8-2015

Optical Tracking System to Monitor Laparoscopic Training

William D. Rytlewski
Grand Valley State University

ScholarWorks Citation

Rytlewski, William D., "Optical Tracking System to Monitor Laparoscopic Training" (2015). *Masters Theses*. 777.
https://scholarworks.gvsu.edu/theses/777

Optical Tracking System to Monitor Laparoscopic Training

William David Rytlewski

A Thesis Submitted to the Graduate Faculty of

GRAND VALLEY STATE UNIVERSITY

In

Partial Fulfillment of the Requirements

For the Degree of

Masters of Science in Engineering with Biomedical Emphasis

Seymour & Esther Padnos School of Engineering and Computing

August 2015

ABSTRACT

Laparoscopic Surgery, also known as Minimally Invasive Surgery, is a surgical technique where surgeons perform surgery through small incisions in the patient's abdomen using a camera to monitor the movements of the instruments inside the patient. In order for the surgery to be performed, the surgeon must possess a unique set of skills obtained through training using a variety of techniques. Simulators are the preferred method of training for laparoscopic surgery since they provide medical residents with real world scenarios as well as a tremendous amount of feedback on what he/she did wrong or right. However, due to the high cost associated with laparoscopic simulators, laparoscopic box trainers are more commonly used, but fail to provide trainees with the necessary feedback to create an effective training experience. The Electronic Laparoscopic Trainer (ELT) is a low cost device that provides users with a virtual reality like experience using a laparoscopic box trainer, but fails to accurately track the motion of the laparoscopic instruments. This paper describes and validates an optical tracking system to monitor the laparoscopic instruments inside of the laparoscopic box trainer that can be added to the ELT to increase its effectiveness during training. The algorithm performs a series of steps that are taken a frame at a time to obtain the 3D real world tracking point of the laparoscopic instrument, which are used to calculate quantitative values for various aspects of the user's performance that represent how effective, controlled, and safe the user's movements were. Testing confirmed that the algorithm can accurately track the distance traveled and direction of up to two laparoscopic instruments in 3D real space and is capable of differentiating between users of varying skill levels by using performance metrics such as the amount of time each instrument is in the field of view and path length.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

I. INTRODUCTION

Laparoscopic Surgery, also known as Minimally Invasive Surgery, is a surgical technique where surgeons perform invasive procedures through small incisions in the patient's abdomen. Surgeons must view their movements with the instruments on a two-dimensional screen using a camera inside the patient. In order for the surgery to be performed safely and effectively, the surgeon must possess a unique set of skills acquired through a specialized system of training developed for laparoscopic surgery. This training is typically comprised of laparoscopic box trainers and/or virtual reality trainers, animal experiments, and operating room experience (1). However, when a surgeon is able to master the skills required to successfully perform laparoscopic surgery through training, his/her patients will experience less pain and a faster recovery time due to the smaller incisions when compared to conventional open surgery.

A. Laparoscopic Training

Surgical residents typically train for laparoscopic surgery in a 4 to 5 year program with basic surgical techniques being taught in the first year, laparoscopic box training and/or virtual reality training in the second year, animal model training in the third year, and operating room experience being obtain during the remaining years (2). Animal labs are considered to be the most effective training method before a medical resident enters the operating room, but are also the most expensive. Since animal labs are expensive to run and maintain, it is important that surgical residents obtain all the cognitive and psychomotor skills necessary to perform laparoscopic surgery before moving onto animal labs. Laparoscopic virtual reality and box training methods allow residents to practice and hone their skills before attending animal labs

However, the amount of time a surgical resident is able to train using a laparoscopic box trainer and/or virtual reality trainer is limited since the training institution owns and controls access to the equipment. Surgical residents are excused from clinical duties to attend training sessions, but have little to no extra time to visit the training facility to practice. Therefore it is vital that a resident gains as much as possible each time he/she trains using a box trainer or virtual reality trainer. Controlled trials have been performed to validate the effectiveness of various laparoscopic box trainers as well as virtual reality trainers to determine the most efficient training method. Box trainers are currently the standard for laparoscopic skills training and assessment, but there has been a growing call for virtual reality trainers to play a greater role because of their perceived increased realism and ability to collect data (3). Both the similarities and differences between laparoscopic virtual reality trainers and box trainers have been well characterized by the variety of studies performed on the topic (4,5).

B. Laparoscopic Box (Lap Box) Trainers

Laparoscopic box trainers come in various forms as to provide surgical residents with different training experiences. They can be organ specific or more generic, accommodate real-tissue specimens, and may even be used by multiple people at a time (6). No matter what the form, box trainers are considered the least effective for of training, but are also the lowest cost. Box trainers are so simple that they can be made at home using a plastic storage box and HD webcam, which explains why the are low cost (7). The typical box trainer includes an enclosed cavity where the training tasks are performed, openings in the shell through which the instruments may be inserted, and a built-in camera that displays the work field in the enclosed cavity.

Normally surgeons performing laparoscopic surgery work in teams with one moving the camera to provide a more dynamic field of view. So a fixed camera with a chosen focal lens is not realistic for surgery, but allows a surgical resident to train without assistance from another person (6). This way, residents only have to worry about finding time to train in their own schedule rather than also having to coordinate with another resident's schedule. The only advantage to box trainers that require two people is that it allows the residents to practice moving the camera and working as a team, which is a more realistic setting than a single person box trainer. For this project, the single person Fundamental of Laparoscopic Surgery (FLS) box trainer, shown in Figure 1 below, was used since it is the most common one used today.

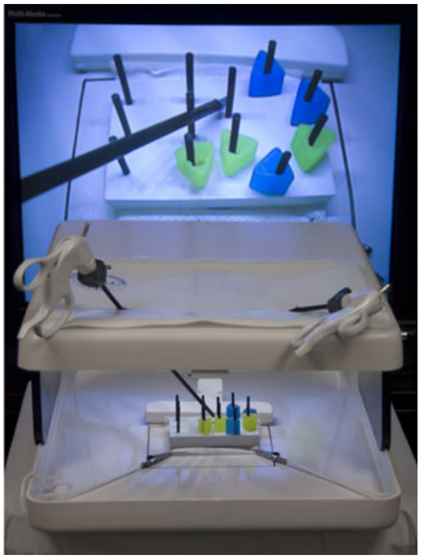


Figure 1: Ready to Use FLS Box Trainer (6)

FLS is a committee formed in the late 1990s by the Society of American Gastrointestinal Surgery (SAGES) that developed an educational program in 2004 to "educate surgeons in the underlying principles and basic skills of laparoscopic surgery" as well as to "document competency in surgical practice" (8). Since its introduction, the FLS program has been validated from both a metrics standpoint and by beta field-testing with general surgeons (8-10). The FLS program is now endorsed by the American College of Surgeons (ACS) and as of 2010 is required by the American Board of Surgery (ABS) for a general surgery resident to qualify for the American Board of Surgery certifying examination (11). The FLS program includes a hands-on skills training component that requires surgical residents to perform 5 different tasks inside the FLS box trainer.

The FLS program uses all 5 training exercises for both training and assessment. Originally, 7 tasks were developed for the hands-on portion of the FLS program, but were reduced to 5 since studies showed that "2 of the exercises failed to contribute any additional discriminatory value to the training or assessment" (8). The 5 tasks include peg transfer, precision cutting, suturing with an intracorporeal knot, suturing with an extracorporeal knot, and ligating loop. Figure 2 shows each task as being performed inside the FLS box trainer.

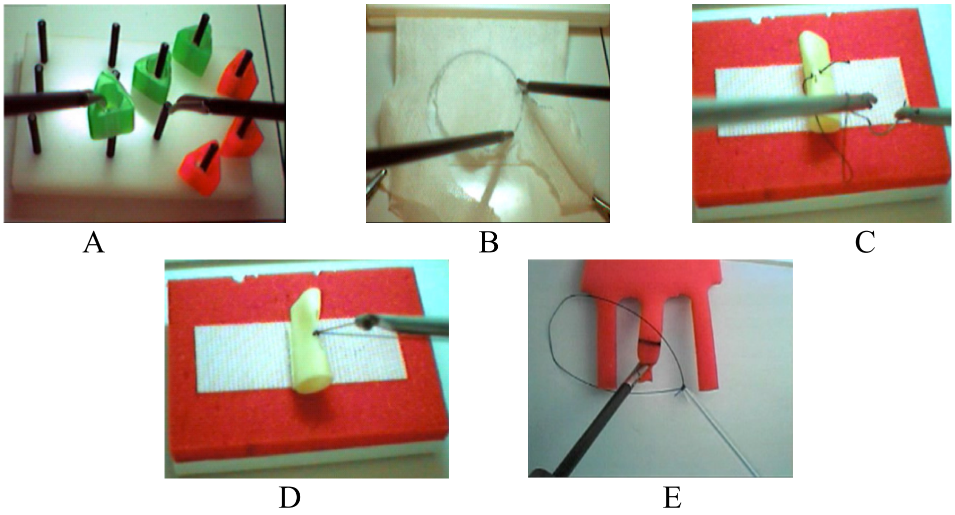


Figure 2: (A) Peg transfer, (B) precision cutting, (C) intracorporeal suture, (D) extracorporeal suture, and (E) ligating loop FLS tasks as they are performed inside FLS box trainer

All tasks are scored using a combination of time and accuracy measures. High scores will result from tasks performed efficiently and without error. Penalties are assessed for errors and a lack of precision, which are specific to each task. The user must also complete each task under the maximum time limit set for each specific task in order to receive a score. However, completing the task under the maximum time limit does not mean the user will receive a passing score.

When a resident uses a laparoscopic box trainer he/she only knows how long it took him/her to complete the FLS program task and whether or not they made any obvious errors. Not only does this encourage inefficient movements and less practice time, but also fails to provide residents with significant information about how effective, controlled, and safe their movements were. An expert surgeon is able to judge the accuracy and efficiency of the resident's performance using the laparoscopic box trainer, which is how certification tests involving box trainers are quantitatively scored (2). However, expert surgeons are not always on hand to provide effective and immediate feedback. Laparoscopic virtual reality trainers can provide feedback to surgical residents they would otherwise only obtain from an expert surgeon.

C. Laparoscopic Virtual Reality Trainers

Laparoscopic virtual reality trainers, such as the one shown in Figure 3, provide surgical residents with an improved training experience compared to box trainers by increasing realism and assessing a greater number of performance metrics. Such additional performance metrics include path length, economy of movement, average speed, average acceleration, smoothness of motion and a total score based on all others, which is all provided quantitatively to the resident in real time. This creates a more effective laparoscopic training experience since it allows residents to gain more from each individual training session. Studies have shown that virtual reality training not only improves effectiveness, but also is able to shorten the learning curve for surgical residents to transition earlier from simulation to surgery (12,13).

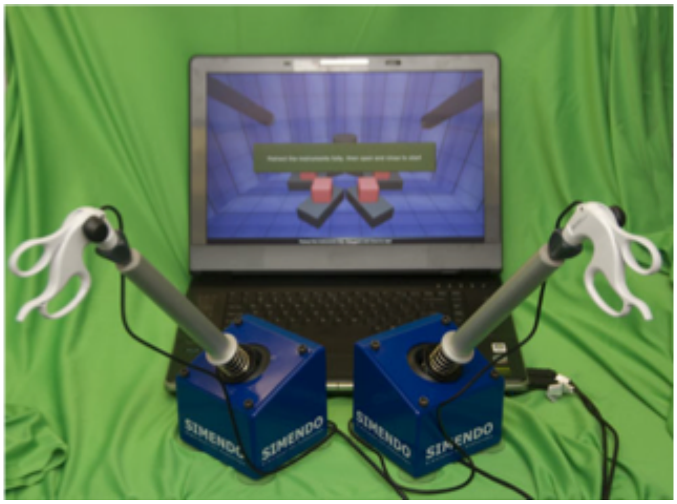


Figure 3: Simendo System (4)

One of the many aims of virtual reality based laparoscopic training is to reduce cost by reducing the training time of surgical residents while keeping surgical errors to a minimum (12). However, the initial high cost associated with laparoscopic virtual reality trainers limits the likelihood that they will be able to reach this goal. Training facilities will continue to use the less costly and less effective laparoscopic box trainers as a result. Hence, it is desirable to incorporate a virtual reality system into an existing laparoscopic box trainer that can provide better feedback at a lower cost.

D. Electronic Laparoscopic Trainer (ELT)

The Electronic Laparoscopic Trainer (ELT) is a low cost device that provides users with a virtual reality like experience using a laparoscopic box trainer (14). The ELT, shown in Figure 4, is a device that fits in a laparoscopic box trainer and has 24 independently illuminating touch sensitive tiles that the surgical resident interacts with to complete tasks. Tasks performed using the ELT including Random Squares, Press and Hold, Two Hands, Circle, and Force Test require the user to tap or hold tiles once they are illuminated using the instrument in a specific hand. Currently, the ELT uses an accelerometer attached to one of the two instruments to determine which hand is being used to tap or hold the illuminated tile. As the accelerometer is only able to detect movement, it only works to detect which hand is being used when the instrument with the accelerometer is completely motionless when not being used. Testing demonstrated that the ELT is capable of increasing the rate of laparoscopic skill development as is, but a better instrument tracking system will increase the effectiveness of training with the ELT by providing the user with even more feedback.

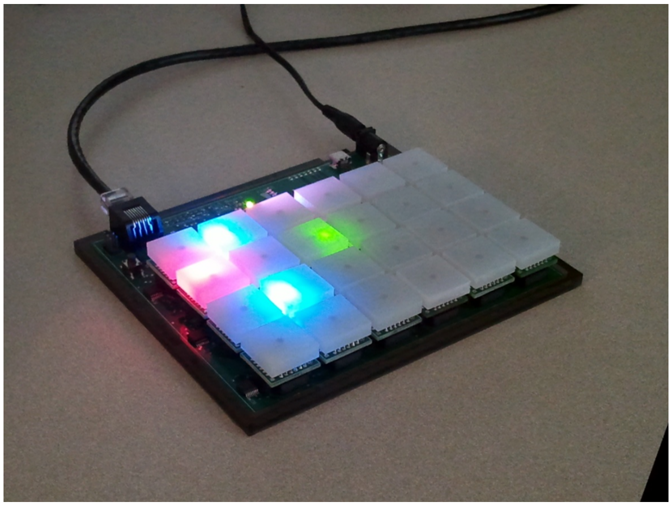


Figure 4: Electronic Laparoscopic Trainer (ELT)

E. Electronic Tracking of Laparoscopic Instruments

Automated electronic tracking of laparoscopic instruments allows a system to determine key values describing the quality of the user's performance as they perform laparoscopic box trainer tasks, such as the ones for the FLS program. A variety of methods exist to determine the location of laparoscopic instruments inside a box trainer with some being better than others. Previous studies have successfully proved that magnets may be used to perform motion tracking of laparoscopic instruments (15,16). The electromagnetic technique uses a series of magnets placed in specific locations around the laparoscopic box trainer to produce a magnetic field, which can then be used to determine the instrument's location in 3D space. Another method uses an accelerometer and gyroscope combination, which was only able to roughly determine the instrument location (17). Both methods require additional hardware to be added to the laparoscopic box trainer or instruments, which is not ideal.

The method determined to work the best for tracking the motion of laparoscopic instruments was optical, which uses the box trainer's built-in camera. Various articles demonstrate that using cameras to optically track laparoscopic instruments during training can be successfully implemented (18,19). The Endoscopic Video Analysis (EVA) tracking system is an existing laparoscopic instrument tracking system that includes a series of 5 steps to obtain an accurate 3D location of each laparoscopic instrument in a given frame (19). The 5 steps include camera distortion correction, image processing, determination of a 2D tracking point, determination of tracking window, and determination of point in real space. It is this EVA system that was used as a model for the development of an instrument tracking algorithm to be implemented with the ELT to increase its effectiveness during training.

II. SYSTEM DESIGN

The algorithm for instrument tracking was developed using MATLAB to analyze video from the existing camera built into the laparoscopic box trainer. The algorithm is based on the EVA tracking system, but does vary in some ways. Just like the EVA tracking system, it follows a series of steps to obtain the 3D real world locations, relative to the built in camera, of the left and right instruments (19). The steps for this algorithm are each taken a frame at a time and include contrast stretching, color-based segmentation, Erosion, Instrument Isolation, Dilation, Canny edge detection, Hough transform, identifying 3D tracking point, and finally, identifying 3D real world tracking point. By obtaining the 3D real world points through instrument tracking, the algorithm is able to determine important performance metrics such as the number of times each instrument left the field of view, path length, and average speed that represent how effective, controlled, and safe the user's movements were.

A. Obtaining Video and Determining Length, Frame Rate, and Size

The first thing the algorithm does is obtain the recorded video of a medical resident using the lap box trainer using the VideoReader function in MATLAB. The algorithm uses other MATLAB functions to determine the length, frame rate, and size of the video, which are used throughout the algorithm. It then goes through the video an image at a time, such as the one shown in Figure 5, performing the steps specified above and described below. Example figures are provided with each step description and root off of the original frame image shown in Figure 5.

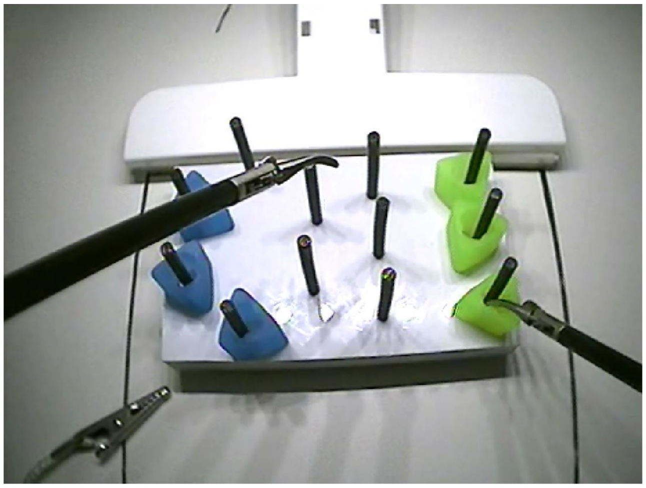


Figure 5: Original Image

B. Contrast Stretching

The first image-processing technique performed as a step toward obtaining 3D points in real space of two laparoscopic instruments is contrast stretching. Contrast stretching will expand the range of intensity levels in an image so that it spans the full intensity range of the recording medium or display device. A piecewise-linear transformation function is used to perform contrast stretching of the original image, which results with the image as seen in Figure 6, below. The resulting image, shown in Figure 6, does not vary much from the original as the original image, shown in Figure 5, already has a large contrast. Contrast stretching would have a greater affect on an image with a low contrast.

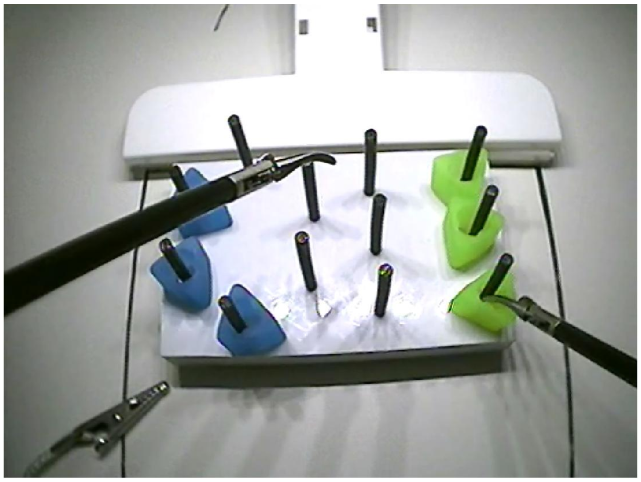


Figure 6: Contrast stretched image

C. Color-Based Segmentation

Once the original frame image has been contrast stretched, the algorithm goes through each pixel in the image to determine if it is black, which by definition is considered thresholding. A new image is created, which illustrates all identified black pixels from the contrast stretched image as white and all other pixels as black. An example of this new black and white image can be seen below, Figure 7. This removes any object that is not black, but as Figure 7 shows, there are a lot of black objects in the image that need to be removed in order to isolate the instruments.

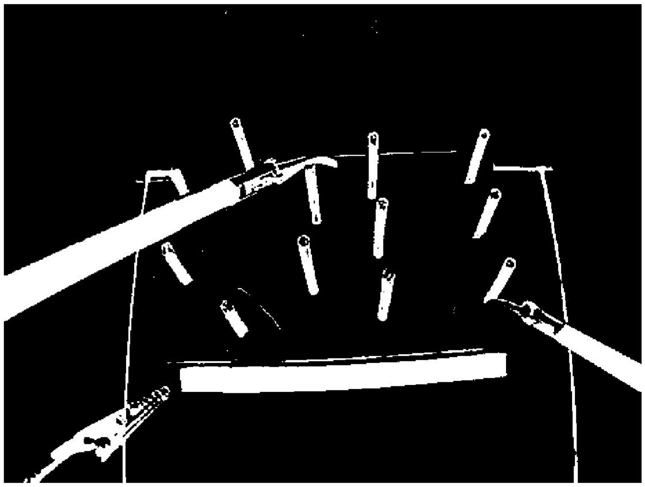


Figure 7: Color-based segmented image

D. Erosion

Erosion is an image processing technique that can be used to remove unwanted smaller objects from an image. In this case, it is used to remove any of the smaller black objects in the image, shown as white in the color based segmented image, Figure 7. As seen in Figure 8, most of the objects from Figure 7 have been successfully removed using the MATLAB function imerode. However, both the left and right instruments have shrunk or become smaller as a result, which can be fixed through dilation. But first, steps are taken to isolate each instrument.

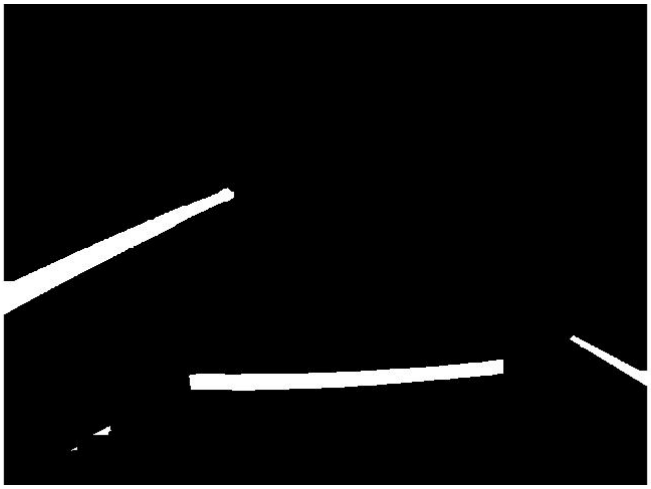


Figure 8: Eroded image

E. Isolating Left and Right Instruments

The eroded image, shown in Figure 8, is used to isolate the left and right instruments through a series of steps. First, the algorithm identifies all the white pixels along the edges of the eroded image. All white pixels located along the left and left half of the bottom edges are placed into one new image to isolate the left instrument. While the white pixels located along the right and right half of the bottom edges are placed into another to isolate the right instrument. Next, the algorithm determines which of the pixels that border the ones found in the previous step are white. It then repeats this step until all the white pixels connected to the original ones on the edge are found. All the identified pixels are turned white in either the left or right instrument image. The result is two images, each containing a white object representing the left or right shrunken laparoscopic instrument. These two images can be seen in Figure 9.

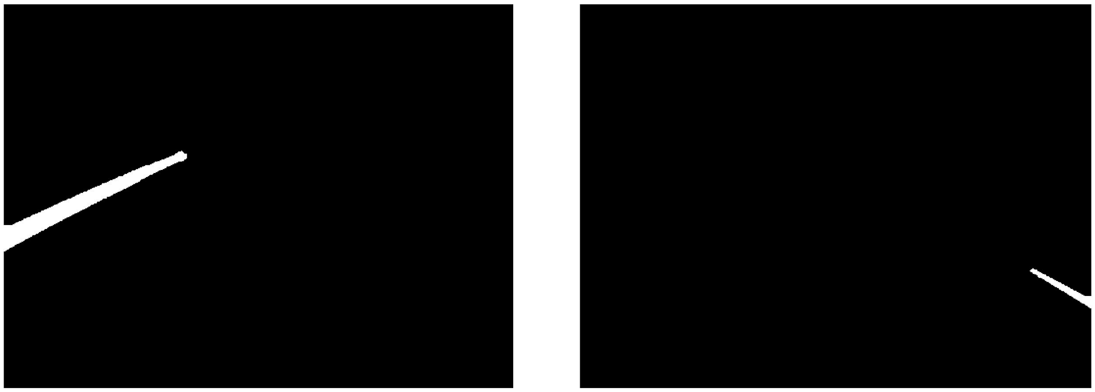


Figure 9: Isolation of left and right instruments

F. Dilation

In order to restore the instruments back to their original size, the images have to be dilated by the same structuring element in which they were eroded. As seen in Figure 10, each instrument has been restored to its original size through dilation, which is done by using the MATLAB function imdilate. The algorithm has isolated and identified each instrument at its original size, now it needs to create an outline of each.

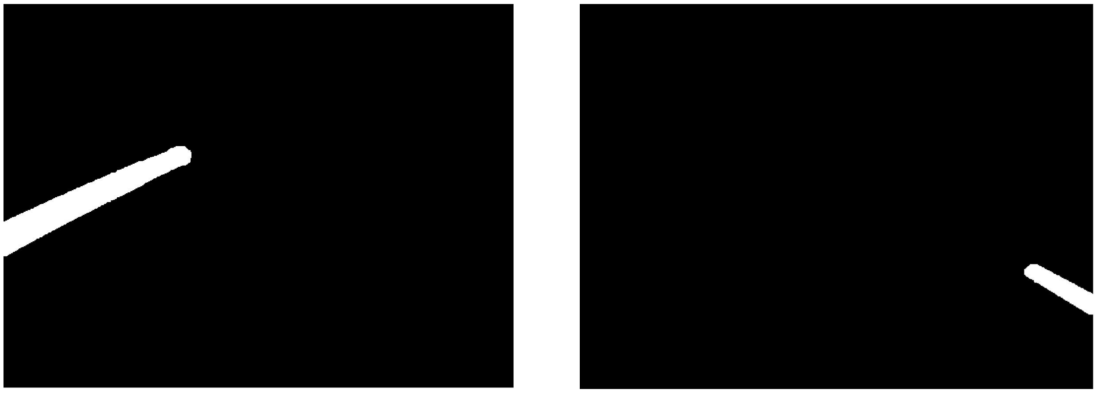


Figure 10: Dilated images

G. Canny Edge Detection

The Canny edge-detecting image processing technique uses a multi-stage algorithm to detect a wide range of edges in images. The developed algorithm uses the Canny edge detection function in MATLAB to create an outline of the laparoscopic instruments as shown in Figure 11, below. By isolating the instruments first, only the edges of the instruments are obtained instead of all edges in the entire original frame image. The next step is to characterize the 2 sides of each laparoscopic instrument using this new resulting image from canny edge detection.

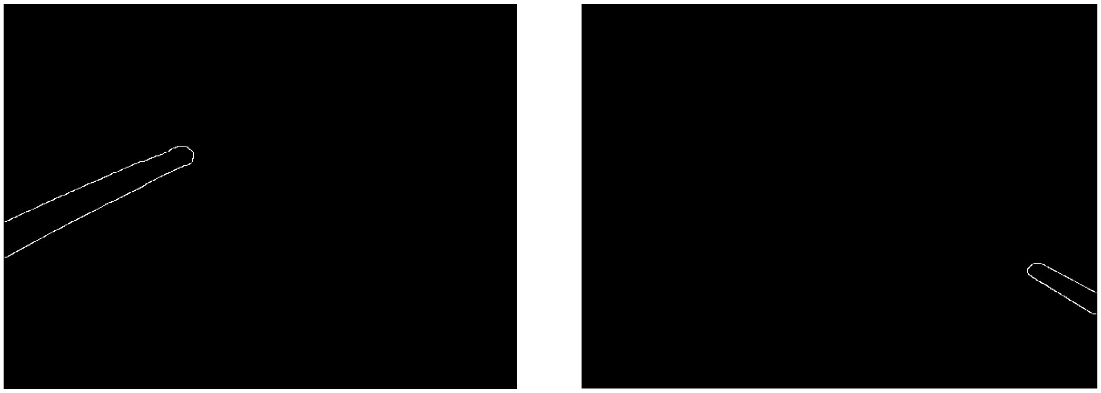


Figure 11: Canny edge detected images

H. Hough Transform

In order to characterize the two edges of each laparoscopic instrument, the Hough transform is used. The Hough transform is used in image processing to identify lines in an image as well as arbitrary shapes such as circles or ellipses. In this case the Hough transform is used to detect only lines. An example Hough transform of the canny edge detected image using the MATLAB function hough can be seen in Figure 12. The image is mostly black, but does have white spots illustrating lines in the image. Once the Hough transform is obtained, the eight longest lines can be determined by finding the points of highest intensity, which are marked in Figure 12 as small squares. The algorithm uses the MATLAB function houghpeaks to find the points of highest intensity and houghlines to obtain the end points of each line.

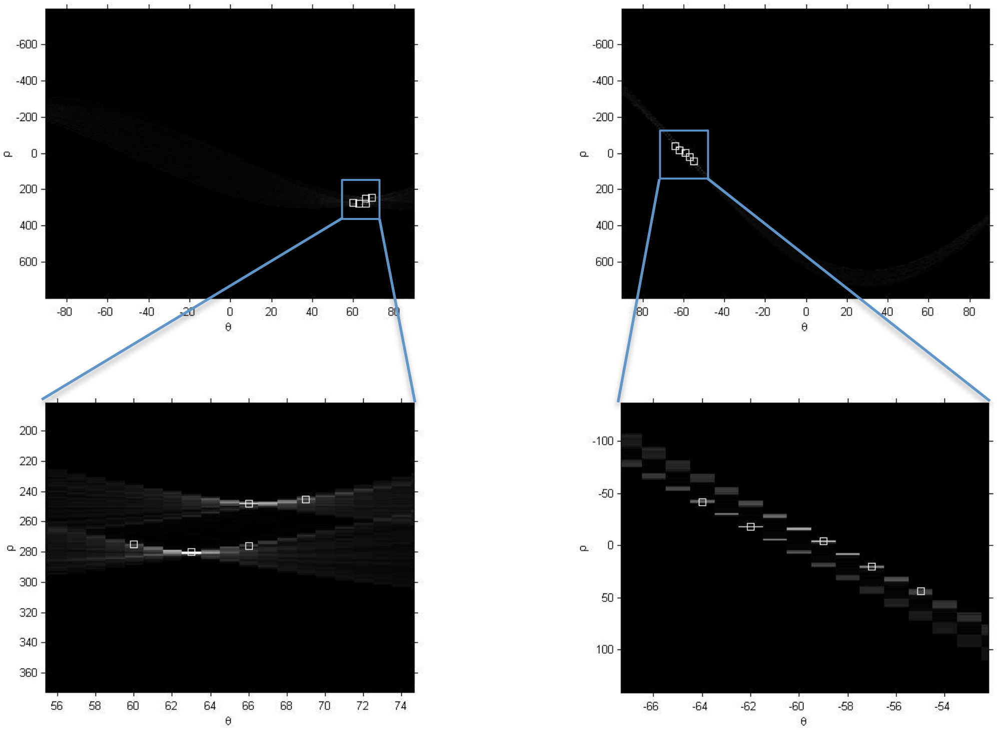


Figure 12: Hough Transforms

The eight longest lines are obtained instead of two because it is likely that one edge of the instrument may produce the two longest lines. By obtaining eight lines and performing a set of checks, the algorithm has a greater chance of properly identifying each edge of the two laparoscopic instruments. An example of the lines obtained from the Hough transform can be seen on the contrast stretched image in Figure 13. From the indicated lines, a total of two will be identified from each image by relative location analysis to characterize the edges of the two instruments and used to identify 3D tracking points of the instruments.

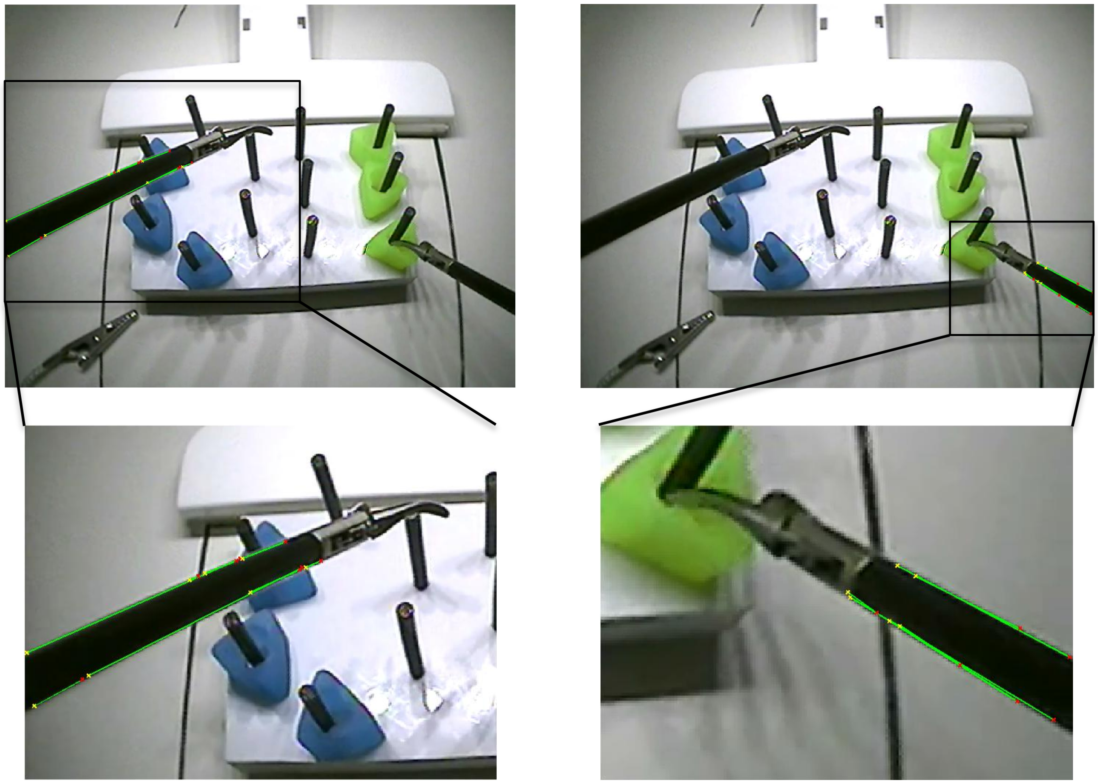


Figure 13: Hough lines on contrast stretched image

I. Identify 3D Tracking Point

Once the edges of each instrument are characterized by line equations, a midline can be created between the two edges of an individual instrument. This midline is used along with the dilated image to determine the location of a point along that line where it goes from white to black. This point is considered the 2D tracking point of the instrument and is combined with the determined diameter of the instrument at that point, distance between lines characterizing the edge of the instrument at the 2D tracking point, to make up the 3D tracking point. Figure 14, below, illustrates the instrument edges, midline, and 2D tracking point of each instrument on the contrast stretched image.

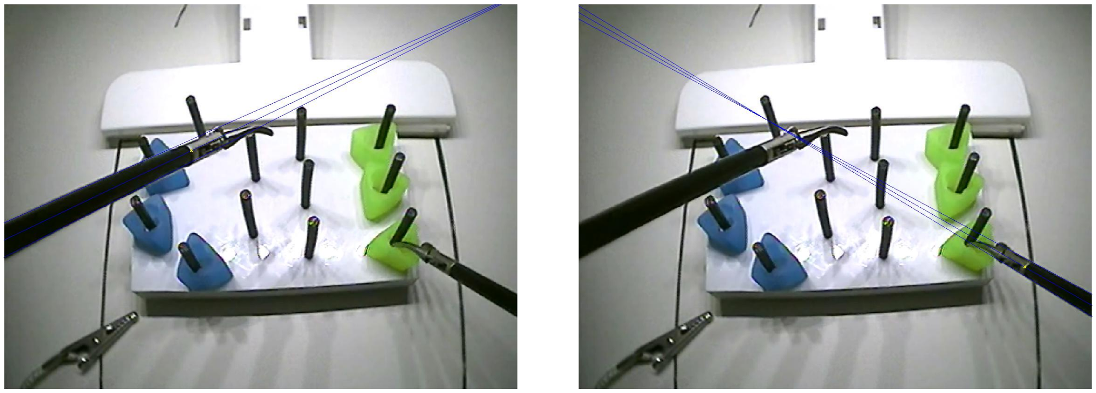


Figure 14: Image showing 2D tracking point of the left and right laparoscopic instruments

J. Identify 3D Real World Tracking Point

The last and final step involved in determining the location of each laparoscopic instrument is calculation of the 3D real world tracking point, which is done using the 3 developed Equations below. Equation 1 is used to determine the real world x-axis point, while Equation 2 and 3 are used to determine that of the x and z-axis. Equation 3 is a line equation derived using the instrument's diameter in pixels when it was at two different and known distances away from the camera, which were determined through testing.

$$REAL_x = \frac{Diameter_{mm}}{Diameter_{pixel}} \cdot \left(Pixel_x - \frac{VideoWidth}{2}\right) \tag{1}$$

Where:

$Diameter_m$ is actual diameter of the instrument in millimeters (5 mm)
$Diameter_{pixel}$ is diameter of the instrument in pixels
$Pixel_x$ is x-axis pixel location
$VideoWidth$ is width of the video frame in pixels

$$REAL_y = \frac{Diameter_{mm}}{Diameter_{pixel}} \cdot \left(Pixel_y - \frac{VideoHeight}{2}\right) \tag{2}$$

Where:

$Diameter_m$ is actual diameter of the instrument in millimeters (5 mm)
$Diameter_{pixel}$ is diameter of the instrument in pixels
$Pixel_y$ is y-axis pixel location
$VideoHeight$ is height of the video frame in pixels

$$REAL_z = \left((-0.16262) \cdot Pixel_z\right) + 11.773 \tag{3}$$

Where:

$Pixel_z$ is z-axis pixel location

K. Performance Metrics

The last thing the algorithm does is utilize the obtained instrument tracking points to calculate quantitative values for various aspects of the users performance using the lap box trainer, as are defined in Table 1. All the resulting performance metrics, shown in Table 1, represent how effective, controlled, and safe the user's movements were with the laparoscopic instruments.

Table 1: Quantitative performance values that will be obtained from developed tracking system

Performance Metric	**Units**	**Definition**
Time In	s	Total amount of time instrument is in field of view
Time Out	s	Total amount of time instrument is out of field of view
Times Out of View	-	Total number of times the instrument exits the field of view
Path Length 2D	pixels	Total distance of instruments path in 2D
Path Length 3D	m	Total distance of instruments path in 3D
Average Speed 2D	pixels/s	Average speed of instrument in 2D
Average Speed 3D	m/s	Average speed of instrument in 3D

III. STUDY DESIGN

A. Step 1

To test how well the algorithm works to optically track laparoscopic instruments, an instrument was moved a specific distance along each axis. The actual distance moved by the instrument was compared to the distance determined by the algorithm. First, the instrument was moved approximately 4 inches right along the x-axis. The next time the instrument was moved 2 inches up along the y-axis. Finally, the instrument was moved 2 inches away from the camera along the z-axis. Once this was completed and the algorithm was verified to successfully track a single instrument in the laparoscopic box trainer, it had to be tested using two instruments. So two instruments were used inside of the lap box trainer to confirm that the algorithm can track both of them. However, further validation was required to show the system is capable of differentiating between users of various skill levels.

B. Step 2

In order to verify that the laparoscopic instrument tracking system may differentiate between users of varying skill levels, videos provided by Grand Rapids Medical Education Partners (GRMEP) of 15 first and 12 second year surgical residents performing the peg transfer task from the FLS program were analyzed using the developed algorithm. The first year residents, a novice group, had no prior hands-on training experience with laparoscopic instruments. In contrast the second year residents, the experienced group, had completed their laparoscopic training at GRMEP and were ready for the FLS examination. The recorded videos were obtained from the built in camera inside the FLS box trainer as the resident performed the task. Before starting, each participant was shown how to successfully complete the peg transfer exercise by his or her instructor.

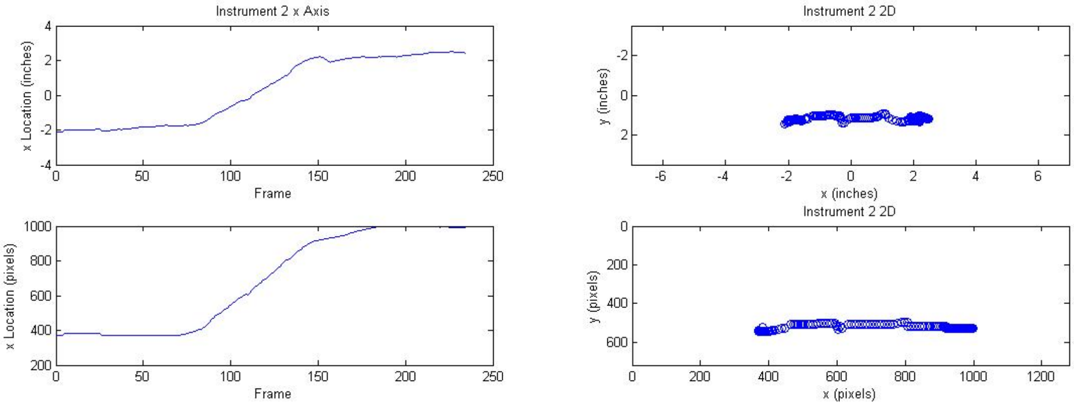
According to the FLS program, the peg transfer exercise starts with six colored objects aligned on the six pegs on the side of the board corresponding to the user's non-dominant hand. To complete this task, the user must lift one of the six objects at a time, first with their non-dominant hand, and transfer it in midair to their dominant hand, which is used to place the piece on one of the six pegs on the opposite side of peg board. Once all six pieces are transferred to the dominant hand side, the user must reverse the process to move all the objects back to the non-dominant side. The user has 300 seconds to complete the task with the timer starting when the first object is touched and ends upon the release of the last object. A penalty is assessed if an object is dropped outside the field of view or if the user can no longer retrieve the object.

Each video was analyzed using the developed algorithm to obtain both the tracking points as well as the performance metrics included in Table 1. Performance metrics between the novice and experienced groups were compared using the Student's t-test, which examines whether the mean of two samples is different. If the t-test determined a performance metric to be significantly different across the two groups, the algorithm may successfully differentiate between users of various skill levels using that performance metric.

IV. RESULTS

A. Step 1

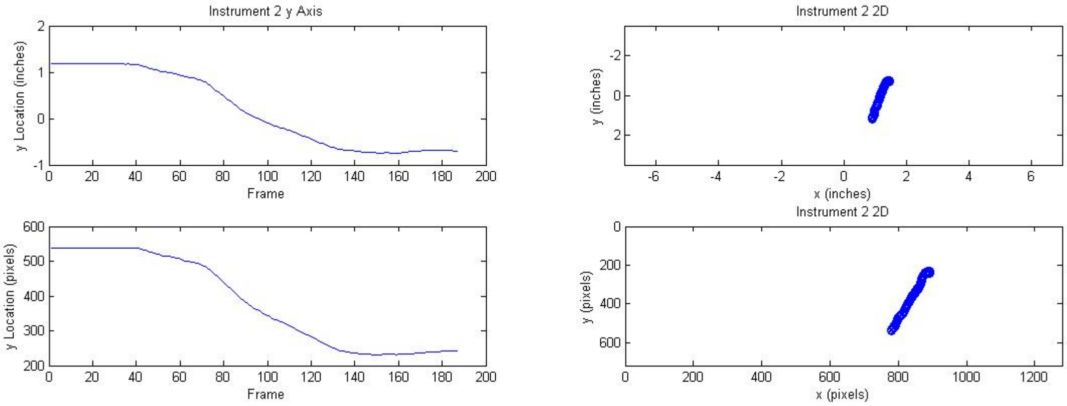
The first step in the validation process for the developed algorithm was to confirm it could accurately track 2 laparoscopic instruments inside of the laparoscopic box trainer. Figure 15 graphically shows the algorithm results when the instrument was moved approximately 4 inches to the right along the x-axis. The plots indicate that the instrument moved approximately 4 inches to the right along the x-axis.



(A) (B)

Figure 15: Plots of (A) movement along the x-axis only and (B) 2D movement when instrument was moved approximately 4 inches to the right

The algorithm results when the instrument was moved approximately 2 inches up along the y-axis resulting can be seen in the plots shown in Figure 16. As one can see, the plots produced by the algorithm show a 2 inch movement upward of the instrument along the y-axis.



(A) (B)

Figure 16: Plots of (A) movement along the y-axis only and (B) 2D movement when instrument was moved approximately 2 inches up

Figure 17, below, shows the plots produced when the instrument was moved 2 inches away from the camera along the z-axis. The figure indicates that the instrument moved 2 inches along the z-axis.

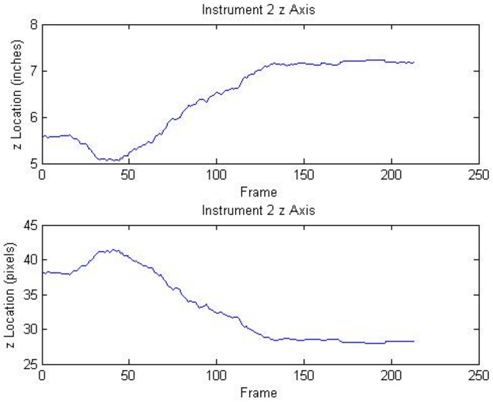
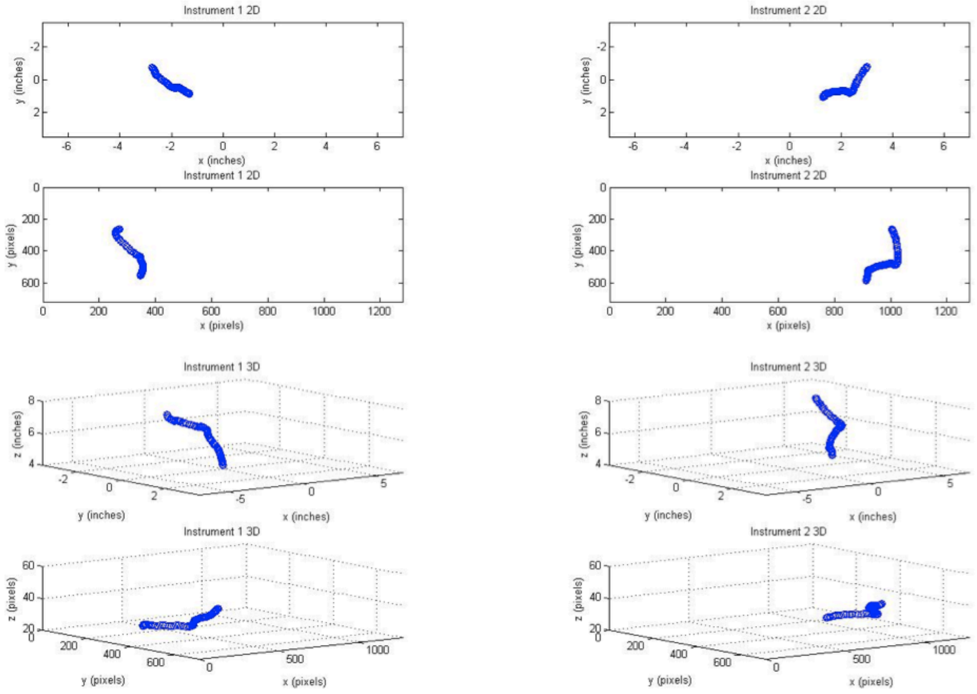


Figure 17: Plot of movement along the z-axis only when the instrument was moved approximately 2 inches further away from the camera

Figure 18 illustrates 2D and 3D plots produced by the algorithm when two instruments were used in the laparoscopic box trainer.



(A) (B)

Figure 18: 2D and 3D plots of instrument (A) 1 and (B) 2 while used simultaneously inside the lap box trainer

B. Step 2

The performance metrics obtained from the second part of the study were broken down within each experimental group by handedness, dominant hand and non-dominant/other hand. Mean and standard deviation values were calculated for each performance metric within each experimental group obtained from tracking the instrument in the resident's dominant hand, non-dominant hand, and an average of the two. The t-test was used to compare the same grouping of values for each performance metric across each experimental group.

Time In and Time Out

The first quantitative performance values obtained from the algorithm are the amount of time in which each instrument was in and out of the field of view. Table 3, below, shows the results from the mean, standard deviation (SD), and t-test calculations.

Table 2: Time In and Out Results

		Time In (s)			Time Out (s)		
		Dominant Hand	Other Hand	Average	Dominant Hand	Other Hand	Average
Novice Group	Mean	146.62	149.34	147.98	9.59	8.55	9.07
	SD	63.02	74.81	68.7	10.52	6.28	5.96
Experienced Group	Mean	63.02	74.81	68.7	7.53	3.41	5.47
	SD	25.31	28.28	26.13	8.67	4.29	4.33
t-Test Results (p)		0.0007	0.003	0.0015	0.291	0.009	0.041

The results of the t-test, shown in Table 2, indicate that the amount of time in which each instrument was in the field of view across the Novice and Experienced groups were significantly different ($p = 0.0007$, $p = 0.003$, $p = 0.0015$). In general, the Experienced group had the instruments in the field of view for less time than the Novice group did, which is illustrated in the Box Plot shown in Figure 19. The amount of time in which each instrument was out of the field of view was found to be significantly different across the two groups for the instrument in the non-dominant hand ($p = 0.009$) and the average of the two instruments ($p = 0.041$) according to the t-test results. It was not significantly different for the instrument in the dominant hand ($p = 0.291$). The box plot shown in Figure 20 represents the data obtained from the algorithm for the amount of time in which each instrument was out of the field of view.

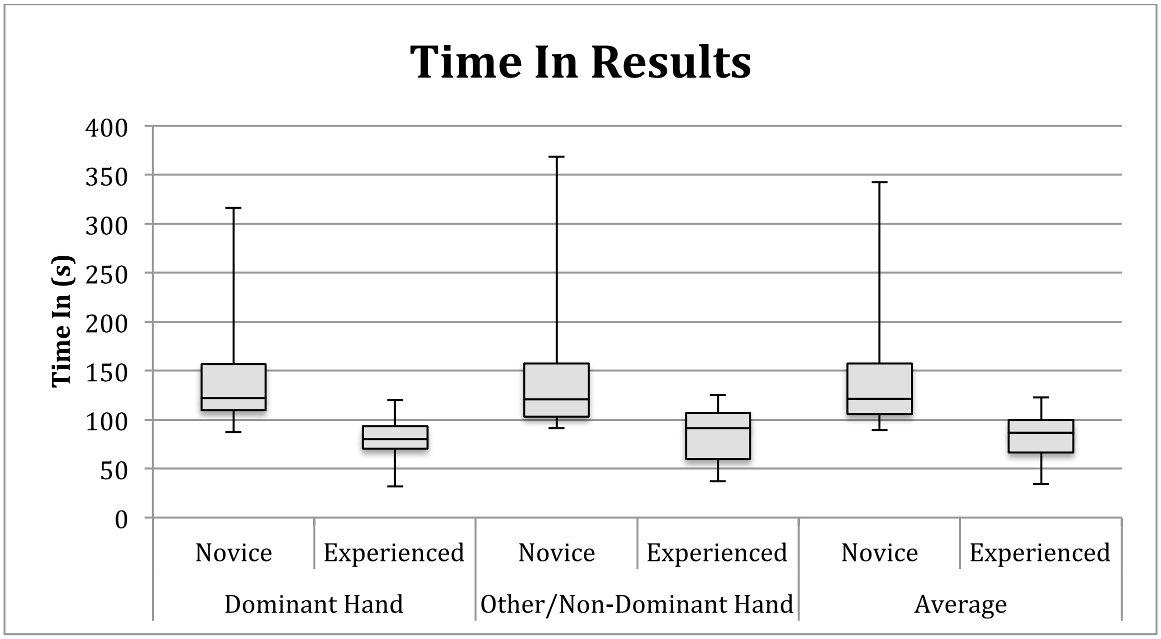


Figure 19: Box Plot of Time In Results

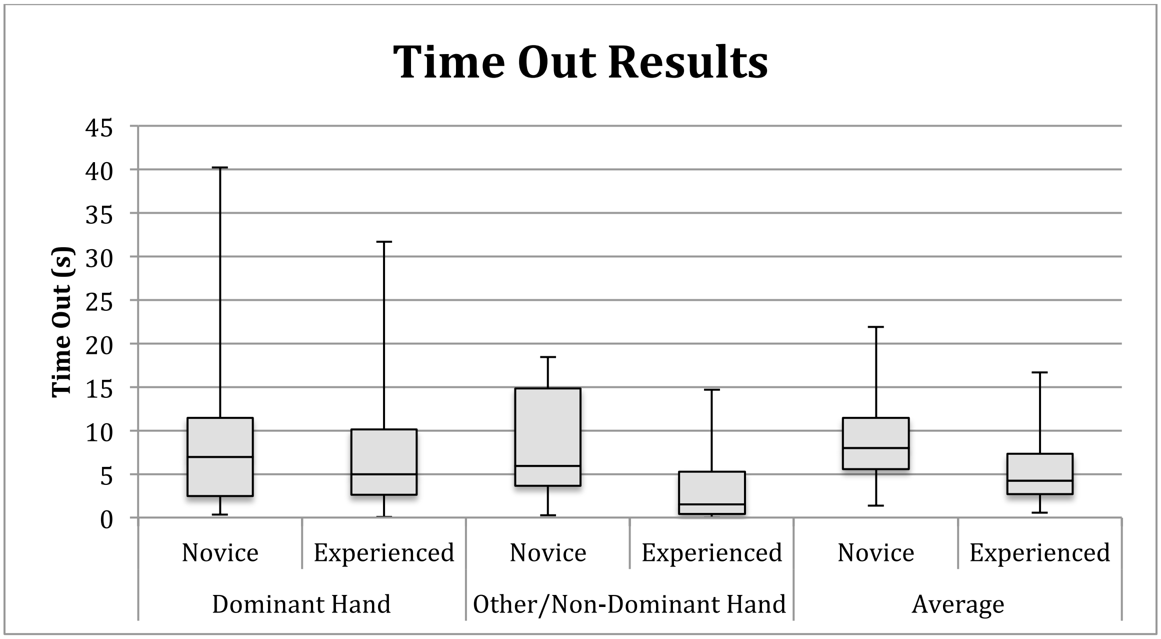


Figure 20: Box Plot of Time Out Results

Times Out of View

Another performance metric obtained from the algorithm is the number of times each instrument leaves the field of view. Table 4, below, shows the results from the mean, standard deviation (SD), and t-test calculations.

Table 3: Times Out of View Results

		Times Out of View		
		Dominant Hand	Other Hand	Average
Novice Group	Mean	15.73	24.67	20.2
	SD	12.44	11.8	10.34
Experienced Group	Mean	17.92	14.75	16.33
	SD	18.52	12.61	12.28
t-Test Results (p)		0.365	0.024	0.197

The results of the t-test, shown in Table 3, indicate that the number of times the instrument in the non-dominant hand left the field of view was significantly different between the experimental groups ($p = 0.024$). However, it was not significantly different for the instrument in the dominant hand ($p = 0.365$) or the average of the two instruments (0.197). The box plot shown in Figure 21 represents the data obtained from the algorithm for the number of times each instrument left the field of view.

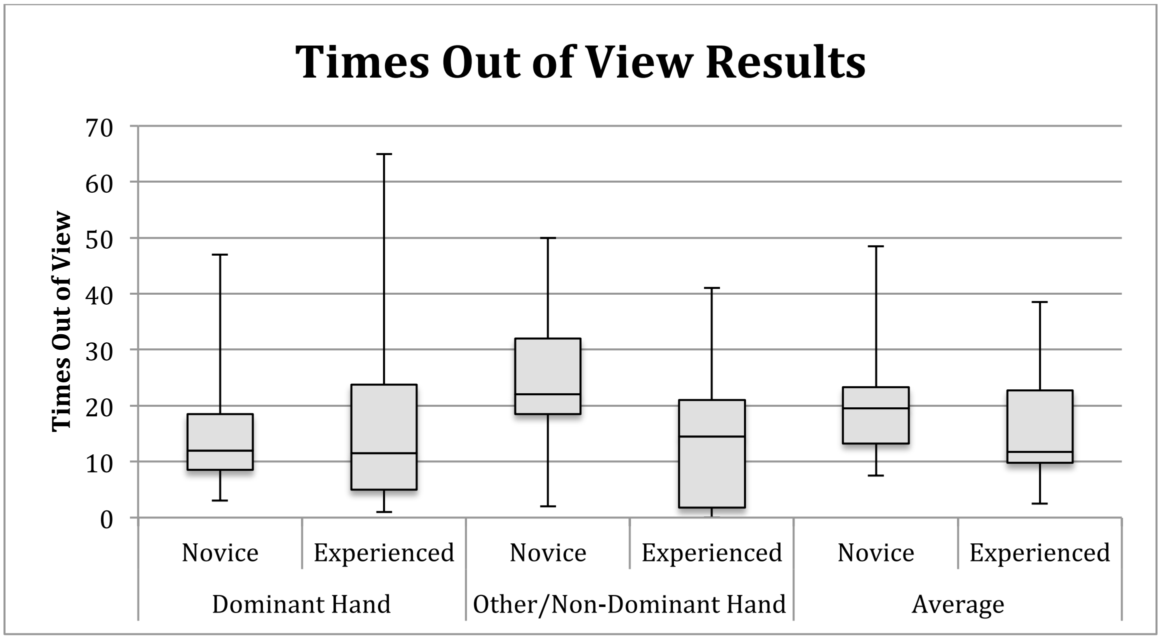


Figure 21: Box Plot of Times Out of View Results

Path Length

The next performance metrics obtained from the algorithm is the total path length of each instrument in 2D (pixels) and 3D (m). Table 4, below, shows the results from the mean, standard deviation (SD), and t-test calculations.

Table 4: Path Length Results

		Path Length 2D (pixels)			Path Length 3D (m)		
		Dominant Hand	Other Hand	Average	Dominant Hand	Other Hand	Average
Novice Group	Mean	34065	33800	33933	1.47	1.22	1.34
	SD	24325	47501	31113	1.32	1.14	1.01
Experienced Group	Mean	16139	16053	16096	0.77	0.85	0.81
	SD	8604	6274	6614	0.44	0.53	0.43
t-Test Results (p)		0.008	0.087	0.023	0.036	0.139	0.041

Both the 2D and 3D path lengths of the instrument being used in the dominant hand were found to be significantly different across the two experimental groups (p = 0.008 and p = 0.036) based on the t-test results shown in Table 4. Neither was significantly different for the instrument in the non-dominant hand (p = 0.087 and p = 0.139). The average path lengths in 2D and 3D of the two instruments were found to be significantly different across the Novice and Experienced groups (p = 0.023 and p = 0.041). In general, the path length in 2D and 3D of the instruments used by the Novice group was longer than that used by the Experienced group, as illustrated in the Box Plots shown in Figure 22 and 23.

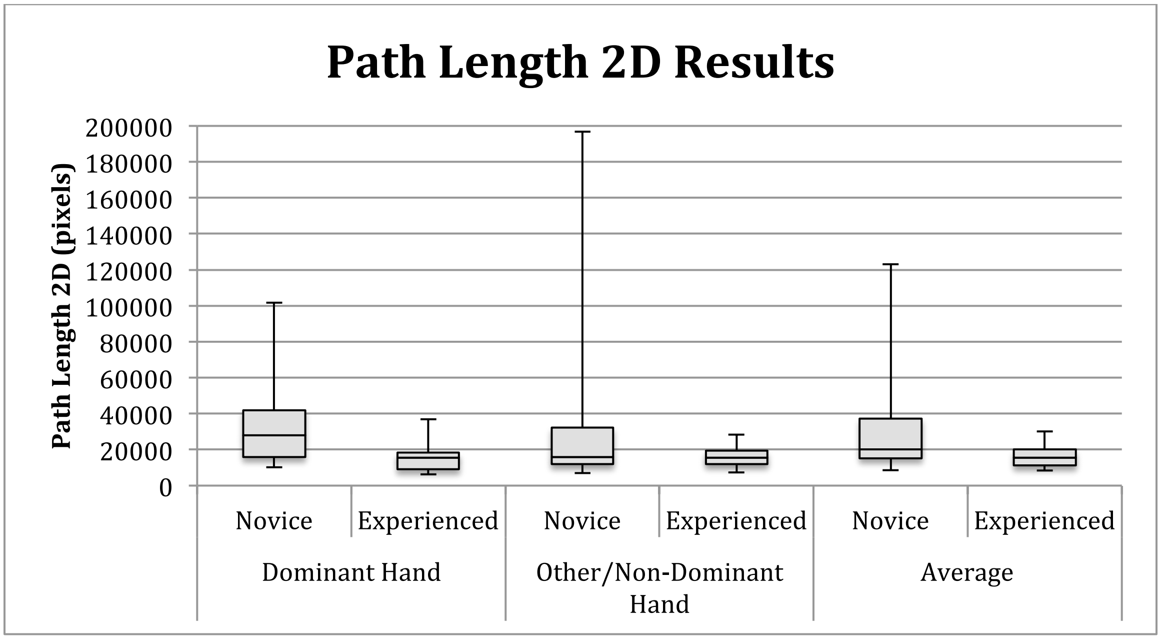


Figure 22: Box Plot of Path Length 2D Results

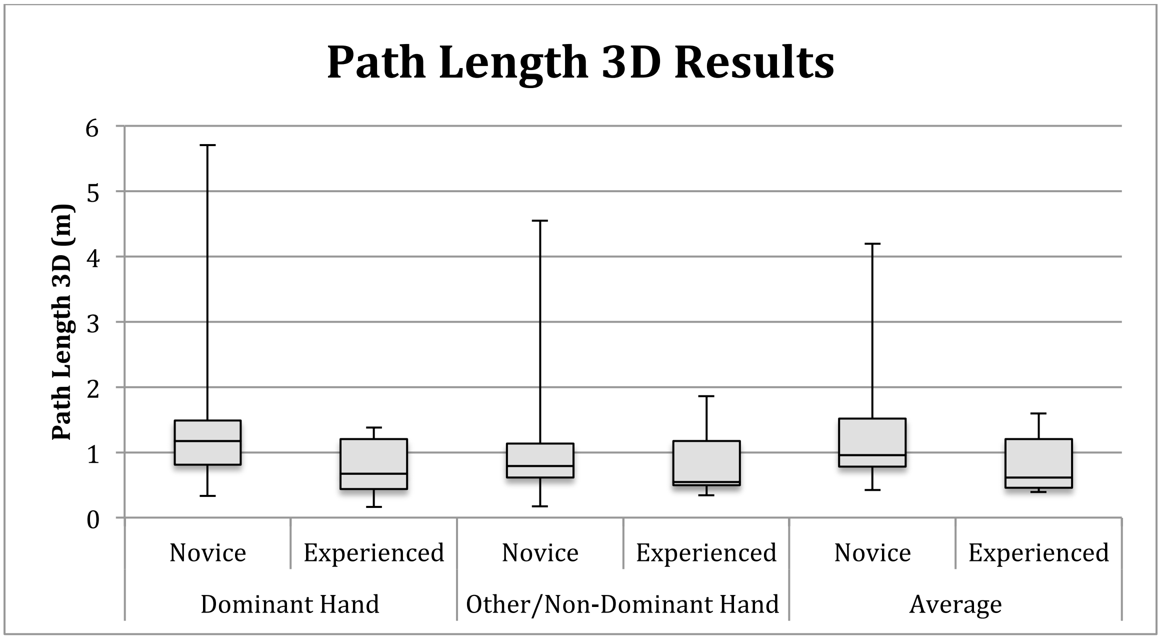


Figure 23: Box Plot of Path Length 3D Results

Average Speed

The final performance metrics obtained from the algorithm is the average speed of each instrument in 2D (pixels/s) and 3D (m/s). Table 5, below, shows the results from the mean, standard deviation (SD), and t-test calculations.

Table 5: Average Speed Results

		Average Speed 2D (pixels/s)			Average Speed 3D (mm/s)		
		Dominant Hand	Other Hand	Average	Dominant Hand	Other Hand	Average
Novice Group	Mean	219	177	198	9.65	7.37	8.38
	SD	97	112	81	5.31	3.56	3.38
Experienced Group	Mean	203	208	205	9.65	10.67	10.16
	SD	75	101	74	4.72	6.55	4.55
t-Test Results (p)		0.314	0.229	0.404	0.49	0.062	0.142

According to the results of the t-test, the average speed in 2D and 3D for each instrument was not significantly different across the Novice and Experienced groups (p = 0.314, p = 0.229, p = 0.404, p = 0.49, p = 0.062, p = 0.142). The box plots shown in Figures 24 and 25 show the there is not much variation between the two groups in regards to average instrument speed in 2D and 3D.

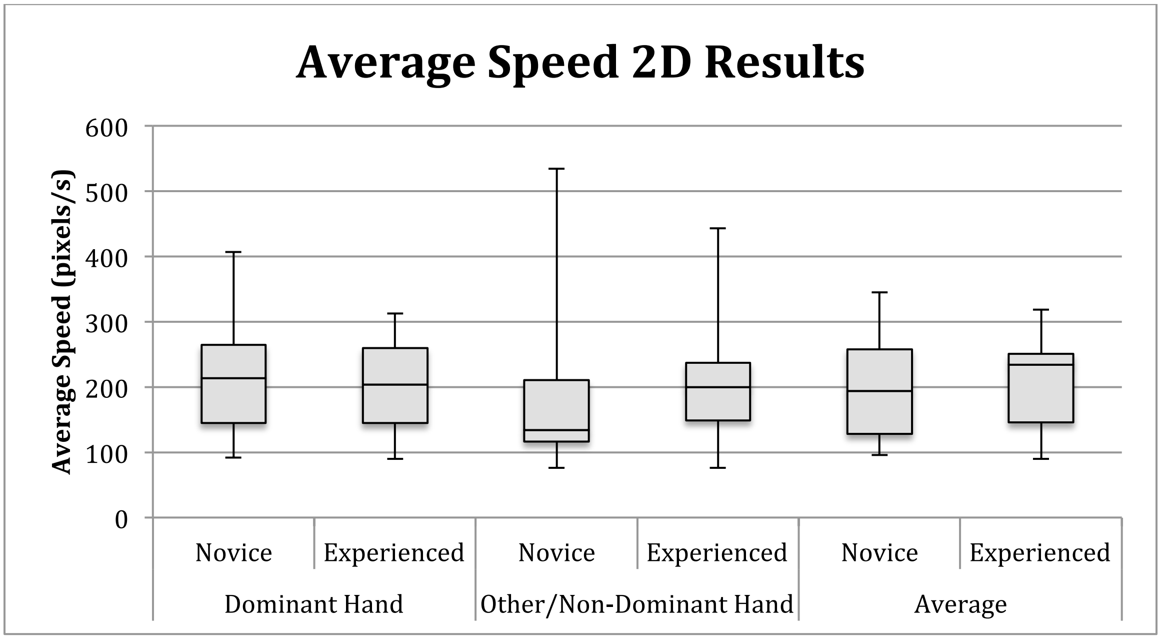


Figure 24: Box Plot of Average Speed 2D Results

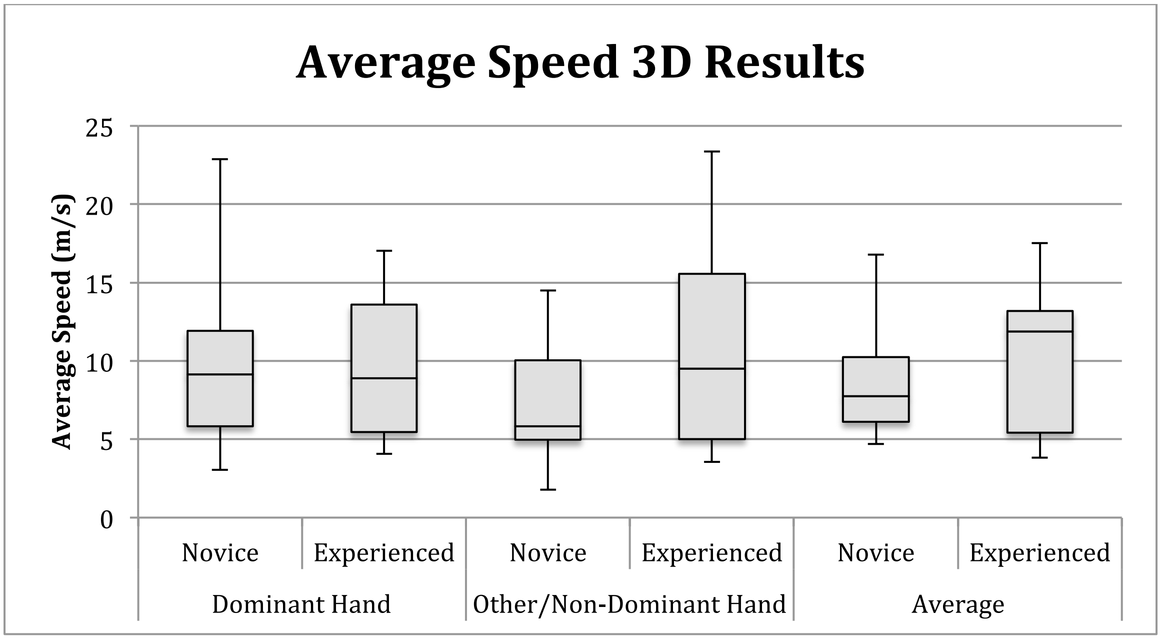


Figure 25: Box Plot of Average Speed 3D Results

V. DISCUSSION

A. Step 1

The plots in Figures 15-17 show that the algorithm correctly identifies the distance in which the instrument traveled along each axis. Therefore, the algorithm successfully tracks a laparoscopic instrument along each axis and produces accurate real world values that describe the movements. Not only does the algorithm correctly identify travel distances of the instrument along each axis, but also the direction as illustrated in Figures 15-17. It was also confirmed that the algorithm is able to track 2 instruments being used at a single time in a laparoscopic box trainer, as seen in Figure 18.

B. Step 2

The algorithm succeeded in differentiating between the Novice and Experienced groups of residents even though differences were not observed for all performance metrics. It was expected that some of the performance metrics may be similar across the two groups, but as long as significant differences exist that make sense the algorithm would be a valid method to differentiate between users of varying skill levels.

Time In

The amount of time each instrument was in the field of view to complete the peg transfer task, as provided by the algorithm, across the Novice and Experienced groups was significantly different. It is also important to note that the algorithm found the amount of time each instrument to be in the field of view was on average less for the Experienced group when compared to the Novice group. This is no surprise as the Experienced group should be completing the tasks in less time than the Novice group, as they did on average. So the algorithm is able to differentiate between users of varying skills by determining the amount of time in which each instrument is in the field of view while completing a task inside of the laparoscopic box trainer.

Time Out and Times Out of View

Only the instrument with the non-dominant hand showed a significant difference between the two groups for both the number of times the instrument left the field of view as well as the total amount of time it was out of view. That means that the instrument in the dominant hand did not show a significant difference. The Experienced group on average had fewer instances of the instrument leaving the field of view as expected, but not by much as there was not a significant difference. The algorithm actually provided results that on average had the Experienced group exiting the field with the instrument in their dominant hand more than the Novice group, which was not expected. The reason for the unexpected results is likely due to the fact that the instrument tip may still be in the field of view, but the algorithm identifies the instrument as being out of view as it uses the black shaft to determine instrument location. That is why these performance metrics are not reliable enough to be used as a method for the algorithm to differentiate between users are varying skill levels.

Path Length

Path length results obtained from the algorithm for the instrument in the dominant hand were found to be significantly different between the Novice and Experienced groups. However, it was not significantly different for the non-dominant hand. Based on the data, the Experienced group moved both instruments a relatively equal amount while the Novice group did not. A possible explanation for this is that the Novice group did not like using their non-dominant hand so they moved it as little as possible relying on their dominant hand to do most of the work while the Experienced group feels more comfortable using their non-dominant hand since they have practiced doing so during training. This is something that should be investigated in the future to determine if a comparison between the two instruments path length could be a valid method for differentiating between users of varying skill levels. For now it seems that the algorithm may only use the path length of the dominant hand to differentiate between users of varying skill levels.

Average Speed

Based on the results, the algorithm was not able to differentiate between the two groups using average speed, as there were no significant differences. The box plots show in Figure 24 and 25 actually show that the average speed is pretty similar between the two groups as well as between the dominant and non-dominant hands. So average speed is not a valid method for the algorithm to use to differentiate between users with varying skill levels.

C. Future Recommendations

Moving forward, the next step would be to implement this designed instrument tracking system into the existing ELT so that they work together in real time to provide the surgical resident with an improved training experience. Together the systems will increase the effectiveness of training by providing the user with a large amount of feedback on how they can improve, which leads to the development of a graphical user interface. A graphical user interface will allow the user to interact with the system and a way for the system to provide feedback to the user. The laparoscopic instrument tracking system will be able to provide feedback to the user pertaining to how much, how fast, and how effective their movements are with the instruments as well as whether or not they remained in the field of view. However, testing needs to be done to determine how much, how fast, and how effective a users movements should be with the instruments. It would also be beneficial to develop an algorithm that not only tracks the laparoscopic instruments being used in the lap box trainer, but to track the FLS program task being performed as well. That way the final system will have a variety of training tasks from the ELT tasks to all five FLS program tasks and be able to provide feedback about the users performance while completing each.

It is important to track the task itself as well as the instruments being used to make sure the user is provided with the optimal amount of feedback on their performance to maximize training effectiveness. Attempts were made to create algorithms to track the FLS program training tasks, but were unsuccessful. The main reason for the failed attempts is the high level of unpredictability and the lack of limitations or guidance as to how the user should complete the task. By providing the user with more guidance and adding limitations, the level of unpredictability will be decreased and an algorithm may be developed that tracks the FLS program task being performed. Examples of additional guidance that would need to be provided to the user in order to develop an algorithm to track the task is camera angle, task location in the lap box, game pieces/materials used, and instruments used. It would also be necessary to either show the camera or manually input some of the end results such as the final circle cut out during the cut test, which are not always seen by the camera in order for the system to analyze and provide results. However, if the work is done to decrease the unpredictability of the FLS program training tasks then an algorithm may be developed to track the tasks and increase training effectivity. To validate such a system a study should be performed comparing an individual's results from the developed algorithm to their actual FLS score.

VI. CONCLUSION

In the end, an optical tracking system to monitor laparoscopic instruments while being used inside of a laparoscopic box trainer was successfully developed. The algorithm performs a series of steps that are taken a frame at a time to obtain the 3D real world tracking point of the laparoscopic instrument. The algorithm then uses the 3D real world tracking points to calculate quantitative values for various aspects of the users performance that represent how effective, controlled, and safe the user's movements were. Testing confirmed that the algorithm can accurately track the distance traveled and direction of up to two laparoscopic instruments in 3D real space. It was also determined that the algorithm is capable of differentiating between users of varying skill levels by using performance metrics such as the amount of time each instrument is in the field of view and path length. As the laparoscopic instrument tracking system works as expected, the next step would be to implement it with the ELT to create a more effective training experience for the trainee.

APPENDIX A: Instrument Tracking Algorithm

```
%-------------------------------------------------------------------------%
% Title: instrumentTracking.m
% Description: Algorithm that determines the 3D real world tracking points
% of the left and right laparoscopic instruments during training.  The
% algorithm also calculates the number of times each instrument leaves the
% field of view, how long each instrument was out of view, how long the
% instrument was in view, 2D path length (pixels and m), 3D path length
% (pixels and m), 2D average speed (pixels/s and m/s), and 3D average speed
% (pixels/s and m/s).
%-------------------------------------------------------------------------%

clear all;

%--------------------------------------------------------------------%
%    OBTAINING VIDEO AND DETERMINING LENGTH, FRAME RATE, AND SIZE    %
%--------------------------------------------------------------------%

% Reading Video
video = VideoReader('PRE01.wmv');

% Determing number of frames in video
nFrames = video.NumberOfFrames;

% Determining frame rate of video
framesPerSec = video.FrameRate;

% Determining height and width of video
vidHeight = video.Height;
vidWidth = video.Width;

%---------------------------------------------------------------------%
%    DETERMINING 3D REAL WORLD TRACKING POINTS FOR EACH INSTRUMENT    %
%---------------------------------------------------------------------%

for i = 1:nFrames
    % Obtaining ith frame of video
    img = read(video,i);

    % Calling function to determine 3D location of left instrument
    [Lx(i),Ly(i),Lz(i)] = leftInstrumentTrack(img,vidWidth,vidHeight);

    % Calling function to determine 3D location of right instrument
    [Rx(i),Ry(i),Rz(i)] = rightInstrumentTrack(img,vidWidth,vidHeight);

end

% Converting pixels to mm
for j = 1:i
    if Lz(j) ~= -1
        mLx(j) = (5/Lz(j))*(Lx(j)-(vidWidth/2));
        mLy(j) = (5/Lz(j))*(Ly(j)-(vidHeight/2));
        mLz(j) = ((-0.16262)*Lz(j))+11.773;
    else
        mLx(j) = 'x';
        mLy(j) = 'x';
        mLz(j) = 'x';
    end
    if Rz(j) ~= -1
        mRx(j) = (5/Rz(j))*(Rx(j)-(vidWidth/2));
        mRy(j) = (5/Rz(j))*(Ry(j)-(vidHeight/2));
        mRz(j) = ((-0.16262)*Rz(j))+11.773;
    else
        mRx(j) = 'x';
        mRy(j) = 'x';
        mRz(j) = 'x';
    end
end

%---------------------------------------%
%    CALCULATING PERFORMANCE METRICS    %
%---------------------------------------%

% Determining when left instrument first enters field view
j = 1;
while Lz(j) == -1
    j = j + 1;
end
leftStart = j;

% Determining when left instrument exits field for last time
j = i;
while Lz(j) == -1
    j = j - 1;
end
leftEnd = j;

% Determining when right instrument first enters field of view
j = 1;
while Rz(j) == -1
    j = j + 1;
end
rightStart = j;

% Determining when right instrument exits field for last time
j = i;
while Rz(j) == -1
    j = j - 1;
end
rightEnd = j;

% Determining number of times the left instrument exited field of view and
% the total time it was out of view (frames)
leftOut = 0;
lTimeOut = 0;
j = leftStart;
while j < leftEnd
    if Lz(j) == -1
        leftOut = leftOut + 1;
        lTimeOut = lTimeOut + 1;
        j = j + 1;
        while Lz(j) == -1
            j = j + 1;
            lTimeOut = lTimeOut + 1;
        end
    else
        j = j + 1;
    end

end

% Determining number of times the right instrument exited field of view and
% the total time it was out of view (frames)
rightOut = 0;
rTimeOut = 0;
j = rightStart;
while j < rightEnd
    if Rz(j) == -1
        rightOut = rightOut + 1;
        rTimeOut = rTimeOut + 1;
        j = j + 1;
        while Rz(j) == -1
            j = j + 1;
            rTimeOut = rTimeOut + 1;
        end
    else
        j = j + 1;
    end
end

% Determining path length of left instrument in both 2D (pixels) and 3D (m)
pathLengthL2D = 0;
pathLengthL3D = 0;
for j = leftStart:leftEnd-1
    if Lz(j) ~= -1 && Lz(j+1) ~= -1
        pathLengthL2D = pathLengthL2D + sqrt((Lx(j)-Lx(j+1))^(2)+(Ly(j)-
Ly(j+1))^(2));
        PathLengthL3D = PathLengthL3D + (sqrt((mLx(j)-mLx(j+1))^(2)+(mLy(j)-
mLy(j+1))^(2)+(mLz(j)-mLz(j+1))^(2))/1000);
    end
end

% Determining path length of right instrument in both 2D (pixels) and 3D (m)
pathLengthR2D = 0;
pathLengthR3D = 0;
for j = rightStart:rightEnd-1
    if Rz(j) ~= -1 && Rz(j+1) ~= -1
        pathLengthR2D = pathLengthR2D + sqrt((Rx(j)-Rx(j+1))^(2)+(Ry(j)-
Ry(j+1))^(2));
        PathLengthR3D = PathLengthR3D + (sqrt((mRx(j)-mRx(j+1))^(2)+(mRy(j)-
mRy(j+1))^(2)+(mRz(j)-mRz(j+1))^(2))/1000);
    end
end

% Determining total time left instrument was in the field of view (s) and
% converting the amount of time it was out of view from frames to s
timeL = (leftEnd + 1 - leftStart - lTimeOut) / framesPerSec;
lTimeOut = lTimeOut / framesPerSec;

% Determining total time right instrument was in the field of view (s) and
% converting the amount of time it was out of view from frames to s
timeR = (rightEnd + 1 - rightStart - rTimeOut) / framesPerSec;
rTimeOut = rTimeOut / framesPerSec;

% Calculating average speed of left instrument in both 2D (pixels/s) and
% 3D (m/s)
avgSpeedL2D = pathLengthL2D / timeL;
AvgSpeedL3D = PathLengthL3D / timeL;

% Calculating average speed of right instrument in both 2D (pixels/s) and
% 3D (m/s)
avgSpeedR2D = pathLengthR2D / timeR;
avgSpeedR3D = pathLengthR3D / timeR;
```

APPENDIX B: Left Instrument Track Function

```
function [ xLocation,yLocation,zLocation ] = leftInstrumentTrack(
img,vidWidth,vidHeight )
%Left Instrument Track: Determines the 3D tracking point of the left
%laparoscopic instrument in the provided image.

        % Image contrast enhancement
        imgD = im2double(img);
        a = max(max(imgD(:)));
        b = min(min(imgD(:)));
        img = (imgD - b) / (a-b);

        % Color-based segmentation (black)
        for k = 1:vidWidth
            for j = 1:vidHeight
                if (img(j,k,1)<0.3 && img(j,k,2)<0.25 && img(j,k,3)<0.25)
                    imgBlack(j,k) = 255;
                else
                    imgBlack(j,k) = 0;
                end
            end
        end

        % Erosion of image
        SE = strel('rectangle',[5 10]);
        imgBlack = imerode(imgBlack,SE);

         % Creating left instrument image
        imgLeft(1:vidHeight,1:vidWidth) = 0;
        imgLeft(1:vidHeight,1) = imgBlack(1:vidHeight,1);
        imgLeft(vidHeight,1:vidWidth/2) = imgBlack(vidHeight,1:vidWidth/2);

        % Isolating left instrument using color based segmentation
        for k = 1:vidWidth-1
            for j = 1:vidHeight
                if (imgLeft(j,k) == 255)
                    if j-1 > 0 && k+1 > 0
                        if (imgBlack(j-1,k+1) == 255)
                            imgLeft(j-1,k+1) = 255;
                        end
                    end
                    if (imgBlack(j,k+1) == 255)
                        imgLeft(j,k+1) = 255;
                    end
                    if j ~= vidHeight
                        if (imgBlack(j+1,k+1) == 255)
                            imgLeft(j+1,k+1) = 255;
                        end
                    end
                end
            end
        end

        for k = 1:vidWidth/2
```

```
    if (imgLeft(vidHeight,k) == 255)
        if k ~= 1
            if (imgBlack(vidHeight-1,k-1) == 255)
                imgLeft(vidHeight-1,k-1) = 255;
            end
        end
        if (imgBlack(vidHeight-1,k) == 255)
            imgLeft(vidHeight-1,k) = 255;
        end
        if (imgBlack(vidHeight-1,k+1) == 255)
            imgLeft(vidHeight-1,k+1) = 255;
        end
    end
end

 for j = 1:vidHeight-2
    for k = 1:vidWidth
        if (imgLeft(vidHeight-j,k) == 255)
            if k ~= 1
                if (imgBlack(vidHeight-j-1,k-1) == 255)
                    imgLeft(vidHeight-j-1,k-1) = 255;
                end
            end
            if (imgBlack(vidHeight-j-1,k) == 255)
                imgLeft(vidHeight-j-1,k) = 255;
            end
            if k ~= vidWidth
                if (imgBlack(vidHeight-j-1,k+1) == 255)
                    imgLeft(vidHeight-j-1,k+1) = 255;
                end
            end
        end
    end
 end

% Dilation of eroded image
imgBlack = imdilate(imgLeft,SE);

% Isolating the instruments edges and Canny filtering
imgCanny = edge(imgBlack,'canny');

% Hough transform
[H,T,R] = hough(imgCanny);

P  = houghpeaks(H,5,'threshold',ceil(0.2*max(H(:))));
x = T(P(:,2)); y = R(P(:,1));

% Find lines
lines = houghlines(imgCanny,T,R,P,'FillGap',5,'MinLength',7);

% Determing 8 longest Hough lines
max_len1 = 0;
max_len2 = 0;
max_len3 = 0;
max_len4 = 0;
```

```
max_len5 = 0;
max_len6 = 0;
max_len7 = 0;
max_len8 = 0;
xy_long1 = 0;
xy_long2 = 0;
xy_long3 = 0;
xy_long4 = 0;
xy_long5 = 0;
xy_long6 = 0;
xy_long7 = 0;
xy_long8 = 0;

for k = 1:length(lines)
    xy = [lines(k).point1; lines(k).point2];

    % Determine the endpoints of the longest 8 line segments
    len = norm(lines(k).point1 - lines(k).point2);
    if ( len > max_len1 && length(lines) >= 1)
        temp_len = len;
        len = max_len1;
        max_len1 = temp_len;
        temp_xy = xy;
        xy = xy_long1;
        xy_long1 = temp_xy;
    end
    if ( len > max_len2 && length(lines) >= 2 && k > 1)
        temp_len = len;
        len = max_len2;
        max_len2 = temp_len;
        temp_xy = xy;
        xy = xy_long2;
        xy_long2 = temp_xy;
    end
    if ( len > max_len3 && length(lines) >= 3 && k > 2)
        temp_len = len;
        len = max_len3;
        max_len3 = temp_len;
        temp_xy = xy;
        xy = xy_long3;
        xy_long3 = temp_xy;
    end
    if ( len > max_len4 && length(lines) >= 4 && k > 3)
        temp_len = len;
        len = max_len4;
        max_len4 = temp_len;
        temp_xy = xy;
        xy = xy_long4;
        xy_long4 = temp_xy;
    end
    if ( len > max_len5 && length(lines) >= 5 && k > 4)
        temp_len = len;
        len = max_len5;
        max_len5 = temp_len;
        temp_xy = xy;
        xy = xy_long5;
```

```
                xy_long5 = temp_xy;
            end
            if ( len > max_len6 && length(lines) >= 6 && k > 5)
                temp_len = len;
                len = max_len6;
                max_len6 = temp_len;
                temp_xy = xy;
                xy = xy_long6;
                xy_long6 = temp_xy;
            end
            if ( len > max_len7 && length(lines) >= 7 && k > 6)
                temp_len = len;
                len = max_len7;
                max_len7 = temp_len;
                temp_xy = xy;
                xy = xy_long7;
                xy_long7 = temp_xy;
            end
            if ( len > max_len8 && length(lines) >= 8 && k > 7)
                max_len8 = len;
                xy_long8 = xy;
            end
        end

        % Determining characteristics of 8 longest lines obtained from Hough
        if xy_long1 > 0
            % Determing slope and y-intercept
            m(1) = (xy_long1(1,2) - xy_long1(2,2)) / (xy_long1(1,1) -
xy_long1(2,1));
            b(1) = xy_long1(1,2) - (m(1) * xy_long1(1,1));

            % Determining point where line intersects border
            point(1) = (m(1)*1) + b(1);
            pointAxis(1) = 'y';
            if point(1) > vidHeight
                point(1) = (vidHeight-b(1)) / m(1);
                pointAxis(1) = 'x';
            end
        else
            point(1) = -1000;
            pointAxis(1) = 0;
        end

        if xy_long2 > 0
            % Determing slope and y-intercept
            m(2) = (xy_long2(1,2) - xy_long2(2,2)) / (xy_long2(1,1) -
xy_long2(2,1));
            b(2) = xy_long2(1,2) - (m(2) * xy_long2(1,1));

            % Determining point where line intersects border
            point(2) = (m(2)*1) + b(2);
            pointAxis(2) = 'y';
            if point(2) > vidHeight
                point(2) = (vidHeight-b(2)) / m(2);
                pointAxis(2) = 'x';
            end
        else
            point(2) = -1000;
            pointAxis(2) = 0;
        end

        if xy_long3 > 0
            % Determing slope and y-intercept
            m(3) = (xy_long3(1,2) - xy_long3(2,2)) / (xy_long3(1,1) -
xy_long3(2,1));
            b(3) = xy_long3(1,2) - (m(3) * xy_long3(1,1));

            % Determining point where line intersects border
            point(3) = (m(3)*1) + b(3);
            pointAxis(3) = 'y';
            if point(3) > vidHeight
                point(3) = (vidHeight-b(3)) / m(3);
                pointAxis(3) = 'x';
            end
        else
            point(3) = -1000;
            pointAxis(3) = 0;
        end

        if xy_long4 > 0
            % Determing slope and y-intercept
            m(4) = (xy_long4(1,2) - xy_long4(2,2)) / (xy_long4(1,1) -
xy_long4(2,1));
            b(4) = xy_long4(1,2) - (m(4) * xy_long4(1,1));

            % Determining point where line intersects border
            point(4) = (m(4)*1) + b(4);
            pointAxis(4) = 'y';
            if point(4) > vidHeight
                point(4) = (vidHeight-b(4)) / m(4);
                pointAxis(4) = 'x';
            end
        else
            point(4) = -1000;
            pointAxis(4) = 0;
        end

        if xy_long5 > 0
            % Determing slope and y-intercept
            m(5) = (xy_long5(1,2) - xy_long5(2,2)) / (xy_long5(1,1) -
xy_long5(2,1));
            b(5) = xy_long5(1,2) - (m(5) * xy_long5(1,1));

            % Determining point where line intersects border
            point(5) = (m(5)*1) + b(5);
            pointAxis(5) = 'y';
            if point(5) > vidHeight
                point(5) = (vidHeight-b(5)) / m(5);
                pointAxis(5) = 'x';
            end
        else
            point(5) = -1000;
            pointAxis(5) = 0;
        end

        if xy_long6 > 0
            % Determing slope and y-intercept
            m(6) = (xy_long6(1,2) - xy_long6(2,2)) / (xy_long6(1,1) -
xy_long6(2,1));
            b(6) = xy_long6(1,2) - (m(6) * xy_long6(1,1));

            % Determining point where line intersects border
            point(6) = (m(6)*1) + b(6);
            pointAxis(6) = 'y';
            if point(6) > vidHeight
                point(6) = (vidHeight-b(6)) / m(6);
                pointAxis(6) = 'x';
            end
        else
            point(6) = -1000;
            pointAxis(6) = 0;
        end

        if xy_long7 > 0
            % Determing slope and y-intercept
            m(7) = (xy_long7(1,2) - xy_long7(2,2)) / (xy_long7(1,1) -
xy_long7(2,1));
            b(7) = xy_long7(1,2) - (m(7) * xy_long7(1,1));

            % Determining point where line intersects border
            point(7) = (m(7)*1) + b(7);
            pointAxis(7) = 'y';
            if point(7) > vidHeight
                point(7) = (vidHeight-b(7)) / m(7);
                pointAxis(7) = 'x';
            end
        else
            point(7) = -1000;
            pointAxis(7) = 0;
        end

        if xy_long8 > 0
            % Determing slope and y-intercept
            m(8) = (xy_long8(1,2) - xy_long8(2,2)) / (xy_long8(1,1) -
xy_long8(2,1));
            b(8) = xy_long8(1,2) - (m(8) * xy_long8(1,1));

            % Determining point where line intersects border
            point(8) = (m(8)*1) + b(8);
            pointAxis(8) = 'y';
            if point(8) > vidHeight
                point(8) = (vidHeight-b(8)) / m(8);
                pointAxis(8) = 'x';
            end
        else
            point(8) = -1000;
            pointAxis(8) = 0;
        end
```

```
% Determining distance of instrument along edge
j = 1;
wOne = [0,0];
wTwo = [0,0];
while (wOne(2) == 0 && j <= vidHeight)
    if imgBlack(j,1) == 255
       wOne = [j,1];
    end
    j = j + 1;
end
if wOne(2) == 1
    j = j + 1;
    while (wTwo(2) == 0 && j <= vidHeight)
        if imgBlack(j,1) == 0
            wTwo = [j,1];
        end
        j = j + 1;
    end
    if wTwo(2) == 0
        j = 1;
        while (wTwo(2) == 0 && j <= vidWidth/2)
            if imgBlack(vidHeight,j) == 0
                wTwo = [vidHeight,j];
            end
            j = j + 1;
        end
    end
else
    j = 1;
    while (wOne(2) == 0 && j <= vidWidth)
        if imgBlack(vidHeight,j) == 255
            wOne = [vidHeight,j];
        end
        j = j + 1;
    end
    while (wTwo(2) == 0 && j <= vidWidth)
        if imgBlack(vidHeight,j) == 0
            wTwo = [vidHeight,j];
        end
        j = j + 1;
    end
end
if wOne(2) == 1 && wTwo(2) == 1
        wDiff = abs(wTwo(1) - wOne(1));
    elseif wOne(1) == vidHeight && wTwo(1) == vidHeight
        wDiff = abs(wOne(2) - wTwo(2));
    elseif wOne(2) == 1 && wTwo(1) == vidHeight
        wDiff = sqrt((vidHeight-wOne(1))^(2) + (wTwo(2)-1)^(2));
    else
        wDiff = 0;
end

% Determining which lines combine to outline each instrument
a1C = 0;
a2C = 0;
```

```
        for j = 1:length(b)
            if pointAxis(j) == 'y'
                if sqrt((1-wOne(2))^(2)+(point(j)-wOne(1))^(2)) < sqrt((1-
wTwo(2))^(2)+(point(j)-wTwo(1))^(2)) && sqrt((1-wOne(2))^(2)+(point(j)-
wOne(1))^(2)) <= wDiff-(wDiff*0.4) %&& m(j) < -0.1 && m(j) > -0.9
                    a1C = a1C + 1;
                    a1(a1C) = j;
                elseif sqrt((1-wTwo(2))^(2)+(point(j)-wTwo(1))^(2)) <= wDiff-
(wDiff*0.4) %&& m(j) < -0.1 && m(j) > -0.9
                    a2C = a2C + 1;
                    a2(a2C) = j;
                end
            elseif pointAxis(j) == 'x'
                if sqrt((point(j)-wOne(2))^(2)+(vidHeight-wOne(1))^(2)) <
sqrt((point(j)-wTwo(2))^(2)+(vidHeight-wTwo(1))^(2)) && sqrt((point(j)-
wOne(2))^(2)+(vidHeight-wOne(1))^(2)) <= wDiff-(wDiff*0.4) && m(j) < -0.1 &&
m(j) > -0.9
                    a1C = a1C + 1;
                    a1(a1C) = j;
                elseif sqrt((point(j)-wTwo(2))^(2)+(vidHeight-wTwo(1))^(2))
<= wDiff-(wDiff*0.4) && m(j) < -0.1 && m(j) > -0.9
                    a2C = a2C + 1;
                    a2(a2C) = j;
                end
            end
        end

        a(1:2) = 0;
        if a1C ~= 0 && a2C ~= 0
            if a1(a1C) > a2(a2C)
                for j = 1:a1C
                    for k = 1:a2C
                        if (abs(point(a1(j))-point(a2(k))) > wDiff-
(wDiff*.25) && abs(point(a1(j))-point(a2(k))) < wDiff+(wDiff*.25)) && a(1) ==
0 && ((pointAxis(a1(j)) == 'y' && pointAxis(a2(k)) == 'y') ||
(pointAxis(a1(j)) == 'x' && pointAxis(a2(k)) == 'x'))
                            a(1) = a1(j);
                            a(2) = a2(k);
                        elseif (sqrt((point(a1(j))-vidHeight)^(2)+(1-
point(a2(k)))^(2)) > wDiff-(wDiff*0.25) && sqrt((point(a1(j))-
vidHeight)^(2)+(1-point(a2(k)))^(2)) > wDiff-(wDiff*0.25)) && a(1) == 0
                            a(1) = a1(j);
                            a(2) = a2(k);
                        end
                    end
                end
            else
                for k = 1:a2C
                    for j = 1:a1C
                        if (abs(point(a1(j))-point(a2(k))) > wDiff-
(wDiff*0.25) && abs(point(a1(j))-point(a2(k))) < wDiff+(wDiff*0.25)) && a(1)
== 0 && ((pointAxis(a1(j)) == 'y' && pointAxis(a2(k)) == 'y') ||
(pointAxis(a1(j)) == 'x' && pointAxis(a2(k)) == 'x'))
                            a(1) = a1(j);
                            a(2) = a2(k);
                        elseif (sqrt((point(a1(j))-vidHeight)^(2)+(1-
point(a2(k)))^(2)) > wDiff-(wDiff*0.25) && sqrt((point(a1(j))-
vidHeight)^(2)+(1-point(a2(k)))^(2)) > wDiff-(wDiff*0.25)) && a(1) == 0
                            a(1) = a1(j);
                            a(2) = a2(k);
                        end
                    end
                end
            end
        end

        % Determining two points at midlines between edges of instruments
        x1 = 0.25 * vidWidth;
        x2 = 0.75 * vidWidth;
        if a(1) > 0
            y1 = (((m(a(1))*x1 + b(a(1))) - (m(a(2))*x1 + b(a(2)))) / 2) +
(m(a(2))*x1 + b(a(2)));
            y2 = (((m(a(1))*x2 + b(a(1))) - (m(a(2))*x2 + b(a(2)))) / 2) +
(m(a(2))*x2 + b(a(2)));
        end

        % Determing slope and y-intercept of midline
        if a(1) > 0
            mMid1 = (y1 - y2) / (x1 - x2);
            bMid1 = y1 - (mMid1*x1);
        end

        % Determining 2D tracking point of left instrument
        if a(1) > 0
            for x = 1:vidWidth
                y = (mMid1*x) + bMid1;
                y = round(y);
                if (y > 0 && y < vidHeight)
                    if (imgBlack(y,x) == 0)
                        xLocation = x;
                        yLocation = y;
                        break;
                    end
                else
                    xLocation = 0;
                    yLocation = 0;
                end
            end
        end

        % Determining points on lines closest to 2D tracking point of
        % instrument 1
        if a(1) > 0
            minX1 = 0;
            minY1 = 0;
            minX2 = 0;
            minY2 = 0;
            minDist1 = 1000;
            minDist2 = 1000;
            for y = 1:vidHeight
                x1 = (y - b(a(1))) / m(a(1));
                x1 = round(x1);
                x2 = (y - b(a(2))) / m(a(2));
                x2 = round(x2);
                dist1 = sqrt((x1-xLocation)^(2) + (y-yLocation)^(2));
                dist2 = sqrt((x2-xLocation)^(2) + (y-yLocation)^(2));
                if dist1 < minDist1
                    minDist1 = dist1;
                    minX1 = x1;
                    minY1 = y;
                end
                if dist2 < minDist2
                    minDist2 = dist2;
                    minX2 = x2;
                    minY2 = y;
                end
            end
        end

        % Calculating diameter of instruments at tracking points (distance
        % between the two determined points)
        x = 0;
        if a(1) > 0
            diameter = sqrt((minX1-minX2)^(2) + (minY1-minY2)^(2));
            x = 1;
        end

        % Determining if Instrument point exists
        if a(1) > 0 && x == 1;
            zLocation = diameter;
        else
            xLocation = -1;
            yLocation = -1;
            zLocation = -1;
        end

end
```

APPENDIX C: Right Instrument Track Function

```
function [ xLocation,yLocation,zLocation ] = rightInstrumentTrack(
img,vidWidth,vidHeight )
%Right Instrument Track: Determines the 3D tracking point of the right
%laparoscopic instrument in the provided image.

        % Image contrast enhancement
        imgD = im2double(img);
        a = max(max(imgD(:)));
        b = min(min(imgD(:)));
        img = (imgD - b) / (a-b);

        % Color-based segmentation (black)
        for k = 1:vidWidth
            for j = 1:vidHeight
                if (img(j,k,1)<0.3 && img(j,k,2)<0.25 && img(j,k,3)<0.25)
                    imgBlack(j,k) = 255;
                else
                    imgBlack(j,k) = 0;
                end
            end
        end

        % Erosion of image
        SE = strel('rectangle',[5 10]);
        imgBlack = imerode(imgBlack,SE);

        % Creating right instrument image
        imgRight(1:vidHeight,1:vidWidth) = 0;
        imgRight(1:vidHeight,vidWidth-1:vidWidth) =
imgBlack(1:vidHeight,vidWidth-1:vidWidth);
        imgRight(vidHeight,vidWidth/2:vidWidth) =
imgBlack(vidHeight,vidWidth/2:vidWidth);

        % Isolating right instrument using color based segmentation
        for k = 1:vidWidth-1
            for j = 1:vidHeight
                if (imgRight(j,vidWidth-k) == 255)
                    if j-1 > 0 && vidWidth-k-1 > 0
                        if (imgBlack(j-1,vidWidth-k-1) == 255)
                            imgRight(j-1,vidWidth-k-1) = 255;
                        end
                    end
                    if vidWidth-k-1 > 0
                        if (imgBlack(j,vidWidth-k-1) == 255)
                            imgRight(j,vidWidth-k-1) = 255;
                        end
                    end
                    if j ~= vidHeight && vidWidth-k-1 > 0
                        if (imgBlack(j+1,vidWidth-k-1) == 255)
                            imgRight(j+1,vidWidth-k-1) = 255;
                        end
                    end
                end
```

```
    end
end

for k = vidWidth/2:vidWidth
    if (imgRight(vidHeight,k) == 255)
        if (imgBlack(vidHeight-1,k-1) == 255)
            imgRight(vidHeight-1,k-1) = 255;
        end
        if (imgBlack(vidHeight-1,k) == 255)
            imgRight(vidHeight-1,k) = 255;
        end
        if k ~= vidWidth
            if (imgBlack(vidHeight-1,k+1) == 255)
                imgRight(vidHeight-1,k+1) = 255;
            end
        end
    end
 end

 for j = 1:vidHeight-2
    for k = 1:vidWidth
        if (imgRight(vidHeight-j,k) == 255)
            if k ~= 1
                if (imgBlack(vidHeight-j-1,k-1) == 255)
                    imgRight(vidHeight-j-1,k-1) = 255;
                end
            end
            if (imgBlack(vidHeight-j-1,k) == 255)
                imgRight(vidHeight-j-1,k) = 255;
            end
            if k ~= vidWidth
                if (imgBlack(vidHeight-j-1,k+1) == 255)
                    imgRight(vidHeight-j-1,k+1) = 255;
                end
            end
        end
    end
 end

% Dilation of eroded image
imgBlack = imdilate(imgRight,SE);

% Isolating the instruments and Canny filtering
imgCanny = edge(imgBlack,'canny');

% Hough transform
[H,T,R] = hough(imgCanny);

P  = houghpeaks(H,5,'threshold',ceil(0.2*max(H(:))));
x = T(P(:,2)); y = R(P(:,1));

% Find lines
lines = houghlines(imgCanny,T,R,P,'FillGap',5,'MinLength',7);

% Determing 8 longest Hough lines
```

```
max_len1 = 0;
max_len2 = 0;
max_len3 = 0;
max_len4 = 0;
max_len5 = 0;
max_len6 = 0;
max_len7 = 0;
max_len8 = 0;
xy_long1 = 0;
xy_long2 = 0;
xy_long3 = 0;
xy_long4 = 0;
xy_long5 = 0;
xy_long6 = 0;
xy_long7 = 0;
xy_long8 = 0;

for k = 1:length(lines)
    xy = [lines(k).point1; lines(k).point2];

    % Determine which lines are longest
    len = norm(lines(k).point1 - lines(k).point2);
    if ( len > max_len1 && length(lines) >= 1)
        temp_len = len;
        len = max_len1;
        max_len1 = temp_len;
        temp_xy = xy;
        xy = xy_long1;
        xy_long1 = temp_xy;
    end
    if ( len > max_len2 && length(lines) >= 2 && k > 1)
        temp_len = len;
        len = max_len2;
        max_len2 = temp_len;
        temp_xy = xy;
        xy = xy_long2;
        xy_long2 = temp_xy;
    end
    if ( len > max_len3 && length(lines) >= 3 && k > 2)
        temp_len = len;
        len = max_len3;
        max_len3 = temp_len;
        temp_xy = xy;
        xy = xy_long3;
        xy_long3 = temp_xy;
    end
    if ( len > max_len4 && length(lines) >= 4 && k > 3)
        temp_len = len;
        len = max_len4;
        max_len4 = temp_len;
        temp_xy = xy;
        xy = xy_long4;
        xy_long4 = temp_xy;
    end
    if ( len > max_len5 && length(lines) >= 5 && k > 4)
        temp_len = len;
```

```
                len = max_len5;
                max_len5 = temp_len;
                temp_xy = xy;
                xy = xy_long5;
                xy_long5 = temp_xy;
            end
            if ( len > max_len6 && length(lines) >= 6 && k > 5)
                temp_len = len;
                len = max_len6;
                max_len6 = temp_len;
                temp_xy = xy;
                xy = xy_long6;
                xy_long6 = temp_xy;
            end
            if ( len > max_len7 && length(lines) >= 7 && k > 6)
                temp_len = len;
                len = max_len7;
                max_len7 = temp_len;
                temp_xy = xy;
                xy = xy_long7;
                xy_long7 = temp_xy;
            end
            if ( len > max_len8 && length(lines) >= 8 && k > 7)
                max_len8 = len;
                xy_long8 = xy;
            end
        end

        % Determining characteristics of 8 longest lines obtained from Hough
        if xy_long1 > 0
            % Determing slope and y-intercept
            m(1) = (xy_long1(1,2) - xy_long1(2,2)) / (xy_long1(1,1) - 
xy_long1(2,1));
            b(1) = xy_long1(1,2) - (m(1) * xy_long1(1,1));

            % Determining point where line intersects border
            point(1) = (m(1)*vidWidth) + b(1);
            pointAxis(1) = 'y';
            if point(1) > vidHeight
                point(1) = (vidHeight-b(1)) / m(1);
                pointAxis(1) = 'x';
            end
        else
            point(1) = -1000;
            pointAxis(1) = 0;
        end

        if xy_long2 > 0
            % Determing slope and y-intercept
            m(2) = (xy_long2(1,2) - xy_long2(2,2)) / (xy_long2(1,1) - 
xy_long2(2,1));
            b(2) = xy_long2(1,2) - (m(2) * xy_long2(1,1));

            % Determining point where line intersects border
            point(2) = (m(2)*vidWidth) + b(2);
            pointAxis(2) = 'y';
            if point(2) > vidHeight
                point(2) = (vidHeight-b(2)) / m(2);
                pointAxis(2) = 'x';
            end
        else
            point(2) = -1000;
            pointAxis(2) = 0;
        end

        if xy_long3 > 0
            % Determing slope and y-intercept
            m(3) = (xy_long3(1,2) - xy_long3(2,2)) / (xy_long3(1,1) -
xy_long3(2,1));
            b(3) = xy_long3(1,2) - (m(3) * xy_long3(1,1));

            % Determining point where line intersects border
            point(3) = (m(3)*vidWidth) + b(3);
            pointAxis(3) = 'y';
            if point(3) > vidHeight
                point(3) = (vidHeight-b(3)) / m(3);
                pointAxis(3) = 'x';
            end
        else
            point(3) = -1000;
            pointAxis(3) = 0;
        end

        if xy_long4 > 0
            % Determing slope and y-intercept
            m(4) = (xy_long4(1,2) - xy_long4(2,2)) / (xy_long4(1,1) -
xy_long4(2,1));
            b(4) = xy_long4(1,2) - (m(4) * xy_long4(1,1));

            % Determining point where line intersects border
            point(4) = (m(4)*vidWidth) + b(4);
            pointAxis(4) = 'y';
            if point(4) > vidHeight
                point(4) = (vidHeight-b(4)) / m(4);
                pointAxis(4) = 'x';
            end
        else
            point(4) = -1000;
            pointAxis(4) = 0;
        end

        if xy_long5 > 0
            % Determing slope and y-intercept
            m(5) = (xy_long5(1,2) - xy_long5(2,2)) / (xy_long5(1,1) -
xy_long5(2,1));
            b(5) = xy_long5(1,2) - (m(5) * xy_long5(1,1));

            % Determining point where line intersects border
            point(5) = (m(5)*vidWidth) + b(5);
            pointAxis(5) = 'y';
            if point(5) > vidHeight
                point(5) = (vidHeight-b(5)) / m(5);
                pointAxis(5) = 'x';
            end
        else
            point(5) = -1000;
            pointAxis(5) = 0;
        end

        if xy_long6 > 0
            % Determing slope and y-intercept
            m(6) = (xy_long6(1,2) - xy_long6(2,2)) / (xy_long6(1,1) -
xy_long6(2,1));
            b(6) = xy_long6(1,2) - (m(6) * xy_long6(1,1));

            % Determining point where line intersects border
            point(6) = (m(6)*vidWidth) + b(6);
            pointAxis(6) = 'y';
            if point(6) > vidHeight
                point(6) = (vidHeight-b(6)) / m(6);
                pointAxis(6) = 'x';
            end
        else
            point(6) = -1000;
            pointAxis(6) = 0;
        end

        if xy_long7 > 0
            % Determing slope and y-intercept
            m(7) = (xy_long7(1,2) - xy_long7(2,2)) / (xy_long7(1,1) -
xy_long7(2,1));
            b(7) = xy_long7(1,2) - (m(7) * xy_long7(1,1));

            % Determining point where line intersects border
            point(7) = (m(7)*vidWidth) + b(7);
            pointAxis(7) = 'y';
            if point(7) > vidHeight
                point(7) = (vidHeight-b(7)) / m(7);
                pointAxis(7) = 'x';
            end
        else
            point(7) = -1000;
            pointAxis(7) = 0;
        end

        if xy_long8 > 0
            % Determing slope and y-intercept
            m(8) = (xy_long8(1,2) - xy_long8(2,2)) / (xy_long8(1,1) -
xy_long8(2,1));
            b(8) = xy_long8(1,2) - (m(8) * xy_long8(1,1));

            % Determining point where line intersects border
            point(8) = (m(8)*vidWidth) + b(8);
            pointAxis(8) = 'y';
            if point(8) > vidHeight
                point(8) = (vidHeight-b(8)) / m(8);
                pointAxis(8) = 'x';
            end
```

```
else
    point(8) = -1000;
    pointAxis(8) = 0;
end

% Determining distance of instrument along edge
j = 1;
wOne = [0,0];
wTwo = [0,0];
while (wOne(2) == 0 && j <= vidHeight)
    if imgBlack(j,vidWidth) == 255
       wOne = [j,vidWidth];
    end
    j = j + 1;
end
if wOne(2) == vidWidth
    j = j + 1;
    while (wTwo(2) == 0 && j <= vidHeight)
        if imgBlack(j,vidWidth) == 0
            wTwo = [j,vidWidth];
        end
        j = j + 1;
    end
    if wTwo(2) == 0
        j = vidWidth;
        while (wTwo(2) == 0 && j >= vidWidth/2)
            if imgBlack(vidHeight,j) == 255
                wTwo = [vidHeight,j];
            end
            j = j - 1;
        end
    end
else
    j = vidWidth;
    while (wOne(2) == 0 && j >= vidWidth/2)
        if imgBlack(vidHeight,j) == 255
            wOne = [vidHeight,j];
        end
        j = j - 1;
    end
    while (wTwo(2) == 0 && j >= vidWidth/2)
        if imgBlack(vidHeight,j) == 0
            wTwo = [vidHeight,j];
        end
        j = j - 1;
    end
end
if wOne(2) == vidWidth && wTwo(2) == vidWidth
    wDiff = abs(wTwo(1) - wOne(1));
elseif wOne(1) == vidHeight && wTwo(1) == vidHeight
    wDiff = abs(wTwo(2) - wOne(2));
elseif wOne(2) == vidWidth && wTwo(1) == vidHeight
    wDiff = sqrt((vidHeight-wOne(1))^(2) + (vidWidth-wTwo(2))^(2));
else
    wDiff = 0;
end
```

```
        % Determining which lines combine to outline each instrument
        a1C = 0;
        a2C = 0;
        for j = 1:length(b)
            if pointAxis(j) == 'y'
                if sqrt((vidWidth-wOne(2))^(2)+(point(j)-wOne(1))^(2)) <
sqrt((vidWidth-wTwo(2))^(2)+(point(j)-wTwo(1))^(2)) && sqrt((vidWidth-
wOne(2))^(2)+(point(j)-wOne(1))^(2)) <= wDiff-(wDiff*0.4) && m(j) > 0.1 &&
m(j) < 0.9
                    a1C = a1C + 1;
                    a1(a1C) = j;
                elseif sqrt((vidWidth-wTwo(2))^(2)+(point(j)-wTwo(1))^(2)) <=
wDiff-(wDiff*0.4) && m(j) > 0.1 && m(j) < 0.9
                    a2C = a2C + 1;
                    a2(a2C) = j;
                end
            elseif pointAxis(j) == 'x'
                if sqrt((point(j)-wOne(2))^(2)+(vidHeight-wOne(1))^(2)) <
sqrt((point(j)-wTwo(2))^(2)+(vidHeight-wTwo(1))^(2)) && sqrt((point(j)-
wOne(2))^(2)+(vidHeight-wOne(1))^(2)) <= wDiff-(wDiff*0.4) && m(j) > 0.1 &&
m(j) < 0.9
                    a1C = a1C + 1;
                    a1(a1C) = j;
                elseif sqrt((point(j)-wTwo(2))^(2)+(vidHeight-wTwo(1))^(2))
<= wDiff-(wDiff*0.4) && m(j) > 0.1 && m(j) < 0.9
                    a2C = a2C + 1;
                    a2(a2C) = j;
                end
            end
        end

        a(1:2) = 0;
        if a1C ~= 0 && a2C ~= 0
            if a1(a1C) > a2(a2C)
                for j = 1:a1C
                    for k = 1:a2C
                        if (abs(point(a1(j))-point(a2(k))) > wDiff-
(wDiff*.25) && abs(point(a1(j))-point(a2(k))) < wDiff+(wDiff*.25)) && a(1) ==
0 && ((pointAxis(a1(j)) == 'y' && pointAxis(a2(k)) == 'y') ||
(pointAxis(a1(j)) == 'x' && pointAxis(a2(k)) == 'x'))
                            a(1) = a1(j);
                            a(2) = a2(k);
                        elseif (sqrt((point(a1(j))-vidHeight)^(2)+(vidWidth-
point(a2(k)))^(2)) > wDiff-(wDiff*0.25) && sqrt((point(a1(j))-
vidHeight)^(2)+(vidWidth-point(a2(k)))^(2)) > wDiff-(wDiff*0.25)) && a(1) ==
0
                            a(1) = a1(j);
                            a(2) = a2(k);
                        end
                    end
                end
            else
                for k = 1:a2C
                    for j = 1:a1C
                        if (abs(point(a1(j))-point(a2(k))) > wDiff-
(wDiff*.25) && abs(point(a1(j))-point(a2(k))) < wDiff+(wDiff*.25)) && a(1) ==
0 && ((pointAxis(a1(j)) == 'y' && pointAxis(a2(k)) == 'y') ||
(pointAxis(a1(j)) == 'x' && pointAxis(a2(k)) == 'x'))
                            a(1) = a1(j);
                            a(2) = a2(k);
                        elseif (sqrt((point(a1(j))-vidHeight)^(2)+(vidWidth-
point(a2(k)))^(2)) > wDiff-(wDiff*0.25) && sqrt((point(a1(j))-
vidHeight)^(2)+(vidWidth-point(a2(k)))^(2)) > wDiff-(wDiff*0.25)) && a(1) ==
0
                            a(1) = a1(j);
                            a(2) = a2(k);
                        end
                    end
                end
            end
        end

        % Determining two points at midlines between edges of instruments
        x1 = 0.25 * vidWidth;
        x2 = 0.75 * vidWidth;
        if a(1) > 0
            y3 = (((m(a(1))*x1 + b(a(1))) - (m(a(2))*x1 + b(a(2)))) / 2) +
(m(a(2))*x1 + b(a(2)));
            y4 = (((m(a(1))*x2 + b(a(1))) - (m(a(2))*x2 + b(a(2)))) / 2) +
(m(a(2))*x2 + b(a(2)));
        end

        % Determing slope and y-intercept of midline
        if a(1) > 0
            mMid2 = (y3 - y4) / (x1 - x2);
            bMid2 = y3 - (mMid2*x1);
        end

        % Determining 2D tracking point of instrument 2
        if a(1) > 0
            for x = 1:vidWidth
                y = (mMid2*(vidWidth-x)) + bMid2;
                y = round(y);
                if (y > 0 && y < vidHeight)
                    if (imgBlack(y,vidWidth-x) == 0)
                        xLocation = vidWidth-x;
                        yLocation = y;
                        break;
                    end
                else
                    xLocation = 0;
                    yLocation = 0;
                end
            end
        end

        % Determining points on lines closest to 2D tracking point of
        % instrument 2
        if a(1) > 0
            minX3 = 0;
            minY3 = 0;
            minX4 = 0;
            minY4 = 0;
            minDist3 = 1000;
            minDist4 = 1000;
            for y = 1:vidHeight
                x1 = (y - b(a(1))) / m(a(1));
                x1 = round(x1);
                x2 = (y - b(a(2))) / m(a(2));
                x2 = round(x2);
                dist1 = sqrt((x1-xLocation)^(2) + (y-yLocation)^(2));
                dist2 = sqrt((x2-xLocation)^(2) + (y-yLocation)^(2));
                if dist1 < minDist3
                    minDist3 = dist1;
                    minX3 = x1;
                    minY3 = y;
                end
                if dist2 < minDist4
                    minDist4 = dist2;
                    minX4 = x2;
                    minY4 = y;
                end
            end
        end

        % Calculating diameter of instruments at tracking points (distance
        % between the two determined points)
        x = 0;
        if a(1) > 0  && yLocation > 1
            diameter = sqrt((minX3-minX4)^(2) + (minY3-minY4)^(2));
            x = 1;
        end

        % Determining if Instrument point exists
        if a(1) > 0 && x == 1
            zLocation = diameter;
        else
            xLocation = -1;
            yLocation = -1;
            zLocation = -1;
        end

end
```

BIBLIOGRAPHY

1. *Laparoscopy Simulators.* **Undre, Shabnam FRCS, and Ara Darzi, KBE, FRCS.** 2007, Journal of Endourology.

2. *Impact of Fundamentals of Laparoscopic Surgery Training During Medical School on Performance by First Year Surgical Residents.* **Edelman, David A. M.D., Mark A. Mattos M.D., David L. Bouwman M.D.** 2011, Association for Academic Surgery, pp. 6-9.

3. *A tale of two trainers: virtual reality versus a video trainer for acquisition of basic laparoscopic skills.* **Debes, AJ, Aggarwal R, Balasundaram I, Jacobsen MB.** 2010, American Journal of Surgery, pp. 840-845.

4. *Laparoscopic virtual reality and box trainers.* **Munz, Y. Kumar, B. Moorthy, K. Bann, S. Darzi, A.** 2004, Surgical Endoscopy, pp. 485-494.

5. *Learning basic laparoscopic skills: A randomized controlled study comparing box trainer, virtual reality simulator, and mental training.* **Mulla, M. Sharma, D. Moghul, M. Kailani, O. Dockery, J. Ayis, S. Grange, P.** 2012, Journal of Surgical Education, pp. 190-195.

6. *Review of available methods of simulation training to facilitate surgical education.* **Bashankaev, Badma, Sergey Baido, and Steven D. Wexner.** 2010, Surgical Endoscopy, pp. 28-35.

7. *Homemade laparoscopic simulators for surgical trainees.* **Khine, M. Leung, E. Morran, C. Muthukumarasamy, G.** 2011, The Clinical Teacher, pp. 118-121.

8. *Development and validation of a comprehensive program of education and assessment of the basic fundametals of laparoscopic surgery.* **Jeffrey, H. Gerald, M. Swanstrom, L. Soper, N. Sillin, L. Schirmer, B. Hoffman, K.** 2004, Journal of Surgical Research, pp. 21-27.

9. *Evaluating laparoscopic skills: setting the pass/fail score for the MISTELS system.* **Fraser, S. Klassen, D. Feldman, L. Ghitulescu, G. Stanbridge, D. Fried, G.** 2003, Surgical Endoscopy, pp. 964-967.

10. *Proving the value of simulation in laparoscopic surgery.* **Fried, G. Feldman, L. Vassiliou, M.** 2004, Ann Surg, pp. 518-525.

11. *Two-year skill retention and certification exam performace after fundametals of laparoscopic skills training and proficiency maintenance.* **Mashaud, L. Castellvi, A. Hollett, L. Hogg, D. Tesfay, S. Scott, D.** 2010, Journal of Surgical Research, pp. 194-201.

12. *Proving the effectiveness of virtual reality simulation for training in laparoscopic surgery.* **Aggarwal, R. Ward, J. Balasundaram, I. Sains, P. Athanasiou, T. Darzi, A.** 2007, Annals of Surgery, pp. 771-779.

13. *Construct validity and assessment of the learning curve for the SIMENDO endoscopic simulator.* **Verdaasdonk, E. Stassen, L. Schijven, M. Dankelman, J.** 2007, Surgical Endoscopy, pp. 1406-1412.

14. *A Novel Laparoscopic Training Device.* **Shields, P.** 2012, Masters Theses, Paper 20.

15. Objective assessment of laparoscopic suturing skills using a motion-tracking system. **Yamaguchi, S. Yoshida, D. Kenmotsu, H. Yasunaga, T. Konishi, K. Ieiri, S. Nakashima, H. Tanoue, K. Hashizume, M.** 2011, Surgical Endoscopy, pp. 771-775.

16. *A new laparoscopic surgical skill assesssment system for bi-hand coordination using magnetic tracking systems.* **Uemura, M. Tomikawa, M. Nagao, Y. Ieiri, S. Hashizume, M.** 2012, Internation Journal of Computer Assested Radiology and Surgery, pp. 139-140.

17. *Design of inertial tracking system for laparoscopic instrument trajectory analysis.* **Gan, C.** 2010, EE 4BI6 Electrical Engineering Biomedical Capstones, paper 35.

18. *Motion planning system for minimally invasive surgery.* **Haniffa, H. Hamilton, A.** 2007, Proceeding of the 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems.

19. *EVA: Laparoscopic instrument tracking based on endoscopic video analysis for psychomotor skills assessment.* **Oropesa, I. Sanchez-Gonzalez, P. Chmarra, M. Lamata, P. Fernandez, A. Sanchez-Margallo, J. Jansen, F. Dankelman, J. Sanchez-Margallo, F. Gomez, E.** 2013, Surgical Endoscopy, pp. 1029-1039.