ be the incorrect position of the end-effector in the robot base frame which is computed through forward kinematics with the noisy joint angle measurements, $\tilde{q}^i_t$. The controller iteratively transforms the goal, $\mathbf{p}^c_g$, to the virtual base defined by the Lumped Error, and the error, $\mathbf{d}^e_t \in \mathbb{R}^3$, in the virtual base is computed as:

$$\mathbf{d}^e_t = \left(\mathbf{T}^c_{b-}\mathbf{T}^{b-}_{n_b}(\hat{\mathbf{w}}_t, \hat{\mathbf{b}}_t)\right)^{-1} \overline{\mathbf{p}}^c_g - \overline{\mathbf{b}}^e_t. \tag{5.1}$$

This error is then used to update the end-effector position

$$\overline{\mathbf{b}}^e_{t+1} = \begin{cases} \gamma_s \frac{\mathbf{d}^e_t}{||\mathbf{d}^e_t||} + \overline{\mathbf{b}}^e_t & \text{if } ||\mathbf{d}^e_t|| > \gamma_s \\ \mathbf{d}^e_t + \overline{\mathbf{b}}^e_t & \text{if } ||\mathbf{d}^e_t|| \leq \gamma_s \end{cases} \tag{5.2}$$

where γ_s is the max step size. The updated end-effector position, $\mathbf{b}^e_{t+1}$, is set on the robotic suction tool using inverse kinematics and joint level regulators which use the noisy joint angle readings, $\tilde{q}^i_t$, as feedback. These operations are repeated until the error, $||\mathbf{d}^e_t||$, is less than some threshold.

5.1.2 Orientation Control

The orientation is regulated in a similar fashion as the position control. Let $\mathbf{R}^c_g \in SO(3)$ and $\mathbf{R}^e_t \in SO(3)$ be the goal orientation in the camera frame and incorrect orientation of the end-effector in the robot base frame computed through forward kinematics with the noisy joint angle measurements, $\tilde{q}^i_t$, respectively. The orientation of the end-effector is iteratively set to

$$\mathbf{R}^e_t = \left(\mathbf{R}^c_{b-}\mathbf{R}^{b-}_{n_b}(\hat{\mathbf{w}}_t)\right)^{-1}\mathbf{R}^c_g \tag{5.3}$$

where $\mathbf{R}^c_{b-} \in SO(3)$ and $\mathbf{R}^{b-}_{n_b}(\hat{\mathbf{w}}_t) \in SO(3)$ are the rotation matrix of $\mathbf{T}^c_{b-}$ and $\mathbf{T}^{b-}_{n_b}(\hat{\mathbf{w}}_t, \hat{\mathbf{b}}_t)$, respectively. Similar to the position of the end-effector, the orientation $\mathbf{R}^e_t$ is set using inverse kinematics and joint level regulators which use the noisy joint angle readings, $\tilde{q}^i_t$, as feedback.

5.2 End-to-emit Motion Plan

In a vessel rupture scenario, the blood naturally flows from the vessel rupture location. Therefore, in order to clear the surgical field, we can generate a naive motion plan which goes from end of the blood flow stream to the source, i.e. end-to-emit. The end-to-emit motion plan is generated from the binary mask $\mathbf{B}_t$, from Section 4.2, which highlights which pixels from the image frame are part of the blood flow. The largest connected region of the mask, $\mathbf{B}_t$, is considered the region with blood flowing if its size is greater than a threshold of γ_B. This threshold keeps a detection from occurring when

Algorithm 4: Trajectory Generation for Suction

```
Input  : Start pixel, $s_t$, end pixel, $e_t$, and mask of blood region $\mathbf{B}_t$
Output : Trajectory $\mathbf{T}_t$
1  Initialize clearance reward $\mathbf{R} \leftarrow \mathbf{0}$
2  Initialize temporary eroded blood region $\mathbf{E} \leftarrow \mathbf{B}_t$
3  Initialize $i \leftarrow 0$
4  while $\mathbf{E} \neq \mathbf{0}$ and $i < \gamma_r$ do
5      $\mathbf{E} \leftarrow \text{erode}(\mathbf{E})$
6      $\mathbf{R} \leftarrow \mathbf{R} + r\mathbf{E}$
7      $i \leftarrow i + 1$
8  Initialize pixels cost to go to goal: $\mathbf{D} \leftarrow \infty$
9  Initialize visited map: $\mathbf{V} \leftarrow !\mathbf{B}_t$
10 Initialize parents of nodes: $\mathbf{K} \leftarrow$ UNDEFINED
11 $\mathbf{D}(s_t) \leftarrow 0$
12 while $\mathbf{V} \neq \mathbf{1}$ do
13     $u \leftarrow \underset{pixel}{\text{argmin}}\ \mathbf{D}$
14     $\mathbf{V}(u) \leftarrow 1$
15     if $u = e_t$ then
16         break
17     for pixel $q$ neighboring $u$ and $\mathbf{V}(q) \neq 1$ do
18         if $\mathbf{D}(q) > ||q - u|| - \mathbf{R}(q)$ then
19             $\mathbf{D}(q) \leftarrow ||q - u|| - \mathbf{R}(q)$
20             $\mathbf{K}(q) \leftarrow u$
21 Initialize trajectory for output: $\mathbf{T}_t \leftarrow [\,]$
22 Initialize parent traversal node: $u \leftarrow e_t$
23 while $u \neq s_t$ do
24     Insert $u$ at beginning of $\mathbf{T}_t$
25     $u \leftarrow \mathbf{K}(p)$
26 return $\mathbf{T}_t$
```

there is no actual blood flowing.

A start and end point must be decided to generate a trajectory for suction. The end point should be roughly near the location of the vessel rupture in order to continuously remove any newly released blood. Meanwhile, the starting point should be downstream of the flowing blood in order to effectively clear the surgical field when suctioning upstream towards the source. Therefore, a simple estimation for the start and end point is done

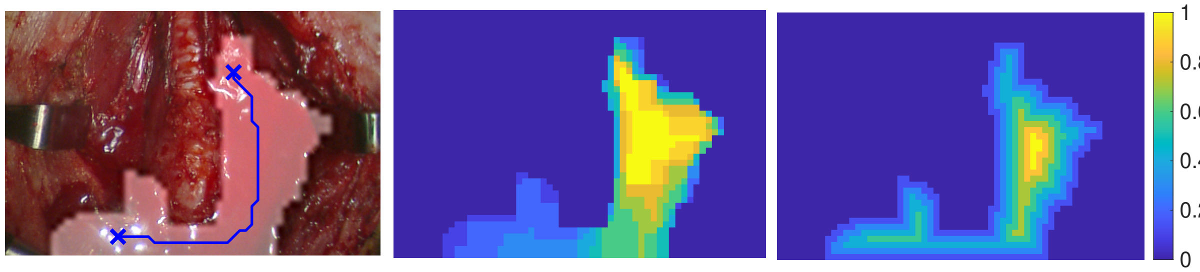


Figure 5.1. Trajectory generation for blood suction from in-vivo scene where the start and end points are chosen from the newest and oldest blood pixels, respectively, (shown in center plot) and a clearance reward is applied to center the path in the blood region (shown in right plot).

based on the age of the pixels in the blood region. The pixel with the largest and smallest ages in the blood region are defined as the end and start points, respectively. To ensure that the end point is not generated at the exact edge of the blood stream, the blood region is eroded before selecting it.

The trajectory generated from the start to end point should also maximize its ability to suction blood while moving upstream. Therefore, using standard minimum distance paths are not ideal as they would tend to plan towards the edges of the blood region rather than the center. To center the trajectory in the blood region, an additional clearance reward is given to the motion planner. The clearance reward is generated by iteratively eroding the blood region for a max of γ_r iterations. The pixels left in the eroded region are given an additional reward of r at each iteration. The final trajectory in the image frame is then generated using Dijkstra's algorithm where the path is constrained to stay within the blood region and the clearance reward is subtracted from the normal distance cost. An outline of this trajectory generation technique is shown in Algorithm 4, and an example is shown in Fig. 5.1. The trajectory is then executed if it is longer than a threshold γ_T. This threshold gives time for the start and end points to stabilize so an effective trajectory can be generated.

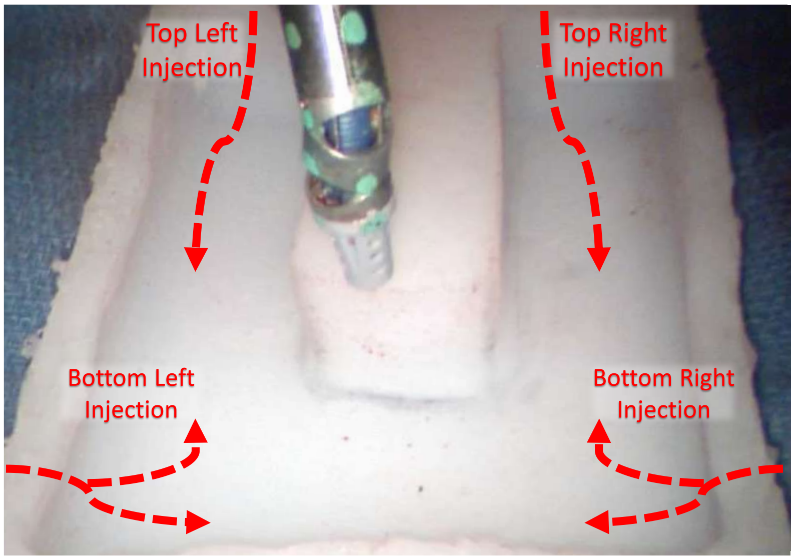


Figure 5.2. Endoscopic view of phantom used for the automated suction experiments. The red arrows highlight the direction of flow at the four injection points tested in the experiment.

5.2.1 Experimentation

Implementation Details

All subsequent experiments were ran on a computer with Intel® Core™ i9-7940X Processor and NVIDIA's GeForce RTX 2080. The blood detection algorithm that provides the binary mask is implemented as done in Section 4.2.3 where the only change is a decrease of image resolution by a factor of 4 to make the algorithm run in real-time. The threshold for region size, γ_B, maximum number of erosions for clearance, γ_r, and trajectory length, γ_T, are set to 20, 4, and 30, respectively. The clearance reward per erosion, r, is set to 0.2.

Experimental Setup

To evaluate the complete autonomous surgical task of recognizing blood flow and performing autonomous suction, a tissue phantom with a cavity for liquid to flow through is constructed from silicone, and water with red coloring dye is drained into the cavity using surgical tubing as shown in Fig. 5.2. A stereoscopic camera with a resolution of

1080×1920 pixels at 30fps on the dVRK [65] is used. The trajectory generated for suction is converted into 3D position commands using the Pyramid Stereo Matching Network (PSMNet) [12], which takes the stereo-images of the cameras and determines the depth of each pixel. PSMNet's weight are provided by their original implementations without any task-specific fine-tuning; the maximum disparity is set to 192. PSMNet estimated depth using an image size of 640 by 480 pixels meanwhile the blood flow detection algorithm from Section 4.2 used a reduced image size of 160 by 120 pixels to improve its speed. A Patient Side Manipulator (PSM) from the dVRK [65] was equipped with da Vinci® Si Suction Tool and followed the generated trajectory to clear the simulated surgical cavity from blood.

The Lumped Error controller from Section 5.1 is ran at 100Hz and is repeated until $||\mathbf{d}_t^e|| < 2$mm per target position, $\mathbf{p}_g^c$, from the generated trajectory. Meanwhile the orientation of the suction tip is set to always be in direction of gravity. The position, $\mathbf{b}_{t+1}^e$, and orientation of the end-effector is converted to joint angles using an analytical inverse kinematic solution. These joint angles are then regulated using dVRK [65].

To account for imperfections in the 3D depth estimation from PSMNet and surgical tool tracking to regulate the end-effector, the suction tool was commanded to oscillate in and out along the direction of gravity an additional 5mm at every point on the trajectory. This probing behavior ensured that the tool always sucked up the blood and neither drifted above the blood nor penetrated and dragged tissue. The Robot Operation System (ROS) is used to encapsulate the image processing and robot trajectory tracking processes [153].

Roughly 50mL of liquid is injected using a syringe into the cavity at one of the four locations highlighted in Fig. 5.2. Before each trial, the end-effector is centered in the middle of the silicon mold such that it does not obstruct any of the injected liquid as shown in Fig. 5.2. The percentage of liquid removed from the cavity, time to react to the injected fluid, the time to complete the trajectory were measured to evaluate the

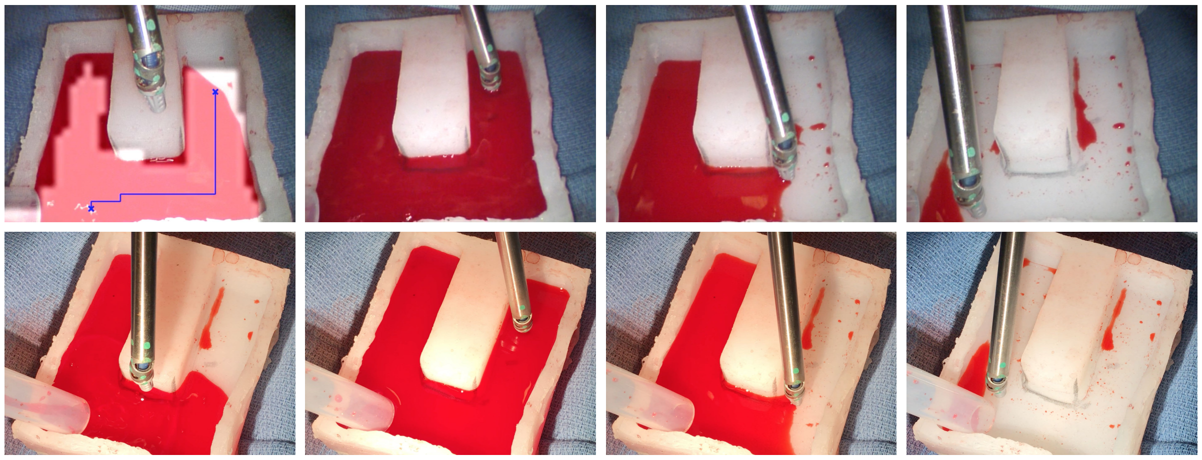


Figure 5.3. Sequence of figures (from left to right) of an automated suction experiment where the liquid is injected at the bottom left corner.

Table 5.1. Mean results from automated suction experiment at each injection point.

Injection Point	**Liquid Suctioned**	**Time to React**	**Time to Execute Trajectory**
Top Left	96.7%	4.1s	45.3s
Top Right	93.6%	2.6s	47.4s
Bottom Left	96.1%	4.7s	23.9s
Bottom Right	96.8%	6.4s	38.1s

performance of the proposed automation method. The percentage of liquid removed was measured by weighing the silicon mold and syringe before and after each trial. Time to react refers to the time taken to detect the flowing blood and generate a trajectory (i.e., completing Algorithm 4) from the first moment that the injected blood is visible in the camera frame. To ensure consistency of the proposed automation method, the experiment is repeated ten times at each of the four injection spots.

Results

The results from the total 40 trials of the automated suction experiment are shown in Table 5.1 and an example sequence is shown in Fig. 5.3. During the experiments, we noticed the liquid generally pooled near the bottom left injection point since it was slightly lower with respect to gravity than the rest of the cavity. This lead to shorter trajectories being generated, and hence less time to execute them as seen in the results, for the bottom left corner experiment compared to the others.

5.2.2 Discussion

In this work, a complete automated solution for clearing the surgical field from blood is presented. The solution provides uses perception defined in Section 4.2, a motion planning technique, and a control strategy from Section 5.1 for the task of clearing blood. The motion plan is informed by the blood flow detection algorithm and is executed using the Lumped Error control strategy. A clearence reward is applied to the motion plan to maximize the blood suctioned by the suction tool and be robust against imperfect blood region estimation. This is the first step taken towards automation of a crucial surgical task, hemostasis, which can occur in any surgery at any time. While the results show promise in clearing the surgical field, there is no guarantee that the end-to-emit style of suction is optimal. Therefore, in the coming section we explore an motion plan technique that optimizes the suctions motion to remove as much blood as quickly as possible.

5.3 Optimizing the Motion Plan

The approach in Section 5.2 is an end-to-emit style of suction to clear the surgical field of blood. While this approach is empirically shown to be effective, there is no guarantee that it is optimal. Therefore, in this section, we cover a novel motion plan approach for robotic suction tools to clear the surgical field from blood. The approach leverages a differentiable simulation of the blood flow and robotic suction tool so a cost function can be optimized via gradient descenent. The cost function is defined and applied in a Model Predictive Control (MPC) formulation to remove as much blood from the surgical scene as quickly as possible.

Algorithm 5: Simulation of Robotic Suction with Liquid

```
   Input  : Previous liquid particle positions and velocities, $\mathbf{x}_{t-1}, \dot{\mathbf{x}}_{t-1}$
   Output : Updated liquid particle positions and velocities $\mathbf{x}_t, \dot{\mathbf{x}}_t$
   // Particle Prediction
1  $\mathbf{x}_t \leftarrow \mathbf{x}_{t-1} + \dot{\mathbf{x}}_{t-1}\Delta t + \frac{1}{2}\mathbf{g}\Delta t^2$
   // Apply Position Constraints
2  for $n_c$ iterations do
3      $\Delta\mathbf{x}_c \leftarrow solveCollision(\mathbf{x}_t)$
4      $\mathbf{x}_t \leftarrow \mathbf{x}_t + \Delta\mathbf{x}_c$
5      $\Delta\mathbf{x}_\rho \leftarrow solveDensity(\mathbf{x}_t)$
6      $\mathbf{x}_t \leftarrow \mathbf{x}_t + \Delta\mathbf{x}_\rho$
7      $\Delta\mathbf{x}_u, \Delta\mathbf{x}_p \leftarrow solveSuctionDisplacement(\mathbf{x}_t)$
8      $\mathbf{x}_t \leftarrow \mathbf{x}_t + \Delta\mathbf{x}_u + \mathbf{n}_y\Delta\mathbf{x}_p$
   // Update Particle Velocities
9  $\dot{\mathbf{x}}_t \leftarrow (\mathbf{x}_t - \mathbf{x}_{t-1})/\Delta t$
10 $\dot{\mathbf{x}}_t \leftarrow dampVelocityAndApplyViscocity(\mathbf{x}_t, \dot{\mathbf{x}}_t)$
11 return $\mathbf{x}_t, \dot{\mathbf{x}}_t$
```

5.3.1 Differentiable Position Based Fluid Simulation

Similar to Section 4.3, the simulation is based on PBF because it has good stability over large timesteps [93]. An outline of the simulation is shown in Algorithm 5 where $\mathbf{x}_t, \dot{\mathbf{x}}_t$ are the particle positions and velocity representing the liquid. Solving the position constraints, $\Delta\mathbf{x}_c, \Delta\mathbf{x}_\rho$, are solved for using (4.12) and (4.15), respectively. The suction displacements, $\Delta\mathbf{x}_u, \Delta\mathbf{x}_p$, are detailed in the upcoming subsection. The simulation timestep process (i.e. the particles going from t to $t+1$) is implemented in a auto-differentiation framework (e.g. PyTorch [122]) such that gradients can be computed like $\frac{\partial \mathbf{x}_{t+1}}{\partial \mathbf{x}_t}$.

5.3.2 Suction Model

The suction force from the surgical tool is modelled as a continuous and differentiable force field that pulls nearby particles towards the nozzle of the tool and subsequently adds a large vertical displacement to the particles. Note that we define y as the vertical direction while the x-z plane represents the horizontal plane. The upward displacement field is in the vertical direction, as this is the typical orientation of a suction tool attacking pooling blood. It is modelled after a 2D-Gaussian probability density function (PDF) over the x-z plane due to its differentiable properties. The magnitude of the upward displacement field experienced by particle i is proportional to the value of the PDF at $([\mathbf{x}_t^i]_x, [\mathbf{x}_t^i]_z)$. For a particle i with position $\mathbf{x}_t^i$, and a suction nozzle at position $\mathbf{b}_t^e \in \mathbb{R}^3$, the upward displacement, Δ_u, is computed as

$$\Delta\mathbf{x}_{u,i} = \mathcal{K} \cdot \frac{\exp\left(-\frac{\left([\mathbf{x}_t^i]_x - [\mathbf{b}_t^e]_x\right)^2}{2\sigma_x^2} - \frac{\left([\mathbf{x}_t^i]_z - [\mathbf{b}_t^e]_z\right)^2}{2\sigma_z^2}\right)}{\sqrt{(2\pi)^2\sigma_x^2\sigma_z^2}} \tag{5.4}$$

where $[\cdot]_x$, $[\cdot]_y$, and $[\cdot]_z$ are the x, y, z coordinates of the position vectors and σ_x and σ_z are the standard deviations of the 2D Gaussian PDF. The standard deviation controls how narrow the suction region is. Finally, $\mathcal{K} > 0$ controls the size of the upward displacement and is adjusted to control the strength of the suction. For the field that pulls particles towards the end-effector at position $\mathbf{b}_t^e$, the i-th particle will experience a displacement given by

$$\Delta\mathbf{x}_{p,i} = \frac{\mathbf{b}_t^e - \mathbf{x}_t^i}{||\mathbf{b}_t^e - \mathbf{x}_t^i||_2} \cdot \frac{1}{||\mathbf{b}_t^e - \mathbf{x}_t^i||_2^2 + d} \tag{5.5}$$

where d is a constant that limits the maximum displacement when the particle is very close to the center of the end-effector as well as to avoid division by 0. Finally, the upward field and the field that pulls particles towards the end-effector are summed and added to the particle positions along with the PBF solver corrections as shown in line 7, 8 of Algorithm 5, where $\mathbf{n}_y$ is a normalized vector that describes the direction of upward displacement in the simulation (i.e. $\mathbf{n}_y = [0, 1, 0]^\top$). This formulation of suction, embedded in a completely differentiable fluid model, enables efficient and stable calculation of gradients from the particle states to the suction tool position (i.e. $\frac{\partial \mathbf{x}_{t+1}}{\partial \mathbf{b}_t^e}$).

5.3.3 Model Predictive Control Formulation

The overall goal of the proposed control problem is to achieve hemostasis as quickly as possible, which corresponds to when all particles in the modelled scene have been lifted from the underlying tissue surfaces of the scene. This will be formatted as an optimization problem where MPC [10] is used to generate a trajectory that the suction nozzle should follow to optimally clear the surgical field from fluids. Let the control input to the MPC be denoted as $\mathcal{U} = \{\mathbf{b}_t^e\}$. The output of the system is the set of particles $\{\mathbf{x}_t^i\}$. The MPC is computed over a short horizon, h, to find a control trajectory $\{\mathbf{b}_t^{e*}, ..., \mathbf{b}_{t+h}^{e*}\}$ which minimizes the loss over the time frame $[t, t+h]$. The optimization problem is written explicitly as

$$\min_{\mathcal{U}} L = \sum_t \sum_i l(\mathbf{x}_t^i) \tag{5.6}$$

over all timesteps and particles. The per-particle loss is

$$l(\mathbf{x}_t^i) = \begin{cases} \frac{1}{2}\|y_{goal} - [\mathbf{x}_t^i]_y\|_2^2, & \text{if } [\mathbf{x}_t^i]_y < y_{goal} \\ 0, & \text{otherwise} \end{cases} \tag{5.7}$$

where y_{goal} is a target height set above the surgical cavity that the upward displacement, $\Delta x_p, i$, pulls the particles towards. The optimal control is solved for by minimizing (5.6) with gradient descent which is only possible due to the differentiable simulation. Only $\mathbf{b}_t^{e*}$ is applied to the system, and then another entire horizon length of control is computed again starting at $t+1$.

To determine a good initial suction point, $\mathbf{b}_0^{e*}$, a Monte-Carlo approach is utilized. The process involves sampling many possible starting points from the fluid particle positions, performing roll-outs using the MPC algorithm with these samples over a look-ahead window m, and finally selecting the best performing one with regards to

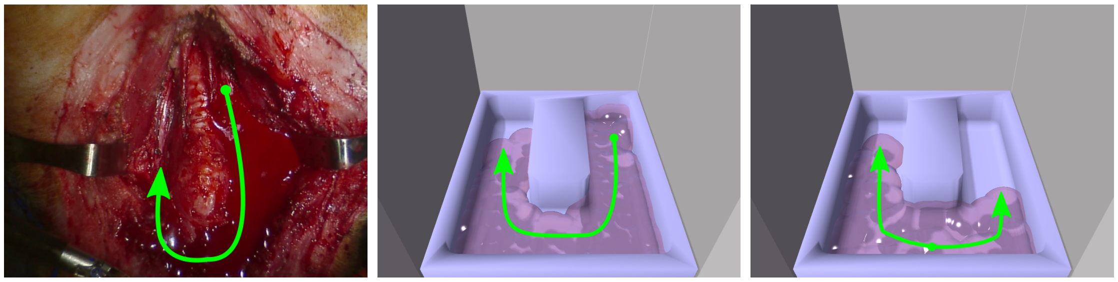


Figure 5.4. Comparison between a real surgical cavity and our two simulated cavity scenes. The green arrows highlight direction of blood flow. The left and right simulated Figures show *case 1* and *case 2*, respectively.

removing the most fluid.

5.3.4 Experimentation

Implementation Details

The simulation used a time step of $\Delta t = 0.01$s, the maximum number of particles was limited to $N = 2000$, and 200 simulation steps were taken before the robot can take its first action. The MPC controller used a horizon $h = 10$, and the initial suction point selection uses a look-ahead window of $m = 100$. A total of 10 samples were used for initial point selection. The target height was set to $y_{goal} = 10$cm, suction strength $\mathcal{K} = 100$, $\sigma_x = \sigma_z = \sqrt{0.5}$. The maximum change in end-effector position was also limited to 0.5mm every step to be realistic. Finally, the gradients used a learning rate of 0.1 with normalized gradients such that their components have a maximum magnitude of 1.

Experiment Setup

Simulated Scene: Two simulated scenes, denoted as *case 1* and *case 2*, used to evaluate the proposed control strategy are shown in Fig. 5.2. These scenes are motivated by real surgical cases where a surgeon error in a live thyroidectomy caused bleeding to occur. Four hand-crafted control policies for controlling the suction nozzle were also developed to compare against the proposed MPC method. They are:

1. *Fixed emission*: stays at the emission point

2. *Fixed end*: stays at the end point where the fluid could flow too

3. *Fixed middle*: stays at the middle of the fluid flow

4. *End-to-emit*: moves from the end to emission point at a constant rate.

Good suction performance should have little blood at the end of the simulation. Another metric was the time it takes to reduce the amount of blood in the cavity by 50% and 90%, computed as

$$\tau_{50\%} = \min \quad t - t_0 \text{ s.t. } f(t) <= 50\% \tag{5.8}$$

where $f(t)$ is the suction curve as a function of time, t_0 is the time at which suction begins, and t_f is the time at which the simulation ends. The time for 90% reduction was computed similarly. Note that the percent reduction time only makes sense if the suction policy actually reduced the amount of blood by the targeted amount. Otherwise the percent reduction time was not computed.

dVRK: In order to validate the suction trajectories generated using our algorithm, we repeated the simulation experiments in a cavity made out of silicone rubber. Water with red dye was used to emulate blood flow, which was manually injected into the cavity with a syringe similar to the previously described simualted scenes. A Patient Side Manipulator (PSM) from the dVRK [65] was fitted with a EndoWrist Suction/Irrigator tool for suctioning. The cavity and PSM arm were converted into a unified camera frame defined by dVRK's stereo endoscope so that trajectories could be defined in a common reference frame. To register the cavity, an Aruco marker was attached to it and the pose was solved for using the Aruco library [40]. Meanwhile the end-effector of the suction tool was controlled using the Lumped Error controller described in 5.1.

The amount of blood in the cavity was estimated using images from the dVRK's stereo endoscope. Color segmentation to detect the blood was done by manually setting thresholds in the HSV color space. We assume that the concentration of red dye in the

Table 5.2. Residual of blood after trajectory is executed and percent reduction time (in simulation timesteps) for *case 1*.

	Our method	Fixed: emission	Fixed: end	Fixed: middle	End-to-emit trajectory
Residual	5.5%	28.0%	64.5%	27.0%	**2.5%**
$\tau_{50\%}$	**36**	87	–	23	90
$\tau_{90\%}$	**162**	–	–	–	255

Table 5.3. Residual of blood after trajectory is executed and percent reduction time (in simulation timesteps) for *case 2*.

	Our method	Fixed: emission	Fixed: end (avg)	End-to-emit trajectory (avg)
Residual	**1.5%**	2.0%	65.0%	3.8%
$\tau_{50\%}$	21	**18**	–	87.5
$\tau_{90\%}$	**102**	**102**	–	172.5

water was uniform, which means the attenuation of light through the fluid is proportional to depth by the Beer-Lambert Law [58]. Calibration was performed by taking images of the cavity filled with different volumes of blood and fitting a relationship between the depth of blood and the pixel values in each of the three channels. The area covered by the blood pixels and the calibrated depth curve in HSV space are used to estimate the volume of blood at each image frame. This model is a simple approximation of the volume of blood, which can only be truly known if one observed the underlying tissue topology before being filled with blood. Naturally, MPC's iterative approach continuously corrects for this volume assumption as more of the tissue topology is revealed, so this first-order approximation is reasonable. The estimated volume is used to generate a suction curve plot. The MPC approach is compared against the same comparisions in the simulated scene and three trials were repeated per policy. In each trial, the cavity was pre-filled with blood, and more blood was continuously injected when suction started to emulate the conditions in the simulated experiments.

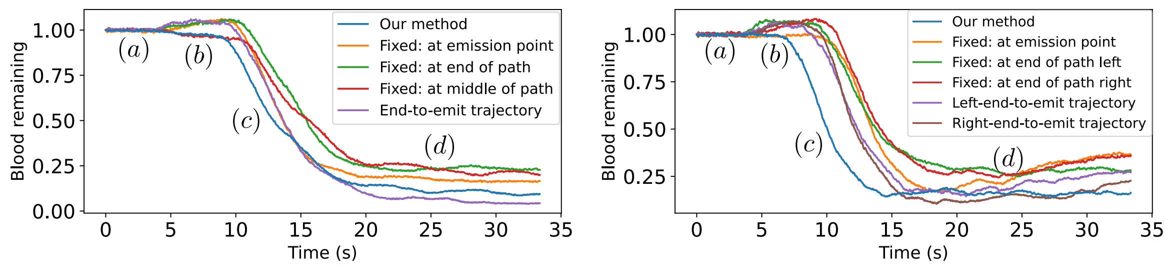


Figure 5.5. The left and right plots show suction results from real world experiments for *case 1* and *case 2*, respectively, normalized by the initial volume.

Results

The results from the experiments are shown in Tables 5.2 and 5.3. As is evident by the results, the proposed method performed as good as, if not better than, the best hand-crafted policies. In *case 1*, the blood flowed around the obstacle from right to left, hence making the best hand-crafted policy, end-to-emit trajectory, slow but complete in suctioning the blood. However, in *case 2* this end-to-emit strategy failed since the blood emits closer to the middle of the cavity hence diverting in two directions. Meanwhile the opposite behavior occurred for the fixed emission control, where it failed in *case 1* but was the best strategy in the *case 2*. This means the hand-crafted policies had limited success only in a single scenario the proposed method successfully generalized to both cases.

The suction curves for dVRK experiment with the silicone rubber cavity are shown in Figure 5.5. We analyze the suction curves by roughly splitting each trial into 4 distinct stages: (a) where the tool moved to initial position without suctioning (b) when suction and injection began (c) where suction engaged with the blood for the same time; (d) where suction and injection had been engaged for a long time and the tool reaches the end of the trajectory. In stage (b), the amount of blood increased as suction started for most of the hand-crafted policies. This means the rate of suction was slower than the rate of blood injection. Meanwhile the proposed method was able to consistently circumvent this, hence being more efficient. In stage (c), suction and injection reached an equilibrium

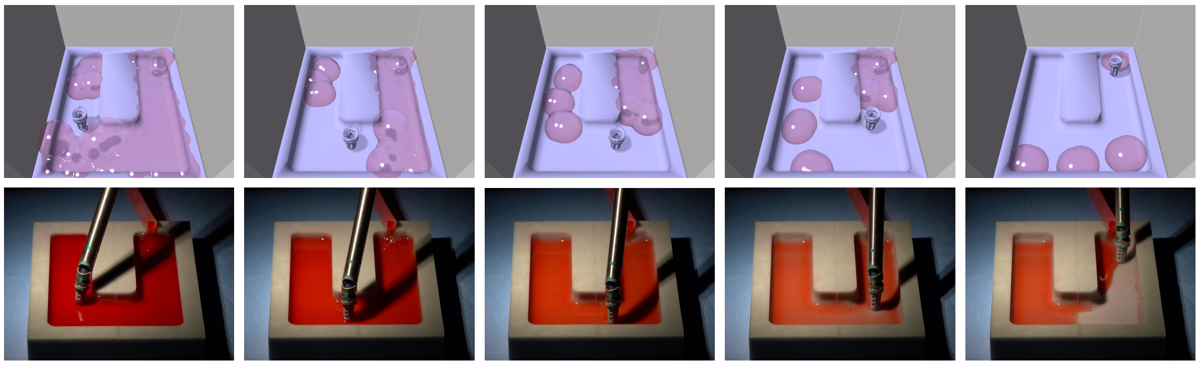


Figure 5.6. Comparison between the simulated scene (top) and experiments with the real cavity (bottom) for *case 1* using our method. Only small amounts of residual blood is left over in the real world due to unaccounted factors such as surface tension and adhesion.

where the percentage of remaining blood decreased at a steady rate. This rate of decrease was seen to be roughly constant for all policies. However, the proposed method was able to reach this state faster due to being more efficient in stage (b).

5.3.5 Discussion

In this work we presented a method for incorporating differentiable fluid modeling into autonomous surgical robotics. We applied this method to the surgical sub-task of controlling suction to clear the surgical field of blood during a vessel rupture. The gradients from the PBF model were used in an MPC framework. A discrepancy between the simulation-and-real world was seen at stage (d) where in the real world a thin remaining layer of blood is left over. An example for *case 1* is shown in Figure 5.6. In the simulation, the blood particles was able to flow without sticking to the face of the cavity, while a patch of blood was stuck in the left half of the cavity in the real experiment due to surface adhesion. Nevertheless, our method still results in a low percentage of remaining blood at the end of the task.

5.4 Acknowledgements

Chapter 5, in part, is a reprint of material from F. Richter, S. Shen, F. Liu, J. Huang, E. K. Funk, R. K. Orosco, M. C. Yip “Autonomous Robotic Suction to Clear the Surgical Field for Hemostasis using Image-based Blood Flow Detection,” in *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1383-1390, 2021. The dissertation author is the primary author of this paper. Chapter 5, in part, is also a reprint of material from J. Huang, F. Liu, F. Richter, M. C. Yip “Model-predictive control of blood suction for surgical hemostasis using differentiable fluid simulations,” in *IEEE International Conference on Robotics and Automation*, pp. 12380-12386, 2021.

Chapter 6

Conclusion

This dissertation presents the first completely automated solution for clearing the surgical field from blood after a vessel rupture during surgery. The solution provides the perception, trajectory generation, and control strategy required for the task of clearing blood. Thus, it is the first step toward automation of hemostatic management as a crucial surgical task, and it represents a critical contribution to the broader robotic and computer vision community.

In Chapter 2, we describe the challenge of localizing a robotic tool based on visual observations that only show a part of the kinematic chain (e.g., surgical robotic tools), and we assess the associated uncertainties in the base-to-camera transform and joint angle measurements. A smaller set of parameters, which we coined as Lumped Error, was derived and shown to compensate for all the described uncertainties. The Lumped Error is mathematically equivalent to a popular instrument tracking method [132], which provides critical insights into its functionality. Experimental tracking of the Lumped Error was shown to efficiently track robotic tools, and it was even successfully extended to eye-in-hand configurations. Through this extension, we successfully tracked for the first time a surgical robotic tool with a moving endoscope that contained a total of 10 DoF and a gripper joint.

A surgical perception framework, SuPer, is proposed in Chapter 3, which describes how SuPer can localize the surgical tool and track the deformable tissue simultaneously. We further improved its robustness and accuracy with deep learning approaches for depth estimation and surgical tool detection. SuPer was evaluated experimentally on a dVRK, which demonstrated its ability to track a surgical scene during manipulation tasks where the instrument caused occlusions and significant tissue deformations. By using the perceived environment as feedback, the controllers applied to the surgical tool have the capability to accomplish tasks in unstructured, deforming surgical environments.

The perception of blood from vessel ruptures is covered in Chapter 4. To ensure robustness against blood stains, the segmentation algorithm relies on temporal information for detection. Our novel blood flow detection and tracking algorithm provides critical perceptual capabilities and offers a unique, probabilistic solution to tracking liquids over 3D cavities and channels under noisy and harsh visibility conditions. We also developed a liquid reconstruction technique that is based on 2D image detections. We limited the scope to 2D surface detections because the color of a liquid is too variable due to reflections and refractions. Our experiments highlight the generalizability of our approach through the wide range of liquids (simulated, water, and milk) and cameras (narrow & wide field of view and 15, 24 & 30 fps).

Finally, Chapter 5 covers newly developed control and motion planning algorithms for a robotic suction tool to autonomously clear the surgical field of blood automating the task of hemostatic management. Specifically, we describe a controller that directs surgical robotic tools to goal positions within the camera frame using the Lumped Error as feedback. A blood perception technique is employed to inform a control function to generate an end-to-emit trajectory when blood has been detected in the surgical field. This technique, which needs to be robust against imperfect blood region estimation, relies on clearance rewards to maximize the blood volume removed by the suction tool. To optimize the motion plan of robotic suction tools, a differentiable fluid simulation function was developed. The gradients from the fluid simulation were used in an MPC framework

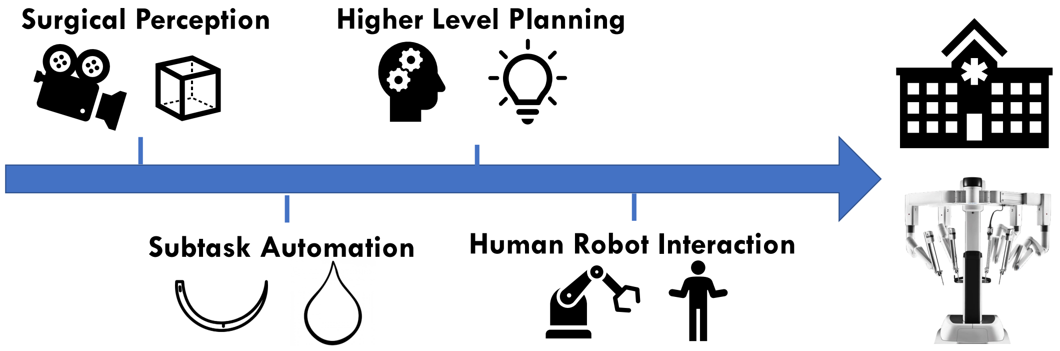


Figure 6.1. While this dissertation advances autonomy with novel automation techniques for hemostatic management, significant advances in perception, subtask automation, higher-level planning, and human-robot interactions are still required to bring autonomy into operating rooms.

to direct the robotic suction tool to clear as much blood as quickly as possible. The resulting real-world trajectories led to low percentages of remaining blood despite differences in the fluid parameters between the real and simulated scenes, such as surface tension and adhesion of the fluid and suction strength.

While the advancement in surgical task autonomy presented in this dissertation is a big step, there are still a significant number of research developments required to bring autonomy into operating rooms, as shown in Fig. 6.1. One of the most critical research categories is in surgical perception, such as the content of Chapter 4 for blood suction. Previous examples include the application of real-to-sim techniques to measure tissue tension for resection [84], suture needle tracking for autonomous suturing [15], and localizing key anatomy [29]. However, these surgical perception techniques still require refinements for their usage in surgery where the lighting conditions, occlusions, and viewing angles contribute to an environment that is much less favorable for automation than the structured laboratory environments in which these techniques are being tested. The usability of automation for certain surgical tasks also needs to be expanded upon. For instance, previously developed surgical task automation techniques for suturing [146] and gauze cutting [158] cannot be applied to the broad range of topology found in surgery.

A motivating example of complex topology is seen in vascular anastomosis. Current state-of-the-art suturing techniques are unable to handle the deformable geometry of blood vessels to perform a vascular anastomosis. Furthermore, surgical task automation approaches should be robust against an incorrect or incomplete perception of the surgical environment, which is a limitation of the approaches proposed in Chapter 5.

Higher-level planning for surgery also needs to be developed to deploy autonomy in surgery. Knowing and identifying the steps during surgery is crucial for inferring the next surgical task. However, state-of-the-art results for estimating surgical steps are not accurate enough for autonomy, and small dataset sizes are cited as the main bottleneck for improving performance [127, 61]. The autonomous agents are also anticipated to work alongside surgeons [171, 48]. Hence, developments in human-robot interactions will be required. Current human-robot interactions are limited to supervision approaches in surgical automation [147] and are not suitable to handle scenarios where a surgeon works simultaneously with the autonomous agent. Furthermore, the autonomous agents need techniques to identify their own confidence in executing the surgical tasks and effectively relay that information to a supervising surgeon to ensure patient safety.

Appendix A

Additional Material for Surgical Tool Tracking

A.1 Camera Projection of a Cylinder

The camera projection of a cylinder is an adaptation of previous work by Chaumette [13]. A cylinder is described by three parameters: a radius $r \in \mathbb{R}$, a directional vector $\mathbf{d}^j \in \mathbb{R}^3$ of its center axis, and a position along its center axis $\mathbf{p}_0^j \in \mathbb{R}^3$. Let $\mathbf{d}^j$ and $\mathbf{p}_0^j$ be defined in joint frame j, which is the insertion shaft, and $||\mathbf{d}^j|| = 1$. Using (2.18) or (2.25) for the stationary endoscope or robotic endoscope cases respectively, $\mathbf{d}^j$ and $\mathbf{p}_0^j$ are transformed to the camera frame and denoted as $\mathbf{d}^c(\mathbf{w}_t, \mathbf{b}_t, \mathbf{e}_t) = [a^c, b^c, c^c]^\top$ and $\mathbf{p}_0^c(\mathbf{w}_t, \mathbf{b}_t, \mathbf{e}_t) = [x_0^c, y_0^c, z_0^c]^\top$ respectively. Note that $(\mathbf{w}_t, \mathbf{b}_t)$ should be replaced with $(\mathbf{w}_t^l, \mathbf{b}_t^l)$ in the robotic endoscope case. The center axis of the cylinder in the camera frame can be described as:

$$\mathbf{p}_a^c = \mathbf{p}_0^c(\mathbf{w}_t, \mathbf{b}_t, \mathbf{e}_t) + \lambda \mathbf{d}^c(\mathbf{w}_t, \mathbf{b}_t, \mathbf{e}_t) \tag{A.1}$$

where $\lambda \in \mathbb{R}$. The cross section of a cylinder that is normal to the center axis can be described as the intersection between the surface of a sphere with radius r centered along $\mathbf{p}_a^c$ and a plane with normal $\mathbf{d}^c$ that contains the point $\mathbf{p}_a^c$. This intersection is described as:

$$\begin{cases} (\mathbf{p}_c^c - \mathbf{p}_a^c)^\top(\mathbf{p}_c^c - \mathbf{p}_a^c) - r^2 = 0 \\ \mathbf{d}^c(\mathbf{w}_t, \mathbf{b}_t, \mathbf{e}_t)^\top(\mathbf{p}_c^c - \mathbf{p}_a^c) = 0 \end{cases} \tag{A.2}$$

where $\mathbf{p}_c^c \in \mathbb{R}^3$ is a point on the perimeter of the circle from the cross section of the cylinder in the camera frame.

By combining (A.1) and (A.2), an expression for the surface of a cylinder can be derived. The resulting expression of the cylinder is:

$$\big(\mathbf{p}_s^c - \mathbf{p}_0^c(\mathbf{w}_t, \mathbf{b}_t, \mathbf{e}_t)\big)^\top \big(\mathbf{p}_s^c - \mathbf{p}_0^c(\mathbf{w}_t, \mathbf{b}_t, \mathbf{e}_t)\big) - \Big(\mathbf{d}^c(\mathbf{w}_t, \mathbf{b}_t, \mathbf{e}_t)^\top \big(\mathbf{p}_s^c - \mathbf{p}_0^c(\mathbf{w}_t, \mathbf{b}_t, \mathbf{e}_t)\big)\Big)^2 - r^2 = 0 \tag{A.3}$$

where $\mathbf{p}_s^c = [x_s^c, y_s^c, z_s^c]^\top$ is a point on the surface of the cylinder in the camera frame.

Without loss of generality, let (X, Y) be the projected pixel coordinates of the cylinder to a unit camera using the pin-hole model. This can be converted to the (u, v) pixel location on a different camera by setting:

$$X = \frac{u - c_u}{f_x} \qquad\qquad Y = \frac{v - c_v}{f_y} \tag{A.4}$$

where f_x, f_y and c_u, c_v are the focal lengths and principal point in pixel units respectively from the camera intrinsic matrix $\mathbf{K}$.

Applying the camera pin-hole model to the surface of the cylinder in the camera frame results in a quadratic:

$$A + \frac{1}{z}B + \frac{1}{z^2}C = 0 \tag{A.5}$$

where

$$\begin{aligned} A &= X^2 + Y^2 + 1 - (a^c X + b^c Y + c^c)^2 \\ B &= -2\Big((x_0^c - a^c\nu)X + (y_0^c - b^c\nu)Y + z_0^c - c^c\nu\Big) \\ C &= \left(x_0^c\right)^2 + \left(y_0^c\right)^2 + \left(z_0^c\right)^2 - \nu^2 - r^2 \end{aligned} \tag{A.6}$$

and

$$\nu = a^c x_0^c + b^c y_0^c + c^c z_0^c \tag{A.7}$$

The quadratic expression with respect to the depth occurs because there can be at most two solutions to the depth for each (X, Y) when the cylinder is projected onto an image plane. One solution is the visible side of the cylinder, and the other is the obstructed side of the cylinder. The case of a single solution to depth would occur only at the two edges of the projected cylinder. This can be enforced by setting the determinant of the quadratic to zero ($B^2 - 4AC = 0$) resulting in:

$$\left(\frac{Br}{-2\sqrt{C}} - \alpha X - \beta Y - \kappa\right)\left(\frac{Br}{-2\sqrt{C}} + \alpha X + \beta Y + \kappa\right) = 0 \tag{A.8}$$

where

$$\alpha = c^c y_0^c - b^c z_0^c \qquad \beta = a^c z_0^c - c^c x_0^c \qquad \kappa = b^c x_0^c - a^c y_0^c \tag{A.9}$$

after simplification.

Therefore, it is evident that the two edges from the projection of the cylinder result in two lines:

$$\left(\frac{r(x_0^c - a^c\nu)}{\sqrt{C}} - \alpha\right)X + \left(\frac{r(y_0^c - b^c\nu)}{\sqrt{C}} - \beta\right)Y + \left(\frac{r(z_0^c - c^c\nu)}{\sqrt{C}} - \kappa\right) = 0 \tag{A.10}$$

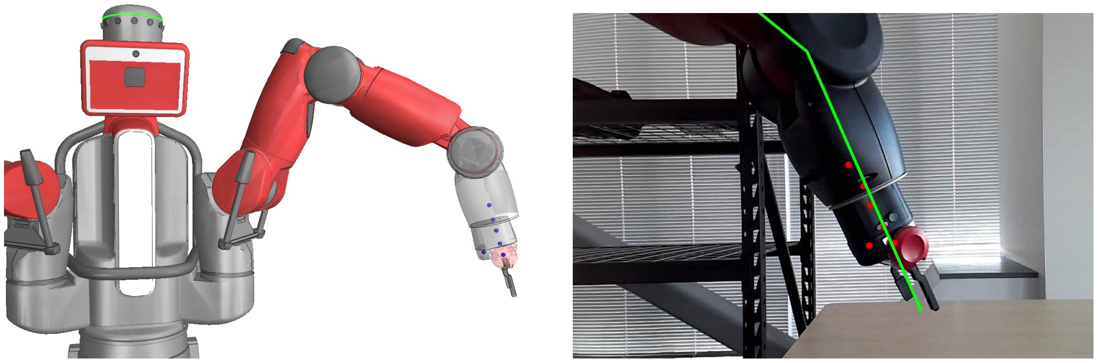


Figure A.1. A DNN was trained in simulation to detect the keypoints, highlighted in blue in the left figure, on the Baxter robot. An example of the detections and the tracked robotic arm from the Baxter experiment is shown in red and green respectively on the right figure.

and

$$\left(\frac{r(x_0^c - a^c\nu)}{\sqrt{C}} + \alpha\right)X + \left(\frac{r(y_0^c - b^c\nu)}{\sqrt{C}} + \beta\right)Y + \left(\frac{r(z_0^c - c^c\nu)}{\sqrt{C}} + \kappa\right) = 0 \qquad \text{(A.11)}$$

Through simple arithmetic and (A.4), both of these edges can be converted to the normal form described in (2.30) for any camera. In the normal form, let the resulting two edges be parameterized by $\big(\hat{\rho}_1(\mathbf{w}_t, \mathbf{b}_t, \mathbf{e}_t), \hat{\phi}_1(\mathbf{w}_t,\mathbf{b}_t, \mathbf{e}_t)\big)$ and $\big(\hat{\rho}_2(\mathbf{w}_t, \mathbf{b}_t, \mathbf{e}_t), \hat{\phi}_2(\mathbf{w}_t, \mathbf{b}_t, \mathbf{e}_t)\big)$.

A.2 Experimentation on Baxter Robot

Implementation Details

The features used to update the particle filter for this experiment were detected using a Deep Neural Network (DNN) rather than markers as used in the previous experiments. This was done to show the flexibility of the presented particle filter with regards to the features used to update it. The DNN choosen was DeepLabCut [100], and it was trained to detect and optimize feature points in simulation using the same method in [89]. The resulting feature points which were detected by the DNN are shown in Fig. A.1. The DeepLabCut detections also provide direct associations for the feature points,

A_m, and a confidence value, $\eta_t^k \in [0,1]$, for each detected feature k. To integrate this with the Lumped Error tracking, the point feature observation model in (2.33) was modified to (3.11) which also removes the association step in line 18 from Algorithm 1. It is important to include the DNN's confidence, η_t^k, in the model because sometimes the detections can be poor and the corresponding update needs to be weighted lower. The parameters for the particle filter are set as follows:

- $\mathbf{a}_{\hat{\mathbf{e}}} = \begin{bmatrix} 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 \end{bmatrix}$
- $\mathbf{\Sigma}_{\hat{\mathbf{e}},t} = \text{diag}\left(\begin{bmatrix} 0.001 & 0.001 & 0.001 & 0.001 & 0.001 & 0.001 & 0.001 \end{bmatrix}\right)$
- $\mathbf{\Sigma}_{\mathbf{w},\mathbf{b},t} = \text{diag}\left(\begin{bmatrix} \mathbf{\Sigma}_{\mathbf{w},t} & \mathbf{\Sigma}_{\mathbf{b},t} \end{bmatrix}\right)$
- $\mathbf{\Sigma}_{\mathbf{w},t} = \text{diag}\left(\begin{bmatrix} 0.001 & 0.001 & 0.001 \end{bmatrix}\right)$
- $\mathbf{\Sigma}_{\mathbf{b},t} = \text{diag}\left(\begin{bmatrix} 0.25 & 0.25 & 0.25 \end{bmatrix}\right)$
- $\mathbf{\Sigma}_{\mathbf{w},\mathbf{b},0} = 10(\mathbf{\Sigma}_{\mathbf{w},\mathbf{b},t})$
- $\gamma_m = 5$

The number of particles is set to 200.

Dataset

A 77 second video segment was recorded of a 7 DoF arm from the Baxter robot with corresponding joint reading data. The first joint link consistently visible in the image frames is after the $n_b = 6$ joint, and the end-effector moved a total distance of 5.25m during the segment. The video was captured on Microsoft's Azure Kinect camera which has and RGB camera and a depth camera. In this experiment, the particle filter which tracked this robotic arm only used the mono-RGB camera data. Meanwhile the depth images were used to evaluate performance of tracking the robotic tool.

To evaluate tracking performance, the depth images were compared against the reconstructed robotic arm using the tracked parameters. The reconstructed scene was rendered using a virtual camera and a Baxter robot model in V-REP [138]. After every image used to update the particle filter, the virtual camera captured a depth image of the reconstructed scene. The tracking error was defined as the relative transform, $\mathbf{T}(\mathbf{w}^{\epsilon}, \mathbf{b}^{\epsilon}) \in SE(3)$, between the rendered point cloud from the virtual camera, $\mathbf{R}$, and the corresponding point cloud from Kinect Azure's depth camera, $\mathbf{G}$. This relative transform is calculated by minimizing

$$\sum_{(\mathbf{r},\mathbf{g}) \in \mathcal{K}} ||\bar{\mathbf{r}} - \mathbf{T}(\mathbf{w}^{\epsilon}, \mathbf{b}^{\epsilon})\bar{\mathbf{g}}|| \tag{A.12}$$

where $\mathbf{r} \in \mathbf{R}$, $\mathbf{g} \in \mathbf{G}$, and $\mathcal{K}$ is the correspondence set between $\mathbf{R}$ and $\mathbf{G}$. The optimization was solved using Open3D's [180] implementation of the Iterative Closest Point algorithm [6]. To filter out poorly converged values, only the results where the amount of corresponded points relative to the total rendered points, $|\mathcal{K}|/|\mathbf{R}|$, is greater than 0.7 were recorded. Similar to the dVRK experiments, the initial calibration, $\mathbf{T}^{c}_{b-}$, was computed using OpenCV's solvePnP [9] using the detected features and their associations on the first image frame.

For additional comparison, we ran OpenCV's solvePnP implementation [9] over the first 20 images of the dataset to solve for a static base-to-camera transform, $\mathbf{T}^{c}_{b}$. The 2D detections are the same used by the particle filter. Their corresponding 3D positions in the base frame of the Baxter robot are generated using forward kinematics with the joint angle readings $\tilde{q}^{i}_{t}$. The resulting $\mathbf{T}^{c}_{b}$ and the joint angle readings are used to generate a rendered point cloud $\mathbf{R}$ in V-REP [32], and the same error metric as previously described is computed.

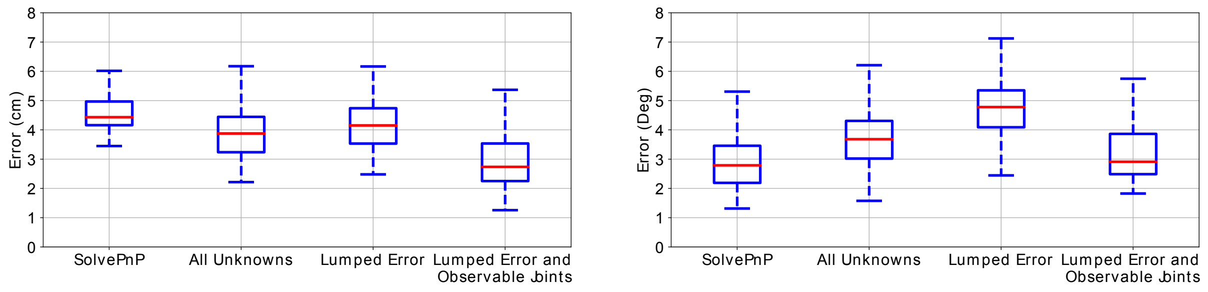


Figure A.2. Distribution of position and orientation errors when calibrating for the base-to-camera transform alone with solvePnP and various particle filter configurations compensating for errors in the base-to-camera transform and joint angles from the Baxter robot experiment.

Results

The distribution of translational errors, $||\mathbf{b}^\epsilon||$, and orientation errors, $||\mathbf{w}^\epsilon||$, for the three different particle filter configurations are plotted in Fig. A.2. Out of the active tracking methods, the Lumped Error parameter reduction technique with the observable joints is the most effective. We believe this is since the kinematic links on the Baxter are much larger hence making the simplification from (2.19) no longer valid. Meanwhile, solvePnP for the static base-to-camera transform performs similar to Lumped Error and Observable joints in orientation error, but performs significantly worse in positional error.

Appendix B

Additional Material for Liquid Reconstruction

B.1 Details for Comparison Study

B.1.1 DSS Rendering

For one of our experimental comparisons, we used Differentiable Surface Splatting (DSS) [170] to minimize the image loss. DSS renders each point as a circle, which projects to an ellipse, where the circle's normal is the surface normal. Surface normals for a particle-represented liquid are computed using the *color field* [105] which is

$$c(\mathbf{x}^i) = \sum_{j=1}^{N} \frac{1}{\rho_i(\mathbf{x})} W(||\mathbf{x}^i - \mathbf{x}^j||, r) \tag{B.1}$$

at $\mathbf{x}^i \in \mathbb{R}^3$. The surface normals should point outwards from the reconstructed liquid which results in a negative change in color field. Therefore the normal is set to:

$$\mathbf{n}^i = -\left(\frac{\partial c(\mathbf{x}^i)}{\partial \mathbf{x}^i}\right) \Bigg/ \left|\left|\frac{\partial c(\mathbf{x}^i)}{\partial \mathbf{x}^i}\right|\right| \tag{B.2}$$

for particle $\mathbf{x}^i$. To compute the liquid volume color's gradient, the following expression is used:

$$\frac{\partial c(\mathbf{x}^i)}{\partial \mathbf{x}^i} = \sum_{j=1}^{N} \frac{1}{\rho_i} \frac{\partial W(||\mathbf{x}^i - \mathbf{x}^j||, r)}{\partial ||\mathbf{x}^i - \mathbf{x}^j||} \left(\frac{\mathbf{x}^i - \mathbf{x}^j}{||\mathbf{x}^k - \mathbf{x}^j||} \right) \tag{B.3}$$

after applying the chain rule to (B.1).

Laplacian smoothing is applied for more consistent normals, similar to [173], by averaging the particle positions the color field is being evaluated about in (B.1). The particle averages are computed as

$$\overline{\mathbf{x}}^j = (1 - \lambda_l)\mathbf{x}^j + \lambda_l \frac{\sum\limits_{k=1}^{N} \mathbf{x}^k W(||\mathbf{x}^j - \mathbf{x}^k||, r)}{\sum\limits_{k=1}^{N} W(||\mathbf{x}^j - \mathbf{x}^k||, r)} \tag{B.4}$$

where λ_l is the Laplacian average weight and $\overline{\mathbf{x}}^j$ replaces $\mathbf{x}^j$ in (B.1). In low particle count situations, the normal computation in (B.2) can produce undesirable effects such as artifacts on the edges of the liquids. To account for this, the evaluation of points $\mathbf{x}^i$ in (B.2) are given a small offset towards the virtual camera which will eventually render the surface. The offset is computed as:

$$\Delta\overline{\mathbf{x}}^i = \lambda_c \frac{\mathbf{c} - \overline{\mathbf{x}}^i}{||\mathbf{c} - \overline{\mathbf{x}}^i||} \tag{B.5}$$

where $\mathbf{c} \in \mathbb{R}^3$ is the position of the virtual camera, λ_c is the amount of the offset, and $\Delta\overline{\mathbf{x}}^i$ is added to $\mathbf{x}^i$ in (B.2).

The particle position, normal pairs, $\{\mathbf{x}^i, \mathbf{n}^i\}_{i=1}^N$, are directly fed into the DSS which renders each point, $\mathbf{x}^i$, as a circle whose plane is tangent to its normal, $\mathbf{n}^i$. The circles are projected to ellipses, denoted as $\mathcal{E}(\mathbf{x}^i, \mathbf{n}^i)$, and averaged with their neighboring projected circles, hence being called Elliptical Weighted Averaging (EWA). In the problem formulation for this work, we assume only knowledge of an observed visibility mask ▯.

Therefore, we simplify the rendering by not conducting the EWA and only render a surface mask from the projected ellipses. Written mathematically, the masked image at pixel $[u,v]^\top$ from a single particle and normal pair is: DSS computes each

$$h_{u,v}(\mathbf{x}^i,\mathbf{n}^i) = \begin{cases} 1 & \text{if } [u,v]^\top \in \mathcal{E}(\mathbf{x}^i,\mathbf{n}^i) \\ 0 & \text{if } \mathbf{x}^i \text{ is occluded} \\ 0 & \text{otherwise} \end{cases} \tag{B.6}$$

The rendered surface is evaluated as a summation of all the masked images from (B.6):

$$\hat{\mathbb{I}}_{u,v}(\mathbf{x}) = \eta_i \sum_{i=1}^{N_s} h_{u,v}(\mathbf{x}^i,\mathbf{n}^i) \tag{B.7}$$

where η_i normalizes the pixel value. Finally, gradients of the rendered liquid surface with respect to particle positions are computed using the approximation presented by Yifan et al. to minimize the image loss [170]. The normal smoothing values are set to $\lambda_l = 0.2$ and $\lambda_c = 0.2r$, and the original proposed kernels are used for (B.1) [105] and (B.4) [173]. The gradient step size and its threshold for detecting a local-minima are set to $\alpha_{\mathbb{I}} = 10^{-4}$ and $\lambda_s = 0.2$ respectively.

B.1.2 Schenck and Fox Constraints

Schenck and Fox previously proposed liquid position constraints that represent: pressure, cohesion, and surface tension [144]. These constraints replaced the proposed density constraint from (4.14) for comparison in our experiments. This is done by replacing $\Delta\mathbf{x}_\rho$ to solve (4.14) in lines 7 and 8 in Algorithm 3 with:

$$\Delta\mathbf{x}_p + \alpha_c \Delta\mathbf{x}_c + \alpha_s \Delta\mathbf{x}_s \tag{B.8}$$

where $\Delta\mathbf{x}_p, \Delta\mathbf{x}_c, \Delta\mathbf{x}_s$ solve the pressure, cohesion, and surface tension constraints respectively and α_c, α_s are the cohesion and surface tension weights respectively. Refer to the original paper for exact expressions to the constraint solutions [144]. The cohesion and surface tension weights are optimized for in the original work to conduct real-to-sim registration. However, this cannot be done with our problem setup because it requires prior information on the amount of liquid volume there is (i.e. how many particles there are). Therefore, instead the weights are preset to $\alpha_c = 0.05$ and $\alpha_s = 0$ (the surface tension constraint only yielded unstable behavior so it was turned off).

B.1.3 Source Estimation

A simple single, static source estimation technique is developed to highlight how the proposed method can be extended. Let $\hat{\mathbf{s}}_t \in \mathbb{R}^3$ be the estimated liquid source location in the camera frame at time t and particles are inserted according to an estimated flow rate of $\hat{f}_t$ particles per timestep at the source location. Note that no velocity prediction is conducted for the inserted particles as there is no initial velocity. This reduces the number of parameters to estimate to $\hat{\mathbf{s}}_t$ and $\hat{f}_t$.

To update the liquid source location, $\hat{\mathbf{s}}_t$, we compare the source particle locations after completing the optimization in (4.10) against their pre-optimized location. Let the initial and optimized particle locations emitted from the source be denoted as $\mathbf{x}_s^n \in \mathbb{R}^3$ for $n = 1, \dots, \hat{f}_t$ and $\mathbf{x}_s^{n*} \in \mathbb{R}^3$ for $n = 1, \dots, \hat{f}_t$ respectively. Then the update rule given to the source location is:

$$\hat{\mathbf{s}}_{t+1} = \hat{\mathbf{s}}_t + \frac{\alpha_{\hat{\mathbf{s}}}}{\hat{f}_t} \sum_{n=1}^{\hat{f}_t} (\mathbf{x}_s^{i*} - \mathbf{x}_s^i) \tag{B.9}$$

where $\alpha_{\hat{\mathbf{s}}}$ is adjusted according to:

$$\alpha_{\hat{\mathbf{s}}} = 1 / \sum_{i=1}^{t-1} \hat{f}_i \tag{B.10}$$

so the source becomes less adjusted as more particles have been inserted since the source

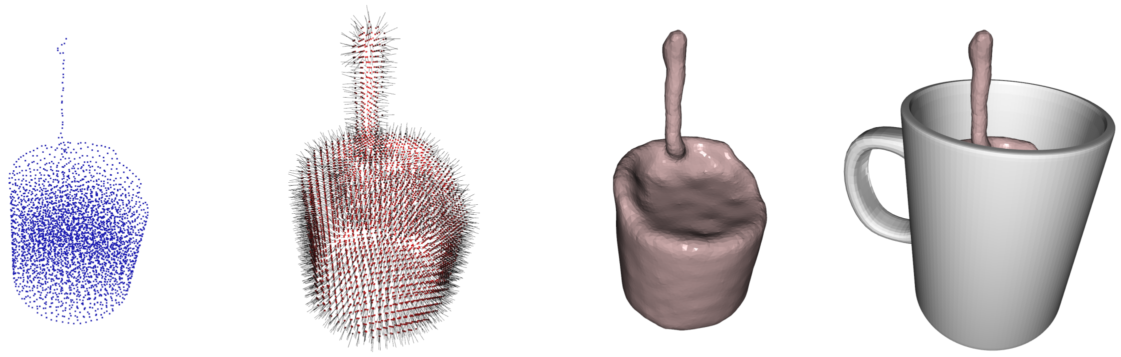


Figure B.1. From left to right, the sequence shows the steps for mesh generation from a reconstructed liquid represented by particles.

is assumed stationary.

The liquid source rate, $\hat{f}_t$, has an integer effect on the reconstruction, and we adjust it at every time step based on how many particles are duplicated or removed during the optimization of (4.10) after inputting the source particles for that timestep. The expression is:

$$\hat{f}_t = \alpha_{\hat{f}} \Delta N_t + \hat{f}_{t-1} - \lambda_f \tag{B.11}$$

where ΔN_t is the cumulative increase of particles (e.g. could be negative if particles are removed) at timestep t, $\alpha_{\hat{f}}$ adjusts the reaction rate to the insertion/removal of particles, and λ_f is a constant decay rate. Note that $\hat{f}_t$ is estimated as a non-integer value, however is applied as an integer by rounding (i.e. only an integer number of particles can be inserted per timestep). The decay rate, λ_f is used ensures stability by driving the flow rate to 0 when no new information from ΔN_t can be leveraged. The reaction rate and decay rates are set to $\alpha_f = 0.1$ and $\lambda_f = \alpha_{\hat{f}}/2$ respectively.

B.2 Mesh Generation from Liquid Particles

For visualization purposes, the reconstructed liquid can be converted to a surface mesh. A dense, uniformly spaced, grid of 3D points is generated. Surface points, $\mathbf{g}^k$, from the grid points are then selected by thresholding the gradient of the color field [105]:

$$\partial c(\mathbf{g})/\partial \mathbf{g}^k \geq \lambda_g \tag{B.12}$$

where the color field, $c(\cdot)$, is defined in (B.1) and λ_g is the threshold. The surface normals for each surface point is computed the same as (B.2). The collection of surface points and normals are then converted to a mesh using Open3D's implementation of [180] Poisson surface reconstruction [66]. Fig. B.1 shows an example of this process. The grid points, which the surface points are selected from, are spaced at 3mm, the gradient threshold, λ_g, is set to 0.5, and the depth for Poission surface reconstruction is set to 12. Note that figures of particles and mesh renderings in this paper are done with Open3D [180].

Bibliography

[1] Farshid Alambeigi, Zerui Wang, Yun-hui Liu, Russell H Taylor, and Mehran Armand. Toward semi-autonomous cryoablation of kidney tumors via model-independent deformable tissue manipulation technique. *Annals of Biomedical Engineering*, 46(10):1650–1662, 2018.

[2] Bradley Atcheson, Wolfgang Heidrich, and Ivo Ihrke. An evaluation of optical flow algorithms for background oriented schlieren imaging. *Experiments in Fluids*, 46(3):467–476, 2009.

[3] Bradley Atcheson, Ivo Ihrke, Wolfgang Heidrich, Art Tevs, Derek Bradley, Marcus Magnor, and Hans-Peter Seidel. Time-resolved 3d capture of non-stationary gas flows. *Transactions on Graphics*, 27(5):1–9, 2008.

[4] Simon Baker, Daniel Scharstein, JP Lewis, Stefan Roth, Michael J Black, and Richard Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92(1):1–31, 2011.

[5] Burak Benligiray, Cihan Topal, and Cuneyt Akinlar. Stag: A stable fiducial marker system. *Image and Vision Computing*, 89:158–169, 2019.

[6] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor Fusion IV: Control Paradigms and Data Structures*, volume 1611, pages 586–606. International Society for Optics and Photonics, 1992.

[7] Kenneth Bodin, Claude Lacoursiere, and Martin Servin. Constraint fluids. *Transactions on Visualization and Computer Graphics*, 18(3):516–526, 2011.

[8] David Bouget, Max Allan, Danail Stoyanov, and Pierre Jannin. Vision-based and marker-less surgical tool detection and tracking: a review of the literature. *Medical Image Analysis*, 35:633–654, 2017.

[9] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[10] Eduardo F Camacho and Carlos Bordons Alba. *Model Predictive Control*. Springer Science & Business Media, 2013.

[11] John Canny. A computational approach to edge detection. *Transactions on Pattern Analysis and Machine Intelligence*, 1986.

[12] Jia-Ren Chang and Yong-Sheng Chen. Pyramid stereo matching network. In *Conference on Computer Vision and Pattern Recognition*, pages 5410–5418. IEEE, 2018.

[13] François Chaumette. *La relation vision-commande: théorie et application à des tâches robotiques.* PhD thesis, L'Université de Rennes I, 1990.

[14] Chien-Chern Cheah, Masanori Hirano, Sadao Kawamura, and Suguru Arimoto. Approximate jacobian control for robots with uncertain kinematics and dynamics. *Transactions on Robotics and Automation*, 19(4):692–702, 2003.

[15] Zih-Yun Chiu, Albert Z Liao, Florian Richter, Bjorn Johnson, and Michael C Yip. Markerless suture needle 6d pose tracking with robust uncertainty estimation for autonomous minimally invasive robotic surgery. *arXiv preprint arXiv:2109.12722*, 2021.

[16] Zih-Yun Chiu, Florian Richter, Emily K Funk, Ryan K Orosco, and Michael C Yip. Bimanual regrasping for suture needles using reinforcement learning for rapid motion planning. In *International Conference on Robotics and Automation*, pages 7737–7743. IEEE, 2021.

[17] Changhyun Choi and Henrik I Christensen. Robust 3d visual tracking using particle filtering on the special euclidean group: A combined approach of keypoint and edge features. *The International Journal of Robotics Research*, 31(4):498–519, 2012.

[18] Cristina Garcia Cifuentes, Jan Issac, Manuel Wüthrich, Stefan Schaal, and Jeannette Bohg. Probabilistic articulated real-time tracking for robot manipulation. *Robotics and Automation Letters*, 2(2):577–584, 2016.

[19] Emanuele Colleoni, Sara Moccia, Xiaofei Du, Elena De Momi, and Danail Stoyanov. Deep learning based robotic tool detection and articulation estimation with spatio-temporal layers. *Robotics and Automation Letters*, 4(3):2714–2721, July 2019.

[20] Blender Online Community. *Blender - a 3D Modelling and Rendering Package.* Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018.

[21] S áB Dalziel, Graham O Hughes, and Bruce R Sutherland. Whole-field density measurements by 'synthetic schlieren'. *Experiments in Fluids*, 28(4):322–335, 2000.

[22] Claudia D'Ettorre, Andrea Mariani, Agostino Stilli, Pietro Valdastri, Anton Deguet, Peter Kazanzides, Russell H Taylor, Gregory S Fischer, Simon P DiMaio, Arianna Menciassi, and Danail Stoyanov. Accelerating surgical robotics research: Reviewing 10 years of research with the dvrk. *arXiv preprint arXiv:2104.09869*, 2021.

[23] Simon DiMaio, Mike Hanuschik, and Usha Kreaden. The da vinci surgical system. In *Surgical Robotics*, pages 199–217. Springer, 2011.

[24] Mingsong Dou, Sameh Khamis, Yury Degtyarev, Philip Davidson, Sean Ryan Fanello, Adarsh Kowdle, Sergio Orts Escolano, Christoph Rhemann, David Kim, Jonathan Taylor, Pushmeet Kohli, Vladimir Tankovich, and Shahram Izadi. Fusion4d: Real-time performance capture of challenging scenes. *Transactions on Graphics*, 35(4):1–13, 2016.

[25] A. Dutta, A. Gupta, and A. Zissermann. VGG image annotator (VIA). http://www.robots.ox.ac.uk/ vgg/software/via/, 2016. Version: 2.0.10, Accessed: 06/30/2018.

[26] Marie-Lena Eckert, Kiwon Um, and Nils Thuerey. Scalarflow: a large-scale volumetric data set of real-world scalar transport flows for computer animation and machine learning. *Transactions on Graphics*, 38(6):1–16, 2019.

[27] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Scandinavian Conference on Image Analysis*, pages 363–370. Springer, 2003.

[28] Irene Fassi and Giovanni Legnani. Hand to sensor calibration: A geometrical interpretation of the matrix equation ax= xb. *Journal of Robotic Systems*, 22(9):497–506, 2005.

[29] James M Ferguson, Leon Y Cai, Alexander Reed, Michael Siebold, Smita De, S Duke Herrell, and Robert J Webster. Toward image-guided partial nephrectomy with the da vinci robot: exploring surface acquisition methods for intraoperative re-registration. In *Medical imaging 2018: Image-guided procedures, robotic interventions, and modeling*, volume 10576, page 1057609. International Society for Optics and Photonics, 2018.

[30] Mark Fiala. Artag, a fiducial marker system using digital techniques. In *Conference on Computer Vision and Pattern Recognition*, volume 2, pages 590–596. IEEE, 2005.

[31] Paolo Fiorini. Automation and autonomy in robotic surgery. In *Robotic Surgery*, pages 237–255. Springer, 2021.

[32] Giuseppe Andrea Fontanelli, Mario Selvaggio, Marco Ferro, Fanny Ficuciello, Marilena Vendittelli, and Bruno Siciliano. A v-rep simulator for the da vinci research kit robotic platform. In *BioRob*, 2018.

[33] Bengt Fornberg. Generation of finite difference formulas on arbitrarily spaced grids. *Mathematics of Computation*, 51(184):699–706, 1988.

[34] Erik Franz, Barbara Solenthaler, and Nils Thuerey. Global transport for fluid reconstruction with learned self-supervision. In *Conference on Computer Vision and Pattern Recognition*, pages 1632–1642. IEEE, 2021.

[35] Yanan Fu, Mrinal Mandal, and Gencheng Guo. Bleeding region detection in wce images based on color features and neural network. In *International Midwest Symposium on Circuits and Systems*, pages 1–4. IEEE, 2011.

[36] Yanan Fu, Wei Zhang, Mrinal Mandal, and Max Q-H Meng. Computer-aided bleeding detection in wce video. *Journal of Biomedical and Health Informatics*, 18(2):636–642, 2013.

[37] W. Gao and R. Tedrake. Surfelwarp: Efficient non-volumetric single view dynamic reconstruction. In *Robotics: Science and System*, 2018.

[38] Xiao-Shan Gao, Xiao-Rong Hou, Jianliang Tang, and Hang-Fei Cheng. Complete solution classification for the perspective-three-point problem. *Transactions on Pattern Analysis and Machine Intelligence*, 25(8):930–943, 2003.

[39] Antonio Garcia-Ruiz, Michel Gagner, Jeffrey H Miller, Charles P Steiner, and Joseph F Hahn. Manual vs robotically assisted laparoscopic surgery in the performance of basic manipulation and suturing tasks. *Archives of Surgery*, 133(9):957–961, 1998.

[40] Sergio Garrido-Jurado, Rafael Muñoz-Salinas, Francisco José Madrid-Cuevas, and Manuel Jesús Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280–2292, 2014.

[41] Andreas Geiger, Martin Roser, and Raquel Urtasun. Efficient large-scale stereo matching. In *Asian Conference on Computer Vision*, pages 25–38. Springer, 2010.

[42] Robert A Gingold and Joseph J Monaghan. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society*, 181(3):375–389, 1977.

[43] Ian Grant. Particle image velocimetry: a review. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 211(1):55–76, 1997.

[44] Jinwei Gu, Shree K Nayar, Eitan Grinspun, Peter N Belhumeur, and Ravi Ramamoorthi. Compressive structured light for recovering inhomogeneous participating media. *Transactions on Pattern Analysis and Machine intelligence*, 35(3):1–1, 2012.

[45] Tatiana López Guevara, Nicholas K Taylor, Michael U Gutmann, Subramanian Ramamoorthy, and Kartic Subr. Adaptable pouring: Teaching robots not to spill using fast but approximate fluid simulation. In *Proceedings of the Conference on Robot Learning*, 2017.

[46] Mohammad Haghighipanah, Muneaki Miyasaka, Yangming Li, and Blake Hannaford. Unscented kalman filter and 3d vision to improve cable driven surgical robot joint angle estimation. In *International Conference on Robotics and Automation*, pages 4135–4142. IEEE, 2016.

[47] Ulrich Hagn, Rainer Konietschke, Andreas Tobergte, Mathias Nickl, Stefan Jörg, Bernhard Kübler, Georg Passig, Martin Gröger, Florian Fröhlich, Ulrich Seibold, Le Tien Luc, Alin Albu-Schäeffer, Alexander Nothhelfer, Franz Hacker, Markus Grebenstein, and Gerd Hirzinger. Dlr mirosurge: a versatile system for research in endoscopic telesurgery. *International Journal of Computer Assisted Radiology and Surgery*, 5(2):183–193, 2010.

[48] Tamás Haidegger. Autonomy for surgical robots: Concepts and paradigms. *Transactions on Medical Robotics and Bionics*, 1(2):65–76, 2019.

[49] Blake Hannaford, Jacob Rosen, Diana W Friedman, Hawkeye King, Phillip Roan, Lei Cheng, Daniel Glozman, Ji Ma, Sina Nia Kosari, and Lee White. Raven-ii: an open platform for surgical robotics research. *Transactions on Biomedical Engineering*, 60(4):954–959, 2012.

[50] Ran Hao, Orhan Özgüner, and M Cenk Çavuşoğlu. Vision-based surgical tool pose estimation for the da vinci® robotic surgical system. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1298–1305. IEEE, 2018.

[51] Tim Hawkins, Per Einarsson, and Paul Debevec. Acquisition of time-varying participating media. *Transactions on Graphics*, 24(3):812–815, 2005.

[52] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition*, pages 770–778. IEEE, 2016.

[53] H. Hirschmuller. Stereo processing by semiglobal matching and mutual information. *Transactions on Pattern Analysis and Machine Intelligence*, 30(2):328–341, Feb 2008.

[54] G Hubens, H Coveliers, L Balliu, M Ruppert, and W Vaneerdeweg. A performance study comparing manual and robotically assisted laparoscopic surgery using the da vinci system. *Surgical Endoscopy and other Interventional Techniques*, 17(10):1595–1599, 2003.

[55] Andrew J Hung, Jian Chen, Zhengping Che, Tanachat Nilanon, Anthony Jarc, Micha Titus, Paul J Oh, Inderbir S Gill, and Yan Liu. Utilizing machine learning and automated performance metrics to evaluate robot-assisted radical prostatectomy performance and predict outcomes. *Journal of Endourology*, 32(5):438–444, 2018.

[56] Andrew J Hung, Jian Chen, and Inderbir S Gill. Automated performance metrics and machine learning algorithms to measure surgeon performance and anticipate clinical outcomes in robotic surgery. *JAMA Surgery*, 153(8):770–771, 2018.

[57] Minho Hwang, Brijen Thananjeyan, Samuel Paradis, Daniel Seita, Jeffrey Ichnowski, Danyal Fer, Thomas Low, and Ken Goldberg. Efficiently calibrating cable-driven surgical robots with rgbd fiducial sensing and recurrent neural networks. *Robotics and Automation Letters*, 5(4):5937–5944, 2020.

[58] James D Ingle Jr and Stanley R Crouch. Spectrochemical analysis. 1988.

[59] Matthias Innmann, Michael Zollhöfer, Matthias Nießner, Christian Theobalt, and Marc Stamminger. Volumedeform: Real-time volumetric non-rigid reconstruction. In *European Conference on Computer Vision*, pages 362–379. Springer, 2016.

[60] Eldar Insafutdinov, Leonid Pishchulin, Bjoern Andres, Mykhaylo Andriluka, and Bernt Schiele. Deepercut: A deeper, stronger, and faster multi-person pose estimation model. In *European Conference on Computer Vision*, pages 34–50. Springer, 2016.

[61] Yueming Jin, Huaxia Li, Qi Dou, Hao Chen, Jing Qin, Chi-Wing Fu, and Pheng-Ann Heng. Multi-task recurrent convolutional network with correlation loss for surgical video analysis. *Medical Image Analysis*, 59:101572, 2020.

[62] Yun Sub Jung, Young Ho Kim, Dong Ha Lee, and Jong Hyo Kim. Active blood detection in a high resolution capsule endoscopy using color spectrum transformation. In *International Conference on BioMedical Engineering and Informatics*, volume 1, pages 859–862. IEEE, 2008.

[63] Yohannes Kassahun, Bingbin Yu, Abraham Temesgen Tibebu, Danail Stoyanov, Stamatia Giannarou, Jan Hendrik Metzen, and Emmanuel Vander Poorten. Surgical robotics beyond enhanced dexterity instrumentation: a survey of machine learning techniques and their role in intelligent and autonomous surgical actions. *International Journal of Computer Assisted Radiology and Surgery*, 11(4):553–568, 2016.

[64] Hiroharu Kato, Deniz Beker, Mihai Morariu, Takahiro Ando, Toru Matsuoka, Wadim Kehl, and Adrien Gaidon. Differentiable rendering: A survey. *arXiv preprint arXiv:2006.12057*, 2020.

[65] Peter Kazanzides, Zihan Chen, Anton Deguet, Gregory S Fischer, Russell H Taylor, and Simon P DiMaio. An open-source research kit for the da vinci® surgical system. In *International Conference on Robotics and Automation*, pages 6434–6439. IEEE, 2014.

[66] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics Symposium on Geometry Processing*, volume 7, 2006.

[67] Ben Kehoe, Gregory Kahn, Jeffrey Mahler, Jonathan Kim, Alex Lee, Anna Lee, Keisuke Nakagawa, Sachin Patil, W Douglas Boyd, Pieter Abbeel, and Ken Goldberg. Autonomous multilateral debridement with the raven surgical robot. In *International Conference on Robotics and Automation*, pages 1432–1439. IEEE, 2014.

[68] Maik Keller, Damien Lefloch, Martin Lambers, Shahram Izadi, Tim Weyrich, and Andreas Kolb. Real-time 3D reconstruction in dynamic scenes using point-based fusion. In *International Conference on 3D Vision*, pages 1–8. IEEE, 2013.

[69] Alex Kendall, Hayk Martirosyan, Saumitro Dasgupta, Peter Henry, Ryan Kennedy, Abraham Bachrach, and Adam Bry. End-to-end learning of geometry and context for deep stereo regression. In *International Conference on Computer Vision*, pages 66–75. IEEE, 2017.

[70] Michael Krainin, Peter Henry, Xiaofeng Ren, and Dieter Fox. Manipulator and object tracking for in hand model acquisition. In *International Conference on Robots and Automation*, pages 1817–1824. IEEE, 2010.

[71] Thomas Kurmann, Pablo Marquez Neila, Xiaofei Du, Pascal Fua, Danail Stoyanov, Sebastian Wolf, and Raphael Sznitman. Simultaneous recognition and pose estimation of instruments in minimally invasive surgery. *Medical Image Computing and Computer-Assisted Intervention*, pages 505–513, 2017.

[72] Jens Lambrecht. Robust few-shot pose estimation of articulated robots using monocular cameras and deep-learning-based keypoint detection. In *International Conference on Robot Intelligence Technology and Applications*, pages 136–141. IEEE, 2019.

[73] Jens Lambrecht and Linh Kästner. Towards the usage of synthetic data for marker-less pose estimation of articulated robots in rgb images. In *International Conference on Advanced Robotics*, pages 240–247. IEEE, 2019.

[74] Christoph Lassner and Michael Zollhofer. Pulsar: Efficient sphere-based neural rendering. In *Conference on Computer Vision and Pattern Recognition*, pages 1440–1449. IEEE, 2021.

[75] Joël L Lavanchy, Joel Zindel, Kadir Kirtac, Isabell Twick, Enes Hosgor, Daniel Candinas, and Guido Beldi. Automation of surgical skill assessment using a three-stage machine learning algorithm. *Scientific Reports*, 11(1):1–9, 2021.

[76] Quoc V Le and Andrew Y Ng. Joint calibration of multiple sensors. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3651–3658. IEEE, 2009.

[77] Timothy E Lee, Jonathan Tremblay, Thang To, Jia Cheng, Terry Mosier, Oliver Kroemer, Dieter Fox, and Stan Birchfield. Camera-to-robot pose estimation from a single image. In *International Conference on Robotics and Automation*, pages 9426–9432. IEEE, 2020.

[78] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp: An accurate o(n) solution to the pnp problem. *International Journal of Computer Vision*, 81(2):155, 2009.

[79] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, 37(4-5):421–436, 2018.

[80] Baopu Li and Max Q-H Meng. Computer-aided detection of bleeding regions for capsule endoscopy images. *Transactions on Biomedical Engineering*, 56(4):1032–1039, 2009.

[81] Yang Li, Florian Richter, Jingpei Lu, Emily K Funk, Ryan K Orosco, Jianke Zhu, and Michael C Yip. Super: A surgical perception framework for endoscopic tissue manipulation with surgical robotics. *Robotics and Automation Letters*, 5(2):2294–2301, 2020.

[82] Yang Li, Jianke Zhu, Steven CH Hoi, Wenjie Song, Zhefeng Wang, and Hantang Liu. Robust estimation of similarity transformation for visual object tracking. In *Conference on Artificial Intelligence*, volume 33, pages 8666–8673. AAAI, 2019.

[83] Michael Liedlgruber and Andreas Uhl. Computer-aided decision support systems for endoscopy in the gastrointestinal tract: a review. *Reviews in Biomedical Engineering*, 4:73–88, 2011.

[84] Fei Liu, Zihan Li, Yunhai Han, Jingpei Lu, Florian Richter, and Michael C Yip. Real-to-sim registration of deformable soft tissue with position-based dynamics for surgical robot autonomy. In *International Conference on Robotics and Automation*, pages 12328–12334. IEEE, 2021.

[85] Jianguo Liu and Xiaohui Yuan. Obscure bleeding detection in endoscopy images using support vector machines. *Optimization and Engineering*, 10(2):289–299, 2009.

[86] Taoming Liu and Murat Cenk Cavusoglu. Needle grasp and entry port selection for automatic execution of suturing tasks in robotic minimally invasive surgery. *Transactions on Automation Science and Engineering*, 13(2):552–563, 2016.

[87] Yonghao Long, Zhaoshuo Li, Chi Hang Yee, Chi Fai Ng, Russell H Taylor, Mathias Unberath, and Qi Dou. E-dssr: efficient dynamic surgical scene reconstruction with transformer-based stereoscopic depth perception. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 415–425. Springer, 2021.

[88] Jingpei Lu, Ambareesh Jayakumari, Florian Richter, Yang Li, and Michael C Yip. Super deep: A surgical perception framework for robotic tissue manipulation using deep learning for feature extraction. In *International Conference on Robotics and Automation*, pages 4783–4789. IEEE, 2021.

[89] Jingpei Lu, Florian Richter, and Michael C Yip. Pose estimation for robot manipulators via keypoint optimization and sim-to-real transfer. *Robotics and Automation Letters*, 7(2):4622–4629, 2022.

[90] Bruce D Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *International Joint Conference on Artificial Intelligence*, pages 674–679. Vancouver, British Columbia, 1981.

[91] Leon B Lucy. A numerical approach to the testing of the fission hypothesis. *The Astronomical Journal*, 82:1013–1024, 1977.

[92] Mitchell JH Lum, Diana CW Friedman, Ganesh Sankaranarayanan, Hawkeye King, Kenneth Fodero, Rainer Leuschke, Blake Hannaford, Jacob Rosen, and Mika N Sinanan. The raven: Design and validation of a telesurgery system. *The International Journal of Robotics Research*, 28(9):1183–1197, 2009.

[93] Miles Macklin and Matthias Müller. Position based fluids. *Transactions on Graphics*, 32(4):1–12, 2013.

[94] Jeffrey Mahler, Sanjay Krishnan, Michael Laskey, Siddarth Sen, Adithyavairavan Murali, Ben Kehoe, Sachin Patil, Jiannan Wang, Mike Franklin, Pieter Abbeel, and Ken Goldberg. Learning accurate kinematic control of cable-driven surgical robots using data cleaning and gaussian process regression. In *International Conference on Automation Science and Engineering*, pages 532–539. IEEE, 2014.

[95] Jeffrey Mahler, Matthew Matl, Vishal Satish, Michael Danielczuk, Bill DeRose, Stephen McKinley, and Ken Goldberg. Learning ambidextrous robot grasping policies. *Science Robotics*, 4(26), 2019.

[96] Nader Mahmoud, Toby Collins, Alexandre Hostettler, Luc Soler, Christophe Doignon, and Jose Maria Martinez Montiel. Live tracking and dense reconstruction for handheld monocular endoscopy. *Transactions on Medical Imaging*, 38(1):79–89, 2018.

[97] Lena Maier-Hein, Anja Groch, Adrien Bartoli, Sebastian Bodenstedt, Guillaume Boissonnat, Ping-Lin Chang, Neil T Clancy, Daniel S Elson, Sven Haase, Eric Heim, Joachim Hornegger, Pierre Jannin, Hannes Kenngott, Thomas Kilgus, Beat P Müller-Stich, D. Oladokun, S. Röhl, Thiago R. dos Santos, Heinz-Peter Schlemmer, Alexander Seitel, Stefanie Speidel, Martin Wagner, and Danail Stoyanov. Comparative validation of single-shot optical techniques for laparoscopic 3-d surface reconstruction. *Transactions on Medical Imaging*, 33(10):1913–1930, 2014.

[98] Andres Marmol, Artur Banach, and Thierry Peynot. Dense-arthroslam: Dense intra-articular 3-d reconstruction with robust localization prior for arthroscopy. *Robotics and Automation Letters*, 4(2):918–925, 2019.

[99] Jiri Matas, Charles Galambos, and Josef Kittler. Robust detection of lines using the progressive probabilistic Hough transform. *Computer Vision and Image Understanding*, 78(1):119–137, 2000.

[100] Alexander Mathis, Pranav Mamidanna, Kevin M Cury, Taiga Abe, Venkatesh N Murthy, Mackenzie Weygandt Mathis, and Matthias Bethge. Deeplabcut: markerless pose estimation of user-defined body parts with deep learning. *Nature Neuroscience*, 21(9):1281, 2018.

[101] Hermann Mayer, Faustino Gomez, Daan Wierstra, Istvan Nagy, Alois Knoll, and Jürgen Schmidhuber. A system for robotic heart surgery that learns to tie knots using recurrent neural networks. *Advanced Robotics*, 22(13-14):1521–1537, 2008.

[102] Muneaki Miyasaka, Mohammad Haghighipanah, Yangming Li, Joseph Matheson, Andrew Lewis, and Blake Hannaford. Modeling cable driven robot with hysteresis and cable-pulley network friction. *Transactions on Mechatronics*, 25(2):1095–1104, 2020.

[103] Joseph J Monaghan. Sph without a tensile instability. *Journal of Computational Physics*, 159(2):290–311, 2000.

[104] Roozbeh Mottaghi, Connor Schenck, Dieter Fox, and Ali Farhadi. See the glass half full: Reasoning about liquid containers, their volume and content. In *International Conference on Computer Vision*, pages 1871–1880. IEEE, 2017.

[105] Matthias Müller, David Charypar, and Markus H Gross. Particle-based fluid simulation for interactive applications. In *Symposium on Computer Animation*, pages 154–159, 2003.

[106] Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. Position based dynamics. *Journal of Visual Communication and Image Representation*, 18(2):109–118, 2007.

[107] Adnan Munawar, Jie Ying Wu, Gregory S Fischer, Russell H Taylor, and Peter Kazanzides. Open simulation environment for learning and practice of robot-assisted surgical suturing. *Robotics and Automation Letters*, 7(2):3843–3850, 2022.

[108] Adithyavairavan Murali, Siddarth Sen, Ben Kehoe, Animesh Garg, Seth McFarland, Sachin Patil, W Douglas Boyd, Susan Lim, Pieter Abbeel, and Ken Goldberg. Learning by observation for surgical subtasks: Multilateral cutting of 3d viscoelastic and 2d orthotropic tissue phantoms. In *International Conference on Robotics and Automation*, pages 1202–1209. IEEE, 2015.

[109] Tanmay Nath, Alexander Mathis, An Chi Chen, Amir Patel, Matthias Bethge, and Mackenzie Weygandt Mathis. Using deeplabcut for 3d markerless pose estimation across species and behaviors. *Nature Protocols*, 14(7):2152–2176, 2019.

[110] Richard A Newcombe, Dieter Fox, and Steven M Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *Conference on Computer Vision and Pattern Recognition*, pages 343–352, 2015.

[111] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *International Symposium on Mixed and Augmented Reality*, volume 11, pages 127–136. IEEE, 2011.

[112] Dat Tien Ngo, Sanghyuk Park, Anne Jorstad, Alberto Crivellaro, Chang D Yoo, and Pascal Fua. Dense image registration and deformable surface reconstruction in presence of occlusions and minimal texture. In *International Conference on Computer Vision*, pages 2273–2281, 2015.

[113] Takayuki Okamoto, Takashi Ohnishi, Hiroshi Kawahira, Olga Dergachyava, Pierre Jannin, and Hideaki Haneishi. Real-time identification of blood regions for hemostasis support in laparoscopic surgery. *Signal, Image and Video Processing*, 13(2):405–412, 2019.

[114] Edwin Olson. Apriltag: A robust and flexible visual fiducial system. In *International Conference on Robotics and Automation*, pages 3400–3407. IEEE, 2011.

[115] Ryan K Orosco, Benjamin Lurie, Tokio Matsuzaki, Emily K Funk, Vasu Divi, F Christopher Holsinger, Steven Hong, Florian Richter, Nikhil Das, and Michael Yip. Compensatory motion scaling for time-delayed robotic surgery. *Surgical endoscopy*, 35(6):2613–2618, 2021.

[116] Krittin Pachtrachai, Max Allan, Vijay Pawar, Stephen Hailes, and Danail Stoyanov. Hand-eye calibration for robotic assisted minimally invasive surgery without a calibration object. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2485–2491. IEEE, 2016.

[117] Krittin Pachtrachai, Francisco Vasconcelos, François Chadebecq, Max Allan, Stephen Hailes, Vijay Pawar, and Danail Stoyanov. Adjoint transformation algorithm for hand–eye calibration with applications in robotic assisted surgery. *Annals of Biomedical Engineering*, 46(10):1606–1620, 2018.

[118] Krittin Pachtrachai, Francisco Vasconcelos, George Dwyer, Stephen Hailes, and Danail Stoyanov. Hand-eye calibration with a remote centre of motion. *Robotics and Automation Letters*, 4(4):3121–3128, 2019.

[119] Guobing Pan, Guozheng Yan, Xiangling Qiu, and Jiehao Cui. Bleeding detection in wireless capsule endoscopy based on probabilistic neural network. *Journal of Medical Systems*, 35(6):1477–1484, 2011.

[120] Frank C Park and Bryan J Martin. Robot sensor calibration: solving ax= xb on the euclidean group. *Transactions on Robotics and Automation*, 10(5):717–721, 1994.

[121] Peter Pastor, Mrinal Kalakrishnan, Jonathan Binney, Jonathan Kelly, Ludovic Righetti, Gaurav Sukhatme, and Stefan Schaal. Learning task error models for manipulation. In *International Conference on Robotics and Automation*, pages 2612–2618. IEEE, 2013.

[122] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[123] Haonan Peng, Xingjian Yang, Yun-Hsuan Su, and Blake Hannaford. Real-time data driven precision estimator for raven-ii surgical robot end effector position. In *International Conference on Robotics and Automation*, pages 350–356. IEEE, 2020.

[124] Jenelle Armstrong Piepmeier and Harvey Lipkin. Uncalibrated eye-in-hand visual servoing. *The International Journal of Robotics Research*, 22(10-11):805–819, 2003.

[125] Ameya Pore, Davide Corsi, Enrico Marchesini, Diego Dall'Alba, Alicia Casals, Alessandro Farinelli, and Paolo Fiorini. Safe reinforcement learning using formal verification for tissue retraction in autonomous robotic-assisted surgery. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4025–4031. IEEE, 2021.

[126] Vijay Pradeep, Kurt Konolige, and Eric Berger. Calibrating a multi-arm multi-sensor robot: A bundle adjustment approach. In *Experimental Robotics*, pages 211–225. Springer, 2014.

[127] Yidan Qin, Max Allan, Yisong Yue, Joel W Burdick, and Mahdi Azizian. Learning invariant representation of tasks for robust surgical state estimation. *Robotics and Automation Letters*, 6(2):3208–3215, 2021.

[128] Arturo Rankin and Larry Matthies. Daytime water detection based on color variation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 215–221. IEEE, 2010.

[129] Arturo L Rankin, Larry H Matthies, and Paolo Bellutta. Daytime water detection based on sky reflections. In *International Conference on Robotics and Automation*, pages 5329–5336. IEEE, 2011.

[130] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv preprint arXiv:2007.08501*, 2020.

[131] Austin Reiter, Peter K Allen, and Tao Zhao. Feature classification for tracking articulated surgical tools. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 592–600. Springer, 2012.

[132] Austin Reiter, Peter K Allen, and Tao Zhao. Appearance learning for 3d tracking of robotic surgical tools. *The International Journal of Robotics Research*, 33(2):342–356, 2014.

[133] Florian Richter, Emily K Funk, Won Seo Park, Ryan K Orosco, and Michael C Yip. From bench to bedside: The first live robotic surgery on the dvrk to enable remote telesurgery with motion scaling. In *International Symposium on Medical Robotics*, pages 1–7. IEEE, 2021.

[134] Florian Richter, Ryan K Orosco, and Michael C Yip. Motion scaling solutions for improved performance in high delay surgical teleoperation. In *International Conference on Robotics and Automation*, pages 1590–1595. IEEE, 2019.

[135] Florian Richter, Ryan K Orosco, and Michael C Yip. Open-sourced reinforcement learning environments for surgical robotics. *arXiv preprint arXiv:1903.02090*, 2019.

[136] Florian Richter, Shihao Shen, Fei Liu, Jingbin Huang, Emily K Funk, Ryan K Orosco, and Michael C Yip. Autonomous robotic suction to clear the surgical field for hemostasis using image-based blood flow detection. *Robotics and Automation Letters*, 6(2):1383–1390, 2021.

[137] Florian Richter, Yifei Zhang, Yuheng Zhi, Ryan K Orosco, and Michael C Yip. Augmented reality predictive displays to help mitigate the effects of delayed telesurgery. In *International Conference on Robotics and Automation*. IEEE, 2019.

[138] Eric Rohmer, Surya PN Singh, and Marc Freese. V-rep: A versatile and scalable robot simulation framework. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1321–1326. IEEE, 2013.

[139] Thomas J Rothenberg. Identification in parametric models. *Econometrica*, 39(3):577–591, 1971.

[140] Mathieu Salzmann and Pascal Fua. *Deformable Surface 3D Reconstruction from Monocular Images*. Synthesis Lectures on Computer Vision. Morgan & Claypool Publishers, 2010.

[141] Hagit Schechter and Robert Bridson. Ghost sph for animating water. *Transactions on Graphics*, 31(4):1–8, 2012.

[142] Connor Schenck and Dieter Fox. Visual closed-loop control for pouring liquids. In *International Conference on Robotics and Automation*, pages 2629–2636. IEEE, 2017.

[143] Connor Schenck and Dieter Fox. Perceiving and reasoning about liquids using fully convolutional networks. *The International Journal of Robotics Research*, 37(4-5):452–471, 2018.

[144] Connor Schenck and Dieter Fox. Spnets: Differentiable fluid dynamics for deep neural networks. In *Conference on Robot Learning*, pages 317–335. PMLR, 2018.

[145] Daniel Seita, Sanjay Krishnan, Roy Fox, Stephen McKinley, John Canny, and Ken Goldberg. Fast and reliable autonomous surgical debridement with cable-driven robots using a two-phase calibration procedure. In *International Conference on Robotics and Automation*, pages 6651–6658. IEEE, 2018.

[146] Siddarth Sen, Animesh Garg, David V Gealy, Stephen McKinley, Yiming Jen, and Ken Goldberg. Automating multi-throw multilateral surgical suturing with a mechanical needle guide and sequential convex optimization. In *International Conference on Robotics and Automation*, pages 4178–4185. IEEE, 2016.

[147] Azad Shademan, Ryan S Decker, Justin D Opfermann, Simon Leonard, Axel Krieger, and Peter CW Kim. Supervised autonomous robotic soft tissue surgery. *Science translational medicine*, 8(337):337ra64, 2016.

[148] Yantao Shen, Dong Sun, Yun-Hui Liu, and Kejie Li. Asymptotic trajectory tracking of manipulators using uncalibrated visual feedback. *Transactions on Mechatronics*, 8(1):87–98, 2003.

[149] Changyeob Shin, Peter Walker Ferguson, Sahba Aghajani Pedram, Ji Ma, Erik P Dutson, and Jacob Rosen. Autonomous tissue manipulation via surgical robot using learning based model predictive control. In *International Conference on Robotics and Automation*, pages 3875–3881. IEEE, 2019.

[150] Yiu Cheung Shiu and Shaheen Ahmad. Calibration of wrist-mounted robotic sensors by solving homogeneous transform equations of the form ax= xb. *Transactions on Robotics and Automation*, 5(1):16–29, 1989.

[151] Jingwei Song, Jun Wang, Liang Zhao, Shoudong Huang, and Gamini Dissanayake. Dynamic reconstruction of deformable soft-tissue with stereo scope in minimal invasive surgery. *Robotics and Automation Letters*, 3(1):155–162, 2017.

[152] Jingwei Song, Jun Wang, Liang Zhao, Shoudong Huang, and Gamini Dissanayake. Mis-slam: Real-time large-scale dense deformable slam system in minimal invasive surgery based on heterogeneous computing. *Robotics and Automation Letters*, 3(4):4068–4075, 2018.

[153] Stanford Artificial Intelligence Laboratory. Robotic operating system.

[154] Danail Stoyanov, Marco Visentini Scarzanella, Philip Pratt, and Guang-Zhong Yang. Real-time stereo reconstruction in robotically assisted minimally invasive surgery. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 275–282. Springer, 2010.

[155] Robert W Sumner, Johannes Schmid, and Mark Pauly. Embedded deformation for shape manipulation. *Transactions on Graphics*, 26(3):80, 2007.

[156] Priya Sundaresan, Brijen Thananjeyan, Johnathan Chiu, Danyal Fer, and Ken Goldberg. Automated extraction of surgical needles from tissue phantoms. In *International Conference on Automation Science and Engineering*, pages 170–177. IEEE, 2019.

[157] Damien Teney and Martial Hebert. Learning to extract motion from videos in convolutional neural networks. In *Asian Conference on Computer Vision*, pages 412–428. Springer, 2016.

[158] Brijen Thananjeyan, Animesh Garg, Sanjay Krishnan, Carolyn Chen, Lauren Miller, and Ken Goldberg. Multilateral surgical pattern cutting in 2d orthotropic gauze with deep reinforcement learning policies for tensioning. In *International Conference on Robotics and Automation*, pages 2371–2378. IEEE, 2017.

[159] Sebastian Thrun. Particle filters in robotics. In *Conference on Uncertainty in Artificial Intelligence*, pages 511–518. Morgan Kaufmann Publishers Inc., 2002.

[160] Titan Medical. Sport surgical system.

[161] Roger Y Tsai and Reimar K Lenz. A new technique for fully autonomous and efficient 3 d robotics hand/eye calibration. *Transactions on Robotics and Automation*, 5(3):345–358, 1989.

[162] Mrinal Verghese, Florian Richter, Aaron Gunn, Phil Weissbrod, and Michael Yip. Model-free visual control for continuum robot manipulators via orientation adaptation. *arXiv preprint arXiv:1909.00450*, 2019.

[163] Feng Wang, Kai Chen, and Xiaoping Chen. An online calibration method for manipulator with joint clearance. *Robot*, (5):2, 2013.

[164] Zerui Wang, Ziwei Liu, Qianli Ma, Alexis Cheng, Yun-hui Liu, Sungmin Kim, Anton Deguet, Austin Reiter, Peter Kazanzides, and Russell H Taylor. Vision-based calibration of dual rcm-based robot arms in human-robot collaborative minimally invasive surgery. *Robotics and Automation Letters*, 3(2):672–679, 2017.

[165] Kengo Watanabe, Takahiro Kanno, Kazuhisa Ito, and Kenji Kawashima. Single-master dual-slave surgical robot with automated relay of suture needle. *Transactions on Industrial Electronics*, 65(8):6343–6351, 2017.

[166] Hongtao Wu and Gregory S Chirikjian. Can i pour into it? robot imagining open containability affordance of previously unseen objects via physical simulations. *Robotics and Automation Letters*, 6(1):271–278, 2020.

[167] Jiaqi Xu, Bin Li, Bo Lu, Yun-Hui Liu, Qi Dou, and Pheng-Ann Heng. Surrol: An open-source reinforcement learning centered and dvrk compatible platform for surgical robot learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1821–1828. IEEE, 2021.

[168] Akihiko Yamaguchi and Christopher G Atkeson. Stereo vision of liquid and particle flow for robot pouring. In *International Conference on Humanoid Robots*, pages 1173–1180. IEEE, 2016.

[169] Menglong Ye, Lin Zhang, Stamatia Giannarou, and Guang-Zhong Yang. Real-time 3d tracking of articulated tools for robotic surgery. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 386–394, Cham, 2016. Springer International Publishing.

[170] Wang Yifan, Felice Serena, Shihao Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. Differentiable surface splatting for point-based geometry processing. *Transactions on Graphics*, 38(6):1–14, 2019.

[171] Michael Yip and Nikhil Das. Robot autonomy for surgery. In *The Encyclopedia of MEDICAL ROBOTICS: Volume 1 Minimally Invasive Surgical Robotics*, pages 281–313. World Scientific, 2019.

[172] Michael C Yip, David G Lowe, Septimiu E Salcudean, Robert N Rohling, and Christopher Y Nguan. Tissue tracking and registration for image-guided surgery. *Transactions on Medical Imaging*, 31(11):2169–2182, 2012.

[173] Jihun Yu and Greg Turk. Reconstructing surfaces of particle-based fluids using anisotropic kernels. *Transactions on Graphics*, 32(1):1–12, 2013.

[174] Guangming Zang, Ramzi Idoughi, Congli Wang, Anthony Bennett, Jianguo Du, Scott Skeen, William L Roberts, Peter Wonka, and Wolfgang Heidrich. Tomofluid: reconstructing dynamic fluid from sparse view videos. In *Conference on Computer Vision and Pattern Recognition*, pages 1870–1879. IEEE, 2020.

[175] Feihu Zhang, Victor Prisacariu, Ruigang Yang, and Philip HS Torr. Ga-net: Guided aggregation net for end-to-end stereo matching. In *Conference on Computer Vision and Pattern Recognition*, pages 185–194. IEEE, 2019.

[176] Zhiqiang Zhang, Lin Zhang, and Guang-Zhong Yang. A computationally efficient method for hand–eye calibration. *International Journal of Computer Assisted Radiology and Surgery*, 12(10):1775–1787, 2017.

[177] Tao Zhao, Wenyi Zhao, Brian D Hoffman, William C Nowlin, and Hua Hui. Efficient vision and kinematic data fusion for robotic surgical instruments and other applications, March 3 2015. US Patent 8,971,597.

[178] Fangxun Zhong, Yaqing Wang, Zerui Wang, and Yun-Hui Liu. Dual-arm robotic needle insertion with active tissue deformation for autonomous suturing. *Robotics and Automation Letters*, 4(3):2669–2676, 2019.

[179] Fangxun Zhong, Zerui Wang, Wei Chen, Kejing He, Yaqing Wang, and Yun-Hui Liu. Hand-eye calibration of surgical instrument for robotic surgery using interactive manipulation. *Robotics and Automation Letters*, 5(2):1540–1547, 2020.

[180] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018.

[181] Jianke Zhu, Steven CH Hoi, and Michael R Lyu. Nonrigid shape recovery by gaussian process regression. In *Conference on Computer Vision and Pattern Recognition*, pages 1319–1326. IEEE, 2009.

[182] Jianke Zhu, Michael R Lyu, and Thomas S Huang. A fast 2d shape recovery approach by fusing features and appearance. *Transactions on Pattern Analysis and Machine Intelligence*, 31(7):1210–1224, 2008.