8-2016

A Suture Training System with Synchronized Force, Motion and Video Data Collection

Naren Nagarajan
Clemson University, narenn@clemson.edu

A SUTURE TRAINING SYSTEM WITH SYNCHRONIZED FORCE, MOTION AND VIDEO DATA COLLECTION

A Thesis
Presented to
the Graduate School of
Clemson University

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

---

by
Naren Nagarajan
August 2016

Accepted by:
Dr. Richard Groff, (Committee Chair), Department of Electrical and Computer Engineering
Dr. Ravikiran Singapogu, Department of Bioengineering
Dr. Ian Walker, Department of Electrical and Computer Engineering

ABSTRACT

Suturing is a common surgical task where surgeons stitch a particular tissue. There is an increasing demand for a tool to objectively quantify and train surgical skills. Suturing is particularly difficult to teach due to various multi-modal aspects involved in the task including applied forces, hand motion and optimal time for suturing.

Towards quantifying the task of suturing, a platform is required to capture force, motion and video data while performing surgical suturing. This objective data can potentially be used to evaluate performance of a trainee and provide feedback regarding improving suturing skill.

In the previous prototype of the platform, 3 key issues faced were synchronization of the three sensors, inadequate construction of the platform and the lack of a framework for image processing towards real-time assessment of suturing skill.

In order to improve the platform, the aforementioned issues have been addressed in specific ways. The data collected in the system is synchronized in real-time along with a video recording for image processing and the noise due to the platform is considerably reduced by making modification to the platform construction. Data was collected on the platform with 15 novice participants. Initial analysis validates the synchronization of the sensor data.

In the future, the suture skill of experts and novices will be analyzed using meaningful metrics and machine learning algorithms. This work has the potential of enabling objective and structured training and evaluation for next generation surgeons.

DEDICATION

This thesis is dedicated to my beloved parents Lalitha and S. Nagarajan for their unconditional support throughout my life. My work is a product of their nurturing and constant motivation.

I also dedicate this thesis to my sister Kavya who has always been there for me and has acted as my living guardian angel.

ACKNOWLEDGEMENT

I would like to thank Dr. Richard Groff for his guidance and wisdom throughout my graduate program. His valuable inputs and advice has been the pillars of this thesis.

I would like to thank Dr. Joseph Singapogu for his continued support and motivation throughout the thesis.

I would like to thank my project mates Anand Jagannathan and Irfan Kil for working alongside me, solving problems together and helping out during the tough times. I would like to especially thank Irfan for having been an integral part of my thesis. His support and guidance, particularly during the software development phase has enabled me to complete this thesis.

TABLE OF CONTENTS

LIST OF FIGURES

TABLE OF FIGURES - APPENDIX

LIST OF TABLES

1. Introduction

1.1. What is suturing

Suturing is the art of stitching a part of the body where there has been a rupture or tear. When a suture is performed, the tear is essentially stitched together using a needle and thread. It is very much similar to the art of sewing or stitching and is sometimes referred to as a "stitch" [1]. However, since it involves tears on the human body, more care needs to be taken. Suturing requires very skillful hand motion and forces need to be applied on the tissue with caution. There are various techniques for performing sutures, but the most common technique is the simple interrupted suture [2]. In this technique, the surgeon performs multiple sutures, tying a knot after each completed suture. Another technique similar to the interrupted suturing is the continuous suture technique. In a continuous suture, a similar technique to interrupted suturing is followed, except a knot is not tied at the end of each suture, but instead all the sutures are performed using the same thread, knotting and cutting the thread only at the end after all the sutures have been completed. This technique consists of multiple steps involving positioning the needle on a needle holder, performing the insertion of the needle into the tissue or object being sutured, followed by the removal of the needle, thus forming a suture. This procedure involves various movements of the hand including supination and pronation (Supination and pronation of the hand/forearm are movements of the forearm such that the palm of the hand faces upwards and downwards respectively). Since this basic yet complex task requires so much perfection,

it is important that students and residents are trained well in the art of suturing before they are ready to become surgeons [1].

Suturing requires very skillful hand motion and forces need to be applied on the tissue with caution. Thus, various characteristics pertaining to the art of suturing have been selected as metrics to help measure the skill level of a surgeon or aspiring resident or student. The different parameters are fundamentally force, motion, position and time based metrics that have been used to determine the skill level of the subject.

1.2. Difficulty in suturing

As mentioned previously, suturing is a basic yet complex task that must be perfected. The complexity comes from the various steps involved and the perfection that is required from the surgeon to perform the procedure. The needle must be held at a particular angle and the subject must have a good grip on the needle holder, which can be best obtained only in certain positions. The surgeon is required to follow the curvature of the needle while performing the suture [3]. This is done so as to apply minimum force on the tissue as the forces are always radial and not on the tissue, thus reducing the risk of tissue damage [2] [4]. Various studies have found that force, motion and video-capture can be used to develop metrics that can help evaluate the performance of the subject [5] [6] [7] [8] [9] [10] [11] [12].

The continuous suture technique [1] is demonstrated in the various figures shown in Figure 1. As one can see, the first step involves holding the needle tangentially with respect to the needle holder, and being inserted into the membrane

perpendicularly (As shown in Figure 1a). The second step involves puncturing the membrane at the point of entry, driving the needle through the membrane and puncturing the membrane again at the point of exit (As shown in Figure 1 - b & c). The first two steps require the subject to traverse the path of the needle so as to apply minimum force on the tissue. The third step involves changing the grip of the needle such that there is sufficient range of motion for following the path of the needle while pulling it out (As shown in Figure 1d). The fourth step involves pulling the needle out of the membrane, once again following the curve of the needle (as shown in Figure 1 - e & f). In all the figures shown in Figure 1, the hand motion can be observed to be following the arc of the needle.

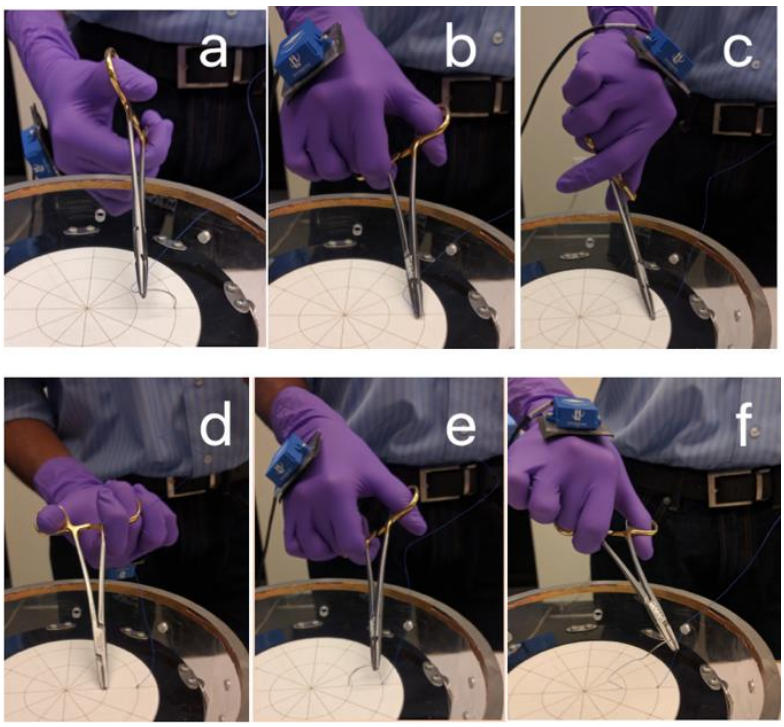

Figure 1: Procedure for performing a continuous suture

1.3. Assessing Suturing Skill

There are multiple methods of evaluating suturing skill. Today, suturing skill is evaluated using methods like operating on humans, cadavers, synthetic models and virtual reality simulators [13] [14] [15] [16]. Although studies have shown that there are not great differences between the different methods of teaching suturing [15], it can be said that the cost of using synthetic models over human cadavers, virtual reality simulators and human patients is much less. Therefore, a suturing platform has been created which using synthetic models for subjects to perform experiments on, while various sensors record data for analysis. Upon completion of the experiment, this recorded data is analyzed and meaningful evaluations can be made based on the data analyzed.

In order to measure the different parameters for evaluating suturing skill quantitatively, a device is required that can measure forces and motion-profile data. Videos also need to be recorded for video analysis. All these sensors can be mounted on a simulator or platform that can run as one unit, measuring all these readings while the subject performs the experiment.

1.4. The Suture Platform

In order to help train residents and novice students in the art of suturing, a device called the Suture Platform was created. The previous version of the platform contained only one camera, placed on a tripod, viewing the experiment from outside the setup [17]. In order to process the experiment through video capture using image processing

algorithms, good quality, uninterrupted footage must be obtained. To do so, another camera was decided to be used, placed under the membrane so that there will be no noise due to external movements or variation in lighting and only vital needle and thread motion would be captured (referred to as an internal camera). Thus the new version of the device provides a platform for subjects to perform a suturing experiment while the device seamlessly records synchronized force and motion-profile data and records the experiment using 2 cameras. The internal camera recording is then processed to provide information regarding the needle and thread. Once the synchronized data has been collected, it will be used to analyze the subject's performance. This analysis can potentially be used to classify subjects based on their suturing skill and hopefully be able to provide feedback to the subject regarding their performance. The Suture Platform is shown below in Figure 2.

Figure 2: The Suture Platform

The ultimate aim of the project is to provide a complete training solution, where subjects can perform experiments, learn from their mistakes based on the evaluations and feedback provided by the platform's algorithm and improve on their skill over successive experiments on the platform. In the future, the device would also cost much less than any other method of suture training, aiming to cost a few hundreds of dollars, rather than thousands of dollars as compared to virtual reality simulators and the option of repeatability when compared to human and animal tissue cadavers.

2. Literature Review

The aim of the study is to come up with a device that can not only identify the skill level of a subject, but also try and provide feedback as to how the subject can improve himself or herself to become better at the task. Multiple theories of motor skill acquisition have been proposed and this aims to set up a device that can ultimately integrate various characteristics of multiple theories to create a complete training module that can identify skill level and provide suggestions for improvements to become better at the task.

2.1. Skill involved in suturing

To understand the art of suturing and what makes it so difficult that only expert surgeons can perform this task at the highest quality level, one must break down the task of suturing. Suturing is a form of motor skill and that requires dexterity, accuracy, precision and finesse. Training for suturing involves fine motor skills that are taught using multiple stages of teaching. These stages are explained by notable motor skill acquisition theories including the *Fitts-Posner* Theory, the *Schmidt* Theory and the *Kopta* Theory. The *Fitts-Posner* theory consists of three phases of motor learning; the cognitive, associative and autonomous phases [18] [19]. The three stages involve understanding the task, repeating the task and fine-tuning the skill by performing the task with improved dexterity, efficiency, accuracy and precision [18] [19].

Schmidt's Schema theory of motor skill focuses on learning from past experiences, which can be eventually incorporated into the system, where the subject learns from his

past mistakes based on reviews provided by the algorithm that identifies problem areas in the subject's experiment. Thus the subject can pick up on the mistakes he or she made in the previous attempt and try and focus on the metrics they performed worse in to try and improve their performance in the next attempt [20] [21]. The Schema Theory also explains the different stages of motor skill development that occur while the subject learns the art. The ultimate goal of the project is to create a device that performs the role of an analyst or teacher where the device will teach the subject how to improve their suturing skill based on the various metrics that are researched and found to be key characteristics of assessing suturing skill. These stages include understanding the task at hand, registering the steps involved in the task and understand and learning from the outcome of the experiment. [14] [21] [20]

Kopta's theory focuses more on the procedural steps that must be memorized and followed during the experiment. The subject has to breakdown the tasks of the experiment to truly understand what is required of them, before they begin the experiment. In this study, subjects who fall under the "novice" category are assumed to have had very basic to no suturing experience. This group is provided with some basic procedures to perform the suture. This basic procedure is explained procedurally, explaining the various different motions involved in the experiment and how to perform each of them individually and as well as how to perform the experiment as a whole. All other categories are expected to know how to suture and, following *Kopta's* theory, must break down the steps required to perform the experiment. [22]

2.2. Need for a training simulator

There are many methods of training that are currently used to train students and residents in the art of suturing. After the theoretical knowledge has been taught, students need to suture practically on tissue or tissue-like samples. Ideally, a live human patient would be used to perform the suture. However, the risk of injury is much greater when it comes to live human patients since the subject is still in training and is prone to making mistakes. This risk cannot be taken and so human cadavers are used. Human cadavers are not available in plenty and definitely not available in similar proportions to the number of students or resident trainees. Therefore, animal tissue and synthetic models of tissue are now used to perform the experiments. This lowers the cost of the experiment and also increases the repeatability. Since synthetic models can be manufactured in bulk, it is easy to provide tissue samples for students to suture on. Synthetic models also allow for a standard to be set since they can be manufactured according to some set standards and metrics. Thus all students are given the same synthetic model so that evaluation of the students is universal.

Haluck & Krummel [23] found that training outside the Operation Theater was actually more beneficial in terms of the stress level of the subject. Subjects were able to perform better in a simulated environment with an optimum stress to learning curve output as compared to when working in the Operation Theater. This has increased the need for surgical simulators which are used outside of the operation theater and do not require the need for human tissue samples, live or as cadavers.

2.3. State-of-the-art for surgical simulators

There are various simulators available today that use different approaches to surgical training. They include cadavers, Virtual Reality Simulators, bench trainers and physical platforms to track various parameters during suturing. Some examples are listed as follows:

- **RoSS by Simulated Surgical Systems**

The Robotic Surgical Simulator is a virtual reality surgical simulator that uses the da Vinci Surgical System by Intuitive Surgical as the base for training subjects. The da Vinci Surgical System is a highly sophisticated surgical platform that enables surgeons to easily perform minimally invasive surgery (MIS) [24]. It provides robotic assistance to the surgeon during the surgery and improves the efficiency of the surgery. RoSS is a platform that enables a surgeon to train on a virtual environment before using the da Vinci in the operation theater. It consists of arms, pedals and a virtual display that allows the subject to view their experiment. Various types of experiments can be conducted as it is a virtual environment [25]. Other virtual reality training simulators like SEP by SimSurgery, MSim and so on are very similar to one other and apply to different modules.

- **TrEndo Tracking System**

The TrEndo is a tracking system that was developed by *M.K Chmarra et al*. [16] which provides both physical instrument-tissue interaction as well as a mechanism for

tracking various motion during Minimally Invasive Surgery. It uses optical computer mouse sensors to capture motion information in 4 Degrees of Freedom (DOF) throughout the experiment. It focusses on minimally invasive surgical procedures and allows the subject to use MIS instruments to perform the experiment. It provides the concept of Augmented Reality in training simulators. [26]

- **ForMoST Force and motion surgical trainer**

ForMoST Force and motion surgical trainer is a similar surgical trainer that uses the TrEndo and ForceTRAP, devices created by *Horeman et al.* at MediShield [27]. This devise also uses physical instrument-tissue interaction which mimics real-life surgeries, unlike virtual reality simulators. This is a very similar concept to the Suture Platform, except the Suture Platform aims to use the data collected to not only give force data, but also provide evaluations and provide feedback for improvement.

- **Platform for force and motion measurement by Tim Horeman et al.**

Tim Horeman et al. [5] [7] have used bench trainers to create a force and motion measurement platform while performing suturing experiments in order to measure various force and motion parameters. This device also uses the TrEndo and ForceTRAP sensors for obtaining and recording data. In one of the devices, the force sensor used is set up on a set of springs that determine the range of the sensor. However, this causes the spring to absorb some of the forces that are being applied on the device as the spring will behave so as to try and oppose any change or motion in the platform resting on the springs.

The Suture Platform has a similar approach to measuring parameters, but uses additional sensors to focus on other parameters as well; and focusses on general suturing techniques and not on minimally invasive surgery in particular.

- **Sensor System for Effective Evaluation of Surgical Skill by Aizuddin et al.**

Aizuddin et al. [28] designed a sensor system for evaluating various metrics using force and motion sensors. Their motion sensing was based on the movement of the artificial skin rather than the movement of the subject's hand. The skill evaluation was based on time and force parameters as well as the motion of the artificial skin.

Their device did not consist of performance evaluation based on image processing or hand motion, which research has shown to be important parameters for skill evaluation.

2.4. Parameters for skill evaluation

Multiple studies have shown that force, motion and video data analysis are key parameters for assessing and evaluating the suturing skill of a subject. The parameters used in this study for evaluating suturing skill are based on various research findings that force, motion-profile and image processing are key parameters that can evaluate the skill level of a subject.

Various studies have shown that forces due to needle-tissue interaction are important parameters to help evaluate suturing skill. Various parameters observed

include absolute forces, mean forces, peak forces, force volume and so on [5] [6] [7] [8] [9] [10] [11] [29].

Aizuddin et. al [28] used analysis of motion of the tissue model for evaluating suturing skill, but other studies have used hand motion as the key motion-parameter for skill evaluation [5] [10] [11] [12].

Frischknecht et al. [30] found that image processing can be used a technique to evaluate objective skill level in suturing. *Islam et al.* [31] used images to evaluate various parameters and metrics for suturing skill like hand movement based on computer vision algorithms to track purple colored gloves worn by the subject.

3. The Suture Platform

The Suture Platform aims to build on all the various parameters that have been observed in other studies and combine multiple parameters into one system [29] [6] [7] [8] [9] [10] [11] [30] [31].

The Suture Platform aims to collect data from multiple sensors in a synchronized manner because once the synchronized data is collected, it will be analyzed by the system's algorithm and provide an evaluation based on the subject's performance against a large training set of data from various levels of suturing expertise. This project aims to build on such a concept and provide an economical approach to train students in the art of suturing [17].

3.1. Issues with the previous version of the Suture Platform

- **Absence of a camera for image processing**

Previous research has shown that image processing can be used to provide needle-movement and thread information. This information can be used to define certain metrics as well as help isolate sutures for improved data analysis. The previous version of the Suture Platform consisted of a camera positioned on a tripod outside of the platform for capturing a video of the top of the membrane. This camera allows the user to see how the subject performs the suture by seeing hand movement and other characteristics that are visible from outside the platform and on top of the membrane. This camera was not capable of capturing precise needle movements and could only be

used to help understand the force sensor and IMU readings. The platform did not have a dedicated camera for recording high quality frames of needle and thread movement from underneath the membrane. Recording a video underneath of the membrane is potentially capable of running image processing algorithms to provide valuable needle and thread information during each suture. [17]

- **Issues with platform construction**

After several iterations of design prototypes, the Suture Platform was finally built in 2014. However, being the first functional prototype, it had several flaws that required necessary modifications to become a viable product that can produce consistent and accurate results. Figure 3 below shows the previous version of the Suture Platform [17].

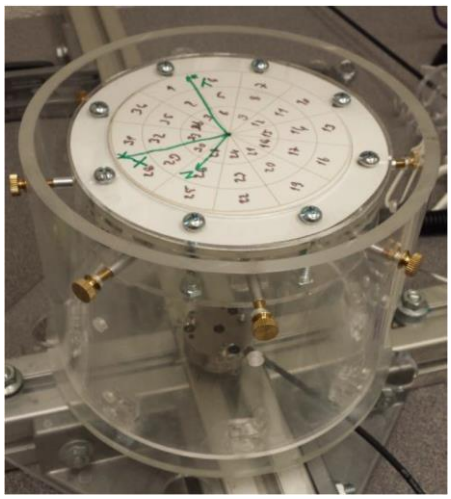

Figure 3: Previous version of the Suture Platform [17]

As one can see, the old design had the entire outer casing and membrane housing as one single unit, mounted on top of an ATI Mini 40 force sensor [32]. This

made the unit bulky and increased the vibrations and settling time of the system due to any movement or disturbance in the platform. This can be seen in experiments conducted to compare the vibrations produced by the old and new design. For instance, whenever the needle pierces the membrane, a sharp peak is observed in the force sensor. Due to the bulkiness of the unit supported by the force sensor, the entire unit vibrated for some time and took a long time to settle. This increased the settling time of the force readings considerably. Ideally, the subject is not supposed to touch the outer casing and it is supposed to act as a boundary for the hand movement, mimicking sutures required in deeper parts of the body, where the range of hand motion is restricted. However, inevitably subjects tend to touch the outer casing occasionally while performing the experiment. When this happens, large forces are observed on the force sensor due to the outer casing. These forces tend to be much greater than the force being applied on the membrane and produces inconsistent membrane force measurements [17]. To eliminate these unnecessary forces produced due to outer casing disturbances, a new design is created where the outer casing was detached from the membrane housing and it is also separated from the force sensor. Therefore, only the membrane housing influences the force sensor and any noise from the outer casing causes negligible variations in the force sensor.

The force sensor was placed on an aluminum cross-shaped framework. This framework was supported by a foam mat, which was used to dampen any noise or disturbances experienced by the setup during the experiment. The setup was also

dependent on a very sturdy table that would not contribute to the noise in the system [17]. In order to eliminate the requirement of a table of certain height, the new design incorporated a framework that replaced any table and made the setup a standalone device, capable of being transported to any location as one unit, ready for use upon arrival.

- **Issues with data synchronization**

The previous version of the platform consisted of the three sensors being recorded in multiple software platforms.

MATLAB was used as the main software application to run majority of the program. The ATI Mini40 force sensor was run using a National Instrument M-Series Data Acquisition Card (hereafter referred to as DAQ). This DAQ was integrated in MATLAB using QUANSER Simulink Blocks. An InterSense InertiaCube 4 Inertial Measurement Unit (IMU) was run using the InterSense SDK, which only ran C code, using the Visual Studio Integrated Development Environment (IDE). The values from the IMU were passed to MATLAB using shared memory protocol. A camera was also used to view the experiment, running on a separate computer. [17] [32] [33] [34] [35]

Shared memory is a method of allocating memory in the RAM that can be used by multiple processes. However, basic shared memory does not offer any synchronization functionality. The data from the sensors had to be processed after collection of the data to synchronize the samples based on the start and end times of each sensor. It was a

very tedious process to synchronize the data from 3 different devices and an easier, more efficient method of synchronization was required [17].

Due to the issues with the platform construction and data synchronization, as well as the requirement of an internal camera for image processing, a new platform design was deemed necessary. This new design was aimed at providing consistent synchronized, noise-free data throughout each experiment over all the experiments conducted.

3.2. New Suture Platform

The new platform consists of a force sensor, an IMU and two cameras for recoding videos of the experiment. A basic system-level diagram is shown below in Figure 4.The IMU and the two cameras are connected to the CPU (Central Processing Unit) via USBs (Universal Serial Bus) while the force sensor is interfaced using an M-Series NI-DAQ (National Instruments Data Acquisition system) [36]. These are processed using a C++ program.

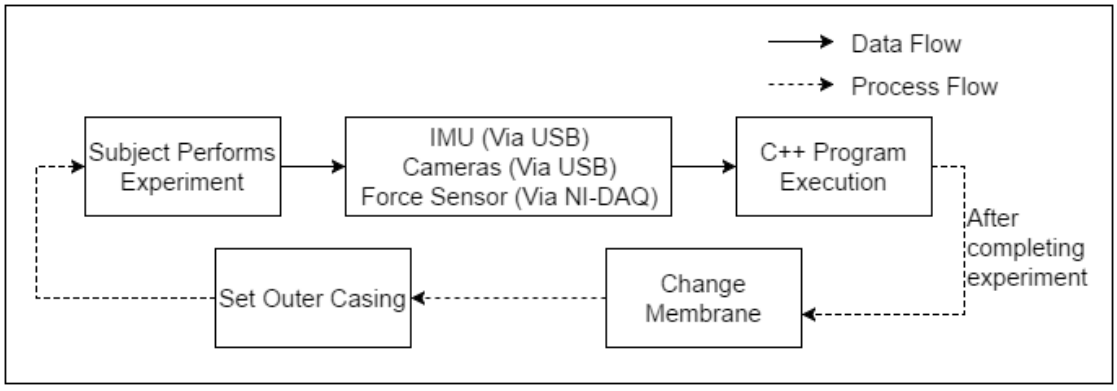

Figure 4: System Level Diagram for the Suture Platform

Once the experiment is completed, the research assistant replaces the membrane used by the subject so that a new membrane is used for each subject. The

outer casing of the platform is set to the desired level. After these steps, the system is ready for the next subject to perform the experiment.

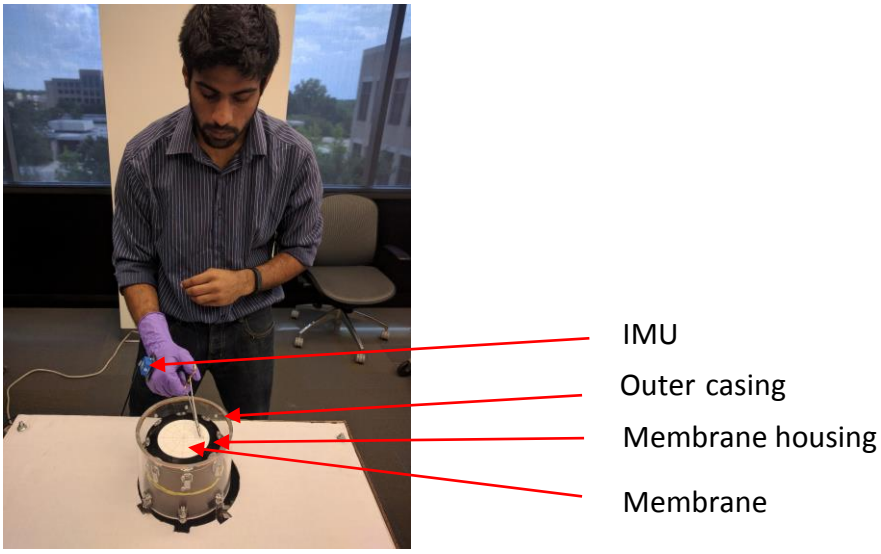

Figure 5: A subject performing an experiment on the new suture platform

3.2.1. Construction of the Suture Platform

- **Membrane Housing**

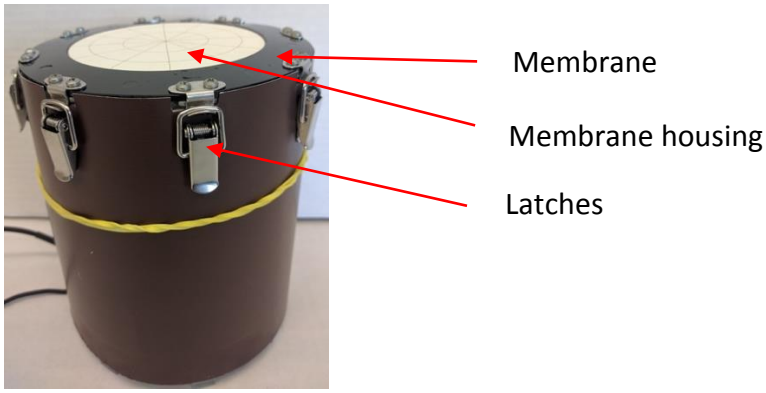

Figure 6: Membrane Housing

The new system requires an internal camera that has a view from under the membrane so that image-processing algorithms may be applied to the frames captured,

thus providing important needle and thread information during the experiment. The old design had no room for an internal camera and therefore a new design had to be created. The new design consists of not only the membrane (whose holes are inserted into acrylic protrusions as shown in Figure 7), made taut by pressing it against an elevated circular acrylic, but also an enclosure for housing the camera (as shown in Figure 6).

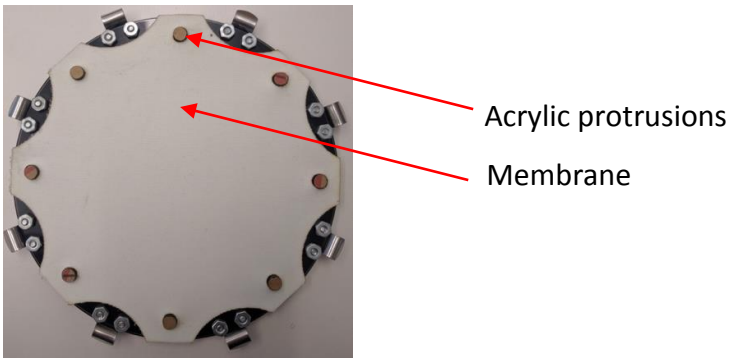

Figure 7: Membrane held by acrylic protrusions

This setup is placed directly on top of the force sensor, on an acrylic plate. Due to the multi-layered design (as shown in Figure 8), it is easy to detach the membrane housing in order to fix issues that may occur in the setup, like shifting of the camera, dust covering the lens, etc. It is also much easier to replace membranes as compared to the earlier design [17].

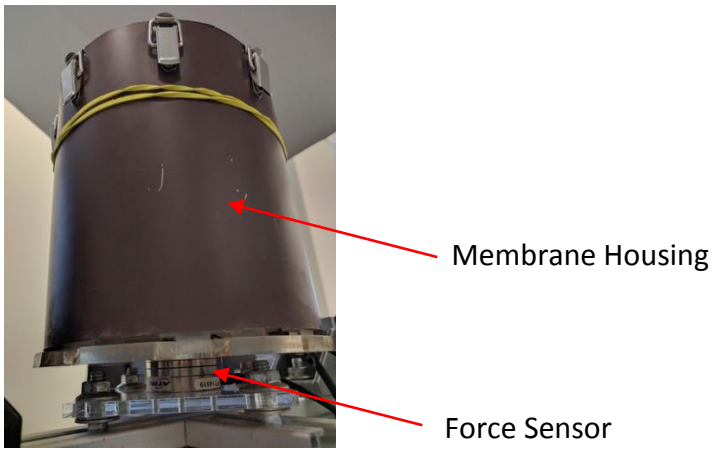

Figure 8: Multi-layered design of membrane housing on force sensor

In the earlier design, the membrane was held tightly by two thin, hollow cylinders of acrylic, screwed together at 8 different points around the circular design. The acrylic membrane housing was in-turn attached to the outer casing by 8 more screws to ensure that the membrane housing is secure. Unscrewing and re-screwing both the membrane and the membrane housing was very cumbersome and a new design was required [17]. The new design provides a solution to this by using latches to hold the two hollow cylinders of acrylic, thus making it much easier to replace membranes. The new process only requires the latches to be opened, the membrane replaced and the latches closed again. Since the outer case is detached from the membrane housing and since the membrane housing infused with the setup placed on top of the force sensor, additional support is not required for the membrane housing as previously needed.

- **Outer Casing**

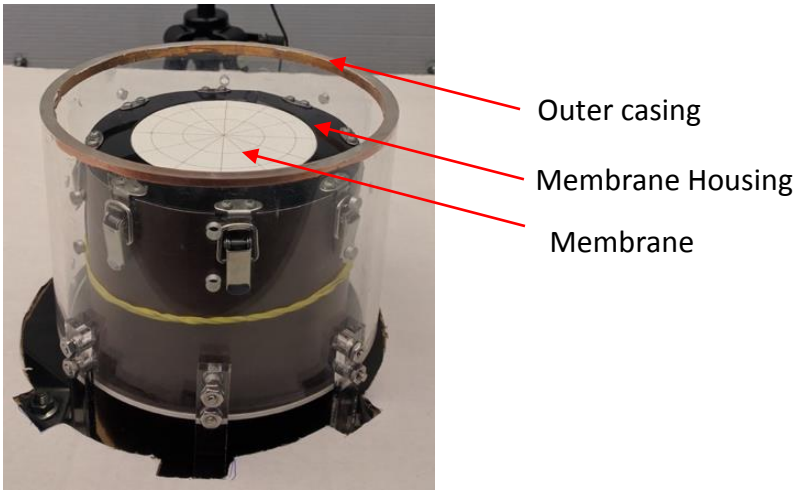

Figure 9: Outer casing with the housing membrane

The outer case is a hollow acrylic cylinder with the radius a little larger than that of the membrane housing, such that the membrane housing is placed inside of the outer casing (as shown in Figure 9). The case acts like a shield for the membrane and can be adjusted to height higher or lower than the membrane. The outer casing is supported by 6 limbs extending well below the force sensor, all attached to a sheet of acrylic (as shown in Figure 10). The sheet of acrylic is controlled by a stepper motor (Mercury Motor) to move upwards or downwards as desired. The inner aluminum framework explained later in the chapter is responsible for housing the stepper motor, which in turn is responsible for the upward or downward movement of the outer casing (as shown in Figure 11). At the start of each experiment, the stepper motor must be excited to make the outer case move upward to the desired position set for the experiment. At

the end of the experiment, the stepper motor is once again excited to lower the outer casing so that the membrane may be replaced for the next subject.

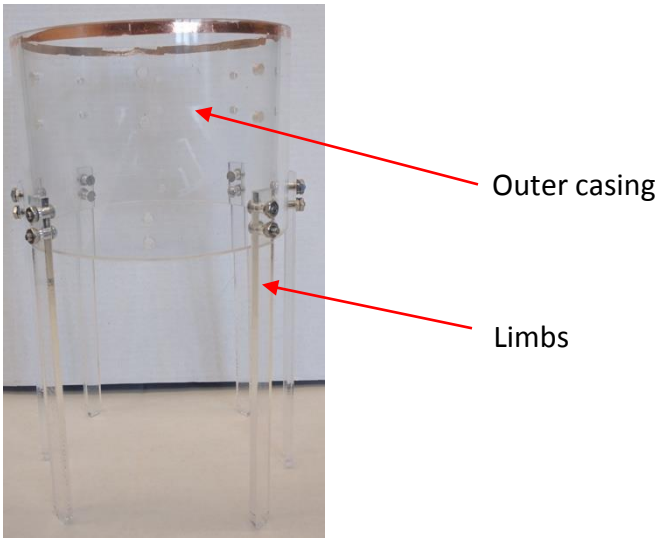

Figure 10: Outer casing

This represents various depth levels of the membrane, thus seeking to mimic real life situations where the suture target is located at some depth inside the body and consequently restricting hand motion while performing a suture. The aim is to ensure that the surgeon is trained in the art of working in confined spaces without touching nearby tissues and organs, thus reducing the risk of damage to the surrounding open tissue and other body parts. To mimic such suturing expertise, the subject is required to train with the outer casing at higher levels than the membrane, thus restricting the space available for the subject to perform the sutures. Ideally, the subject must not touch the outer case, which is symbolic of surrounding open tissue and other body parts. In the old design, the outer casing was a part of the construction that was placed on top of the force membrane [17]. Due to this, every time the subject touched the

outer casing by accident, large forces were observed. To eliminate this noise so as to obtain forces only coming from the membrane, the new design has been incorporated. The new membrane housing does not include the outer case and therefore weighs around half of what the old setup weighed, thus further reducing the impact of the membrane housing on the forces read by the force sensor. The old design has large noises in terms of settling force and torque readings. The readings take time to settle after a sharp rise in force or torque readings and this can be observed from forces applied on the membrane as well as the outer case. The new design thus eliminates the forces from the outer case and any noise that it may contribute to in the force and torque readings.

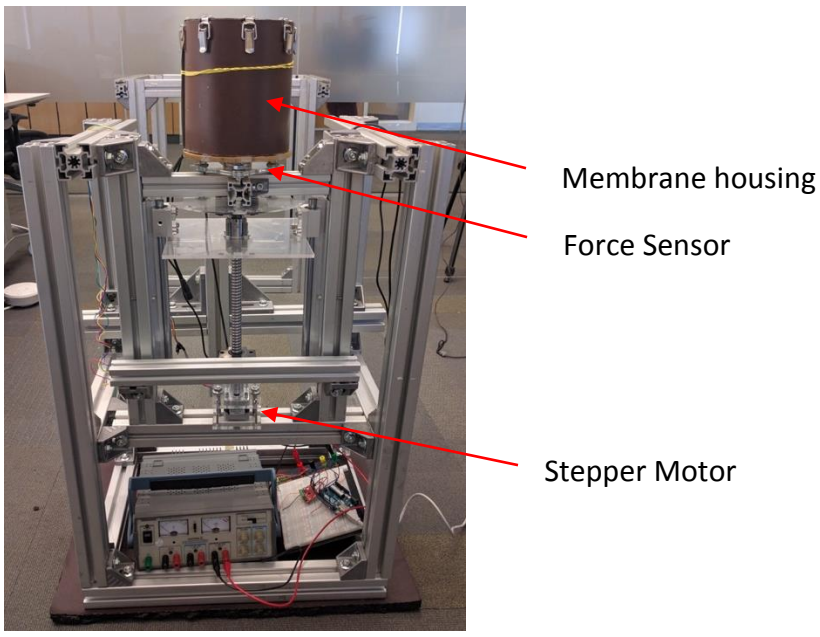

Figure 11: Stepper motor setup

- **Internal Camera**

The new design incorporates a new camera at the bottom of the membrane to view entry and exit of the needle as well as the thread once a suture has been performed. Since this camera is stationary and is enclosed to prevent any external movement or noise from being recorded, it is used to process the images obtained frame by frame and the processed images are stored as a video. In order to view the entire membrane in the frame, the camera is placed at a distance of around 13 cm from the membrane. Due to this distance, the setup that houses the membrane had to be modified to adjust for the addition of the camera.

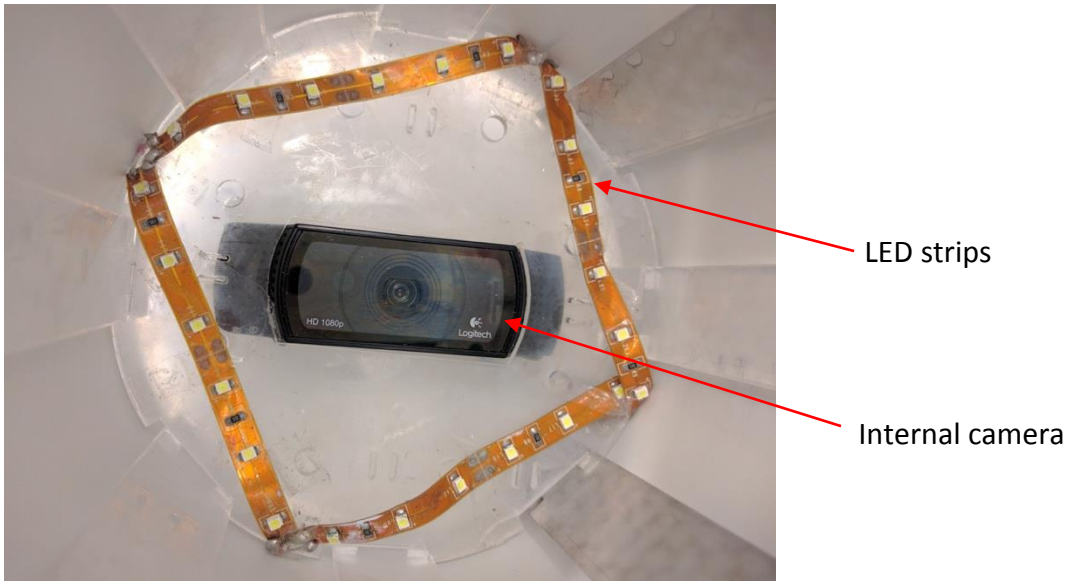

Figure 12: Internal camera and LED lighting setup

The entire setup is enclosed to prevent external light from affecting the frames obtained by the camera. To provide sufficient lighting, LEDs are taped along the base that houses the camera (as shown in Figure 12). The lighting underneath the membrane

plays a key role in the image processing algorithms used. This is the only light that the camera can see, as it is bright enough to overpower the external light coming through the membrane from the outside lights as well as from natural lighting. This internal lighting also helps to create a consistent lighting scheme for every frame read by the camera. The threshold values are set to a particular value based on the lighting system used on base of the camera housing. Any additional lighting is considered as “noise” and can potentially disrupt the threshold values and consequently the image processing algorithms. To prevent this possibility, the acrylic piece that holds the membrane is painted black so as to prevent any light from entering the membrane housing and affect the video capture of the internal camera.

- **External Camera**

The external camera is placed outside the membrane such that the entire membrane, outer casing and hand motion can be observed for future analysis. It helps obtain vital information about unexpected noises and disturbances that might be experienced during the experiment. The video playback can help isolate these noises during data analysis. The camera must be placed at the same level as the membrane and hand position in order to capture events like the approximate point and time of needle entry/exit, hand motion-profile and any disturbances or unexpected noises that may occur during the course of the experiment. A webcam is mounted on a tripod and placed in front of the entire setup. It is positioned in front of the subject so that the front view of the hand motion-profile is captured. This position provides the best view

since it almost never blocked by the subject, since the subject is expected to perform the entire experiment from the opposite end of the platform. This camera can be seen in Figure 2, seen viewing the membrane housing.

- **Aluminum Frame**

The entire framework of the device is built using Bosch Rexroth Aluminum frames. It consists of two aluminum frames, an inner frame and an outer frame. The two frames are positioned such that they never touch each other. The functionality of each framework is detailed below.

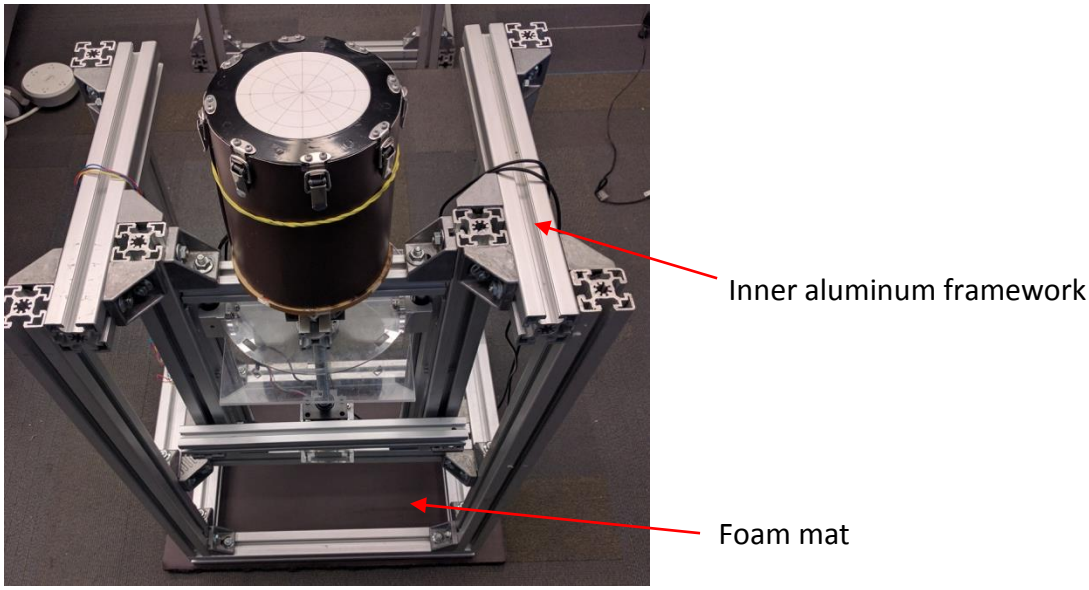

Figure 13: Inner framework

The inner frame is constructed to the shape of a stable cuboidal structure, in the center of which exists the stepper motor and rotating column that allows for the movement of the outer case (as shown in Figure 13). There are two main aims for the construction of the framework. The first is to make the entire device portable and

standalone. This means that the device no longer has to depend on a sturdy, immovable table to be placed on top of. The entire device can be transported easily as one piece and will be ready for use once plugged in to the computer. The second reason is to house the stepper motor for electronically moving the outer case to the desired height. Originally, the membrane housing had to be detached from the outer case and readjusted to the desired height. An Arduino microcontroller powers and commands the stepper motor to move the outer case to the desired height. The entire framework rests on a foam mat that helps dampen any disturbances in the device or framework. The pad helps reduce the settling time of force impulses by up to 150 milliseconds.

Figure 14: Outer framework

The drawback of the aluminum framework is that as the amount of metal used in the framework increases, the amount of vibration produced due to any contact with the system increases. These vibrations cause noise in the force sensor and greatly affect the

reading of the sensor. To overcome this problem, an outer framework is designed (as shown in Figure 14). This outer aluminum framework acts as a barrier between any external contact or unintentional touch by the subject and the inner framework. Due to this, the inner framework and hence the force sensor is isolated from any noise. The cardboard sheet placed on top of the framework is also made to lie on top of the external framework and not the internal framework.

3.2.2. Sensors

- **Force Sensor**

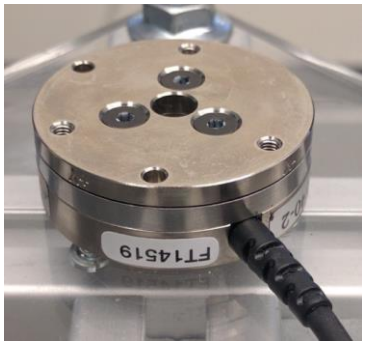

Figure 15: Force Sensor ATI Mini40 [32]

The force Sensor is an ATI Mini 40 6 axis force - torque sensor (as shown in Figure 15) [32] that is interfaced with the CPU using an M - Series National Instruments Data Acquisition (NI-DAQ) system. This sensor is used to measure any force that is observed on the membrane and surrounding areas. This DAQ has a C++ software Development Kit (SDK) that can be run on Microsoft Visual Studio [34]. The force sensor is set to a sampling frequency of 1KHz and is used to measure forces and torques, both positive and negative, in the x, y and z directions.

The DAQ used to interface the force sensor is an expensive data acquisition card and is only compatible with PCI slots in CPUs. Therefore, a CPU with PCI compatibility has to be used. Most stock CPUs today are manufactured with PCI-Express slots and not PCI slots. The computer used for the old version of the platform is used again due to the DAQ PCI compatibility issue.

- **Inertial Measurement Unit**

Figure 16: InterSense InertiaCube 4 [33]

The Inertial Measurement Unit (IMU) used is an InterSense InertiaCube 4, 3DOF IMU (as shown in Figure 16) [33]. It provides readings of the subject's hand and wrist motion while performing the experiment. The IMU is run using InterSense's C SDK, which is converted to C++ code compatible with a C++ project solution in Visual studio [34]. The IMU has a maximum sampling frequency of 200Hz [33]. The hand motion during the experiment is captured by the inertia cube, which can be seen in the series of figures in Figure 1.

- **Cameras**
    - **Internal Camera**

The internal camera used is a Logitech C920 HD Webcam. It has a 15 Megapixel camera with 1080p HD, 30 frames per second (FPS) video recording capability [37]. It's flat rectangular shape makes it very convenient to mount on the acrylic base designed especially for this camera. It does not require any special driver to run the camera in a software application like Visual Studio and is compatible with OpenCV [34] [38]. The internal camera is used to process the video using various image processing algorithms. The algorithms are used to provide vital information regarding the needle's interaction with the membrane, like the frame in which the needle entered or exited the membrane, the trace of the needle tip and so on. This image processing can be implemented in real time while the subject performs the experiment or it can be implemented as post-experiment processing, where the image processing is performed based on a video recording of the entire experiment.

- **External Camera**

An external camera is also used to record hand motions and other details about the experiment that may have been missed by the other devices. These details can include disturbances in the device (like touching on the outer case, touching the aluminum frame et al.), breaks taken by the subject, variations in external lighting that might affect the internal camera's video quality and so on. At first, an HP Pro webcam was used to

perform the experiments [39]. However, it was later found that under certain lighting conditions, the camera did not provide good quality video due to increased lighting in the room, which was found to be due to a high Exposure and Gain setting in the camera. Nonetheless, by the time a solution was found for varying the exposure, a much better camera was found in the lab, which was used for all experiments henceforth. This new camera is a replica of the internal camera, a Logitech C920 HD camera [37]. The external camera is not used for any image processing application and is purely used to watch the subject perform the experiment. It only views the membrane and the subject's hand movements during the experiment.

3.2.3. Software Algorithm

The force sensor, IMU and two cameras are required to run in parallel to ensure data synchronization. The system level diagram shown in Figure 17 below represents the general program flow of the system. In order to run all sensors in parallel, multithreading has been implemented.

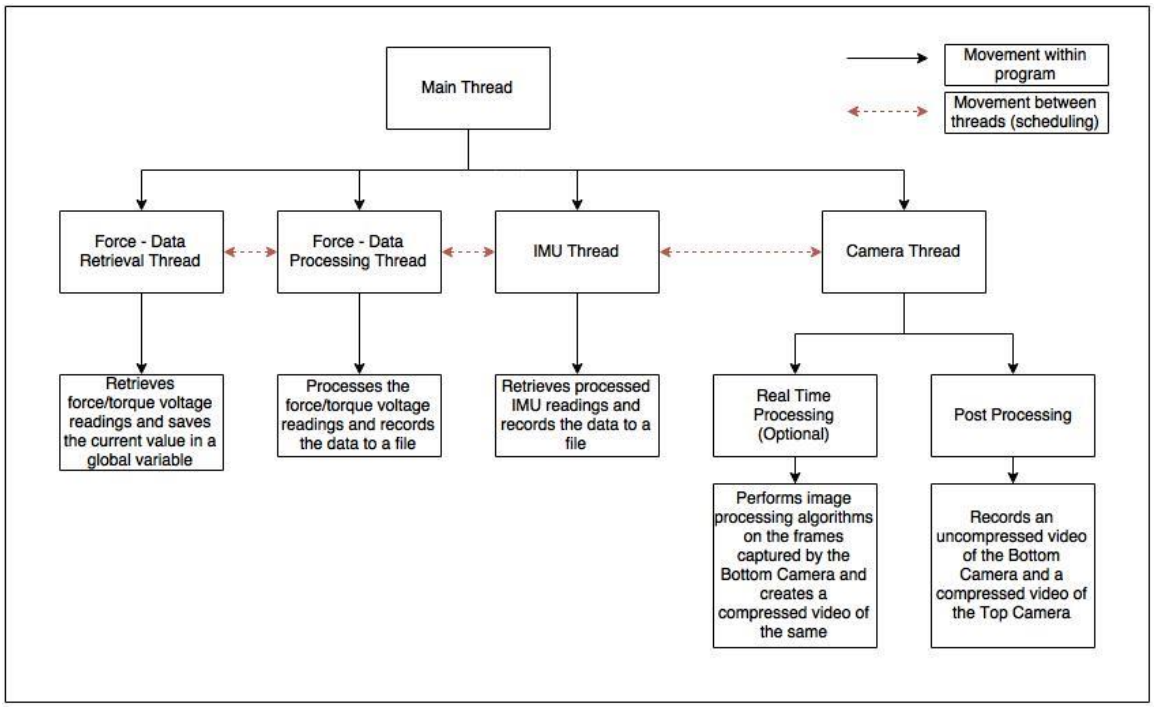

Figure 17: General program flow of the C++ program

The force sensor, IMU and camera have been interfaced using C++ on Visual Studio 2013 while the processing of the data has been implemented on MATLAB. Microsoft Visual Studio (2013) is used as the Integrated Development Environment (IDE) and the entire software program is written and executed in C++. A Project solution is created in Visual Studio 2013 and is customized based on the various requirements for each of the sensors. The software algorithm consists of the combination of four sensors; namely a force sensor, an IMU and 2 cameras. These 4 sensors are manufactured by different companies and have different setups and varying methods of integration with the C++ project. [34]

3.2.3.1. Setting up the project

The force sensor requires the NIDAQmx C++ library to access various functions used for running the sensor in C++. This is set up in the project properties by including the required Dynamic-Link Libraries (DLLs) and library files obtained from the National Instruments software download page for the data acquisition device [40]. The DAQ uses an internal clock to record timestamps, which is only accessible to National Instruments software.

The IMU is run using a sample code provided by InterSense [33]. Here too, the IMU uses an internal clock to timestamp the data, but this clock is not available to the user apart from when the IMU readings are read.

The two cameras capture videos using OpenCV 3.0.0, which was found to be the most compatible version with maximum functionality [38]. Further details regarding each of the sensor's software algorithms are detailed below.

In order to have synchronized data across all the sensors, a common clock must be used. As mentioned before, the DAQ uses an internal clock for setting up the sampling frequency of the force sensor. The IMU uses an internal clock to sample the frequency of the IMU, but is based on the CPU clock time that the program uses. Even then, the offset due to the internal clock can cause a shift in the phase of the sampling function used. Therefore, a standard clock needs to be used to ensure synchronization of the data. The internal clock of the CPU is used as the standard clock since Microsoft Visual Studio uses the same clock to time the C++ program [34]. The function ftime is

used to measure the number of milliseconds elapsed since the start of the data program. While reading data, all sensors are referenced to this millisecond timestamp to ensure synchronization of the sensors.

The key project property modifications are as follows:

- The entire program runs in Win32 configuration instead of x64. If supported by the CPU, x64 configuration may also be used.
- Both Release and Debug mode work, but release mode is used during data collection and debug mode is only used for testing and debugging purposes. It is found that the Release mode reduces the compilation time of the program significantly and performs more efficiently than the Debug mode since the Release mode is set to compile in an optimized manner in the project properties. This means that many of the debug symbols are not stored for the Release mode.
- The Run-time library is set to Multithreaded (/MT) for Release mode and Debug Multithreaded (/MTd) for Debug mode since the program implements multithreading. Failure to make this modification will result in a run time error.

Various inputs are incorporated into the system at the start of the program. The user is asked to choose between real time processing and post processing of the video frames for computer vision and providing the name of the file to distinguish data between subjects. Actual names of subjects are not used, following a standard format of "Subject" + a unique number in ascending order.

3.2.3.2. Force Sensor

As previously mentioned, the force sensor is used to measure forces and torques, both positive and negative, in the x, y and z directions, set up to sample data at a frequency of 1KHz. The NIDAQmx library provides a number of inbuilt functions that can be used to read data from the data acquisition system. For this application, continuous acquisition with digital start configuration is used to acquire voltage readings from the DAQ. The program flow for the force sensor threads is shown in Figure 18 below. Multiple threads are used to acquire and process the force data. This is due to the access restrictions within the NIDAQmx library where National Instruments has secured the data acquisition algorithm within a DLL, which the user does not have access to.

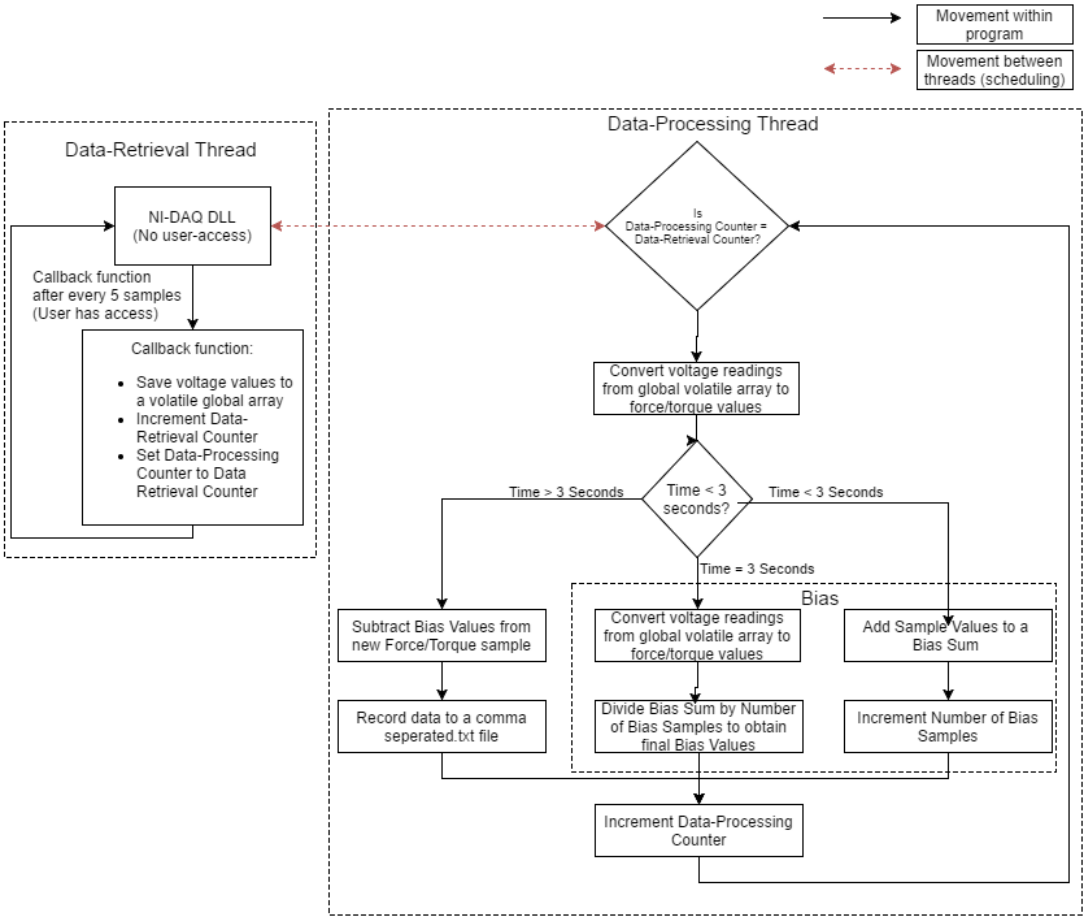

Figure 18: Program flow for force threads

The force data provides the DAQ with voltage readings, which is in turn read by this continuous acquisition program. A calibration matrix is provided by ATI to convert voltage readings into force value in Newton. The calibration matrix is used and various calculations are performed on these voltage readings to obtain the force in Newton. When the function DAQmxRegisterEveryNSamplesEvent (referred to as EveryNSamplesEvent) is called, some code runs in the DLL provided by National Instruments which in turn sets up a callback function, EveryNCallback that is called after N samples have been received by the buffer. Once the EveryNSamplesEvent function is

called, the sensor is said to have begun the process of transmitting force data to the program through the DAQ. The function blocks provided by the SDK only allows the user to access the data after N samples have been registered by the sensor, in the EveryNCallback function. As Mentioned before, the NI DAQ uses an internal clock to adjust the frequency of samples and this clock is not accessible by other software. Therefore, a common timestamp is used to time all the sensors. The first instance to register a timestamp for the force sensor is in the EveryNCallback function of the NI-DAQ SDK, which is called immediately after N samples have been registered by the DAQ.

Since the IMU can provide readings at a maximum rate of 200Hz, the force sensor is set to provide data after 5 samples have been collected. This gives a timestamp at every 5^{th} millisecond, thus mimicking a 200Hz sample period. This does not restrict the force sensor to a sampling rate of 200Hz, but merely provides the user with every 5 samples as a packet, still sampling at 1Khz. Since the first timestamp that is available to the program is after these 5 samples have been registered, it is assumed that the 5 samples are obtained 1 millisecond apart, starting from 5 milliseconds before the registered timestamp (Since the frequency is set to 1000 samples/second). This can be seen in Table 1 in the getMilliSpanatReading and individualTime columns. The getMilliSpanAtReading is only updated every 5 milliseconds, but it is assumed that every 5 samples that are obtained from the buffer are received exactly 1 millisecond apart. Thus the individualTime is assigned starting at 5 milliseconds before the getMilliSpanAtReading timestamp, each allocated 1 millisecond apart. The timestamp

can be allocated to the samples received at 2 instances. One is right after the callback function has been triggered by the DAQ DLL, while the other is during file writing. Even though the program executes at a very fast pace, it cannot keep up with a 1 millisecond requirement. This can be seen perfectly in Table 1, where the getMillliSpanAtWriting is the timestamp obtained while writing the force values into a file, while getMilliSpanAtReading is the timestamp obtained at the exact time when data samples were made available, in the callback function. One can observe that getMilliSpanAtWriting provides an approximate timestamp, always in multiples of 5. However, as discussed, this may or may not be the case and a more precise timestamp is used for recording the IMU data.

The force and torque data are received as voltages. A calibration matrix is used to convert these readings into meaningful force and torque values, measured in Newton and Newton-Meter respectively. The samples recorded in the first 3000 milliseconds are used to calibrate the sensor and calculate the offsets. After 3000 milliseconds, the offset values are set and these are subtracted from all force values obtained throughout the remainder of the experiment. The sensor needs to be recalibrated at the start of each experiment.

- **Different threads for receiving and processing force data**

As mentioned previously, the only access to the force sensor for data access (and hence, manipulation) provided by the DAQ SDK is in the EveryNCallback function. This function is called in the DLL where the EveryNSamplesEvent function is defined, which

the user does not have access to. The EveryNCallback function needs to return to the DLL as soon as possible to start timing the next call to the callback function after the next 5 samples have been received in the buffer. The processing of the force data received as voltages and the process of converting them to force values, subtracting offsets and recording the data to a file takes up hundreds of microseconds. Due to this, if all the processing takes place in the same thread, it takes much longer for the force sensor program to return to the DLL to start timing the next 5 samples. This causes a delay every time a sample packet is read and eventually results in a buffer overflow, raising an error in the system. The only solution to this is to either lower the frequency of the sensor to allow for the delay in processing the data, or to process the data in a separate thread altogether. Instead of reducing the frequency of data, a separate thread is used to process the force data and the EveryNCallback function is made to run with as few instructions as possible, containing only vital counters and timestamps that must be placed in the callback function.

- **Varying processing speed of the two force threads**

The EveryNCallback function is set to be called after 5 new samples are read into the buffer, which is approximately 5 milliseconds since the sensor samples at a 1KHz frequency. This means that the force samples are received only once every 5 milliseconds, while the force data processing thread runs at a much faster rate, considering it does not have a sample rate and is only influenced by the processor speed for performing calculations and writing the data into a file. It is observed that the

processing thread runs around 7-8 times between successive calls to the EveryNCallback function. Due to the increased speed of the processing thread, if a check isn't in place to ensure new data has been read, the same samples will be used and repeated until new data arrives. Therefore, to overcome this issue, multiple counters are set and incremented in both threads to ensure that new samples are arriving and that the two threads are in sync with each other. There is also a fail-safe in place to correct any instances of the two threads going out of sync, as the threads are re-assigned to new, incremented counter values every time the everyNCallback function is called.

- **Repeated timestamps for unique force samples**

Due to the usage of a CPU clock timestamp instead of the DAQ clock, sometimes the data packet is received a few milliseconds late or early. When this situation occurs and the next packet is received at the expected time, multiple data is obtained for the same timestamp. One can never know which sample contains the correct timestamp and which sample has been shifted. Therefore, when this condition occurs, the average of the two samples is considered and assigned to the common timestamp. Thus after processing the repeated timestamp samples, a force data set is obtained which contains only unique samples with unique timestamps. A sample set of values obtained by the force sensor are shown below in Table 1.

getMilliSpanAtWriting	getMilliSpanAtReading	individualTime
11385	11385	11381
11385	11385	11382
11385	11385	11383
11385	11385	11384
11385	11385	11385
11390	11389	11385
11390	11389	11386
11390	11389	11387
11390	11389	11388
11390	11389	11389
11395	11395	11391
11395	11395	11392
11395	11395	11393
11395	11395	11394
11395	11395	11395

Table 1: Sample force sensor timestamps

In this example above, 5 samples at 11385 (getMilliSpanAtReading timestamp) were processed at the correct time, but the next 5 samples, 11886-11390 ended up being processed at 11389 instead of 11390. Therefore, the first value is assumed to have come early and will contain the same individual timestamp as the last sample of the previous 5 sample data packet. In this case, the 1385th individualTime timestamp will therefore have 2 unique force and torque readings for the same timestamp (marked in red in Table 1). To solve this issue, the average of the two readings is taken as one can never know whether the reading was fast/slow to reach the CPU or the program had to run other commands before attending to the force sensor.

3.2.3.3. Inertial Measurement Unit

The software algorithm is modified based on the source code provided by InterSense [33] for programming the IMU. Different keystrokes perform different actions like starting the sensors; logging the data from the time the log key is presses ('l'), stopping the application. Various outputs like 3 axis Gyroscope, accelerometer and magnetometer data is recorded, using the same timestamp. The IMU runs as a separate thread and records at a frequency of 200 Hz. The IMU is originally calibrated based on a default vertical position. It can be reset and recalibrated by pressing 'r', which is done after the subject places their hand on the membrane prior to starting the experiment. The IMU, like the force sensor, first receives the samples within the DLL of the InterSense SDK [33], which the user does not have access to. The first instance where the user has access to samples is right after the function for reading the samples is called. This may cause a delay of a few microseconds, but it is assumed that this is negligible and that registering a timestamp at the end of the function does not cause a delay in the timestamp allocated to the particular sample.

The data begins logging after the 'l' key is hit on the keyboard, informing all the threads that the user has requested to begin logging data. The IMU then waits for the 'q' key to be hit, which represents the end of the experiment. At this time, the IMU thread resets flags for the other sensors threads, thus informing them that the experiment has concluded and that the threads may be stopped.

3.2.3.4. Cameras

Two cameras are used in the system, one internal camera that records the needle and thread movement from underneath the membrane and one external camera that records hand motion and any disturbances that might occur during the experiment. Only the internal camera is used for image processing while the external camera is used purely for recording and playback purposes. The internal camera is a Logitech C920 HD webcam. Some of the experiments were recorded with an HP Pro webcam [39]for the external camera, but this was later replaced by another Logitech C920 HD webcam for better quality video capture [37].

- **OpenCV**

The internal camera is interfaced with the system using OpenCV. For this project, OpenCV 3.0.0 is used and all the DLLs and Library files are linked through the project properties page [38]. The program provides an option to the user to run real time processing of the video or to opt to process the video separately after having completed the experiment. This is implemented since it is unclear as to who would be present at the time of experimentation to review real time data and whether it would be possible to monitor the progress as the subject performs the experiment. The real time processing has been incorporated and provided as an option in lieu of the eventual need for real time analysis of data as the subject performs the experiment.

- **Issues with real-time processing**

Real-time processing has been incorporated as an option at the beginning of the program. It allows the user to choose whether they want the video recording to be processed in real-time or if they would like to process the video at a later time after the completion of the experiment. There are various pros and cons to the real-time processing option. The main pro is that the analyst, user or subject can have a look at the subject's performance in real-time, as they perform the experiment.

However, the real-time processing has a few drawbacks. The main drawback is that the algorithms used in processing the frames consume a lot of time. Therefore, when run in real-time, the video processing thread takes up a lot of the RAM and processor scheduling. This leads to lower efficiency and output rate of the other sensors as well as the camera frame rates. The cameras record at 10-25 frames per second (fps) as compared to a consistent 30 fps frame rate obtained when only post processing is performed. The real time processing also causes the other sensor threads to slow down. When run in post-processing mode, the force sensor has an average data loss rate of 0.1% - 0.2% due to timestamp discrepancies. However, when run in real-time processing mode, this data loss rate climbs to 1% - 2%, which is too much of data loss.

To prevent the loss of data and to maintain a consistent, high frame rate, post processing is chosen for the experiments over real-time processing. In subsequent versions of the Suture Platform, the program can be optimized to efficiently run the

real-time processing mode in an embedded RTOS environment, possible leading to consistent and high frame rates and low data loss percentages for the other sensors

3.2.3.5. Combining all the sensors

Each sensor is made to run individually and they all run at their respective speeds with good consistency and speed. The main problem faced is when all three sensors are made to work in unison and made to provide synchronized data. If the entire program is run as one thread, the program only executes one function at any given period of time. Due to this, all the other sensors must wait for the current sensor being read to finish executing, only after which the program will move to the next sensor's function. This will result in the sensors never having synchronized data and will also result in buffer overflows in the sensors will be waiting for the program to execute them, while accumulating data in the buffer. To prevent this from happening, multithreading is executed. Each sensor is run as a separate thread so that at any given instance, all of the sensor functions are running simultaneously. This results in the CPU never running one sensor alone, but processing all sensors simultaneously, thus executing one line at a time from each thread. Though the processor will execute only one line at a time and though it may be from any of the sensor functions, the other sensor functions are also in the queue for being executed soon (within tens of microseconds). Therefore, neither can one know which thread is going to execute nor can one assign a particular thread to run. Although it is theoretically possible to schedule threads to the processor, one cannot afford to have a given thread run entirely while the other sensor threads wait for

it to complete. This would eventually cause a buffer overflow due to the wait and create an error in the system. Therefore, thread scheduling has not been used and the CPU is allowed to schedule the different threads in random order, knowing that all the threads do run simultaneously. Due to the random multithreading scheduling, sometimes the force sensor thread might not be executed for tens of milliseconds.

Multiple flags are used to control the start and end of threads for each sensor. There are 4-5 threads (depending on whether real-time processing or post processing of the video is chosen) that are run simultaneously.

3.2.3.6. Synchronized Platform

The only method of knowing if a given data sample from the force sensor corresponds to a given sample from the IMU and camera frame is to use the same clock and timestamp. A function to calculate the time elapsed in milliseconds since the start of the program is used as the millisecond timestamp calculator. It uses the predefined ftime function from the sys/timeb library to calculate the elapsed time in milliseconds. Other accurate clock functions may also be used. Originally, the sleep function from the windows.h library was used to check the accuracy of the ftime function, but it was later found that the sleep function is not accurate below 10 millisecond intervals. A timing diagram in Figure 19 below shows the general sample availability at a given timestamp.

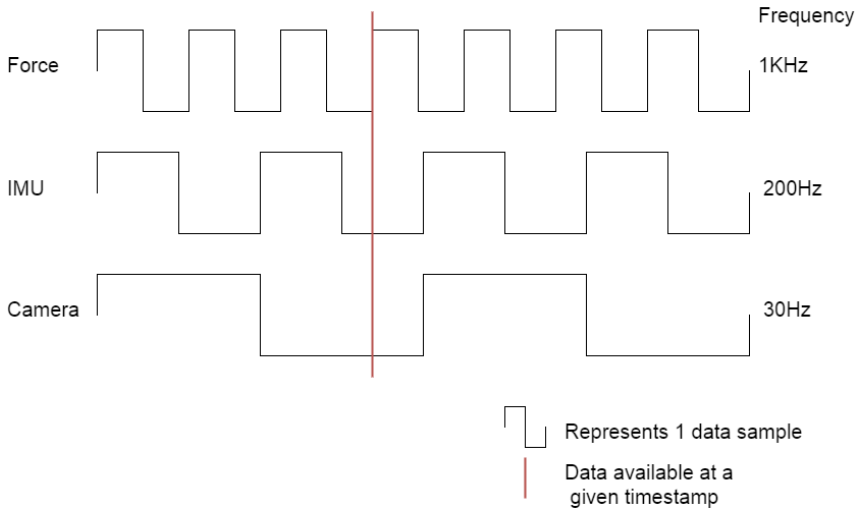

Figure 19: Timing Diagram of all 3 sensors

3.2.4. Data Collection

- **Participant pool**

The subject pool for collecting data is divided into various categories. The first category of data is Surgeon data, which acts as the benchmark for novices and intermediates to achieve in order to improve their suturing expertise. It is very difficult to get surgeons to devote some of their valuable time to provide data for analysis. The second category is that of intermediate subjects, comprising of residents and students experienced in suturing. The third category is that of amateur students who have been taught basic suturing lessons and may even possess some basic suturing experience on animal tissues or similar simulated surgical platforms. The last category consists of novices who have little to no experience in suturing. Institutional Review Board (IRB) approval was sought prior to the start of the data collection process. This is a mandatory requirement as the participant pool involves human subjects as part of the research.

At the time of writing, data has been collected from 15 subjects, consisting of students at MedEx Academy, an internship program at the University of South Carolina (USC) School of Medicine - Greenville. The students are all pre-medical school students who have had less than 10 hours of any kind of suturing experience. These students are all classified in the “Novice” category of suturing expertise. Towards the end of the data collection, the aim is to conduct experiments to collect data from all the other skill level categories as mentioned previously. This includes residents and fellows in the “intermediate” category and surgeons in the “Expert” category.

- **Procedure**

When the subject enters the room, he/she is requested to fill out a basic questionnaire pertaining to basic questions about themselves and some questions regarding their suturing skill and fine motor skills. All subjects are handed a consent form as well as an experiment guideline describing the task to be performed. If the subject is classified as a “Novice”, he/she is also provided with a write-up describing the basic procedure for performing a suture.

Prior to the start of each experiment, various checks must be made to ensure consistency in data collection. First, the internal camera is set to the 15MP photo capture setting. This is necessary to ensure video capture occurs at 30 fps. Different megapixel values can tremendously affect the fps of the video, sometimes recording at 5 fps. The camera’s autofocus is turned off as the height of the membrane housing has been established for a particular focus level. Turning on the autofocus feature will cause

the camera to re-focus when sutures are performed, quite often focusing on the wrong object, thus essentially capturing frames that are out of focus. The program is set to Win32 - Release mode configuration and all other applications on the computer are closed to ensure maximum RAM usage by the program.

Each subject is handed a new needle and thread for performing the experiment. The membrane used to perform the previous experiment is replaced with a new, unused membrane at the start of each experiment. This standard is adopted to ensure that all the forces measured are only due to the subject's performance and no data is lost due to existing holes from previously used membranes. The subject is also provided with a needle driver to hold the needle and a glove to be worn on the dominant hand (The hand used to perform the experiment). All subjects are allowed to suture on a spare membrane two times, to become familiar with the membrane material and tautness of the membrane.

The subject is told to start from a particular point on the membrane and perform the experiment in a counter clockwise manner to ensure consistency in the experiment and for ease of analysis. The membrane consists of 2 concentric circles, through which 6 equidistant lines exist, all intersecting at the center of the membrane. The subject is required to suture along the intersection of these lines and circles. The suture is started from the outer circle intersection and is completed at the same line's inner circle intersection. All subjects are asked to begin the experiment from the same intersection point. 12 sutures are performed in total, beginning from the first suture and moving in a

counter - clockwise manner. At the end of the experiment, the subject informs the user that he/she has completed the experiment and the program is exited. During the experiment, one of the researchers observes the various movements made by the subject and looks for any abnormalities that may occur during the course of the experiment. These can be anything as simple as touching the outer casing or disturbing the setup with their foot or experimental issues like breaking of the needle, unintentionally forgetting to suture some points and so on.

Prior to the start of the experiment, the IMU is placed on top of the glove using a Velcro patch. Before the subject begins suturing on the membrane, he/she is asked to place their dominant hand on top of the membrane in order to calibrate the IMU with respect to the membrane orientation by pressing the “r” key on the keyboard. Once the IMU has been calibrated with respect to the membrane, the force sensor and camera threads are started in the program by pressing the “s” key. When the threads begin, the force sensor is calibrated for 3 seconds and the sensor offsets are calculated. When the subject is ready to begin, the “l” key is pressed to begin logging. At this time, the terminal window and the two camera frame windows are separated for easy viewing during the experiment. Only after the windows have been separated, the subject is informed that they may begin the experiment. This is an important step because when the camera frame windows are separated, they tend to pause for the duration of the separation process, leading to a loss of frames over that time frame. Therefore, the subject must be asked to perform the experiment only after the windows have been

separated and not before. When the subject has completed the experiment, the “q” key is pressed to stop logging and end the program.

- **Collected Data**

Suturing data has been collected over a span of 2 weeks with, resulting in a data set of 15 subjects. During the course of the first 3 days, various observations were made regarding the suturing practices of the students. These observations were related to practices that were not expected of a subject, but were performed nonetheless. Some examples are touching the outer case, using the outer case as a pivot to perform a suture, applying heavy lateral forces on the outer case and so on. These errors resulted in various misrepresentations of data as the platform was not built for such practices. There were also some technical issues experienced during the initial days like not being able to obtain an optimum light setting in the room to be able to view the external camera clearly. However, as each day progressed, the observations led to minor modifications in the design that ultimately resulted in a fool-proof system that can withstand reasonable unwanted force without causing a disturbance in the data collected by the system.

- **Discarded data**

Over the course of the data collection, the modifications made to the system resulted in non-uniformity in data collection. On the first day, only one subject’s data was collected. The subject was ever so often in contact with both the outer case and the platform table top, which the subject was asked to be conscious about to not come in

contact with the outer case. Due to these disturbances, considerable noise was observed in the force readings. To overcome these disturbances, a new outer aluminum framework was designed that would hold the table top as well as prevent the subject from coming in contact with the inner force sensor framework or the foam mat. This modification resulted in all the external forces becoming negligible, improving the efficiency of the data collected. Due to the modification, the subject's data could not be used as a part of the final data set.

Similarly, on the second day of data collection, it was observed that the lighting in the room affected the quality of images recorded by the external camera. Due to the auto exposure and gain setting, it was difficult to improve the quality of the external camera's video capture. Thus, a new better quality camera was used the following day, one which could capture good quality images even through slight changes in the light coming to the object in focus. Since the video recording from the external camera was only used as a reference during data analysis, the data collected was not discarded and was added to the final data set.

4. Results

4.1. Data Synchronization

The data that has been collected consists of unique, time-stamped data which can be analyzed based on timestamps. Every sample in any sensor can be compared to another sensor's sample at the same timestamp. The force sensor data has been recorded at 1KHz., IMU data at 200Hz and camera recordings at 30 FPS (30Hz). Since the 3 sensors are recorded at different sampling frequencies, the closest sample (depending on the sensor's sampling frequency) to the required timestamp from each sensor can be accessed. For example, if there exists a force sensor sample at the N^{th} millisecond timestamp, there may or may not be an IMU sample corresponding to that timestamp. However, there will be a sample within 5 milliseconds of that force sample since the update rate or frequency of the IMU is 200Hz. Similarly, the two cameras record at 30 FPS and therefore may or may not have a frame corresponding to the N^{th} millisecond timestamp, but there will be a frame within 30 milliseconds of the timestamp.

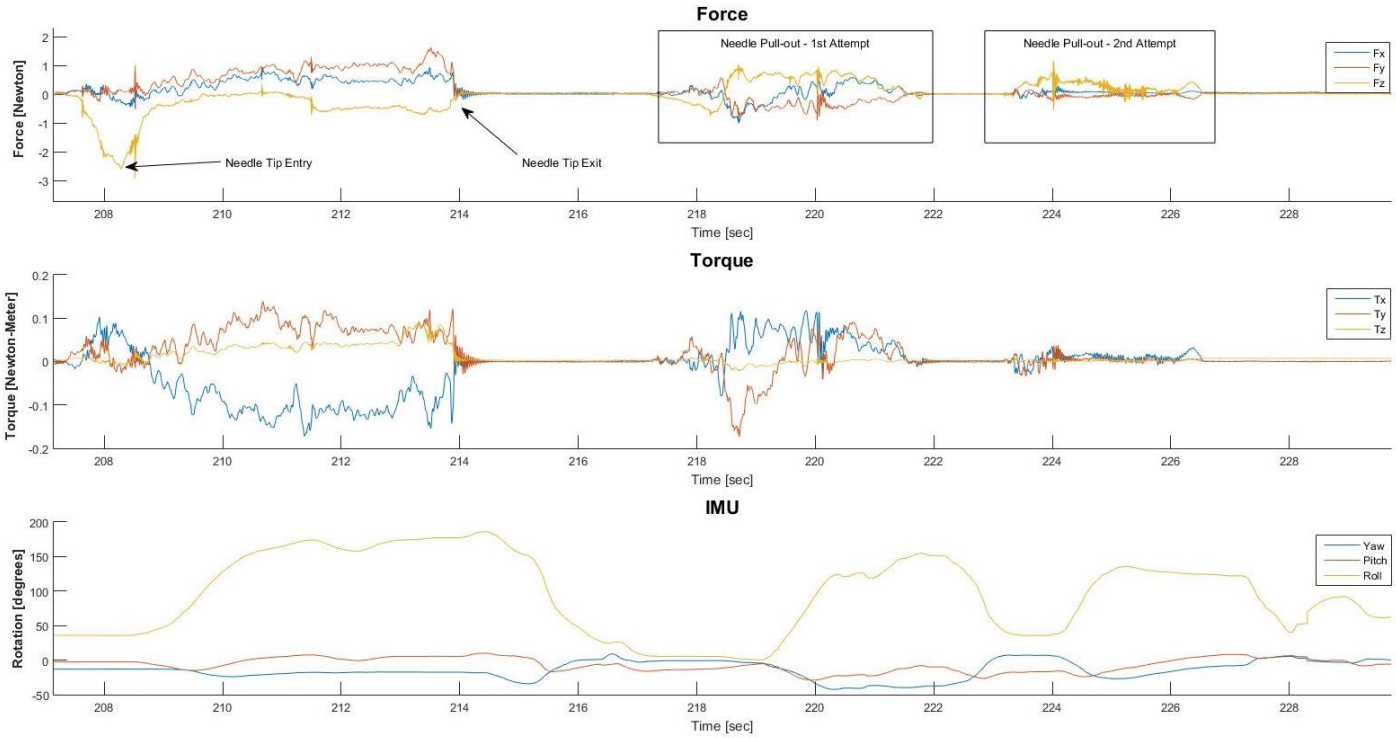

Figure 20: Synchronized Force, Torque and IMU data for a standard suture

Synchronized force, torque and hand-rotation data can be seen in Figure 20 above.

The force and IMU data is found to be synchronized with each other based on the timestamp allotted to each sample while conducting the experiment. The figure represents the various graphs describing one suture. This includes the entry of the needle tip into the membrane, exit of the needle tip out of the membrane, switching the grip of the needle holder and pulling the needle out of the membrane. Distinct force peaks and sudden changes in force and IMU data are seen, representing the different needle tip and needle base entry and exit times. Represents a frame grab from the internal camera at a timestamp after the needle entered the membrane. Figure 20

represents a suture performed by one of the researchers in the project. Enlarged graphs of subject's data can be found in the appendix in Figure A 1 and Figure A 2.

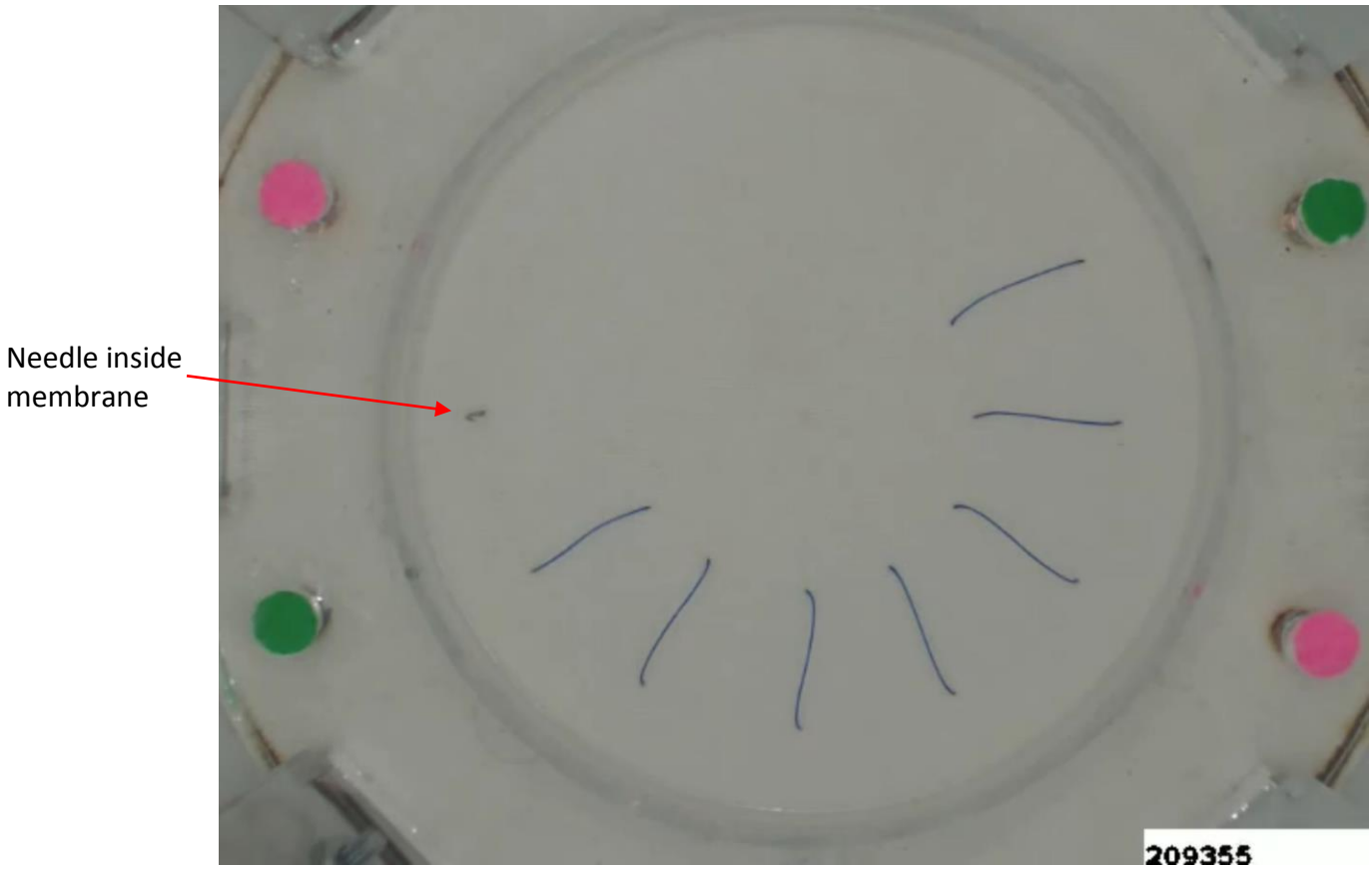

Figure 21: Internal camera frame after needle entry

4.2. Lost and discarded data

As one cannot know which thread is being executed by the processor, sometimes the processor fails to schedule a sensor's thread for tens of milliseconds. Due to this, all the force and IMU data in that timeframe is lost as the data buffer is overwritten. Increasing the data buffer is not an option as there is no way of saving the timestamp in the buffer along with each sample.

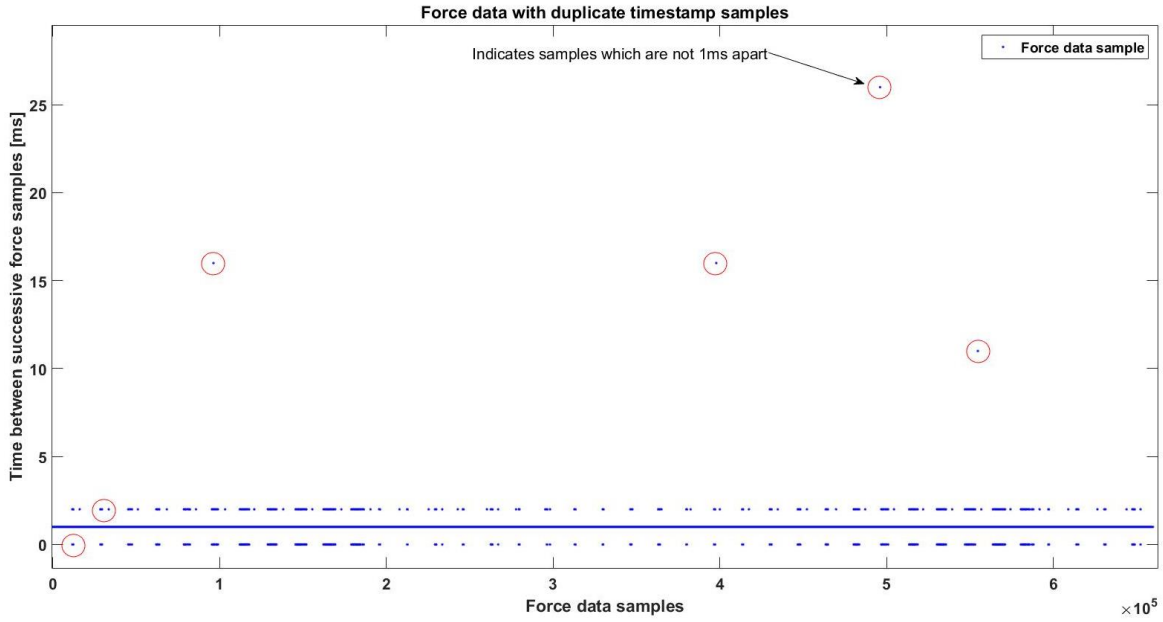

Figure 22: Force data with duplicate-timestamp samples

Figure 22 above shows the time difference between successive force samples. As this figure is zoomed in to show the y axis count close to 0, the higher values are not shown in the figure. However, even time differences of around 350 milliseconds are observed between successive force samples. Due to the issue of repeated timestamps in force readings as shown in Table 1, a number of samples with the same timestamp (i.e., 0 time difference between successive samples) are seen. This is shown in Figure 22. In order to eliminate this, all force samples with duplicate timestamps are averaged and as a result only one force sample is available for every timestamp in the data set. This results in a truly synchronous system where all samples are time-stamped in milliseconds and each timestamp has a corresponding sample from the force sensor, IMU and the two cameras. The camera frames and IMU readings will repeat for a set of force values until a new camera frame or IMU sample is obtained. This result is shown in

Figure 23 below. As one can see, there are no samples having 0 time-difference between successive timestamps, which indicates that all the samples have some time difference between them. Further, we observe that almost all the time differences at 1 millisecond, while there are a few samples out of the entire force sample data set that have time differences of 2 or more milliseconds. For this particular experiment, it is found that out of 773915 force samples originally collected, only 1023 samples had the same timestamp. These samples are averaged to result in a 0.13% data loss.

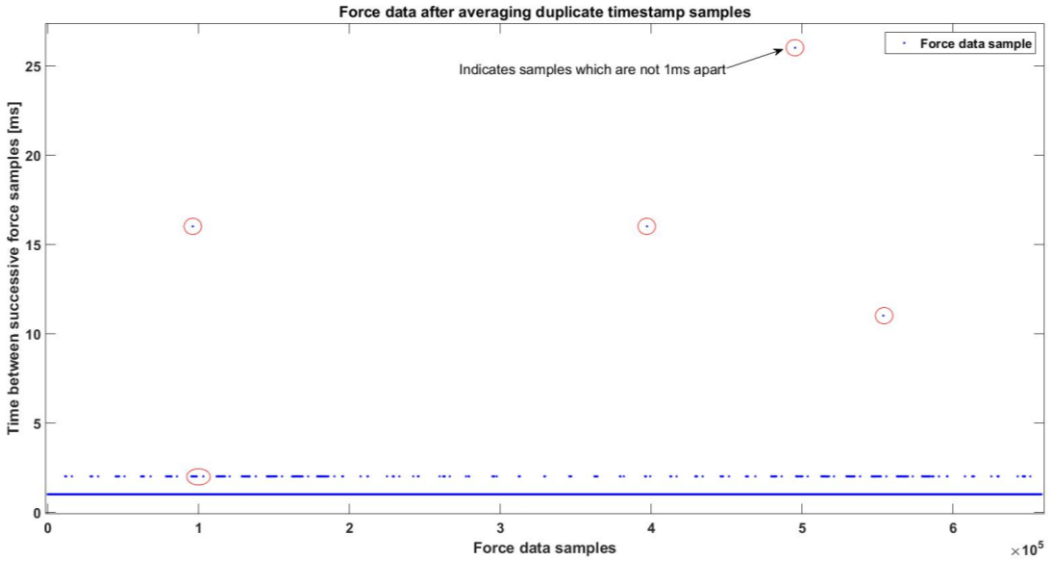

Figure 23: Force data after averaging duplicate timestamp samples

4.3. Noise in the System

The platform was reconstructed to so that the noise in the system would be lesser than that seen in the old version. The new platform shows negligible noise in the force readings for any forces applied on the outer casing as well as the surrounding areas of the platform. This is so because the platform is not physically connected to the outer

casing or the surroundings due to the introduction of the outer aluminum frame. The noise seen when performing a suture was also compared with the results of the old platform. This is shown in Figure 24 and Figure 25. For these particular instances, the settling time of a force impulse in the old version is around 200 milliseconds while the same in the new version of the platform is found to be around 160 milliseconds, thus improving the efficiency by 20%.

It is not known as to why the frequency of the vibration is lower in the new platform as compared to the old platform (as observed in these particular instances), but it may be due to the tautness of the membrane or may be due to the shaking of the entire membrane housing. The height of the housing could be a factor as well.

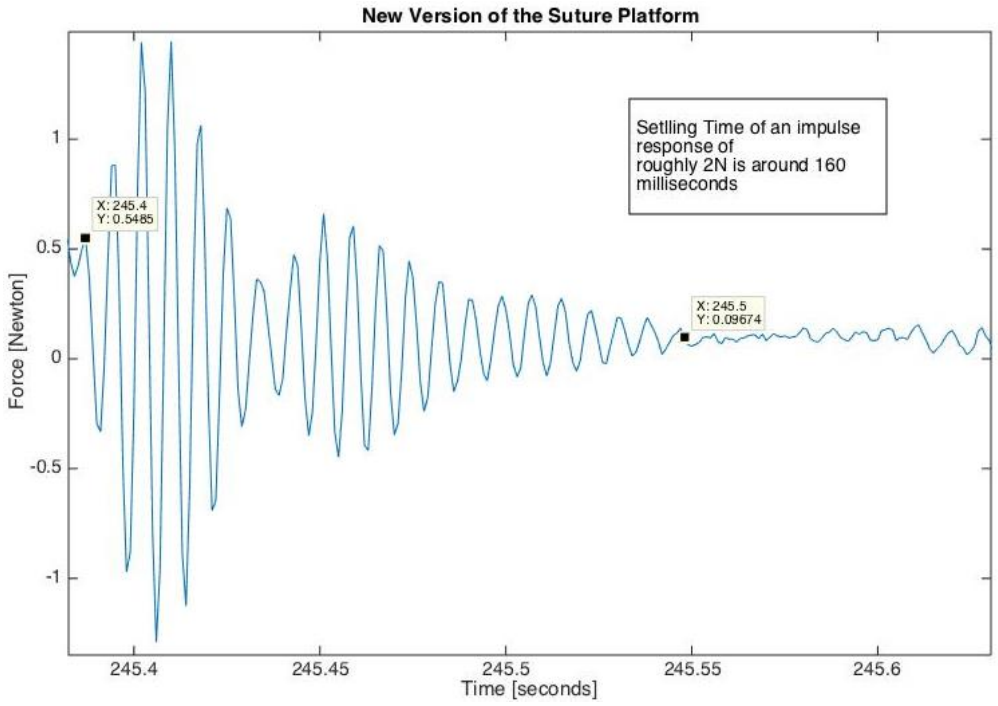

Figure 24: Settling time for a force impulse in the new Suture Platform

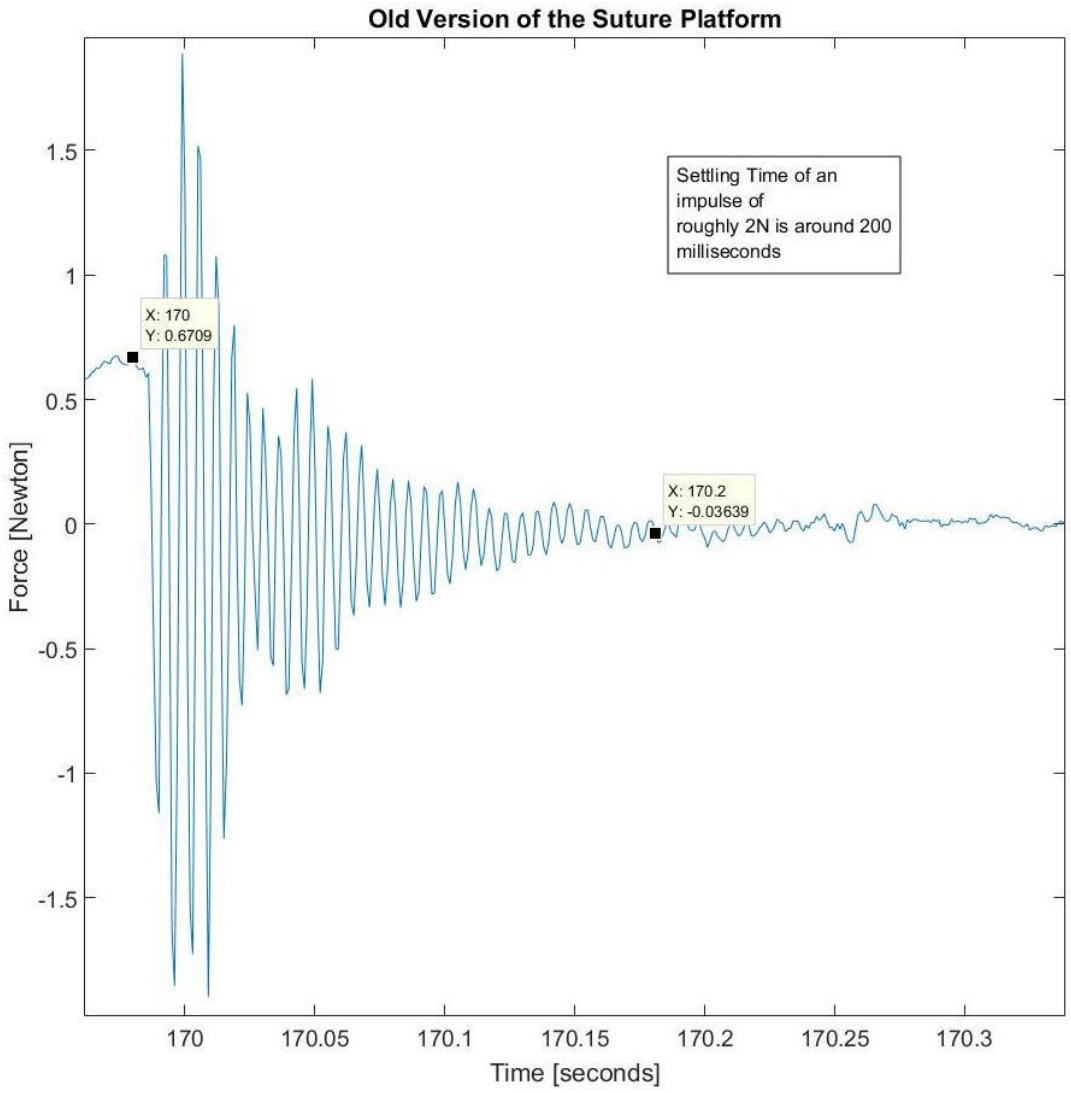

Figure 25:Settling time for a force impulse in the old Suture Platform

5. Future Work

If the project were to receive appropriate funding, the first step would be to buy a USB (Universal Serial Bus) compatible DAQ to interface the force sensor. This would remove the dependency of the system to be run only on CPUs with PCI (Peripheral Component Interconnect) slots. Using a USB interfaced DAQ would also allow the device to be shifted to an RTOS (Real Time Operating System) platform in an embedded device like the Raspberry Pi. This would tremendously lower the space required by the device and it will also allow the embedded processor to focus solely on running the program without worrying about background applications that might consume RAM (Random Access Memory) and slow down the program.

The data collected on the device will be used to ascertain various metrics to classify subjects into various categories of suturing skill level. This would be achieved over many hours of data analysis during which various metrics used in previous research as well as newly hypothesized metrics will be used. This data set will thus become the training set for the machine learning algorithm that will be implemented at a later stage.

The current device costs a lot more than it ideally should due to the use of sensors and other equipment that are readily available in the lab. Should funding be provided for this project, the aim would be to replace the current sensors with much cheaper sensors that provide similar data samples. The goal would be to reduce the cost from the current $5000 - $6000 cost to completing the device within a few hundred dollars.

In terms of settling time of noises while reading force and torque readings, future membranes could consist of less taught membranes that would not have as fierce vibrations as currently seen when a suture is made. A 3 dimensional membrane might also help dampen the vibrations. Newer versions of the membrane could also include torn membranes that the subject would have to stitch together, thus allowing the device to also measure the tightness of the suture, which could be a vital metric.

Using this preliminary data as well as future data to be recorded, various machine-learning algorithms can be used to provide feedback to the subject. Potential feedback can be as detailed as to how much force the subject is currently applying and how much force the subject should apply in order to improve their suturing skill. The ultimate aim of the project is to build a device that does not require an analyst to read the data and decide the subject's skill level, but to have an algorithm that can assess the subject. One can go one step further and design a predictive algorithm that can suggest changes that the subject should make in order to improve their results and thus in turn, improve their suturing skill.

This follows with the end objective of the project to build a suturing platform that can not only give an evaluation of suturing skill, but also provide feedback to the subject regarding weaknesses in their performance and how they can improve their suturing skill, eventually substituting the role of instructors in today's suture training practices.

6. Appendix

6.1. Additional Figures

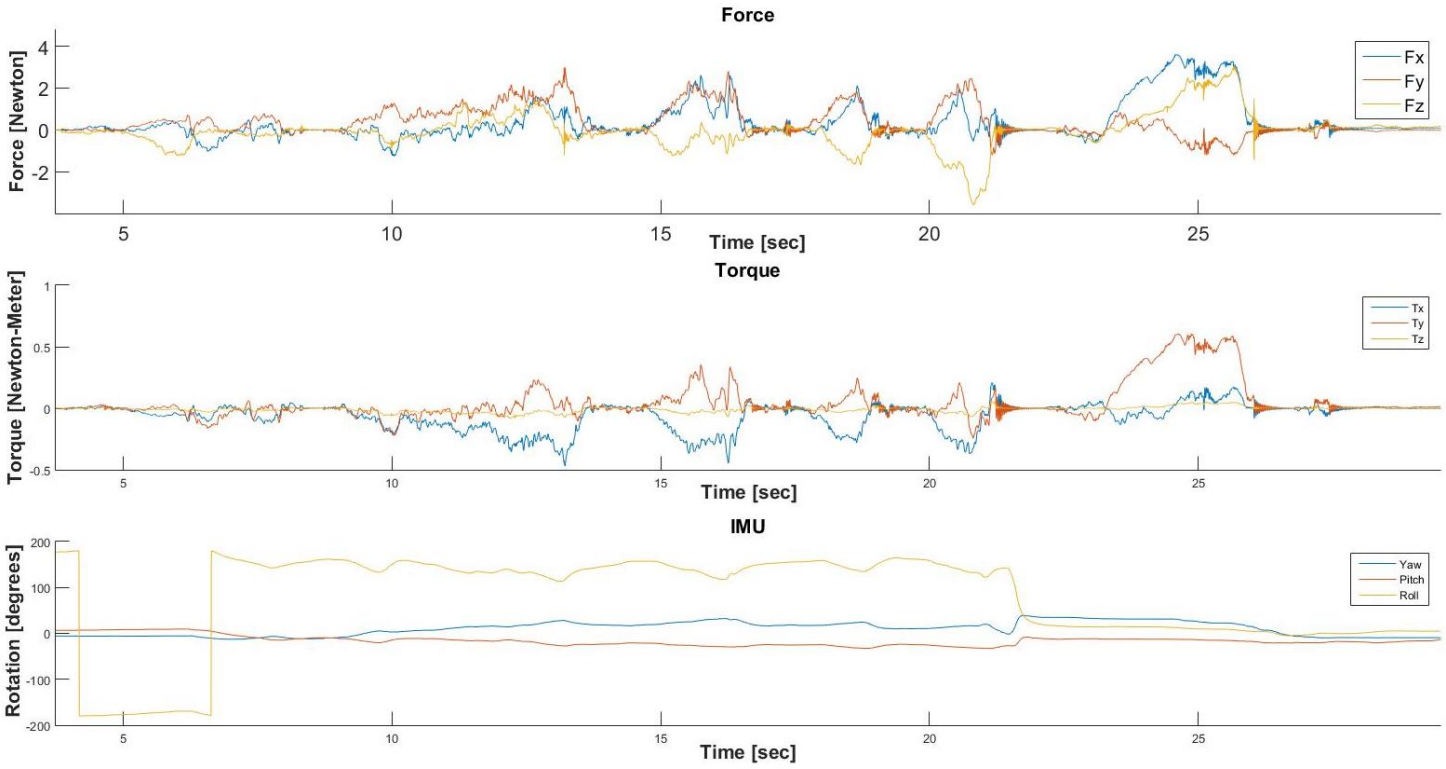

Figure A 1: Synchronized data of a single suture from a Novice subject's experiment

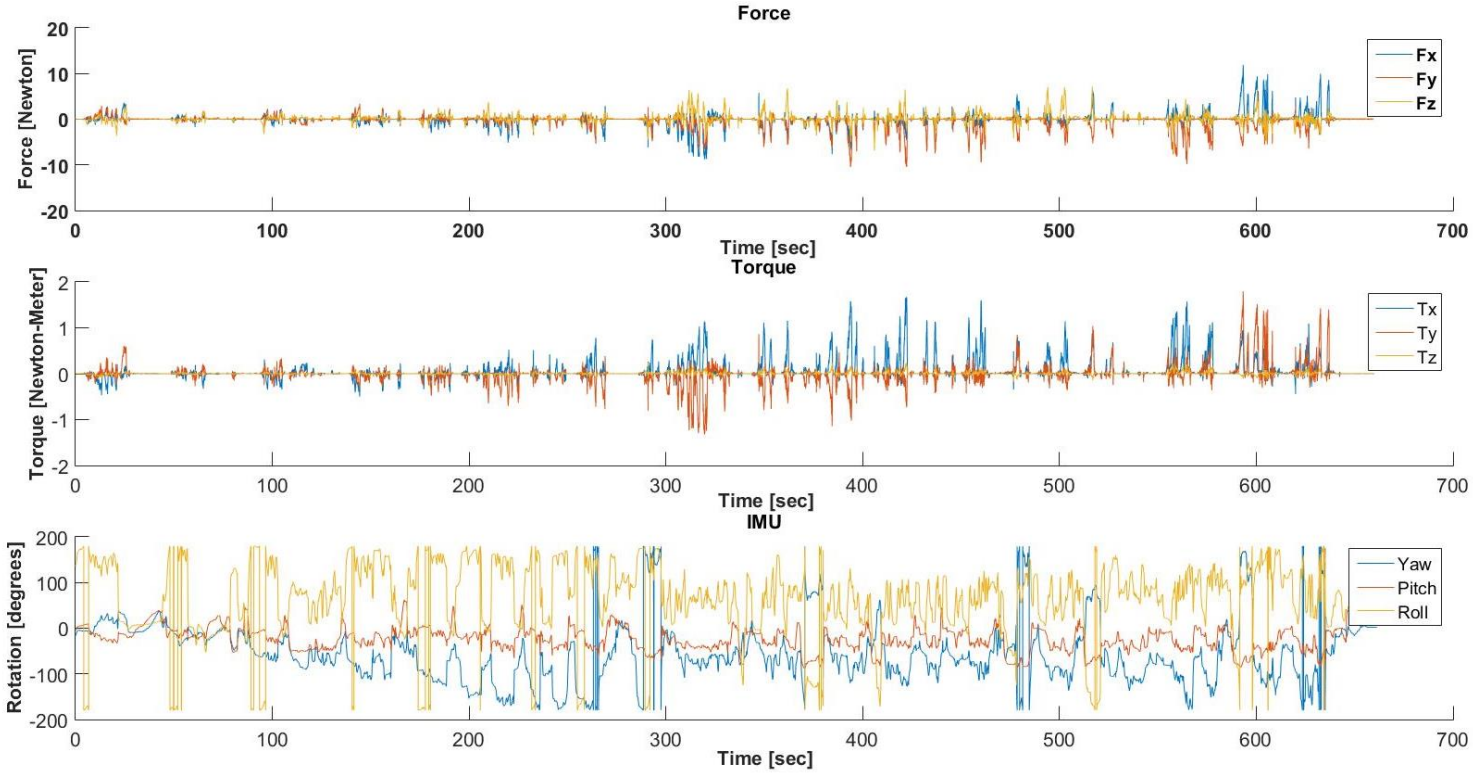

Figure A 2: Synchronized data from a Novice subject's experiment

7. References

[1] W. W. LaMorte, "Simple Continuous Stitch - Surgery | Boston University," [Online]. Available: http://www.bumc.bu.edu/surgery/training/technical-training/simple-continuous-stitch/. [Accessed 17 July 2016].

[2] P. Odland and C. Murakami, "Simple Suturing Techniques and Knot Tying," [Online]. Available: http://www.dermatology.ucsf.edu/education_training/Residency%20Program/VA/Simple%20Suturing%20Techniques%20&%20Knot%20Tying%20by%20PB%20Odland%20&%20CS%20Mu.PDF. [Accessed 17 July 2016].

[3] W. W. LaMorte, "Suturing Basics - Basics of Wound Closure and Healing," Boston University School of Medicine - Surgery, [Online]. Available: http://www.bumc.bu.edu/surgery/training/technical-training/suturing-basics/. [Accessed 18 July 2016].

[4] W. W. LaMorte, "Simple Interrupted Stitch," [Online]. Available: http://www.bumc.bu.edu/surgery/training/technical-training/simple-interrupted-stitch/. [Accessed 17 July 2016].

[5] T. Horeman, J. Dankelman, F. W. Jansen and J. J. van den Dobbelsteen, "Assessment of Laparoscopic Skills Based on Force and Motion Parameters," *IEEE Transactions On Biomedical Engineering,* vol. 61, no. 3, pp. 805-813, March 2014.

[6] T. Horeman, S. P. Rodrigues, F. W. Jansen, J. Dankelman and J. J. van den Dobbelsteen, "Force Parameters for Skill Assessment in Laparoscopy," *IEEE Transactions on HAPTICS,* vol. 5, no. 4, pp. 312-322, October 2012.

[7] T. Horeman, S. Rodrigues, F. W. Jansen, J. Dankelman and J. J. van den Dobbelsteen, "Force measurement platform for training and assessment of laparoscopic skills," *Surgical Endoscopy,* vol. 24, no. 12, pp. 3102-3108, December 2010.

[8] A. Trejos, R. Patel, R. Malthaner and C. Schlachta, "Development of force-based metrics for skills assessment in minimally invasive surgery," *Surgical Endoscopy,* vol. 28, no. 7, pp. 2106-2119, July 2014.

[9] R. C. Jackson and M. C. Çavuşoğlu, "Modeling of Needle-Tissue Interaction Forces During Surgical Suturing," *IEEE International Conference on Robotics and Automation (ICRA),* pp. 4675-4680, May 2012.

[10] A. Dubrowski, R. Sidhu and H. Carnahan, "Quantification of motion characteristics and forces applied to tissues during suturing," *American Journal of Surgery,* vol. 190, no. 1, pp. 131-136, July 2005.

[11] A. Sánchez, . O. Rodríguez, R. Sánchez, G. Benítez, . R. Pena, O. Salamo and V. Baez, "Laparoscopic Surgery Skills Evaluation: Analysis Based on Accelerometers," *Journal of the society of laparoendoscopic surgeons,* vol. 18, no. 4, pp. 1-5, Oct/Dec 2014.

[12] A. Dosis, R. Aggarwal, F. Bello, K. Moorthy, Y. Munz, D. Gillies and A. Darzi, "Synchronized video and motion analysis for the assessment of procedures in the operating theater," *Arch. Surg.,* vol. 3, no. 140, pp. 293-299, March 2005.

[13] M. S. Khan, S. Bann, A. Darzi and P. E. M. Butler, "Assessing Surgical Skill," *Plastic & Reconstructive Surgery,* vol. 112, no. 7, pp. 1886-1889, Jul 2002.

[14] J. A. Wong and E. D. Matsumoto, "Primer: cognitive motor learning for teaching surgical skill - how are surgical skills taught and assessed?," *nature clinical practice UROLOGY,* vol. 5, no. 1, pp. 47-54, January 2008.

[15] L. Sutherland, P. Middleton, A. Anthony, J. Hamdorf, P. Cregan, D. Scott and J. G. Maddern, "Surgical Simulation - A Systematic Review," *Annals of Surgery,* vol. 243, no. 3, pp. 291-300, March 2006.

[16] M. K. Chmarra, N. Bakker, C. 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, February 2006.

[17] T. Kavathekar, "DEVELOPMENT OF A SUTURING SIMULATION DEVICE FOR SYNCHRONOUS ACQUISITION OF DATA," Clemson, 2015.

[18] P. Fitts and M. Posner, Human Perfomance, Belmont, CA: Brooks/Cole Pub. Co., 1967.

[19] R. K. Reznick and H. MacRae, "Teaching Surgical Skills - Changes in the Wind," *N. Engl. J. Med.,* vol. 355, no. 25, pp. 2664-2669, December 2006.

[20] K. R. Wanzel, M. Ward and R. K. Reznick, "Teaching the surgical craft: From selection to certification," *Current Problems in Surgery,* vol. 39, no. 6, pp. 583-659, June 2002.

[21] R. A. Schmidt, "A schema theory of discrete motor skill learning," *Psychological Review,* vol. 82, no. 4, pp. 225-260, 1975.

[22] J. A. Kopta, "The Development of Motor Skills in Orthopaedic Education," *Clinical Orthopaedics and Related Research,* vol. 75, pp. 80-85, March/April 1971.

[23] S. Haluck and M. Krummer, "Computers and virtual reality for surgical education in the 21st century," *Arch Surg,* vol. 135, p. 786–792, 2000.

[24] Intuitive Surgical, "The da Vinci® Surgical System," [Online]. Available: http://www.intuitivesurgical.com/products/davinci_surgical_system/. [Accessed 17 July 2016].

[25] S. Rehman, J. Raza, P. Stegemann, K. Zeeck, R. Din, A. Llewellyn, L. Dio, M. Trznadel, W. Y. Seo, J. Chowriappa, T. Kesavadas, K. Ahmed and K. Guru, "Simulation-based robot-assisted surgical training: A health economic evaluation," *International Journal of Surgery,* vol. 11, pp. 841-846, 2013.

[26] P. J. van Empel, J. P. Commandeur, L. B. van Rijssen, M. G. Verdam, J. A. Huirne, F. Scheele, H. J. Bonjer and W. J. Meijerink, "Learning curve on the TrEndo

laparoscopic simulator compared to an expert level," *Surgical endoscopy,* vol. 27, no. 8, pp. 2394-2939, 2013.

[27] MediShield BV, "ForMoST laparoscopic boxtrainer for objective skills assessment," [Online]. Available: http://www.medishielddelft.com/wp-content/uploads/2012/08/ForMoST-a4.pdf. [Accessed 17 July 2016].

[28] M. Aizuddin , N. Oshima, R. Midorikawa and A. Takanishi, "Development of Sensor System for Effective Evaluation of Surgical Skill," in *The First IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics*, Pisa, 2006.

[29] Y. Yamauchi, J. Yamashita, O. Morikawa, R. Hashimoto, M. Mochimaru, Y. Fukui, H. Uno and K. Yokoyama, "Surgical Skill Evaluation by Force Data for Endoscopic Sinus Surgery Training System," in *5th International Conference on Medical Image Computing and Computer-Assisted Intervention-Part I*, Tokyo, 2002.

[30] A. Frischknecht, S. J. Kasten, S. Hamstra, N. Perkins, R. Gillespie, T. Armstrong and R. Minter, "The Objective Assessment of Experts' and Novices' Suturing Skills Using An Image Analysis Program," *Acad. Med.,* vol. 88, no. 2, pp. 260-264, 2013.

[31] G. Islam, K. Kahol, J. Ferrara and R. Gray, "Development of Computer Vision Algorithm for Surgical Skill Assessment," *Ambient Media and Systems,* vol. 70, pp. 45-51, 2011.

[32] "F/T Sensor: Mini 40," ATI, [Online]. Available: http://www.ati-ia.com/products/ft/ft_models.aspx?id=Mini40.

[33] Thales, "InertiaCube4," Thales, [Online]. Available: http://www.intersense.com/pages/18/234/.

[34] Microsoft, "Visual Studio Downloads," Microsoft, [Online]. Available: https://www.visualstudio.com/downloads/download-visual-studio-vs.

[35] The MathWorks, "MATLAB," The MathWorks, [Online]. Available: http://www.mathworks.com/products/matlab/.

[36] National Instruments, "PCI Multifunction Data Acquisition (DAQ)," National Instruments, [Online]. Available: http://sine.ni.com/nips/cds/view/p/lang/en/nid/209158.

[37] Logitech, "HD PRO WEBCAM C920," Logitech, [Online]. Available: http://www.logitech.com/en-us/product/hd-pro-webcam-c920.

[38] OpenCV, "OpenCV," [Online]. Available: http://opencv.org/.

[39] HP, "AU165AA - HP Pro Webcam," HP, [Online]. Available: http://www.hp.com/ecomcat/hpcatalog/specs/provisioner/05/AU165AA.htm.

[40] National Instruments, "NI-DAQmx 15.1.1f3," National Instruments, [Online]. Available: http://www.ni.com/download/ni-daqmx-15.1.1/5665/en/.