12-2015

Development of a Suturing Simulation Device for Synchronous Acqusition of Data

Tanmay Kavathekar
Clemson University

DEVELOPMENT OF A SUTURING SIMULATION DEVICE FOR SYNCHRONOUS ACQUISITION OF DATA

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
Electrical Engineering

---

by
Tanmay Kavathekar
December 2015

Accepted by:
Dr. Richard Groff, Committee Chair
Dr. Ravikiran Singapogu
Dr. Ian Walker

ABSTRACT

There have been tremendous technological advancements in the field of surgery with new devices and minimally invasive techniques rapidly being developed. As a result, there is a corresponding need to train novice surgeons and residents to use these new technologies. Due to new regulations in medical education, an increasing the amount of surgical skills training is designed for outside the operation room using surgical simulators.

In this work, a device called the suture platform was conceptualized for assessing and training basic suturing skills of medical students and novice surgeons. In the traditional approach of “open” surgery, which has not benefitted as much from simulation, suturing is one of the most foundational surgical maneuvers. The specific task developed on the suture platform is called radial suturing and was prescribed by expert surgeons as one of five core “open” vascular skills.

In the initial phase of the platform development, a six-axis force sensor was used to obtain data on the device and the procedure was video-recorded for analysis. Pilot data was analyzed using force-based parameters (e.g. peak force) and temporal parameters with the goal of examining if experts were distinguished from novices. During analysis, it became apparent that future development of the device should focus on obtaining synchronized data from video and other sensors. In the next phase of development, a motion sensor was added to capture wrist motion of the trainee and to obtain richer information of the suturing process. The current system consists of a graphical user interface (GUI) that captures data during a radial suturing task that can be analyzed using force, motion and vision metrics to assess and inform surgical suturing skill training.

DEDICATION

This thesis is dedicated to family for their constant support, love and encouragement.

ACKNOWLEDGMENTS

First and foremost I offer my sincerest gratitude to my professors, Dr. Groff, Dr. Burg and Dr. Singapogu, who have supported me throughout my thesis with their patience and knowledge. I would like to thank Dr. Burg for giving me an opportunity to conduct research that has helped learn and grow. I would like to thank Dr. Groff for his insightful comments and encouragement. I would like to especially thank Dr. Singapogu for his continuous encouragement; patience and motivation in helping me get my research to fruition. I would also like to thank my lab colleagues (Karla and Irfan) for helping with the research. I would like to thank my family who have put faith in me and gave me an opportunity for completing my graduate studies. I am thankful to my friends who have helped me throughout my graduate student life in general.

TABLE OF CONTENTS

Table of Contents (Continued) Page

LIST OF TABLES

LIST OF FIGURES

List of Figures (Continued)

Figure Page

List of Figures (Continued)

Figure Page

CHAPTER ONE

INTRODUCTION

Sir William Halsted introduced a German-style residency training system at Johns Hopkins Hospital in 1889 that laid an emphasis on graded responsibility [1]. This format of residency system has remained a keystone of surgical training in North America since its inception. However, with the tremendous growth in technical advancement of surgical devices and techniques, many are questioning the efficacy in relying on this model of training [2].

In the current surgical training environment, opportunities to work with real patients have diminished. In some cases, guidelines and practice patterns in medicine have also changed, e.g., surgery to treat peptic ulcer that was on the general surgical list, is hardly performed now [3]. There is also considerable discussion regarding optimizing the resources allocated to train students in the operating room or labs. This focus is partly due to the outcomes-based system of healthcare assessment in current society. Consequently, medical errors are highly scrutinized and reported. More patients are taking the initiative in becoming better informed of their medical conditions due to increased access to medical information. Trainee error is not accepted easily and there has being an increase in number of legal claims being made by patients dissatisfied with their treatment [4].

The advancement in medical knowledge has also resulted in new techniques and procedure that are less invasive but difficult to learn. Haptic (force) feedback and 3D viewing of information are the modes of feedback that are heavily relied upon while gaining new “open” surgical skills. As these senses cannot be used during minimally invasive surgeries, surgeons are at a huge disadvantage. As a result, adequate practice is required before undertaking minimally invasive surgery in a safe manner. One additional consideration is the reduction of working hours (80 hours a week) that limits the time available to provide adequate exposure to certain procedures. This has caused residency programs to increase in duration and also create a number of subspecialties that require further specialized training [4]. As a cumulative consequence of all these factors an increasing amount of surgical skills training must now take place outside the operating room. Thus, simulators and simulation methods are an exciting new era of surgical training [3].

Although it was believed that surgical expertise is an innate ability of individuals, empiric research has confirmed that repeating tasks increases proficiency, and aids in maintaining the high level of skill [4]. Deliberate practice requires focus on a well-defined task to improve performance. It also requires repeated practice along with coaching and feedback on performance [2].

Simulators can be considered to be instruments that reproduce, under artificial conditions, components of surgical tasks that are likely to occur under normal circumstances [3]. They have been efficient in the airline industry to train pilots. Simulators can be used for practical training before performing supervised operation. They can be used to practice complications and surgical emergencies and their management [3]. Thus building a simulator device that can be used to train suturing techniques and acquire data to provide feedback has been the motivation of the research project.

The device called the suture platform was conceptualized for training basic surgical skills to medical students and novice surgeons. Its targeted area of surgical specialty is “open” surgery, which has not benefitted as much from simulation. The specific skill pattern adopted for training is based on radial suturing and is one of the one of the five core open vascular skills adopted by the expert vascular surgeons. The platform has gone through several iterations starting from a single stamped square suture patch held by clips to a more sturdy and robust device with sensors for gathering information. The current platform consists of a suture disc mounted inside a cylinder at three different levels. The disc holds the suture patch with radial (or a “clock face”) pattern marked on it under tension. The cylinder is attached to a custom-designed base that is mounted on a force sensor (ATI Mini-40; ATI Industrial, Apex, NC). The trainee wears an InertiaCube (v.4; Thales Visionix, Inc., Billerica, MA) device that measures hand acceleration while using the needle holder. It is mounted on the dorsum of the hand. A camera (Firefly MV, Point Grey Research Inc., Richmond, BC, Canada) records the motion of the needle on the suture patch for analysis. The data that is being collected can be synchronized using a custom-built GUI (MATLAB v.2014). The GUI also provides tools for the analysis of the data, for e.g. average forces, peak forces, finding peaks (local maxima).

CHAPTER TWO

LITERATURE REVIEW

There has been a tremendous growth in medical knowledge and a lot of pressure in the clinical environment to keep in step with new technologies [2]. The advancement in medical knowledge has also resulted in new techniques and procedures that are less invasive but difficult to learn. The consequences of constraints of time, money and other resources along with the advances in medical knowledge has resulted in an increasing amount of surgical skills training to be taking place outside the theatre and in a simulated environment (laboratories) and on simulated patients or tissue [2]–[4]. Simulators can be considered as instruments that approximate, under artificial conditions, components of surgical tasks that are encountered during surgical procedures. Simulation-based methods have been demonstrated to be efficient in the airline industry to train pilots. In surgical training, they have been proposed for practical training of residents and novices before performing supervised operations [3]. They can be used to practice complications and surgical emergencies and their management. In this chapter, a brief overview of the various approaches to skill acquisition is provided along with specific aspects for surgical skills acquisition.

1. Theories for skill acquisition

As can be expected, the ability to perform surgery requires proficiency in many domains. Technical skills is one of these skill domains and is very important to surgical capability. Various theories have been developed to explain acquisition of skills and they are mentioned below

Kopta’s theory [4] involves three phases of motor skills acquisition. The first is the cognitive phase where one observes new procedures, and gains knowledge by reading, listening and asking questions. The second step is the integrative phase, where the trainee receives feedback and learns to integrate the knowledge with the appropriate motor responses. The autonomous phase is the last phase, where continued practice without cognitive inputs results in efficient performance input. Kopta’s theory stresses on the importance of observation followed by practice.

Schmidt’s theory [4] suggests that motor skills are acquired on the basis of previous experiences. Hence, the first phase is planning of the movement with respect to the current environment. In the second phase, specific muscle commands are generated to perform the movement. The third phase is feedback (tactile, visual and auditory), and the final phase is knowledge acquisition of the outcome. Both practice and feedback are necessary according to Schmidt’s schema theory.

In the traditional apprenticeship model [4], the student first observes the mentor and then begins to practice surgical tasks under the mentor. After initial exposure, mentors only provide hints or feedback to trainees. The traditional apprenticeship model shows the importance of the gradual attainment of skills.

In the cognitive apprenticeship model [4], derived from the traditional apprenticeship model, the mentor has three roles (modeling, coaching and scaffolding) and student has three roles (articulating, reflecting and exploring). In Modeling, the mentor explains the process and the trainee begins performing the task while the mentor provides feedback and suggestions to improve the student’s performance. The student then replicates from the performance of the expert. Support is provided till independent performance is gradually obtained. The mentor’s role is gradually reduced with the improving performance of the trainee. Eventually, the student’s knowledge reaches a point at which they can synthesize and articulate the information. During reflection, the student compares his or her own problem solving ability with an expert or another student. Reproduction of the performance (recording video and using it for analysis) can help in reflection.

Ericsson’s model [4] highlights the importance of focused attention and deliberate practice in acquiring expert skill. Ericsson found that time of day was an important factor in successful learning of skills (ability to perform complex cognitive activities was the highest in morning). Ericsson also recognized the importance of rest (practice > 4 hours/day causes fatigue). These two points could help in planning the training regime. In summary, looking at the various theories we can see that deliberate practice and feedback play a vital role in surgical training. Simulators fit into the role very well because of the fact that can be used repeatedly to provide feedback essential for training.

2. Effectiveness of simulator training

Studies have been conducted by *Stefanidis et al.* which show that automaticity in performing the task and proficiency based training on simulators does result in long-lasting skills [5]. They also go on to prove that simulators can be used effectively for surgical training [6]. At the same time studies also in hint that overlearning and training can also lead to skill degradation. There is also skill degradation over the period of time and hence practicing skills learnt at regular intervals [6] has been suggested. Simulator training has helped participants to handle their stress in a better manner [5]. We can appreciate the fact that simulator requires less human intervention which saves on a surgeons time.

3. Simulator models

The type of models that can be used for simulations have been summarized below along with their advantages and disadvantages

Bench models are cheap, portable and reusable models with minimum risk and supervision. These models are low fidelity models and are meant for novice learners to learn basic and discrete skill, e.g., ethylene vinyl acetate (EVA) for suturing techniques. Fidelity of a model refers to how realistic can it be with respect to the human body [2].

The next type of "simulators" is the use of *live animals*. They can be used to practice homoeostasis and entire surgical procedures. The drawbacks of using animals are ethical concerns, costs, maintenance of special facilities, single use and anatomical differences. One of their unique uses is to improve knowledge of blood flow and dissection skills [2].

Cadavers can be considered to be a high fidelity and true anatomy simulators. They can be used to practice entire operations. This mode of simulation too faces problems of cost, availability and single use. Compliance of tissue and infection risk is its other drawbacks. It can be used to gain advanced procedural knowledge and learn about dissection [2].

Human performance simulators are one of the other types of simulators that are in current use. The advantages of these systems are reusability, high fidelity, data capture and interactivity. The cost and maintenance of these simulators is high and there are limited applications. They have mostly being used for team management and crisis training [2].

Virtual reality surgical simulators are reusable, perform data capture and require minimal setup time. Cost and maintenance of these devices is high and the three dimensions are typically not well simulated. They are not easily accepted by trainees because of their lack of realism. These devices have mostly been used in teaching basic laparoscopic, endoscopic and transcutaneous skills [2].

Looking at these models will help us gain insight on the various approaches that can be used for the development of a cheap and efficient simulator.

4. Current surgical simulators

The use of minimally invasive surgery (MIS) has been rapidly increasing because it offers many advantages over open surgery. More effective and affordable training is required due to the complexity of MIS [7]. Simulators like box trainers with motion detection and virtual reality trainers have been developed for surgical skill training. A few examples of the simulation devices in the literature are listed below.

The TrEndo device consists of a gimbal mechanism with three optical computer-mouse sensors. The gimbal guides the MIS instrument, while optical sensors measure the movements of the instrument. The resolution for the device is 0.06 mm for translation and 1.27° for rotation of the MIS instrument around its axis; the angle for rotation around incision point is 0.23°. The accuracy of the sensor in TrEndo is higher than 95% [8].

Another example of an “augmented reality” simulator is the ProMIS. It consists of a torso-shaped metallic mannequin with neoprene cover connected to a computer. The mannequin contains three separate camera-tracking systems, arranged to identify any instrument inside the simulator from three different angles. The laparoscopic camera serves as the main viewing camera displayed on the computer screen for subjects and is positioned at the mannequin's pubic region looking towards the head. The camera tracking systems capture instrument motion with Cartesian coordinates in the x, y, and z planes at an average rate of 30 frames per second (fps). The two pieces of yellow electrical tape at distal end of instrument shaft serve as a reference point for the camera tracking systems. It has evidence demonstrating some efficacy in differentiating between experts and novices, and has been suggested for use in skills training [9].

The SIMENDO (Simulator for Endoscopy) consists of one instrument handle on a box. The training program has exercises designed to train hand–eye coordination using abstract tasks without force feedback. The users manipulate a virtual endoscope or instruments during laparoscopic surgery and can choose between four different tasks: piling up of cylinders, manipulation of a 30° endoscope, clipping an artery, and dissecting a gallbladder. A game called “catch the needles” is also featured in the system [10].

Tim Horeman et al. [11] have developed a force-based system for the assessment of laparoscopic skills. For the sake of discussion, the force platform construction can be divided into hardware for the platform and software for data acquisition. The software consists of a C++ program that records rotation and translation vectors at 60 Hz. The hardware consists of a base plate fixed to the housing of a SpaceNavigator 3D mouse and the table (upper plate) is mounted on the cap of the SpaceNavigator. The SpaceNavigator is used for 3D movement in virtual environments. Horeman and colleagues modified the device by placing springs mounted between the table and the base surrounding the SpaceNavigator, which are held using spring holders. In this setup, the stiffness of the springs determines the force range. Calibration of the force platform is done using standard weights. A frame was built to exert well-defined force and torques, in all directions, at the center of the platform. Each axis was calibrated three times using regression-based methods. The accuracy of the platform was also checked using a set of forces and torques [11].

5. Parameters for skill training

A number of studies have been conducted for finding suitable parameters on simulator devices that distinguish between expert and novice skill levels. Several studies have explored the possibility of using images, motion and force data on simulators to determine these parameters. A few studies are surveyed below that are pertinent to the device developed in this work.

5.1 Force and motion parameters

Horeman et al. [11] conducted a study where a force-based system was used to assess surgical skills. They observed that surgeons mostly used rotation to drive the needle while the novices used rotation and translation to drive the needle. Furthermore all novices pressed the needle driver against the tissue during the task. The average maximum force and the average mean force for the surgeon was relatively lower that the novices [11]. Similarly, another study conducted by *Horeman et al.* [12] explored whether motion parameters correlated with force parameters moreover if a combination of force, time and motion parameters can be used to assess skills of the trainee. Of the many parameters that were explored for skill analysis, they found that path length (PL)-left, PL-right, maximum absolute force (MAF), task time, PL-left, PL-right, max force area (MFA) and mean distance between tips (MDBT) were significantly different between experts and novices for tissue attachment under traction (task 1). For the specific task of placement of silicone wire in the study, task time, PL-left and PL-right and out of view time (OVT)-left were significantly different between expert and novices [12].

Dubrowski et al. [13] conducted a study in which six junior surgical residents (novices) and seven faculty surgeons (experts) participated. They were asked to perform twenty sutures in an artificial artery tissue model. They novices were given one demonstration and verbal instructions before the task. A magnetic marker secured to dorsum of hand (right only) and the tissue model was placed on a 6D force/torque sensor. The parameters that were identified for skill assessment were peak velocity, roll, peak and average forces, roll-force delay and time. The expert's show greater wrist rotation, short roll-force delay, high average forces and short suturing time [13].

Pagador et al. [14] also conducted a study with motion parameters using a box trainer for knot tying skills. The task was divided into 4 subtasks, i.e. needle puncture, first knot, second knot and third knot. The bot trainer has two surgical tools for which individual data was collected. In this case, total time, total path length and number of movements for each tool were the parameters that differentiated between skill levels.

Dosis et al. [15] also used motion analysis for assessing skills using the Imperial College Surgical Assessment Device (ISCAD). In a recent study, *Sanchez et al.* [16] used the popular iPhone for capturing the hand motion for skill assessment.

Trejos et al. [17] used "sensorized" instruments to perform tasks like palpation, cutting, tissue-handling, suturing and knot-tying in a box trainer. The instruments were able to track force, torque and position data for the task performed. Various parameters were explored using the data that was collected from the device including speed peaks and speed consistency, acceleration, jerk, minimum and maximum force, velocity and acceleration, force derivatives. Combined metric, wherein scaling was used to optimized strong correlations, were also constructed. An example of a combined metric would be a combination of jerk, difference in MAPR (proportion of time where movement speed was 25% > max speed) value between the two hands, the total volume, the number of peaks in speed, and the integrals and derivatives of the grasping and Cartesian forces. The authors report that force-based metrics and combined metrics show a stronger correlation with experience level than the position based metrics [17].

5.2 Computer vision based parameters

Image analysis has also been used for quantifying surgical skill. *Frischknecht et al.* [18] used image analysis programs to obtain objective variables for the assessment of surgical skills to distinguish experts from novices. Participants for the study performed a running suture to close a five-centimeter incision on a foam pad. Sutures were identified using color contrast algorithms and using the incision as a reference line. Total bite size (*B*), stitch length (*L),* number of stitches, travel, symmetry across the incision, total bite-size-to-travel ratio (*B/T*), stitch orientation were some of the variables that were used as assessment parameters. It was found that experts used fewer stiches, had shorter bites and had longer travel than novices. The value of stitch orientations was more for experts than novices. As in this study, image analysis could also be used to assess surgical skills [18].

Islam et al. [19] also examined the possibility of using image processing algorithms for surgical skill assessment. Participants in their study had to perform the peg transfer exercise on the Fundamental of Laparoscopic Surgery (FLS) trainer using purple gloves as they stand out from the background. Glove detection was performed by programs using open source computer vision (OpenCV) libraries. Movement of the data is extracted from the video using motion segmentation to show the change in images over time. For every dataset, the pixel values in each frame are normalized with respect to the idle frame. The frame having the lowest pixel value is considered to be ideal frame. Arithmetic mean and standard deviation is calculated for the all the data. A clear distinction between expert, intermediate and novice was reported in their analysis [19].

6. Mathematical and computer models for needle tissue interaction

There has substantial work done in modelling the needle-tissue interaction forces using Finite Element-based Models (FEM). 3D models generated using FEM have an automated needle path and are difficult to solve in real time [20]. If 2D models are used instead of 3D models computational efficiency is increased at the cost of model accuracy. In Finite Element modeling, the properties of materials play a vital role in calculations and it is known that the properties vary considerably for different tissue types [20]. Some studies using this approach are listed below.

Frick et al. [21] conducted a study to measure the resistance forces acting on suture needles. Different tensile loads were applied to skin and tendon respectively and a straight cutting needle was used to penetrate the tissue material at different velocities. Load vs displacement graphs were calculated for both materials. The tension was proportional to the penetration force required for the needle, while needle displacement rate did not affect the resistance to needle penetration. This study offers a simple model to measure force-feedback during needle penetration [21].

Misra et al. [22] conducted a study wherein a model for needle tissue interaction was proposed and the loads on bevel tip of the needle during needle-tissue interaction were assessed. The model used for calculation is based on the properties of the tissue (gel) and the geometry of the bevel tip. Results indicated that the needle bends more when the bevel angle is small due to larger transverse tip forces [22].

Jackson et al. [20] studied the needle-tissue interaction forces experienced during suturing for a rigid suture needle. The forces were modeled as lumped parameters. In their case, the parameters were friction, tissue compression and cutting forces. The cutting force acts on the needle tip, the frictional force acts along the axis and the tissue compression force depends on the area swept by the needle. To validate the model a needle is mounted eccentrically on motor disc equipped with a 6D force/torque sensor so that it doesn't follow the natural curve of the needle. The needle is driven through artificial tissue model for acquire the needle-tissue interaction forces. The experimental setup validates the lumped parameter model [20].

These studies provide a foundation and frame of reference for research in the area of suturing skill assessment and training. Further, these models can also be used in developing virtual reality simulators.

CHAPTER THREE

CONSTRUCTION OF SUTURING PLATFORM AND ANALYSIS OF PILOT DATA

With a goal of building a suturing device for skill assessment and training the construction of the suture platform was taken in hand [23]. The prototypes of the platform has gone through several iterations starting from a single stamped square suture patch held by clips to a more sturdy and robust device with sensors for gathering information. Examining the literature that was present related to suturing, we realized that force could be one of the parameters that could be used for training novices [9], [11], [12], [15], [17], [24]. We are also aware that surgeons need to apply optimum forces so that they do not damage the tissue. For the wound to heal, the dermal components must meet and heal together. If the edges are inverted, the wound will not heal as quickly or as well as you would like (Figure 1). Precise placement of the needle and orthogonal needle insertion is necessary to obtain ideal wound edge approximation. Excessive forces in lateral direction contribute to tearing of the tissue making it difficult to control bleeding. Unnecessary punctures are always undesirable and will result in excessive tissue trauma. Considering these factors the force sensor was added to the platform. After the platform was constructed the data was obtained from experts and novices to check whether it would help in skill assessment.

1. Evolution of the platform

The first prototype of the suture platform was introduced as a cylinder with a diameter of 6.5 inches, wall thickness of 0.277 inches and height of 4.9 inches. It had 4 holes with a diameter of 0.213 inches were drilled at a height of 2.5 inches from the bottom of the cylinder. From those 4 holes there were alligator clips attached to an elastic chord that held the fabric that was to be sutured (Figure 2).

The trainees would have to suture on a pattern similar to a clock face but having two concentric circles. The distance between the two concentric circles was nearly the same distance as a person would have to cover in case he was following a curvature of the needle, trying to suture on a human tissue. Participants had to also make sure that there was no significant movement in the position of the cylinder. The trainees required an experienced surgeon to give their opinions about their performance and objective scaled feedback to measure performance or accuracy was unavailable. The objective of our group was to develop a method to gather information using the conceptual prototype that could help novice medical students in training and improving their suturing with a simple and cost efficient suturing platform.

The second prototype had the same cylinder dimensions and four hooks were attached to prevent the cylinder from moving when upward force was applied (pulling the needle), additionally the attachments on the base restricts the movement of the cylinder. The base was attached to ATI-mini40 sensor that measures forces and torques in all three axes (Figure 3). The idea behind using a force/torque sensor was that usually a medical suture requires a right amount of force to be applied so that the tissue, excessive force could lead to tissue rupture. Figuring the optimal force needed to be applied along all axes can assist in training novices force control required whilst suturing.

Constructing the third prototype improved the design of the platform, making it more stable and less bulky. Cylinder dimensions were retained from the first prototype but the amount of holes were doubled (from 4 to 8) to attach the cylinder to the base, boosting device stability. These holes were present on the cylinder at a distance of 4.5 inches from the top .The depth at which the disc holding the suture patch could be held was increased to 3 levels. First level is 1 inch from the top, second one is 2 inches from the top, and third one is 4 inches from the top. The decreasing height of each level from the base reduced the maneuverable area for the novice increasing the difficulty of the device. A circular base was cut and mounted on the sensor with central axis of the cylinder and the base being the same. The cylinder was screwed in place using the grooved components glued to the base. The screw sizes for both the base and the suturing platform are 0.157 inches (4mm) (Figure 4). On the first and second prototype of the platform, only a piece of fabric with the suturing pattern (suturing patch) was clamped in place by the alligator clips (Figures 2 and 3).

In the third prototype it was vital that the suture patch be secured in place rather than holding it using an elastic string, since we were unsure of its effect on the force readings. The suture disc was mounted on the cylinder and consisted of 2 parts. The 1st part was where the disc would be attached to the cylinder, this consists of 2 circular hoops with 8 - 0.194 (5mm) holes to screw the material fabric in place (Part 2) and 8 small rectangles, with holes in the middle of each, joining both hoops and attaching the platform to the cylinder. These two hoops had a thickness of 0.118 inches (3mm each), 5.5 inches outer diameter, 3.95 inches inner diameter and the distance between the hoops is 0.6 inches (See Figure 5).

The second part consists of another 2 hoops with the same material thickness as the first two and the same outer diameter size. The only differences are found in the inner diameters, one of the hoops has the same diameter as the previous hoops but an extra hoop was glued on top of it. This hoop is 0.1 inches thick and 0.19 inches tall (Figure 6).

The fabric would be placed on top of this hoop and then secured in place by the other hoop with an inner diameter of 4.14 inches. The fabric would be secured in place by placing 8 bolts in place (approximately 1.5 inches long with a diameter of 0.187 inches) and securing them with 8 nuts. Then this part would be attached to Part 1 and then the suture disc would be secured to the cylinder (Figures 7 and 8). Finally the third prototype would be mounted on the ATI Mini40 Force sensor to acquire data (Figure 9).

2. Placement of the force/torque sensor

The placement of the force sensor is at the center of the base and the axis passing through the center of the sensor and the base is the same. This will ensure that the forces on the suture patch are equally distributed. (Figure 9)

3. Reduction in noise

While testing the platform a lot of unwanted noise was observed from the sensor, which was thought to be because of the vibrations of the table. Thus to reduce noise half cut tennis balls were used to dampen the vibrations, but it was difficult to balance the platform and it did not yield adequate results. Consequently a memory foam mat was placed below the platform resulting in a significant reduction in noise.

4. Placement of the camera

The sensor used in building the suture platform is very sensitive and susceptible to noise. When the data was being collected we envisioned a possibility of trainees touching the cylinder whilst the platform was in use and inducing unwanted forces on the sensor. Therefore to keep a track of various motions that were performed and to identify the unwanted noises during analysis, a camera was mounted on the wall with the top view of the suture platform in focus.

5. DAQ System

The force/torque sensor (transducer) is a compact, rugged, monolithic structure that converts force and torque into analog strain gage signals. The ATI mini40 has electrically shielded integrated cable for transmission protection and to connect to an Interface Power Supply (IFPS) box [25].

The IFPS box was used with the ATI-mini40 sensor to supply power to the transducer and electronics, as well as condition the sensor signals to be used with a data acquisition system. The IFPS is equipped with a 12-pin female connector for the transducer cable connection and a 26-pin male connector for interfacing with the DAQ Device. The IFPS box can be powered using the wall mounted power supply (12V) or through the power supply cable (5V) from the Data acquisition device (DAQ). The IFPS box only requires one source, if both sources are connected, the IFPS box will only use the 12 V source ignoring the 5 V source [25].

The data acquisition system converts the analog voltage signals from the sensor into digital data that is subsequently processed by ATI software or using Simulink blocks converting it to force and torque values. Since the electronic hardware measures the change in resistance, calculations must be performed to obtain the loads being sensed by the F/T sensor. The sensor reports the loads as composite values that need to be converted into the six Cartesian axes [25].

6. Software Algorithm

The data from the sensor is processed and saved in Simulink. For real time acquisition and storage of data a specialized library provided by QUANSER called QUARC was used. QUARC seamlessly integrates with Simulink for rapid controls prototyping and hardware-in-the-loop testing. It is fully integrated with Simulink coder. The steps that are followed during sensor data acquisition are as follows:

The ‘HIL (Hardware-in-loop) initialize’ block was used to associate a name with the DAQ board and was used to connect initialize the various parameter like the number of analog input channels, the minimum and maximum value for analog inputs, the initial value for the outputs, the frequency at which data is obtained, generate PWM (pulse width modulation) outputs etc. [26].

Once the board was selected and the channels are determined, the next step was to read the data from the analog channels. This was done using the ‘HIL Read Analog block’ which immediately reads the specified input channels [26]. The variable ‘qc_get_step_size’ is the fundamental step size of the sensor data acquisition system. This value was used for sampling the data.

In the next step, the data that is obtained is multiplied by the calibration matrix. This converts the values that are obtained from the sensor to six Cartesian axes.

Subsequently the values obtained are then used to bias the system to bring it to zero value. During bias the offset is calculated using the first 10 samples that were collected previously in the system. The offset is then added to the current value. The observed values obtained from the sensor directly i.e. an unbiased system will have non-zero values (exclude noise), but when biased the values are very close to zero.

Thereafter the biased values are plotted and saved. The data is saved using the 'To Host File' block is utilized to stream data to disk on the host machine [26]. The advantage of the block is that you don't need to save variables into MATLAB workspace before saving them to disk. The 'To Host File' block creates a single variable in the file containing the data from the signal attached to its input. Multiple blocks can be employed to stream data to disk on the host machine in either normal simulation or external mode.

7. Hypothesis

Viewing videos pertaining to suturing [27] we feel that most of the forces applied on the tissue are along the z-axis. Therefore, we hypothesized that the force along z-axis will contribute significantly in distinguishing skills between and expert and novices. We also hypothesized that the surgeons will exert considerably less forces in the lateral plane (in x and y directions) and will require significantly less time to complete the task compared to novices.

8. Results

8.1. Experiment protocol

Before conducting the experiment, participants were first shown a presentation explaining the basics of suturing, a brief description of the study and explanation on how to execute the suturing task on the platform.

When the participants were ready a new suture patch was mounted on the platform and were each given a needle holder, forceps and a SH needle (Ethicon, USA) to suture.

- Each participant was given an identification number which was used in the name of the data file generated when the sensor data acquisition system was executed. After starting the video capture, the execution of sensor data acquisition file was started.
- The participant had to perform twelve sutures per experiment in clockwise and counter-clockwise direction.
- Everyone had to avoid touching the cylinder supporting the suture disc while concurrently trying to insert and exit the suture patch on the intersection of the concentric circles and the radial lines on the suture patch. This specific pattern is based on radial suturing is designed to be one of the five primary open vascular skills.

8.2 Participants

Ethics approval was secured for this study before collecting data. In this pilot experiment, three participants were enlisted from a local hospital: a surgeon with extensive surgical experience (of cases > 200) was considered to be an expert, a clinical fellow was considered to be an intermediate, and a medical student considered to be a novice. Each participant performed the experiment at the lowest and the highest level of the platform and both in clockwise and counter clockwise direction, amassing four data sets individually.

8.3. Suture Cycle

Examining the data from the experiments led to the division of the suturing events into subevents (entry, driving, exit and pullout) for better understanding. Executed in the fore mentioned order, these subevents are the elements of a suture cycle (figure 10). Twelve such suture cycles performed at different locations on the suture patch, conforming to the protocol. Let us interpret the phases of the suture cycle with a possibility of gleaning parameters required for skill assessment.

8.3.1. Entry

A standard suture needle (SH 3-0 prolene) was used to puncture the tissue (Figure 11). The sharp end of the needle needs to be perpendicular to the plane of the tissue in addition to minimum application of lateral forces, to achieve least penetration force and reduce tissue trauma.

8.3.2. Driving

The needle passes through the tissue, resulting in (Figure 12) forces that are proportional to the tissue's properties during driving phase. The thin suture patch can be considered to be a two dimensional object that will hardly offer any resistance, winding up with negligible driving force. However, there will be a temporal parameters associated with the event.

8.3.3. Exit

The exit event occurs when the needle emerges out of the suture patch from the downward direction (Figure13). During exit that lateral forces on the tissue should be small and the perpendicular force should be optimum for exit. This can only be accomplished if the curvature of the needle is being followed.

8.3.4. Pullout:

During the needle pullout phase the needle is extracted from the tissue and there is no contact between the needle and tissue. Precaution should be taken to prevent needle tip damage in order to ensure that optimum forces are applied on the tissue during the entire experiment (Figure 14).

8.4. Force and time metrics

Let us now look at the results of the force and time metrics that were used to analyze the data that was obtained from the surgeon.

Location	Average force (N)	Absolute average force (N)	Peak Force (N)	Time per suture cycle (sec)
1	0.0655	0.3889	2.3634	16.4000
2	-0.3900	0.5143	-3.4171	11.5000
3	-0.9953	1.0088	-4.5554	16.9000
4	-0.2029	0.4787	-2.0136	13.8000
5	-1.2900	1.4091	-9.5533	22.0000
6	-0.5779	0.6486	-2.3790	8.2000
7	-0.9470	0.9803	-5.4059	16.0000
8	-0.7041	0.7818	-4.5252	14.8000
9	-0.8573	0.9660	-3.4071	13.7000
10	0.2244	0.4955	-3.4651	18.1000
11	0.0447	0.7055	-2.5800	17.0000
12	-0.5657	0.7832	-7.5150	31.2000

Table 1.1: Novice: Metrics for force in z-direction

Location	Average force (N)	Absolute average force (N)	Peak Force (N)	Time per suture cycle (sec)
1	-0.7447	0.8382	-1.9751	3.5000
2	-0.0879	0.2073	-0.5782	2.6000
3	-0.1531	0.3136	-2.2943	4.7000
4	-0.1660	0.3808	-2.2458	6.1000
5	-0.2372	0.4114	-2.2903	4.0000
6	-0.3137	0.4573	-2.2416	5.1000
7	-0.2164	0.3084	-1.8827	6.5000
8	-0.1042	0.2886	-2.0834	4.2000
9	-0.0484	0.4239	2.1339	5.7000
10	-0.0500	0.3539	-1.4727	4.9000
11	-0.3034	0.4003	-3.5326	12.8000
12	-0.5681	0.6812	-2.3224	3.8000

Table 1.2: Expert: Metrics for force in z-direction

The tables (Table 1.1 and 1.2) and graphs mentioned (in Figure 19 and 20) present results from z-axis (downward) forces, since they were of primary importance to our hypothesis. Observation of the lateral forces indicates difficulty in needle control which might be due to the thinness of the suture patch. Consequently the lateral force data might be unable to distinguish surgical skill. Force and video data were approximately synchronized using video and force data of the first needle insertion, and the two data streams were then correlated based on frame rates and sample rates. The Wilcoxon rank sum tests for z-axis forces between the expert and novice were performed in MATLAB (r. 2015a). The differences observed ($p<=0.05$) between the two groups for peak force, absolute average force and time per suture cycle for each of the twelve suture locations was significant. The total time for the trial for expert was 149.4 seconds while for the novice was 459.4 seconds. The lateral (x- and y-) forces did not exhibit any significant differences.

9. Discussion

The data provides promising preliminary results and substantiates the use of force data to assess suturing skills. These results support similar studies involving the use of force data for quantifying surgical skill where force data has been used to quantify surgical skills. The device built by *Horeman et al.* uses a 3D mouse and converts acceleration data into forces and the range has to be adjusted by changing springs of different stiffness [11]. While in the suturing platform that is developed by us, we are able to obtain force results directly and with higher resolution. The design of the suture patch could also improve the dexterity of the trainee. Trainees find it difficult to complete the task at particular angles on the radial patch and this might be the reason why we witness high forces and number of peaks for some suture cycles. We also observe that the surgeons require less time than the novices for the task and similar results have been seen in various other studies related to suturing [11], [11], [14], [15], [17]. This could be attributed to the fact that the surgeon is confident about his needle placement and suturing skills and may not require much time to guess where exactly the needle has to be places and has a better idea where it will exit. It could also be due to the fact that novices are more careful while suturing. Surgeons are observed to have less peaks as compared to novices possibly due to their motion being smoother as compared to novices and can corresponds to other studies [17]. The collection of video data can also be used for efficient analysis because different people have different ways to perform the same task. This would help in analyzing the variability and could help in devising better methods to improve skills. However the pilot data was collected for a very small sample population and greater numbers of people need to be enlisted for more conclusive results. Other parameters need to be explored for better understanding of the various subevents in the suture cycle so that subevents can be targeted individually for better performance training [14]. Synchronizing force and video data accurately was laborious and tricky; therefore a need to collect data so that it can be easily synchronized was realized.

CHAPTER FOUR

CONSTRUCTION OF SYCHRONIZED PLATFORM AND ANALYSIS OF DATA

Currently the simulator can be divided into two systems, as they run as separate files, namely the data acquisition system which acquires data from the force sensor and the inertial sensor and the video acquisition system which captures a video of the entire experiment for further analysis. The main goal was to assimilate the lessons from the pilot data study and then build a system that could be used for obtaining synchronized data for further analysis. The synchronization of data makes it easier to analyze the data and correlate it to the video containing the motion of the needle on the suture platform. In this chapter we will talk more about the problems that we faced as we went along synchronizing data and how we were able to overcome this problems and succeed in building a system which could be used to collect and analyze synchronized data.

1. Sample rate problems with force/accelerometer sensor

The initial data acquisition system (i.e. Simulink file) for the pilot data was set to capture forces at 10Hz. During analysis of the pilot data we realized that the time taken by the surgeon to complete the suture cycle was really very small (1-2 secs, resulting in small number of data samples obtained for very event which were insufficient to estimate and understand the finer nuances and the changes that were happening during the suture event. The sample rate was then increased to 200Hz before being raised to 500 HZ in order to remove the noise using filters. In this case a high-pass window filter was applied and the results were analyzed. Meanwhile, observations of the experiment video revealed a possibility of using hand motion for the analysis of surgical skill assessment. Therefore, after careful consideration, the InertiaCube sensor (an inertial measurement unit) was used to record the motion of the hand. Using the accompanying software development kit, data was obtained from the InertiaCube sensor and was forwarded to MATLAB, saving all acquired data using the same timestamp. Communication between the SDK and MATLAB takes place via shared memory protocol. For the sake of preventing data losses due to the sensor update rate being 200 Hz and because the execution of the Simulink file and the inertia cube code start at different time instants, the sample rate in the Simulink file logging data from InertiaCube is increased to 1 kHz. Although, the value of the InertiaCube reading is retained and logged unless a new value is received, this however does not affect the calculation of the peak value, average and absolute average value.

2. Sample rate problems with video

Pilot data was collected using a Logitech webcam at 30 fps on the same computer using the Logitech software. The quality of the captured video files was not apt for analysis of the needle movement; consequently the point grey camera was adopted due to its compatibility with the real-time blocks used for image acquisition and high resolution. During further testing, video was recorded in a separate Simulink file using normal execution mode on the same computer with a different sample rate. As the real-time library does not contain blocks to save the video, the normal mode utilizes ‘to multimedia’ block which can be used only in normal mode. This resulted in the execution time for the data acquisition system and the video acquisition system to be different. Hence to minimize this difference; reduction of the resolution, the sample time and the frame rate of 25 fps was implemented. At the same time the ‘Real-time sync’ block was used to achieve synchronization with the data acquisition system by using the same clock used in the external mode and thus indirectly running the video capture in real-time.

3. Video collection problems

The video acquisition system initially contained the image capture blocks from the real time library and were used along with ‘to multimedia’ block to save the video. This created issues (in image quality and the frame rate) with saving the video and hence the default image capture blocks were used. We think that the execution priority for the systems on the computer might be the reason that we obtain different execution times for the data acquisition systems. Also increasing the frame rate (fps) also affects the execution time, as we need to get more images and store them, which in turn requires more execution time.

Simulink has a few encoding options during video logging, the AVI format generates large files while the ‘ffmpeg encoder’ reduced the file size but at cost of drastic drop in resolution.

While collecting the video it was assumed that the ‘host keyboard ‘block would read the key press from the keyboard at the same time in both the acquisition systems. But during execution we realized that there is a delay for the video acquisition system. Similarly, we also tried to stop the systems at the same time using key press from the keyboard and here too we observed delay between the two systems.

The shared memory communication protocol was used to record the video, wherein the data acquisition system would send the value of ‘1’ when the data is being collected and the video acquisition system would save the video file only when it receives this value.

After all this the attempts of collect video on the same system were abandoned, using visualization markers while collecting video data on a different computer was decided. In this case a red led was used to indicate the start of the data collection system. This led is connected to the ’q8’ (USB QUANSER board) and is turned on by the user using the ‘host keyboard’ block. The ‘FlyCapture2’ software utility provided by Point Grey itself was tested. The use of the software resulted in dropped frames which were dependent on the built in encoders and were compensated by reducing the frame rate to 30fps. If the encoders were not implemented, the files that were being generated were too large and therefore were unable to use the camera device up to its full potential and hence other options had to be explored.

Thereafter, we decided to use MATLAB’s ’Image Acquisition’ application which could not only record files at 60 fps but also significantly reduced file size. This application also allows for preview of data being captured.

4.   Shared Memory Protocol

Shared memory is memory that may be simultaneously accessed by multiple programs with in order to provide communication among them or to avoid redundant copies. While using shared memory one process will create an area in the RAM that can be accessed by other processes. QUARC library provides various blocks that use the shared memory protocol for communication, identified using ‘shem’ in the URI (Universal Resource Identifier; URI's are used to identify resources). QUARC uses

URI's for all of its communications because they provide a uniform, extensible and flexible means of identifying the communication medium and protocol to use and associated communication parameters.

The readings from the InertiaCube sensor were obtained using a C program provided in the software development kit (SDK). Using the SDK, data from the sensor could be logged in a text file, but the timestamp for the sensor was obtained from the time the C program starts running. Whereas the timestamp for Simulink file is calculated from the time when the model starts to run. Thus to make it easier to synchronize the data and video collection it was decided to store the InertiaCube data in Simulink using shared memory protocol. The data from the sensor is obtained via USB in the C program and a connection between the C program and Simulink is established. After the connection is established the data is transmitted to the model which is running as long has the communication blocks in the Simulink file have the same URI as the C program. The data is written to the shared memory as array. The Simulink block reads the data from the block and indicates whether a new value is received.

5. MATLAB GUI for data analysis

The data that was obtained during pilot experiments consisted of the video and the force sensor readings whose data acquisition files start at different instants of time. The best way to make sense of the force readings was to view the video along with the force data which called for a simple user interface where in the user could view the force and video reading simultaneously making it easier to analyze the data that was acquired. The initial iteration of the GUI (graphical user interface) consisted of a window which displayed the video frame by frame, another window to display the force profile for the entire experiment and yet another window was added to display a zoomed in graph of the area under observation due to the time scale. Buttons to play, stop and load the video were added along with buttons to plot the forces in individual direction and together. Another window to display video of the graph with respect to time was added to the GUI. The videos would be played individually till the appeared to be in sync, after which they were run together. The GUI also displayed the execution status of the video, the frame number and the time for which the video had been running. The start and end points for a particular region could be entered and the area under the graph for the give region could be colored so as to distinguish the various events in a suture cycle. At the same time we could also calculate the maximum and minimum value of the graph in the given region.

The second iteration was done while focusing on the force and video synchronization. During data acquisition for synchronization, the video and force data acquisition files were started one after the other using a MATLAB script. For second iteration of the GUI focused on just one window and graph as plotting two graphs made the individual graphs smaller on the y-axis. Also the second window that was being used to display the graph video was removed, as it seemed unnecessary. Instead a slider was used to focus on a given area and the zooming distance in terms of time was input by the user. The slider was also used to move through the force data graph for the entire experiment and display the corresponding image frames from the video assuming synchronization was achieved. The delays for starting the video and force files were manually inserted by the user and so was the frame rate for the video. Icons were added to zoom in/out, get value at a point and pan the graph.

The final iteration of the GUI (Figure 21) was implemented when the InertiaCube sensor was integrated and the video acquisition system was being synchronized with the data acquisition system. The slider was removed from the system instead a pointer would run from the start point to the endpoint on the graph on which forces along each axis and resultant can be plotted as per the start and end points input by the user. The user can input the initial frame and the initial time the data starts recording which are in turn used as starting points for the synchronized system. The GUI display's either the original data or the filtered data for the experiment. There is list of options that you can choose for analysis e.g. maximum and minimum values between two points or the peaks that are present in the graph. The minimum, maximum and the average value for a given area of interest (time-based) are calculated using the `min`, `max` and `mean` function in MATLAB, respectively. To select an area of interest from a given matrix in MATLAB we need to obtain the indices of the start and end point of the region of interest. Indices are found using the `find` function on the corresponding time matrix associated with the data to be analyzed. Likewise to calculate the peaks in the graph we use the `findpeaks` MATLAB function. The find peaks function was used to identify the local maxima and works only for positive value. To find negative peaks the negative data is made positive using the absolute function following which `findpeaks` function is applied. The find peaks function given the indices with respect to the area of interest and to convert them to the global region these indices are added to the index of the lower coordinate input.

Analysis option for thresholding the images frames was being developed. Like the previous GUI there are options to zoom, get value at a point and pan the window.

6. Results:

6.1. Experiment protocol

Before the experiment the participants were first shown the basics of suturing wherein explained how to hold the needle holder and how to place the needle in the needle holder. After which they were given hints and demonstrations of how suturing is performed and allowed to practice till they got an idea of how it is done. Thereafter a brief description of the study and explanation of the task was given.

The participants were provided with a needle holder and a SH needle to suture. Each participant was given a number for identification and this was used in the name of the data file that was generated when the sensor data acquisition system was executed. After starting the video capture, the execution of sensor data acquisition file was started.

The participants had to perform twelve sutures per experiment in the clockwise direction within 10 minutes At the end of the experiment the suture patch is prodded once to indicate the end of suture.

The participants were instructed not to touch the cylinder supporting the suture disc while suturing and a try to insert the needle on the intersection of the concentric circles and the radial lines on the suture patch.

6.2. Participants

Ethics approval was obtained and five participants with no prior experience in suturing were selected for the task. The aim here was to determine whether the system was able to synchronize the force/accelerometer data with the video that was being recorded for the experiment.

6.3. Synchronization Results

The data logging for the entire system is terminated after approximately 10 minutes. The time of execution of the sensor data acquisition system is determined obtaining the difference between the time when the sensor data acquisition system starts and the time when its stops execution. The start time for the sensor data acquisition system is obtained by extracting the time stamp for the first non-zero value (start time) that is observed in the force data that is being collected, whereas the last reading taken by the system is considered to be the time when the system stops executing (end time). Similarly for the video acquisition system, we can find the time when the sensor data started recording by obtaining the frame number when the LED turns on (first frame number). The end time can be found when the LED is no longer blinking (last frame number).

The program for the GUI was written so that the start times for both the video and the sensor data are input to synchronize the starting point of the data. After which when the video is played it increments and 60 fps while the data increments at 1 KHz. While running the GUI interface for synchronized execution for the participants, it was observed that the time of execution for the video acquisition system was showing a small difference of approximately 0.25 secs more than the sensor data acquisition system for 10 minutes of data capture (Table 2).

Participant number	Start frame	End frame	Total time Video recording	Start time	End time	Total Time Sensor data	Total time delay	% error
1	229	36735	608.433	3.342	611.528	608.186	0.247	0.041
2	205	37010	613.417	3.553	616.736	613.183	0.235	0.038
3	170	36509	605.65	1.271	606.672	605.401	0.249	0.041
4	167	38463	638.267	2.263	640.271	638.008	0.259	0.041
5	387	28707	472	2.844	474.682	471.838	0.162	0.034

Table 2: The data required for synchronized system and the error between the sensor and video acquisition systems

Hence to compensate for this difference a small increment was added every time sample for the video code to increment the frames accordingly. The formula used to calculate the approximate value for this increment is given by,

Time Increment for every sample= (a-b)/(s*b) (i)

Where,

a=difference between the first frame number/fps and the last frame number/fps

b=difference in timestamp for the start time and end time

s=sampling rate of the sensor data acquisition system

Let us consider participant number 5 and use the formula (i) and the data from table (table 2) to calculate the time increment for the given data set.

$$\text{Time increment} = \frac{[(28707/60-387/60)-(474.682-2.844))]}{[1000*(474.682-2.844)]}$$

$$= \frac{472-471.78}{1000*471.78}$$

$$= 3.4*10^{-7}$$

After adding the time increment for every sample (i.e. per millisecond) the results that are observed in the GUI are promising. We observe that the pointer on the data plot move along with the video data in synchronization and the forces applied on the tissue can be associated with the hand movements in the video (Figures 22-25).

We are aware that the closer the needle is to the perpendicular position, lesser is the force required to penetrate the tissue. This can be interpreted by observing the force peak value in downward direction after needle entry in the force profile (Figure 36).

The data obtained from the platform could provide novices a lot of information on how the progress through every stage of the suturing cycle. The force data could help isolate angles on the suture patch that are difficult to suture, and thus help users to control the amount the force that is applied on the tissue (Figures 30-32). The forces applied by the trainee for every sub-event of a suture cycle can be compared to surgeons for learning (Figure 36).

The feedback from motion data could help prevent sudden movements and bring more finesse to the suturing task. The peaks in motion data indicate sudden movement; this could be when the tissue breaks during the needle penetration the resistance decreases causing the sudden spike, a similar phenomenon can be observed when the needle is pulled out from the tissue (Figures 33-35 and 37). This can be observed looking at the graphs of the force and acceleration in z-direction along with videos, it is seen that the local peak for the force and acceleration correspond after entry and pullout (Figures 36 and 37).

The following steps are employed to calculate the average, absolute average and peak forces/accelerations and the time for each suture cycle.

1. The video is first studied to obtain the approximate frame number of the entry, exit, pick up and pullout points of the needle for each suture cycle. Trainees execute 12 suture cycles for each experiment and the fore mentioned frame numbers are recorded for each suture cycle.
2. Similarly, the frame number when the data starts recording is also noted. These frame numbers are then converted to time by dividing them by the frame rate of the video acquisition system.
3. The actual time (since the recording started) for each sub-event is then calculated by subtracting the time for the first frame form the recorded time for each sub-event.
4. The value obtained from the above calculation is then added to the start time of force/acceleration sensor data acquisition system to get the corresponding force/acceleration reading.
5. As mentioned earlier (Table 2) the video acquisition system captures data slightly faster and time increments are added to the force data to get the corresponding frame in the video. In this case since frame numbers are used to obtain force values the time increment for force values is subtracted from the time obtained from the frame number.
6. The time values are then used to calculate the average, absolute average and peak forces/accelerations using the `min`, `max` and `abs` functions available in MATLAB.

The `findpeaks` function was used to plot the local maxima and the average time is obtained by subtracting the time for entry and pullout.

Let's observe the calculated parametric values for forces and linear acceleration for each axis, for a participant (Tables 3.1-3.3 and 4.1-4.3).

avg. force	abs avg. force	peak force	avg. time
0.0872	0.3537	3.5835	15.0500
0.0110	0.3627	2.7663	11.6000
0.1949	0.3110	2.6485	9.7000
0.1199	0.2120	1.6489	25.9170
0.0873	0.2306	1.9611	11.8670
0.0702	0.2435	-2.4646	15.0330
-0.4574	0.5439	-2.7967	14.6500
-0.2685	0.3471	-1.6578	13.6160
-0.1019	0.3921	4.0269	23.4500
-0.0699	0.2321	-0.9697	10.4830
-0.1512	0.2535	1.2390	11.9670
-0.2039	0.3442	-1.7966	20.5000

Table 3.1: Novice: Force in x-direction for synchronized system

avg. force	abs avg. force	peak force	avg. time
0.1100	0.1887	-1.3205	15.0500
0.0251	0.2807	-1.7528	11.6000
-0.0502	0.3865	-2.9085	9.7000
-0.0812	0.2533	-2.0147	25.9170
-0.4861	0.5664	-2.5708	11.8670
-0.1531	0.3014	1.6541	15.0330
0.0037	0.2854	2.2003	14.6500
-0.3337	0.3605	-1.6000	13.6160
0.1739	0.4734	3.8240	23.4500
0.2939	0.4381	3.2142	10.4830
0.1151	0.3783	2.2526	11.9670
-0.0126	0.2558	-2.2788	20.5000

Table 3.2: Novice: Force in y-direction for synchronized system

avg. force	abs avg. force	peak force	avg. time
-0.1471	0.2274	-2.5029	15.0500
-0.1163	0.2372	-1.6684	11.6000
-0.0664	0.1734	-2.4276	9.7000
-0.1421	0.2090	-1.6083	25.9170
-0.0767	0.1965	-2.6617	11.8670
-0.2126	0.2700	-2.4832	15.0330
-0.0261	0.2480	-2.0078	14.6500
0.0232	0.2498	-2.4646	13.6160
-0.5157	0.6155	-8.6340	23.4500
-0.1330	0.2764	-2.6553	10.4830
-0.0826	0.2080	-1.7143	11.9670
-0.3971	0.4232	-4.9656	20.5000

Table 3.3: Novice: Force in z-direction for synchronized system

avg. acceleration	abs avg. acceleration	peak acceleration	avg. time
-0.0149	0.3155	7.4538	15.0500
-0.0636	0.3334	-3.9192	11.6000
0.0111	0.4095	-4.2823	9.7000
0.0268	0.4099	5.4043	25.9170
0.0242	0.5632	-8.1555	11.8670
-0.1376	0.4865	-6.7301	15.0330
0.0860	0.4727	5.2293	14.6500
-0.0174	0.5064	7.9396	13.6160
-0.0336	0.3983	-23.6410	23.4500
0.0272	0.5628	-18.1008	10.4830
-0.0506	0.4522	-5.9238	11.9670
-0.0837	0.3808	7.8347	20.5000

Table 4.1: Novice: Acceleration in x-direction for synchronized system

avg. acceleration	abs avg. acceleration	peak acceleration	avg. time
0.0418	0.3844	4.5122	15.0500
0.0326	0.3772	10.3482	11.6000
-0.0136	0.3998	-6.3635	9.7000
-0.0294	0.2621	-2.2265	25.9170
0.0247	0.2921	2.6362	11.8670
0.0001	0.4590	9.0665	15.0330
0.1381	0.5086	3.7068	14.6500
0.0903	0.3637	-2.5731	13.6160
0.0560	0.2730	21.8144	23.4500
0.1399	0.3354	18.9100	10.4830
0.1468	0.2562	7.7241	11.9670
0.0459	0.2930	-3.7118	20.5000

Table 4.2: Novice: Acceleration in y-direction for synchronized system

avg. acceleration	abs avg. acceleration	peak acceleration	avg. time
-0.0145	0.4352	7.8828	15.0500
-0.0126	0.4844	6.7705	11.6000
0.0111	0.5175	-5.5241	9.7000
0.0021	0.4367	6.9583	25.9170
-0.0258	0.6276	-8.3375	11.8670
-0.0961	0.5620	-9.2952	15.0330
0.1376	0.4517	-4.5534	14.6500
0.1323	0.3969	-6.4174	13.6160
0.1067	0.5090	-13.8312	23.4500
0.1163	0.5072	-26.6289	10.4830
0.1092	0.5153	-5.4486	11.9670
-0.0413	0.3267	-7.9836	20.5000

Table 4.3: Novice: Acceleration in z-direction for synchronized system

7. Discussion

We can see that the video and force data can be displayed and analyzed in synchronization (Figures 21-29). Although it was observed that the time of execution for the video acquisition system was showing a small difference of approximately 0.25 secs more than the sensor data acquisition system for 10 minutes of data capture (Table 3). For the synchronization to work we need to know the start frame and the end frame and the start time and end time for the video acquisition and sensor data acquisition systems respectively. These values can be used to calculate the small time increments in time that need to be added while incrementing the frames during execution. These increments are significant as the time for which the data is collected increases. After the experiment we prodded the tissue before shutting down the data collection (Figure 21). This was done because the prodding of the tissue gives a sharp peak that is easier to identify and can be used to verify synchronization. Also, when we observe the force data we observe that there are vibrations that are introduced after needle is pulled out. The may be tissue vibrations that are generated when the needle is pulled out and the tissue under tension tries to regain its position. The capture of video data helps us visit the data anytime for analysis. Studies also suggest that every individual has a way of performing a certain task and this variability can be better observed using video data along with the sensor data [15]. A lot of the systems that are present are developed for minimally invasive surgery. The system developed by us is based on basic suturing skills and is developed for “open” surgery based skills. Also, most of the systems that are present have not explored the possibility of using both forces and motion in the same system [8]–[10], [14], [15]. Studies indicate that depending on the task and its complexity force based parameters may be the significant parameters while in other cases the motion based parameters may be the significant parameters [11]–[17]. Thus having a system that can record both force and motion based parameters could be advantageous in assessment of suturing skills.

CHAPTER FIVE

FUTURE WORK

Current the system that we have is able collect data and synchronizes it for analysis. But a lot of work needs to be done before it turns in a product.

1. The system right now is coded in MATLAB/Simulink. Therefore for it to be used on any other device will require the MATLAB/Simulink software to be installed, this requires a license. Thus porting the code form MATLAB/Simulink to C++ or any other language that makes it stand-alone software would be a better option.
2. Also the Data Acquisition Card that is being used for this system requires the PCI slots that have been replaced by the PCI express slots, due to which we are unable to use the computational power provided by the new computers.
3. The synchronization that we have right now is for post data capture. It would be more helpful if the device is able to provide real time feedback while capturing data for synchronization. This will help trainees make changes as and when they are performing the task.
4. Development of good metrics for distinguishing surgical skills and that will help train novices will also be essential.
5. Likewise a lot of changes need to be made to the platform so that it is easier to used and quicker to replace parts of platform. Instead of having three levels of height for the suture disc, the disc can be mounted on the base of the platform and cylinders can be stacked one over the other so that the same level of difficulty is obtained. This will also eliminate the use of the long screws that hold the suture patch and disc

together. Another change that can be made is instead of having screws to hold the suture platform other methods of securing the platform should be implemented.

6. During video recording we observe that the hand overlaps the needle making it difficult to analyze the needle movement during that time period. Two cameras can be mounted opposite each other to overcome this problem. Similarly one can mount a small camera that can capture the movement of the needle once it has punctured the suture patch and can be used to further analysis (e.g. the distance between the actual and supposed needle insertion point could determine the accuracy of the needle placement).
7. It is observed that the trainees usually touch the inner edge of the cylinder while performing the task. This induces noise in the sensor readings. Therefore to be aware of this type of contact, a circular ring with its outer surface coated with conducting material could be mounted inside the cylinder at a small distance from the inner surface of the cylinder. The disc could be separated from the cylinder surface with help of small springs. The inner surface that can come in contact with the cylinder is also coated with conducting material. When the trainee touches the cylinder the inner disc will come in contact with the cylinder surface completing an electrical circuit making a LED glow to indicate contact.
8. Various other tasks can be introduced by replacing the suture disk for training other skills.

APPENDICES

Appendix A

Figures

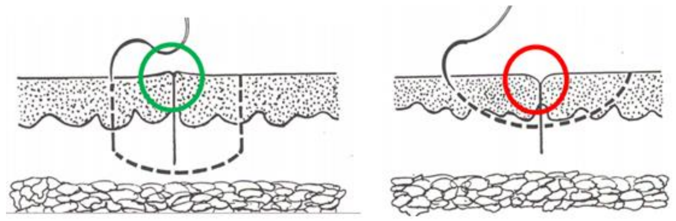


Figure 1: Suturing Example

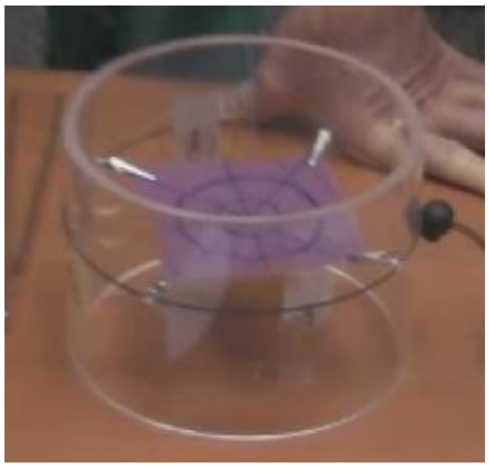


Figure 2: First prototype

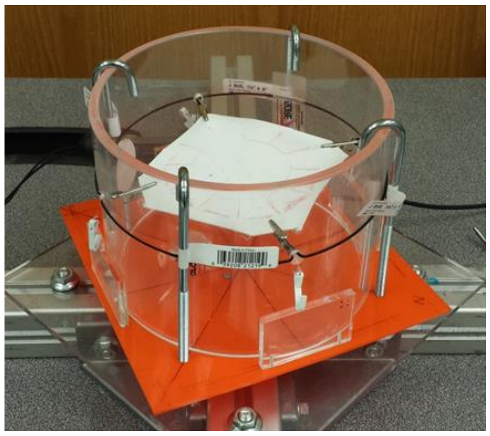


Figure 3: Second Prototype

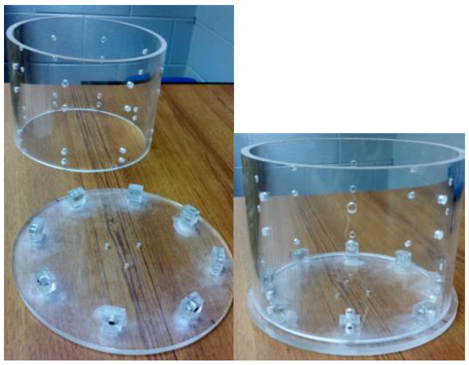


Figure 4: Third Prototype (Cylinder and Base)

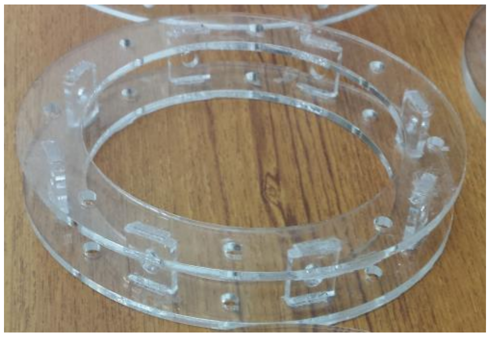


Figure 5: Third Prototype Part 1 of Platform

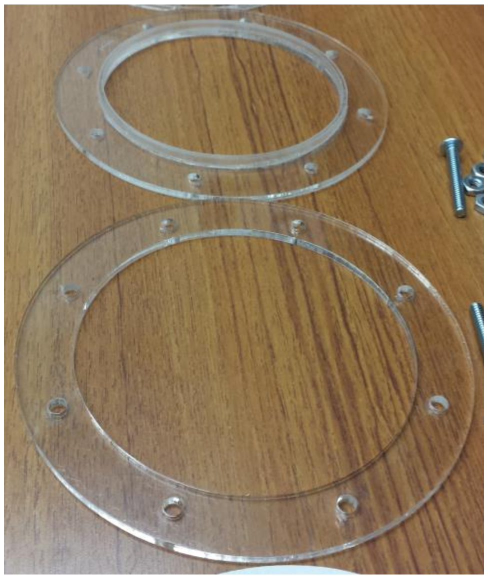


Figure 6: Third Prototype Part 2 of Platform

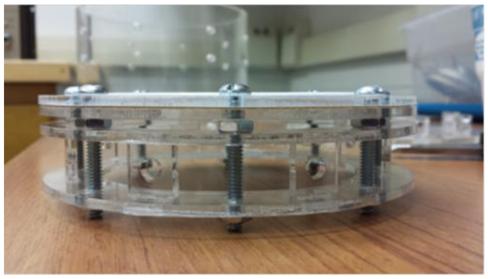


Figure 7: Part 1 and 2 Attached

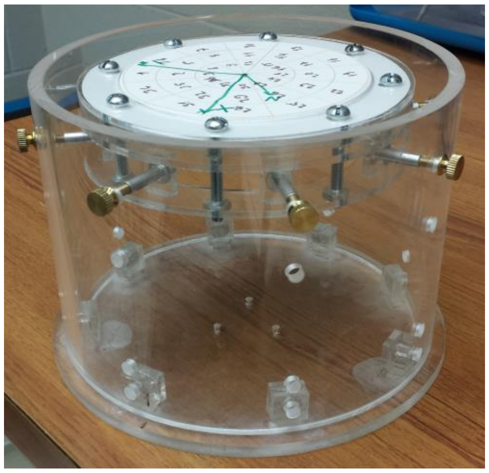


Figure 8: Platform and Cylinder

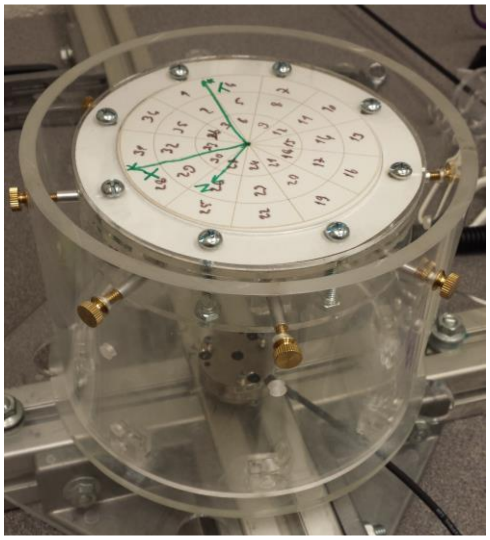


Figure 9: Third Prototype Completed

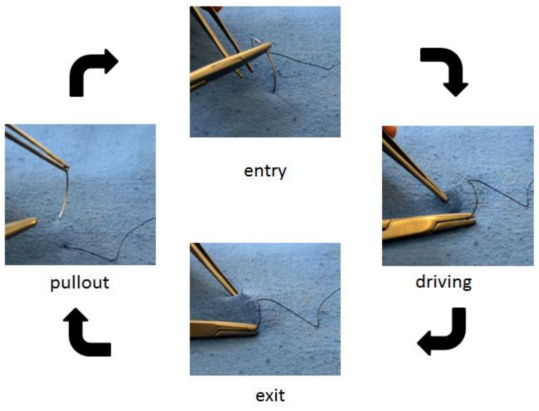


Figure 10 : Suture cycle [27]

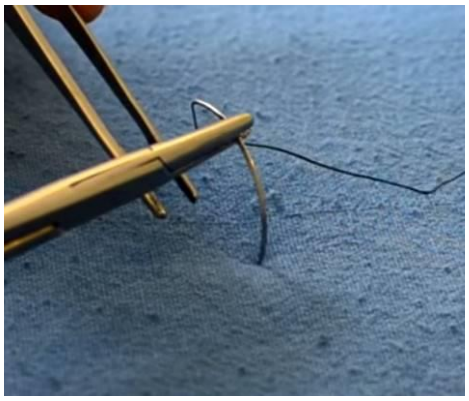


Figure 11: Needle Entry [27]

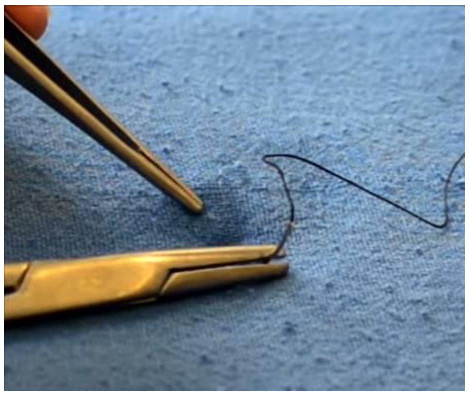


Figure 12: Needle Driving [27]

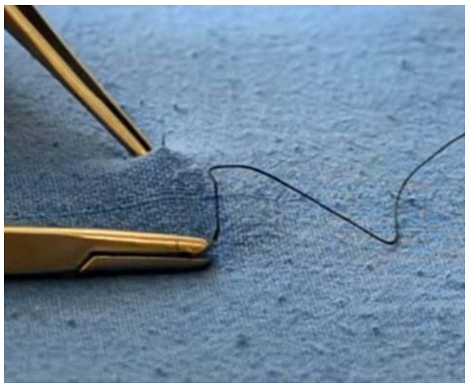


Figure13: Needle Exit [27]

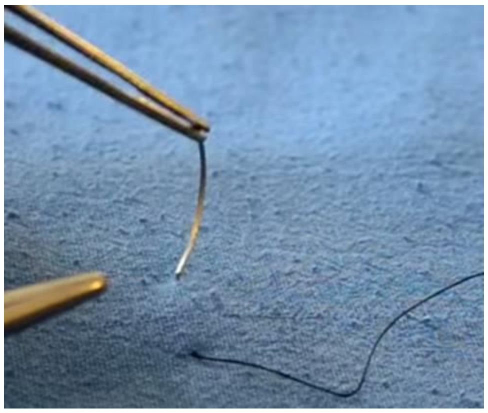


Figure 14: Needle Pullout [27]

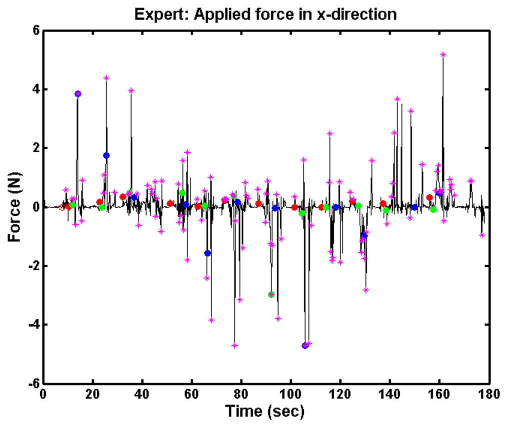


Figure 15: Force in x-direction for expert with events in the 12 suture cycles estimated

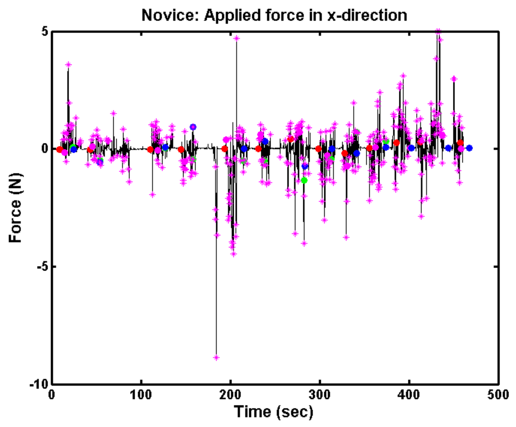


Figure 16: Force in x-direction for novice with events in the 12 suture cycles estimated

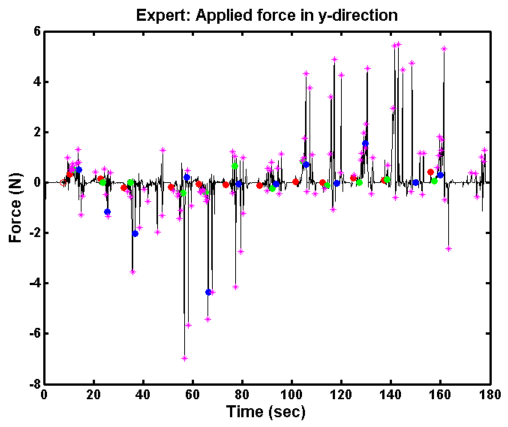


Figure 17: Force in y-direction for expert with events in the 12 suture cycles estimated

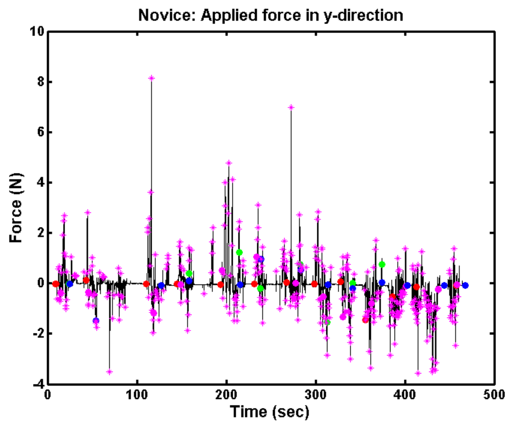


Figure 18: Force in y-direction for novice with events in the 12 suture cycles estimated

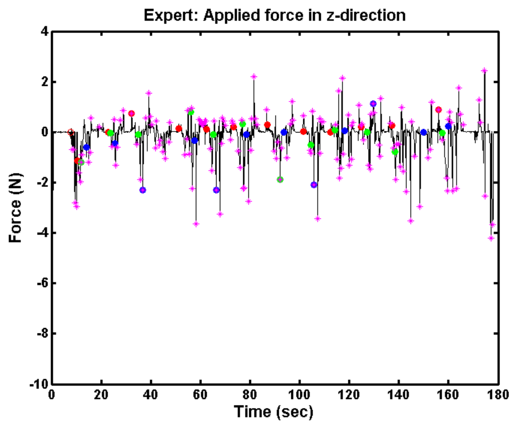


Figure 19: Force in z-direction for expert with events in the 12 suture cycles estimated

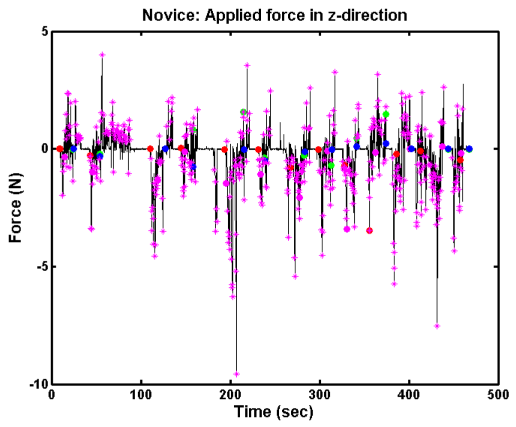
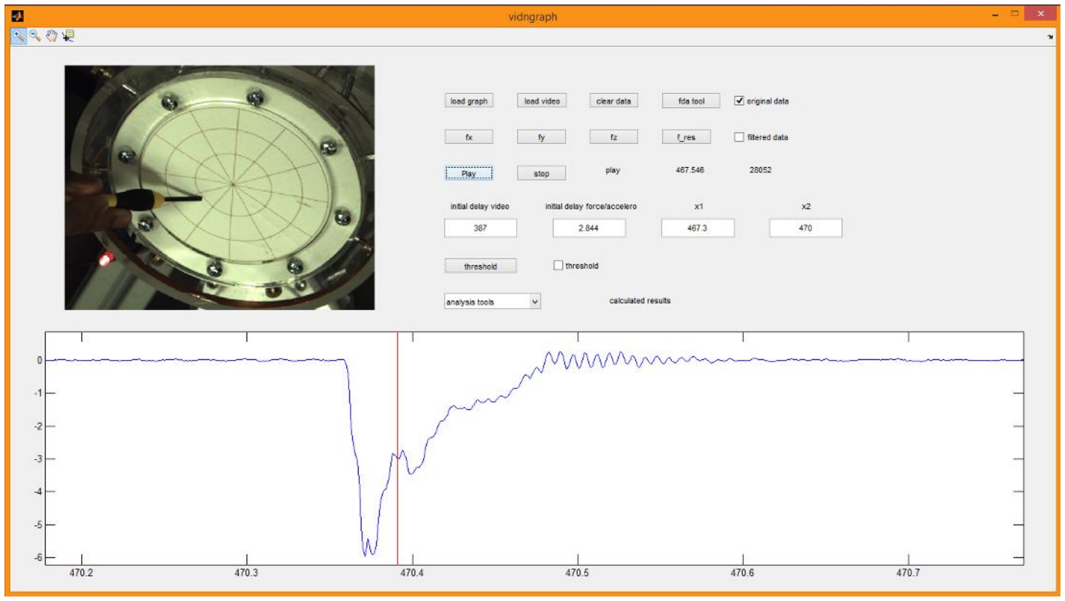


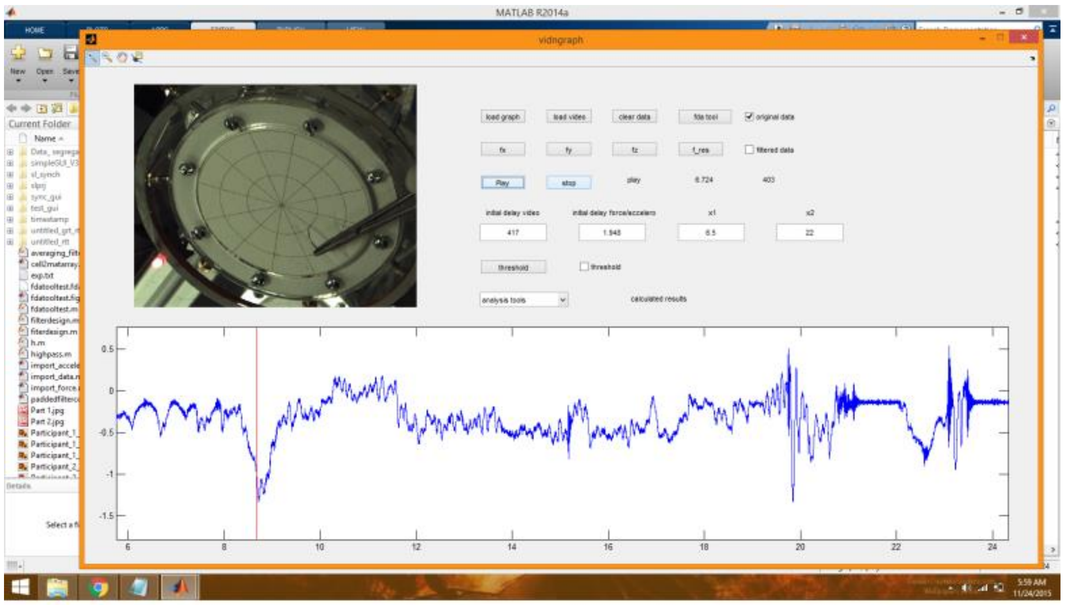
Figure 20: Force in z-direction for novice with events in the 12 suture cycles estimated



☑ original data
☐ filtered data
play 467.546 28852

initial delay video: 387
initial delay force/accelero: 2.844
x1: 467.3
x2: 470
☐ threshold

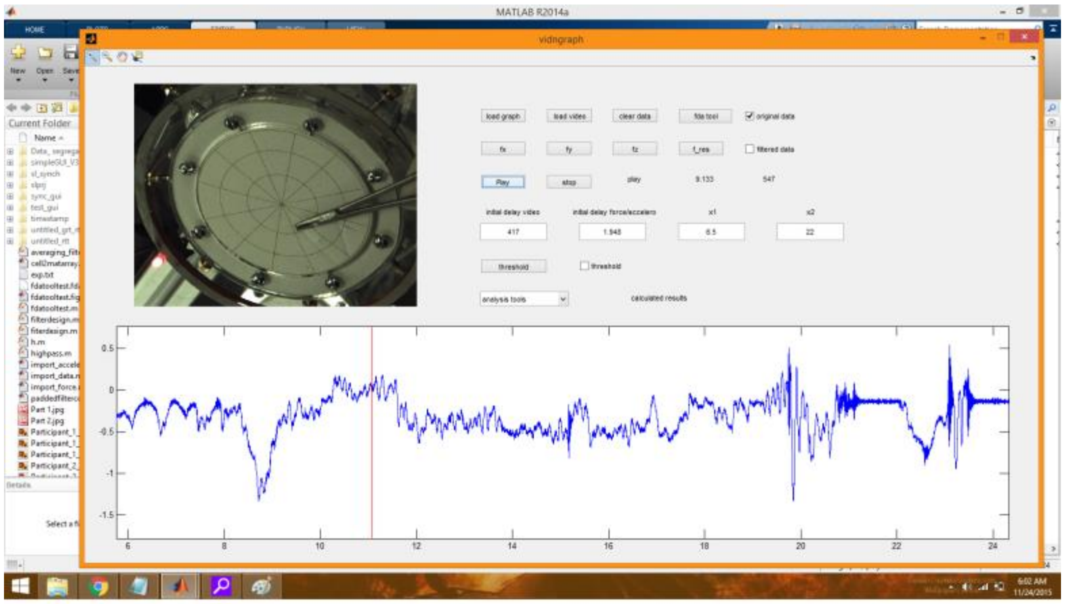
Figure 21: GUI used to synchronize the force/accelerometer data along with the video



☑ original data
☐ filtered data
play 6.724 403

initial delay video: 417
initial delay force/accelero: 1.948
x1: 6.5
x2: 22
☐ threshold

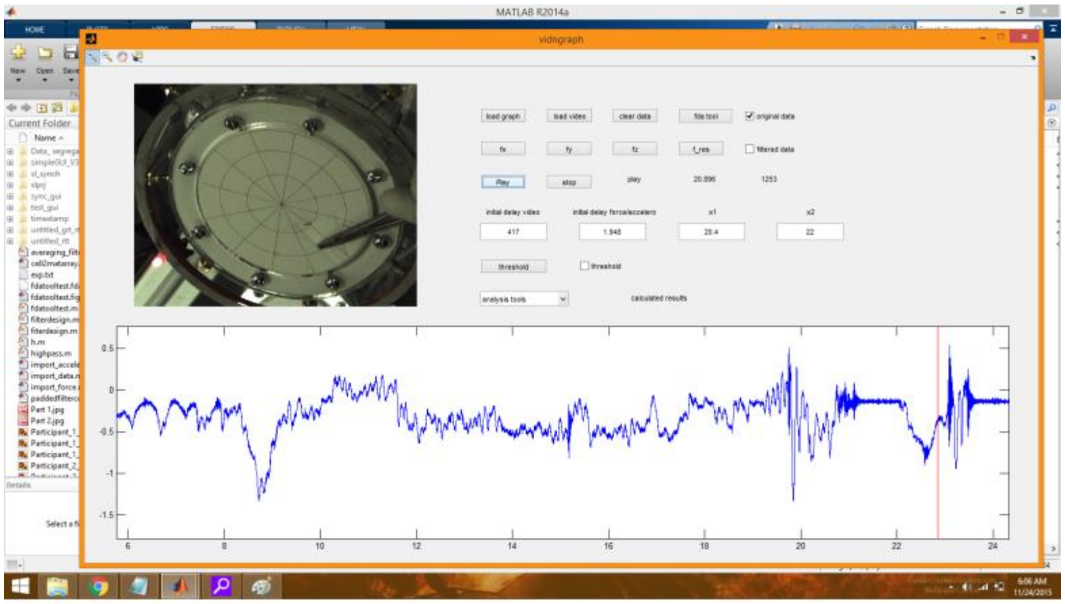
Figure 22: Suture event for synchronized system (force): Entry



play 9.133 547

initial delay video: 417
initial delay force/accelero: 1.948
x1: 6.5
x2: 22

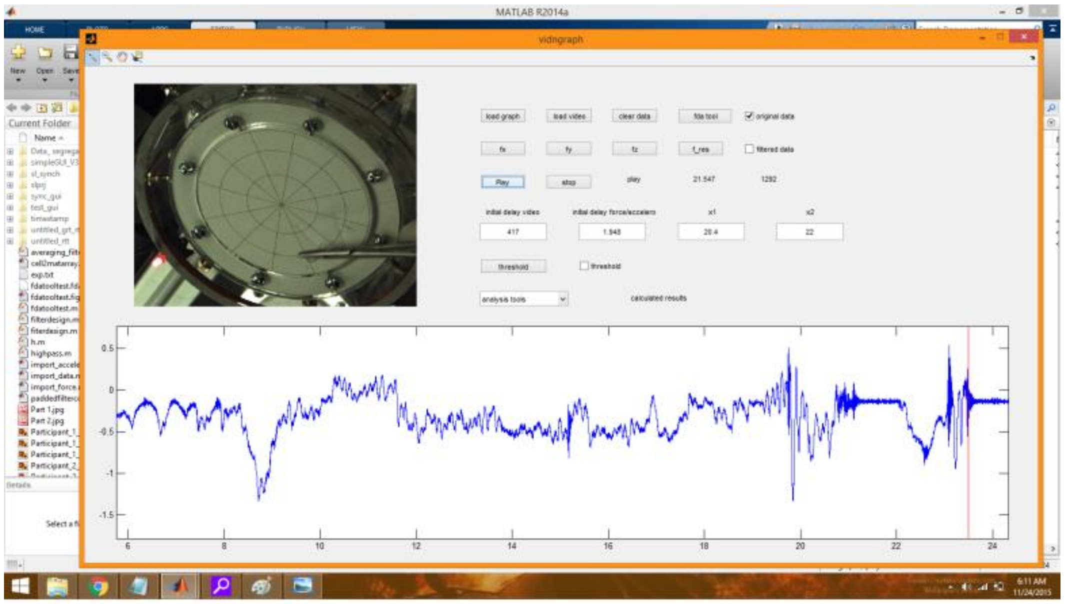
Figure 23: Suture event for synchronized system (force): Exit



play 20.896 1253

initial delay video: 417
initial delay force/accelero: 1.948
x1: 20.4
x2: 22

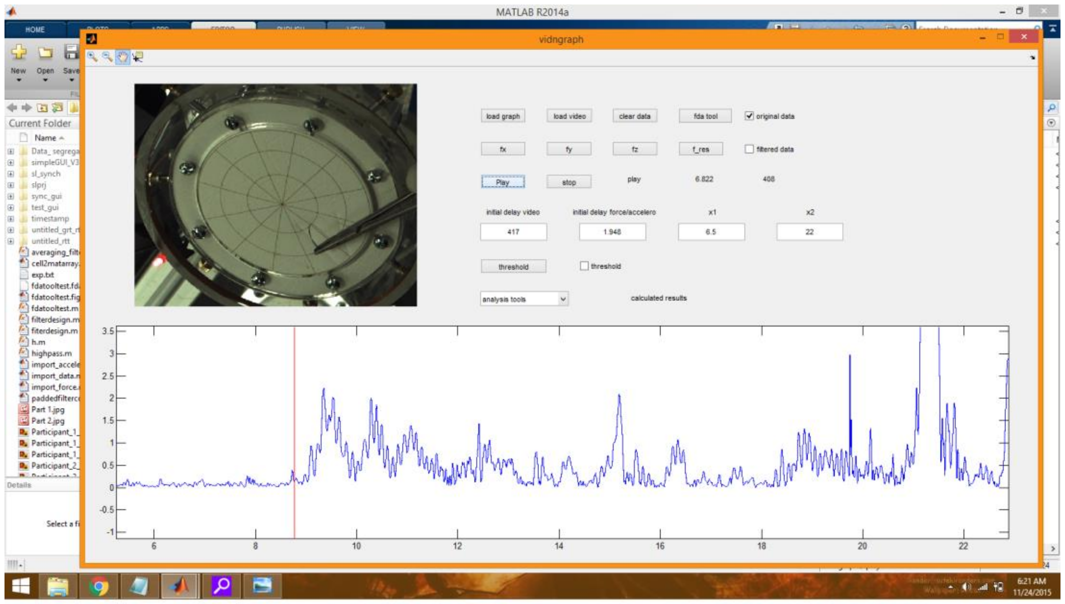
Figure 24: Suture event for synchronized system (force): Pick



21.547 1292

initial delay video 417
initial delay force/accelero 1.948
x1 28.4
x2 22

Figure 25: Suture event for synchronized system (force): Pullout



6.822 408

initial delay video 417
initial delay force/accelero 1.948
x1 6.5
x2 22

Figure 26: Suture event for synchronized system (acceleration): Entry

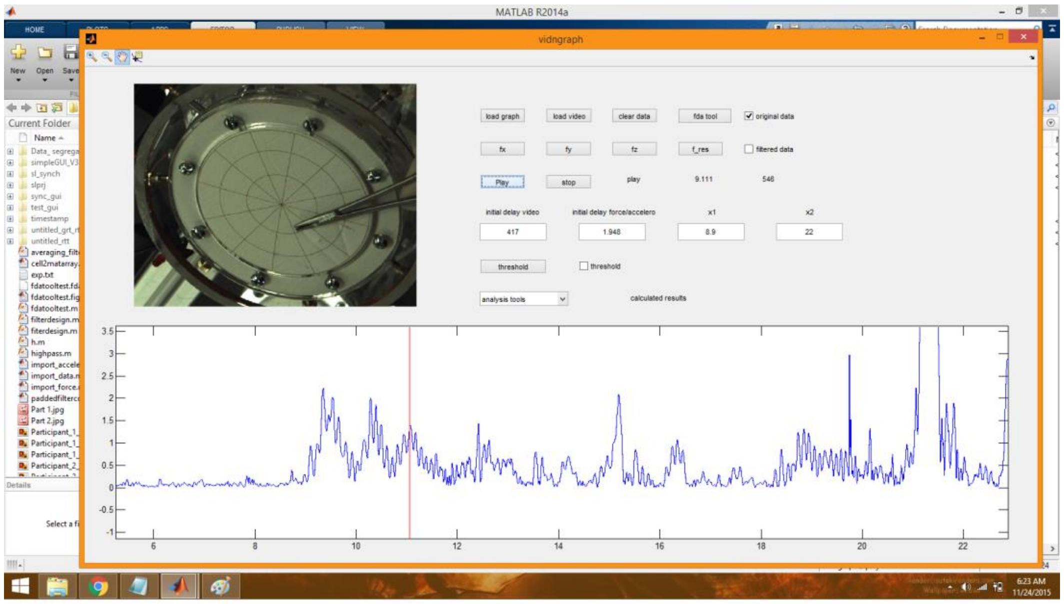


Figure 27: Suture event for synchronized system (acceleration): Exit

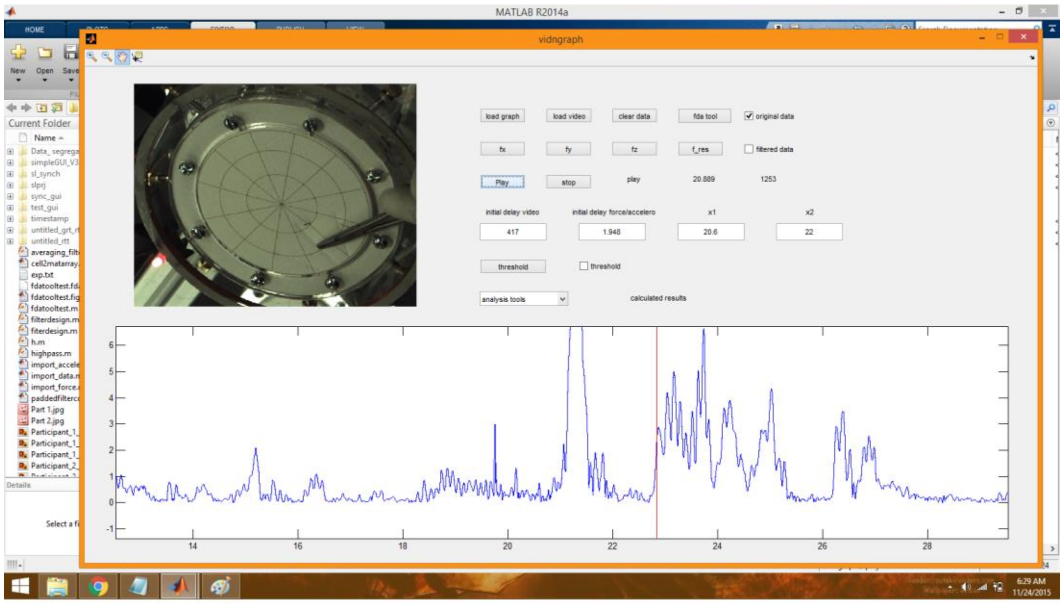
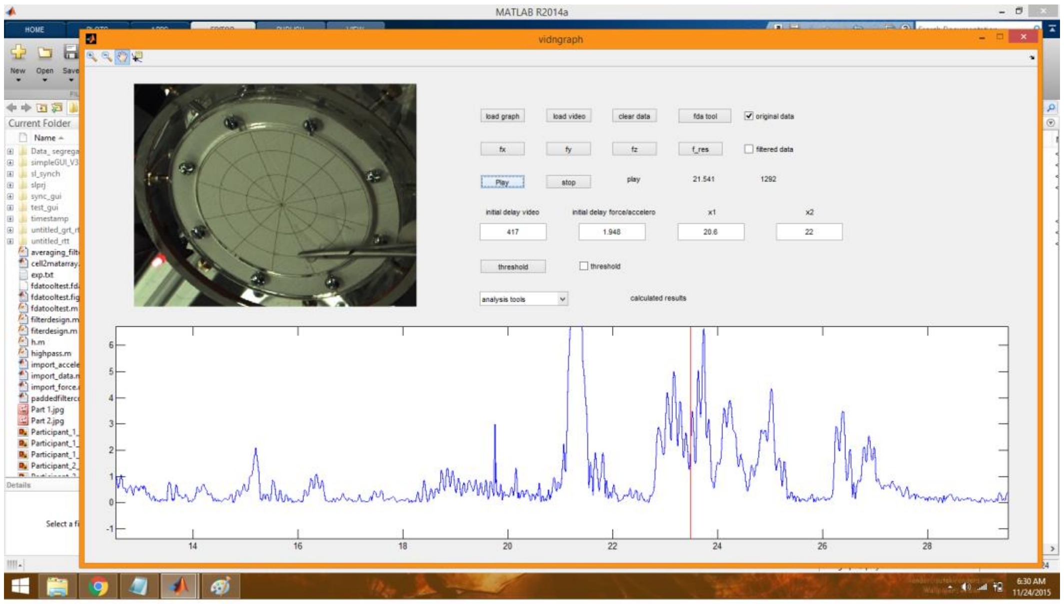


Figure 28: Suture event for synchronized system (acceleration): Pick



play 21.541 1292

initial delay video: 417

initial delay force/accelero: 1.948

x1: 20.6

x2: 22

Figure 29: Suture event for synchronized system (acceleration): Pullout

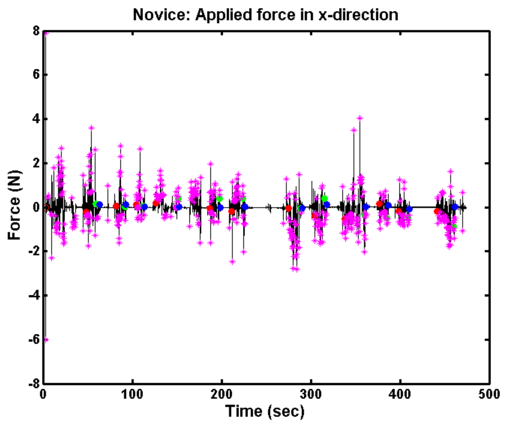


Figure 30: Novice: Force in x-direction using synchronized system

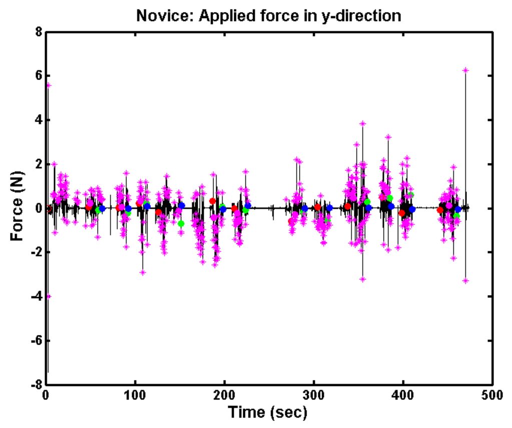


Figure 31: Novice: Force in y-direction using synchronized system

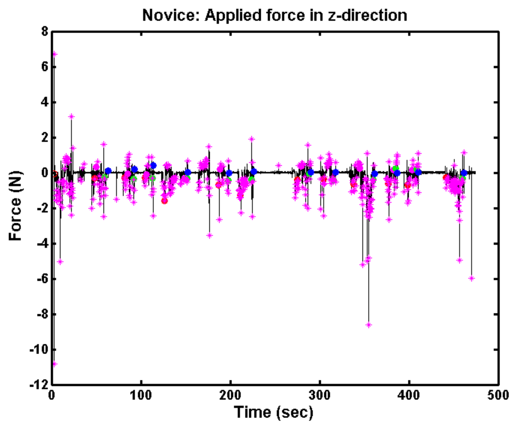


Figure 32: Novice: Force in z-direction using synchronized system

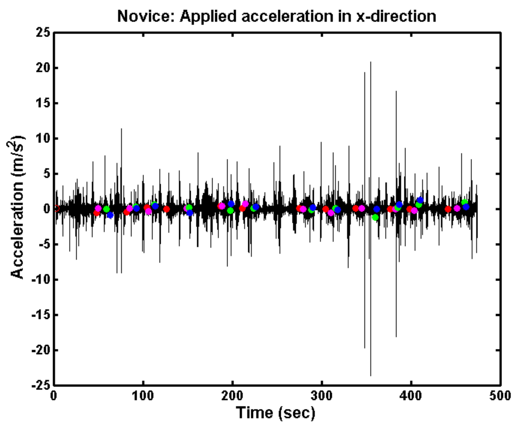


Figure 33: Novice: Acceleration in x-direction using synchronized system

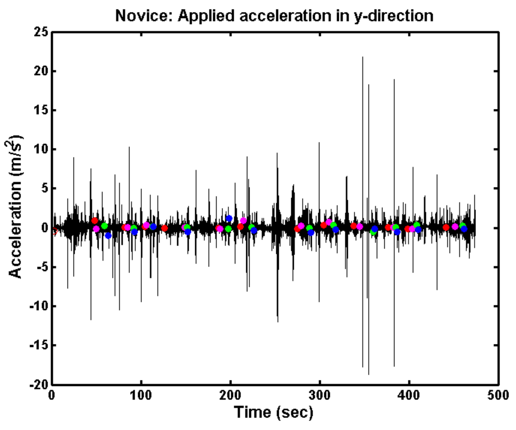


Figure 34: Novice: Acceleration in y-direction using synchronized system

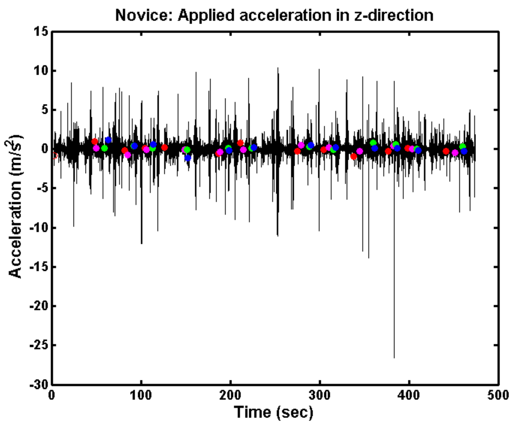


Figure 35: Novice: Acceleration in z-direction using synchronized system

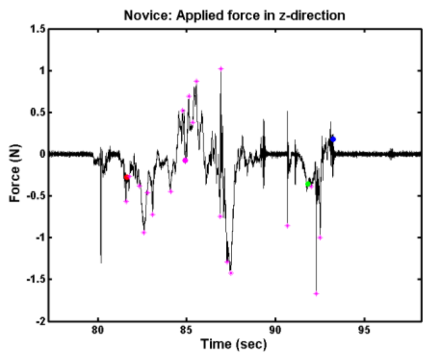


Figure 36: Novice: Force in z-direction for a suture cycle using synchronized system

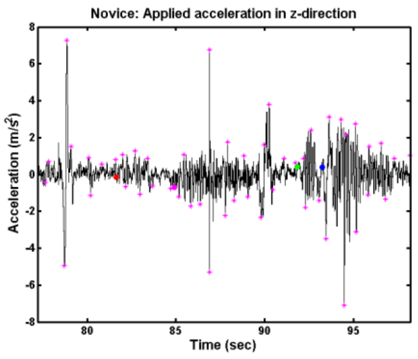


Figure 37: Novice: Acceleration in z-direction for a suture cycle using synchronized system

Appendix B

GUI Code

MATLAB GUI:

```
function varargout = vidngraph(varargin)
% VIDNGRAPH MATLAB code for vidngraph.fig
%      VIDNGRAPH, by itself, creates a new VIDNGRAPH or raises the
existing
%      singleton*.
%
%      H = VIDNGRAPH returns the handle to a new VIDNGRAPH or the
handle to
%      the existing singleton*.
%
%      VIDNGRAPH('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in VIDNGRAPH.M with the given input
arguments.
%
%      VIDNGRAPH('Property','Value',...) creates a new VIDNGRAPH or
raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before vidngraph_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to vidngraph_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help vidngraph

% Last Modified by GUIDE v2.5 23-Oct-2015 17:06:06

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @vidngraph_OpeningFcn, ...
                   'gui_OutputFcn',  @vidngraph_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
```

```
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before vidngraph is made visible.
function vidngraph_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to vidngraph (see VARARGIN)

% Choose default command line output for vidngraph
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes vidngraph wait for user response (see UIRESUME)
% uiwait(handles.figure1);




% --- Outputs from this function are returned to the command line.
function varargout = vidngraph_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;




function vid_delay_Callback(hObject, eventdata, handles)
% hObject    handle to vid_delay (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of vid_delay as text
%        str2double(get(hObject,'String')) returns contents of
vid_delay as a double
```

```
% --- Executes during object creation, after setting all properties.
function vid_delay_CreateFcn(hObject, eventdata, handles)
% hObject    handle to vid_delay (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes on button press in load_graph.
function load_graph_Callback(hObject, eventdata, handles)
% hObject    handle to load_graph (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
run('vidngraph_global_variables.m')
global fx fy fz f_res time ;
tic
%% Open File
      %Prompt user for filename
      [fname, pname] = uigetfile('*.csv');
       %Create fully-formed filename as a string
       filename3 = fullfile(pname, fname);
       %Check that file exists
       assert(exist(filename3,'file')==2, '%s does not exist.',
filename3);
%        %Read in the data, skipping the first row
%        data = csvread(filename3,1,0);

%% Initialize variables.
filename = filename3;
delimiter = ',';

%% Read columns of data as strings:
% For more information, see the TEXTSCAN documentation.
formatSpec = '%s%s%s%s%[^\n\r]';

%% Open the text file.
fileID = fopen(filename,'r');

%% Read columns of data according to format string.
% This call is based on the structure of the file used to generate this
% code. If an error occurs for a different file, try regenerating the
code
% from the Import Tool.
```

```
dataArray = textscan(fileID, formatSpec, 'Delimiter', delimiter,
'ReturnOnError', false);

%% Close the text file.
fclose(fileID);

%% Convert the contents of columns containing numeric strings to
numbers.
% Replace non-numeric strings with NaN.
time=str2double(dataArray{1,1});
fx=str2double(dataArray{1,2});
fy=str2double(dataArray{1,3});
fz=str2double(dataArray{1,4});
toc
time(1)=[];
fx(1)=[];
fy(1)=[];
fz(1)=[];
f_res=sqrt(fx.^2+fy.^2+fz.^2);
%plot([1:1:10],sin([1:1:10]),'r')
toc
guidata(hObject, handles);

% --- Executes on button press in play.
function play_Callback(hObject, eventdata, handles)
% hObject    handle to play (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.status,'String','play');
guidata(hObject,handles);
pause(0.1)
play(hObject,eventdata,handles);

function play(hObject,eventdata,handles)
global time fx fy fz new_time new_y_hpf;

k1=str2double(get(handles.x1,'String'));
[~,k1]=ismember(k1,new_time);
k2=str2double(get(handles.x2,'String'));
[~,k2]=ismember(k2,new_time);
s=get(handles.status,'String');
init_vid_delay=str2double(get(handles.vid_delay,'String'));
init_force_delay=str2double(get(handles.force_delay,'String'));
[~,init_force_delay]=ismember(init_force_delay,new_time);
pause(0.5)
%vid_t=(init_vid_delay)*(1e-5);
hold on
f=init_vid_delay+round((new_time(k1)+(4.1e-7)*(k1))*60);
i1=f;
thres=get(handles.thresh,'Value');
X=zeros(524,664);
```

```
while(1)
    if(k1<=k2 && strcmpi(s,'play'))
        %axes(handles.graph)
        p2=line([new_time(k1+init_force_delay)
new_time(k1+init_force_delay)],get(handles.graph,'YLim'),'Color','r');

%p2=plot(handles.graph,new_time(k1+init_force_delay),new_y_hpf(k1+init_
force_delay),'r*');
        set(handles.time,'String',new_time(k1));
        %set(p2,'visible','on');
        f=init_vid_delay+round((new_time(k1)+(3.5e-7)*(k1))*60);
        if(f>i1 && f>0)
        i1=f;
        set(handles.frame_no,'String',round(new_time(k1)*60));
        img=read(handles.obj,f);
        %axes(handles.vid);
        if(thres==1)
        X=zeros(524,664);
        for i=1:524
            for j=1:664
                if(  img(i,j,1)>210 && img(i,j,2)>200 && 120>img(i,j,3)
&& img(i,j,3)>70)
                      X(i,j)=255;
                end
            end
        end
        imshow(X,'Parent',handles.vid);
        else
            imshow(img,'Parent',handles.vid);
        end
        end
        k1=k1+1;
        pause(0.001);
        delete(p2);
    else
        set(handles.status,'String','stop');
        guidata(hObject,handles);
        break
    end
    guidata(hObject,handles);
    s=get(handles.status,'String');
end
hold off

% --- Executes on button press in stop.
function stop_Callback(hObject, eventdata, handles)
% hObject    handle to stop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.status,'String','stop');
guidata(hObject,handles);
pause(0.1)
```

```
% --- Executes on button press in load_vid.
function load_vid_Callback(hObject, eventdata, handles)
% hObject    handle to load_vid (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.status,'String','stop');
[fname, pname] = uigetfile('*.mp4');
%Create fully-formed filename as a string
filename1 = fullfile(pname, fname);
%Check that file exists
assert(exist(filename1,'file')==2, '%s does not exist.', filename1);
temp_obj1=VideoReader(filename1);
handles.obj= temp_obj1;
handles.nframes=temp_obj1.NumberOfFrames;
guidata(hObject,handles);
img=read(handles.obj,1);
set(handles.frame_no,'string',1);
axes(handles.vid);
imshow(img,[]);
handles.frameno=1;
guidata(hObject,handles);

% --- Executes on button press in x.
function x_Callback(hObject, eventdata, handles)
% hObject    handle to x (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global time fx fy fz h_hpf h_lpf new_time new_y_hpf f_res;

axes(handles.graph);
ylim(handles.graph,'auto')

if  (get(handles.orig,'value'))
plot(time(1:(end-500)),fx(1:(end-500)))
end

if(get(handles.filt,'value'))
ylim(handles.graph,'auto')

fx_m = fx;
ti_x = time;
temp_mean=mean(fx_m);
fxm = fx_m- temp_mean;

y_hpf = filter(h_hpf,1,fxm);

new_time=ti_x;
new_y_hpf=y_hpf;
new_time((length(ti_x)-498):end)=[];
new_y_hpf(1:499)=[];
```

```
hold on
plot(new_time,new_y_hpf,'b');
hold off
ylim([-1 1]);

end
guidata(hObject,handles);

% --- Executes on button press in y.
function y_Callback(hObject, eventdata, handles)
% hObject    handle to x (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global time fx fy fz h_hpf h_lpf new_time new_y_hpf f_res;

axes(handles.graph);
ylim(handles.graph,'auto')

if (get(handles.orig,'value'))
plot(time(1:(end-500)),fy(1:(end-500)))
end

if(get(handles.filt,'value'))
ylim(handles.graph,'auto')

fx_m = fy;
ti_x = time;
temp_mean=mean(fx_m);
fxm = fx_m- temp_mean;

y_hpf = filter(h_hpf,1,fxm);

new_time=ti_x;
new_y_hpf=y_hpf;
new_time((length(ti_x)-498):end)=[];
new_y_hpf(1:499)=[];

hold on
plot(new_time,new_y_hpf,'b');
hold off
ylim([-1 1]);

end


guidata(hObject,handles);

% --- Executes on button press in z.
function z_Callback(hObject, eventdata, handles)
% hObject    handle to z (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global time fx fy fz h_hpf h_lpf new_time new_y_hpf f_res;

axes(handles.graph);
ylim(handles.graph,'auto')

if (get(handles.orig,'value'))
new_time=time;
new_y_hpf=fz;
plot(time(1:(end-500)),fz(1:(end-500)))
end

if(get(handles.filt,'value'))
ylim(handles.graph,'auto')

fx_m = fz;
ti_x = time;
temp_mean=mean(fx_m);
fxm = fx_m- temp_mean;

y_hpf = filter(h_hpf,1,fxm);

new_time=ti_x;
new_y_hpf=y_hpf;
new_time((length(ti_x)-498):end)=[];
new_y_hpf(1:499)=[];


hold on
plot(new_time,new_y_hpf,'b');
hold off
ylim([-1 1]);

end


% --- Executes on button press in res.
function res_Callback(hObject, eventdata, handles)
% hObject    handle to res (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --- Executes on button press in x.

global time fx fy fz h_hpf h_lpf new_time new_y_hpf f_res;

axes(handles.graph);
ylim(handles.graph,'auto')

if (get(handles.orig,'value'))
plot(time(1:(end-500)),f_res(1:(end-500)))
end
```

```
if(get(handles.filt,'value'))
ylim(handles.graph,'auto')

fx_m = f_res;
ti_x = time;
temp_mean=mean(fx_m);
fxm = fx_m- temp_mean;

y_hpf = filter(h_hpf,1,fxm);

new_time=ti_x;
new_y_hpf=y_hpf;
new_time((length(ti_x)-498):end)=[];
new_y_hpf(1:499)=[];

hold on
plot(new_time,new_y_hpf,'b');
hold off
ylim([-1 1]);

end
guidata(hObject,handles);


function x1_Callback(hObject, eventdata, handles)
% hObject    handle to x1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of x1 as text
%        str2double(get(hObject,'String')) returns contents of x1 as a
double


% --- Executes during object creation, after setting all properties.
function x1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to x1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function x2_Callback(hObject, eventdata, handles)
% hObject    handle to x2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of x2 as text
%        str2double(get(hObject,'String')) returns contents of x2 as a
double


% --- Executes during object creation, after setting all properties.
function x2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to x2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in orig.
function orig_Callback(hObject, eventdata, handles)
% hObject    handle to orig (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of orig


% --- Executes on button press in filt.
function filt_Callback(hObject, eventdata, handles)
% hObject    handle to filt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of filt


% --- Executes on button press in clr_data.
function clr_data_Callback(hObject, eventdata, handles)
% hObject    handle to clr_data (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

cla(handles.graph)
```

```
function force_delay_Callback(hObject, eventdata, handles)
% hObject    handle to force_delay (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of force_delay as text
%        str2double(get(hObject,'String')) returns contents of
force_delay as a double


% --- Executes during object creation, after setting all properties.
function force_delay_CreateFcn(hObject, eventdata, handles)
% hObject    handle to force_delay (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes on selection change in analysis_tools.
function analysis_tools_Callback(hObject, eventdata, handles)
% hObject    handle to analysis_tools (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns
analysis_tools contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
analysis_tools
global time fx fy fz h_hpf h_lpf new_time new_y_hpf f_res;

contents = get(hObject,'Value');
k1=str2double(get(handles.x1,'String'));
[~,k1]=ismember(k1,new_time);
k2=str2double(get(handles.x2,'String'));
[~,k2]=ismember(k2,new_time);
init_force_delay=str2double(get(handles.force_delay,'String'));
[~,init_force_delay]=ismember(init_force_delay,new_time);

switch contents
    case 2
        temp=new_y_hpf((k1+init_force_delay):(k2+init_force_delay));
        [~,cal_results]=max(abs(temp));
        cal_results=cal_results+k1+init_force_delay-1;
        tempstr=strcat('max value = ',num2str(new_y_hpf(cal_results)));
        set(handles.cal_results,'String',tempstr);
        hold on

plot(handles.graph,new_time(cal_results),new_y_hpf(cal_results),'g*')
        hold off
     case 3
        temp=new_y_hpf((k1+init_force_delay):(k2+init_force_delay));
        [~,cal_results]=min(abs(temp));
        cal_results=cal_results+k1+init_force_delay-1;
        tempstr=strcat('min value = ',num2str(new_y_hpf(cal_results)));
        set(handles.cal_results,'String',tempstr);
        hold on

plot(handles.graph,new_time(cal_results),new_y_hpf(cal_results),'g*')
        hold off
    case 4
        f=new_y_hpf;
        thresh=0.1;
        I1=find(f>thresh);
        I2=find(f<-thresh);
        t_I1=f(I1);
        t2_I1=time(I1);
        [p_I1,loc_I1]=findpeaks(t_I1,'MinPeakDistance',400);
        t_I2=f(I2);
        t2_I2=time(I2);
        [p_I2,loc_I2]=findpeaks(abs(t_I2),'MinPeakDistance',400);
        hold('on')
        plot(handles.graph,t2_I1(loc_I1),t_I1(loc_I1),'m*');
        plot(handles.graph,t2_I2(loc_I2),t_I2(loc_I2),'m*');
        hold('off')
    otherwise
end


% --- Executes during object creation, after setting all properties.
function analysis_tools_CreateFcn(hObject, eventdata, handles)
% hObject    handle to analysis_tools (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in fda_tool.
function fda_tool_Callback(hObject, eventdata, handles)
% hObject    handle to fda_tool (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
fdatool


% --- Executes on button press in thresh_val.
function thresh_val_Callback(hObject, eventdata, handles)
% hObject    handle to thresh_val (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global img;
img=getimage(handles.vid);
[~,~,temp]=impixel;
pause(0.3)
disp(temp)
handles.thres=temp;

% --- Executes on button press in thresh.
function thresh_Callback(hObject, eventdata, handles)
% hObject    handle to thresh (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of thresh
```

Appendix C

Code for parameter calculation and plotting graphs

Menu

```
try
prompt=strcat('select one of the following options:\n','[0]clear
workspace\n','[1]import data\n','[2]change parameters\n','[3]analyze
data\n','[4]plot data\n','[5]zoom data\n','[6]findpeaks\n','[7]save
data\n','[8]close all figure window\n','[9]plot(x,y,z) for initial
points\n');
option=input(prompt);

switch option
    case 0
        clear
    case 1
        run('import_data.m');
        run('import_timestamp.m');
        pause(0.5)

X={'pname','raw','rawData','regexstr','result','row','invalidThousandsS
eparator','me','numbers','numericData','col','data','dataArray','delimi
ter','fileID','filename','filename3','fname','formatSpec','R','ans'};
        clear(X{:});
    case 2
        fname=input('input video filename:\n','s');
        g_title=input('enter participant skill:\n','s');
        init_frame=input('frame no (initial force):\n ');
        t_init=input('time(initial force):\n ');
        frame_rate=input('frame rate:\n ');
    case 3
        run('analysis_timestamp.m')
    case 4
        g_handle=[];
        force_g_handle={};
        f_handle=[];
        zoom=[];
        run('plot_timestamp.m')
    case 5
        run('zoom_timestamp.m')
    case 6
        run('findpeaks_timestamp.m')
    case 7
        run('save_timestamp.m')
    case 8
        close(findall(0,'Type','figure'));
        g_handle=[];
        force_g_handle={};
        f_handle=[];
        zoom=[];
    case 9
        run('plot_test.m')
    otherwise
        disp('execute the code and enter the right value')
end
catch
    warning('enter right integer value');
end
```

Functions

```
function
[t_init_v,t_in_v,t_out_v,t_pick_v,t_cout_v]=f_get_indices_timestamp(t_i
nit,init_frame,frame_rate,time,t_in,t_out,t_pick,t_cout)
t_vinit=init_frame/frame_rate;
tdiff=t_vinit-t_init;
t_in=t_in-tdiff;
t_out=t_out-tdiff;
t_pick=t_pick-tdiff;
t_cout=t_cout-tdiff;
for i=1:length(t_in)
    t_in(i)=str2double(sprintf('%.3f',t_in(i)));
end

for i=1:length(t_out)
    t_out(i)=str2double(sprintf('%.3f',t_out(i)));
end

for i=1:length(t_pick)
    t_pick(i)=str2double(sprintf('%.3f',t_pick(i)));
end

for i=1:length(t_cout)
    t_cout(i)=str2double(sprintf('%.3f',t_cout(i)));
end
[~,t_init_v]=ismember(t_init,time);
[~,t_in_v]=ismember(t_in,time);
[~,t_out_v]=ismember(t_out,time);
[~,t_pick_v]=ismember(t_pick,time);
[~,t_cout_v]=ismember(t_cout,time);

function
f_findpeaks_timestamp(i,force_g_handle,g_handle,fx,fy,fz,time,thresh)
f=eval(force_g_handle{i});
I1=find(f>thresh);
I2=find(f<-thresh);
t_I1=f(I1);
t2_I1=time(I1);
[p_I1,loc_I1]=findpeaks(t_I1,'MinPeakDistance',200);
t_I2=f(I2);
t2_I2=time(I2);
```

```
[p_I2,loc_I2]=findpeaks(abs(t_I2),'MinPeakDistance',200);
hold(g_handle(i),'on')
plot(g_handle(i),t2_I1(loc_I1),t_I1(loc_I1),'m*');
plot(g_handle(i),t2_I2(loc_I2),t_I2(loc_I2),'m*');
hold(g_handle(i),'off')

function
f_save_analysis_timestamp(fname,avg_force_sec,abs_avg_force_sec,peak_fo
rce_sec,avg_time_sec,time_total)

fileID = fopen(strcat(fname,'_analysis.txt'),'wt');
fprintf(fileID,'%0s %17s %15s %12s\n','avg force','abs avg force','peak
force','avg time');
fprintf(fileID,' %.4f \t\t %.4f \t\t %.4f \t\t %.4f\n',[avg_force_sec
abs_avg_force_sec peak_force_sec avg_time_sec]');
fprintf(fileID,'%0s\n','time_sec');
fprintf(fileID,'%.4f',time_total);
fclose(fileID);

function
[avg_force_sec,abs_avg_force_sec,peak_force_sec,avg_time_sec,time_total
]=f_analyze_timestamp(f,time,t_in_v,t_cout_v)
len=length(t_in_v);
avg_force_sec=zeros(len,1);
abs_avg_force_sec=zeros(len,1);
peak_force_sec=zeros(len,1);
avg_time_sec=zeros(len,1);

for i=1:len
    avg_force_sec(i)=mean(f(t_in_v(i):t_cout_v(i)));
end

for i=1:len
    abs_avg_force_sec(i)=mean(abs(f(t_in_v(i):t_cout_v(i))));
end

for i=1:len
    [~,j]=max(abs(f(t_in_v(i):t_cout_v(i))));
    peak_force_sec(i)=f(t_in_v(i)+j-1);
end

for i=1:len
    avg_time_sec(i)=time(t_cout_v(i))-time(t_in_v(i));
end

time_total=time(t_cout_v(len))-time(t_in_v(1));
```

Import data

```
%% Initialize variables.
filename = filename3;
```

```
delimiter = ',';

%% Read columns of data as strings:
% For more information, see the TEXTSCAN documentation.
formatSpec = '%s%s%s%s%[^\n\r]';

%% Open the text file.
fileID = fopen(filename,'r');

%% Read columns of data according to format string.
% This call is based on the structure of the file used to generate this
% code. If an error occurs for a different file, try regenerating the
code
% from the Import Tool.
dataArray = textscan(fileID, formatSpec, 'Delimiter', delimiter,
'ReturnOnError', false);

%% Close the text file.
fclose(fileID);

%% Convert the contents of columns containing numeric strings to
numbers.
% Replace non-numeric strings with NaN.
raw = repmat({''},length(dataArray{1}),length(dataArray)-1);
for col=1:length(dataArray)-1
    raw(1:length(dataArray{col}),col) = dataArray{col};
end
numericData = NaN(size(dataArray{1},1),size(dataArray,2));

for col=[1,2,3,4]
    % Converts strings in the input cell array to numbers. Replaced
non-numeric
    % strings with NaN.
    rawData = dataArray{col};
    for row=1:size(rawData, 1);
        % Create a regular expression to detect and remove non-numeric
prefixes and
        % suffixes.
        regexstr = '(?<prefix>.*?)(?<numbers>([-
]*(\d+[\,]*)+[\.]{0,1}\d*[eEdD]{0,1}[-+]*\d*[i]{0,1})|([-
]*(\d+[\,]*)*[\.]{1,1}\d+[eEdD]{0,1}[-+]*\d*[i]{0,1}))(?<suffix>.*)';
        try
            result = regexp(rawData{row}, regexstr, 'names');
            numbers = result.numbers;

            % Detected commas in non-thousand locations.
            invalidThousandsSeparator = false;
            if any(numbers==',');
                thousandsRegExp = '^\d+?(\,\d{3})*\.{0,1}\d*$';
                if isempty(regexp(thousandsRegExp, ',', 'once'));
                    numbers = NaN;
                    invalidThousandsSeparator = true;
                end
            end
            % Convert numeric strings to numbers.
            if ~invalidThousandsSeparator;
                numbers = textscan(strrep(numbers, ',', ''), '%f');
                numericData(row, col) = numbers{1};
                raw{row, col} = numbers{1};
            end
        catch me
        end
    end
end


%% Replace non-numeric cells with NaN
R = cellfun(@(x) ~isnumeric(x) && ~islogical(x),raw); % Find non-
numeric cells
raw(R) = {NaN}; % Replace non-numeric cells

%% Allocate imported array to column variable names
t_in = cell2mat(raw(:, 1));
t_out = cell2mat(raw(:, 2));
t_pick = cell2mat(raw(:, 3));
t_cout = cell2mat(raw(:, 4));

t_in(1) = [];
t_out(1) = [];
t_pick(1) = [];
t_cout(1) = [];
```

Analyze data

```
[t_init_v,t_in_v,t_out_v,t_pick_v,t_cout_v]=f_get_indices_timestamp(t_i
nit,init_frame,frame_rate,time,t_in,t_out,t_pick,t_cout);
[avg_force_sec,abs_avg_force_sec,peak_force_sec,avg_time_sec,time_total
]=f_analyze_timestamp(fx,time,t_in_v,t_cout_v);
f_save_analysis_timestamp(strcat(fname,'_fx'),avg_force_sec,abs_avg_for
ce_sec,peak_force_sec,avg_time_sec,time_total);

%[t_init_v,t_in_v,t_out_v,t_pick_v,t_cout_v]=f_get_indices_timestamp(t_
init,init_frame,frame_rate,time,t_in,t_out,t_pick,t_cout);
[avg_force_sec,abs_avg_force_sec,peak_force_sec,avg_time_sec,time_total
]=f_analyze_timestamp(fy,time,t_in_v,t_cout_v);
f_save_analysis_timestamp(strcat(fname,'_fy'),avg_force_sec,abs_avg_for
ce_sec,peak_force_sec,avg_time_sec,time_total);

%[t_init_v,t_in_v,t_out_v,t_pick_v,t_cout_v]=f_get_indices_timestamp(t_
init,init_frame,frame_rate,time,t_in,t_out,t_pick,t_cout);
[avg_force_sec,abs_avg_force_sec,peak_force_sec,avg_time_sec,time_total
]=f_analyze_timestamp(fz,time,t_in_v,t_cout_v);
f_save_analysis_timestamp(strcat(fname,'_fz'),avg_force_sec,abs_avg_for
ce_sec,peak_force_sec,avg_time_sec,time_total);
```

Plot data

```
figx=figure();
hfigx=gca;
p1=plot(time,fx,'k');
title(strcat(g_title,': Applied acceleration in x-direction'));
xlabel('Time (sec)');
ylabel(' Acceleration (m/s^2)');
hold on
p2=plot(time(t_init_v),fx(t_init_v),'ro');
p3=plot(time(t_in_v),fx(t_in_v),'ro','MarkerFaceColor','r');
p4=plot(time(t_out_v),fx(t_out_v),'mo','MarkerFaceColor','m');
p5=plot(time(t_pick_v),fx(t_pick_v),'go','MarkerFaceColor','g');
p6=plot(time(t_cout_v),fx(t_cout_v),'bo','MarkerFaceColor','b');
%plot(time(1:(end-1)),diff(fx),'r')
hold off
%legend([p1 p2 p3 p4 p5 p6],'force data','initial force','start
suture','needle out','pick needle','end suture');
set(figx,'Units', 'Normalized', 'OuterPosition', [0 0 1 1]);

set(hfigx,'LineWidth',2.5,'FontWeight','bold','FontSize',14);
figtitle=get(hfigx,'Title');
set(figtitle,'FontWeight','bold','FontSize',16);
figxlabel=get(hfigx,'XLabel');
figylabel=get(hfigx,'YLabel');
set(figxlabel,'FontWeight','bold','FontSize',16);
set(figylabel,'FontWeight','bold','FontSize',16);


figy=figure;
hfigy=gca;
p1=plot(time,fy,'k');
title(strcat(g_title,': Applied acceleration in y-direction'));
xlabel('Time (sec)');
ylabel(' Acceleration (m/s^2)');
hold on
p2=plot(time(t_init_v),fy(t_init_v),'ro');
p3=plot(time(t_in_v),fy(t_in_v),'ro','MarkerFaceColor','r');
p4=plot(time(t_out_v),fy(t_out_v),'mo','MarkerFaceColor','m');
p5=plot(time(t_pick_v),fy(t_pick_v),'go','MarkerFaceColor','g');
p6=plot(time(t_cout_v),fy(t_cout_v),'bo','MarkerFaceColor','b');
%plot(time(1:(end-1)),diff(fy),'r')
hold off
%legend([p1 p2 p3 p4 p5 p6],'force data','initial force','start
suture','needle out','pick needle','end suture');
set(figy,'Units', 'Normalized', 'OuterPosition', [0 0 1 1]);

set(hfigy,'LineWidth',2.5,'FontWeight','bold','FontSize',14);
figtitle=get(hfigy,'Title');
set(figtitle,'FontWeight','bold','FontSize',16);
figxlabel=get(hfigy,'XLabel');
```

```
figylabel=get(hfigy,'YLabel');
set(figxlabel,'FontWeight','bold','FontSize',16);
set(figylabel,'FontWeight','bold','FontSize',16);


figz=figure;
hfigz=gca;
p1=plot(time,fz,'k');
title(strcat(g_title,': Applied acceleration in z-direction'));
xlabel('Time (sec)');
ylabel(' Acceleration (m/s^2)');
hold on
p2=plot(time(t_init_v),fz(t_init_v),'ro');
p3=plot(time(t_in_v),fz(t_in_v),'ro','MarkerFaceColor','r');
p4=plot(time(t_out_v),fz(t_out_v),'mo','MarkerFaceColor','m');
p5=plot(time(t_pick_v),fz(t_pick_v),'go','MarkerFaceColor','g');
p6=plot(time(t_cout_v),fz(t_cout_v),'bo','MarkerFaceColor','b');
%plot(time(1:(end-1)),diff(fz),'r')
hold off
%legend([p1 p2 p3 p4 p5 p6],'force data','initial force','start
suture','needle out','pick needle','end suture');
set(figz,'Units', 'Normalized', 'OuterPosition', [0 0 1 1]);

set(hfigz,'LineWidth',2.5,'FontWeight','bold','FontSize',14);
figtitle=get(hfigz,'Title');
set(figtitle,'FontWeight','bold','FontSize',16);
figxlabel=get(hfigz,'XLabel');
figylabel=get(hfigz,'YLabel');
set(figxlabel,'FontWeight','bold','FontSize',16);
set(figylabel,'FontWeight','bold','FontSize',16);

zoom=horzcat(zoom,[0 0 0]);
force_g_handle=vertcat(force_g_handle,{'fx';'fy';'fz'});
f_handle=horzcat(f_handle,[figx figy figz]);
g_handle=horzcat(g_handle,[hfigx hfigy hfigz]);
clear('p1','p2','p3','p4','p5','p6')
```

Find peaks

```
for i=1:length(g_handle)
    f_findpeaks_timestamp(i,force_g_handle,g_handle,fx,fy,fz,time,0.2);
end
```

Zoom graph

```
range=input('input range [x1 x2]: ');
run('plot_timestamp.m')
pause(0.5)

zoom(end-2:end)=[figx figy figz];
set(hfigx,'Xlim',range)
set(hfigy,'Xlim',range)
set(hfigz,'Xlim',range)
```

Save graphs

```
for i=1:length(f_handle)
    fig.PaperPositionMode = 'auto';
    if zoom(i)~=0

savefig(f_handle(i),strcat(fname,'_',force_g_handle{i},'_zoom'));

print(f_handle(i),strcat(fname,'_',force_g_handle{i},'_zoom'),'-
dpng','-r0');
    else
        savefig(f_handle(i),strcat(fname,'_',force_g_handle{i}));
        print(f_handle(i),strcat(fname,'_',force_g_handle{i}),'-
dpng','-r0');
    end
end
```

REFERENCES

[1] B. N. Carter, “The fruition of Halsted’s concept of surgical training,” *Surgery*, vol. 32, no. 3, pp. 518–527.

[2] R. K. Reznick and H. MacRae, “Teaching Surgical Skills — Changes in the Wind,” *N. Engl. J. Med.*, vol. 355, no. 25, pp. 2664–2669, Dec. 2006.

[3] J. Torkington, S. G. Smith, B. I. Rees, and A. Darzi, “The role of simulation in surgical training.,” *Ann. R. Coll. Surg. Engl.*, vol. 82, no. 2, pp. 88–94, Mar. 2000.

[4] J. A. Wong and E. D. Matsumoto, “Primer: cognitive motor learning for teaching surgical skill[mdash]how are surgical skills taught and assessed?,” *Nat Clin Pr. Urol*, vol. 5, no. 1, pp. 47–54, Jan. 2008.

[5] D. Stefanidis, M. W. Scerbo, P. N. Montero, C. E. Acker, and W. D. Smith, “Simulator Training to Automaticity Leads to Improved Skill Transfer Compared With Traditional Proficiency-Based Training: A Randomized Controlled Trial,” *Ann. Surg.*, vol. 255, no. 1, pp. 30–37, 2012.

[6] D. Stefanidis, J. R. Korndorffer Jr., R. Sierra, C. Touchard, J. B. Dunne, and D. J. Scott, “Skill retention following proficiency-based laparoscopic simulator training,” *Surgery*, vol. 138, no. 2, pp. 165–170, Aug. 2005.

[7] R. E. Glasgow, K. A. Adamson, and S. J. Mulvihill, “The benefits of a dedicated minimally invasive surgery program to academic general surgery practice,” *J. Gastrointest. Surg.*, vol. 8, no. 7, pp. 869–873, Nov. 2004.

[8] M. K. Chmarra, N. H. Bakker, C. A. Grimbergen, and J. Dankelman, “TrEndo, a device for tracking minimally invasive surgical instruments in training setups,” *Sens. Actuators Phys.*, vol. 126, no. 2, pp. 328–334, Feb. 2006.

[9] K. R. Van Sickle, D. A. M. III, A. G. Gallagher, and C. D. Smith, “Construct validation of the ProMIS simulator using a novel laparoscopic suturing task,” *Surg. Endosc. Interv. Tech.*, vol. 19, no. 9, pp. 1227–1231, Sep. 2005.

[10] E. G. G. Verdaasdonk, L. P. S. Stassen, L. J. Monteny, and J. Dankelman, “Validation of a new basic virtual reality simulator for training of basic endoscopic skills,” *Surg. Endosc. Interv. Tech.*, vol. 20, no. 3, pp. 511–518, Mar. 2006.

[11] T. Horeman, S. Rodrigues, F.-W. Jansen, J. Dankelman, and J. van den Dobbelsteen, “Force measurement platform for training and assessment of laparoscopic skills,” *Surg. Endosc.*, vol. 24, no. 12, pp. 3102–3108, Dec. 2010.

[12] T. Horeman, J. Dankelman, F. W. Jansen, and J. J. van den Dobbelsteen, “Assessment of Laparoscopic Skills Based on Force and Motion Parameters,” *Biomed. Eng. IEEE Trans. On*, vol. 61, no. 3, pp. 805–813, Mar. 2014.

[13] A. Dubrowski, R. Sidhu, J. Park, and H. Carnahan, “Quantification of motion characteristics and forces applied to tissues during suturing,” *Am. J. Surg.*, vol. 190, no. 1, pp. 131–136, Jul. 2005.

[14] J. B. Pagador, F. M. Sánchez-Margallo, L. F. Sánchez-Peralta, J. A. Sánchez-Margallo, J. L. Moyano-Cuevas, S. Enciso-Sanz, J. Usón-Gargallo, and J. Moreno, “Decomposition and analysis of laparoscopic suturing task using tool-motion analysis (TMA): improving the objective assessment,” *Int. J. Comput. Assist. Radiol. Surg.*, vol. 7, no. 2, pp. 305–313, Mar. 2012.

[15] A. Dosis, r Aggarwal, f Bello, and et al, “Synchronized video and motion analysis for the assessment of procedures in the operating theater,” *Arch. Surg.*, vol. 3, no. 140, pp. 293–299, Mar. 2005.

[16] A. Sánchez, O. Rodríguez, R. Sánchez, G. Benítez, R. Pena, O. Salamo, and V. Baez, “Laparoscopic Surgery Skills Evaluation: Analysis Based on Accelerometers,” *JSLS*, vol. 18, no. 4, p. e2014.00234, 2014.

[17] A. Trejos, R. Patel, R. Malthaner, and C. Schlachta, “Development of force-based metrics for skills assessment in minimally invasive surgery,” *Surg. Endosc.*, vol. 28, no. 7, pp. 2106–2119, Jul. 2014.

[18] A. C. Frischknecht, S. J. Kasten, S. J. Hamstra, N. C. Perkins, R. B. Gillespie, T. J. Armstrong, and R. M. Minter, “The Objective Assessment of Experts’ and Novices’ Suturing Skills Using An Image Analysis Program,” *Acad. Med.*, vol. 88, no. 2, 2013.

[19] G. Islam, K. Kahol, J. Ferrara, and R. Gray, “Development of Computer Vision Algorithm for Surgical Skill Assessment,” in *Ambient Media and Systems*, vol. 70, S. Gabrielli, D. Elias, and K. Kahol, Eds. Springer Berlin Heidelberg, 2011, pp. 44–51.

[20] R. C. Jackson and M. C. Cavusoglu, “Modeling of Needle-Tissue Interaction Forces During Surgical Suturing,” *Robot. Autom. ICRA 2012 IEEE Int. Conf. On*, pp. 4675–4680, May 2012.

[21] T. . Frick, D. . Marucci, J. . Cartmill, C. . Martin, and W. . Walsh, “Resistance forces acting on suture needles,” *J. Biomech.*, vol. 34, no. 10, pp. 1335–1340.

[22] S. Misra, K. B. Reed, B. W. Schafer, K. T. Ramesh, and A. M. Okamura, “Observations of Needle-Tissue Interactions,” *Robot. Autom. 2009 ICRA 09 IEEE Int. Conf. On*, pp. 2687–2692, May 2009.

[23] R. Singapogu, T. Kavathekar, J. Eidt, R. Groff, and T. Burg, “A Novel Platform for Assessment of Surgical Suturing Skill: Preliminary Results.”

[24] S. Yamaguchi, D. Yoshida, H. Kenmotsu, T. Yasunaga, K. Konishi, S. Ieiri, H. Nakashima, K. Tanoue, and M. Hashizume, “Objective assessment of laparoscopic suturing skills using a motion-tracking system,” *Surg. Endosc.*, vol. 25, no. 3, pp. 771–775, Mar. 2011.

[25] “Six-Axis Force/Torque Sensor System Installation and Operation Manual.” ATI industrial Automation.

[26] “QUARC User’s Guide.” Quanser Inc., 29-Aug-2013.

[27] T. Raman, \iBasic Knotting and Suturing Using a Needle Holder. https://www.youtube.com/watch?v=HD6mll1wN0I.