Final Report

Development of a Pediatric Laparoscopic Training and Skills Assessment Simulator

Authors:

Atallah Madi

Esraa Alaa Aldeen

Huda Sheikh

Youssef Megahed

Group 27

(Biomedical and Electrical Engineering)

Supervised by:

Dr. Carlos Rossa

Report Date:

April 10th, 2024

SYSC 4907

Department of Systems and Computer Engineering

Faculty of Engineering and Design

Carleton University

Abstract

Pediatric laparoscopic surgery is a minimally invasive technique used for training trainees on basic surgical movements while providing them with comprehensive feedback. The simulator incorporates advanced technologies such as augmented reality (AR), sensor technology, signal processing, and machine learning to provide an efficient learning experience that mimics a complex pediatric laparoscopic environment. It addresses challenges such as operating in 3D while viewing in 2D, enhancing learning, and eliminating expert supervision. This simulator's unique features include real-time feedback and performance assessment through Dynamic Time Warping (DTW) and deep learning, allowing for comparison with expert surgeons. The simulator is designed to be user-friendly, with an intuitive UI and data processing pipeline, making surgical training more accessible and efficient.

Keywords: *Pediatric Laparoscopic, Simulator, Augmented Reality, Machine Learning, Deep Learning, Surgical Gestures, Instrumentation, Feedback.*

Acknowledgements

Group 27, comprising Atallah Madi, Esraa Alaa Aldeen, Huda Sheikh, and Youssef Megahed would like to extend our sincere thanks to Dr. Carlos Rossa for his invaluable mentorship, and remarkable support throughout our capstone project. We would also like to acknowledge Dr. James Green for his constructive advice and practical suggestions, Dr. Ahmed Nasr from C.H.E.O, and Dr. Georges Azzie from SickKids for their medical insights prior to our development of the Pediatric Laparoscopic Surgery Simulator.

We would also like to express our appreciation to Carleton University for publishing an article featuring our project, Biomechatronic Lab for providing us with the necessary tools and working environment, the Systems and Computer Engineering Department and Kostiuk Engineering Funding Collective for their generous funding, which made this project possible.

All these individuals and organizations have been instrumental in our project's success, and we are deeply thankful for everyone and their contributions.

Table of Contents

Table of Figures

List of Tables

1. Introduction

This report provides a final update on our capstone project, a state-of-the-art Pediatric Laparoscopic Surgery Simulator. This simulator, which incorporates advanced augmented reality technology with machine learning, aims to provide enhanced feedback for performance evaluation. Along with improved evaluation schemes, users are immersed in an interactive, risk-free training environment that verbally and visually guides them throughout their learning experience. From previous iterations, we have now integrated our machine-learning-driven object detection for surgical tasks to be used for performance evaluation in real-time as the user interacts with the user interface. We have also improved the hardware configuration of the simulator, equipping it with a computer and display screen to ensure it is a compact, portable device that can be utilized in various settings. Finally, significant improvements were made to enhance the user's experience while interacting with the user interface and viewing their feedback after performing a surgical task. The changes implemented in this iteration will be detailed in the sections below. This report begins with an overview of the project's primary goal, emphasizing the critical need for innovative training tools in this specialized field of laparoscopic surgery. We then offer in-depth background information crucial to understanding the project and the work we have accomplished this year. This includes an overview of pediatric laparoscopic surgery, the current standards and requirements for pediatric laparoscopic surgical training, and an analysis of recent technological advancements in simulation tools that enhance the skill set of surgeons. In the following section, we delve into the relevance of this project to the academic pursuits of each team member within the fields of Biomedical and Electrical Engineering, outlining the specific responsibilities and roles undertaken by each member during the project. The Implementation section will cover the four areas of our project, namely the Hardware Instrumentation, Surgical Gesture Recognition, Data

Processing & Performance Evaluation, and User Interface sections. The Testing section will provide a general overview of how we conducted multiple tests to ensure the functionality of our instrumentation and software setup. Next, the Discussion and Analysis section will discuss some general problems we encountered and how we resolved them. This section follows with the Future Improvements section, which provides future capstone groups with suggestions on how the simulator can be developed in later stages. Finally, to conclude, we discuss the benefits of the innovative simulator we have created.

2. Background

Laparoscopic surgery, often called ‘keyhole’ surgery, is a minimally invasive surgical (MIS) technique that differs from traditional ‘open surgery.’ Instead of making a large incision on a patient’s abdomen, laparoscopy involves several smaller, 0.5-1 cm diameter incisions, each called a ‘port’ [1]. A trocar placed inside a hollow sleeve or cannula at each incision or port is slipped through to create an access point for surgery [2]. A trocar is a pen-shaped instrument (triangular at one end) placed inside a hollow tubular instrument [2]. An image of a trocar is shown below in Figure 1.

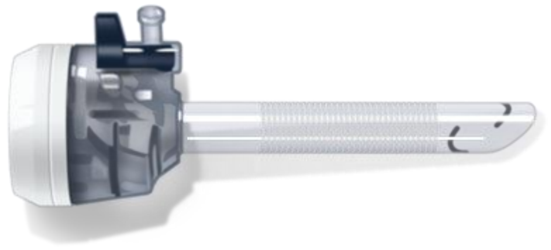


Figure 1 : Image of a trocar.[2]

In these incisions, the laparoscopic tools, a tube to pump gas into the abdominal cavity, and a specialized camera called a laparoscope are placed through [3]. A sample placement of these trocars is shown in Figure 2 below.

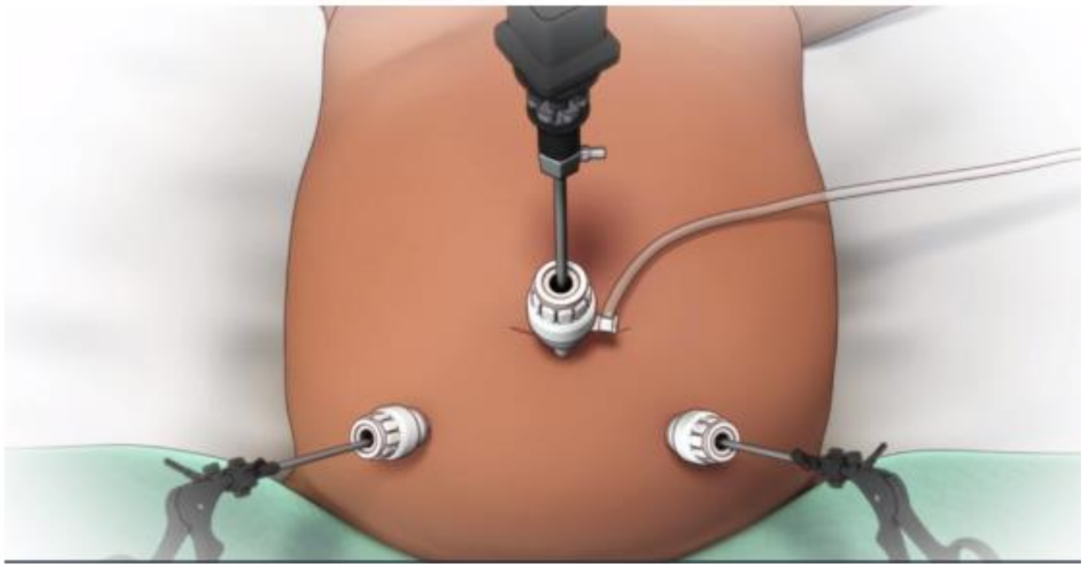


Figure 2: Pediatric laparoscopic surgery setup with a laparoscope and instruments inserted through trocars [3]

Various laparoscopic instruments available today are made of durable stainless-steel material with narrow shafts up to 3-5mm diameter to fit through the ports [2]. Along with a laparoscope, which is a thin telescope equipped with a camera and light source, there are a variety of other tools available for various functions [2]. Needle drivers, composed of handles, joints, and jaws, grasp suturing needles while securing wounds and surgical incisions [2]. Their tips may be curved or straight [2]. Bowel graspers, 3 - 5 mm in diameter, handle, and grasp tissue, allowing surgeons to observe and perform various procedures [2]. Other instruments used in laparoscopy include scissors, suturing needles, and surgical meshes [2]. An image of these tools is given in Figure 3 below.

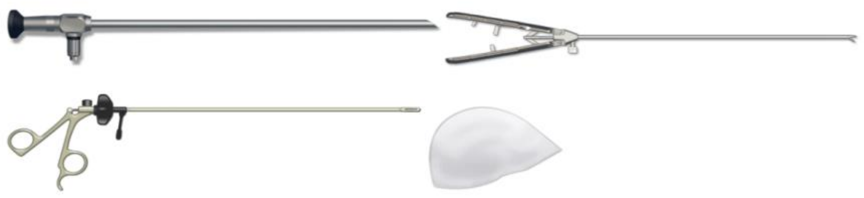


Figure 3: Laparoscopic instruments commonly used today. From left to right, laparoscope (top left), needle grasper (top right), bowel grasper (bottom left), and surgical mesh (bottom right) [2]

Additionally, in laparoscopy, surgeons are limited to four degrees of freedom of motion, which are the yaw, pitch, (counterclockwise/clockwise) roll, and (forward/backward) surge movements [4]. They lose the heave and sway motions due to the pivots that hold the instruments in place [4].

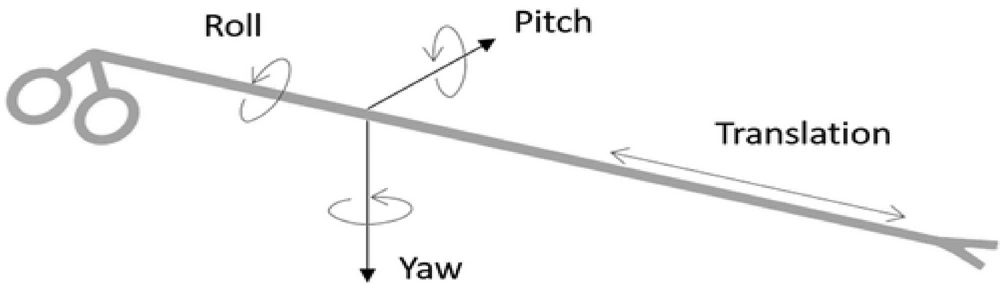


Figure 4:The four degrees of motion surgeons are limited to in laparoscopy are the roll, pitch, yaw, and surge (translation) movements [4]

Regarding postoperative outcomes, due to smaller incisions, laparoscopy is associated with reduced pain post-operation, reduced scarring, reduced hospital stays, and lower incidences of significant wound complications [5]. That means patients can return to everyday life and engage in normal activities much sooner after their operation than open-surgery patients [5].

However, laparoscopy is known to have a steeper learning curve for trainees than conventional open surgery, and each patient-specific procedure can present additional complications for surgeons and trainees [6]. Thus, training for laparoscopy is different from conventional open surgery training in several ways. Trainees must work within restricted spaces and exercise less intuitive and limited movement through the trocars [7]. They must also enhance their hand-eye coordination capabilities due to the added challenges of operating while looking at a screen instead of operating under direct vision, like in open surgery [7]. There is also an increased need to develop manual dexterity skills, like fine motor skills, since the long instruments used in laparoscopy amplify even small movements and errors made by the trainee [7]. Trainees also experience the fulcrum effect, which describes the inversion of movement while operating [7]. For instance, this refers to when the surgeon moves their hand to the right, and the instrument's tooltip is moved to the patient's left on the screen [7].

Additionally, trainees have to grapple with challenges like only using four out of six degrees of motion, the lack of physical sensation or touch from their hands, and the lack of 3-dimensional images like in open surgery [7]. In a clinical setting, laparoscopic training can largely depend on the variety of cases that arrive at the hospital [6]. Due to the increased need for the unique skills required for laparoscopy, a need arose to develop relevant simulators and training programs. Simulators provide trainees with a safe, ethical, and friendly learning environment to develop their hand-eye coordination abilities, improve their manual dexterity, and soften the learning curve of acquiring laparoscopic skills [6].

When minimally invasive surgery (MIS) was first adapted for infants and children, training surgeons and residents on pediatric laparoscopy presented additional challenges [8]. The use of smaller workspaces, shorter and narrower instruments, smaller margins of error, and the expansion of MIS in children and infants warranted the development of specific learning approaches and simulators for pediatric laparoscopy [8]. This type of procedure demands a more unique set of operating skills than adult laparoscopy. The abdomens of children and infants present a small workspace whereby damage to nearby vessels, the bowel, and other structures can quickly occur and lead to grave consequences [9]. While the instruments are smaller and narrower, surgeons must still maneuver these within smaller anatomical structures and a restricted operating field [9,10]. Due to these limitations, there also exists a lower tolerance of forces to surrounding tissues [10]. Thus, the demand for the unique skill set to perform the procedure has led to the use and development of pediatric laparoscopic simulators. These simulators must mimic the challenging surgical environments in their setup by using smaller instruments, smaller needles, more sensitive force-sensing devices, and more limited surface areas for training tasks [10].

The Fundamentals of Laparoscopic Surgery (FLS) was a program developed by the Society of American Gastrointestinal and Endoscopic Surgeons (SAGES) in 2004 to develop a framework that assesses knowledge and skills acquisition for laparoscopic surgery [11]. The program has been widely regarded as a valid and reliable tool and educational curriculum to train and assess surgeons [11]. The Canadian Association of General Surgeons endorses the FLS program as a valuable and comprehensive training and assessment tool to gauge residents' knowledge and skills in basic laparoscopic training. The FLS examination consists of a web-based cognitive examination, followed by a manual-skills assessment on five fundamental laparoscopic tasks, namely, the peg transfer task, precision cutting task, ligating loop task, suturing with extracorporeal knot-tying task, and suturing with intracorporeal knot-tying task [11]. For now, and for our group's proposed surgical simulator, the focus will be on the peg transfer task and the suturing tasks with extracorporeal knot-tying and intracorporeal knot-tying [11]. More tasks may be added to our simulator if time, resources, and efforts allow.

The most straightforward task, the peg transfer task, involves using two Maryland grasper dissectors [12]. It gets trainees to grasp each ring with their non-dominant hand one by one and transfer it to their dominant hand in mid-air to place it on a peg on the opposite side of the board [12]. The process is repeated until all rings are on the other side of the board [12]. This process is then repeated in reverse, starting with the dominant hand and then transferring the rings to the non-dominant hand [12]. Penalties are applied if the ring falls out of the field of view or is no longer retrievable, but trainees are still expected to continue the procedure should this happen [12]. If a ring is dropped within the field of view, trainees can pick it up with their hands and continue the procedure [12]. The FLS recommends that trainees complete this entire procedure in 300 seconds, with the time starting from the minute the grasper touches the first ring till the time the grasper lets go of the last ring [12].

In the extracorporeal suturing task, trainees are expected to use two needle drivers (or one Maryland grasper and one needle driver) and endoscopic scissors to place a long suture through two marks in a Penrose drain and then perform three single throw knots (extracorporeally) to secure the slit on the drain [12]. Then, once the three throws are secured, both ends of the suture are cut [12]. Penalties are applied for a knot that falls apart, for deviating from the suturing marks on the Penrose drain, and for improper closure of the slit [12]. This procedure is expected to be completed within 420 seconds, beginning when the first tool is visible through the camera and ending when the cuts are made to either end of the suture [12].

Similarly, in the intracorporeal suturing task, trainees are expected to use two needle drivers (or one Maryland grasper and one needle driver) and endoscopic scissors to place a short suture through two marks in a Penrose drain and then perform three throws of a knot (one double throw, and two single throws) intracorporeally to secure the slit on the drain [12]. Then, once the three throws are secured, both ends of the suture are cut [12]. Penalties are applied for a knot that falls apart, for deviating from the suturing marks on the Penrose drain, and for improper closure of the slit [12]. This procedure is expected to be completed within 600 seconds, beginning when the first tool is visible through the camera and ending when the cuts are made to both ends of the suture [12]. While a scoring rubric from the FLS is not publicly available, plenty of scoring methodologies developed by medical professionals exist online. They can be used to assess the performance of trainees. For instance, many experts have developed independent scoring rubrics for the five basic laparoscopic tasks [13].

To adapt the FLS training and assessment framework for pediatric laparoscopy, smaller and softer rings for the peg transfer task, as well as the 3 mm graspers instead of the 5 mm ones used for adult laparoscopy, should be employed [10]. The size of the Penrose drain used in the intracorporeal, and extracorporeal suturing tasks should be reduced by 6, and smaller needles and 3 mm instruments should be used [10]. Additionally, the surface area of the Velcro securing the drain to the board should be decreased by a factor of 5 to increase the risk of avulsion that can occur due to the application of even minimal force in pediatric laparoscopy [10]. The timing and penalty scores could be the same as that for adults, except additional parameters or penalties may need to be added to the scoring rubric for pediatric laparoscopies, such as assigning the trainee a score of zero if avulsion of the Penrose drain occurs [10].

2.1. Description of Current Surgical Simulators

The advent of surgical simulators has opened a new avenue in surgical training. It provides surgical residents and trainees with a safe, trainee-friendly learning environment to practice new instruments, procedures, and skills before operating on actual patients [1]. Due to the rapidly changing nature of instruments and laparoscopy's less intuitive techniques/procedures, it is difficult to use the traditional apprentice approach when teaching laparoscopic skills to novices [1]. Thus, many surgical simulators have focused on developing simulation training for laparoscopic surgery [1]. These have included low-tech and high-tech simulators like video-box trainers, organic simulators (on human cadavers and animal models), hybrid trainers, virtual reality (VR) trainers, and augmented reality (AR) simulators [14].

Video-box trainers resemble traditional box trainers in that they provide a box and holes for trocar insertion and various laparoscopic tasks, but also a camera projected on a monitor for the trainee [14]. While they can help practice hand-eye coordination skills and particular laparoscopic tasks, they need more objective expert feedback and focus only on specific tasks/techniques rather than an entire procedure [14]. They require the presence of an expert to assess the performance of a trainee, which is only sometimes available. The FLS uses the McGill Inanimate System for Training and Evaluation of Laparoscopic Skills (MISTELS) system to train novices on the five core laparoscopic tasks as part of its laparoscopic training program [14].

Hybrid trainers, or virtual reality (VR) trainers, integrate video-box simulators with virtual reality to guide a trainee throughout a performance with interactive haptic feedback and feedback metrics without needing an expert surgeon to be present [14]. The user can practice individual skills like suturing, knot-tying, etc. or go through an entire procedure in a realistic computer-generated environment in more advanced simulators [14]. Objective feedback can be in the form of time taken for a particular procedure or the strength of a knot [14]. Several VR trainers for different subspecialties have become available, including LapSim and Lapmentor simulators for laparoscopy [14]. Sometimes, these simulators may require additional time and effort to set up and maintain as they are not portable. However, they train residents to make sound decisions, preparing them for intense operating environments [14].

Finally, augmented reality (AR) trainers, which overlay computer-generated information into the user's field of view, connect the virtual world to the real world [14]. Some of the approaches covered by AR simulators include overlaying computer graphics on anatomical structures, mapping instrument paths during a procedure, creating a realistic training environment that involves real instruments interacting with actual artifacts in the surgical environment, and providing objective force, motion, and haptic feedback without the need of an expert observing the performance in real-time [14].

The current implementation of our Pediatric Laparoscopic Surgical Simulator project is an augmented reality (AR) simulator, overlaying the surgical environment in the box trainer with computer-generated information to guide the user through a laparoscopic task, like the peg transfer task. Currently, thresholds from sensory information regarding force, velocity, and acceleration of movement are used to assess trainee performance. Our current focus is to enhance the AR simulator by adding more advanced laparoscopic tasks and integrating it with powerful machine learning/ signal processing algorithms and an interactive user interface to enable it to recognize surgical gestures and classify a user's performance based on accurately collected expert data.

2.2. Machine Learning and Surgical Simulation

Current literature on laparoscopic surgical simulators has focused on applying artificial intelligence (AI) to surgical simulators. AI aims to mimic human intelligence by learning to predict, classify, and learn a particular task to make decisions [15]. AI can enhance surgical simulators today by assessing a trainee's performance, providing more personalized feedback, and enhancing the visualization of the surgical environment itself [15]. Among high costs and long-term maintenance, current VR simulators like LapSim and AR simulators like ProMIS present other limitations, like insufficient metrics/performance criteria to assess trainees [16]. For instance, the ProMIS AR laparoscopic simulator uses "time of performance," "path length," and "smoothness of motion" to assess trainee performance [16].

On the other hand, the LapSim simulator uses "time of performance," "tissue damage," and "path length" to assess trainee performance [16]. "Time of performance" cannot be the most important criterion with which to assess trainees, especially since the primary area of concern is often whether the trainee has used the correct technique and tied a knot properly [16]. Most current simulators do not accurately assess specific surgical techniques, compare a trainee's performance to an expert's performance and provide constructive feedback that the user can understand or implement [17]. Instead, they assume agreed-upon minimum threshold values for completion time, path length, collisions, force, or velocity to assess a user's performance [17].

Machine learning can evaluate performance more accurately and provide more personalized feedback to the user. For instance, machine learning algorithms can be trained to recognize patterns of movement and behaviour in novice and expert performers to provide more detailed feedback to the user. In this way, incorporating machine learning into our simulator will provide more instructional and informative feedback to the user, increasing the educational value and usefulness of the simulator itself.

There have been several AI-related applications in surgical simulators in recent years [15]. A group at McGill University developed a machine learning algorithm that classified the skill levels of surgeons during a brain tumour resection task and provided detailed feedback [15]. This was done by getting machine learning algorithms to select important performance metrics from an expert-defined list and use these to distinguish between the skill levels of surgeons in a virtual reality simulator [15]. The algorithms could select a novel list of performance metrics and achieve high classification accuracies when classifying trainee performances along training levels [15]. However, the challenges and problems introduced by applying AI and machine learning to surgical simulation training must be addressed. For one, the lack of valid and objective performance metrics for an algorithm to use, and another, the inability of the algorithm to accurately correlate between a trainee's performance on a simulator and in the operating room [15]. In addition, these algorithms may not consider the variety of methods and techniques trainees can use to achieve a particular goal. Potential solutions to these problems could be to divide a procedure into sub-tasks and obtain surgeon-led consensus on evaluation metrics in laparoscopy.

We aim to integrate our current AR pediatric laparoscopic surgical simulator with machine learning and signal processing algorithms to 1) classify individual gestures or sub-tasks within a surgical procedure (like the peg transfer and suturing tasks) and 2) evaluate a trainee's performance and compare them to an expert's performance. Our conversation with pediatric surgeons Dr. Ahmed Nasr and Dr. Georges Azzie on September 27th, 2023, focused on how the machine learning algorithms that we will develop can assess a trainee's success in completing each sub-task of a procedure by evaluating their smoothness of movement, patterns of motion, and application of force. Each trainee's performance will be compared against the performances of actual, standard, expert data we have collected.

3. Project Objective

The objective of this project was to develop a portable, easy-to-use, cost-effective, and user-friendly simulator for pediatric laparoscopic surgery that would provide trainees with comprehensive feedback about their performance without the need for constant expert supervision. The objective was to design the simulator to evaluate the performance of trainees after they complete the assigned task and provide them with various types of feedback, they can independently use to assess their skills. We designed the simulator to compare the trainee's performance against an expert's to easily see where they may have erred. This comparison formed the basis for the various types of feedback we intended to provide in the form of graphs, videos, and maps of tool trajectories. In addition to the comprehensive feedback and evaluation measures, we used an existing version of the simulator. We equipped the simulator with additional instrumentation to measure motion and force-related parameters. We also designed the box simulator to be a more portable, compact, and easy-to-use device. Finally, to connect all the components, we consolidated everyone's parts together and integrated them with a more powerful and interactive user interface. The major functional parts of this project included equipping the simulator with the appropriate instrumentation to ensure accurate performance capture, gathering performance data from these sensors, analyzing the gestures of a trainee's performance, processing all this data to compare with an expert's performance, and providing users with an interactive user interface that they can use to view different types of feedback. Over the past months, we have developed a simulator compact and powerful.

4. Relation to Studies

Now that the project has been completed for the 2023-24 year, it is essential to reflect on the relevance and importance of this project to our academic studies and how it has deepened our understanding in our respective areas.

All members of the team - Atallah, Esraa, Huda, and Youssef - are currently in their final year of their undergraduate degree program in Biomedical and Electrical Engineering. The degree has equipped them with a shared foundational knowledge and skill set essential for their collaborative effort. While each member brought unique strengths and expertise to the project, they all shared a solid academic background, ensuring their overall project contributions aligned with their studies and objectives.

Atallah's academic journey has given him a deep understanding of project design, biomedical instrumentation, digital and analog electrical circuits, and various programming languages. All these fundamental concepts came together to build his role in the project, where he focused on enhancing the platform design, upgrading its instrumentation, and creating the system data pipelines. His academic background in hardware, software, and electrical equipped him with skills essential to the project. His introductory engineering course in his first year laid the groundwork for designing an innovative platform, employing CAD models, and developing a new, portable screen mount. Furthermore, his co-op position as a hardware developer was instrumental, providing him with the practical experience necessary to upgrade the force sensors and resolve various sensor-capturing discrepancies. With his improvements, Atallah has ensured that data are captured with enhanced accuracy, and the simulator now boasts portability, allowing for a broader range of training scenarios and environments. These advancements mark a significant leap forward in surgical education, making training more accessible and practical.

Esraa leveraged her academic knowledge to refine her skills and capability, significantly contributing to the gesture recognition component of laparoscopic tasks. Her work was realized through deep learning and convolutional networks, which were adept at extracting robust features from real-time data. This extraction facilitated the provision of varied guidance tailored to the object states during training. Furthermore, she automated the labour-intensive task of labelling surgical gestures. Utilizing a deep learning model, she enabled automatic labelling of user/trainee gestures alongside their respective timestamps. This process constituted a crucial preprocessing step later integrated into our pipeline for thorough performance evaluation.

Huda, throughout her degree and her coursework, has worked on several projects related to biomedical engineering in academics, research, and industry. Throughout these projects, she has gained experience in image processing, biomedical imaging, sensor analysis, software development, and project management. Huda's primary goal in this project consisted of consolidating the project's signal processing and machine learning aspects (the performance assessment and gesture recognition algorithms) with a powerful, interactive user interface (UI) that guides and assesses a trainee throughout their performance. This ensured that the laparoscopic surgical simulator functioned correctly to present the user interactive feedback, as the performance assessment and gesture recognition algorithms provide.

Youssef's degree provided him with comprehensive exposure to hardware and software development, signal processing, machine learning, and biomedical engineering knowledge. Beyond academic studies, Youssef gained hands-on experience in medical imaging research, where he harnessed the capabilities of deep learning. Additionally, as a data science intern, he honed his skills, adeptly using machine learning and Python. His primary focus was signal processing and evaluating user performance using techniques such as Dynamic Time Warping.

For the project, Youssef developed a robust signal-processing algorithm that delivers feedback to the user through videos, graphs, and tool trajectory maps. This algorithm processes simulation data to assess trainee performance accurately. Additionally, he applied machine learning to develop an LLM-based chatbot, enabling users to ask questions about surgical training. This chatbot is based on the GPT-3.5 Turbo model. Youssef also took on the task of cleansing and standardizing raw sensor data, organizing it into structured data frames, and saving it as CSV files. This prepared data feeds into evaluation algorithms, providing valuable insights into trainee performance. Throughout his studies and professional experiences, Youssef's diverse techniques and methods have laid the groundwork for his contributions, ensuring that users of the training simulator receive clear and precise results.

5. Contributions

5.1. Project Individual Contributions

Table 1: Outline of each member's contribution to the capstone project

Contributor	Section
Atallah Madi	• Hardware and sensor installation. • Sensor verification and calibration. • Schematics, electrical wiring, and soldering. • CAD modelling and 3D printing. • Camera and HSV calibration. • Microcontroller program and data pipeline. • Participated in data collection.
Esraa Alaa Aldeen	• Gesture recognition modelling. • Data collection and labelling videos dataset. • Implementation SSD for gesture detection. • Developing a new improved model for gesture recognition. • Training and implementation the ST Model. • Models tuning and evaluation.
Huda Sheikh	• User Interface Design. • Assisted in UI Integration. • Software debugging, verification, and calibration. • Implementing tool trajectory feedback. • Camera and HSV calibration. • Participated in data collection for gesture recognition model. • Assisted in data labelling for gesture recognition model.
Youssef Megahed	• Assisted in sensor verification. • Data processing pipeline implementation. • Signal processing techniques implementation. • Software implementation of all the different types of feedback. • UI integration. • Assisted in software debugging and verification. • Participated in data collection for gesture recognition model. • Worked on the new gesture recognition model.

5.2. Report Individual Contributions

Table 2: Outline of each member's contribution to the final report

Contributor	Section
Atallah Madi	• Acknowledgement • Relation to Studies • Implementation (Embedded system) • Testing • Discussion and Analysis • GitHub repository
Esraa Alaa Aldeen	• Abstract • Relation to Studies • Implementation (Surgical Gesture Recognition)
Huda Sheikh	• Introduction • Background • Project Objective • Relation to Studies • Implementation (User Interface Design)
Youssef Megahed	• Introduction • Relation to Studies • Implementation (Data Processing Pipeline, Performance Evaluation, and Development and integration of new features in the user interface) • Testing • Discussion and Analysis • Future Improvements • Conclusion • GitHub repository

6. Implementation

6.1. Embedded system

6.1.1. Sensors and Microcontroller

The system incorporates two types of sensors, the MPU6050 and PMW3389, placed at the trocars to capture the four movements: Roll, Surge, Pitch, and Yaw. Furthermore, the system's platform is instrumented by using load cells and HX711 (analog-to-digital convert and amplifier), which captures the applied bidirectional forces.

All these hardware sensors require a microcontroller to operate and collect the data. Thus, the use of an Arduino Nano board. It is equipped with digital (I2C and SPI) and analog communication busses which the MPU6050, PMW3389, and force sensor use. The speed of communication is uniform for all sensors at a 31250 baud rate which provides high-speed data transmission. The entire project circuitry with all the used pins on the microcontroller and the data pipeline are shown in Figures 5 and 6 receptively.

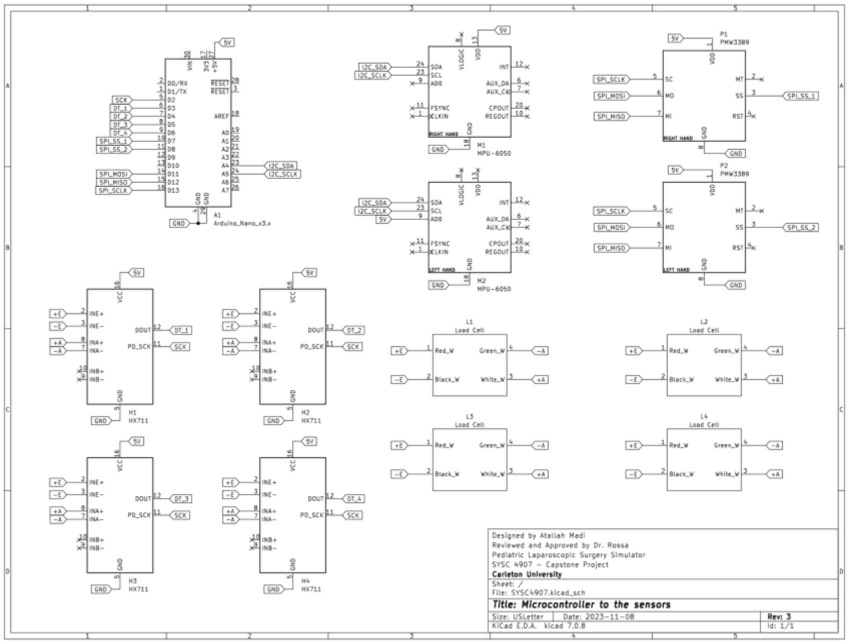


Figure 5: Project schematic

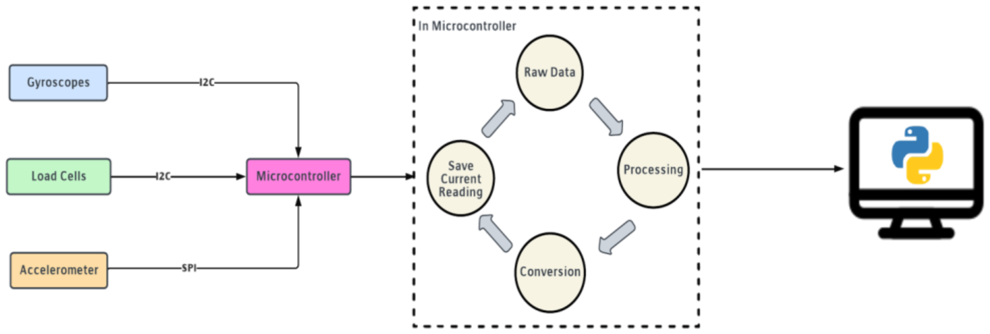


Figure 6: Data pipeline.

I2C (Inter-Integrated Circuit) is a two-wire communication protocol widely used for connecting various peripheral devices to microcontrollers, such as Arduino boards. Each I2C sensor requires two essential lines: SDA (Serial Data) and SCL (Clock Line), to establish the connection [18]. The SDA line is used for bidirectional data transfer between the Arduino and the sensor, allowing them to exchange information. Meanwhile, the SCL line carries clock signals generated by the Arduino, synchronizing the timing of data transmission between devices. Together, these two lines enable efficient and synchronous communication between the Arduino and multiple I2C sensors (MPU6050, and HX711) connected on the same bus.

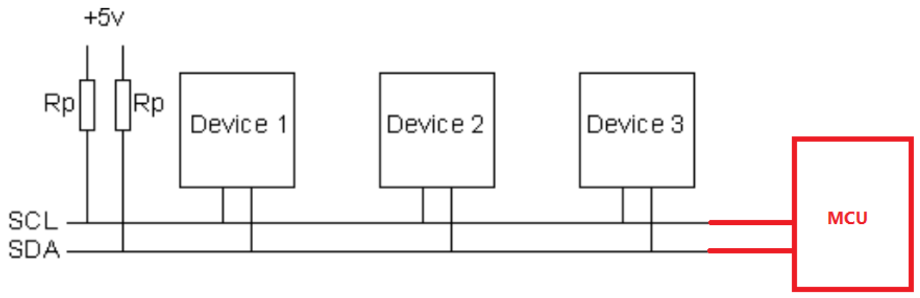


Figure 7: I2C Bus circuitry [18].

SPI (Serial Peripheral Interface) is a synchronous serial communication protocol commonly used for short-distance communication between microcontrollers and peripheral devices. Unlike I2C, SPI typically requires four communication lines: MOSI (Master Out Slave In), MISO (Master In Slave Out), SCK (Serial Clock), and SS (Slave Select). MOSI is used by the master device to send data to the slave device, while MISO is used in the opposite direction [19]. SCK generates the clock signal, synchronizing data transfer between the master and slave devices. The SS line allows the master device to select the specific slave sensor (PMW3389) with which it wants to communicate. SPI operates in full-duplex mode, meaning data can be transmitted and received simultaneously, providing faster communication compared to I2C [19].

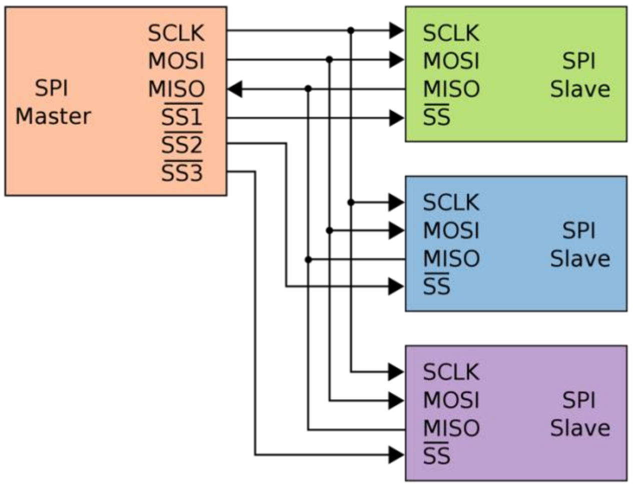


Figure 8: SPI Bus circuitry [19].

The MPU6050 sensor is a six-axis motion-tracking module that combines a 3-axis gyroscope and a 3-axis accelerometer [20]. This sensor captures pitch, and yaw movements by measuring angular velocity and linear acceleration along each axis. The gyroscopic component accurately detects rotational movements, while the accelerometer provides orientation and tilt information.

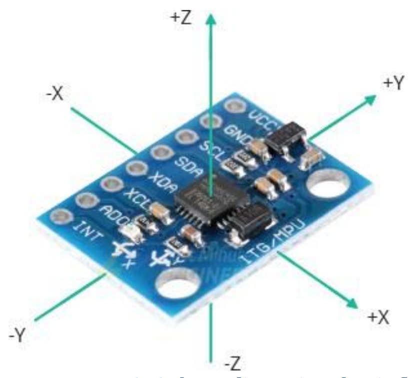


Figure 9: MPU6050 three-dimensional axis [20].

The PMW3389 sensor is primarily utilized for precise tracking of linear movements and rotation, making it ideal for detecting surge (linear movements) and roll (rotational movements). It uses optical motion technology which is commonly found in applications such as computer mice and robotics. The sensor resolution is 16,000 CPI (Counts Per Inch) which refers to the number of digital counts the sensor reports for every inch the surface moves on over it [21]. The reading is captured by infrared light that reflects off to the image sensor which takes continuous snapshots of the surface and compares the images to determine the distance and direction of travelling in X and Y direction [21].

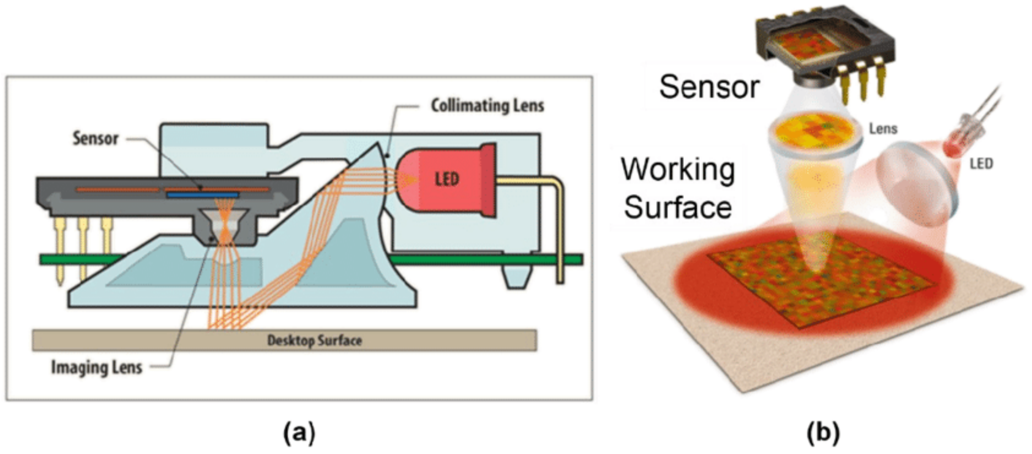


Figure 10: (a): Illustration of the reflection and imaging sensor. (b): Working principle of the optical sensor [22].

To determine surge, the CPI value in the Y displacement must be converted into mm. The sensor outputs the CPI value in two’s complement format (for negative displacement) using 32,767 bits to represent 16,000 different CPI values. Since CPI is in inches it must be converted to mm (1 inch = 25.4 mm) resulting in using 1290 bits for 1mm. This conversion is implemented in the code using equation 1 with the corresponding data variable (*data.1.dy* and *data2.dy*) for left and right respectively [23].

$$Surg\ (mm) = L\_PMW\_Y = \frac{data1.dy}{1290.0} + L\_PMW\_Y \qquad \textbf{(1)}$$

To determine roll, the CPI value in the X displacement must be converted into mm which is similar to the surge movement. However, since the sensor reads the shaft's movements from an external view, where it is placed on the tool shaft. The degree of rotational roll could be calculated by using the arc length equation with the tool shaft radius (r =1.5mm). This conversion is implemented in the code using equation 2 with the corresponding data variable (*data.1.dx* and *data2.dx*) for left and right respectively [23].

$$Arc_{length}\ (mm) = L\_PMW\_Y = \frac{data1.dy}{1290.0} + L\_PMW\_Y$$

$$Arc_{length}\ (mm) = L\_PMW\_Y = 2\pi \times r \times (\frac{\vartheta_{roll}}{360°})$$

$$\vartheta_{roll} = 360° \times \frac{L\_PMW\_Y}{2\pi \times 1.5mm} \qquad (2)$$

The load cell employs strain gauges and the HX711 amplifier to measure both compression and tension forces, enabling measurements in both upward and downward directions. When a force is applied, whether pushing down (compression) or pulling up (tension), the strain gauges deform, altering their electrical resistance utilizing the Wheatstone bridge circuit [24]. This change in resistance is amplified and translated into a voltage signal by HX711, providing a precise indication of the applied force. There are four different bar load cells each capable of measuring up to 1 kilogram, equivalent to 9.81 newtons, mounted on the platform.

During the calibration process, the readings from all the load cells are collectively treated as one, as they are calibrated to measure the reference weight as a single unit rather than individual readings to be averaged. This unified approach ensures consistency and accuracy across all load cells, aligning their measurements with the predetermined reference weight.

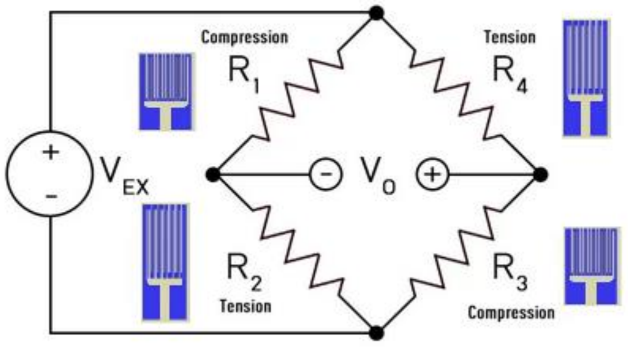


Figure 11:Load cells equivalent circuit [24]

6.1.2. Peripherals

In the setup, the main camera view is captured by a higher-resolution camera with 2K resolution (2560 x 1440) at 30 FPS (frame per second), providing sharper and more detailed images. This camera is positioned atop the training box, and it is complemented by using white LED lighting, ensuring optimal illumination for clear imaging. Additionally, a touchscreen monitor, and computer are integrated into the setup as part of the user interface and compatibility. The computer serves as the central processing unit for the system, and it is responsible for running the main program, managing data flow, handling image processing tasks, and powering all the hardware parts: monitor, microcontroller, camera, and LED light. Hence, achieving a portable, compact system.

6.1.3. Computer-Aided Design (CAD) Models

Both the PMW3389 and MPU6050 sensors were installed within an enclosed unit positioned atop the trocars. The housing unit and trocar were designed using Fusion 360 software. The trocar itself features two rings to facilitate pitch and yaw movements while remaining securely within the box. Moreover, a 3.4 mm-wide opening in the trocar ensures smooth tool insertion without friction, enabling the PMW3389 sensor to track the tool shaft's movement and rotation. For visual reference, the CAD models depicting the trocar design are provided in Figure 12. The sensor housing is designed to allow mechanical protection for the sensors while also eliminating light interference that could affect the optical sensor when detecting the tool shaft [23].

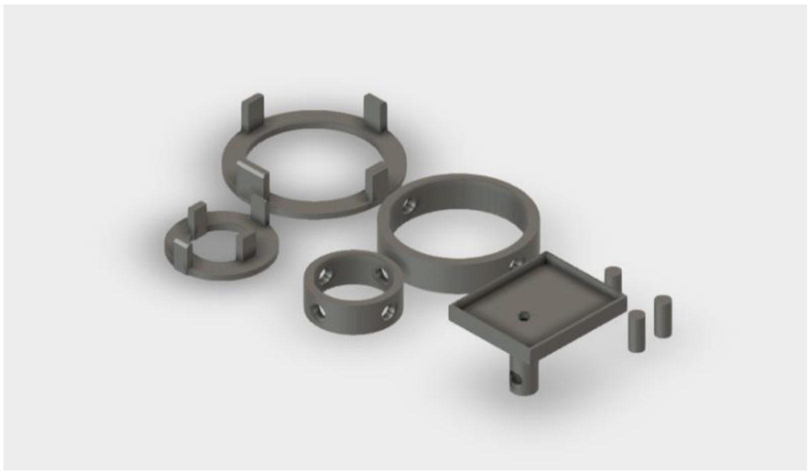


Figure 12: Trocar model the disassembled [23].

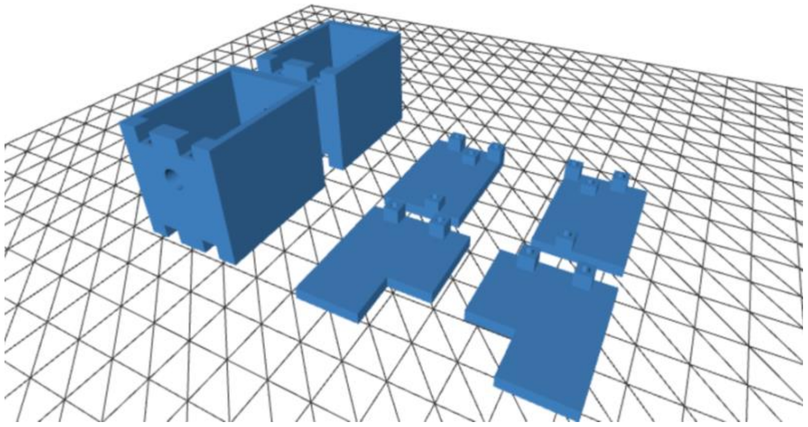


Figure 13:Housing units for the sensors disassembled [24].

The platform design is made of two plates with two main goals: smoothly adding new load cells and addressing stability concerns. To make room for the new load cells, added spaces and mounting points in the design. This ensures that the load cells are perfectly positioned for accurate force measurements, with fixing points both upward and downward to align with any applied force in those directions as shown in Figure 14.

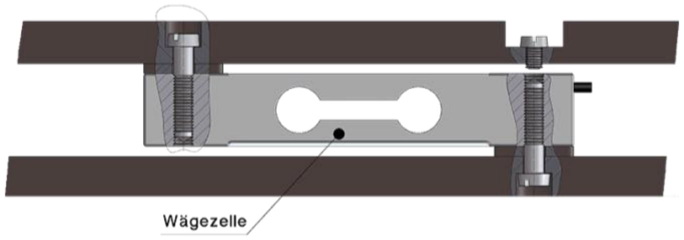


Figure 14: Mounting position for one of the new load cells [25].

Integrating the new load cells also required us to include support structures, like additional fixtures at the bottom plate, to secure and stabilize the amplifiers. These fixtures act as strong supports for the amplifiers, preventing any unintended movements or vibrations that could affect the accuracy of force readings.

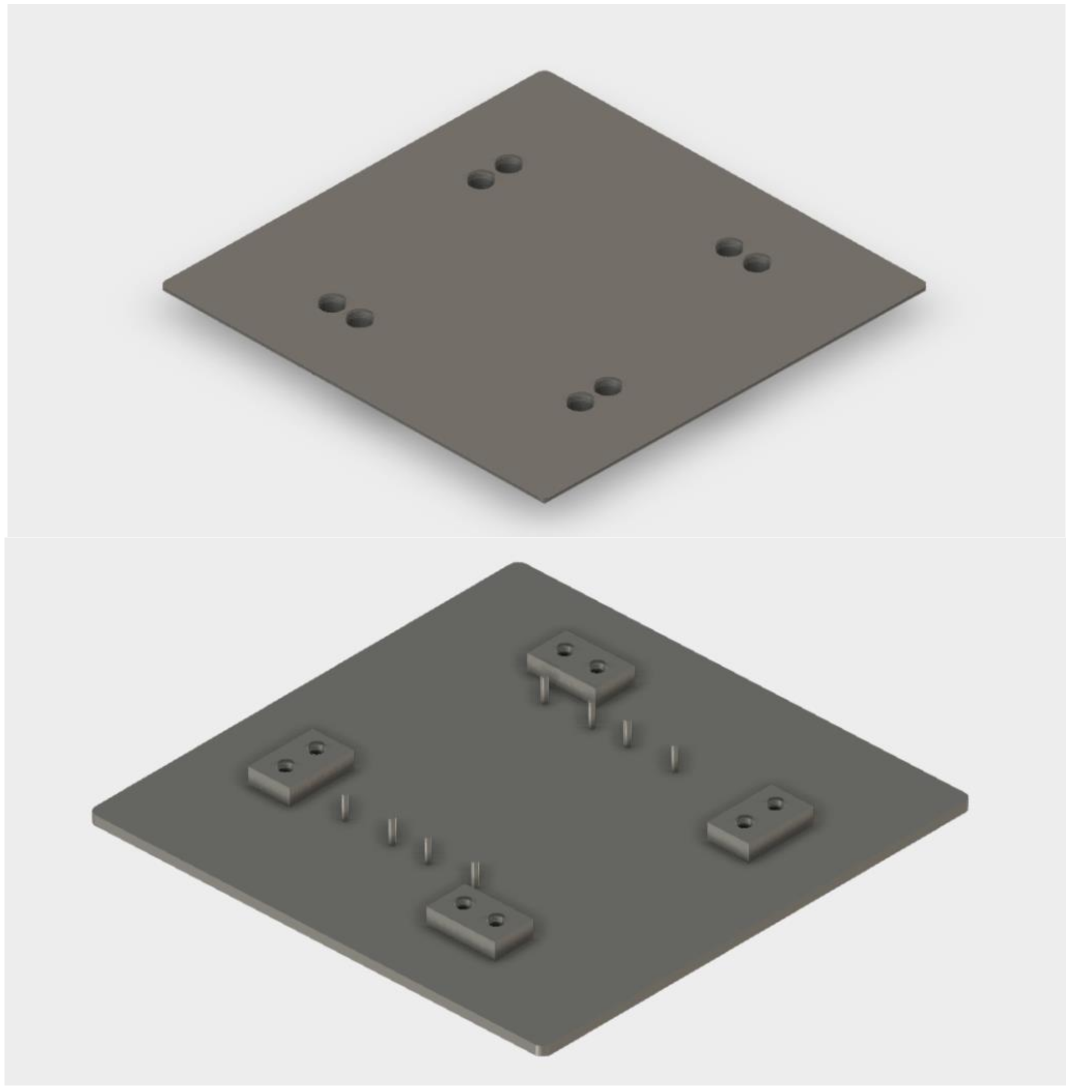


Figure 15: Top and base plates for the platform.

After soldering all sensors to a prototyping board. This prototyping board needed a housing box along with the Arduino to both reduce exposure of the circuit and tidy up the workspace for the user. The box contains a variety of entry and exit points for wires, as well as points to bolt the prototyping board to the housing using M2 screws.

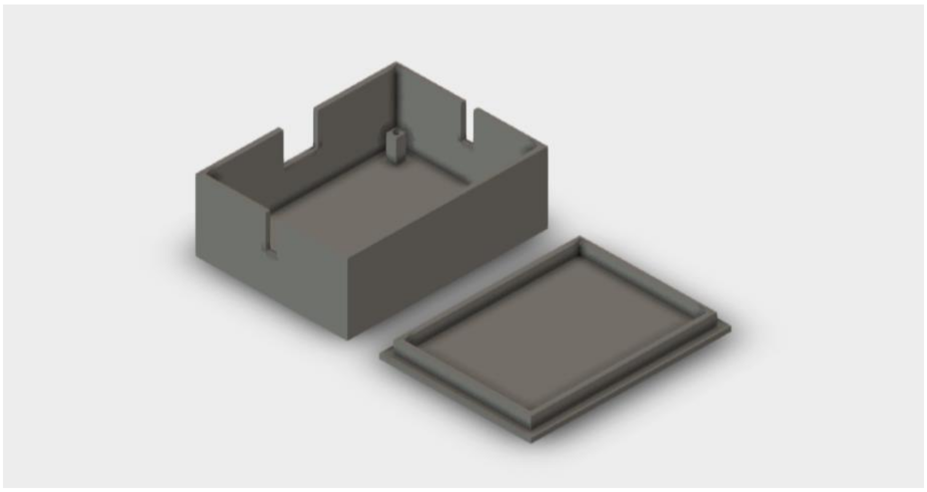


Figure 16: Housing unit for the microcontroller and the circuit.

The monitor has its components exposed therefore a case was necessary to cover them and provide protection to the monitor. The case dimensions are larger than the 3D printer leading to slicing the case in half to be able to print it and then glue them together using epoxy glue. In addition, a screen holder and a mount were 3D printed along with other miscellaneous parts such as cable management holders, and case fixtures which are provided in the project repository (Appendix A).

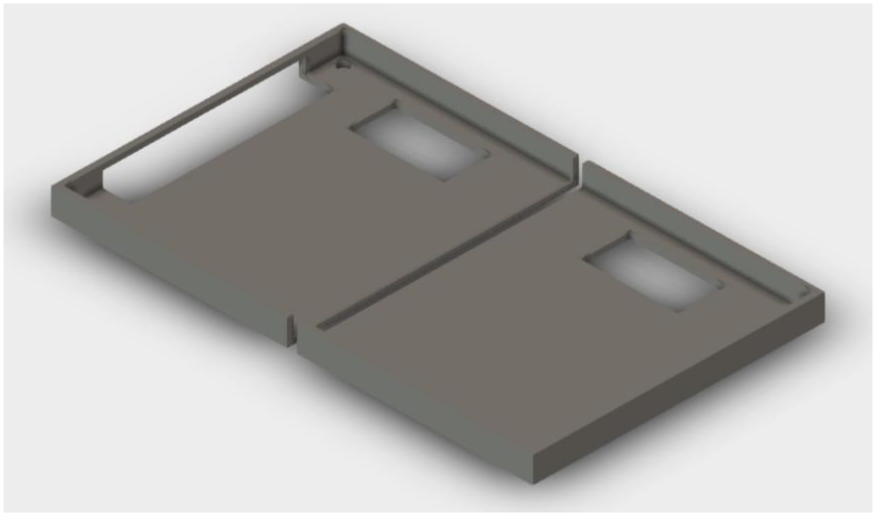


Figure 17:Screen case with holes for buttons and speakers.

6.2. Surgical Gesture Recognition

6.2.1. Task Description

Timely and accurate feedback is crucial for improving the efficacy of surgical training. However, the traditional approach, which relies on manual evaluation for feedback, can be subjective, inconsistent, and resource intensive. Additionally, manually labelling surgical gestures in training videos for feedback purposes is time-consuming and prone to human error. This limitation restricts delivering immediate, objective, and detailed performance analysis. The reliance on manual approaches to evaluate surgical training activities poses a substantial barrier to reaching the highest standards of surgical proficiency. This approach must be adjusted to maintain training efficiency and feedback quality. There is a clear need for a technology-driven solution that enhances the accuracy and reliability of performance assessments in surgical training by streamlining the feedback mechanism.

The proposed solution is to automate the gesture labelling process using state-of-the-art gesture recognition techniques. This automation aims to replace manual gesture labelling, traditionally performed by experts, and then recorded in spreadsheets for analysis. The method will specify, classify, and label different surgical gestures from training videos using deep learning-based gesture recognition methods. This process improves precision and consistency in gesture recognition and significantly reduces the time and effort involved in manual labelling.

To this end, our task is to develop an automated system using deep learning treal-timeccurately recognize and label surgical gestures in real time. This system will analyze surgical training videos, identify specific gestures, and label them without human intervention. Secondly, automated labelling should be used to evaluate trainees and provide them instant feedback quickly and objectively. The system will automatically analyze the trainees' performance, recognize their gestures, and automatically generate feedback information. This feedback will be systematically recorded in a structured format, such as an Excel sheet, for easy access and analysis.

The two main objectives guided this task are:

- Developing gesture modelling specifically tailored for training tasks.
- Implementing a deep learning-based gesture recognition method in our surgical simulator can support real-time applications.

Through this approach, the aim is not only to enhance the accuracy and efficiency of our model but also to contribute innovative solutions in gesture detection and tracking to the more comprehensive field of surgical training.

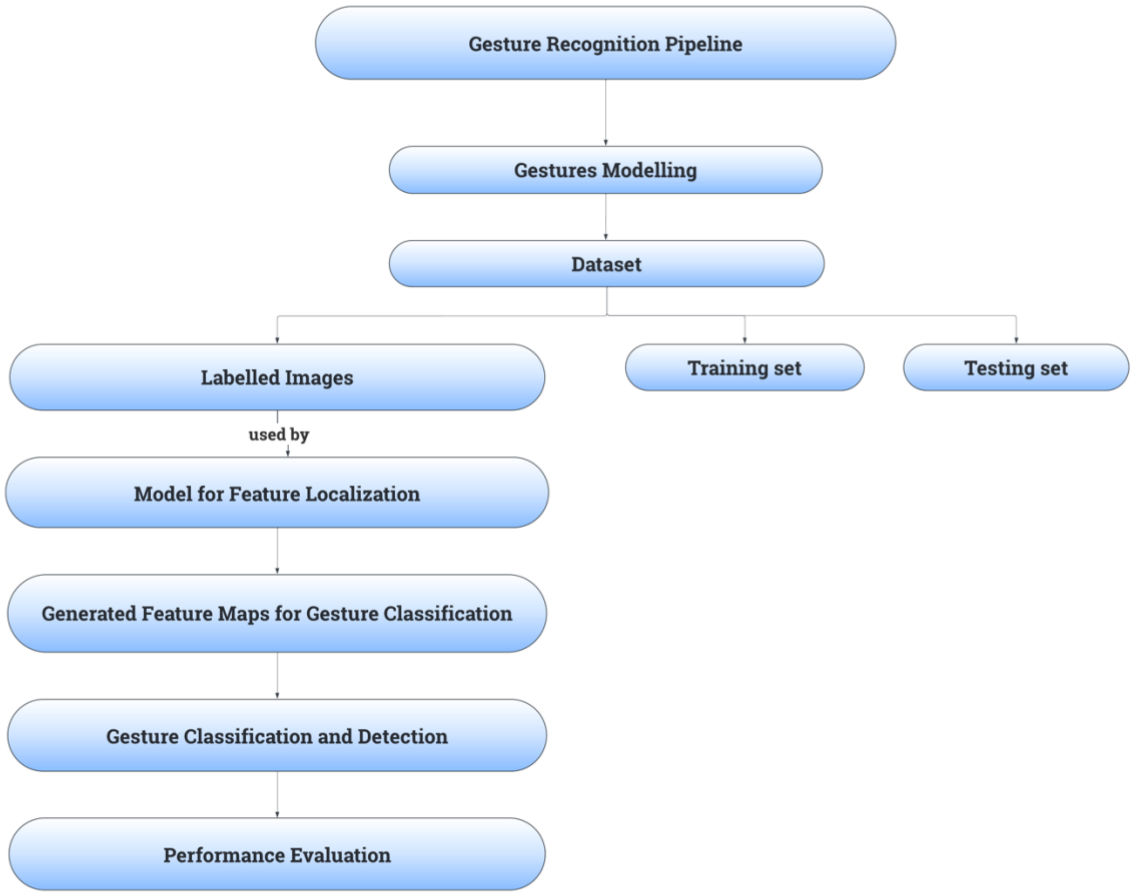

Gesture Recognition Pipeline
Gestures Modelling
Dataset
Labelled Images
Training set
Testing set
used by
Model for Feature Localization
Generated Feature Maps for Gesture Classification
Gesture Classification and Detection
Performance Evaluation

Figure 18: Gesture recognition pipeline.

6.2.1.1. Gesture Modelling

In creating the gesture model for the machine learning application, the focus was on describing the critical elements required for the surgical training simulations, such as identifying the left and right graspers and the rings and their various states and movements. To accomplish this, we outlined two fundamental attributes for each of these gestures:

- Location: This attribute precisely describes the grasper's pixel coordinates in the simulation to ensure proper spatial tracking.
- Movement: We recognized movement as an important relational attribute that captures the interactions between objects, such as holding the ring together with the left grasper, which indicates a movement of "Ring transfer. This attribute is required to recognize each action's purpose and context within the simulation.

As described in Table 3, the gesture model demonstrates a comprehensive list of potential movements and states. These include the idle state of the graspers (LG0 and RG0), various activities towards and away from the operating table (e.g., LG1, RG1), and interactions with the rings, such as clasping or transferring (e.g., LG3, RG6). This table provides a structured framework for the gesture's potential movements and relations within the simulation. Further, we used a semantic gesture model approach, which augments the essential attributes of location and movement with their semantic implications. This increases the machine learning algorithm's ability to study the activities within the simulation, moving beyond mere coordinate tracking to understanding the semantic purpose of each movement. For instance, the semantic model allows the algorithm to determine that "LG3": Left pick ring: Clasping ring represents an action where the left grasper picks and clasps the ring instead of identifying the spatial coordinates of the grasper and ring. The semantic gesture model facilitates gesture state detection by focusing on location, making it more efficient. This approach allows our machine learning model to accurately track location and analyze contextual relationships in surgical training simulations for real-time feedback and assessment. Based on the description of Peg transfer task, an ideal performance required the following sequence of gestures:

Peg Transfer: RG0→ RG1 → RG2 → RG3 → RG4 → RG5 → LG1 → RG6 → LG2 → RG7 → LG3→ RG8→ LG4→LG5→LG6→LG7

Table 3: The Gestures Classification by Movement and Location.

Gesture Index (Right) and (L: Left)	**Objects Movement**	**Jaws (Open or Closed)**
RG0	Ring on peg: Idle state (or hovering)	C
RG1	Right grasper: Moving vertically downwards towards peg on right side of operating table	O/C
RG2	Ring out peg: Positioning ring between the jaws and clasping it	O
RG3	Right pick ring: Moving upwards with clasped ring	O
RG4	Right Carry ring: Moving to center of operating table to meet left grasper	O
RG5	Right carry ring: Hovering with ring between jaws for left grasper to pick	O
RG6	Ring transfer: Holding ring together with left grasper	O
RG7	Right pick ring: Opening jaws to let go of ring	O
RG8	Moving aside	O/C
LG0	Idle state (or hovering)	C
LG1	Left grasper: Moving to center of operating table to meet right grasper	O/C
LG2	Left pick ring: Positioning ring between the jaws	O
LG3	Left pick ring: Clasping ring	O
LG4	Left carry ring: Moving horizontally (straight or diagonally) towards a peg on left side of operating table	O

Gesture Index (Right) and (L: Left)	**Objects Movement**	**Jaws (Open or Closed)**
LG5	Left carry ring: Positioning ring over peg	O
LG6	Ring on peg: Opening jaws and dropping ring through peg	O/C
LG7	Ring on peg: Moving aside after transferring all three rings	O/C

6.2.1.2. Deep Learning: Single Shot Detection (SSD)

After setting up a comprehensive gesture model, our next task is implementing gesture detection using deep learning, primarily via Single Shot Detection (SSD). A single-shot detector is a one-phase detector that predicts multiple classes. In other words, a real-time gesture detection framework that merges the tasks of object localization and type into a single unified architecture [27]. This section will discuss the training process as shown in Figure 18, including data preparation, annotation, and fine-tuning hyperparameters to optimize the model's performance for our specific use matter. Challenges encountered, methods used to overcome them, and the initial results will also be presented, delivering a clear view of the progress made and the steps forward.

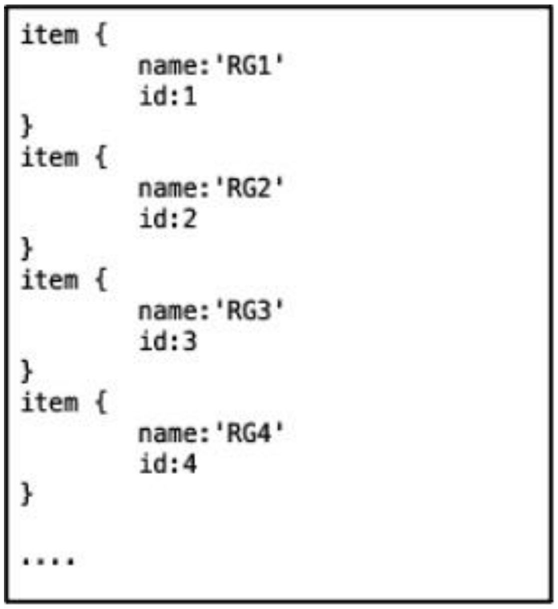
6.2.1.2.1. Dataset Collection and Preparation

For this task, we have created a dataset from laparoscopic training videos, each ranging from 75 seconds to 2 minutes. This dataset includes frames extracted from these videos, carefully annotated to function as a training foundation for our SSD model. The annotation process was facilitated using LabelImg, an open-source image annotation tool [28], which enabled the labelling process and resulted in each image having an associated .xml file. These .xml files were transformed into .csv files for an overview and. tfrecord files for use in the training pipeline. Upon completing the annotation process, we classified the images and their corresponding .xml, .csv, and. tfrecord files into separate datasets for training, testing, and evaluation, following an 80:20

split. This split is created to deliver a complete training experience while ensuring a thorough validation of the model's effectiveness.

6.2.1.2.2. Label Map

A required component of the dataset preparation was preparing a label map. This map is a directory for the training and detection algorithms, linking each gesture class to a specific integer value. For the dataset, we formulated a label map with 15 unique labels; each label corresponds to a particular laparoscopic peg transfer gesture, such as 'RG1' for the first gesture involving the right grasper, 'LG1' for the left grasper, and so forth, as shown in Figure 19 These labels are instrumental for the SSD model to recognize and categorize the precise gestures performed during the laparoscopic procedures. Then, the label map is saved in a. pbtxt file format and is required to train the SSD model to accurately identify and classify the different instruments observed during detection. Figure 19 illustrates an example of our label map configuration.



```
item {
        name:'RG1'
        id:1
}
item {
        name:'RG2'
        id:2
}
item {
        name:'RG3'
        id:3
}
item {
        name:'RG4'
        id:4
}

....
```

Figure 19: Label map file classes.

6.2.1.2.3. Data Augmentation

In the data augmentation phase of our project, we used the TensorFlow Object Detection API Image Preprocessor tool to introduce variations to our dataset [30]. This step was necessary and reassuring, as it improved the model's ability to perform under various states without compromising the integrity of the original data. Our augmentation techniques, including random horizontal flips and adjustments to brightness and contrast, are instrumental in training the model to accurately identify and classify gestures under various lighting conditions and orientations, thereby showcasing its robustness.

Including data augmentation options in our pipeline configuration emphasizes robust training. For instance, the `random_horizontal_flip` and `random_crop_image` options within our pipeline config file's `train_config` section introduce randomness and variability into the training process. This helps create a more adaptable dataset and significantly improves the model's generalization ability, making it more adept at recognizing gestures in previously unseen images. [29].

6.2.1.2.4. Model Architecture

For our model's foundation, we selected the SSD MobileNet v2 320x320, pre-trained on the COCO 2017 dataset, from the TensorFlow 2 Detection Model Zoo [29]. This model is a useful starting point for training on our dataset, effectively good speed, and precision for real-time recognition tasks. The architecture, shown in Figure 20], includes a feature pyramid network (FPN), improving the model's ability to accurately detect objects across different scales.

The SSD MobileNet V2 architecture uses depthwise separable convolutions—a computationally efficient option to standard convolutions—allowing for faster processing without a substantial loss in accuracy. This feature is important for real-time applications, as our project required immediate surgical gesture detection. As the model processes input frames, such as the 'Input for RG1' displayed, it generates bounding boxes at multiple scales, pinpointing the location of surgical gestures within the frame.

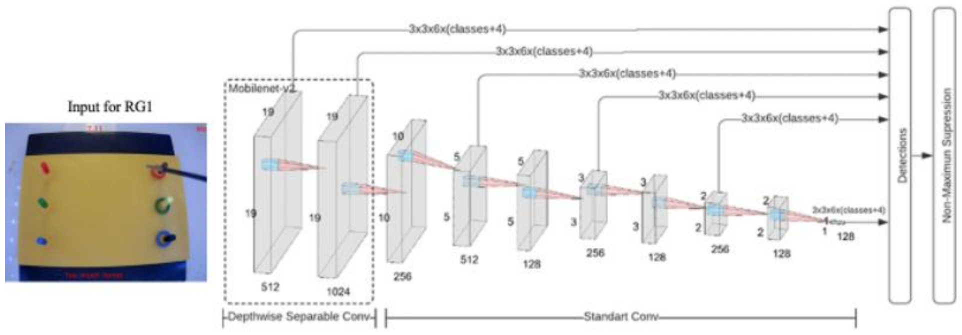


Figure 20: Real-time Surgical Gesture Recognition with SSD MobileNet v2 320x320 Architecture [29].

Our pipeline configuration refines our pre-trained model to classify 19 classes. To improve the model's efficiency and effectiveness in extracting features, we use the RELU activation function within the convolutional layers. Additionally, we have specified important hyperparameters such as the depth multiplier, minimum depth, and L2 regularization to ensure that the model performs optimally.

Our training process adopts a batch size of 4 and includes data augmentation techniques such as random horizontal flips and random cropping. These techniques help improve the model's generalization ability based on the training data. The process is also supported by a momentum optimizer with a cosine decay learning rate set to 11k steps, ensuring a robust learning curve. Our model's efficiency is an important aspect of its success. One feature is the weight-shared convolutional box predictor in the `box_predictor` settings. This feature reduces the computational burden by sharing weights across various feature maps, eventually optimizing the model's performance. Further, the model is equipped with a detailed anchor generator and post-processing configurations, which, coupled with the weight-sharing feature, allow it to reach high precision when detecting surgical gestures.

Overall, this setup, from the architectural choices to the training regimen, positions our model as a competent tool for detecting surgical gestures in real time and lays a solid foundation for advancing surgical training simulations.

6.2.1.3. Evaluation

We developed a system capable of identifying and categorizing various gestures in images. We then assessed it using several metrics, including precision and recall. Precision scores indicate whether the system makes accurate judgments, while recall allows us to determine whether it misses critical information. By analyzing these metrics, we evaluated the system's accuracy and completeness.

We tested our system to evaluate its ability to recognize and categorize differently. The results of this test are shown in Figure 21. Figure A shows the system's real-time output, demonstrating the detected gestures. At the same time, Figure B is a reference for the system's expected performance, showing the manually labelled gestures. We found that the system identified certain gestures with confidence, such as 'RG0' for the right grasper in idle position and 'LG5' for the left grasper positioning the ring over the peg, as depicted in the annotated test images in Figure 23. Initially, the system's detection of these gestures is quite good, with 'RG0' identified with a 52% confidence level and 'LG5' at 68%. These percentages reflect the precision of our system, which is its ability to label gestures it sees in the training videos correctly.

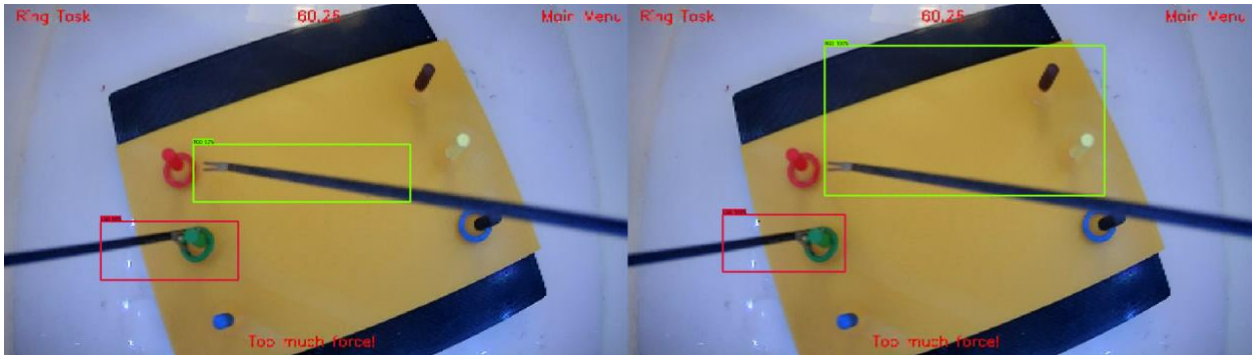


Figure 21: Comparative Analysis of Automated Accurate Gesture Detection: SSD MobileNet v2 320x320 vs. Manual Labeling. Figure (A) on the right and Figure (B) on the left.

During a recent test, the system demonstrated a successful detection, with the model identifying the correct gestures shown in Image C Figure 22. However, the system encountered some limitations in its performance. For example, as shown in Figure 22, Image D, the system incorrectly identified a gesture as 'RG8' with a 42% confidence score instead of recognizing it as the correct 'left gesture. This highlights the need for further system accuracy enhancements, especially in critical applications like surgical training, where precision is paramount.

To improve the system's performance, we need to expand the training set and fine-tune the model parameters. This will help us achieve a better balance between precision and recall, which is crucial for developing a robust design that can reliably support the complex requirements of gesture recognition in various environments.

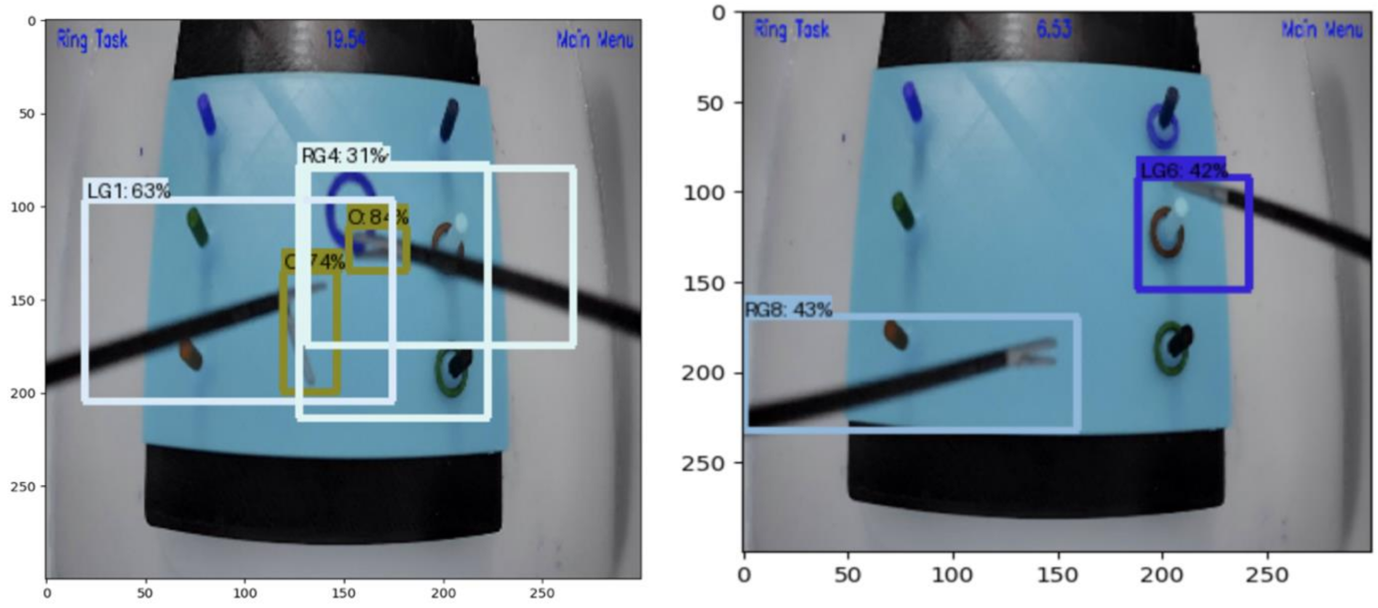


Figure 22: Comparative of Automated Accurate Gesture Detection on the right (Image C) and inaccurate on the left Image(D).

6.2.1.4. Results

During the testing stage, we developed a detailed script to automate the detection and labelling of surgical gestures from video data. The script loaded the pre-trained model, processed each video frame, and applied the detection function to identify gestures within the footage. Whenever a gesture was detected with adequate confidence, the script recorded the event in a CSV file and its timestamp, creating an organized log for performance analysis as shown in Table 4.

Table 4: The automated output logging Gesture.

Time Start	Time Stop	Right Gesture	Left Gesture
	0.0666666666666670	RG0	LG0
0.0666666666666670	0.1333333333333300	RG0	LG6
0.1333333333333300	0.2666666666666700	RG0	LG0
0.2666666666666700	0.333333333333330	RG0	LG6
0.333333333333330	0.4	RG0	LG0
0.4	0.666666666666670	RG0	LG6

6.2.2. Model Improvement

Timely and accurate feedback is crucial for improving the efficacy of surgical training. However, the traditional approach, which relies on manual evaluation for feedback, can be subjective, inconsistent, and resource intensive. Additionally, manually labelling surgical gestures in training videos for feedback purposes is time-consuming and prone to human error. This limitation restricts delivering immediate, objective, and detailed performance analysis. The reliance on manual approaches to evaluate surgical training activities poses a substantial barrier to reaching the highest standards of surgical proficiency. This approach must be adjusted to maintain training efficiency and feedback quality. There is a clear need for a technology-driven solution that enhances the accuracy and reliability of performance assessments in surgical training by streamlining the feedback mechanism.

6.2.2.1. Integrating Spatiotemporal Dynamics with Self-Supervised Learning in Video Analysis

The previous method of analyzing surgical training videos focused on spatial features. This model involved examining each frame separately to detect gestures and movements, like reviewing a series of snapshots independently to deduce what is happening. This technique can be effective for recognizing certain gestures based on visual cues within a single frame but lacks the contextual understanding of movement over time. Understanding complex surgical gestures that unfold across several frames is important. Hence, a more comprehensive approach is needed to provide a better understanding of the movement of surgical instruments over time. Such an approach would require analyzing the gestures and movements in the context of the entire surgical procedure rather than examining each frame independently. The proposed system's architecture captures the intricate movement patterns of surgical procedures by breaking down the video clip into smaller clips. This results in a detailed timeline of movement. Figure 23 shows that the process starts by pre-processing peg transfer task videos, which are then sub-sampled and resized to ensure compatibility with the model's input requirements. These videos' extracted clips and volumes serve as the foundation for analyzing the short-term (ST) and long-term (LT).

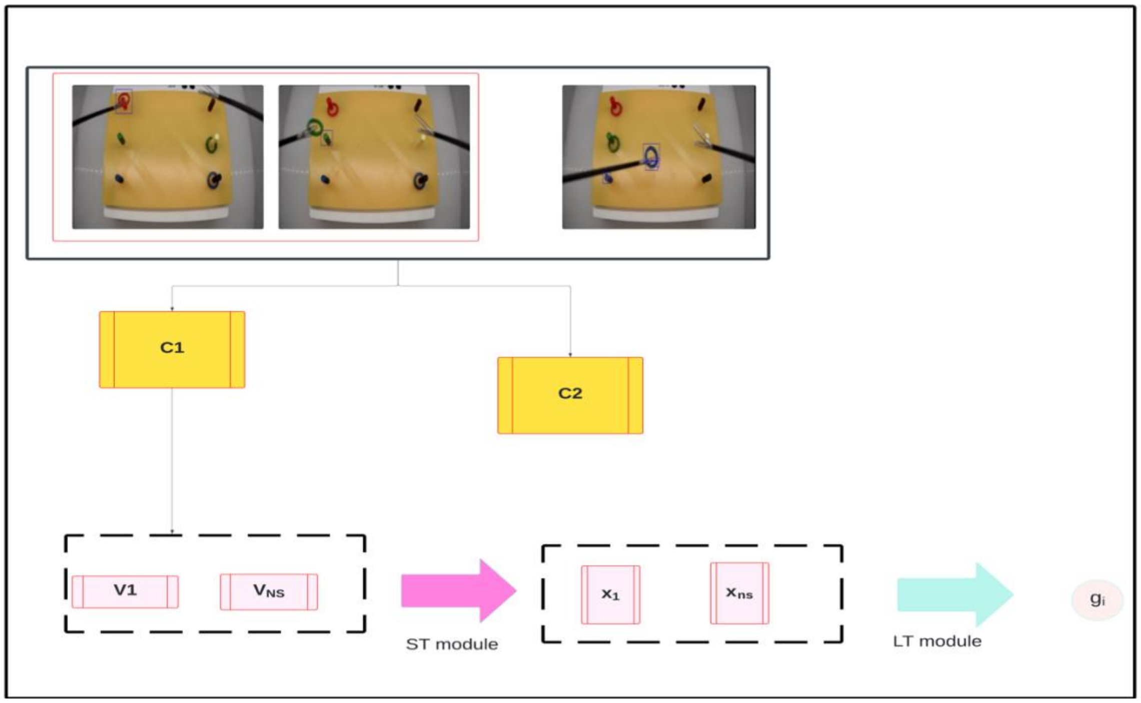


Figure 23: Workflow of Gesture Recognition Architecture.

Our primary objectives as mentioned in the task description section remain the same:

- Tailored Gesture Modelling for Training.

Our first objective was to develop a gesture modelling system tailored to the peg transfer training task. However, we have reduced the number of gesture indexes to improve the model. This refinement is reflected in the updated gesture classifications, as shown in Table [2], resulting in a more precise and defined set of classes. We simplified the model without compromising its complexity, enhancing its clarity. One of the main challenges in gesture recognition in this context is the repetitive nature of certain movements, making it difficult for the model to distinguish between different gestures accurately. However, our focused approach ensures that the model remains proficient and nuanced in its operational output. The updated gesture model is now more user-friendly and manageable, making it an effective training tool. Based on the description of the peg transfer task, an updated performance required the following sequence of gestures:

PT: RG0→ RG1 → RG2 → LG1 → RG3 → LG2→ RG0 → LG3 → LG4

Table 5: The Modified Gestures Classification by Movement and Location.

Gesture Index	Gesture Description	Jaws (Open or Closed)?
RG0	Idle state (or hovering) or moving aside	C
RG1	Moving vertically downwards towards peg on right side of operating table and Positioning ring between the jaws and clasping it	O/C
RG2	Moving to meet left grasper	O
RG3	Handing ring to left grasper to pick	O
LG0	Idle state (or hovering) or moving aside	C
LG1	Moving to center of operating table to meet right grasper.	O
LG2	Positioning ring between the jaws and clasping it and moving horizontally (straight or diagonally) towards a peg on left side of operating table	O
LG3	Positioning ring over peg and dropping ring through peg	O
LG4	Moving aside after transferring all three rings	O

6.2.2.2. Labelled Data

We selected a single video to create a labelled dataset, focusing on the gestures necessary for the peg transfer task. We used the Computer Vision Annotation Tool (CVAT) to label each gesture in the video precisely. We classified these gestures into 12 different categories, such as

RG1 to RG3 for actions performed with the right grasper, LG1 to LG4 for movements with the left grasper, 'O' and 'C' for the open and close states of the tool jaws, and 'unknown' for any gestures that did not fit the predefined classes. Through annotation, we ensured high-quality, accurately labelled data for a robust learning foundation.

6.2.2.3. Unlabelled Data

We used an unlabeled video in combination with our labelled dataset to improve the performance of our model using a semi-self-supervised approach. The unlabeled video experienced the same preprocessing steps as the labelled one without detailed gesture annotations. Our decision to include the unlabeled data was a strategic move to enhance the model's ability to learn from unannotated segments due to the low amount of labelled data. This approach can reveal patterns and features not explicitly marked in the labelled dataset.

6.2.2.4. Video Dataset Processing

We needed to process these videos to ensure they were compatible with our model's requirements. The videos were recorded at 15 frames per second (fps), but we subsampled them to 25 fps and resized them to 224 × 224 pixels to fit the pretrained model. This resizing made the videos appropriate for the model's input shape and improved the training process's computational efficiency.

After the initial preprocessing, the videos were divided into smaller clips and volumes for a more detailed analysis. The clips were extracted with a step size of 3 frames, each lasting 15. The volumes were further decomposed with a step size of 1 frame, resulting in 8 frames per volume. This split allowed the model to examine the surgical tasks at different detail levels, from comprehensive clip-based movements to better volume-based gestures [31].

These clips and volumes were obtained from labelled and unlabeled datasets and fed into different CNN architecture modules. As shown in Figure 23, they performed as the foundational data for training our model's short-term (ST) and long-term (LT) components. This data annotation, labelling, and preprocessing approach ensured a comprehensive and nuanced dataset for training an advanced gesture recognition model for our training simulations.

6.2.2.5. Short Model: Semi Self-Supervised Learning with ST Model

The ST module uses a self-supervised learning strategy and an encoder-decoder mechanism to extract spatiotemporal features. This approach is designed to reduce the reliance on much-annotated datasets. Hence, the model training process is streamlined, especially for complex tasks such as surgical gesture recognition. As shown in Figure 24, in the Short-Term (ST) module of the model, the process starts with the input video volume (V). This volume transforms the Inception3D (I3D) network, operating as an encoder. The I3D network compresses (Vi) into a deep feature vector (Xi), capturing the spatial features and the initial temporal details within the sequence.

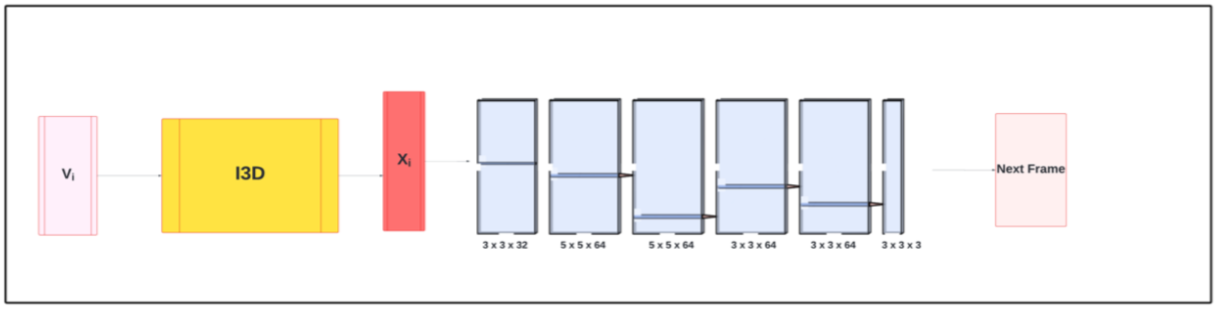


Figure 24: The Short Module Architecture.

6.2.2.5.1. Encoder: Inception3D (I3D) Network

Our encoder is the Inception3D (I3D) architecture. We selected this architecture for its ability to understand spatial features within a temporal frame. The I3D encoder is pre-trained on the Kinetics-400 dataset and is trained in processing RGB inputs, providing a reliable foundation for our model to build upon. This encoder compresses a video volume into a compact feature vector, encapsulating the spatiotemporal features needed for the next decoding phase [32].

6.2.2.5.2. Decoder: TransConv2D Architecture

As shown in Figure 24, In our model, the decoder is responsible for reconstructing subsequent video frames from the compact feature vectors encoded by the I3D network. This is achieved through a sequence of TransConv2D layers that allow the model to learn the complexities of surgical gesture dynamics over time. The deep features (Xi) progress through the TransConv2D layers, with each layer using ReLU activation and batch normalization to improve refinement. Dropout layers are also included to prevent overfitting, ensuring the model generalizes well to new, unseen data.   The model restores the spatial details condensed during the I3D encoding phase by upscaling the feature maps through the decoder. This ensures the model comprehends the spatial and temporal nuances of the surgical gestures. The final step, the final layer, a Sigmoid activation function is used to output the reconstructed frame, normalized between 0 and 1. This output mimics the next frame in the sequence, which teaches the model to learn the flow of movement in surgical tasks. [31]

6.2.2.5.3. Semi Self-Supervised Training Approach

We aim to reduce the need for manual annotation efforts, and we achieve this with the ST Model by using a semi-self-supervised training methodology. The model uses a significant portion of unlabeled data for training and learns to predict the next frames, therefore understanding gesture dynamics autonomously. Additionally, we use a smaller subset of labelled data, annotated with tools like CVAT, to fine-tune the model and ensure accurate gesture recognition.

During the model training, a strategic approach called pseudo-labelling is applied. This technique involves the model assigning predicted labels to unlabelled data using the knowledge it acquired through training. A "BalancedBatchSampler" creates training batches, ensuring a balanced mix of labelled and pseudo-labelled data. The model is evaluated multiple times to optimize its accuracy and prevent overfitting. This evaluation process involves implementing an early stopping protocol and using checkpoints to capture and retain the model's most effective states.

6.2.2.6. Integration with LT Module

Due to time constraints, the Long-Term (LT) module has not been fully implemented. However, its implementation is planned for future improvements to enhance the model's ability to capture long-term dependencies in surgical gestures. The LT module will use a transformer architecture and feature vectors generated by the ST model to analyze sequences of gestures over extended periods, providing a comprehensive analysis of surgical tasks. The ST Model is an improvement for our gesture recognition technique. It uses a semi- self-supervised training approach, which reduces reliance on comprehensive annotations while improving the model's ability to learn complicated spatiotemporal patterns in peg transfer procedures. This method improves the model's performance and makes adapting to new tasks and datasets easier.

6.2.2.7. Evaluation and Analysis of ST Model

Evaluation and Analysis of ST Model The confusion matrix in Figure 25 is an evaluation tool for our ST Model. It quantifies the model's predictions, with the rows representing the actual classes and the columns depicting the predicted classes. Diagonal elements denote accurate predictions, whereas off-diagonal elements indicate misclassifications. For example, class 'RG0' (index 0) and class 'RG1' (index 1) correctly show most of their predictions on the diagonal, indicating high accuracy for these gestures. Conversely, overlaps in predictions between 'LG2' (index 6) and 'LG4' (index 8), or between 'LG3' (index 7) and 'O' (index 9), indicate challenges the model faces in distinguishing these gestures due to their feature similarity. Also, this matrix shows that while the model can correctly predict the 'C' gesture for closing jaws (index 10) with high precision, it also frequently confuses the 'unknown' class (index 11) with 'LG4' (index 8), which could indicate a fault in the representation of the unknown class within the training data.

After analyzing the confusion matrix, we can identify the categories where the model performs well and those where it struggles. We can expand the dataset by adding more diverse samples of the 'unknown' gesture to improve the model. This will expose the model to a wider range of gestures not included in the predefined categories, resulting in more accurate classification. Also, we can review and refine the feature extraction process to better distinguish between similar classes. These measures aim to improve the model's generalization ability and increase the accuracy beyond 60%.

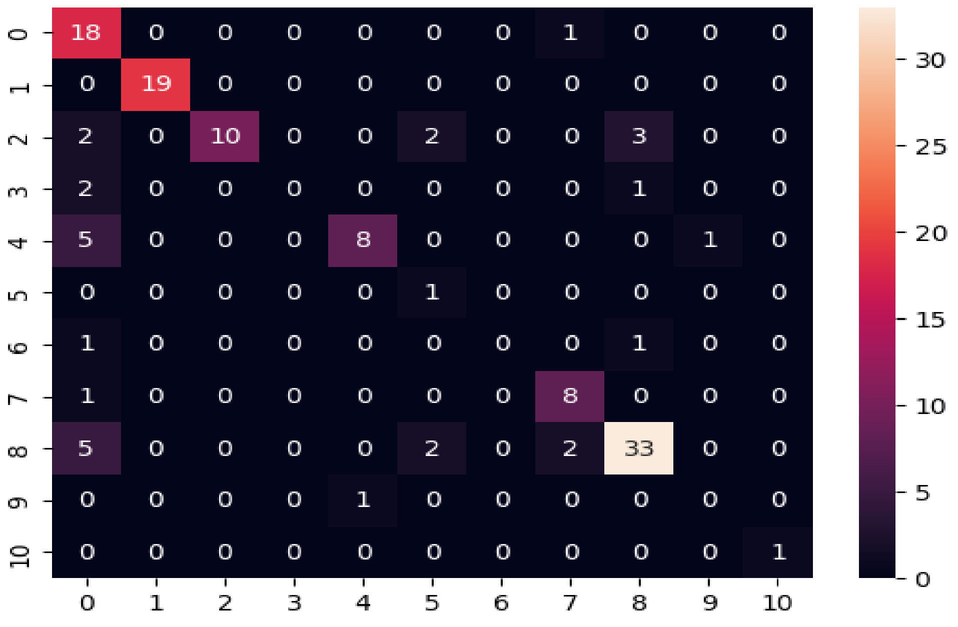


Figure 25: Matrix of ST Model.

6.3. Data Processing Pipeline

6.3.1. Initial Processing

In the latest development of our laparoscopic surgery simulator, the processing of real-time data during simulations is efficiently managed by an Arduino microcontroller. This setup is ideal for seamless data acquisition from each sensor through a signal controller. The system captures a comprehensive dataset, including rotation, position, speed, acceleration, and force parameters. A significant step in this phase is differentiating between each metric's various measures. Text files are used to store the raw data collected from the sensors. Therefore, a Python script was developed to facilitate the rapid processing of this data. This script includes a dialogue window using the 'tkinter' library that allows users to select the appropriate text file containing sensor data for cleaning. Following the selection, the data is cleaned, organized into a DataFrame, and saved as a CSV file using the 'pandas' library, enhancing the efficiency of data visualization and preparation for subsequent algorithms and processes.

6.3.2. Data Segmentation

The subsequent phase in our data processing and preparation pipeline involves a critical step: segmentation. This process entails dividing the entire procedure into three distinct subtasks. Each ring is a subtask with a total of 3 rings. Such segmentation proves immensely beneficial for multiple aspects of our system:

A. Enhanced Algorithm Feeding: By breaking down the procedure into smaller, manageable subtasks, we can feed data into our algorithms in a more structured and effective manner. This segmentation allows the algorithms to process data in chunks that are more representative of specific aspects of the procedure, leading to more accurate and relevant outcomes.

B. Refined User Performance Evaluation: Segmenting the procedure enables a more granular evaluation of the user's performance. By analyzing each subtask independently, we better understand the user's strengths and areas needing improvement. This detailed approach ensures that performance assessments are holistic and nuanced, capturing the complexities of the user's interactions with the simulator.

C. Targeted Feedback for Improvement: One of the primary objectives of our simulator is to provide constructive feedback to users. The feedback can be specifically tailored to each segment by dissecting the procedure into subtasks. This targeted feedback is instrumental in guiding users on improving their performance in each specific area of the procedure, ultimately enhancing their overall skill set.

Segmenting the data into these subtasks was executed using Python. Here is an overview of how this was achieved:

1. Class Initialization - 'DataSegmentation': This class forms the foundation for segmenting the data. It initializes with three critical file paths: 'CleanedSensorData_Ref.csv' for the expert's cleaned sensor data, 'user_file_path' for the user to select their procedure data through a file dialogue interface, and 'Labelled Videos.xlsx' for essential labelled information used in segmenting with deep learning methods used in section 6.2.

2. Loading Data - 'load_data' Method: This method is vital for importing necessary datasets into the Python environment for processing. It uses 'pandas,' a powerful data analysis library, to read cleaned reference and user data from CSV files and the labelled information from Excel sheets. The data from two procedure videos, one for an expert and one for the user, are loaded separately with their respective labels, which is critical for comparative performance analysis.

3. Processing Segments - 'process_segments' Method: This method takes the loaded data and labels as inputs, creating data segments based on specified timestamps in the labels. It iterates through each row in the labels DataFrame, filtering corresponding rows. This step is crucial for isolating specific portions of the time series data for focused analysis on defined intervals of each subtask.

4. Combining and Saving Segments - 'combine_and_save_segments' Method: This method consolidates individual segments into more significant task-specific segments. Segments are grouped using predefined ranges based on the labels Excel sheet, forming a comprehensive dataset for each subtask. These concatenated segments are then saved as new CSV files, named to indicate the source video and the specific subtask segment. This organization of data is vital for subsequent independent analysis of each subtask.

5. Execution - 'run' Method: This method orchestrates the entire segmentation process. It calls the load_data method to import datasets, process_segments to create individual time-based segments and then combine_and_save_segments to compile these into subtask-based datasets. This method processes reference and user data, setting the stage for comparative analysis using the Dynamic Time Warping (DTW) algorithm in later stages.

6. Overall Significance: The segmentation phase is integral to the project as it structures the data into meaningful and analyzable parts. Breaking down the data into specific segments based on labelled time intervals allows for more precise and focused analysis for each subtask. This structured format is essential for the upcoming steps, where the DTW algorithm will be applied to each segment for detailed time series analysis and comparison.

In summary, the data segmentation process is a systematic and crucial step. It transforms the cleaned data into well-defined segments, preparing it for the following intricate analysis, including the application of the DTW algorithm and the subsequent evaluation of user performance against the reference data, as well as for better visualization.

6.4. Performance Evaluation

6.4.1. Time Series Alignment and Performance Comparison in DTW

Data mining involves sifting through raw data to uncover underlying patterns and extract valuable insights [33]. This process can vary significantly among individuals, leading to notable differences in the duration and pacing of data analysis between users and experts in a laparoscopy surgery setting. Dynamic Time Warping, or DTW, would come into play in this regard. DTW is a method used to calculate the optimal matching between two sequences that may not be perfectly synchronized, often temporal in nature [34]. This technique is prevalent in various fields, including data mining, speech recognition, and financial markets.

DTW leverages dynamic programming to find the best temporal match between elements of a two-time series. An advanced form of DTW, online and dynamic time warping, has been proposed to enhance time series data mining efficiency. This approach involves segmenting a long time series into shorter subsequences and then applying DTW to measure the similarity of each pair of subsequences [35]. To effectively compare a user's performance with an expert, it is essential to align the user's time series signals with the reference ones, simulating concurrent completion of the procedure. The alignment process is structured as follows [36]:

A. Each index in the first sequence must correspond to one or more indices in the second sequence, and the reverse must also hold true.

B. The initial index of the first sequence should align with the first index of the second sequence. However, this need not be its exclusive match.

C. Similarly, the final index of the first sequence should align with the last index of the second sequence without being restricted to a singular match.

D. The mapping from indices of the first sequence to those of the second must show a monotonically increasing relationship, and the same applies in reverse. This means if 'j > i' are indices from the first sequence, there must not exist two indices 'l > k' in the second sequence such that index 'i' aligns with 'l' and 'j' aligns with 'k,' and vice versa.

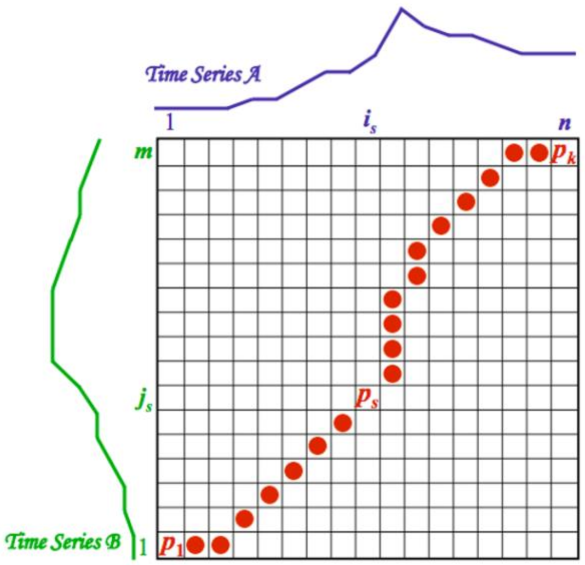
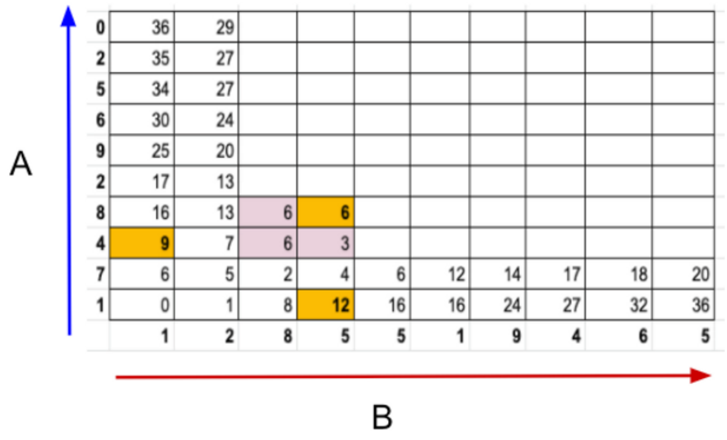


Figure 26: Series A and B distance normalized over time [36].

Mathematically, DTW arranges two sequences, X and Y, into an n-by-m grid. Each point in this grid represents the alignment between elements of X and Y [36]. The warping path W is then computed to map the elements of X and Y to minimize the distance between them. This path is essentially a sequence of grid points.

$$D_{min}(i_k, j_k) = \min_{i_{k-1}, j_{k-1}} D_{min}(i_{k-1}, j_{k-1}) + d(i_k, j_k | i_{k-1}, j_{k-1}) \qquad \textbf{(3)}[36]$$

Calculating the optimal path to a specific point (i_k, j_k) involves using Euclidean distance. Subsequently, the overall path cost is calculated, representing the cumulative distance over the warping path. Here is an actual example of how it works:



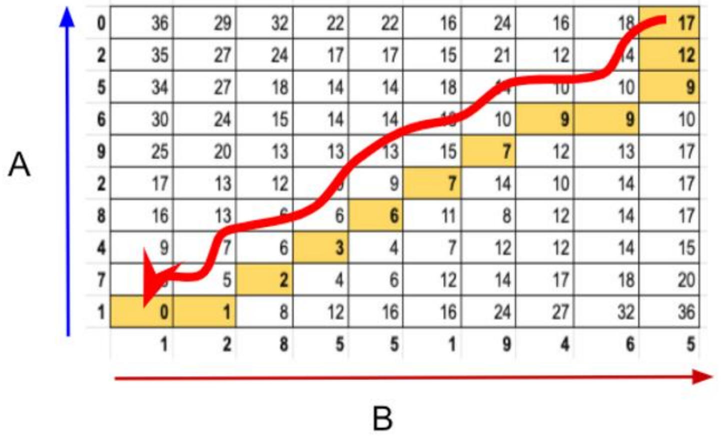
0	36	29								
2	35	27								
5	34	27								
6	30	24								
9	25	20								
2	17	13								
8	16	13	6	6						
4	9	7	6	3						
7	6	5	2	4	6	12	14	17	18	20
1	0	1	8	12	16	16	24	27	32	36
	1	2	8	5	5	1	9	4	6	5

Figure 27.: DTW calculation example [36]

Consider two-time series, A and B, represented on the A and B axes. To calculate the value in the yellow highlighted block (6), we use the following equation:

$$M(i,j\ ) = |A(i) - B(j)| + \min(M(i-1,j-1),\ M(i,j-1),\ M(i-1,j)) \qquad \textbf{(4)}[36]$$

For example, |8 –5| + min(6, 6, 3) = 6. Once the calculation for the entire matrix is completed, we compute the warping path to obtain the DTW distance, as illustrated in the following figure:



0	36	29	32	22	22	16	24	16	18	17
2	35	27	24	17	17	15	21	12	14	12
5	34	27	18	14	14	18	[illegible]	10	10	9
6	30	24	15	14	14	[illegible]	10	9	9	10
9	25	20	13	13	13	15	7	12	13	17
2	17	13	12	[illegible]	9	7	14	10	14	17
8	16	13	6	6	6	11	8	12	14	17
4	9	7	6	3	4	7	12	12	14	15
7	[illegible]	5	2	4	6	12	14	17	18	20
1	0	1	8	12	16	16	24	27	32	36
	1	2	8	5	5	1	9	4	6	5

Figure 28: Cost matrix with its final path [36].

The paths are chosen based on the minimum value criterion: The DTW distance, D, is calculated as the average of the computed values over the total number of chosen paths:

$$D = \frac{17\ +\ 12\ + 9\ +\ 9\ +\ 9\ +\ 7\ +\ 7\ +\ 6\ +\ 3\ +\ 2\ +\ 1\ +\ 0}{12} = 6.83$$

Moreover, DTW implementation requires consideration of certain constraints to ensure efficiency and accuracy. These include the boundary condition, which ensures the warping path starts and ends with both signals' start and end points, which is shown in the aligned signal in Figure 30; the monotonicity condition, preserving the time order of points; and the continuity condition, which limits the path transitions to adjacent points in time. The warping window condition can also restrict allowable points within a specific width.

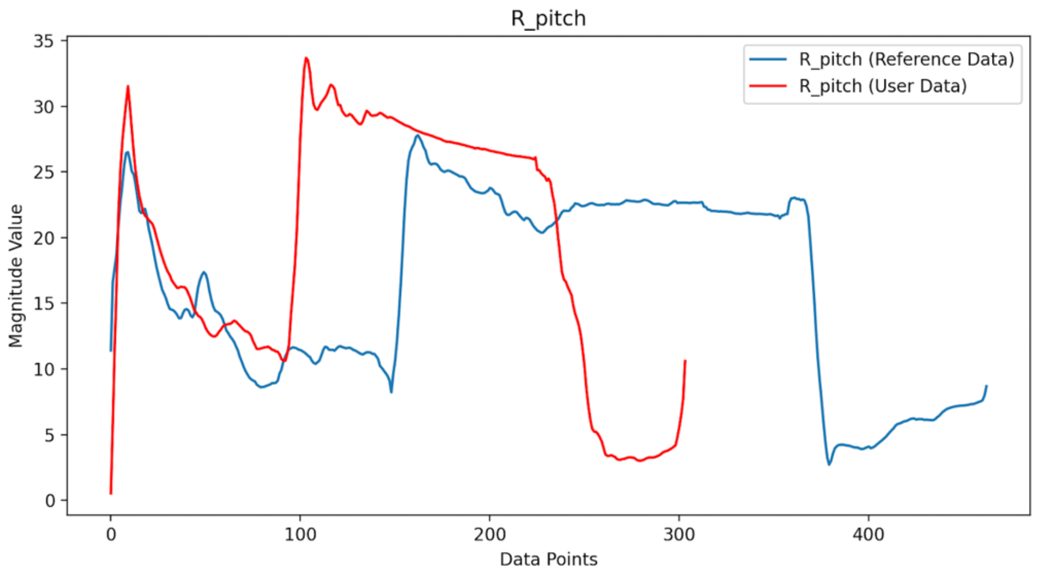


Figure 29: Right laparoscopic surgery grasper pitch movement before applying DTW.

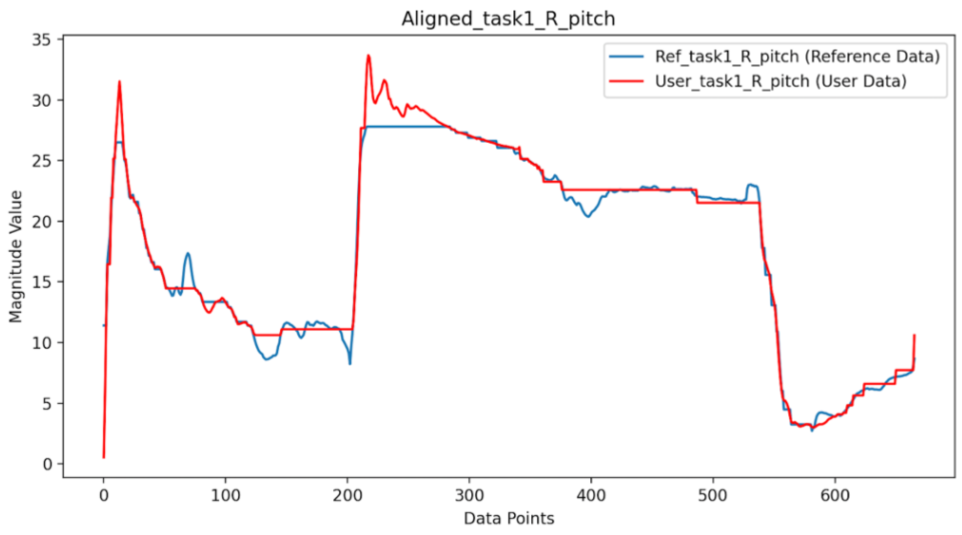


Figure 30: Right laparoscopic surgery grasper pitch movement after alignment using DTW.

In summary, DTW is a powerful technique in data mining for aligning and comparing time series data, especially when there are variations in timing or speed between sequences. Its mathematical foundation and constraints make it a robust and efficient method for analyzing temporal data from the laparoscopy surgery simulator sensors.

6.4.2. TaskPerformanceAnalyzer: Python Class for Time Series Analysis

A Python Class for Time Series Analysis, the TaskPerformanceAnalyzer class is designed to analyze and compare performance in tasks involving video data and associated time series signals. Upon initialization, the class takes paths to the segmented reference and user data in CSV and video formats that were done in section 6.3. It uses the 'pandas' library to load and process the CSV data, dropping unneeded columns and recording signal column names. The class features a custom method, dtw_distance, to compute the Dynamic Time Warping (DTW) distance between two-time series, as explained earlier. This function establishes a matrix and iteratively calculates the minimum distance, reflecting the series's similarity despite potential timing or speed variances. The align_data method employs the 'fastdtw' function, an efficient DTW implementation for data alignment. This method aligns the reference and user data based on DTW distances and saves the aligned data into a new CSV file.

The class also includes a step for normalizing to keep the data well-behaved and processing data in fixed size moving windows, typically set to 10 seconds for better analysis. It calculates the DTW distance for each window, categorizing performance and saving results, including statistics like mean, maximum, minimum, and standard deviation of DTW distances, for further analysis. In the video processing stage, the class uses the DTW distance analysis to pinpoint segments in the reference and user videos where the user's performance significantly diverges. This step is done by setting a threshold of the mean DTW distances in all moving windows added to one standard deviation so that any distance that exceeds this threshold will be detected and presented to the user in the upcoming stages. Then, these segments are extracted using the ffmpeg_extract_subclip function from the 'moviepy' library. The extracted clips, highlighting weak performance areas compared to the expert data, are then combined, and annotated with text labels for clarity, as shown in Figure 31 (Youssef is the user, and Atallah is the expert in this case)

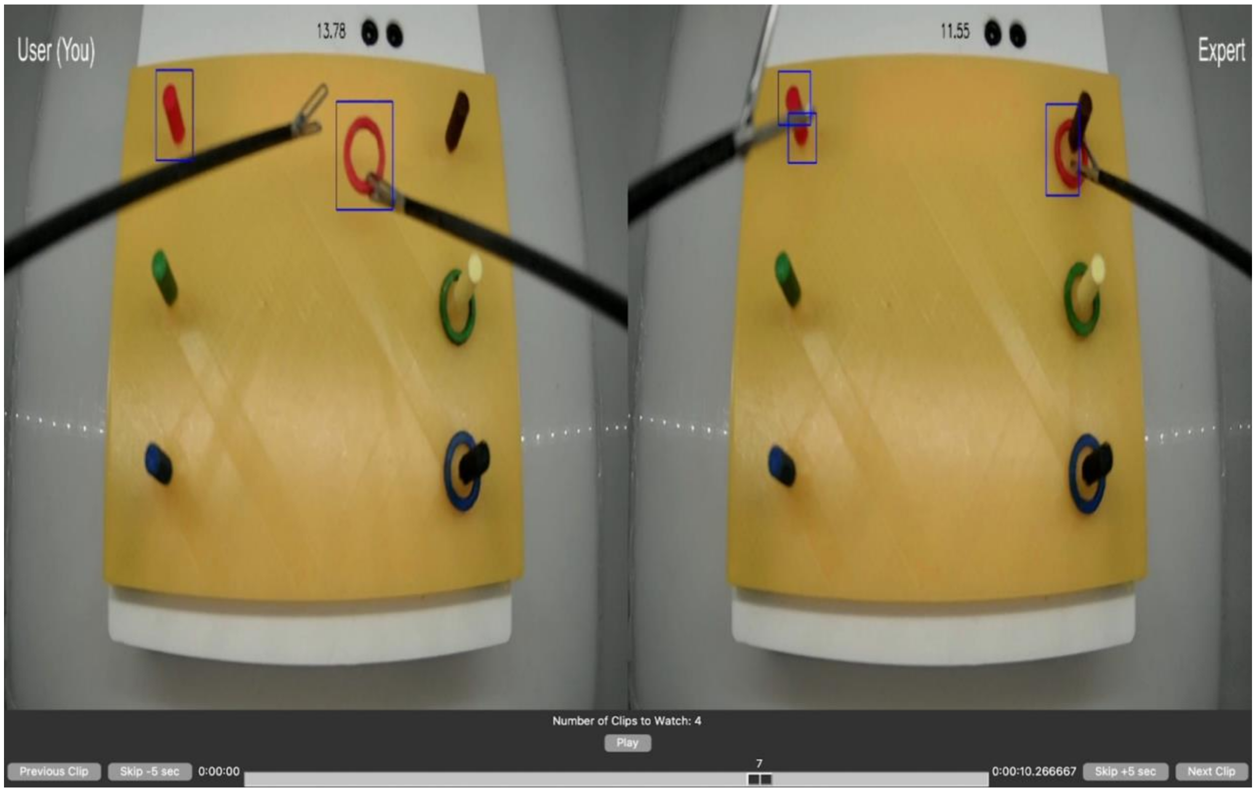


Figure 31: Video tool showing areas for improvement in Youssef's performance (the user).

In the illustrative Figure above of the tool's interface, we observe a split-screen comparison that plays a crucial role in the analysis and enhancement of surgical skills. On the left panel labelled 'User (You)' (Youssef in this case as mentioned), we see a segment of a trainee's performance, whereas the right panel, marked 'Expert' (Atallah), displays the corresponding segment from an expert surgeon's procedure supposedly. These are specific moments where the tool has identified significant deviations in technique or efficiency between the user and the expert. Timing metrics are visible at the top of each panel, allowing for precise comparisons down to the second. Below the video playback, the interface provides controls and a progress indicator, 'Number of Clips to Watch, as shown in Figure 31, which assists in navigating through the segmented procedural tasks. These visual aids are designed to draw the trainee's attention to critical aspects of the procedure, fostering a direct, comparative learning experience. By focusing on these moments, the simulator not only aids in recognizing the nuances of skilled performance and provides straightforward, actionable insights for the trainee's self-improvement. The overall workflow of the class involves comprehensive analysis through aligning time series data, normalizing and segmenting the data for detailed examination, and visually comparing video segments where user/trainee performance differences are most noticeable.

In conclusion, the significance of this section in developing a laparoscopy surgery simulator cannot be overstated. This innovative tool is pivotal in enhancing surgical training by offering detailed, informative feedback on trainees' performance. The simulator facilitates targeted improvement by identifying and displaying clips of areas within each segmented procedural task. This feature is precious as it allows trainees to refine their skills and address potential weaknesses independently, without the constant oversight of an expert surgeon. This autonomous learning approach not only accelerates skill development but also fosters a deeper understanding of surgical techniques, ultimately contributing to the advancement of laparoscopic surgery education.

6.4.3. TaskPerformanceAnalyzer: Python Class for Time Series Analysis

In an effort to enhance the educational experience for trainees in pediatric laparoscopic surgery, we have developed a cutting-edge chatbot powered by Large Language Models (LLMs), explicitly utilizing the GPT-3.5 Turbo model through OpenAI’s API. This chatbot is designed to provide users with a convenient and interactive platform to ask questions and gain insights about various aspects of surgical training. The development of this chatbot was also a part of my learning journey with LLMs, exploring their potential applications in medical education.

The chatbot is built on a foundation of Python programming and leverages the Streamlit library to create a user-friendly web interface. Upon initiating the chatbot, users are greeted with a welcoming message and prompted to ask any questions related to pediatric laparoscopic surgery as shown in Figure 32.

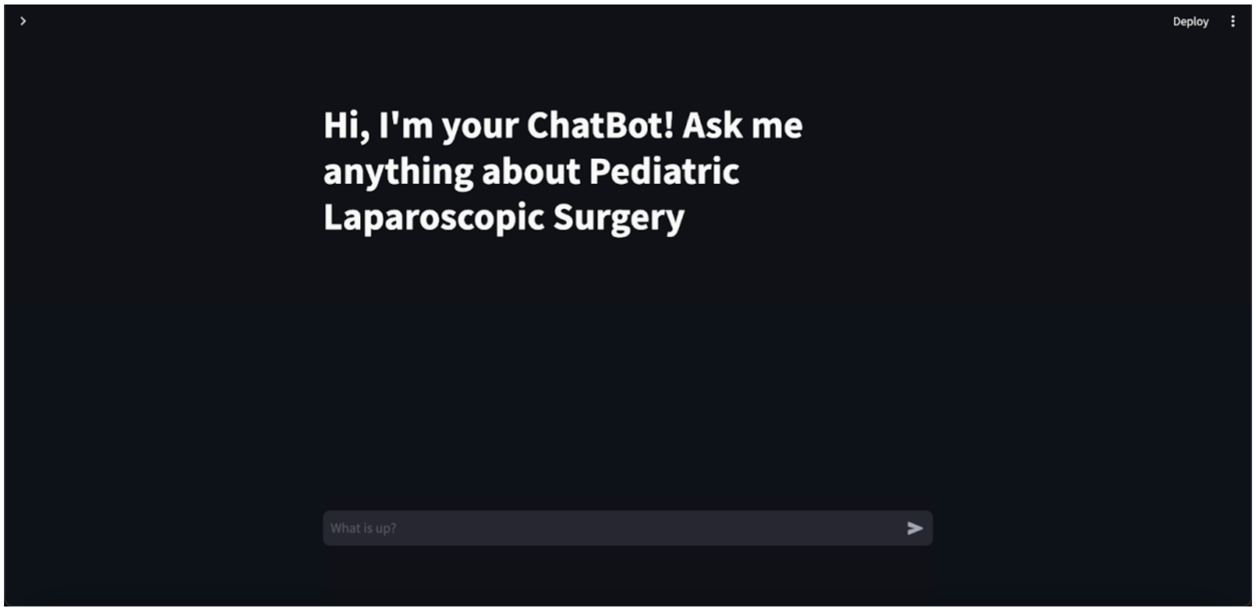
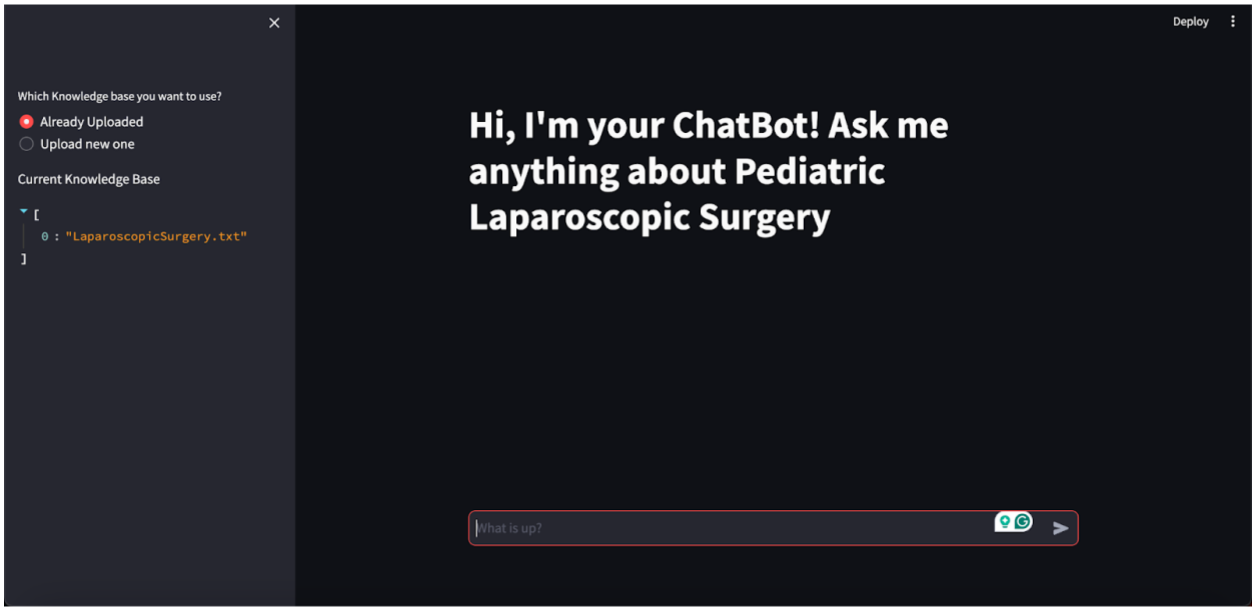

Hi, I'm your ChatBot! Ask me anything about Pediatric Laparoscopic Surgery

Figure 32: Initial interface of the Pediatric Laparoscopic Surgery training chatbot.

The chatbot's backend is configured to handle user queries efficiently, processing them through the GPT-3.5 Turbo model to generate accurate and informative responses. To ensure the chatbot is equipped with relevant and up-to-date knowledge, we have implemented a system that allows for the dynamic uploading of textual documents. These documents serve as the knowledge base for the chatbot, enabling it to provide answers based on the latest research, guidelines, and best practices in the field of laparoscopic surgery. Users can upload new documents to the knowledge base, ensuring the chatbot's information remains current and comprehensive. The chatbot's interface includes a sidebar that allows users to choose between using an existing knowledge base or uploading new documents as visible in Figure 33.



Which Knowledge base you want to use?
[✓ Already Uploaded] Upload new one

Current Knowledge Base

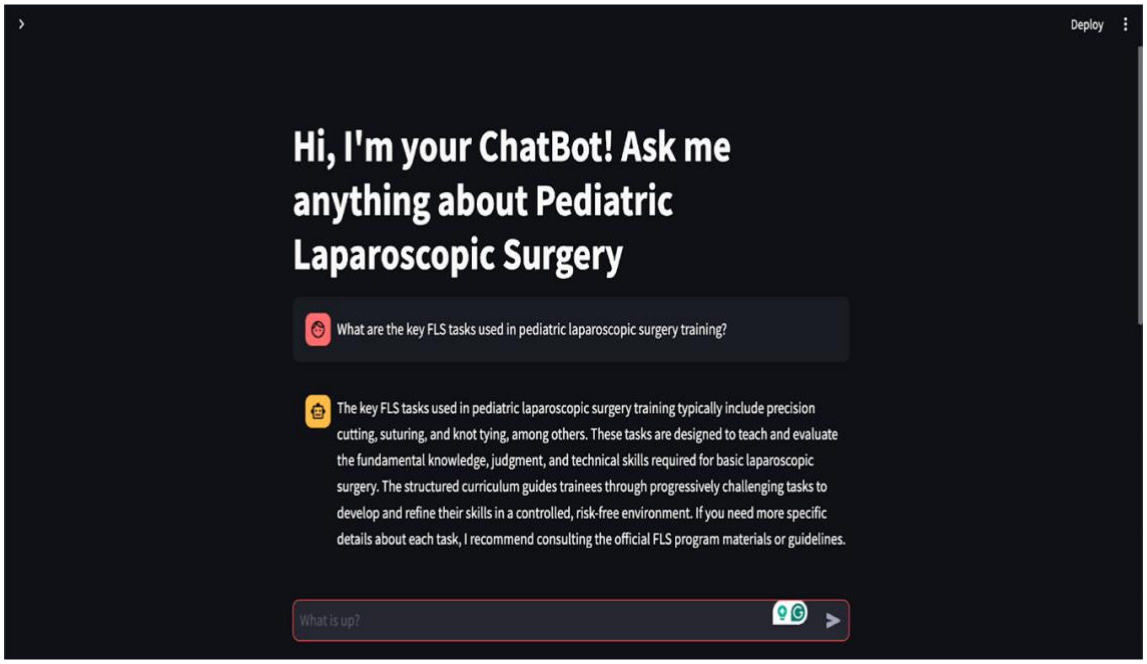
```
[
  0 : "LaparoscopicSurgery.txt"
]
```

Hi, I'm your ChatBot! Ask me anything about Pediatric Laparoscopic Surgery

Figure 33: Chatbot interface displaying knowledge base selection options.

This feature ensures the chatbot can cater to various inquiries, from basic concepts to advanced surgical techniques. The uploaded documents are processed and stored in a temporary directory, segmented, and indexed for efficient information retrieval. When a user submits a question, the chatbot employs a combination of text splitting, embedding, and vector storage techniques to analyze the query and retrieve relevant information from the knowledge base. The chatbot utilizes the Chroma vector store and the SentenceTransformerEmbeddings model to perform similarity searches and identify the most pertinent documents. The matching documents are then used to generate a comprehensive answer and present it to the user in the chat interface.

The chatbot's implementation includes several features to enhance the user experience, such as caching to improve response times, dynamic updates to the knowledge base, and the ability to handle multiple documents. Additionally, the chatbot visually represents the sources used to construct the answer, offering users transparency and the opportunity to explore the underlying information in more detail. It is important to note that if a user asks an irrelevant question or one that is outside the scope of the chatbot's knowledge base, the chatbot is designed not to hallucinate or provide arbitrary responses. Instead, it will inform the user that the question is irrelevant to the current knowledge base. The following figures will show some examples of questions asked to the chatbot.

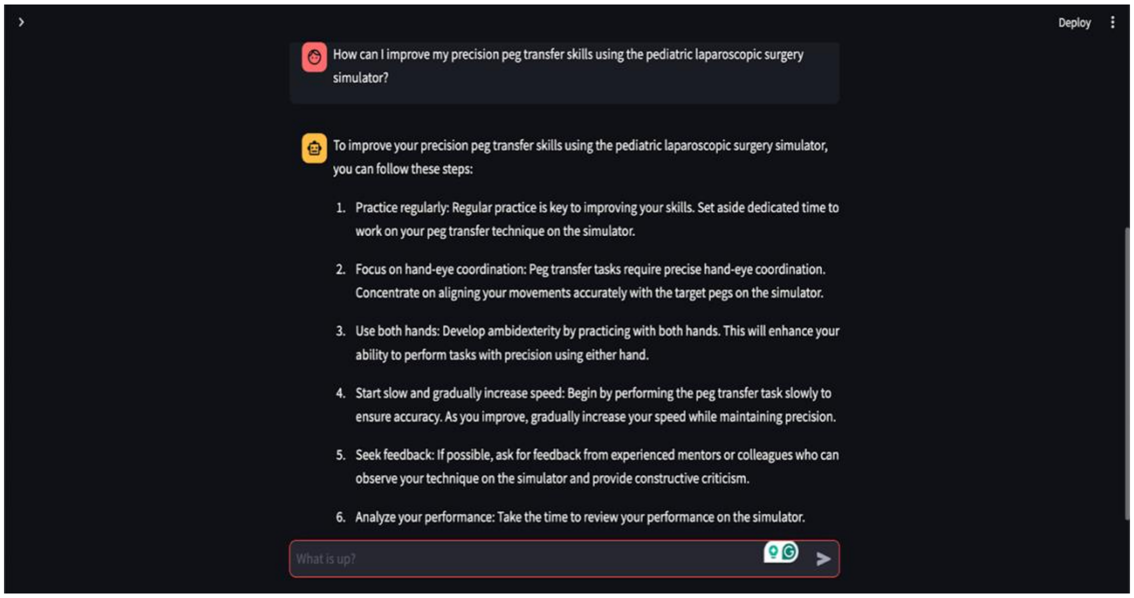


Hi, I'm your ChatBot! Ask me anything about Pediatric Laparoscopic Surgery

What are the key FLS tasks used in pediatric laparoscopic surgery training?

The key FLS tasks used in pediatric laparoscopic surgery training typically include precision cutting, suturing, and knot tying, among others. These tasks are designed to teach and evaluate the fundamental knowledge, judgment, and technical skills required for basic laparoscopic surgery. The structured curriculum guides trainees through progressively challenging tasks to develop and refine their skills in a controlled, risk-free environment. If you need more specific details about each task, I recommend consulting the official FLS program materials or guidelines.

Figure 34: Chatbot responding to query on Fundamental Laparoscopic Surgery task.

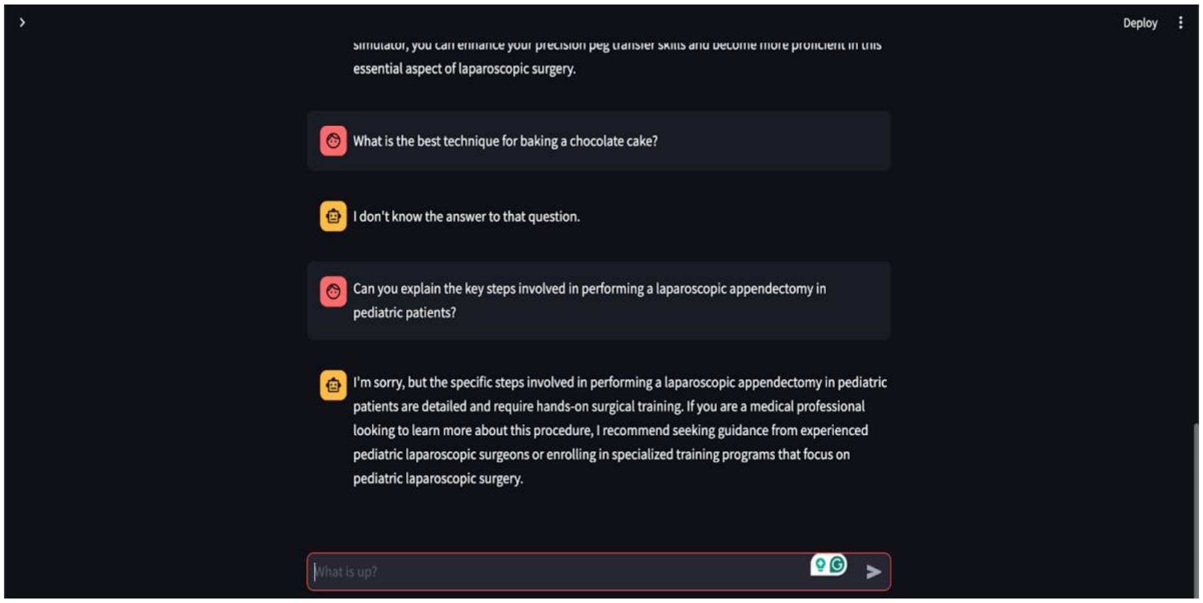


How can I improve my precision peg transfer skills using the pediatric laparoscopic surgery simulator?

To improve your precision peg transfer skills using the pediatric laparoscopic surgery simulator, you can follow these steps:

1. Practice regularly: Regular practice is key to improving your skills. Set aside dedicated time to work on your peg transfer technique on the simulator.
2. Focus on hand-eye coordination: Peg transfer tasks require precise hand-eye coordination. Concentrate on aligning your movements accurately with the target pegs on the simulator.
3. Use both hands: Develop ambidexterity by practicing with both hands. This will enhance your ability to perform tasks with precision using either hand.
4. Start slow and gradually increase speed: Begin by performing the peg transfer task slowly to ensure accuracy. As you improve, gradually increase your speed while maintaining precision.
5. Seek feedback: If possible, ask for feedback from experienced mentors or colleagues who can observe your technique on the simulator and provide constructive criticism.
6. Analyze your performance: Take the time to review your performance on the simulator.

Figure 35: Chatbot offering strategies to improve precision peg transfer skills.


simulator, you can enhance your precision peg transfer skills and become more proficient in this essential aspect of laparoscopic surgery.

What is the best technique for baking a chocolate cake?

I don't know the answer to that question.

Can you explain the key steps involved in performing a laparoscopic appendectomy in pediatric patients?

I'm sorry, but the specific steps involved in performing a laparoscopic appendectomy in pediatric patients are detailed and require hands-on surgical training. If you are a medical professional looking to learn more about this procedure, I recommend seeking guidance from experienced pediatric laparoscopic surgeons or enrolling in specialized training programs that focus on pediatric laparoscopic surgery.

Figure 36: Chatbot responsiveness to irrelevant and slightly relevant queries.

Figures 34 and 35 show that the chatbot provided informative responses based on the data uploaded to the database (as shown in Figure 33). However, in Figure 36, when posed with an irrelevant question, the chatbot gracefully acknowledged its limitations by stating that it didn't have an answer. Yet, when a slightly relevant question was asked in the second prompt, it offered a brief response without straying into conjecture and advised seeking guidance from expert surgeons. Consequently, this chatbot demonstrates its potential as a valuable tool for new trainees learning on the simulator, provided that the database is populated with the appropriate information.

In summary, the development of this LLM-based chatbot represents a significant advancement in surgical training. By providing trainees with an interactive and intelligent platform to ask questions and learn, the chatbot contributes to a deeper understanding of pediatric laparoscopic surgery. It supports the continuous improvement of surgical skills. While this chatbot has been a valuable part of my exploration of LLMs, future iterations of the project may decide to integrate it with the existing user interface or opt not to, depending on its perceived utility and relevance to the overall training experience.

6.5. User Interface Design

In the previous implementation of the pediatric laparoscopic simulator, the user interface was designed to be a simple application that relied heavily on the precise location of a user's mouse movements to progress through the program. In addition, the absence of buttons and relevant graphics made for a less sophisticated design that would not work with our requirements or our additions to the data/signal processing element of our program. Thus, this prompted the creation of a new user interface that would use some functionality of the old program but would be based on more sophisticated GUI capabilities from Python's Tkinter library. The subsequent sections below will present the new and improved user interface, along with the features that were added to enhance the functionality of the existing program.

For simplicity, and ease of use and integration, Python was chosen as the selected language to implement the new user interface. Since the old program was built in Python, and the data processing, signal processing, and machine learning steps are also executed in Python, it was decided that the language's tkinter library has extensive GUI capabilities. The object-oriented programming paradigm will separate large portions of code written using the functional programming paradigm into distinctly defined classes.

6.5.1. The Design Process of the UI

6.5.1.1. Developing a prototype

Before initiating the programming process, a prototype for the user interface was developed to create some sense of organization. However, the actual program looks somewhat different from what we prototyped, because of multiple modifications and design improvements. Figure 37 shows some examples from the prototype we developed using Microsoft PowerPoint. This exact moving video background was used in the actual program, however, fonts button sizes, and text background and layout have changed.

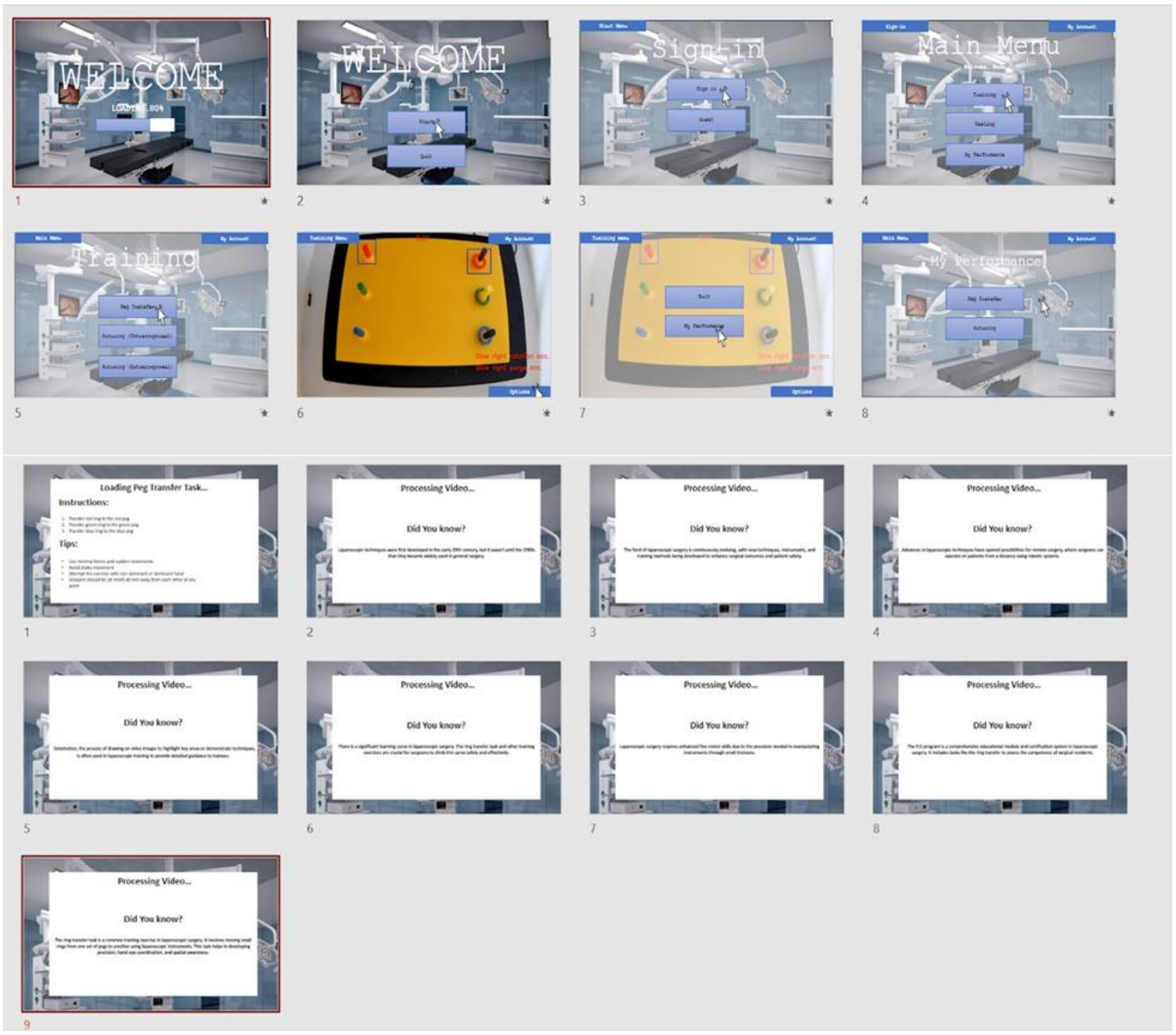


Figure 37: Sample prototype for our user interface showing various menu screens (top) and loading screens (bottom) for the first iteration of our design.

6.5.1.2. Code Hierarchy

The UML class diagram of the code (found in the project repository in Appendix A), which is organized into a main class and its dependencies. Class ‘Application’ runs the main window of the program which the user interacts with most of the time. It is the ‘face’ of the user interface. Class ‘LoginRegister’ takes care of user login, sign-in, and registration capabilities. The ‘TaskExecution’ class implements the sensor data capture and camera functionalities for the three laparoscopic tasks the application currently supports. This class takes care of launching a training task, copying sensor data into a text file for subsequent data processing, and opens or closes the camera view for a particular task depending on the program's state. The rest of the classes deal with data processing, task segmentation, and/or performance evaluation. The 'SensorDataCleaner' class is responsible for sorting through the sensor data text file and converting it to a CSV file. The 'DataSegmentation' class segments the data acquired into three subtasks for the analysis process to follow. The 'TaskPerformanceAnalyzer' class executes the algorithms responsible for performance comparison and evaluation, as detailed in Section 6.4. The 'DataPlotter' class aligns the user's data signals with the expert's data signals using DTW to present them on the same plot for visual performance evaluation. Finally, the 'VideoPlayer' class aligns videos of the user and expert performing the task so they can be viewed and compared side by side when prompted. Some classes have to open a sub window derived from the main window to implement their intended functionality. Subsequent iterations of the user interface will add other classes as more features are integrated with the simulator's software. This structure for the user interface has provided a good foundation for future development.

6.5.1.3. Design flow of the new user interface

Figure 38 shows the design flow for some of the multiple screens and windows that were developed for the program. These include the welcome screen, login/registration screen, the main menu, the task menu, the loading screen, the camera view screen, the feedback menu, the evaluation menu, the video playback menu, the history menu, the video feedback menu, the visual feedback menu, tool trajectory menu, and the subtasks menu.

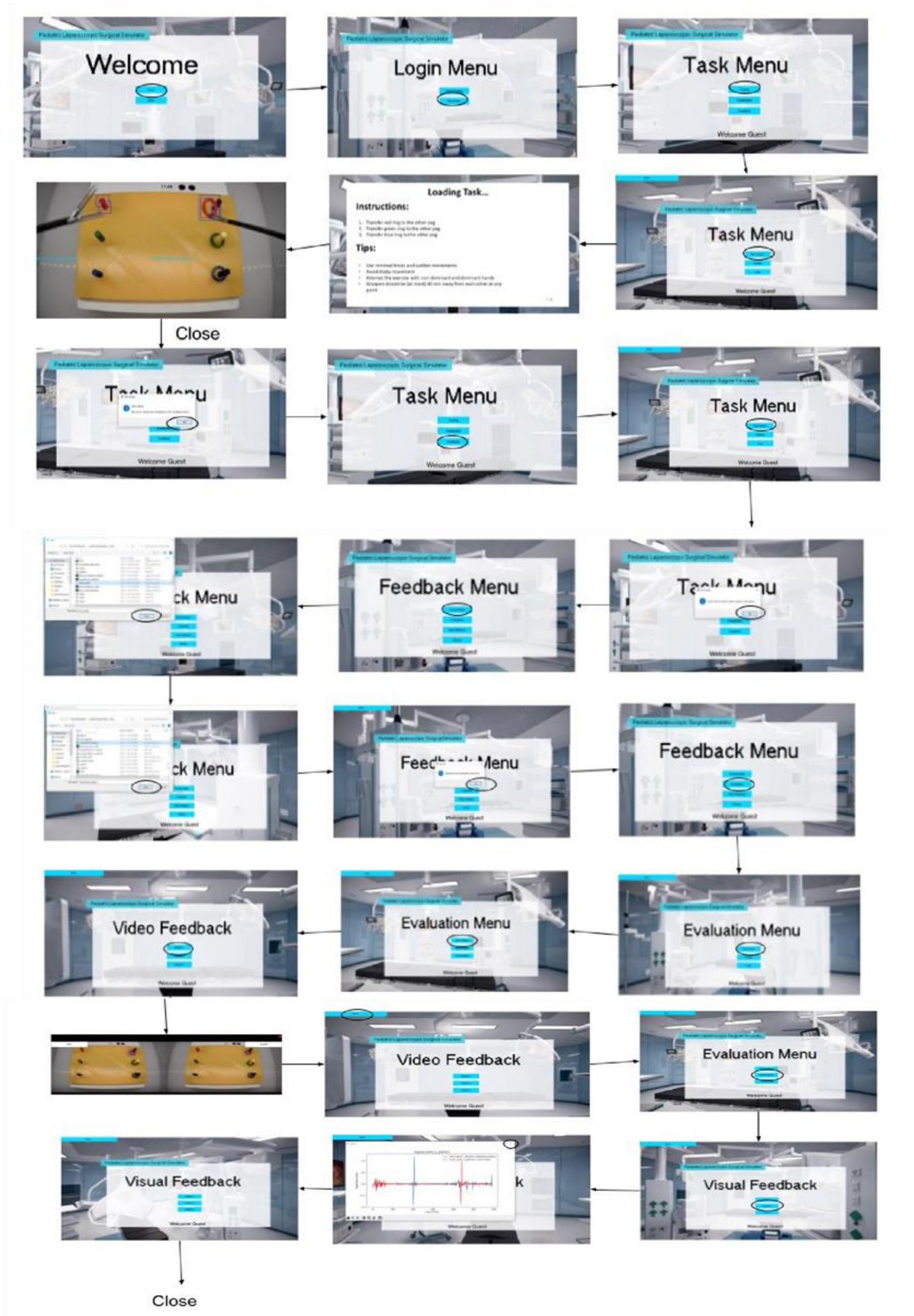


Figure 38: Design flow of new user interface shown with windows/screens that appear when certain buttons are clicked.

The welcome menu leads the user to the login menu where they can choose to login as a user, login as a guest, or register themselves. Once they are signed as a guest or authenticated user, they will be led to the main menu. The main menu allows the user to choose between the training, examination, and feedback menus. Training mode will allow the user to practice with direct feedback from the program while they perform the task. Examination mode, not shown in Figure 37 above, is identical to the 'Training' path, but tests the user's abilities and provides no indication of crossed thresholds or direct feedback to help the user but its functionality has not been implemented yet. The training and examination menus will lead the user to a task menu where they can choose which task they want to perform. The task menu allows the user to choose between the peg transfer, suturing, or ligating loop tasks. Once they click on any one of these tasks, the program will launch the loading screen which gives the user an overview of the task to be performed. After 16 seconds or so for sensor calibration, the camera view will pop up and the user can immediately begin the task. For now, only the peg transfer task has been fully implemented, while the template for adding other tasks is already there. Once they finish, they can close the camera view, and will be prompted to check their feedback. When they return to the main screen on the main window, they can lead themselves to the feedback menu, where they select which task, they wish to view the feedback for. From there, they will be sent to another screen of the feedback menu where they can process their data, access evaluations of their performance, watch a video playback of the task they just performed, or access historical data saved within their directory. Clicking on the video playback button will open a sub-window where the user can play the video of the task they just performed. Before the user clicks on the 'Evaluation button' on the feedback menu, they must process their data, and this instruction is given to the user before they enter the feedback menu. Once they process their data, and go to the evaluation menu, they have a few options. By clicking on the ‘Visual Feedback’ button, they can select a given subtask of the procedure, and then view a visual evaluation of their performance in different subtasks compared to a referenced expert. This consists of a visual overlay of the signals pertaining to the user and the expert’s for different motion-related parameters. This is shown as a sub-window that allows the user to flip through the different graphs of motion-related parameters. The ‘Video Feedback’ button allows the user to view videos of their subtasks and an expert’s at the same time side by side. The ‘tool trajectory’ button will open a window that will show the user’s tool trajectory throughout different subtasks of a procedure. As seen in Figure 38 above, while buttons are used to seamlessly transition between the multiple screens, the design flow can get complicated very quickly, especially when new features are being added.

6.5.2. Development and integration of new features in the user interface

6.5.2.1. New graphics and Tkinter elements

Based on the pre-developed prototype of the graphical design discussed previously, features of Python’s tkinter library like text boxes, buttons, and threading capabilities for moving video backgrounds were used to create the program. The moving video background of surgical room equipment along with the blue tkinter buttons was used to develop a theme that mimics the environment of an actual operating theatre. This theme was promoted throughout the whole program for consistency and professionalism. This allowed for code duplication and reusability, which made the development of new screens and the integration of new code much easier and more efficient to manage.

6.5.2.2. User login/registration feature

Figure 39 above presents the new log-in/sign-in, and registration feature that will be used for users to create an account or log into one, in order to save their historical and current data in one folder within the program directory. This presents a way for older data to be saved, so a user's past and current performance over time can be tracked and evaluated more extensively if they so choose. Every time someone registers, a new folder for them is created within the program directory if they do not already have an account. If the user signs in successfully, their data will be saved to the folder where their older data was saved. If a user wants to use the program as a 'guest user', their data will be saved to the program's guest folder. The login functionality will be further developed in future iterations to ensure proper storage, privacy, and protection of user data.

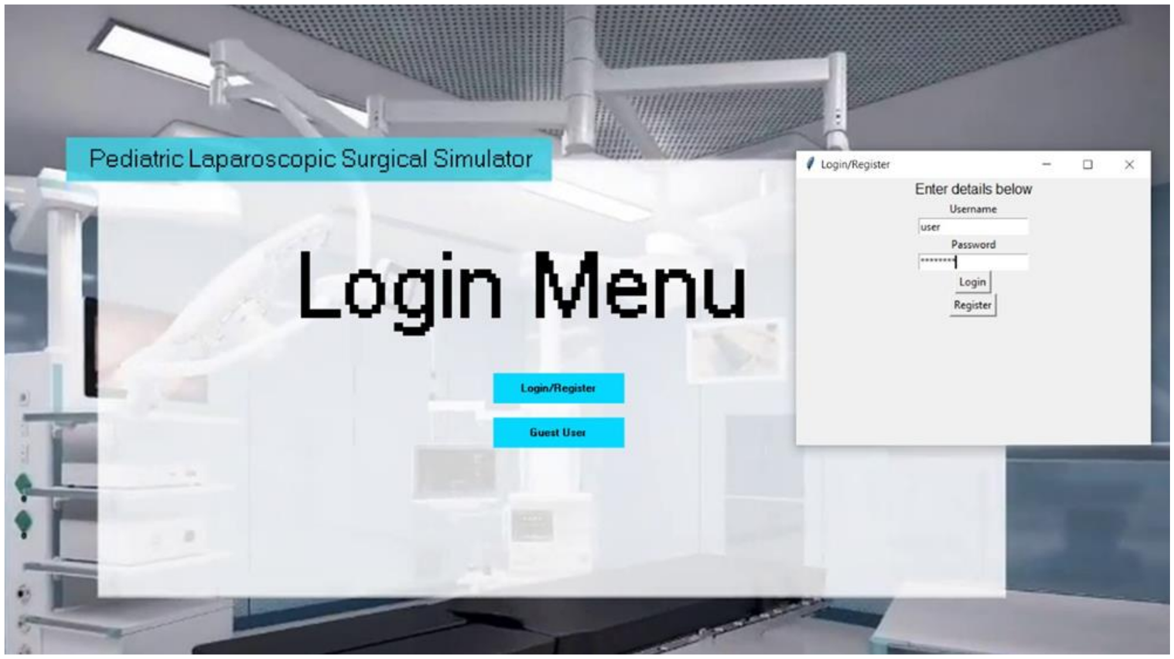


Figure 39: Login/registration screen with associated pop-window.

6.5.2.3. Integration of video playback feature log-in/registration feature

In the latest enhancement of our user interface, we have incorporated a pivotal feature: a media player script named "Video Playback." This innovative tool is designed to allow users to load and review their procedure videos post-training, providing an invaluable performance analysis resource.

Developed using Python, the script uses the Tkinter library, renowned for its robust graphical user interface (GUI) capabilities, as shown in Figure 40. The interface is designed for user-friendliness and efficient interaction. It includes a 'Load' button, enabling users to upload their video files. This is complemented by the 'Play/Pause' button and additional controls that allow users to skip through the video, enhancing their experience.

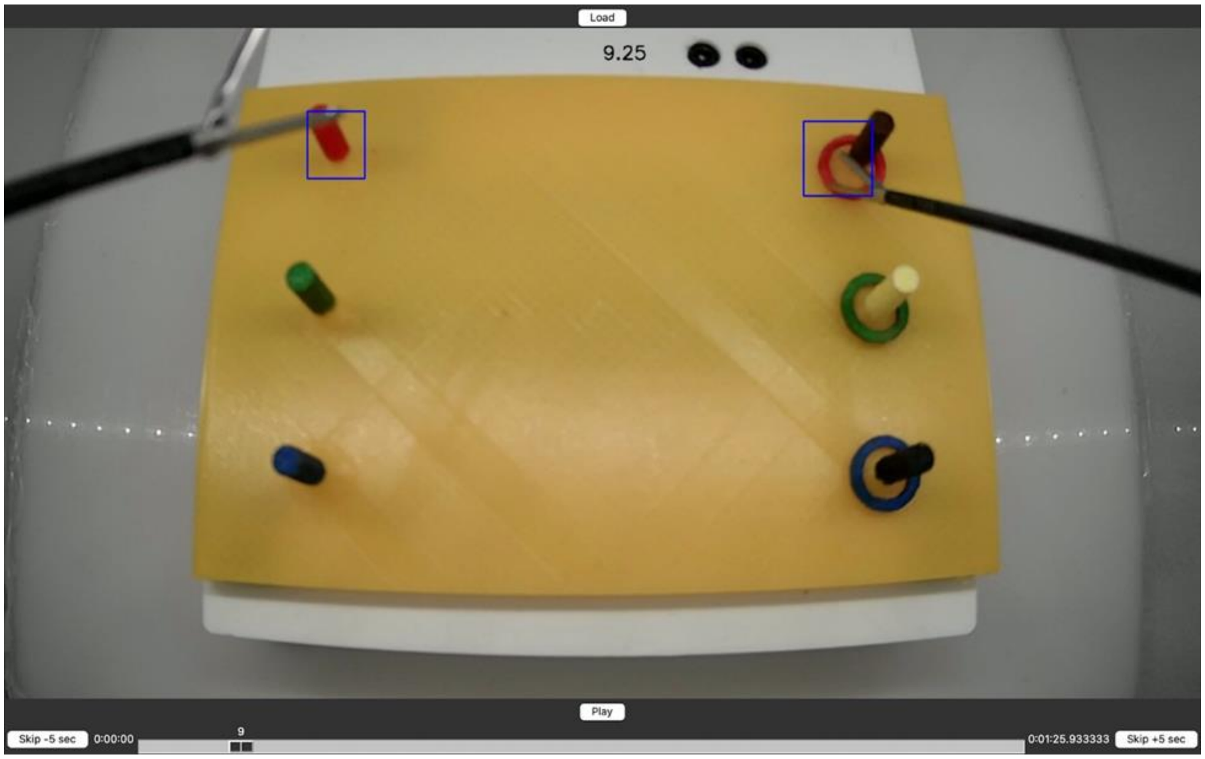


Figure 40: A media player application for procedure videos post-training.

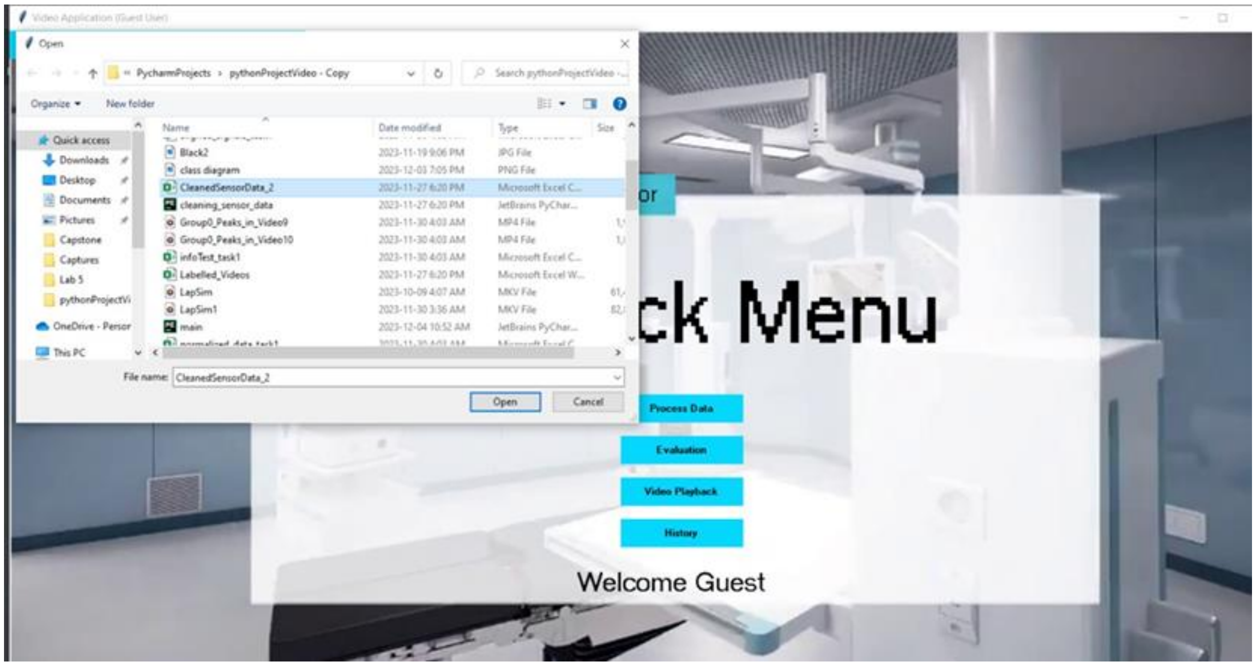
The script's heart lies in its video playback functionality, handled by the TkinterVideo component. This component ensures seamless video display and supports scaled playback within the interface, adapting to different user needs. The progress slider is a standout feature, allowing users to track their progress through the video and seek specific segments easily. This functionality is further enriched by dynamically updating start and end time labels, clearly representing the video's timeline.

Event handling is managed within the script, with custom functions addressing user interactions such as video loading, seeking, and skipping. These functions are pivotal in rendering the playback experience both responsive and intuitive. Additionally, the script is adept at detecting when the video has ended, resetting the playback controls and slider to their initial states, thereby ensuring a smooth user experience. Executed within the main block, the script initializes the VideoPlayerApp class and activates the Tkinter main loop. This initiation is crucial as it keeps the graphical interface responsive and engaging for the user.

Overall, the "Video Playback" feature is a significant stride in enhancing our interface's usability. It simplifies the review process for users and is crucial in facilitating their learning and improvement, marking an important milestone in our commitment to delivering user-centric solutions.

6.5.2.4. Integration of new data processing procedure

In the previous implementation of the user interface, the sensor data was processed behind the scenes whenever the user wanted to view their performance for different motion-related parameters throughout a procedure. The processing only consisted of reading the sensor data from the text file, and then simply graphing it for the user.



Name	Date modified	Type	Size
Black2	2023-11-19 9:06 PM	JPG File	
class diagram	2023-12-03 7:05 PM	PNG File	
CleanedSensorData_2	2023-11-27 6:20 PM	Microsoft Excel C...	
cleaning_sensor_data	2023-11-27 6:20 PM	JetBrains PyChar...	
Group0_Peaks_in_Video9	2023-11-30 4:03 AM	MP4 File	1,
Group0_Peaks_in_Video10	2023-11-30 4:03 AM	MP4 File	1,
infoTest_task1	2023-11-30 4:03 AM	Microsoft Excel C...	
Labelled_Videos	2023-11-27 6:20 PM	Microsoft Excel W...	
LapSim	2023-10-09 4:07 AM	MKV File	61,
LapSim1	2023-11-30 3:36 AM	MKV File	82,
main	2023-12-04 10:52 AM	JetBrains PyChar...	
normalized_data_task1	2023-11-30 4:03 AM	Microsoft Excel C	

File name: CleanedSensorData_2

Figure 41: Opening of file explorer menu upon clicking 'Process Data'.

However, the new script allows the user to select their collected sensor data text file from the directory once they click the ‘Process Data’ button, as shown in Figure 41. This data is cleaned and transformed into a CSV file. It is then labelled and segmented into gestures as described in Section 6.2-6.3, and the user is prompted with a pop-up window that informs it to pick the correct file with the labelled data in order to prepare for the algorithmic steps that will follow later. This new data processing procedure was integrated into the new user interface through the inclusion of the relevant classes. The relevant buttons were linked to the corresponding functions in the Application class, which instantiated the classes required for data processing.

6.5.2.5. Integration of new performance evaluation features

The previous implementation did not include any way for a user to compare their performance to an expert in a direct and specific manner. Continuing from the previous section, after the user selects the labelled procedure file, they must wait for some time while the program segments the procedure into three subtasks and generates parameters for algorithms. These algorithms will allow for the comparison of an expert’s signals to the user’s signals for various motion-related variables, both graphically, and visually through videos. A loading screen will pop up to entertain the user while they wait. Once the program finishes, the user can click on the ‘Video Feedback' or ‘Visual Feedback’ buttons to view their graphical or video feedback for the three subtasks of the procedure. A sample of the feedback the user could receive is shown in Figure 42.

This new procedure for performance evaluation was integrated into the new user interface through the inclusion of the relevant classes. The relevant buttons were linked to the corresponding functions in the Application class, which instantiated the classes required for running the algorithms related to performance evaluation.

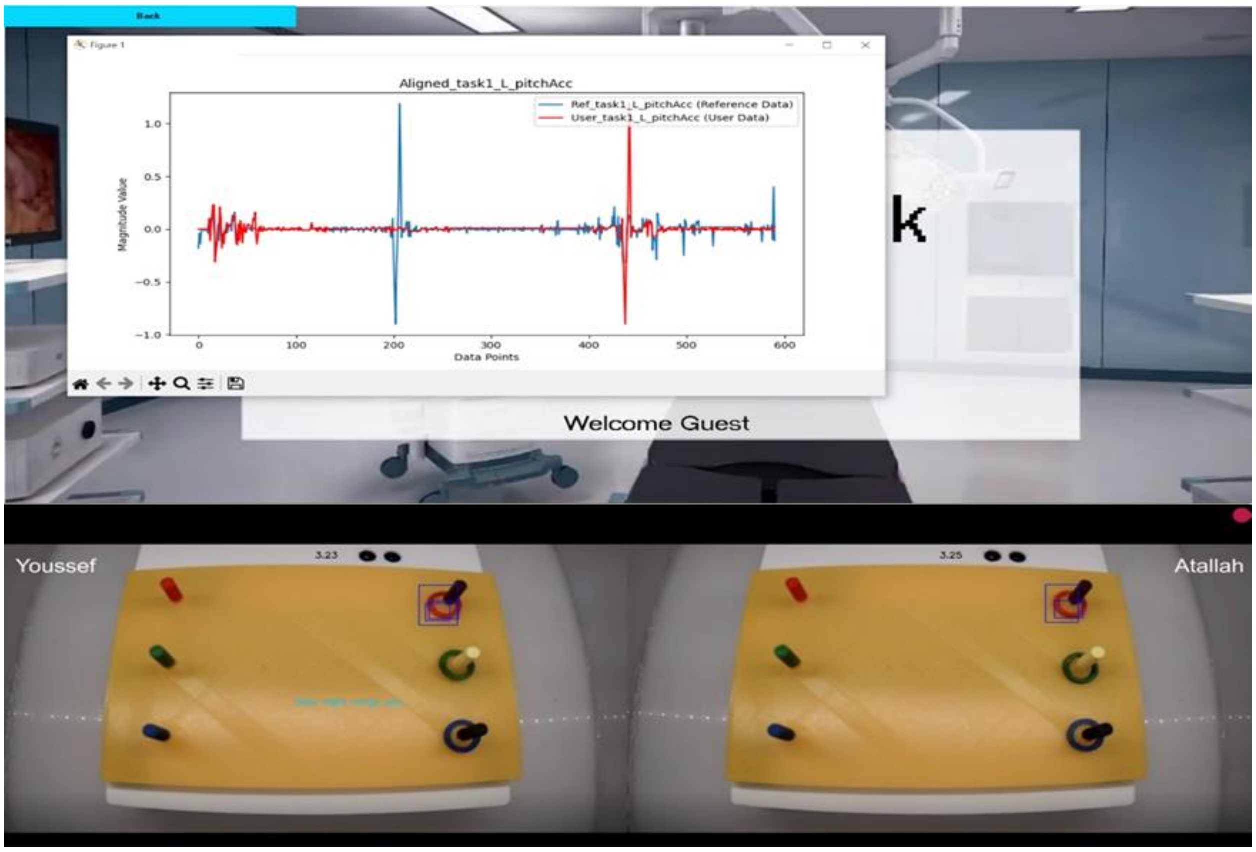


Figure 42: Integration of enhanced visual (top) and video (bottom) feedback.

6.5.3. Additions to the User Interface

6.5.3.1. Integration of Tool Trajectory Feedback

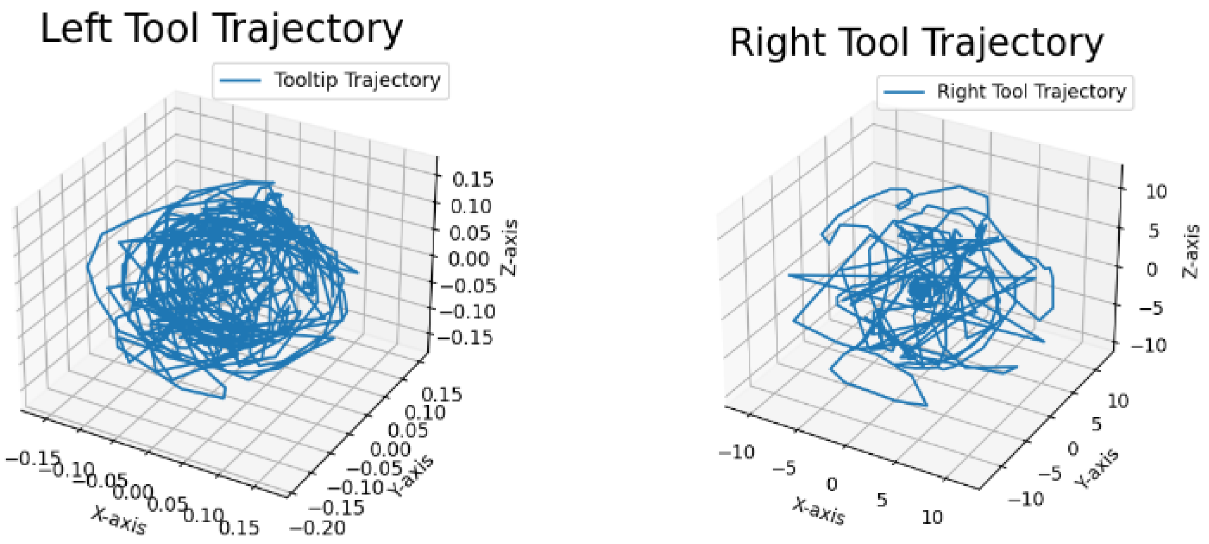
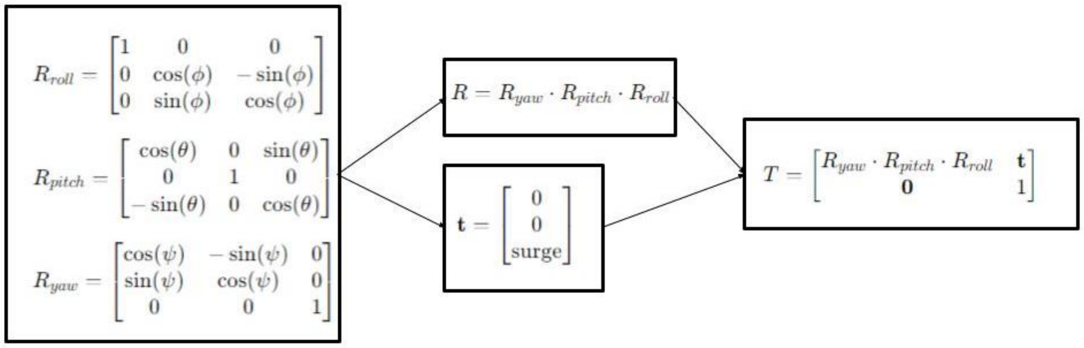


Figure 43: Sample tool trajectory feedback of the left and right instruments.

Figure 43 above shows a sample output of the tool trajectory feedback provided to trainees from our simulator. Tool trajectory feedback in pediatric laparoscopy is crucial for enhancing surgical precision, safety, and outcomes in the constricted anatomical spaces of pediatric patients. From tool trajectory mappings, we can derive information about the bilateral coordination of both hands during surgery, as well as assess the smoothness of a trainee's movement. As can be seen in Figure 18, it appears that the left tool looks like it was moved more than the right tool. This indicates this particular trainee may have a dominant left hand that they may be unconsciously/consciously moving more often than the other. This indicates the trainee must improve their bilateral coordination. The mathematical formulas and matrices used to derive the tool trajectory for both tools are given in Figure 44 below.



$$R_{roll} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix}$$

$$R_{pitch} = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}$$

$$R_{yaw} = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R = R_{yaw} \cdot R_{pitch} \cdot R_{roll}$$

$$\mathbf{t} = \begin{bmatrix} 0 \\ 0 \\ \text{surge} \end{bmatrix}$$

$$T = \begin{bmatrix} R_{yaw} \cdot R_{pitch} \cdot R_{roll} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}$$

Figure 44: Mathematical formulas and matrices used to compute the tool trajectory.

In Figure 44 above, each of the rotations around the principal axes (roll around the x-axis, pitch around the y-axis and yaw around the z-axis) are defined by rotation matrices [37]. By combining these three rotation matrices, we create a single rotation matrix that captures the tool's orientation in 3-dimensional space [37]. The tool's translation is captured by the tool's translation matrix which is represented by the surge movement along the z-axis [37]. Finally, the 4x4 composite transformation matrix includes the 3x3 rotation matrix in the top left corner and the translation vector in the rightmost column [37]. This approach is often used to track surgical tools in 3-dimensional space.

6.5.3.2. Examples Menu

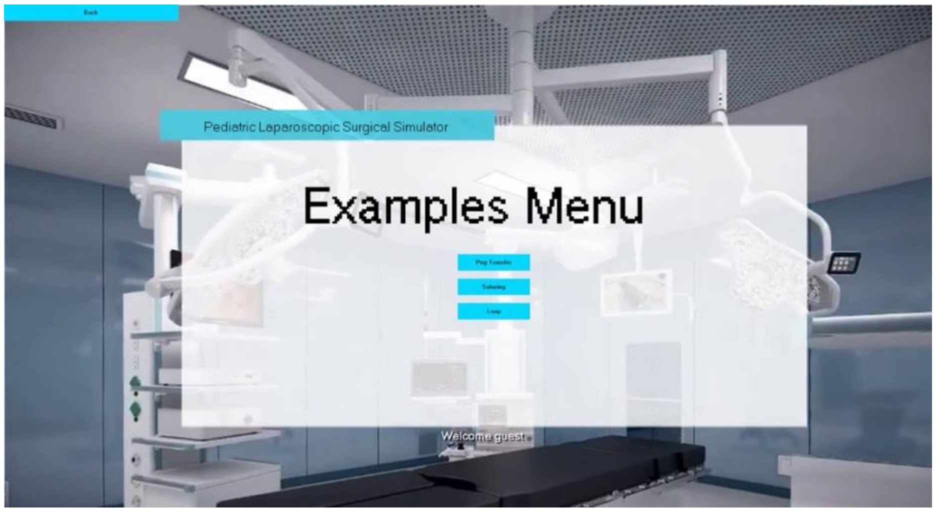


Figure 45: Newly added Examples Menu which features expert FLS videos of laparoscopic tasks.

Figure 45 above shows the examples menu which features expert FLS videos of three laparoscopic tasks, namely the peg transfer, suturing, and ligating loop tasks. Trainees can click on any task and view a video of how to perform a given task before they go ahead to attempt one by themselves.

6.5.3.3. Voice-Assisted Navigation

In the previous phase of the project, it was noticed that to new users, the program flow is not intuitive and in fact quite confusing to navigate through. Trainees would benefit from verbal instructions and information on how to navigate the user interface in order to make it even more interactive and easy to use. As a result, we used AI text-to-speech tools available online to implement voice-assisted navigation for each screen of the user interface, to guide the user before, during, or after their performance. This has significantly enhanced the user's experience when navigating through the feedback sections of the program.

7. Testing

Each hardware-related sensor undergoes rigorous testing and analysis to validate its performance under various conditions. Since the MPU6050 and PMW3389 readings were previously verified and the sensors remained unchanged from the previous iteration, there was no need to reproduce the test results. Therefore, the results mentioned below are from the previous iteration and were not reproduced.

The MPU6050 was tested with a clamp and protractor, yielding a working space of approximately ±30 degrees. Placed flat inside a housing unit, perpendicular to the shaft, it sensed pitch and yaw angles to two decimal places. The total workspaces around 60 degrees (+30 to -30). In Figure 46, pitch angles stayed within the expected range, while in Figure 47 yaw angles reached -50 degrees, exceeding the expected -35 degrees. This discrepancy of -15 degrees is due to the trainer box trocars/sensor housing being tilted at a 15-degree slope, affecting calibration. Despite this, both pitch and yaw angles functioned as intended [23].

The PMW3389 performed accurately on a flat surface, measuring absolute displacements (x and y in mm) to two decimal places without detecting changes when moved. Roll and Surge tests were conducted over six seconds within the trocar and sensor housing. The roll test (degrees vs. time) in Figure 48 showed a positive roll with clockwise rotation from time zero and a negative roll with counter clockwise rotation. The PMW3389 can express both positive and negative angles, potentially exceeding 360 degrees, depending on wrist or finger movements. In the surge test (mm vs. time) in Figure 49, the absolute point at time zero determined a positive or negative surge, with insertion yielding a positive surge and extraction resulting in a negative surge [23].

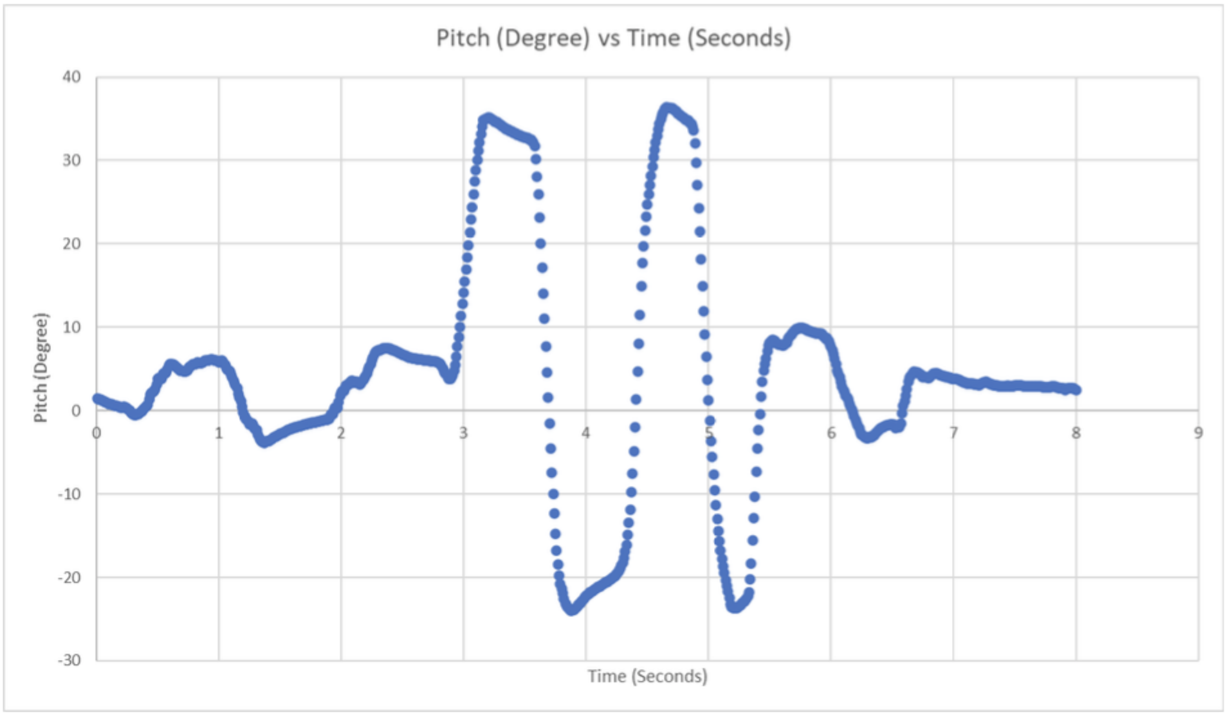


Figure 46: Pitch (Degrees) vs Time (Seconds) test with MPU inside sensor housing on box [23].

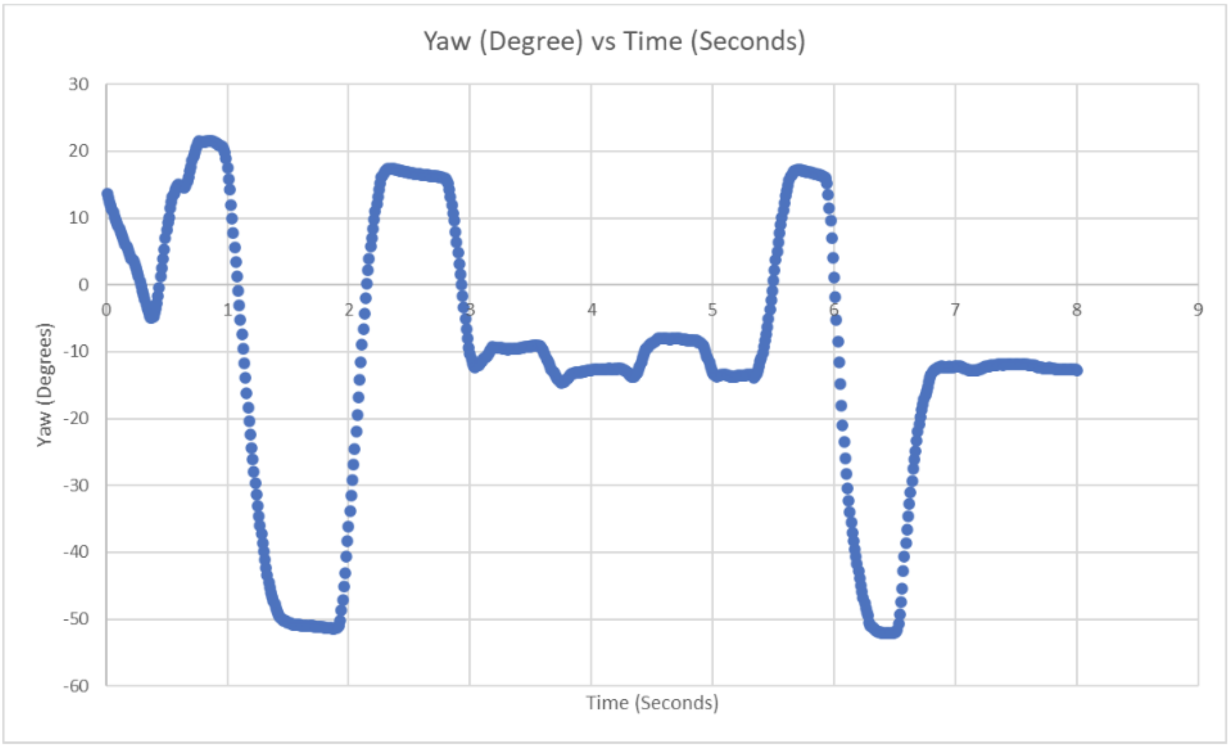


Figure 47: Yaw (Degrees) vs Time (Seconds) test with MPU inside sensor housing on box [23].

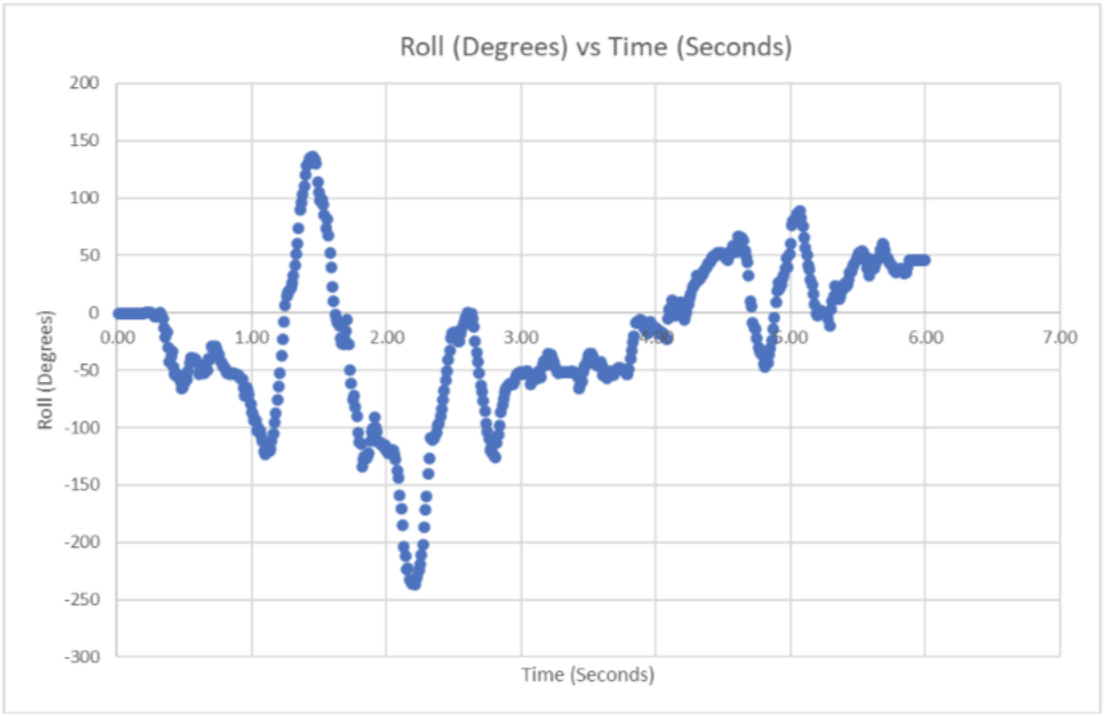


Figure 48: Roll (degree) vs Time (seconds) test done with the tool shaft inside the sensor housing[23].

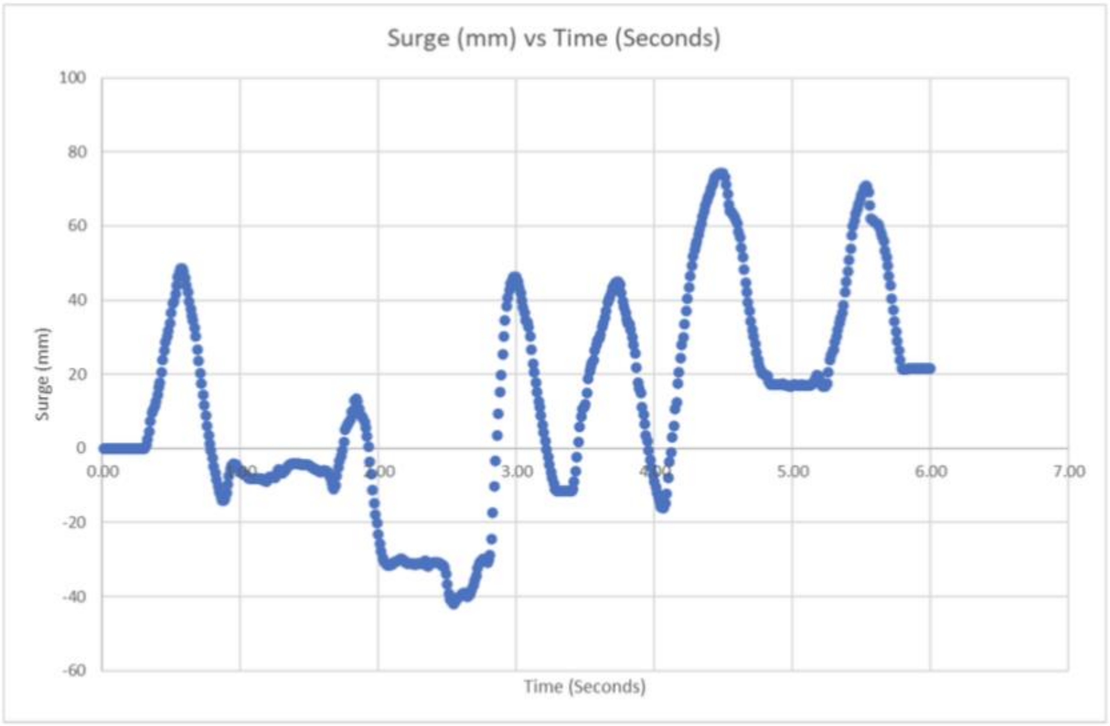


Figure 49: Surge (mm) vs Time (Seconds) with the tool shaft inside the sensor housing [23].

The strain gauge load cells were added in this iteration, prompting the need for a verification process. Testing and calibrating the load cells with an HX711 involves placing a reference weight (500g) on the load cell to record output readings. These readings are used to calculate a calibration factor, which is then applied to convert future readings into accurate force measurements in the main program. This calibration was conducted using a calibration code on the Arduino to ensure that the load cell's readings reliably reflect the applied forces, and it was found the accuracy of the sensors varied by ±2%.

The augmented reality functionality being tested includes the ring task and objective highlighting. The ring task involves three colours: red, green, and blue. After manual calibration using HSV.py, the program should highlight the matching red ring and peg initially. When a ring is placed on its matching peg, the highlights merge into one, and the program advances to the next state after two seconds. This process repeats for green and blue rings. Due to the new LED light being added a new recalibration was conducted using the steps above. The program must not advance unless only one object is highlighted, indicating the correct placement. Finally, after the blue ring and peg are combined, the program should return to the main menu automatically.

Throughout the development of our pediatric laparoscopic surgery simulator, rigorous testing was conducted to ensure the reliability and accuracy of each software and integrated feature. To validate the data processing pipeline, we tested the initial processing and segmentation stages. This involved ensuring that the Arduino microcontroller accurately captured and transmitted sensor data to the Python script. We simulated various surgical movements to generate diverse datasets, then processed them to verify that the script correctly cleaned and organized the data into DataFrames. The segmentation process was tested by dividing the data into subtasks and confirming that each segment contained the appropriate data corresponding to the predefined timestamps.

Testing the algorithms involved feeding segmented data into the DTW algorithm. We compared the output of these algorithms against expected results to ensure their accuracy. For gesture recognition, we verified that the system correctly identified specific hand movements and checked their accuracy. For the DTW algorithm, we checked that the time series alignment and performance comparison accurately reflected the similarities and differences between the user's and expert's performances. The user interface was tested for usability and functionality. This included verifying that all buttons and menu options were responsive, navigation between different program sections was smooth, and that tutorial videos and playback features worked correctly. The feedback system was tested by ensuring users received appropriate visual and audio feedback based on their performance. We also checked that the augmented reality components functioned correctly, providing real-time guidance during tasks. The performance evaluation component was tested by comparing the user's performance data against expert data. We verified that the TaskPerformanceAnalyzer class accurately calculated DTW distances and categorized performance. Additionally, we tested the video processing stage by ensuring that the system correctly identified segments where the user's performance significantly diverged from the expert's and that these segments were properly extracted and annotated in the video comparison feature.

Finally, integration testing was conducted to ensure that all simulator components worked harmoniously. End-to-end testing involved simulating a complete training session, from data acquisition to performance evaluation and feedback. This comprehensive testing ensured the simulator provided users with a seamless and practical training experience.

8. Discussion and Analysis

In the Hardware implementation, the force sensor encountered several issues at the beginning of the project. We aimed to incorporate more of the load cells that were utilized in the previous iteration to enhance the accuracy of the force plate. However, after consulting with the surgeons, they emphasized the necessity of bidirectional force sensors rather than single-direction ones, which had been limited by the funding allocated for load cells prior to the meeting. Despite these financial constraints, we managed to procure strain gauge load cells with an HX711 amplifier. These load cells effectively met the requirement and accurately captured bidirectional forces, as verified using a reference weight. Additionally, due to their lower cost, the amplifiers had a slower sampling rate, which resulted in a slowdown of the entire pipeline. We attempted to mitigate this issue through multithreading, but in the C language, such an approach proved infeasible. Despite the accuracy of the readings and the successful implementation of the load cells, it is imperative to replace the amplifier with a faster one to enable quicker sampling. Other parts of the hardware were implemented fully without any issues, and the thresholds for all the movements are illustrated in Table 6. In addition, the camera can be selected in the python program by settling it to (id = 0) and connecting to the microcontroller on communication port 3 by setting (serial = ‘COM3’), which can be change accordioning to the available ports.

Table 6: Measurement Thresholds [23].

Measurand	Thresholds
Force	$\mathbf{1\ N}$
Surge acceleration	$\mathbf{1\ m/s^2}$
Roll acceleration	$\mathbf{1\ m/s^2}$
Yaw acceleration	$\mathbf{1 \times 10^{-3}\ \theta/s^2}$
Pitch acceleration	$\mathbf{1 \times 10^{-3}\ \theta/s^2}$

The software implementation of our pediatric laparoscopic training simulator seamlessly integrates all aspects of the project. It begins by offering users the option to register and save their progress with an account or to sign in as a guest for quicker access. The software imports and stores data from various sensors, ensuring it is clean and organized in CSV files for algorithmic processing. It assesses the user's performance, comparing it to an expert surgeon, and presents the results in various feedback forms. A key feature of the software is the integration of gesture recognition, which provides real-time feedback focused on the user's hand movements and adherence to surgical techniques. The user interface is enhanced with tutorial videos that clearly explain the objectives and methodologies of different FLS tasks such as the peg transfer, suturing, and ligating loop task. Additionally, a playback feature allows users to review their selected performance or trial video, complete with the feedback received during the task. The software also includes an AI voice that guides users through the interface and explains the different feedback forms. Augmented reality components are integrated into the new user interface, enriching the learning experience. This comprehensive approach facilitates a deeper understanding of laparoscopic procedures and encourages self-improvement through reflective learning without needing expert supervision.

The objective of the feedback system is to reduce the reliance on constant, expert supervision in training by providing constructive feedback in various forms. Utilizing the simulator's sensors and augmented reality features, we quantify the sensor data to inform the user of their performance. Users receive warning messages on the screen whenever they exceed predetermined movement thresholds, ensuring they remain vigilant. Visual feedback is provided by aligning the user's movements with an expert's, allowing users to observe speed, rhythm, and steadiness differences. This alignment is achieved using Dynamic Time Warping (DTW), which has proven effective in our application. DTW distances are used to determine the deviation in performance for side-by-side video comparisons using an expert procedure. The threshold for feedback is not arbitrary but a calculated value. It's set at the mean of movements across different moving windows plus one standard deviation. This approach ensures that we accurately capture deviations between user and expert performance. Ideally, we would have wanted professionals to complete tasks on our simulator, collecting their data as the actual expert benchmark. However, we used a team member's performance as a temporary expert reference. The good news is that threshold adjustments can easily be made in future iterations, making our system adaptable and precise.

9. Future improvements

The user interface will undergo multiple iterations until all new software and hardware features have been implemented. First, with the integration of new tasks like the suturing, new code will be needed to assess the user's execution of the task through a camera view, and additional screens and functions will also need to be added. Other signal and data processing elements still need to be integrated into the user interface for new tasks, requiring additional organization or re-factoring of the code. File paths and user-related variables will need to be defined globally since now our simulator can save multiple runs and be tailored for each registered user.

In addition, improved feedback can be given while the user performs a task, like using a beep alarm sound to indicate when they have crossed specific parameters, like exceeding the limits of the task's workspace. We also plan to enhance the simulator by integrating an augmented reality feature that displays a realistic range of the procedure. A visual bar representing the distance between the surgical tools, specifically the graspers, will achieve this. This bar provides immediate feedback on the spacing used during the procedure. The colour coding of the bar is intuitive and informative: red signifies that the tools are nearing the edge of the camera's view with the clear beep sound that was mentioned, indicating a potential loss of visibility. Yellow represents a moderate range, alerting the user to adjust accordingly, while green indicates the optimal spacing range. This ideal range, measuring 40x40 mm, was determined in consultation with two experienced surgeons we collaborated with earlier in the semester, Dr. Ahmed Nasr, and Dr. Georges Azzie.

Furthermore, to enhance the precision and efficiency of our gesture recognition model as mentioned in section 6.2, we have begun implementing a new deep-learning architecture that offers several advantages over our current development. It comprises two models: a short-term model that accounts for spatial and short-term temporal features extracted from our simulator training clips and a long-term model designed to capture long-term temporal dependencies. The short-term model employs a pre-trained model based on the inflated Inception3D (I3D), initially trained on the Kinetics-400 dataset, to extract features. The benefit here is that the Kinetics dataset consists of action-oriented videos, meaning the pre-trained model is adept at extracting meaningful features from such content and can be fine-tuned for our surgical tasks to create a more robust model. All the enhancements mentioned in this section have been initiated and are poised for continuation by the succeeding group in the next iteration of the project.

10. Conclusion

The development of the Pediatric Laparoscopic Surgery Simulator has been a significant endeavour, incorporating advanced augmented reality technology, machine learning, comprehensive feedback mechanisms, and an advanced user interface (UI). This project created a portable, user-friendly, and cost-effective simulator to enhance the training experience of pediatric laparoscopic surgery trainees, providing them with a platform to evaluate their performance without constant expert supervision.

Throughout this project, we successfully integrated hardware improvements, including the use of sensors and microcontrollers, to capture precise data on tool movements and bidirectional forces applied during surgical tasks. The software implementation streamlined the data acquisition, cleaning, and organization process, facilitating the efficient analysis of trainee performance. A notable achievement of this project is developing a constructive feedback system that leverages gesture recognition and Dynamic Time Warping algorithms. This system provides real-time feedback on hand movements, adherence to surgical techniques, and comparison of trainee performance with expert benchmarks. The feedback is presented in various forms, including visual feedback, tool trajectory, aligned motion signals, tutorial videos, and augmented reality components, enriching the learning experience and fostering self-improvement. The simulator's user interface has undergone significant enhancements, offering an interactive and intuitive platform for trainees to engage with the simulator. Features like video playback, voice-assisted navigation, and a comprehensive evaluation menu provide a seamless training experience.

Future improvements for the simulator include the integration of additional surgical tasks, refinement of the gesture recognition model, and the incorporation of new augmented reality features to provide more immersive and realistic training. The ongoing development and refinement of this simulator hold immense promise, inspiring a future where pediatric laparoscopic surgery training is more accessible, effective, and advanced. In conclusion, the Pediatric Laparoscopic Surgery Simulator significantly advances surgical training tools. It offers a platform that is innovative but also practical and effective in enhancing the skills of trainees.

References

[1] “Laparoscopic Surgery - What is it? | ASCRS,” *fascrs.org*. https://fascrs.org/patients/diseases-and-conditions/a-z/laparoscopic-surgery-what-is-it

[2] “What Tools are Used in Laparoscopic Surgery | Laparoscopic.MD,” https://www.laparoscopic.md/surgery/instruments#:~:text=Laparoscopic%20instruments%20are%20made%20of.

[3] “Laparoscopy in Children,” *myhealth.alberta.ca*. https://myhealth.alberta.ca/health/Pages/conditions.aspx?hwid=acg9455

[4] Bojan Gavrilović, A. S. Fahy, B. Carrillo, A. Nasr, J. Ted Gerstle, and Georges Azzie, “Development of an Open-Source Laparoscopic Simulator Capable of Motion and Force Assessment: High Tech at Low Cost,” *Journal of Laparoendoscopic & Advanced Surgical Techniques*, vol. 28, no. 10, pp. 1253–1260, Oct. 2018, doi: https://doi.org/10.1089/lap.2018.0126.

[5] P. K. H. Tam, “Current topic: Laparoscopic surgery in children,” *Archives of Disease in Childhood*, vol. 82, no. 3, pp. 240–243, Mar. 2000, doi: https://doi.org/10.1136/adc.82.3.240.

[6] E. Westwood, B. Malla, J. Ward, R. Lal, and K. Aryal, “The Impact of a Laparoscopic Surgery Training Course in a Developing Country,” *World Journal of Surgery*, vol. 44, no. 10, pp. 3284–3289, Jun. 2020, doi: https://doi.org/10.1007/s00268-020-05606-y.

[7] M. Nagendran, C. D. Toon, B. R. Davidson, and K. S. Gurusamy, “Laparoscopic surgical box model training for surgical trainees with no prior laparoscopic experience,” *Cochrane Database of Systematic Reviews*, Jan. 2014, doi: https://doi.org/10.1002/14651858.cd010479.pub2.

[8] Y. A. Oquendo, E. W. Riddle, D. Hiller, T. A. Blinman, and K. J. Kuchenbecker, "Automatically rating trainee skill at a pediatric laparoscopic suturing task," *Surgical Endoscopy*, vol. 32, no. 4, pp. 1840–1857, Oct. 2017, doi: https://doi.org/10.1007/s00464-017-5873-6.

[9] P. Korzeniowski, C. S. Chacon, V. R. Russell, S. A. Clarke, and F. Bello, "Virtual Reality Simulator for Pediatric Laparoscopic Inguinal Hernia Repair," *Journal of Laparoendoscopic & Advanced Surgical Techniques*, vol. 31, no. 11, pp. 1322–1330, Nov. 2021, doi: https://doi.org/10.1089/lap.2020.0423.

[10] G. Azzie *et al.*, "Development and validation of a pediatric laparoscopic surgery simulator," *Journal of Pediatric Surgery*, vol. 46, no. 5, pp. 897–903, May 2011, doi: https://doi.org/10.1016/j.jpedsurg.2011.02.026.

[11] H.-C. Hur, D. Arden, L. E. Dodge, B. Zheng, and H. A. Ricciotti, "Fundamentals of Laparoscopic Surgery: A Surgical Skills Assessment Tool in Gynecology," vol. 15, no. 1, pp. 21–26, Jan. 2011, doi: https://doi.org/10.4293/108680810x12924466009122.

[12] "FLS Manual Skills Written Instructions and Performance Guidelines Important Scoring Information." Available: https://www.flsprogram.org/wp-content/uploads/2014/03/Revised-Manual-Skills-Guidelines-February-2014.pdf

[13] S. Miles and N. Donnellan, "Learning Fundamentals of Laparoscopic Surgery Manual Skills: An Institutional Experience With Remote Coaching and Assessment," *Military Medicine*, Apr. 2021, doi: https://doi.org/10.1093/milmed/usab170.

[14] M. Varras, N. Nikiteas, V. Varra, F. Varra, E. Georgiou, and C. Loukas, "Role of laparoscopic simulators in the development and assessment of laparoscopic surgical skills

in laparoscopic surgery and gynecology (Review)," *World Academy of Sciences Journal*, Mar. 2020, doi: https://doi.org/10.3892/wasj.2020.41.

[15] J. J. Park, J. Tiefenbach, and A. K. Demetriades, "The role of artificial intelligence in surgical simulation," vol. 4, Dec. 2022, doi: https://doi.org/10.3389/fmedt.2022.1076755.

[16] S. M. B. I. Botden, S. N. Buzink, M. P. Schijven, and J. J. Jakimowicz, "Augmented versus Virtual Reality Laparoscopic Simulation: What Is the Difference?," *World Journal of Surgery*, vol. 31, no. 4, pp. 764–772, Mar. 2007, doi: https://doi.org/10.1007/s00268-006-0724-y.

[17] C. Sewell *et al.*, "Providing metrics and performance feedback in a surgical simulator," *Computer Aided Surgery*, vol. 13, no. 2, pp. 63–81, Jan. 2008, doi: https://doi.org/10.3109/10929080801957712.

[18] Brian, "All About the I2C Standard & Protocol. How I2C Works | Circuit," Circuit Crush, Aug. 19, 2020. https://www.circuitcrush.com/i2c-tutorial/.

[19] "LPC1768: SPI Programming," Tutorials. https://www.exploreembedded.com/wiki/LPC1768:_SPI_Programming

[20] Last Minute Engineers, "Interface MPU6050 Accelerometer and Gyroscope Sensor with Arduino," Lastminuteengineers.com, 2021. https://lastminuteengineers.com/mpu6050-accel-gyro-arduino-tutorial/

[21] Alldatasheet.com, "PMW3389DM-T3QU Datasheet(PDF) - PIXART imaging inc..," ALLDATASHEET.COM - Electronic Parts Datasheet Search. [Online]. Available: https://www.alldatasheet.com/datasheet-pdf/pdf/1179019/PIXART/PMW3389 DM-T3QU .html.

[22] Design and Implementation Of A Sensing Unit For Tactile Vision Aid (Theoretical Background) - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/16-PC-Optical-Mouse-a-Anatomy-of-the-Device-b-Working-Principle-Reprinted-from_fig10_344253007

[23] N. Mezzomo, M. Sutherland, and T. Priscu. (2023). "Haptic Simulator for Pediatric Laparoscopy Training." SYSC Dep., Carleton University. Ottawa, Canada.

[24] N. Agnihotri, "How load cells work," Engineers Garage. https://www.engineersgarage.com/load-cells-types-working/

[25] "Single point load cell H10A," BOSCHE. https://www.bosche.eu/en/scale-components/load-cells/single-point-load-cell/single-point-load-cell-h10a

[26] Brachman, R. J., & Levesque, H. J. (2009). Knowledge representation and reasoning. Elsevier.

[27] Liu, W. et al. (2016). SSD: Single Shot MultiBox Detector. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds) Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science (), vol 9905. Springer, Cham. https://doi.org/10.1007/978-3-319-46448-0_2 [28] Tzutalin. (2023). LabelImg. GitHub repository. https://github.com/tzutalin/labelImg

[28] Tzutalin. (2023). LabelImg. GitHub repository. https://github.com/tzutalin/labelImg

[29] Simard, P. Y., Steinkraus, D., & Platt, J. C. (2003). "Best practices for convolutional neural networks applied to visual document analysis". In Proceedings of the Seventh International Conference on Document Analysis and Recognition, 958-963. DOI:10.1109/ICDAR.2003.1227801.

[30] TensorFlow 2 Detection Model Zoo. Available at: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md

[31] Gazis, A.; Karaiskos, P.; Loukas, C. Surgical Gesture Recognition in Laparoscopic Tasks Based on the Transformer Network and Self-Supervised Learning. *Bioengineering* **2022**, *9*, 737. https://doi.org/10.3390/bioengineering9120737

[32] J. Carreira and A. Zisserman, "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017, pp. 4724-4733, doi: 10.1109/CVPR.2017.502. keywords: {Videos;Three-dimensional displays;Kinetic theory;Two dimensional displays;Kernel;Feature extraction;Solid modeling},

[33] B. Thuraisingham, “A primer for understanding and applying data mining.” It Professional, 2(1), 28-31, (2000). Available: https://ieeexplore.ieee.org/abstract/document/819936.

[34] M. Müller, “Dynamic time warping.” In: Information retrieval for music and motion, 69-84, (2007). Available: https://doi.org/10.1007/978-3-540-74048-3_4.

[35] H. Li, “On-line and dynamic time warping for time series data mining.” Int. J. Mach. Learn. & Cyber. 6, 145–153, (2015). Available: https://doi.org/10.1007/s13042-014-0254-0.

[36] N. Chauhan, “Dynamic Time Warping (DTW) Algorithm in Time Series,” the AI dream, (2022). Available: https://www.theaidream.com/post/dynamic-time-warping-dtwalgorithm-in-time-series.

[37] S. LaValle, Yaw, pitch, and roll rotations, https://msl.cs.uiuc.edu/planning/node102.html

Appendix A (GitHub Repository)

All code and setup instructions for the main program, sensors, and other related files are available on the following GitHub repository:

https://github.com/SYSC4907-24/Pediatric-Laparoscopic-Surgery-Simulator.git