Vision Based Surgical Tool Tracking with Robot Kinematics Prior

Yun-Hsuan Su

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Master of Science in Electrical Engineering

University of Washington

2017

Reading Committee:

Blake Hannaford, Chair

Howard Jay Chizeck

Program Authorized to Offer Degree:
Electrical Engineering

University of Washington

Abstract

Vision Based Surgical Tool Tracking with Robot Kinematics Prior

Yun-Hsuan Su

Chair of the Supervisory Committee:
Professor Blake Hannaford
Electrical Engineering

Robot assisted minimally invasive surgery combines the skill and techniques of highly-trained surgeons with the robustness and precision of machines. Through a teleoperation scheme, surgeons can execute high-level surgical tasks by commanding instruments controlled by precise robotic devices. Several advantages arise. To name a few: (1) achieved precision is beyond that of human dexterity alone (2) a greater number of kinematic degrees of freedom are possible at the surgical tool tip (3) surgeons are able to operate remotely, i.e. agnostic of patient location given a suitable communication line. Despite the numerous advantages over traditional key-hole or laparoscopic surgery, the lack of realistic and real-time force feedback is a major drawback — discerning tool-tissue interactions can be unintuitive and can ultimately result in unintentional tissue damage. Directly sensing forces at the tool-tissue interface is theoretically possible using tool tip mounted force sensors, but this approach is not amenable to required sterilization procedures. Thus, a vision based force estimation method is proposed to infer the applied force based on real-time analysis of tissue deformation.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1
INTRODUCTION

Ever since the first documented use of a robot-assisted surgical procedure back in 1985 [1] , great progress has been done towards advancing the technology of robot manipulated surgical practice. Yet, obtaining accurate force feedback during such kinds of surgical operations is a heated topic. In fact, no current commercialized surgical robots provide information about the applied force, and thus this field of research still holds great potential.

1.1 Background

In minimally invasive surgeries (MIS), surgeons rely on endoscope images to operate on patients. Recently, surgical robots have been employed in MIS. If they control the instruments via haptic devices, force feedback can be sent as a warning when the surgical tool tip approaches important arteries or nerves, which could promote a higher level of safety [7]. However, force feedback can be hard to obtain. Due to size, cost, constraints and difficulties in sterility maintenance, directly applying force sensors on the tip of surgical tools is often impractical. An alternative is to visually measure the tissue indentation due to surgical contacts in endoscope images, then infer force accordingly from the level of deformation.

1.2 Overview of the Four Stages

The proposed method for achieving visual force estimation can be divided into four stages as shown in Fig.1.1. First, image segmentation classifies pixels into either tool or tissue. The tissue pixels neighboring the surgical tool tip is our region of interest. In stage two,

[1]The robot, PUMA 200 (Westinghouse Electric, Pittsburgh, PA), was used for needle placement in a CT-guided brain biopsy[13].

a depth map within this region of interest is generated and one can quantify the level of deformation by re-projecting pixel points to 3D space. In stage three, the mapping between deformation of tool-tissue contact and applied force is established. Lastly, a haptic device is used to actually feel the force feedback while teleoperating the surgical robot arm. Below are more in-depth explanations for each of the four stages.

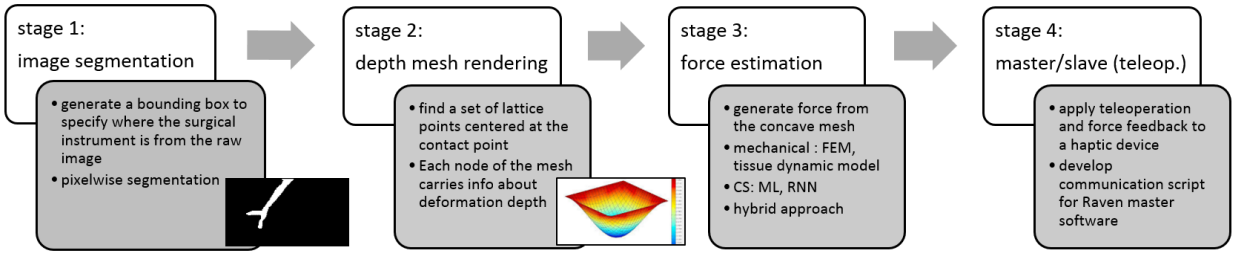

Figure 1.1: The workflow of four stages.

1.2.1 Image Segmentation

In vision-based force estimation of MIS as well as in related research, surgical instrument tracking from endoscopic images is a prerequisite, since the very first step is to define our region of interest where tool-tissue interaction occurs. Instead of creating a bounding box, pixel-wise segmentation is necessary in our application, because tissue deformation should be analyzed only on the non-tool pixels, where those pixels are determined by the segmentation result.

Some challenges in surgical tool segmentation include motion blur, partial occlusion, specular reflections[8] on wet tissue surface, lighting changes from the point light source. The metallic parts of surgical tools often reflect tissue colors, which increases the difficulty in color filtering.

Considering the above factors, Fig.1.2 shows a broad spectrum of different ways previous researchers have dealt with the task of image-based surgical instrument detection. According to a survey paper[10] in 2016, there are two main approaches - marker and marker-less, where

a marker can be further classified into either a visual marker or a non-visual marker. In terms of visual markers, topology markers stand out in instrument detection because distortions of the tags still yield a positive detection, unless they change the topology. Since most surgical tool shafts have a cylindrical shape, when putting a marker on the tool shaft, it is often not fixed on a flat surface, distortion is thus inevitable. Yet in general, visual markers suffer from lack of robustness to occlusion, which can easily happen during surgical operations when the tool is stained with blood. Contrarily, non-visual markers are not subject to occlusion, some of which include RFID sensors[17], acoustic[14] or electromagnetic trackers[16] etc. Some concerns about using those markers are that preferably no external electronics should be placed inside patients' body, and so it is not desirable to place those sensors or trackers on the surgical tool tip, besides some of them are relatively expensive, and others have receivers that are undesirably big in size.

Despite the effectiveness of using markers, for easy applicability to current hospital and clinical setup, markerless surgical instrument segmentation is considered in this work. Because not all surgical tools currently in use have external sensors or markers attached.

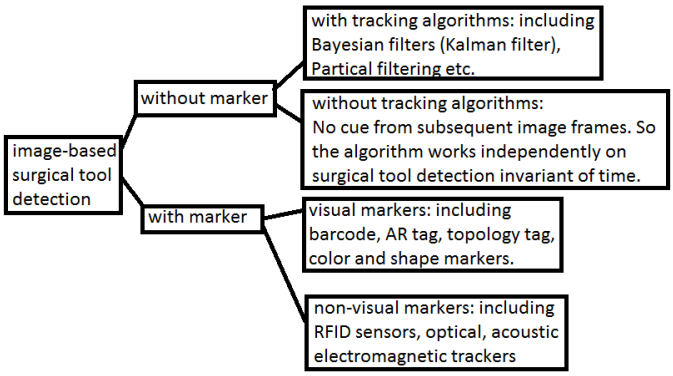

Figure 1.2: Different approaches for image-based surgical instrument detection.

In this work, the Raven Surgical Robot System[5] is used, where robot kinematics information is available, that being said we will have 3D position of the surgical tool tip as well as all the joint angle values. Given the extrinsic matrix of the camera with respect to the Raven base frame, robot kinematics would provide a shape prior as to where the surgical instrument may appear on the image frame by projecting the 3D points of the surgical tool links onto the 2D image. Afterwards, a color filtering scheme can be applied to modify our predicted shape prior using shape matching algorithm in the frequency domain using DFT. Finally, with a modified shape prior mask, a color mask is applied to do pixel-

wise classification within the surgical tool region. An overview of the work done in this stage can be seen in Fig.1.3.

Given the segmented surgical tool pixels, the goal in this stage is to generate a depth lattice centered at the surgical instrument contact point, so that it can later be turned into deformation map caused by tool-tissue contact. This depth map is derived from the disparity map and intrinsic information from our stereo camera set. Since re-projecting pixel points back to 3D space is a relatively time-consuming task, the surgical instrument location from the segmentation result in the previous stage can be utilized, so that re-projection is only done within a certain neighboring region of interest.

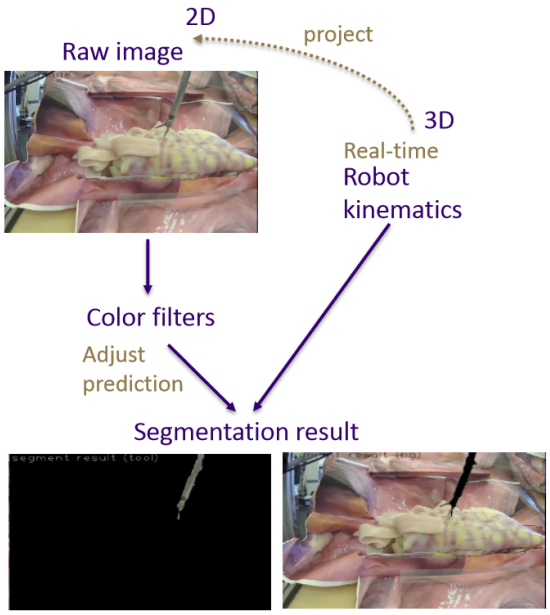

Figure 1.3: This is an overview of the work done in stage 1: image segmentation.

1.2.2 Depth Lattice Rendering

In Fig.1.4, a flowchart for deriving deformation from the stereo image frames is shown. Using robot kinematics, one can create a spherical region of interest centered at the surgical tool tip in the 3D space. This sphere can be projected onto the image frame which will look like a circle centered at the projection point of the surgical tool tip on the image frame as illustrated in Fig.1.6. Since the sphere region of interest is fixed sized, the radius of the circle is inversely proportional to distance between surgical tool and camera.

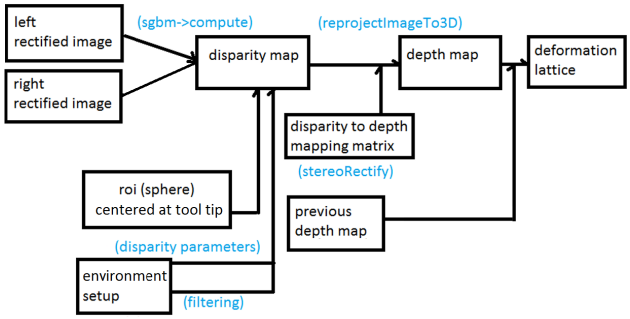

Figure 1.4: Flowchart for stereo processing.

Disparity map is only needed within the circle, which saves computation time. Also, disparity parameters and post filtering can be done when generating the disparity map according to the environment setup. Afterwards, the segmented result from stage 1 is used to determine pixels on the disparity map that belong to the tissue class and the surgical instrument class. Of course some noise may occur especially near the outer contour of the tool, but this can potentially be handled via some filtering. Then re-projecting the tissue pixels on the disparity map back into the 3D space, one can get the terrain near the tool-tissue interaction point. This re-projection can be done with a disparity to depth mapping matrix derived from a stereo rectification process. By doing this, a set of 3D points with respect to the camera frame is derived, where the z component of the points forms the depth map, because the z axis is vertical to the camera image frame. Finally by comparing subsequent depth maps, the incremental deformation level can be computed. Fig.1.5 is an overview of work done for this stage.

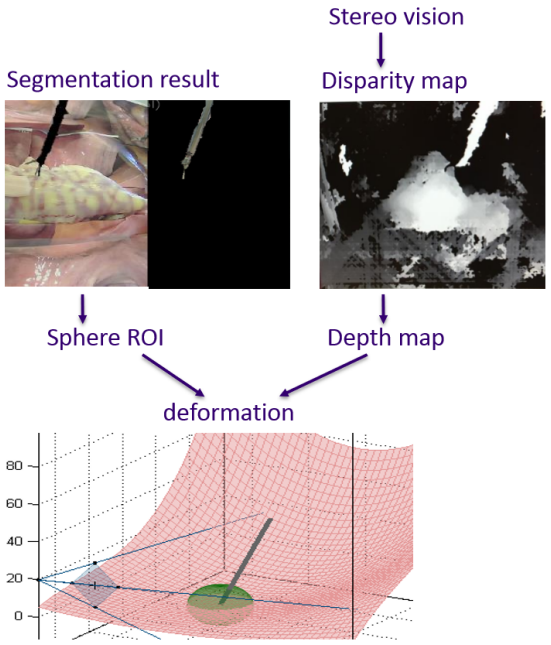

Figure 1.5: This is an overview of the work done in stage 2: depth lattice rendering.

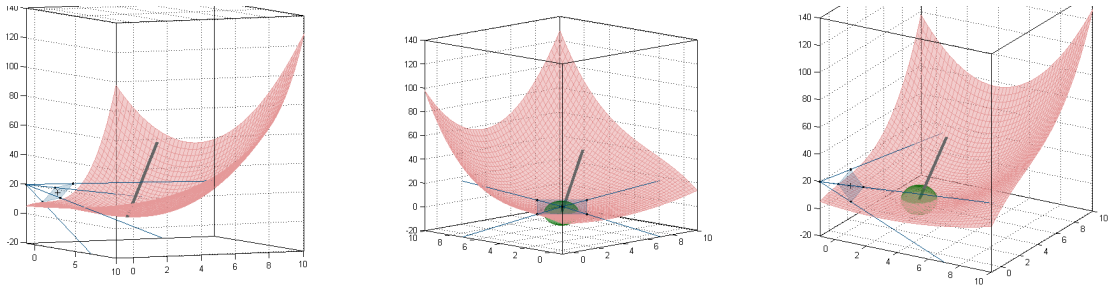

Figure 1.6: The spherical bounding box around surgical tool tip.

1.2.3 Force estimation

From the previous two stages, a deformation map centered at the surgical tool interaction point is obtained. Based on the deformation and a known tissue dynamics model, the applied force can be estimated[7].

In vision-based force estimation for robot-assisted minimally invasive surgery, force rendering relies on analysis of tissue deformation due to surgical tool contact. There are roughly two main routes to estimate force. In a mechanical or bio-medical approach, nonlinear viscoelastic dynamic models of tissue are proposed, then parameters in the tissue physical model are determined either by offline tissue identification or directly treated as prior knowledge from a medical database, finally force can be estimated by such a model [22].

On the other hand, in the computer-science perspective, they use machine learning techniques to bridge the blackbox relation between tissue deformation over time and the applied force. In many case, Recurrent Neural networks (RNN) is used because it handles the time variant feature between the training inputs tissue deformation level, and neural network output the estimated applied force. But in this sense, they either use a relatively simple mass-spring model to simulate tissue dynamics behavior or go entirely model-free [2].

Combining both approaches, one can create a hybrid method that contains a more sophisticated and realistic tissue model, while implementing a way to identify those model parameters by machine learning online.

1.2.4 Haptic Implementation

At this stage, the estimated force is sent back to a haptic device used to remotely control the surgical robot arm. In other words, a master-slave system will be setup and surgeons can perform surgical operations via a haptic device while getting real-time force feedback as though they were operating in-person.

1.3 Research Focus

From the overview, this task consists of four stages, image segmentation, depth lattice rendering, force estimation, and haptic implementation. Due to time constraints, the main focus of this thesis research is on the first two stages, but since stage three is the actual force rendering part, a rough prototype will also be presented.

In particular, as shown in Fig.1.7, detailed methodologies for stage one and two are covered in this article. Further improvements on the first two stages as well as the implementation of stage three and four will be listed as future work.

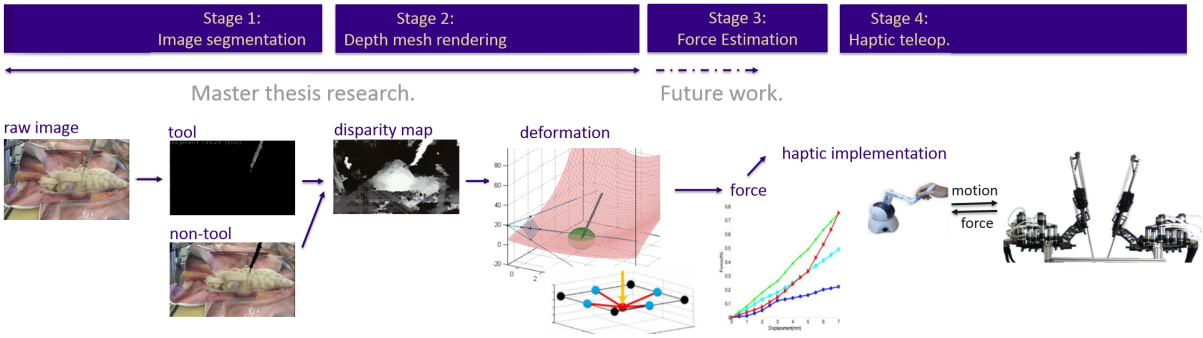

Figure 1.7: Illustration of the four stages and the research focus for this thesis.

Chapter 2

METHOD

The theoretical background, algorithms and research methods used throughout the course of this project are listed below.

2.1 Raven Coordinates Change

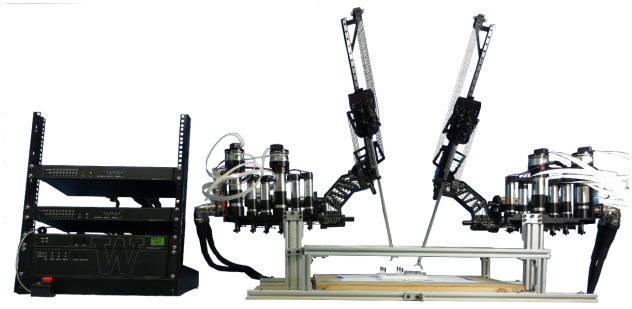

Figure 2.1: Raven Surgical Robot System.

The Raven Surgical Robot System, as shown in Fig.2.1 is an open-architecture and open-source surgical robot for laparoscopic surgery research from Applied Dexterity, a spin-off from University of Washington. There are two cable-driven 7 DOF arms, with motors all mounted at the base, so the robot arm itself is very light weight. It is intended to facilitate collaborative research on advances in the field of surgical robotics. Since Raven is in use for this work, understanding its the coordinate system is important. As illustrated in Fig.2.2, there are two coordinate systems in Raven, the zero frame and the base frame[11].

The Raven control system software, including forward and inverse kinematics and feed-forward gravity compensation, is based on a 1000-Hz real-time servo layer. It is an open source C++ program implemented as a ROS node called "r2control".

2.1.1 The zero frame

This is the origin of the serial kinematic chain for kinematic analysis of the robot. Its origin is the center of motion of the spherical mechanism, namely the intersection of axes Z_1, Z_2,

and Z_3, which stays stationary throughout the motion range of the robot arm.

2.1.2 The base frame

A frame centered on the bolt pattern on the surface of the Raven base. The origin of the frame lies between the two bottom bolt holes, directly below the center bolt of the top row, and coincident with the mounting surface on the robot.

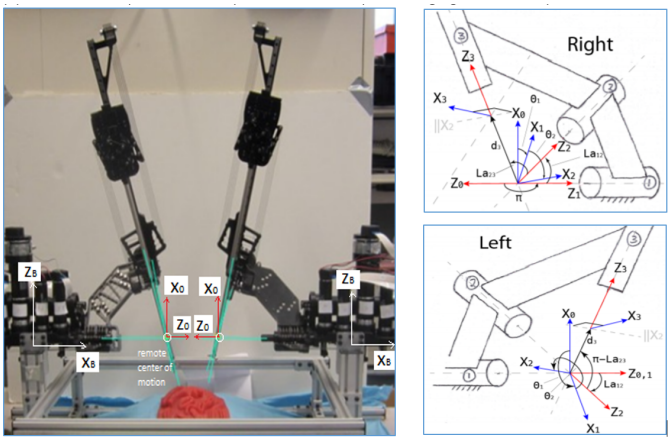

Figure 2.2: The zero frame and the base frame of the Raven Surgical Robot System.

The *ravenstate* ROS topic shows instantaneous robot position and orientation with respect to the zero frame. In fact, both the incremental motion command for controlling the robot arms and the *ravenstate* ROS topic are described in the zero frame. In terms of units, the position is presented in micro meters, and the orientation value is in radians.

Although it is straightforward to describe the robot pose with respect to the zero frame, it's origin is a virtual point which can be hard to locate, and so in this work, instead of using the zero frame, all raven kinematics information is transformed back into the base frame before any further processing. The transformation matrices and details are presented in the Raven kinematics documentation[11]. By doing so, the camera extrinsic matrix is also determined under the raven base frame, which will be covered in the following section.

2.2 Camera Pose Estimation

To use robot kinematics as prior information for surgical instrument segmentation, finding out the camera pose with respect to raven base frame is a prerequisite. This pose estimation problem is often referred to as the *Perspective-n-Point* problem or PNP. In a PNP problem, the aim is to find the pose of an object given a calibrated camera, yet the goal of interest is

to find the pose of the camera with respect to the object, so it is actually the reverse of the PNP result.

2.2.1 Algorithm Input

The inputs to this algorithm are the locations of n 3D points on the object and their corresponding 2D projections on the image, and the camera intrinsic matrix.

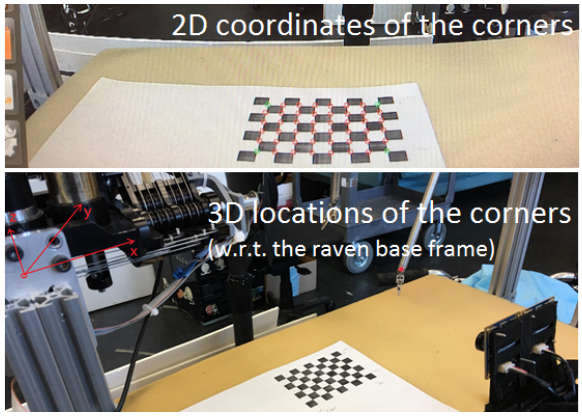

Figure 2.3: The 2D and 3D coordinate input of chessboard corners to the PNP algorithm.

2D coordinates of a few points

One needs the 2D (x, y) locations of a few points in the image. In this work, the OpenCV function *cvFindChessboardCorners*[6] is used to detect chess board corners on the image frame. This can be seen in the upper half of Fig.2.3. If all the $8 \times 6 = 48$ corners are found, the function returns true and the list of 2D coordinates of corners on the image. The usage of this function is shown below.

$$cvFindChessboardCorners(I, cvSize(W, H), C, CN);$$

, where W, H, I, C, CN are respectively the width of the chessboard, the height of the chessboard, the grayscale image frame, the output corner pixel coordinates array and the output corner count.

3D locations of the same points

The 3D location of the 48 chessboard corners can be in any arbitrary reference frame or coordinate system, which will be considered the world coordinates (a.k.a the Raven Base Frame). Then the camera extrinsic matrix will also be in the world coordinate system.

The true 3D representation of those 48 points can be obtained by manually measuring the position of each corner with respect to the raven base frame. The lower half of Fig.2.3 demonstrates the idea.

Intrinsic parameters of the camera

As mentioned before, in this problem the camera is assumed to be calibrated. In other words, one needs to know the focal length of the camera, the optical center in the image and the radial distortion parameters which can be obtained using the camera calibration function in OpenCV or packages in ROS.

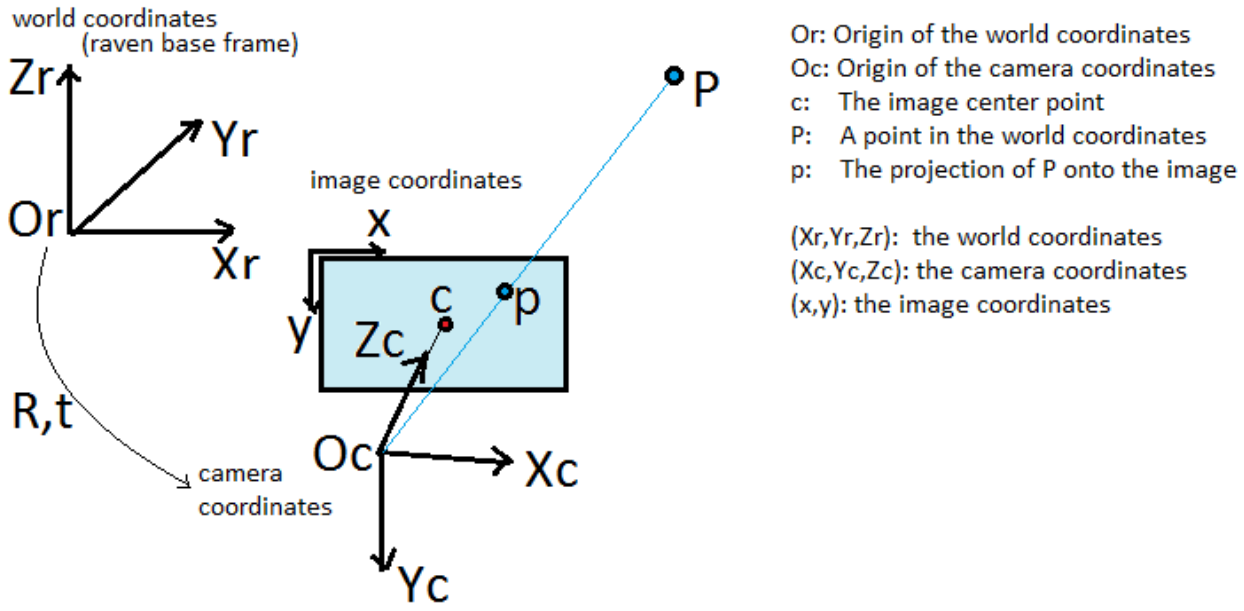

Figure 2.4: The coordinates in the PNP algorithm.

2.2.2 Theory Behind the Algorithm

With the three inputs mentioned above, one is able to figure out the camera pose with respect to the world coordinates using the PNP algorithm. The PNP algorithm works like this, suppose coordinates (X_r, Y_r, Z_r) are world coordinates, which is the Raven base frame

in our implementation, (X_c, Y_c, Z_c) is the camera coordinates. The goal would be to find the rotation $R_{3\times3}$ and translation $t_{3\times1}$ of the world coordinates with respect to the camera coordinates. The relation is illustrated in Fig.2.4, and once $R_{3\times3}$ and $t_{3\times1}$ are found, the following transformation in Eq.2.1 holds true for all points.

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = R \begin{bmatrix} X_r \\ Y_r \\ Z_r \end{bmatrix} + t \tag{2.1}$$

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = [R|t] \begin{bmatrix} X_r \\ Y_r \\ Z_r \\ 1 \end{bmatrix}$$

This can also be expressed as the expanded form in Eq.2.2:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} r_{00} & r_{01} & r_{02} & t_x \\ r_{10} & r_{11} & r_{12} & t_y \\ r_{20} & r_{21} & r_{22} & t_z \end{bmatrix} \begin{bmatrix} X_r \\ Y_r \\ Z_r \\ 1 \end{bmatrix} \tag{2.2}$$

Now considering the transformation between the camera coordinates and the image coordinates, the following Eq.2.3 holds.

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = s \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} \tag{2.3}$$

where f_x, f_y are the focal length, c_x, c_y are the optical center point, and s is the scaling factor. Suppose any point P is joined to the camera center c, no matter how far away the point P is from the camera, the projection p always stays exactly at the image center point c. That being said, given any single image, it is impossible to perceive the depth from a pixel point, such ambiguity is contained in the scaling factor s.

Direct Linear Transform

As described above, the image coordinates representation (x, y) and the world coordinates representation (X_r, Y_r, Z_r) of the 48 chessboard corners are found via OpenCV function *cvFindChessboardCorners*[6] and manual measurement respectively. Furthermore, the camera intrinsic parameters f_x, f_y, c_x, c_y and the scaling factor s are known. Thus, combining Eq.2.2 and Eq.2.3, one can potentially solve for $R_{3\times3}$ and $t_{3\times1}$ if enough corresponding points in 2D and 3D are provided.

Multiplying Eq.2.2 by $s \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$, one can get:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = s \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{00} & r_{01} & r_{02} & t_x \\ r_{10} & r_{11} & r_{12} & t_y \\ r_{20} & r_{21} & r_{22} & t_z \end{bmatrix} \begin{bmatrix} X_r \\ Y_r \\ Z_r \\ 1 \end{bmatrix} \tag{2.4}$$

To solve Eq.2.4, there is this method called *Direct Linear Transform* or DLT[1]. However, it ignores the intrinsic parameters assumed to be known here, and therefore tends to result in less stable camera pose estimate. In fact, it first computes the projection matrix, and then retrieve the extrinsic matrix by clamping the known intrinsics, which often lead to poor accuracy.

Levenberg-Marquardt Optimization

Although DLT solves for Eq.2.4, there are no contraints applied in the computation process to regulate $R_{3\times3}$ to be an orthogonal rotation matrix. In fact, while $R_{3\times3}$ has 9 entries, it contains only 3 degrees of freedom, and so in most cases the solution mentioned above is not very accurate.

Moreover, reliability of the solution to Eq.2.4 is based on the accuracy of re-projection from the image coordinates (x, y) to the world coordinates (X_r, Y_r, Z_r). Thus, the cost

function that one wants to minimize can be defined as the re-projection error. But, no external cost function is considered during the DLT computation. So, there is no guarantee that the solution is an optimized outcome to the objective of interest. Fortunately, *Levenberg-Marquardt optimization* approach is applied to iteratively reduce the re-projection error and the result can be significantly improved[15].

2.2.3 Implementation in Source Code

The OpenCV functions *solvePnP* and *solvePnPRansac* can be used to estimate pose. Several algorithms are implemented in *solvePnP*. And users can select the desired algorithm using a parameter flag. The flag *SOLVEPNP_ITERATIVE* is used by default which is essentially the DLT solution followed by *Levenberg-Marquardt optimization*.

2.3 Color Filter and Thresholding

In this work, robot kinematics information serves as a shape prior for surgical tool segmentation, afterwards color filtering is applied to the whole image to allow for fine adjustments of the prior guess and to examine if the prior information is reasonable. The color filtering scheme adapted here is based upon research[12] on components of color spaces which provide the most discriminative power to separate surgical tool pixels from background pixels. According to the study, hue and saturation in the HSV colorspace as well as $Opponent_1$ and $Opponent_2$ components derived from RGB color components were found to give the best result, where:

$$\begin{aligned} Opponent_1 &= G - R \\ Opponent_2 &= B - Y = B - (G + R) \end{aligned}$$

Moreover, when trying to compare the difference in color, non-Euclidean colorspaces like HSV can cause complications. Thus, *coneHSV* colorspace was developed to resolve the issue[18], where the three components (H, S, V) are transformed into $(V, S\cos(H), S\sin(H))$.

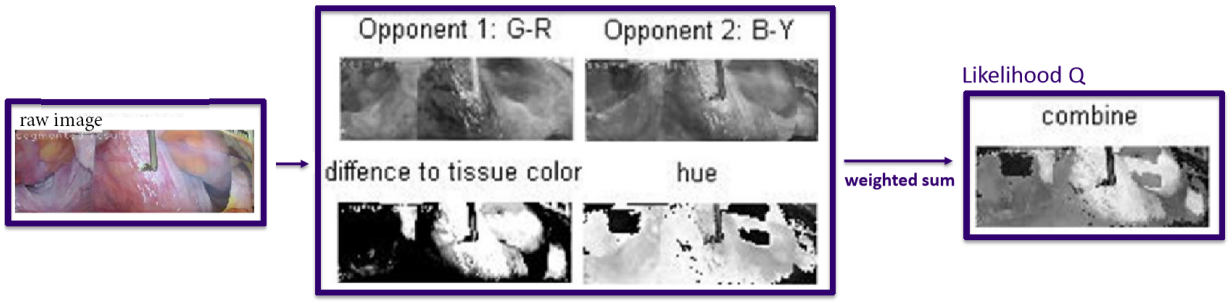

Figure 2.5: The color filter used for surgical instrument segmentation, which is a weighted sum of hue, saturation, $Opponent_1$ and $Opponent_2$

Another potential feature is the difference to tissue color. To create this feature, several videos are taken and an average color of the background tissue is computed. Then the Euclidean distance between the color for each pixel and this average tissue color becomes the feature "difference to tissue color" mentioned above.

The color filter used in this work is a linear combination of hue, saturation, $Opponent_1$, $Opponent_2$, and difference to tissue color, shown on the bottom right of Fig.2.5. More sophisticated color filtering method is not considered because of time. Furthermore, robot kinematics information available, it gives us great hint about the surgical tool configuration and predicted location of projection on the image frame, so there is no necessity in weighing down the computation load in this part. An ideal post filtering image will appear bright for the tissue pixels and significantly darker for surgical tool pixels.

2.4 Shape Matching with DFT

At this point, using robot kinematics, a shape prior mask U is obtained, and using color filtering, we have a likelihood map Q of intensities indicating the probability that reflects the probability of each pixel color being tool or non-tool. In the shape prior mask U, the parts of the image that we consider to be surgical tool pixels are marked with white while the remaining parts are black, contrarily in the likelihood map Q, the more aQ and vice versa. pixel color resembles a surgical tool color, the darker it is, and if it's similar to background

tissue color, it appears bright. Now, given the color contradiction between U and Q, the ideal case would be when all the dark pixels in U correspond to the bright pixels in Q and vice versa. Yet, the shape prior U may not align well with the likelihood map Q, so the task of shape matching is basically to find an optimal translation and rotation for U such that the modified U would match with Q best. This shape matching procedure is done using *Discrete Fourier Transform* (DFT). Finally, a color threshold is applied to do binary classification on each individual pixel within the adjusted tool region. Below are further explanations about shape matching.

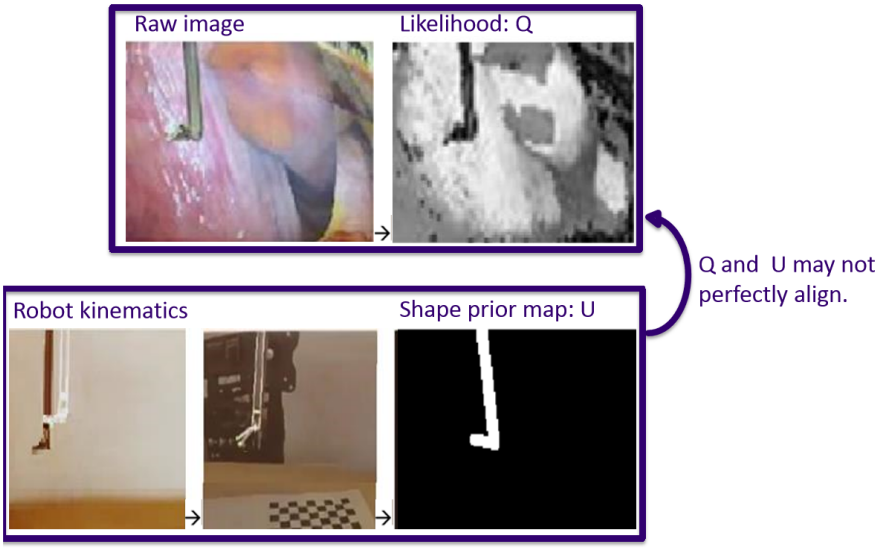

Figure 2.6: These are the two maps of interest during the process of shape matching. On the top is the likelihood map Q generated from color filtering, and the bottom row is the shape prior mask U generated from robot kinematics. The goal of shape matching is to move U around until it aligns well with Q.

2.4.1 U — the shape prior

After getting robot kinematics information, one can project the surgical tool pose back on to the image frame, and create a mask with the white area being the predicted tool pixels, and the background being black, which is shown in Fig.2.7-b. Comparing this mask to the

raw image in Fig.2.7-a., one can observe that they are not perfectly overlapping.

Possible causes for such error include inaccuracy with camera extrinsic calibration, error in the instantaneous robot configuration published from the *ravenstate* ROS topic, or the timing mismatch between the received robot pose versus the perceived image frame, because normally the images takes longer to transmit and display compared to the robot kinematics ROS topic information.

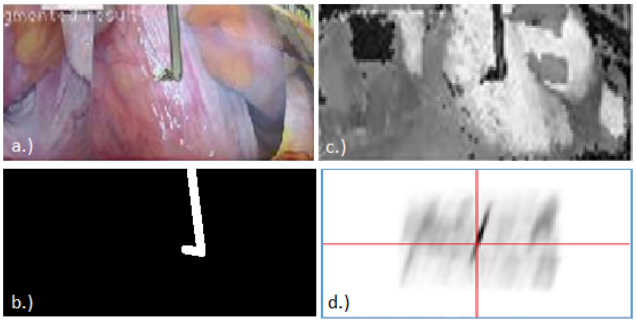

Figure 2.7: The shape matching procedure using DFT. a.) The raw image frame. b.) The shape prior mask derived from robot kinematics. c.) The color filtering result indicating the likelihood of each pixel being tool or non-tool. d.) The energy function of how much the shape prior should be shifted in order to match with the likelihood best.

2.4.2 Q — the likelihood map

To examine the correctness of this shape prior U, the color filter mentioned in the previous section is applied to the raw image frame, which results in a likelihood map Q shown in Fig.2.7-c. In this likelihood map, pixels with colors closer to a surgical tool appear darker, while the ones closer in color to the background tissue appear brighter.

2.4.3 E_t — the energy map

Comparing the shape prior U and the likelihood map Q, one may notice the fact that while surgical tool pixels appear brighter in Q, they are darker in U. This is an interesting observation because we can define the objective function to evaluate how well the two maps align as the sum of pixel-wise multiplication between the two maps. Suppose there are N_R number of rows and N_C number of columns in U and Q, the *energy* value can be written as Eq.2.5.

$$energy = \sum_{r=1}^{N_R} \sum_{c=1}^{N_C} U(r,c)\, Q(r,c) \tag{2.5}$$

Ideally, all the white pixels should correspond to the black pixels on the other map and vice versa. So, the smaller the *energy* value is, the better the two maps align. As mentioned above, the two maps are not always aligned perfectly. Possible misalignments include shifting, rotating and scaling. All the cases will be discussed in the following sections.

Figure 2.8: The optimal offset for U in order for it to align with Q is the vector from the image center of the energy map to the min valued point in the energy map.

Shifting

This is a case where the shape prior is shifted away from the actual surgical tool location by a constant offset on the image coordinates. One naive way of finding out the optimal offset value is to test out all the possible offsets, by doing so, an *energy map* is obtained, where every element $energy\ map(t_r, t_c)$ is the *energy* value with map U shifted by an offset of $t = (t_r, t_c)$, as shown in Eq.2.6.

$$energy\ map\,(t_r, t_c) = \sum_{r=1}^{N_R} \sum_{c=1}^{N_C} U\,(r - t_r, c - t_c)\,Q\,(r, c) \tag{2.6}$$

Suppose the optimal offset value is $\hat{t} = \left(\hat{t_r}, \hat{t_c}\right)$, the *energy* value for this particular offset should be the minimum among the entire *energy map*, as shown in Fig.2.8. Therefore, the Eq.2.7 should suffice.

$$\begin{aligned} \min\,(energy\ map) &= energy\ map\left(\hat{t_r}, \hat{t_c}\right) \\ \arg\min_t\,(energy\ map) &= \left(\hat{t_r}, \hat{t_c}\right) \end{aligned} \tag{2.7}$$

This guarantees an optimal solution because the value t is bounded, but since the procedure is similar to convolution, it is computationally expensive. In this paper[20], an alternative method is proposed using the duality property between the spacial and frequency domain.

Basically, doing convolution in the spacial domain is almost analogous to doing multiplication in the frequency domain. In the frequency domain perspective, Eq.2.6 can be rewritten as the following.

$$energy\ map = \mathbb{F}^{-1}\left(\mathbb{F}_Q \mathbb{F}_U^*\right) \tag{2.8}$$

The computational complexity for the exhaustive search drastically decreases from $\mathbb{O}\left(N^4\right)$ in the spacial domain using Eq.2.6 down to $\mathbb{O}\left(N^2 \log N\right)$ in the frequency domain with Eq.2.8 using *Discrete Fourier Transform* (DFT) or *Fast Fourier Transform* (FFT).

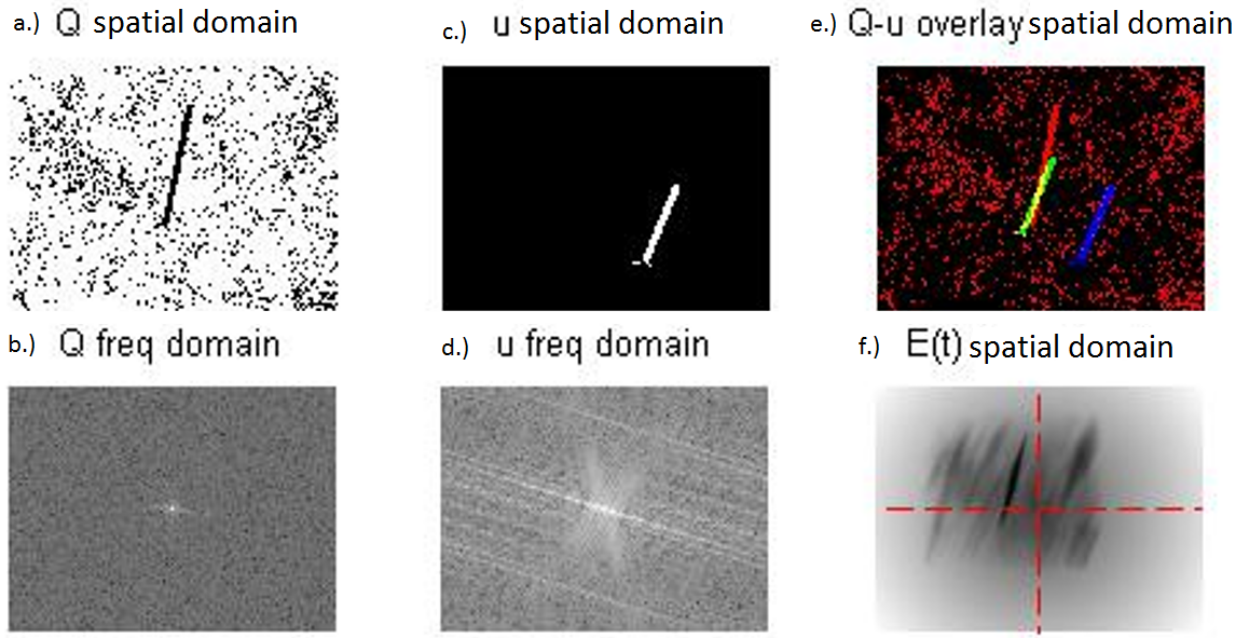

Figure 2.9: This is a simulation of DFT shape matching algorithm in the case of shifting. a.) A noisy likelihood map Q. b.) The Fourier transform of map Q. c.) The shape prior mask U. d.) The Fourier transform of mask U. e.) The result of overlaying Q, U, and post adjustment of U after shifting. f.) The *energy map* used to determine the optimal translational offset.

In Fig.2.9, a simulation is done to prove the concept. Similar to actual implementation, a noisy likelihood map Q is generated and the darker region on the map indicates higher probability of it being part of the surgical tool pixels. Then, there is shape prior U that is close, but slightly shifted, scaled and rotated from the actual tool configuration. Using the

algorithm above, one is able to compensate for the translational offset. From Fig.2.9-e., the red indicates the perceived tool pose from Q, the blue indicates the shape prior U, and the green is the post-translational U, which apparently matches with Q much better.

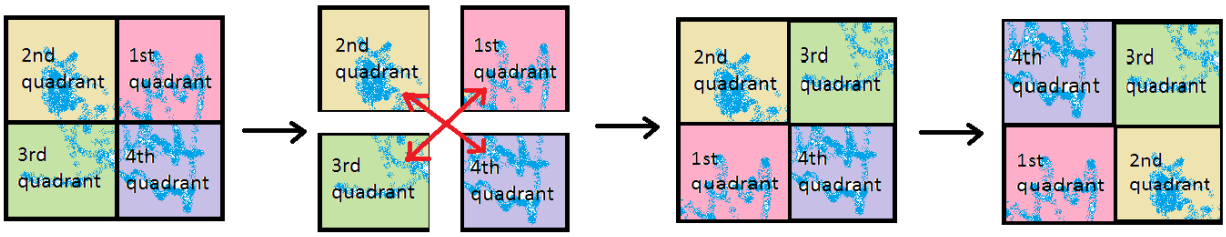

Figure 2.10: The quadrants adjustment for the energy map.

Due to periodicity in Fourier transform, one can place zero offset $t = (0, 0)$ at the center of the *energy map* by swapping the region of image in the first and third quadrant and do the same for the second and fourth quadrant, as shown in Fig.2.10. Hence, the *energy map* will look similar to Fig.2.9-f, where intersection of the red dashed lines at the center of the map indicates the *energy* value with zero offset, and all the other points can be interpreted accordingly. Note that the post translational U is much closer to Q, but the orientation and size are still off. In the following section, this will be resolved using a similar method with DFT.

Rotating and Scaling

Not all misalignment between U and Q result from translational offset, in fact, rotation and scale are also very common. Although it is not intuitive how one can deal with this exhaustive search problem effectively using the same trick by duality between the frequency and spacial domain, it is actually possible. Inspired by this paper[20], consider transforming representation methods of pixels from the Cartesian space (x, y) to the log-polar coordinate system (a, b), where $a = \log\sqrt{x^2 + y^2}$ and $b = tan^{-1}\left(\frac{y}{x}\right)$. In this case, the entry a is correlated with scaling and resizing and entry b is in charge of rotation. Therefore, the exact same translational shifting approach using DFT can be applied.

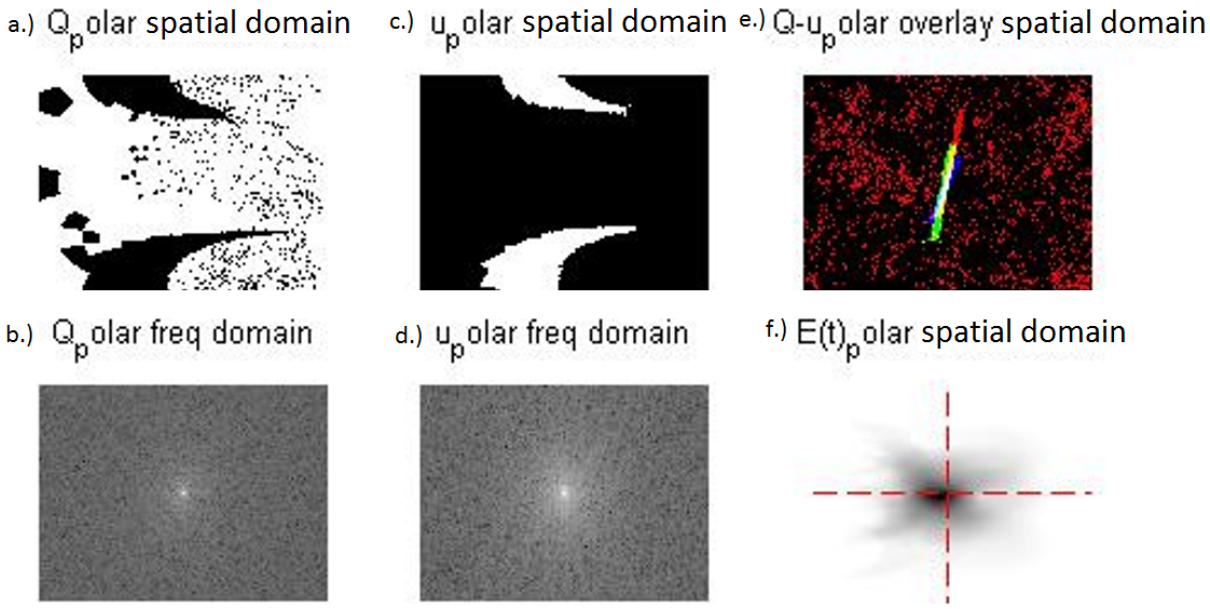

Figure 2.11: This is a simulation of DFT shape matching algorithm in the case of rotation and resizing. a.) The log-polar representation of noisy likelihood map Q. b.) The Fourier transform of map Q. c.) The log-polar representation of shape prior mask U. d.) The Fourier transform of mask U. e.) The result of overlaying Q, U, and post adjustment of U after rotation and scaling. f.) The *energy map* used to determine the optimal rotational and resizing factors.

Continue with the simulation, in Fig.2.11-a. the map Q remapped to the log-polar coordinate system, again a pixel with dark color shows high probability of it a tool pixel. Then, the log-polar representation of shape prior U is shown in Fig.2.11-c. Using the Fourier transform of both U and Q, the algorithm above yields optimal rotational and scaling factors. From Fig.2.11-e., the red indicates the perceived tool pose from Q, the blue indicates the shape prior U, which is also the post-translational U from the previous section, and the green is the new modified U, which matches Q even more.

In log-polar coordinate systems, the horizontal axis is a and the vertical axis is b, which shows the scale and orientation. Suppose the center of the image is the origin, since the raw images Q and U are not circular, the maximum size $a_{k(max)}$ is different for individual orientation b_k, which means there will be some undefined points on the right in Fig.2.11-a

and Fig.2.11-b where the size is big. In this case, the null points should be padded with white in Q and black in U, so that there is no awkward borders near the right edge, and that they would not affect the result for optimal rotational and scaling factors. Alternatively, hamming window can be applied on the log-polar maps to eliminate the complication.

Finally, after many iterations interchangeably between the two sections above, the shape prior U approaches Q and segmentation quality will improve.

2.5 Color Mask

After the shape prior is modified using color filtering, a color threshold value of $Opponent_1$ - $Opponent_2$ is defined to make binary classification of the pixels as either tool or non-tool. The reason for adding this step is because sometimes, the surgical instrument is partially occluded, and although the configuration of the surgical tool and its projection is correct, those pixels that are blocked should still be excluded from the class of tool pixels. Also, due to kinematic errors, the edges may not be exactly aligned between the prior guess and the actual tool pose, and this color threshold can account for this imperfection. Interestingly, since the color threshold is defined by $Opponent_1$ - $Opponent_2$, and both $Opponent_1$ and $Opponent_2$ are linear combinations of the RGB colorspace, the resulting threshold actually forms a plane in the RGB space. In Fig.2.12, this color threshold is applied to the entire color spectrum, assuming lighting (brightness) variations does not cause different classification result, the color spectrum is reduced to two dimensional space in Fig.2.12 where the vertical axis is saturation and the horizontal axis is hue, the classification result is shown. Apparently, the colors closer to red and blue appear to be background tissue and that the colors with lower saturation, which matches the grayish tone of the surgical tool, are classified as surgical tool.

2.6 Dice's Coefficient Analysis

At this point, the image segmentation for surgical instrument is done. To quantify the performance of the segmentation result, the *SrensenDice index*, also known by *Dice's Coefficient*

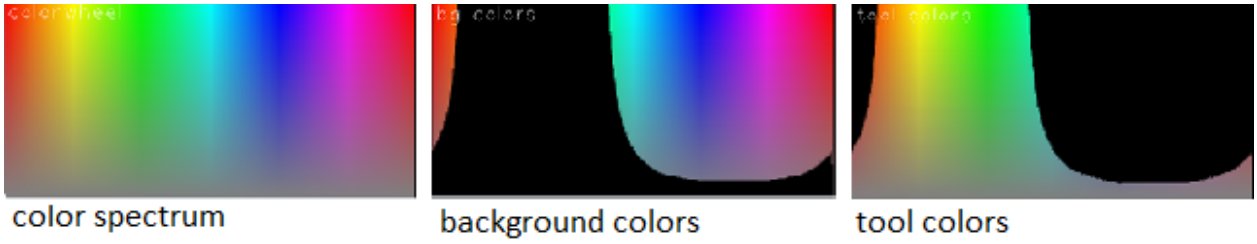

Figure 2.12: The color threshold that does binary classification all the colors in the color space and divide them into either surgical tool colors and non-tool colors.

is used. *Dice's coefficient* is a statistic used for comparing the similarity of two samples, the value for this coefficient ranges from 0 to 1. Suppose $|X|$ and $|Y|$ are the cardinality of the two samples X and Y, the *Dice's Coefficient* is defined as the following:

$$Dice's\ Coefficient = \frac{2|X \cap Y|}{|X| + |Y|} \tag{2.9}$$

Applying the idea to image segmentation, let X be the set of pixels that are labeled as tool using the proposed image segmentation algorithm, and let Y be the set of actual tool pixels, which can be obtained by manually segment the image by hand. Then if the value is 0, that means the segmented region is completely disjoint with the ground truth, and the value being 1 indicates that the segmentation result is completely identical to the ground truth.

2.7 Stereo Rectification Check

In stereo vision, disparity map is the apparent pixel difference or motion between a pair of stereo image. Conceptually the closer an object is to the camera. The greater the pixel difference between left and right image frames it shows, contrarily, if the object is far away, the pixel difference appears trivial between left and right images. This is the key idea to estimating distance and the 3D scene given two stereo images. But before identifying corresponding feature points between left and right images, one need to ensure that the pixel difference only happens horizontally along the x axis. If the two cameras are aligned correctly

to be coplanar, the search for feature points is simplified to one dimension - a horizontal line. In fact, such simplification should be intuitive for human because the brains use binocular disparity to extract depth information from the two-dimensional retinal images in stereopsis, which means that given a known location of a point in the left image, the corresponding point can be found by searching through this horizontal line called epipolar line.

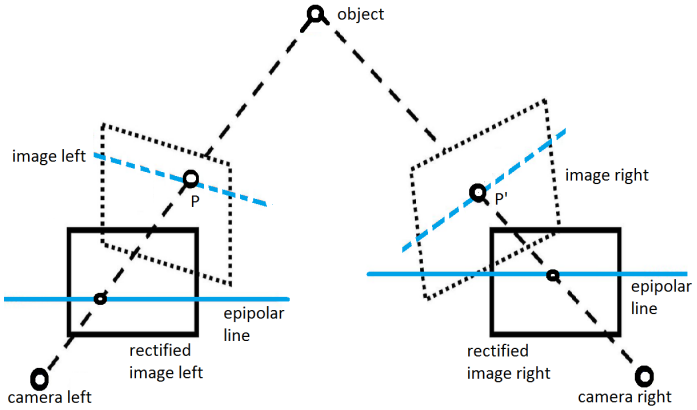

Figure 2.13: The idea of image rectification.

Yet, this only holds true in the perfectly parallel stereo camera setting and that both cameras have no distortion, In general cases, image rectification, an alternative to perfect camera alignment, is needed.

Image rectification is a transformation process used to project two or more images onto a common image plane as shown in Fig.2.13. Which can be derived from an OpenCV function called *stereoRectify*[6]. One way to examine if rectification is done correctly is to overlay the grayscale images from the stereo image pair to the same plot using two separate color channels. If the images are rectified correctly, all corresponding feature points should only contain horizontal shifting. Due to complications during calibration or manufacturing error, the rectified images may not be exactly row aligned. In this case, one can manually shift one image along the y axis by a constant value, in order to ensure good disparity output. This step is important when debugging for a poor disparity map, because there are several disparity parameters one can tune for improving disparity output depending on current

system setup. But if rectification itself is not done correctly, none of the disparity parameter values can correct for the problem.

2.8 Modification for the Disparity Values

In OpenCV, to calculate disparity map, there are *StereoBM* and *StereoSGBM*, which respectively stands for block matching algorithm and semi-global block matching algorithm. The former is tremendously more time effective, but the latter allows for more parameter adjustments and if proper values are chosen, it produce finer disparity results. Since the region of interest is narrowed down to pixels neighboring to the surgical tool tip, real-time implementation is doable using *StereoSGBM*.

Now, assume the stereo image pair is perfectly rectified, disparity value for every point is the exact number of pixel difference between projections on the left and right images. In real world application, this is not always the case, sometimes shifting and scaling of the disparity map is necessary before getting the true disparity. Below are some descriptions respectively on the two sub topics.

2.8.1 Disparity Map Shifting

Though the two cameras should be parallel, in general the right camera can be slightly rotated with respect to the left one, which leads to changes in the reprojection equations. Assume the angle of rotation is really small, one can use small angle approximation to derive the following, in which two cases arise:

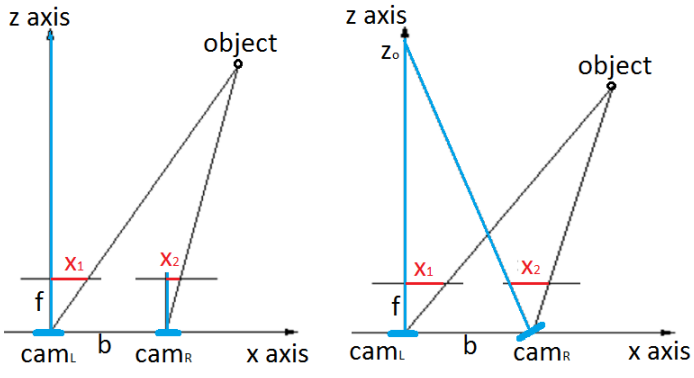

Figure 2.14: Affects on disparity values when two cameras are parallel and non-parallel.

Case 1: Perfectly Parallel

In a standard stereo setting shown the left side of Fig.2.14, there are two cameras marked as cam_L and cam_R, the vertical blue lines are the z axes for each camera, f is the focal length and b is the baseline between the stereo camera pair. x_1 and x_2 are the x coordinate of the projection points of the object onto left and right images respectively. Since the two cameras are perfectly parallel, the object location in 3D can be derived in Eq.2.10.

$$\begin{aligned} Z &= \left(\frac{bf}{x_1 - x_2}\right) \\ X &= \left(\frac{Zx_1}{f}\right) \\ Y &= \left(\frac{Zy_1}{f}\right) \end{aligned} \tag{2.10}$$

Case 2: Rotation About y Axis

On the right side of Fig.2.14, the right camera is rotated on slightly around the y axis by an angle θ, so they still share a common XZ plane. Suppose the z axis of two cameras intersect in the fixation point $(0, 0, Z_0)$, where Z_0 can be either positive or negative. Under small angle approximation, $Z_0 = {}^{b}/_{\tan\theta}$, and that one can still assume the right image plane to be nearly parallel to the left image plane. The object location in 3D can be derived in Eq.2.11.

$$\begin{aligned} Z &= \left(\frac{bf}{x_1 - x_2 + \frac{bf}{Z_0}}\right) \\ X &= \left(\frac{Zx_1}{f}\right) \\ Y &= \left(\frac{Zy_1}{f}\right) \end{aligned} \tag{2.11}$$

Now comparing Eq.2.10 and Eq.2.11, only the denominator of the Z component is different. And recall the definition of disparity is the pixel difference $x_1 - x_2$. The whole thing will be identical to Case 1 if the raw disparity is shifted by a scalar value ${}^{bf}/_{Z_0}$. This is the part where one potentially needs to shift the disparity by some offset. Interestingly, the offset can be derived by doing some experiments, which will be covered in the following section.

Case 3: Rotation About x Axis

In this case, the right camera rotates around X axis by an angle of θ. This only influences the Y coordinate during re-projection. The object location in 3D can be derived in Eq.2.12.

$$\begin{aligned} Z &= \left(\frac{bf}{x_1 - x_2}\right) \\ X &= \left(\frac{Zx_1}{f}\right) \\ Y &= \left(\frac{Zy_1}{f} + Z\tan\theta\right) \\ &= \left(\frac{Z\left(y_1 + f\tan\theta\right)}{f}\right) \end{aligned} \tag{2.12}$$

Comparing Eq.2.10 and Eq.2.12, one can tell that the re-projection equation are actually the identical simply by adding an offset of $f\tan\theta$ to the y_1 component. In other words, this can be fixed by simply shifting the y values a little before re-projection to 3D.

Case 4: Rotation About z Axis

This situation is often fixed before computing the disparity by simply doing rotation about the image center, so normally problem like this would not occur.

2.8.2 Disparity Map Scaling

This is an often uncared-for detail when using OpenCV disparity computation function *StereoSGBM*. Basically the output disparity map is a 16-bit signed single-channel image same sized as input images. It contains disparity values scaled by 16. To get true disparity, pixel-wise scalar multiplication of $1/16$ is needed. This is a extremely simple step, but can generate great error if not taken care of.

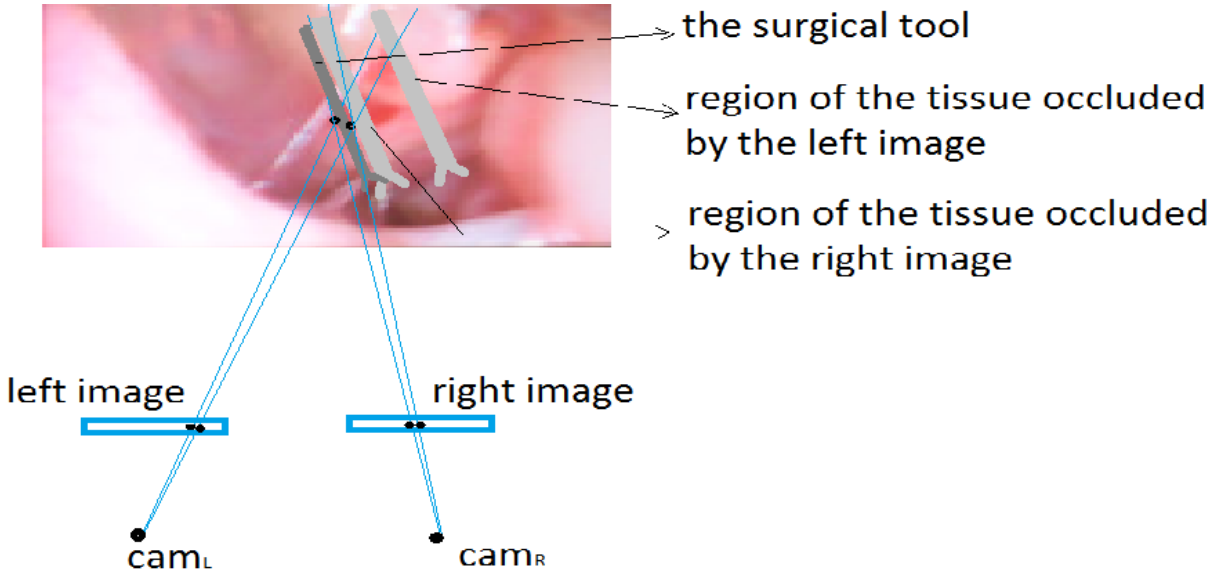

Figure 2.15: The cause of disparity inaccuracy in surgical environment.

2.9 Disparity Post Filtering

In texture-less, half-occluded regions or where there are discontinuities in depth, holes or broken points in disparity map is common. This is a result of *StereoBM* and *StereoSGBM* using techniques to predict potential inaccurate disparity values on the image based on the above mentioned factors and disregard the disparity result on those pixels, which leads to our disparity map being semi-sparse.

As one can see in Fig.2.15, the sparsity in the disparity map is particularly not negligible in this application because surgical tool shafts are generally very thin, and so the portion of background tissue occluded by the surgical tool can be almost disjoint between the left and right images, additionally, depth discontinuities also happen near the tool edges. Thus inaccuracy in disparity values occurs around the surgical tool contour, which unfortunately lies exactly within the region of interest.

To recover from that, two filtering methods are tested, including the max filtering and

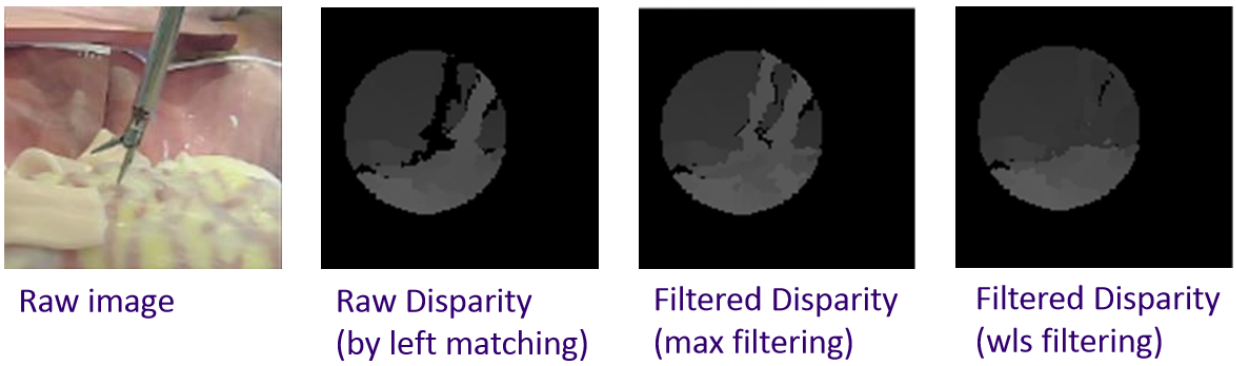

Figure 2.16: Raw disparity map and the two post filtering schemes tested.

weighted least square post filtering. The raw disparity was obtained using left matching which means that we use the left image to match with the right image, but we can actually do the reverse and get a right matching disparity. Then the max filtering is basically comparing the disparity results from left and right matching disparity and find pixel-wise maximum. On the other hand, the weighted least square post filtering *createDisparityWLSFilter* in OpenCV is applied to align the disparity map edges with those of the source image, and then propagate disparity values from high to low confidence level. Both post filtering approaches are capable of reducing broken points significantly in many cases if proper parameter values are chosen.

Fig.2.16 shows and example of a disparity map from left matching method and results from the two post filtering schemes. As one may notice, in the raw disparity, the tool location appears brighter because it is closer to the camera, but there is a phantom of the tool on the left of the actual tool. In fact, similar but reversed result is shown if we try the right matching disparity, only this time the actual tool appears on the left and the phantom of the tool can be found on the right. In this case, we can see how the max filtering yields two surgical tools instead of one, which can be a little misleading to our understanding. Now if the weighted least square post filter in applied, the main observation is that it sort of ignores the tool entirely due to large disparity value difference between left and right matching disparity around the surgical tool region, so those pixels are thus considered noise

and gotten rid of. This filtering method although does not provide useful information of the surgical tool depth, but if one's aim is to reconstruct the background tissue topology, this filtering scheme is actually pretty helpful since it automates the part where one needs to neglect disparity information of the surgical tool pixels.

Chapter 3

EXPERIMENT AND RESULT

The experimental setup is shown in Fig.3.1. Two USB cameras are installed on a 3D-printed stereo camera mount which is then fixed beside the Raven Surgical Robot System. The back ground is a big cardboard printed with endoscopic images to simulate a surgical operation scene. Also, during analysis tissue deformation, the artificial tissue is used to perform tool-tissue interaction experiments. On the software side, this work is mainly done with C++ and ROS, with some post experiment data analysis using Matlab. Below are explanation for the experiment results throughout this work.

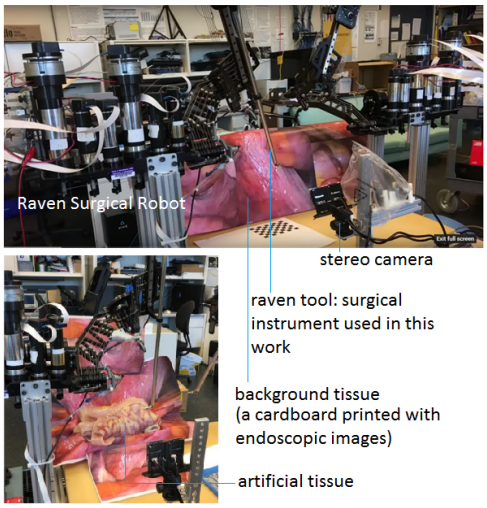

Figure 3.1: Environment setup for experiments in this work

3.1 Robot Kinematics Prior

3.1.1 3D Offset for the Robot Pose

For stage 1, the aim is to do surgical tool segmentation of the image frames obtained from the stereo camera set. After the camera is calibrated and the extrinsic matrix is found, the real-time robot pose published by the *ravenstate* ROS topic is projected onto the image and one can yield the pixel coordinates of all the joints of the robot. Fig.3.2 illustrates a stick figure of the projection on the image.

Due to minor error each time Raven initializes and error with camera extrinsic calibration using PNP algorithm. The true projection and predicted projection may not align perfectly. Two solutions are presented to fix the problem. First, using shape matching with DFT, one can compensate for the error, which will be covered in a later section. On the other hand, Although it may not be obvious in the 2D coordinates projection, it is often merely a matter of a translational offset in 3D space. When we move the surgical tool around, the true projection and predicted projection do not appear to have a constant difference, but it is actually a constant in the world coordinates, or in this case, the raven base frame.

So, after receiving *ravenstate* ROS topic information about robot kinematics in 3D, an user-defined offset $(x_{offset}, y_{offset}, z_{offset})$ is added before projecting onto the image frame. The predicted projection after adding the offset is shown on the right side of Fig.3.2.

3.1.2 Thickness Prediction

In image segmentation, having the shape prior in stick figure is not very helpful, because without knowledge in width or thickness, it is hard to form a shape prior mask of pixels containing the surgical tool. Fortunately, since the physical thickness for the surgical tool is known, in using the triangle similarity the perceived thickness on the image plane can be derived.

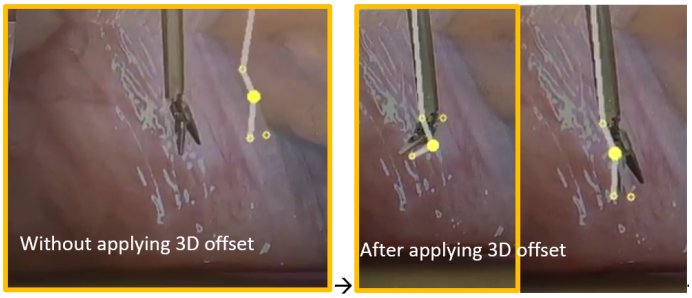

Figure 3.2: On the left is the raw projection of surgical tool joints to the image plane, using robot kinematics. On the right, an offset in 3D is applied before projection to compensate for the error.

Suppose one has object with a known width W, which is placed some distance D away from the camera with a focal length of F. The apparent width in pixels would be P in the following equation Eq.3.1.

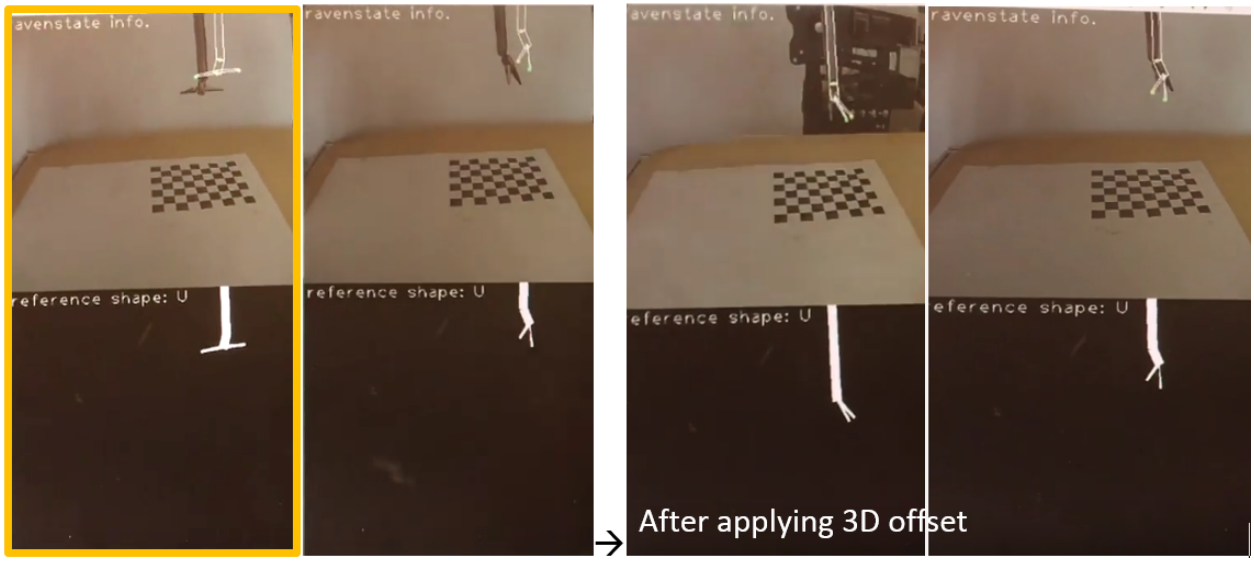

Figure 3.3: The raw and modified projection of surgical tool on the image including information about surgical tool width or thickness.

$$P = {}^{FW}/_{D} \tag{3.1}$$

Using this triangular similarity equation, Fig.3.3 shows the projection of the entire surgical tool, including the thickness information, and then one can also obtain the shape prior mask U from the predicted projection. This shape prior mask in shown on the bottom row of Fig.3.3. And as mentioned above, on the right half of Fig.3.3 a user-defined 3D offset is added to improve correctness of the prediction. Non static inaccuracies are further dealt with in the next section.

3.2 Shape Matching with DFT

Even with the constant offset, the predicted projection still may not align with the true projection nicely, which is mainly due to the latency difference between transmitting images and robot kinematics. That is, the *ravenstate* ROS topic often returns instantaneous robot pose prior to the image frame of that time instance arrives. And so if the robot arm is moving

around and having bigger motion, the predicted projection tends to move faster than the true projection.

This error is harder to deal with because it is dependent on the CPU speed, the extent of instantaneous robot motion, and is not static over time. Thankfully, an effective method to finding optimal shifting, scaling and rotational factors is proposed using DFT in the frequency domain.

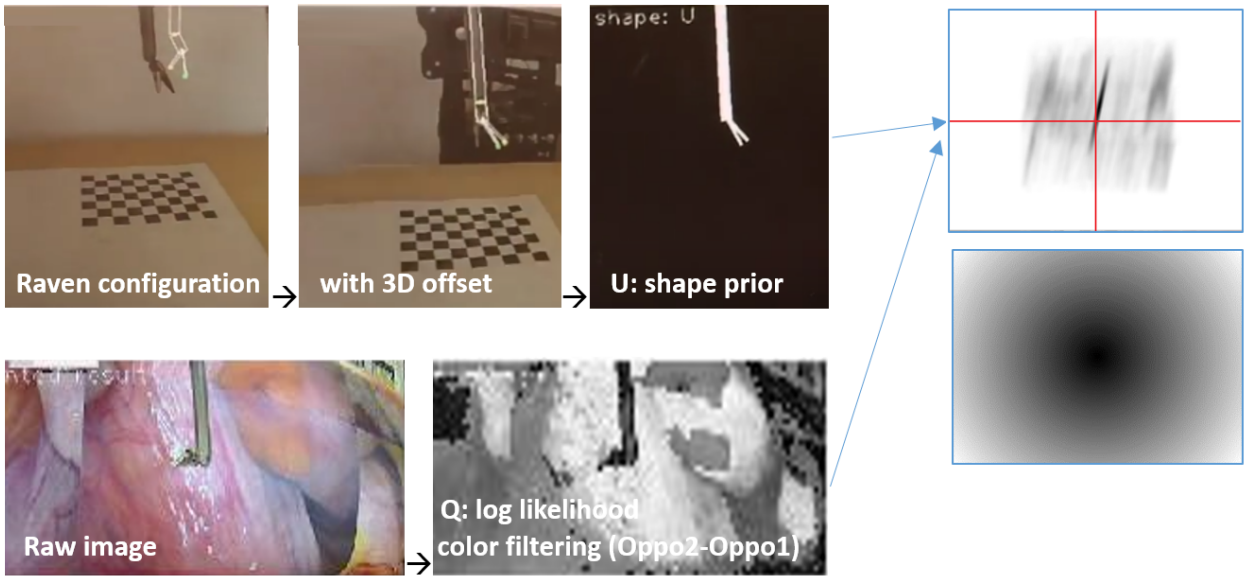

Figure 3.4: This is the flowchart for procedures in getting a nice shape prior U that aligns nicely with Q using DFT shape matching method and 3D static offset.

Hence, the flowchart of creating a good shape prior is shown in Fig.3.4. On the top left corner, one projects the robot kinematics directly to the image, but since it does not align well, the user can include a non-zero offset in the 3D space to the system, so that a modified version of the predicted projection is generated. Next, one can get the shape prior U based on the predicted projection. On the bottom left, The log likelihood map Q is formed using the color filtering scheme described previously in this article. Combining the two maps U and Q, Eq.2.6 is able to find an optimal offset on U that matches the dark region in Q and the bright region in U the best. In order to achieve this, one find argument min of the *energy*

map interchangeably in the Cartesian and log-polar coordinates representation of maps U and Q, which results in optimal translational, scaling and rotational factors that make U approach to Q.

Such *energy map* is shown on the top right of Fig.3.4. But there may be regions of the tissue where Q map still appears dark, which may confuse the DFT computation, and even falsely result in incorrect translational, scaling and rotational factors. In this case, since a static 3D offset is already applied, only small translational, scaling and rotational factors are considered. Thus, a penalty map on the bottom right of Fig.3.4 is multiplied with the raw *energy map*, so when finding minimum *energy* through the entire map, the points closer to the origin at the center are more likely to be picked. Once all these are done, a good shape prior U is ready for use.

3.3 Color Mask

At this point, a shape prior mask U that aligns nicely with the likelihood map Q is derived. But considering the possibility for partial occlusion, and some trivial difference near the edges, not all the tool pixels in U are true tool pixels in Q. This is why a final color mask is used. As illustrated in Fig.3.5, there are four steps to turn the nicely aligned shape prior U into the actual binary segmentation result.

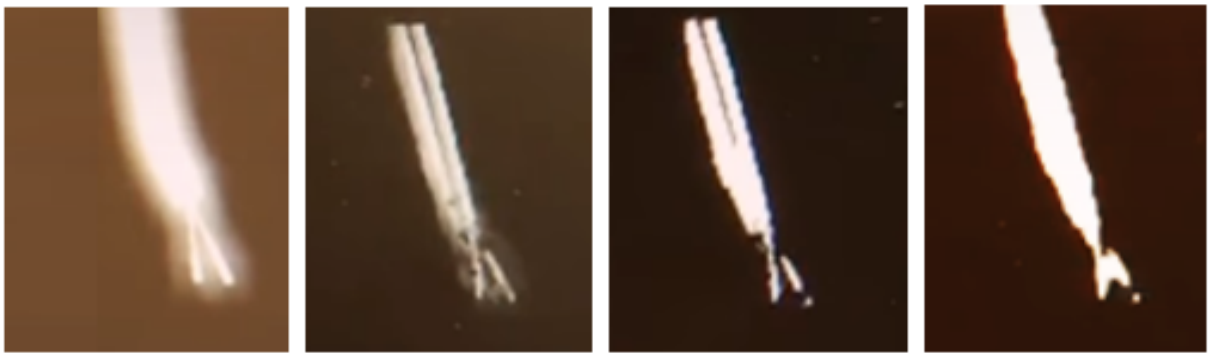

Figure 3.5: Here is the process of using a color mask to do binary classification in determining whether each pixel within the white part of the shape prior mask U belongs to the tool class or the non-tool class.

3.3.1 Border Expansion Step

In this step, shown in the left most figure in Fig.3.5, the borders are slightly expanded outward to tolerate for trivial edge misalignment. It is done using OpenCV functions *dilate* and *blur*[6].

3.3.2 Color filtering Step

This dilated shape mask is then processed using color filtering, which is the same as algorithm of generating log likelihood mask Q. This step ensures that even within the shape prior region, if it is partially occluded by tissue, those pixels will be disregarded. The result from this step can be found on the second left most figure in Fig.3.5.

3.3.3 Binary Thresholding Step

A binary threshold is set to binarily determine whether each pixel belongs to the tool or non-tool class. It is done using OpenCV function *threshold*[6] with the parameter *cv::THRESH_BINARY*. Afterwards, the result is shown in third figure from the left of Fig.3.5.

3.3.4 Noise Elimination Step

As one may notice, Step 2) creates a undesired broken part splitting the surgical tool vertically in half from the middle. This is mainly due to the color filter incapability of classifying pixels within the reflection on the metallic surface of the surgical tool. It is done using OpenCV function *morphologyEX*[6] which is known to eliminate the black spots within a white region or vice versa. Finally the segmentation result is shown on the right of Fig.3.5.

The OpenCV function parameters can be tuned on-line by the users. And the four steps together forms the color mask that eventually produces surgical tool segmentation result.

3.4 Image Segmentation Result

This is a step by step result of work for stage 1 in Fig.3.6. Real-time imaged based surgical tool segmentation with robot kinematics shape prior is achieved. This can be done at close to 5 to 6 Hz using a regular CPU.

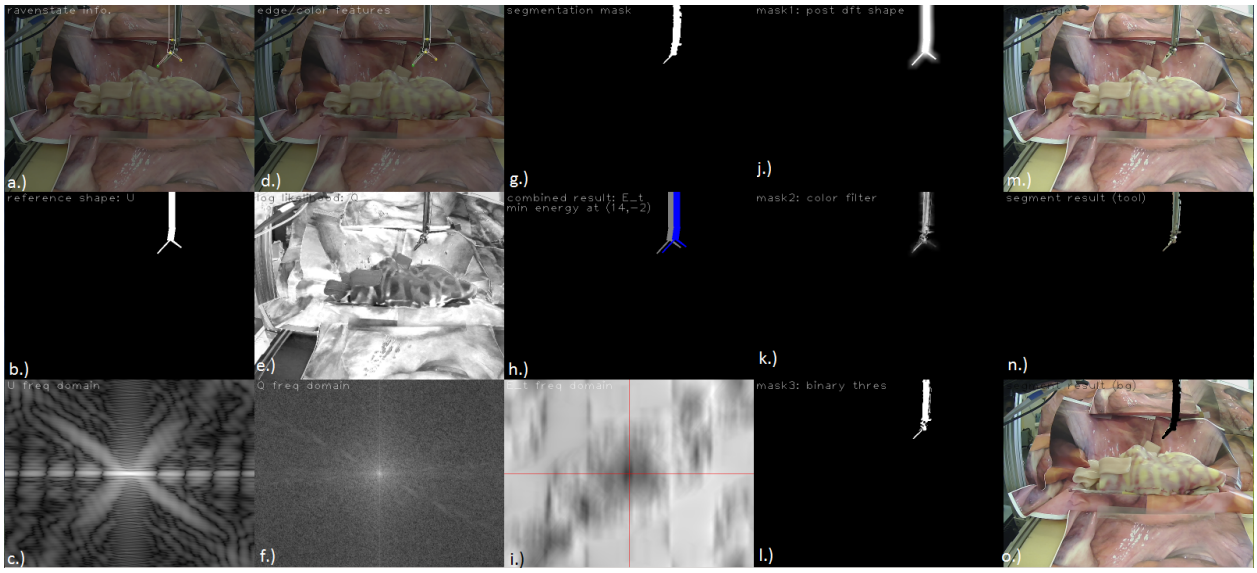

Figure 3.6: This is the segmentation result with each individual step displayed in detail.

The figure in a.) is the robot kinematics information with a static 3D offset added along with thickness estimation. b.) is the shape prior U generated from a.). The log likelihood map Q resulting from color filtering is shown in e.). Then, discrete Fourier transform is performed on U and Q respectively, which yields the magnitude plot in the frequency domain displayed separately in c.) and f.). Afterwards, i.) shows the *energy map* multiplied with the penalty map. The center of the penalized *energy map* where two red lines intersect indicates zero offset. And by finding the minimum value in the penalized *energy map*, one can find the optimal translational offset to U so that U and Q align the best. h.) shows the shape prior map U before and after shifting.

Finally, the final color mask is applied to the nicely aligned shape prior U. The color

mask procedure consists of four steps - border elimination, color filtering, binary threshold and noise elimination. Which are shown individually in j.), k.), l.) and g.). And so the binary classification of tool and non-tool pixels is shown in g.).

Fig.**??**-m. is the raw image frame, and overlaying the binary classification result with it, one can get the foreground and background segmentation output in n.) and o.).

3.5 Dice Coefficient Analysis

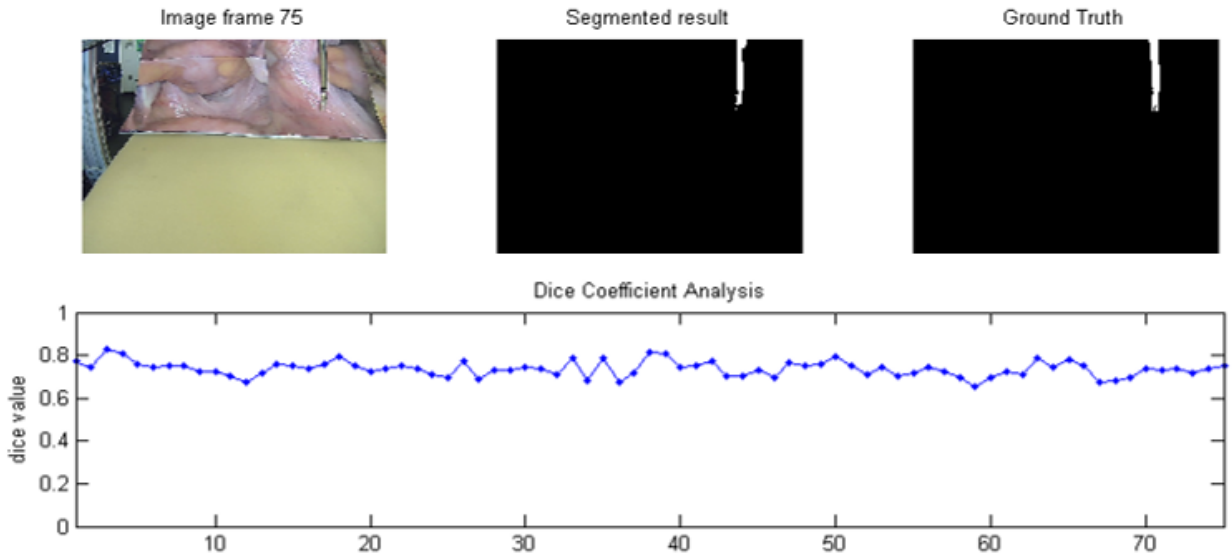

Figure 3.7: The dice coefficient analysis of surgical tool segmentation result overtime.

Dice coefficient can be used as a tool to quantify the performance of an image segmentation algorithm. To do that, a video is recorded with Raven surgical robot moving around and the surgical tool segmentation code running. 75 linearly spaced sample image frames are selected and manually segmented. Comparing the segmentation resulting from the algorithm and the ground truth with manual segmentation. The dice coefficients in each individual frame overtime are shown in Fig.3.7. The horizontal axis is the index of the 75 image frames, and the vertical axis shows the dice coefficient. One can observe that the average dice coefficient is around 0.75 to 0.8.

On the other hand, it is interesting to see if the dice coefficient value has dependencies

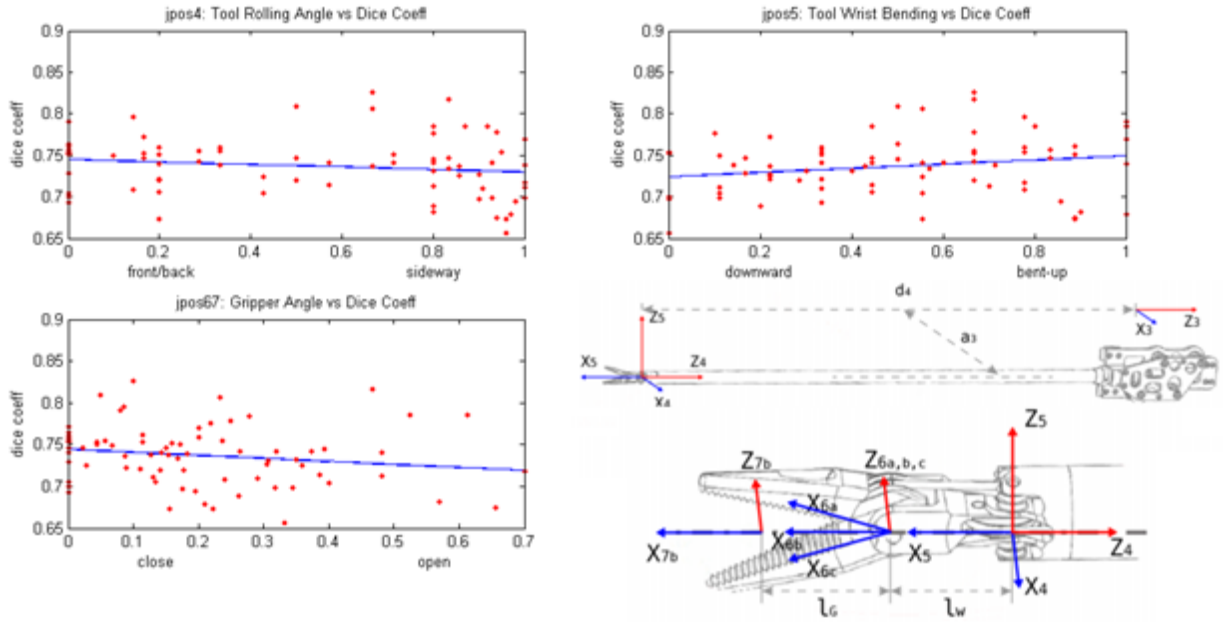

Figure 3.8: The correlation between each Raven tool joint angle and dice coefficient output.

on the surgical tool pose. In other words, there may be certain tool poses that increase the segmentation accuracy. So the correlation plot between the angle of each of the four tool joints is drawn in Fig.3.8, and a regression line is displayed. Although, slight correlation exists, one can claim in this work that the pose of surgical tool is almost independent to the segmentation performance, which is actually a desirable outcome.

3.6 Color Statistics

Color filtering in this work stands an important role in modifying the surgical tool pose given by robot kinematics information. Currently, the color filter is a weighted sum of the Opponent, RGB and HSV entries, which is fast in computation but relatively simple. If one collects a huge amount of images from surgical operations and do statistical analysis on the probability of occurrence that each color in the entire color space being a tool or non-tool pixel, a discriminative function specifically designed to distinguish tool and non-tool colors can be generated. Moreover, if one develop a specialized transformation matrix that

transforms the colored image into the likelihood of each pixel being tool or non-tool, it will have great potential contribution to all related work on surgical instrument segmentation. And for this work, the color filtering scheme may be further improved.

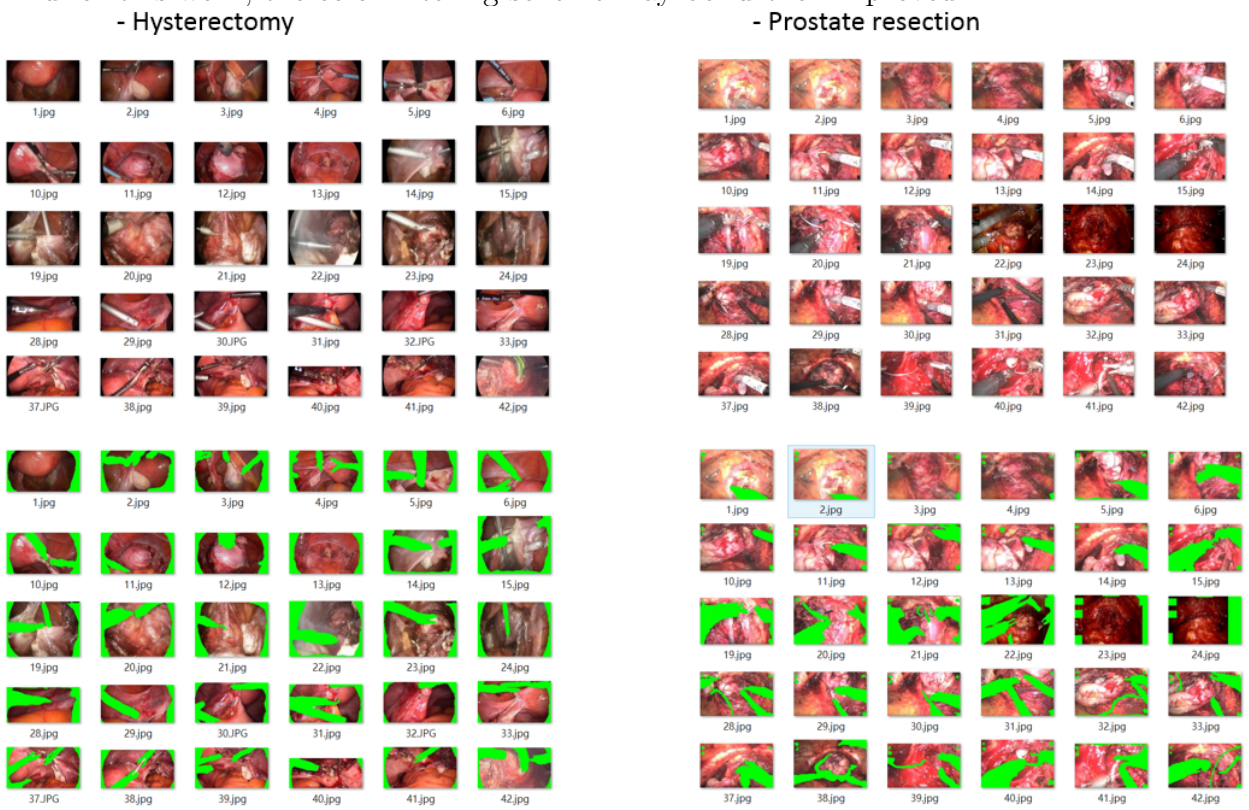

Figure 3.9: Manual segmentation for non-tool pixels in surgical images from Hysterectomy and Prostate Resection operations.

These are a few motivations of do the following analysis. So, tool pixels in this work are collected by recording numerous videos are manually segment the image frames. But since the background in use is artificial, it has less generalizable to real applications. Alternatively, endoscopic image frame from minimally invasive surgery videos are used for collecting the non-tool pixel data. According to sources, two of the most common robot-assisted minimally invasive surgeries include Hysterectomy and Prostate Resection. Thus, a total of 75 images of different video sources are selected from each of the two types of surgical procedures. Partially shown in Fig.3.9, the images are segmented by hand, and probability map is created for all colors in the color space being a non-tool pixel.

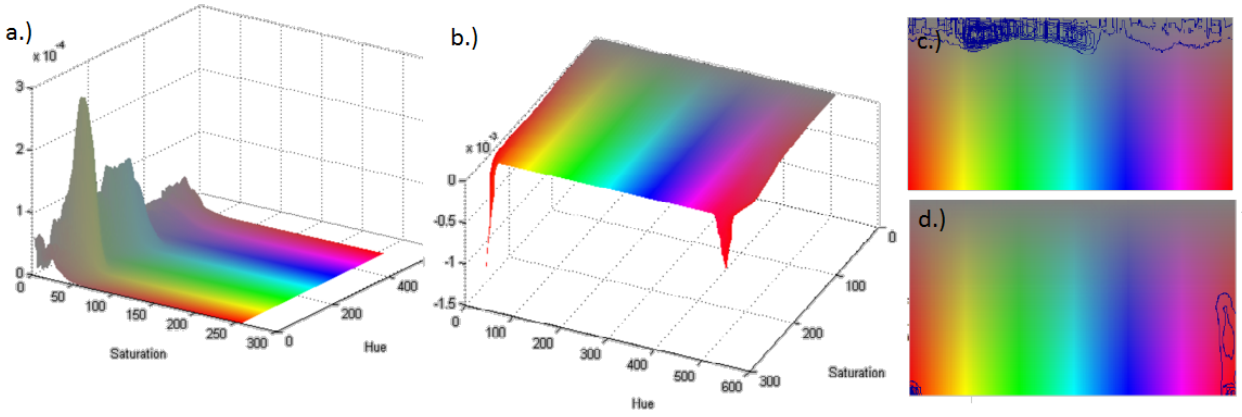

Figure 3.10: The probability of occurrence that each color in the color space being tool or non-tool. a.) The 3D probability map that each color being a tool pixel. b.) The 3D probability map that each color being a non-tool pixel. c.) The 2D terrain representation for a.). d.) The 2D terrain representation for b.) Note that the probability value is negated so that a discriminative function can be generated by dividing a.) and b.).

Fig.3.10 shows the statistical result. For each color, the greater the value is in a.) indicates higher chance that it is a tool pixel, and the smaller the value is in b.) indicates higher chance that it is a non-tool pixel. 2D representation of a.) and b.) are shown in the form of terrain plots shown in c.) and d.).

3.7 Image Rectification

The second stage of this work is depth map rendering. This is when stereo vision is used. And the first step in computing the disparity map is image rectification. In Fig.3.11-a, the left and right rectified images are individually plotted on the blue and yellow channels. Note that the corresponding feature points in the raw rectification image pair is not nicely aligned in the vertical axis, which can be caused by error during camera calibration or manufacturing imperfection of the cameras. This causes bad disparity output, shown in Fig.3.11-c, since matching feature points failed to be paired along horizontal epipolar lines. Interestingly, this offset in the vertical axis stays constant over time and space, so we shift one of the rectified image until it is row aligned with the other, which yields Fig.3.11-b. The disparity map in

Fig.3.11-d is significantly improved afterwards.

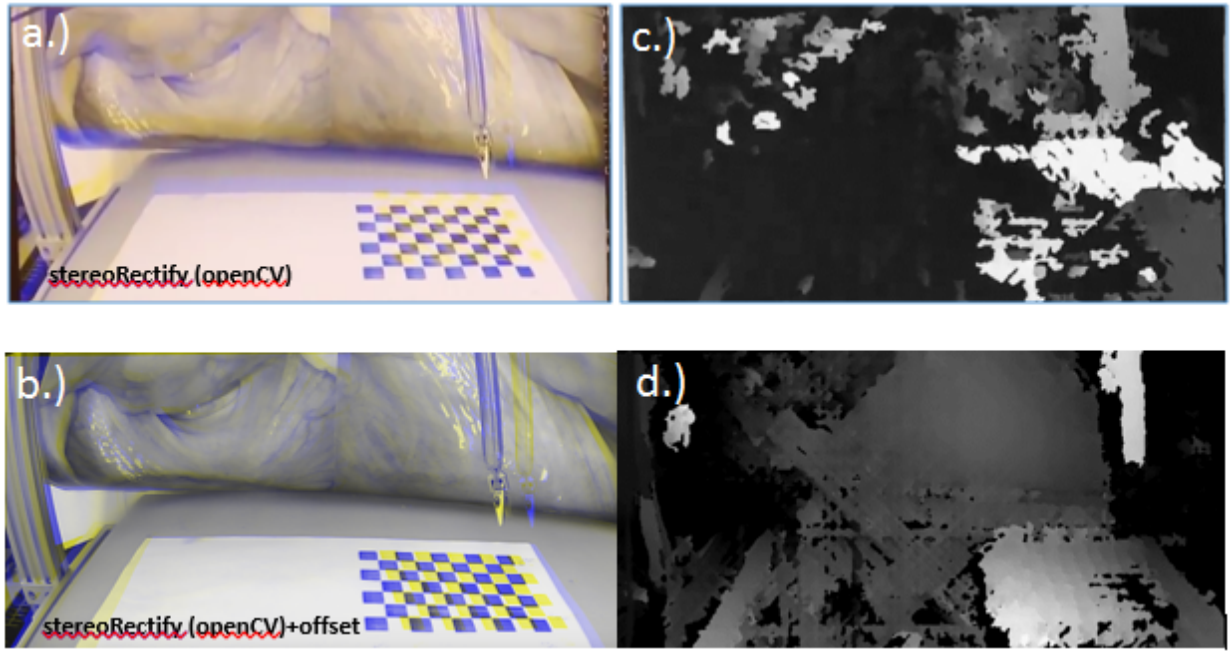

Figure 3.11: The accuracy of rectified image pairs affects the quality of the disparity map. a.) The raw rectified image pair that are not row aligned. b.) The modified rectified image pair. c.) The poor disparity result from a.). d.) The improved disparity result generated from b.).

3.8 Disparity Correction

At this point, although the disparity map seems reasonable with closer objects being bright and distant objects being darker in color, one is yet to determine if the disparity values are actually correct. From the discussion in earlier sections, after scaling the disparity values by $1/16$, it is possible that those values are still off by a constant offset. Fig.3.12 shows the experiment used to verify whether the disparity values are correct or not.

In the experiment, the aim is to find the constant offset in disparity due to one camera slightly rotated around the y axis away from the other camera, so that they are not perfectly parallel. This theory is described in math form in Eq.2.11.

On the top right corner of Fig.3.12, a few points are marked on a white cardboard, and

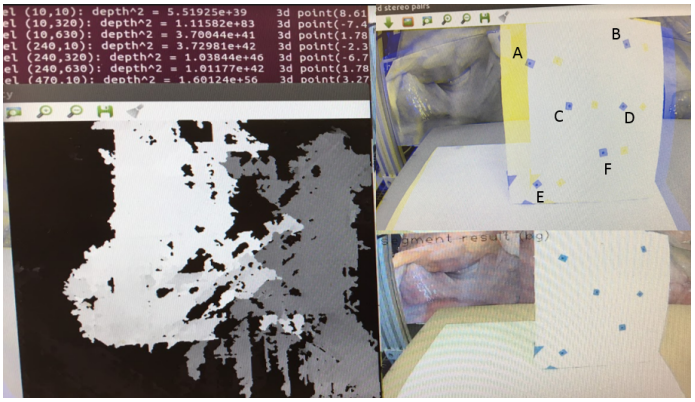

Figure 3.12: The experiment to correct for a shifted disparity value.

the blue and yellow dots respectively indicates the marker location viewed from the left and right camera. Since both images are rectified, the pixel difference between corresponding blue and yellow markers should only be along the horizontal axis. Here is the data collected from the experiment in TABLE.3.1.

pt	(x_{left}, x_{right})	d	Z	$Z_{(true)}$	$d_{(true)}$	Δ
A	$(386, 442)$	56	336	190	99	43
B	$(571, 608)$	37	509	230	81	44
C	$(456, 507)$	51	369	200	94	43
D	$(560, 60)$	42	448	220	85	43
E	$(401, 436)$	35	538	230	81	46

Table 3.1: This is a table showing experiment data for disparity correction by finding a constant disparity offset.

The first column shows the marker index, the second column is the x entry of marker locations in the left and right rectified images. On the third column, disparity d is computed directly by $(x_{right} - x_{left})$, and using the equation $d = {}^{bf}/_{Z}$, one can obtain the computed distance to the image plane Z, with baseline $b = 40\,(\text{mm})$ and focal length $f = 471$, which is shown

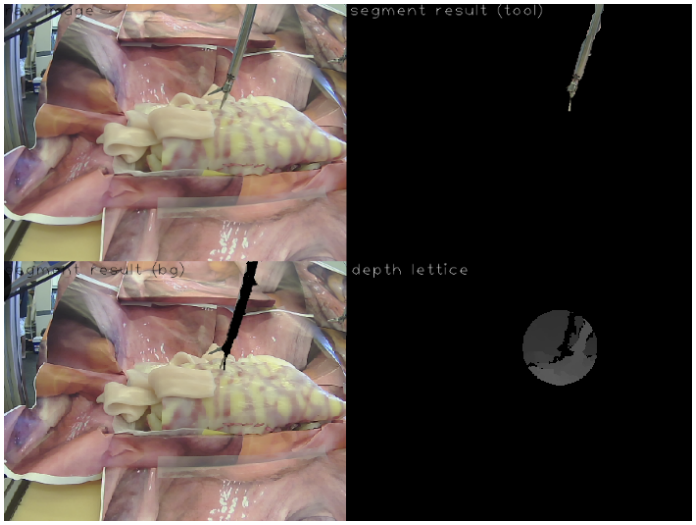

Figure 3.13: The disparity map within region of interest.

on the fourth column. By measurement, the true distance to the image plane is recorded in the fifth column, and using the same equation above, one can compute for the true disparity values that should appear on the disparity map, which is listed on column six. Finally, the last column shows the difference $\Delta = d_{(true)} - d$. Note that the quality of disparity map on the left side Fig.3.12 is poor, but this is absolutely normal because the cardboard on the right only contains a few features, so it is hard to obtain accurate disparity based on it. This is why disparity values are directly derived based on pixel differences between left and right rectified images.

At this point, judging from the average value of Δ, one can claim that there is indeed some rotation around the y axis and that the two cameras are not exactly parallel, and the value we should shift the disparity by is around 45. Derivation of the exact value to shift the disparity by is shown in the following.

In TABLE.3.2, consider different points in 3D that project to the image plane at the same point at the left rectified image center. The distance from each of these points to the camera

pt	(x_{left}, x_{right})	d	Z	$Z_{(true)}$	$d_{(true)}$	Δ
1	$(320, 395)$	75	251	155	121	46
2	$(320, 376)$	56	336	184	102	46
3	$(320, 360)$	40	471	219	86	46

Table 3.2: Disparity correction experiment on points with various distance from camera.

is different, and so the disparity values should also be different. Then one can tell that all the perceived disparity value from our algorithm is 46 units smaller than the true disparity measured manually by hand. After this observation, it is determined that the disparity map should be modified by adding a scalar offset of 46 for every pixel.

3.9 Disparity Post Filtering

So far, the disparity values are correct, but there are holes and broken pixels scattered within the disparity where the corresponding matching points cannot be found. In Fig.3.13, the top left corner shows a raw image, the bottom left and top right are the segmentation result of the background and foreground, which is developed in stage 1. And on the bottom right corner is the disparity. The disparity computation can be time-consuming, so only the circle region resulting from projection of a sphere centered at the surgical tool tip onto the image plane is considered. To further improve the quality of the disparity map and reduce the number of broken pixels, weighted least square filtering can be applied.

Fig.3.14-a is the raw image and Fig.3.14-b is the raw disparity derived from regular left matching disparity. The right matching disparity can be done by flipping both the left and right images horizontally, then feed the flipped images to the part of the code in reverse order, then flip the output disparity map back. And the weighted least square filter is actually an algorithm where it compares the left and right matching disparity and infer the missing information from one another. The result from weighted least square filtering is

shown in Fig.3.14-d. Note that by doing so, the surgical tool itself is almost non-visible from the filtered disparity, because it is a thin metal stick compared to the background and by left and right matching, the projection of surgical tool onto the left and right images might be completely disjoint. And so the right and left matching disparity would not agree in values near the surgical tool region, which lead to those values being considered as noise and replaced with the background disparity values. Since it sort of ignore the surgical tool, the result from wls filter does not seem promising but it may be useful in force rendering when we actually care only about the tissue deformation, and by doing so it automatically disregards the tool pixels.

Another naive approach to get rid of broken pixels in disparity is choosing the max between right and left matching disparities. It is displayed in Fig.3.14-c. And interestingly, two projections of the surgical tool is visible on the resulting disparity. However, this might cause complications when trying to reconstruct the scene, so the method is not adopted eventually.

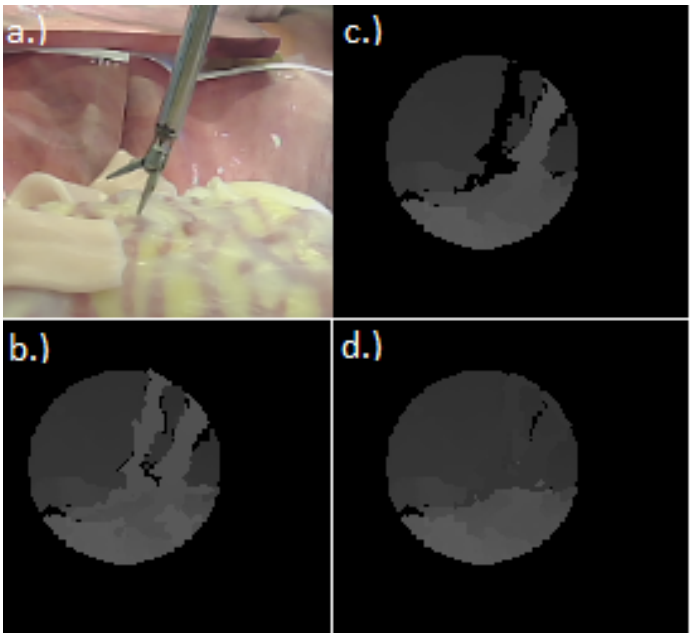

Figure 3.14: Methods of reducing broken pixels in disparity.

3.10 Depth Map Result

This is the real time disparity computation when the surgical tool gradually approaches the tissue surface and eventually contacts with it. In Fig.3.15 the circular region of interest moves along with the surgical tool and the deformation of the tissue is actually visible at the right most frame.

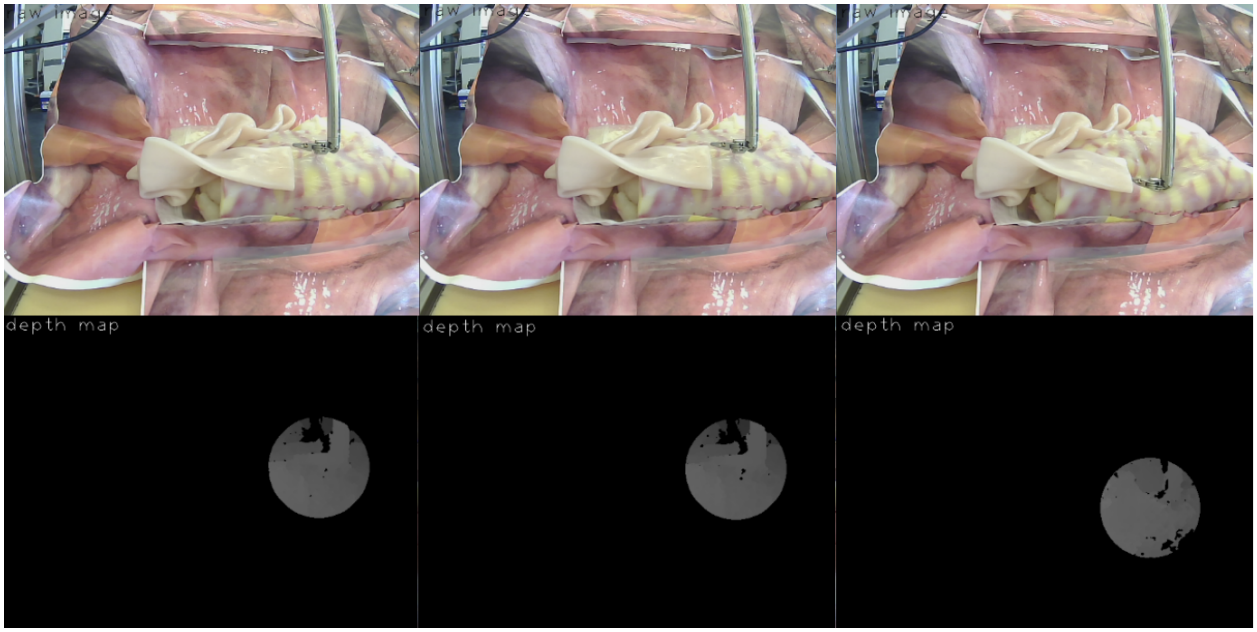

Figure 3.15: The disparity changes in subsequent image frames during tool-tissue interaction.

Chapter 4
CONCLUSION

In this work, a vision-based surgical instrument segmentation algorithm with robot kinematics prior is proposed. Then using stereo vision, one can get the disparity map and further compute the depth map and deformation of the tissue near the tool-tissue interaction point. Having the deformation, one is able to solve for the applied force. In the future, in-depth force rendering method and haptic implementations will be proposed.

Before staring this work, some literature review was done to gather the current existing research in this field. In particular, I was looking for other vision based marker-less surgical instrument segmentation with robot kinematics prior. These three papers were found [3][4][9]. A few contributions that my work is distinct from those existing research are listed as follows:

4.1 Real-time

My work is applicable in real-time with an average processing rate of 4.99Hz including both image segmentation and depth rendering. Although vision based object tracking in real-time is common, doing pixel-wise segmentation is more time-consuming, thus real-time implementation is more difficult. In the paper [3] and [4], they respectively have processing rate of 1.2sec/frame and 3sec/frame. For the paper [9], it runs at 30fps but it only does tool tracking with a bounding box and not segmentation. So without the help of GPU, our implementation is actually pretty fast.

4.2 Good Dice Coefficient Performance

None of the papers listed above utilize Dice coefficient to analyze their segmentation performance, but in the standard of segmenting interested parts in CT scan images, a Dice

value of over 0.7 is considered pretty promising. Taking into account the fact that this program is running in real time and that no GPU is used, an average Dice coefficient of 0.75 is satisfactory.

4.3 Raven Surgical Tool

Unlike most relating research, including the three papers listed above, my work was done on the Raven Surgical Tool. Although this task is still not well-studied, more work are focusing on segmenting Da Vinci tools, so it would be great to develop a segmentation algorithm and software aiming particularly at Raven Surgical Tool.

4.4 Robust to partial occlusion

Since we are combining color filtering with robot kinematics, our software is able to handle difficult situations where the surgical tool tip is partially occluded by the tissue or even strained with blood. On the contrary, in paper [3], the way they determine the accuracy of segmentation is by seeing whether all surgical tool pixels are within a pair of dashed line they marked as tool shaft. In this case, it would not have detailed information as of whether within the marked tool region, if all the pixels are all truly tools. So our algorithm stands out in this sense.

Chapter 5

FUTURE WORK

5.1 Improve Stage 1: Image Segmentation

5.1.1 Alternative Color Filtering Scheme

Currently, the background tissue is artificial, because it is a board printed with surgical endoscopic images, and so the "difference to tissue color" feature may be slightly biased. But this can be improved in the future, using the statistical result of the whole color spectrum about probability of occurrence for tool and non-tool pixels. Actually the statistics were done and visualized as previously discussed in *Experiment and Result* section.

5.1.2 Auto-correct Robot Kinematics Offset

The image segmentation is done using color filtering along with prior information of surgical tool location based on robot kinematics. Yet, due to minor inaccuracy in camera extrinsic matrix and error for cable driven robot, the predicted surgical tool projection doesn't always align with the true configuration. Currently, user is allowed to apply an offset in 3D to compensate for this misalignment, but since the likelihood map based on color filters is also available, one can potentially automate the adjustment process and determine the 3D offset value using Kalman filtering or other tracking algorithms.

5.1.3 Dice Coefficient Analysis with Different Scenarios

In this work, an average Dice value of 0.75 for the image segmentation is achieved. Also, analysis on the relation between surgical joint pose and Dice values are presented. But we are also interested in studying the performance of image segmentation, without robot kinematics

or without color filtering. So more analysis on those will be performed in the future.

5.2 Improve Stage 2: Depth Lattice Rendering

5.2.1 Reduce Baseline for Stereo Setup

Here in this work, two usb cameras are mounted together with a baseline of 40 mm which forms the stereo setup. But, in actual minimally invasive surgeries, the stereo endoscopes normally have smaller baselines about 7-9 mm, so a reduction in camera baseline is necessary if one wants to improve the practicality.

5.2.2 Precision Requirement

Currently, the background tissue reconstruction accuracy is around 1 cm and the surgical tool 3D reconstruction is more precise since we have the help from robot kinematics. However, the error for tissue is not trivial in the case of force estimation because great difference in applied force is present with just a millimeter difference in tissue deformation. So it would be necessary to bring the precision of 3D reconstruction down to the millimeter scale.

5.3 Work on Stage 3 and Stage 4

The applied force versus tissue deformation mapping is time-invariant also the direction of force may not be vertical to the tissue surface at all times, so developing a robust but practical force rendering method is essential and will be a major part of my future work.

In fact, in force rendering, tissue characterization also plays an important role. Despite using a hybrid method combining the computer science and the mechanical approach. It would be interesting to explore other possible cues that help with on-line tissue identification, such as tissue color changes when in contact with the surgical instrument, the specular reflections or shape from shading. Hopefully, then one will be able to combine the advantages of both approaches and come up with a better force rendering result. Then eventually implement the whole system on a haptic device controlled surgical robot arm system.

BIBLIOGRAPHY

[1] Y. Abdel-Aziz and H. Karara. Direct linear transformation from comparator coordinates into object space coordinates in close-range photogrammetry. *In Proc. ASP/UI Symp. CloseRange Photogrammetry*, pages 1–18, 1971.

[2] Pilar Sobrevilla Angelica I. Aviles, Arturo Marban. A recurrent neural network approach for 3d vision-based force estimation. *Image Processing Theory, Tools and Applications (IPTA), 2014 4th International Conference*, 2014.

[3] Peter K. Allen Austin Reiter and Tao Zhao. Feature classification for tracking articulated surgical tools. *Springer-Verlag Berlin Heidelberg*, pages 592–600, 2012.

[4] Tao Zhao Austin Reiter, Peter K. Allen. Marker-less articulated surgical tool detection. 2012.

[5] Phillip Roan Daniel Glozman Blake Hannaford, Hawkeye King. Raven-ii: An open platform for surgical robotics research. *IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING*, 60(4):954–959, 2013.

[6] G. Bradski. Dr. dobb's journal of software tools. 2000.

[7] Lei Cheng. Computation and measurement of force and tissue damage for the grasper-tissue interface in robot-assisted minimal invasive surgery. *Doctoral thesis, University of Washington*.

[8] Guang Zhong Yang D. Stoyanov. Removing specular reflection components for robotic assisted laparoscopic surgery. *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, (8856983):III – 632–5, 2005.

[9] Maneesh Dewan William Lau Ming Li Henry Lin Panadda Marayong Nicholas Ramey Darius Burschka, Jason J. Corso. Navigating inner space: 3-d assistance for minimally invasive surgery. 2005.

[10] Danail Stoyanov Pierre Jannin David Bouget, Max Allan. Vision-based and marker-less surgical tool detection and tracking: a review of the literature. *1361-8415/ 2016 Elsevier B.V.*, pages 633–654, 2016.

[11] Blake Hannaford Department of Electrical Engineering The University of Washington Hawkeye King, Sina Nia Kosari. Kinematic analysis of the raven-iitm research surgical robot platform. UWEETR-2012-0006, Revised 2016.

[12] T. G. K. van de Sande and C. G. Snoek. Color descriptors for object category recognition. *In European Conference on Color in Graphics, Imaging and Vision*, 2:378381, 2008.

[13] Jonckheere EA Hayati S Kwoh YS, Hou J. A robot with improved absolute positioning accuracy for ct guided stereotactic brain surgery. *IEEE transactions on bio-medical engineering*, page 153160, 1988.

[14] J. Dankelman M. K. Chmarra, C. A. Grimbergen. Systems for tracking minimally invasive surgical instruments. *Minimally Invasice Therapy Allied Technology*, 16(6):328–340.

[15] James P. Sethna Mark K. Transtrum. Improvements to the levenberg-marquardt algorithm for nonlinear least-squares minimization. *Computational Physics,Data Analysis, Statistics and Probability (physics.data-an)*, 2012.

[16] Jonathan Kleefield Harsha Gopal Edward Reardon Bryan T. Ho Frederick A. Kuhn Marvin P. Fried, FACS. Image-guided endoscopic surgery: Results of accuracy and performance in a multicenter clinical study using an electromagnetic tracking system. *COSM Meeting in Orlando, Florida*, 107:594–601, 1997.

[17] Marsic I. Burd R.S. Parlak, S. Activity recognition for emergency care using rfid. *6th International ICST Conference on Body Area Networks*, (1-936968-29-0), 2011.

[18] Voros S. Hager G.D. Pezzementi, Z. Articulated object tracking by rendering concsistent appearance parts. *Robotics and Automation ICRA*, pages 3940–3947, 2009.

[19] Matas dela Fuente Klaus Radermacher Robert Elfring, MBA. Assessment of optical localizer accuracy for computer aided surgery systems. *Computer Aided Surgery*, 15, 2010.

[20] Richard J. Radkea Siqi Chena, Daniel Cremersb. Image segmentation with one shape prior a template-based formulation. *Image and Vision Computing*, 2012.

[21] Z.R.Zhoua X.Baoa., M.Lub. Experiment study on puncture force between mis suture needle and soft tissue. 2(2):49–58, 2016.

[22] Tams Haidegger rpd Takcs, Imre J. Rudas. Reaction force and surface deformation estimation based on heuristic tissue models. *Takcs, ., Rudas, I.J., Haidegger, T. Med Biol Eng Comput*, 2016.