July 11, 2021

Dr. W. Craig Scratchley
School of Engineering Science
Simon Fraser University
Burnaby, BC
V5A 1S6

Re: ENSC 405W/440 Design Specifications for Smart Swim by Smart Swim Analytics

Dear Dr. Scratchley,

Attached to this letter is a document outlining the design specifications for the Smart Swim as requested for ENSC 405W. The Smart Swim relies upon machine learning models to detect and track a swimmer using swimming competition footage or in real-time using a tilt-pan camera, and to provide an estimation of the swimmer's stroke rate. It provides valuable data for competitive swimmers and trainers.

The design specifications report will propose and justify designs to meet the requirement specifications that are necessary for the alpha phase development of the product. The problem is split into four systems, namely the Data Collection System, the Model Training System, the Real-Time Swimmer Tracking System, and the Swimmer Positioning System. Under each system, the report outlines the corresponding functional requirements such as hardware and software specifications.

Smart Swim Analytics is a team of 6 senior Computer Engineering students, Tim Woinoski, Kiran Brar, Kuro Chen, Ethan Cai, and Ray Kim, and Systems Engineering student Kudus Elbo-Iswadi. Together, we aim to utilize our combined knowledge to make the concept of Smart Swim a reality.

Thank you for taking the time to read the Smart Swim requirement specifications. For any questions or concerns, please feel free to contact Tim Woinoski at tim_woinoski@sfu.ca.

Kind Regards,

Tim Woinoski
Project Lead
Smart Swim Analytics

DESIGN SPECIFICATION

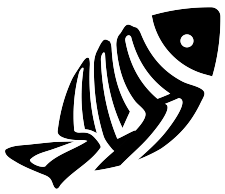
for

Smart Swim Analytics System

Version 1.0

Prepared by

Kudus Elbo-Iswadi, Gurkiran Brar, Ethan Cai, Kuro Chen, Ray Kim, Tim Woinoski



Smart Swim Analytics

July 12, 2021

1. Abstract

Smart Swim Analytics aims to help Canadian swimmers improve their performance in a more cost effective and inclusive manner than presently available. Thus we have designed Smart Swim, inspired from research complete by Tim Woinoski [1]. This system will determine a swimmer's stroke length and stroke rate, which are important metrics for measuring performance. Not only will the system be quicker than the current method of measuring these metrics by hand, it will also be cheaper than such methods [2].

The product will consist of an annotation tool that lets users easily annotate videos for use in training swimmer models. Such models are vital for the completion of the proposed tasks. Annotation will only need to be performed once at a given pool, after which the models will be ready to use with the Real-time Swimmer Tracking System and Swimmer Positioning System, which detect the swimmers in real time and determine their position in the pool, respectively. This document details the proposed design for all parts of this system.

Contents

List of Figures

List of Tables

Revision History

Name	Date	Reason For Changes	Version
Tim Woinoski	2021-07-05	Start of Document	0.0

2. Introduction

2.1. Purpose

This documentation outlines the detailed design for every part of the Smart Swim system, a complete Automated Swimming Analytics System. As the system changes and improves this document will be updated until it is deemed necessary to split the document into sub-modules. Such revisions will be recorded in Table 1.

2.2. Product Perspective

In swimming, data is collected for swimmers at competitions utilizing ORV of swimmers, by a variety of people. Of particular note, RaceTek [2] provides Canadian swimmers with racing data, as seen in Figure 2.1, for every major and some minor competitions. RaceTek has been doing so for many years now but unfortunately, their services are very expensive, and data can not be released to the public. There are other groups that provide similar services in practice environments, such as Form Swim goggles and TritonWare [14, 15].

Referencing Figure 2.1, the basic services provided by RaceTek include; Swimmer velocity (meters per Second), stroke rate analysis (Strokes Per Minute), and stroke distance analysis (Distance Per Stroke in Meters) collected from videos recorded at swim competitions. All calculations are done by hand using video footage to analyze swimming. These tasks are time consuming, but have the potential to be automated by a computer. The goal is to create a system that will automate these tasks while also making more accurate measurements of the swimmer. As such this system will greatly reduce the cost of collecting swimmer data. In addition, such a system would save coaches and athletes across the world many hours of analyzing post-race videos manually. Additionally, the ability to process swimming footage in real-time is more beneficial to a swimmer than if they are given feedback days later.

This automated swimming analytics system is a new self-contained project. Automated swimming analytics has been attempted by a few organizations [16, 17] and its completion would put Canada ahead in terms of swimming data analytics. In addition to this, the assumptions made in this proposed system will be even less than the ones made in [16, 17]. As such this project has additional complexities added to it. But, in turn, has some additional benefits such as flexibility in terms of recording footage.

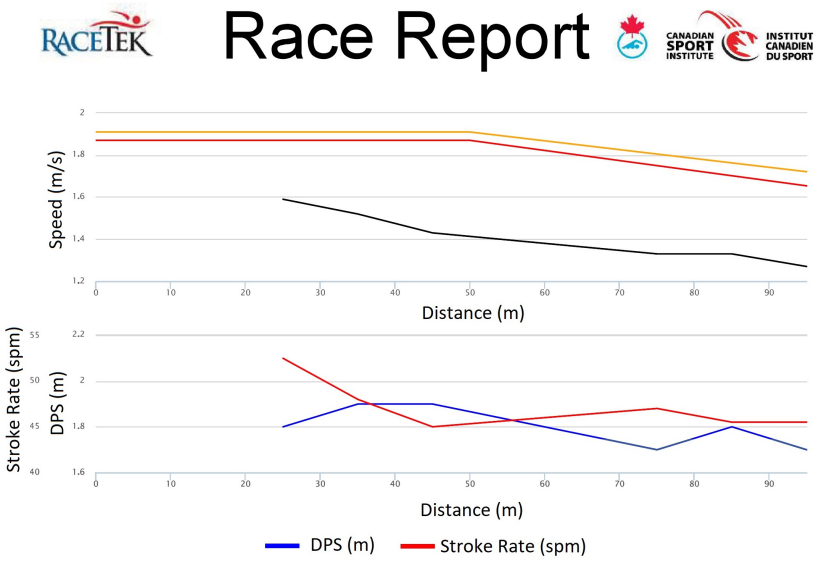


Figure 2.1.: A modified example report from RaceTek [2], found under Video Race Analysis (VRA). Source: Adapted from [2]

2.3. Document Conventions

This documentation was created using LaTeXbook style. Its format was copied from the template given by [18].

Note that this document is building off of the requirements document "REQUIREMENTS SPECIFICATION for Smart Swim Analytics System Version 1.0", thus there may be reuse of information. This also includes referencing requirements, which are of the format "RX.X.X", where X is a number.

2.4. Intended Audience and Reading Suggestions

This document is for developers and project managers. The document is organized as follows.

- Chapter 3 gives a general overview of the components of Smart Swim.
- Chapter 4 gives details on how each component will be implemented.
- Chapter 5 gives a quick summary of the major design choices made in this document.

The appendix contains supporting information for this document.

- Appendix A gives test plans for the major modules of Smart Swim.
- Appendix B gives backup options for any proposed designs that include major risk.

- Appendix C gives detail into the graphical design and it's considerations for the Smart Swim system.
- Appendix D is given as a reference to people who are unfamiliar with swimming

2.5. Project Scope

The Smart Swim system allows for the collection of swimmer analytics from general non-static footage of swimmers, also known as general overhead race video (ORV) as seen in Figure 2.2. This system makes almost no assumptions about the manner in which the footage is or was collected. In other words it is very robust. Many analytics can be collected by humans utilizing ORV, however only two will be collected with this system. Such metrics will be Strokes Per minute (SPM) and Distance Per Stroke (DPS), defined in Appendix D. There are five main sub-systems to this system, more detail is given in Chapter 3. Each part works together in order to allow the analytics system to be utilized in any swim competition setting while maintaining robust analytics results quickly and efficiently.

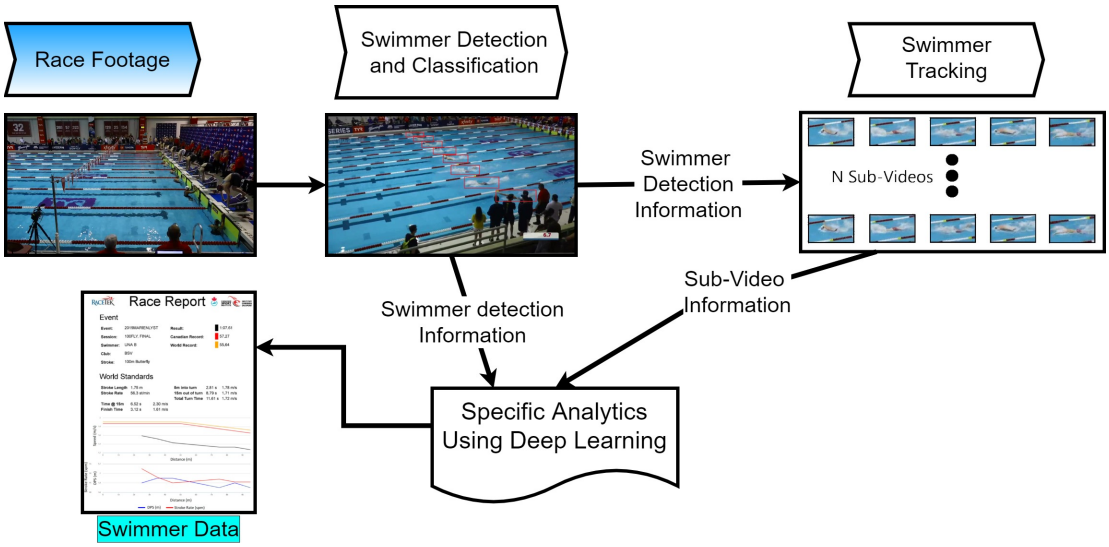


Figure 2.2.: A pictorial representation of the Smart Swim System

3. System Overview

This chapter gives high level overview of the proposed Smart Swim system and how it will be implemented. The chapter references all of the high level components in the system and directs the reader to the section in the Detailed Design chapter that corresponds to that component.

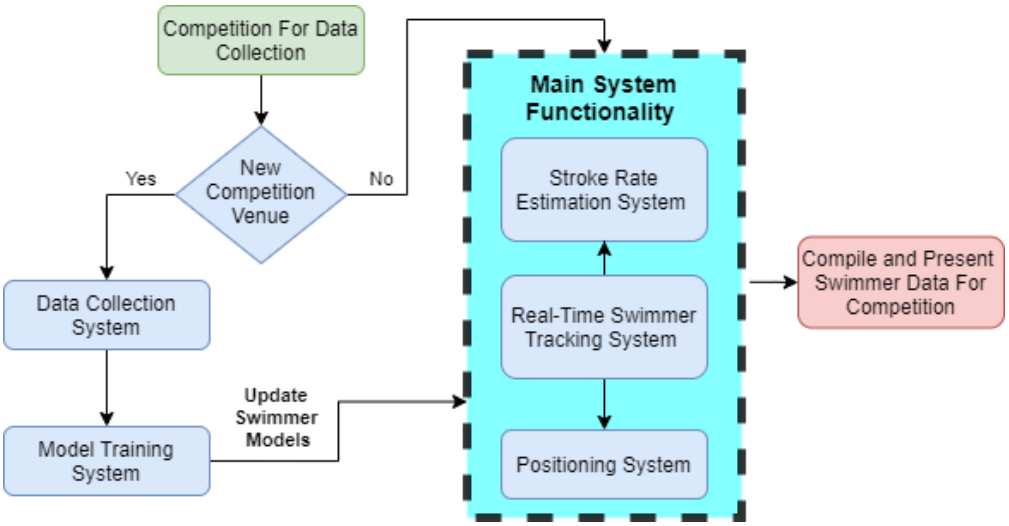


Figure 3.1.: A summary of the expected use of system functions

3.1. Product Functions

The motivation for this project is to develop a system that will automate the collection of swimming analytics in swim competition videos using image-based processing methods and tracking algorithms. Specifically, those analytics will be the SPM and DPS, as defined in appendix D, of a particular swimmer selected before the commencement of a race. In addition to the collection of these metrics the analytics system must work for a general competition setting. To solve these problems we propose the following sub-systems which support each other in the following order.

1. A data collection system: Which allows for the collection of training and test data for creating and augmenting swimmer models.

2. A model training and testing system: Which utilizes collected swimmer data to train, augment and test swimmer models.

3. A real-time swimmer tracking system: Which utilizes the swimmer models to direct a camera to track a swimmer of interest.

4. A swimmer stroke analysis system: Which utilizes the swimmer models to collect the stroke count data required as the first part of one of the required product function.

5. A swimmer positioning system: Which takes the real-time swimmer tracking results and estimates the positions of the swimmers in the pool; the second part of the required product function.

3.2. Intended Use of System

With the systems functions defined, the intended use of each of the systems functionalities can be given. When given a competition venue for data collection utilizing the Smart Swim system the following procedures should be followed, which can also be seen in Figure 3.1. If the competition venue of interest is new or if the view point at which the competition will be analysed from is different[1], then the Data Collection System is utilized by the Data Collection Specialist[2]to create training data in the competition of interest. This data will be used for updating or creating models utilizing the Model Training System, facilitated by the Sports Analyst[2] so that Smart Swim will work in a general setting. The Main System Functionality is monitored and overseen by Footage Collectors[2]. The output of the main functionality is received by the Sports Analyst, interrupted and possibly modified such that it can utilized most effectively by the Athletes and Coaches[2].

3.3. System Architecture

Figure 3.2 details how all the sub-systems are implemented with software, hardware, and a data base. It also depicts how all parts of the system work together to implemented the mentioned sub-systems which in turn, collects the required data in the database; that being analytics files, annotated footage and model weights. Lines with arrows depict flow of information or data, to and from user, hardware and database.

The blue boxes denote all hardware required in running the proposed functionality. The following is a summary of the hardware utilized in this project.

- Section 4.3.1: Embedded GPU and Processor
- Tilt-Pan Hardware and Camera

[1]For more information on how to place the Smart Swim Analytics system in the most optimal spot for data collection see Section 4.1.

[2]see Appendix C for more details on Data Collection Specialist, Footage Collectors, Sports Analysts and Athletes/Coaches.

  - Section 4.3.2: Raspberry Pie 2 Model B [19]
  - Section 4.3.4: FLIR Firefly DL Image Sensor [11]
  - Section 4.2.2: Hitec HS-422 Servo Motor [3]
- Linux OS Computer
- High Powered GPUs
- General Computer (Linux OS, Windows OS, Mac OS)

The green boxes, denote software implementations which run on the hardware they are connected to by a line with a circle attached.

- Section 4.4.3: Data Collection Software
- Section 4.4.6: Swimmer Positioning Software
- Section 4.4.2: Swimmer Tracking Software
- Section 4.4.1: Tilt-Pan Firmware (Implemented in the Tilt-Pan Hardware)
- Section 4.4.5: Swimmer Analytics Software
- Section 4.4.7: Model Training Software

Lastly, the database is represented by the purple circle 4.4.4.

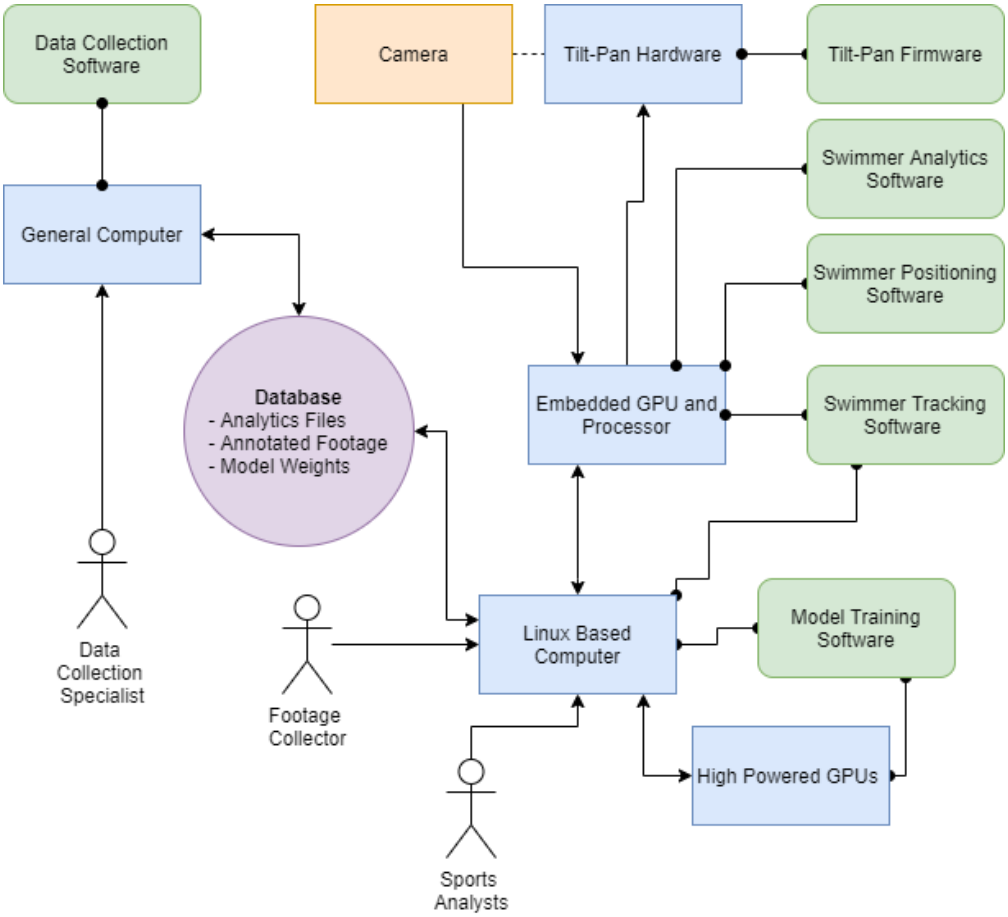


Figure 3.2.: The main system functions and how they work together

4. Detailed Design

This chapter gives detailed design specifications for the construction of the Smart Swim Analytics system. This chapter is organized into the following sections.

- System Placement Design 4.1
- Mechanical Design 4.2
- Hardware Design 4.3
- Software Design 4.4

The details of how each item in each section will be implemented are found in one of the above mentioned sections. Appendix B gives backup options to some the designs given in this section associated with uncertainty and risk.

4.1. System Placement Design

This section details how a user should place the Smart Swim equipment for optimal data collection in a general pool setting when reading this section refer to Appendix D.2 for details.

4.1.1. Camera Placement Considerations

The course of a pool has a big effect on where cameras are placed. In an SCM or SCY race, one camera can easily capture an entire race. In an LCM race, however, the pool is long enough to require multiple cameras to avoid reducing the relative size of the swimmers to a very small fraction of the frame. Another point worth considering is that one venue can host all three courses depending on how the pool is built. Less than optimal performance could be achieved if a model were to be trained on one configuration and then be tested on another, even in the same pool, without other similar training data to support it.

When analysing swimmers, the most useful camera positions are the ones where the swimmers are racing in a direction perpendicular to the view of the camera. A Cartesian coordinate system can be applied to the pool to describe the location of cameras and thus the camera angles that can be accomplished. The x-axis will be defined as the closest wall in the view of the camera such that the swimmers are swimming parallel to it. The z-axis of this coordinate system is parallel to the vertical direction relative to the pool, and the y-axis is parallel to the direction of the blocks or the wall that swimmers turn on.

In general, the position of the camera along the z-axis is usually at the pool level or the viewing level. Pool level is roughly the height of a standing person on deck and viewing level is a height at which all swimmers can be put in view by the camera. Ideally, all races are recorded at viewing level. The advantage to having pool-level footage is that quick and complex movements such as dives can be more easily captured, but the disadvantage is not having a good view of all swimmers. The advantage of view-level footage is that all swimmers can be seen, but swimmers appear smaller in the scene and so it is more difficult to see what a swimmer is doing.

The camera position along the x-axis, i.e., along the length of the pool, usually captures one of the following three views: (1) the dive view, which means the camera is anywhere before the ten-meter mark closest to the blocks; (2) the turn view, which means the camera is past the ten-meter mark of the turn end of the race; and (3) a mid-pool view, which is anywhere between the dive view and the turn view. Typically, three main (x,z) camera positions are used: pool-level at dive view, viewing-level at mid-pool view, and pool-level at turn view.

4.1.2. Conclusion

When placing the Smart Swim Analytics system, as noted in the previous section, the optimal position will be in the middle of the pool at a viewing level of 45° greater than

the pool level. Any deviation from this location would result in bias for data collection towards a particular direction of viewing. At this point, there is no proof that this is in fact the optimal placement of the Smart Swim system. For future work, attempt will be made to analytically prove more specific placement parameters based on camera quality an other unforeseen parameters.

4.2. Mechanical Design

This section details the design of all required mechanical part of the Smart Swim system. The section is broken down into the following parts.

- Embedded systems Housing 4.2.1
- Pan and Tilt Functionality 4.2.2
- System Power Supply 4.2.3

4.2.1. Embedded Systems Housing

The embedded GPU allows the Smart Swim system to deal with computationally expensive algorithms allowing for real-time tracking. Without the embedded GPU it is unlikely that real-time tracking could take place. As such, the embedded GPU must be close to the camera when swimming analytics is taking place. This means the embedded GPU must be able to operate in a pool environment without sustaining damage in expected conditions.

Pool Conditions Around the Embedded Systems

Pools have a wide variety of architecture, of all possible pools a good way to divide them is by considering outdoor and indoor pools. An outdoor pool environment is much like any stadium and exposed to the elements of the pools location. An indoor pool has much more control of its environment however it is common to find most indoor pools to have high temperature and humidity. In addition to the architectural and geographical environment that come with pools one must consider the situational environment. Pools contain water and thus there is always a possibility of splashing. As such, the embedded systems housing must be designed for some subset of these circumstances.

For the alpha stage of development the Smart Swim System will be design to be housed in an indoor pool setting. This allows the design to disregard outdoor environmental conditions due to being outdoors. **For the beta stage** all possible circumstances should be considered.

Embedded GPU Housing

The embedded GPU housing design specifications are given in table 4.1. The main concern with its design is temperature control, splash resistance and shock resistance.

Tilt-Pan Controller Housing

The tilt-pan controller housing design specifications are given in table 4.2. The main concern with its design is temperature control, splash resistance and shock resistance.

Spec ID	Specification Details	Req ID
D4.2.1.1 *Stage: beta*	The splash resistant housing will be made from a waterproof plastic material that encases the embedded GPU	R4.3.1.5
D4.2.1.2 *Stage: beta*	The shock resistant housing will be made from a non-conductive plastic material that encases the embedded GPU	R4.3.1.5
D4.2.1.3 *Stage: beta*	The housing should regulate the temperature utilizing fans and strategically placed holes	R4.3.1.5
D4.2.1.4 *Stage: beta*	The housing allow the system to be powered by allowing the power cords access to the embedded system	R4.3.1.5

Table 4.1.: Embedded GPU Housing Design Specification

Spec ID	Specification Details	Req ID
D4.2.1.1 *Stage: beta*	The splash resistant housing will be made from a plastic material that encases the controller	R4.3.1.1
D4.2.1.2 *Stage: beta*	The shock resistant housing will be made from a non-conductive plastic material that encases the controller	R4.3.1.1
D4.2.1.3 *Stage: beta*	The housing should regulate the temperature utilizing strategically placed holes	R4.3.1.1
D4.2.1.4 *Stage: beta*	The housing allow the system to be powered by allowing the power cords access to the embedded system	R4.3.1.1

Table 4.2.: Tilt-Pan Controller Housing Design Specification

Spec ID	Specification Details	Corresponding Requirements ID
D4.2.2.1 *Stage: beta*	The servo should provide tilting and panning functionalities	R4.3.1.1 R4.3.1.2
D4.2.2.2 *Stage: beta*	The servo should provide stable and accurate turning functionality in order for letting the camera centering the object	R4.3.1.6.b

Table 4.3.: Pan and Tilt Functionality Design Requirements.

4.2.2. Pan and Tilt Functionality

In order for a stable, solid and consistent platform for supporting the camera, the Lynx B - Pan and Tilt Kit was chosen as it meets all the requirements for the supporting of camera and providing motion trail.
The Lynx B - Pan and Tilt Kit is made up of these following components.

- 1x Aluminum Multi-Purpose Servo Bracket
- 1x Aluminum "C" Servo Bracket
- 2x HS-422 (57 oz. in.) Standard Servo

The PoC version of Pan and Tilt Functionality assembly is mainly composed of the items listed above and see in Figure 4.1. The high quality aluminum servo bracket works great for making multi-axis joints for use and supporting heavy duty servo hinge. The HS-422 Standard Servo is able to provide stable and accurate turning functionality even with heavy load onto the circular turning plate(the exact parameter for setting of the turning speed and angle will be tested on further experiments in Beta phase). Considering of the power consumption, we decided to apply the 5V DC directly from the Raspberry Pi to the servo, and the reason is in our design the purpose of the motor is to provide slow and continuous movement thus we do not need to meet the maximum value of the voltage to achieve quick turning speed which is apparently inaccurate for the camera.

Table 4.4 lists some of the specifications of the HS-422 Standard Servo that are relevant to our system.

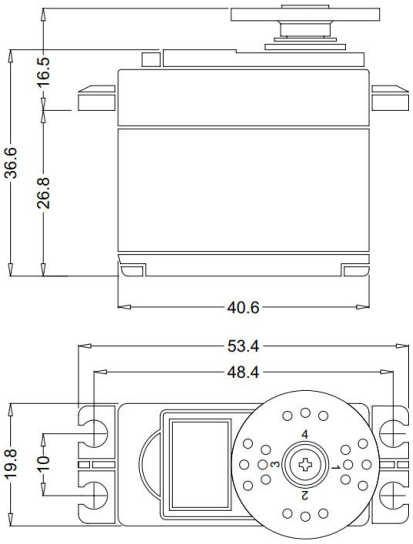
4.2.3. System Power Supply

In this part we mainly demonstrated the power distribution and supporting solution to the whole system. The power consuming components in our system and the corresponding specification data are listed below.

(a) C Servo Bracket

(b) Multi-purpose Servo Bracket

(c) HS-422 Standard Servo



(d) Dimension Structure of Servo Motor

Figure 4.1.: Examples of the mechanical parts. Source: [3]

- Nvidia Jetson Tx2 Development Kit
- Software execution consumption
- Raspberry Pi 2 Model B board
- 2x HS-422 Standard Servo
- FLIR Firefly DL camera
- All the connection wires and electronic elements overhead

Figure 4.2.: Servo Assembly Sample 01. Source: [3]

Figure 4.3.: Servo Assembly Sample 02. Source: [3]

Specification type	**Value**
Operation Voltage Range	4.8V to 6.0V
Operating Speed	0.21sec / 60 degree at no load, at 4.8V 0.16sec / 60 degree at no load, at 6.0V
Stall Torque	3.3kg.cm(45.82oz.in), at 4.8V 1kg.cm(56.93oz.in), at 6.0V
Operating Angle	45 degree/one side pulse traveling 400 microsec
Operating Range	180 degree
Direction	Clock wise/pulse traveling 1500 to 1900 microsec
Current Drain	8mA/IDLE, 150mA/no load running
Connector Wire Length	300mm(11.81in)
Dimensions	40.6 x 19.8 x 36.6mm(1.59 x 0.77 x 1.44in)
Weight	45.5g(1.6oz)

Table 4.4.: Servo Motor Design Requirements. Source: [10]

Specification type	Value
Power	DC 5V 800mA

Table 4.5.: Raspberry Pi 2 Model B Power Requirements. [11].

Specification type	Value
Power Consumption	2.2 W
Power Consumption	5 V via USB 3.1 interface

Table 4.6.: Sensor Power Requirements. [11].

Table 4.5 lists the power specification of the Raspberry Pi 2 Model B that are relevant to our system.

Sensor

Table 4.6 lists power specifications of the FLIR Firefly DL camera.

Referring to all the relevant specification data from Table 4.7, we can make a calculation on the total consumption of the power we need. Since the Jetson Development Kit comes with a power supply itself, and the Raspberry Pi is supported with 2x 5V DC output power supply which is suitable and sufficient for the two HS-422 Standard Servo to process. Based on the theoretical value, we came up with a potential solution for supporting the entire system, which is to use a power adapter that accepts an input voltage, for example 10V, and produces a 5V DC voltage for the Raspberry Pi and use safety resisters to protect all other components which needs charging.

Spec ID	Specification Detail	Req ID
D4.2.3.1 *Stage: beta*	The system power supply must provide sufficient and safe source of power to all electronic devices in the whole system	None

Table 4.7.: System Power Supply Design Specification

Spec ID	Specification Details	Req ID
D4.3.1.1 *Stage: alpha*	The Embedded GPU processor must be able to handle at least 65.8 billion floating point operations in real-time	R4.3.1.5
D4.3.1.2 *Stage: alpha*	The Embedded GPU processor must have 8GB of RAM	R4.3.1.5
D4.3.1.3 *Stage: alpha*	The Embedded GPU processor must no consume more than 15w worth of power	None
D4.3.1.4 *Stage: alpha*	The Embedded GPU processor must be CUDA enabled	None

Table 4.8.: Embedded GPU Design specifications

4.3. Hardware Design

This section details the design of all required hardware of the Smart Swim system. The section is broken down into the following parts.

- Embedded GPU Processor 4.3.1
- Tilt Pan Controller 4.3.2
- Hardware Communication 4.3.3
- Sensor 4.3.4

4.3.1. Embedded GPU Processor

For our YOLO V3 model, we require the GPU to handle roughly 65.8 billion FLOPS in real-time. A single frame is approximately 372 KB, and at 30 FPS, we have 11 mBPS. Table 4.8 summarizes these requirements.

4.3.2. Tilt Pan Controller

For the Tilt-Pan Controller, we have 2 options. First, the Raspberry Pi 2 Model B which we are currently using as the controller for the Alpha phase prototype due to its availability and second, an Arduino Uno Rev 3 micro-controller board. Although the Raspberry Pi is currently sufficient for demonstrating PoC, we will switch to using the Arduino Uno in Beta phase as it is more effective for real-time applications. This is due to the fact that the Raspberry Pi operating system "Raspbian", a standard Linux based operating system, is not designed for real-time operation. Whereas Arduino, in contrast, executes only the application code on its processor without an operating system, therefore having minimal delays which is necessary for real-time applications. [20]. And since our system is classified as a "firm real-time application", the efficiency of the controller is required for executing code within a guaranteed time frame.[20].

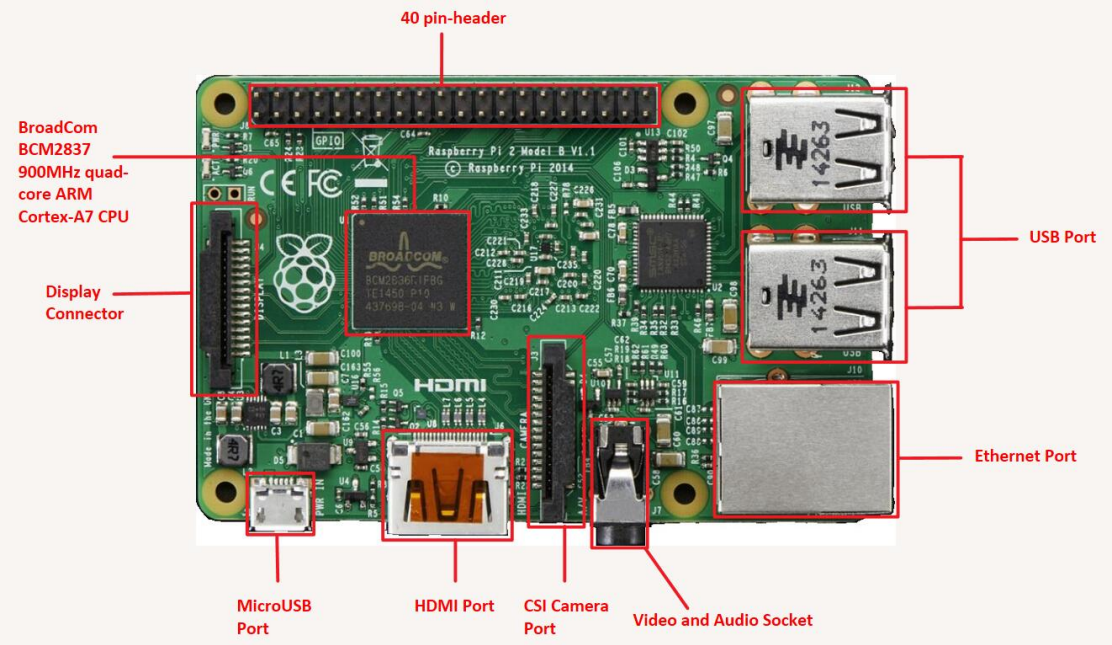


Figure 4.4.: Raspberry Pi 2 Model B plan structure display. Source: [4]

Spec ID	Specification Details	Req ID
D4.3.2.1 *Stage: beta*	The Raspberry Pi is equipped with Unix OS environment and easy to operate	R4.4.2.3
D4.3.2.2 *Stage: beta*	The PWM signal from the Raspberry is able to provide complete control to the movement of servo	R4.3.1.1 R4.3.1.2

Table 4.9.: Raspberry Pi 2 Model B Design Requirements.

The Raspberry Pi 2 Model B is equipped with a 900MHz quad-core Arm Cortex-A7 CPU and 1GB RAM. The reason we choose this board as a controlling platform in our system depends on various aspects: operating convenience; supported power supply; applicable Pulse Width Modulation(PWM) output signal; multiple connection ports for different applications.

4.3.3. Hardware Communication

GPIO Specification

The Figure 4.5 shows the specific layout of the Raspberry Pi 40-head GPIO pins. The 5V power pins are what we use for supporting the power to the 2 HS-422 Standard Servos. These pins are connected directly to the Pi's power input and provides a pull about 1.5A output current and the exact value varies by Pi model and the adapter used.

GPIO 12 and 18 are what we use to connect to the servos. It is used by Pulse Width

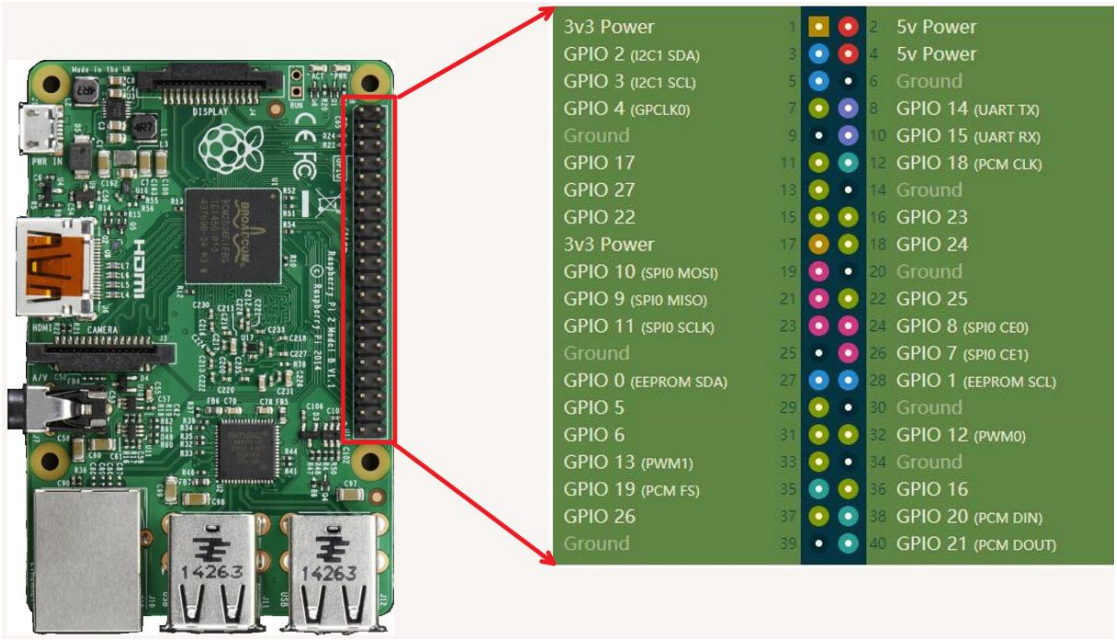


Figure 4.5.: GPIO 40-head pin structure display. Source: Adapted from [4], [5]

Spec ID	Specification Details	Req ID
D4.3.3.1 *Stage: alpha*	The communication must be faster than the frame rate of the video taken in order to keep up with the commands and have short latency	R4.3.1.1
D4.3.3.2 *Stage: beta*	The communication must be easy to set up by the user, preferably automatic	R4.3.1.1

Table 4.10.: Tilt-Pan controller and GPU communication requirements.

Modulation(PWM) to provide a clock signal to an external device. The PWM output is particularly useful in combination with some devices which requires very specific timings.

Communication Between Tilt-Pan Controller and GPU

This connection is required for the purpose of communicating control signals from the GPU to the tilt pan controller. This will allow for the detection and tracking models to position the camera as required in real time. Table 4.10 gives an overview of the resulting design specifications.

4.3.4. Sensor

The sensor to be used in the real-time system is the FLIR Firefly DL camera. We selected this camera because it is designed to be embedded into real-time systems and allows for a trained neural network to be deployed to it and reduce system complexity and cost by

making decisions on-camera. [11]. The camera will communicate with the Jetson board by connecting a USB 3.1 Type-A to Micro-B (Locking) Cable to the Jetson board's USB 3.0 Type A port. This connection will allow for the data input from the camera to be used in the detection and tracking models as well as for the feedback control determined by the detection and tracking models.

Spec ID	Specification Detail	Req ID
D4.3.4.1 *Stage: beta*	The output data from the camera must be in good quality for further operation	None
D4.3.4.2 *Stage: beta*	The camera must be under fully control by the command from the Jetson board	R4.4.1.1 - R4.4.1.6
D4.3.4.3 *Stage: beta*	The data process of centering the frame from the camera detection must be within the range of the tolerance in pixels	R4.3.1.6.b

Table 4.11.: Sensor Design Requirements.

4.4. Software Design

This section details the design of all required software of the Smart Swim system. The section is broken down into the following parts.

- Tilt Pan Controller Firmware 4.4.1
- Real-time Detection and Tracking Software 4.4.2
- Data Annotation Software 4.4.3
- Database 4.4.4
- Swimmer Analytics Software 4.4.5
- Swimmer Positioning Software 4.4.6
- Deep Learning Utilities Software 4.4.7
- System Testing Software 4.4.8

4.4.1. Tilt Pan Controller Firmware

For the tilt pan controller firmware, our initial plan is to use the Raspberry Pie [19]. As such the firmware should be compatible with the Raspberry Pie. The purpose of the controller is to allow another hardware to control the position of the camera that is tracking the swimmers. Based on the commands from the hardware the Tilt Pan Controller will send the necessary signals to the servos that positioning the camera. The firmware has four controls: on, off, angle tilt, angle pan which are summarized in table 4.12.

4.4.2. Real-time Detection and Tracking Software

Applying Detection and Tracking Models on Real-Time Video Stream

To apply the tracking and detection models in real-time, we will use OpenCV to capture a video stream and use threading to reduce latency in the stream. To capture the video

Spec ID	Specification Details	Req ID
D4.4.1.1 *Stage: alpha*	Given a tilt command the firmware must command the tilt servo to move to the specified angle $[0°, 180°]$	R4.3.1.1
D4.4.1.2 *Stage: alpha*	Given a pan command the firmware must command the pan servo to move to the specified angle $[0°, 180°]$	R4.3.1.2
D4.4.1.3 *Stage: alpha*	Given an on command the firmware must be ready to take commands for the pan and tilt control	R4.3.1.1
D4.4.1.4 *Stage: alpha*	Given an off command the firmware must return the servos to 90° and then kill power to the motors	R4.3.1.1

Table 4.12.: Tilt-Pan Controller firmware commands

stream we will use the OpenCV "cv2.VideoCapture" method. [21]. We will access the camera's video stream by calling the VideoStream class with OpenCV and start reading frames in real-time. [21]. Once we have imported the frames, we must convert them to a format that is compatible with the Detection model (blob), and that will be used as the input for our detection neural network. [22]. Furthermore, once the frame is passed into the detection model, the real-time detection and tracking process will follow a similar route to that of detection and tracking with recorded swimming footage. Additionally, the detection and tracking models will also send a signal to the controller to tilt or pan the camera as required.

4.4.3. Data Annotation Software

The API will be used by Data Collection Specialists to annotate footage of swimmers for use as ground truth data. This data is used as training, validation and test data for the model (refer to Figure 3.2).

Algorithm

The API is based off of previous work done by our project lead Tim Woinoski [9], which was written in C++, and uses OpenCV [23] and Eigen [24].With [9] you can currently do the following:

a) Annotate a video with boxes to create a region of interest (ROI) around visible portions of the swimmer

b) Count strokes of a swimmer

c) Create data for YOLO (currently uses YOLOv3) with the option to also generate the JPEG images

d) Create sub-videos of swimmers

Spec ID	Specification Details	Req ID
D4.4.3.1 *Stage: alpha*	The API is written in C++ and uses OpenCV and Eigen to handle the video annotations	R4.1.1.1, R4.1.1.2
D4.4.3.2 *Stage: alpha*	The API uses [9] to allow the user to annotate videos of swimmers at a comfortable speed for the user	R4.1.1.2-R4.1.1.7, R4.1.1.9-R4.1.1.12
D4.4.3.3 *Stage: alpha*	The API uses [9] to allow the user to create sub-videos of single swimmers from the original video in which there were multiple swimmers	None
D4.4.3.4 *Stage: alpha*	The API uses [9] to allow the user to create annotations for (a swimmer's) stroke counting	R4.1.2.1-R4.1.2.9
D4.4.3.5 *Stage: alpha*	The API uses [9] to allow the user to create create data for YOLO with the option to generate corresponding JPEG images or not	None
D4.4.3.6 *Stage: alpha*	The API [9] uses YOLOv3 and this will be updated to YOLOv4	None

Table 4.13.: Data annotation software algorithm requirements.

Spec ID	Specification Details	Req ID
D4.4.3.7 *Stage: alpha*	The API will allow the user to specify the position of a swimmer in a given frame. This task will be done with aid from OpenCV and [9].	R4.1.3.1

Table 4.14.: Data annotation software algorithm requirements continued.

Note however, regarding the current use of YOLOv3, we aim to upgrade to the more recent YOLOv4. This is all captured in table 4.13.

To track swimmers the API uses Simple Online and Real-Time Tracking (SORT) in conjunction with the Hungarian Algorithm and the Kalman Filter. The Hungarian Algorithm will optimally map the Kalman Filter predictions to the predictions of the detection system [1]. For the creation of sub-videos, the API uses OpenCV's KalmanTracker class, as well as an implementation of the Hungarian Algorithm. This is all part of [9].

In addition to the actions the API currently allows us to perform, we would like to add the ability to annotate positioning of a swimmer. That is, we want to allow the user to specify the position of the swimmer at given intervals. This is captured in table 4.14.

User Interface

The foundation [9] is command line. A GUI is necessary to ensure all users feel comfortable with using the application. We have chosen QT [25] to build the GUI. We chose QT due to how it works well with OpenCV, and also because it is a popular option for cross-platform app development which ensures we have a variety of online sources for

Spec ID	Specification Details	Req ID
D4.4.3.8 *Stage: beta*	The API will use QT [25] for building the GUI.	R4.1.1.2, R4.1.1.6, R4.1.1.9- R4.1.1.13, R4.1.2.1- R4.1.2.9

Table 4.15.: Data annotation software UI requirements.

Spec ID	Specification Details	Req ID
D4.4.3.7 *Stage: beta*	The API will use QwT [26] for improving upon the graphing capabilities of [9].	R4.1.2.8, R4.1.2.9

Table 4.16.: Data annotation software UI requirements continued.

help as we have little experience in this area. This is captured in table 4.15.

However there are aspects of [9] that we plan on keeping or just improving upon. Specifically, the way a user draws on the frame when creating a ROI around the swimmer (with help from OpenCV) and how that looks for the user is something we will keep. On the other hand, the way the user can view the stroke annotations of a single swimmer by playing the video and a vertical bar (representation of current time) moving across the sinusoid (annotations) is also something we will still have but improve upon. This is captured in table 4.16.

Backup options to this design can be found in Appendix B.1.

4.4.4. Database

There is a lot of data created from use of the Data Collection System (refer to Section 4.4.3). This data needs to be stored in a database for easy access and updating, as there will be a lot of data for each new swimming pool.

Detection Data

Once annotations are completed using the Data Collection System (refer to Section 4.4.3) the data of all the annotations must be stored to be used for training/updating the model (refer to Figure 3.2). For a single video, the following information about the video must be stored:

a) Frames Per Second (FPS)

b) Resolution (RES)

c) Total number of frames in the video (TNF)

d) Skip rate (frames to skip between annotations)

Spec ID	Specification Details	Req ID
D4.4.4.1 *Stage: beta*	The Detection Data API will use MongoDB [27] and JSON.	R4.1.1.8, R4.1.1.10, R4.1.1.14, R4.1.1.16

Table 4.17.: Detection data requirements regarding the database

In addition to this, for every frame there must be information stored on what annotations have been made. In a single frame, for every annotation done on a swimmer (referred to as box) the following must be stored:

a) The x location of the top left corner of the box

b) The y location of the top left corner of the box

c) The height of the box

d) The width of the box

e) The class of the swimmer (if they are: on block, diving, swimming, underwater, turning, or finishing)

f) The lane number the swimmer is in

Thus, if you considered a vector consisting of variables to represent each item listed, if there are three annotations in a single frame (for three individual swimmers) then there would be three vectors to uniquely classify each annotation for that single frame.

An option is to use JSON to store the information, and use MongoDB [27]. This is captured in table 4.17. However we have other options present in B.4.

Stroke Data

There is a lot of data created from sub-videos when annotating strokes of a (single) swimmer using the Data Collection System (refer to Section 4.4.3). Once annotations for the stroke of a single swimmer are complete, the following information must be stored for the video:

a) Frames Per Second (FPS)

b) Resolution (RES)

c) Total number of frames in the video (TNF)

Since we have singled out that swimmer there is just the following that must be stored per frame:

a) Frame Number

Spec ID	Specification Details	Req ID
D4.4.4.2 *Stage: beta*	The Stroke Data API will use MongoDB [27] and JSON.	R4.1.2.11

Table 4.18.: Stroke data requirements regarding the database

Spec ID	Specification Details	Req ID
D4.4.4.3 *Stage: beta*	The wall and lane detection data API will use MongoDB [27] and JSON.	None

Table 4.19.: Wall and lane detection data requirements regarding the database

b) Y value on the sinusoid

c) If the swimmer is *swimming* or *not swimming*

d) What stroke is being performed by the swimmer (fly, back, brest, or freestyle)

An option is to use JSON to store the information, and use MongoDB [27]. This is captured in table 4.18. However we have other options present in B.4.

Wall and Lane Detection Data

For pool wall identification, data is required to train, validate and test wall detection methods.

The chosen method is to assign each wall, which is one of the four edges of the rectangular pool) and each line divider two points (a pair of points). This pair of points' linear connection defines the entirety of the wall or lane. In addition to this pair of points, there would be two classes that can be assigned: wall and lane dividers.An example of what this would look like can be found in Figure 4.6.

However, if we encounter issues, there is another method presented in B.4.

We could store in a JSON file the following:

- Frame Number
- Pair of points (can be stored as starting and ending point)
- Classification (wall or lane divider)

This is captured in table 4.19.

Swimmer Analytics Tracking Data

Swimmer analytics tracking data describes a "tracklet", which is a trajectory of as wimmer as a function of frames. This tracklet is described by the following

a) Frame Number

Figure 4.6.: Example of annotated pool for Wall and Lane Detection

Spec ID	Specification Details	Req ID
D4.4.4.4 *Stage: beta*	The swimmer analytics tracking data will be stored in a CSV file.	None

Table 4.20.: Swimmer analytics tracking data requirements

b) Unique tracklet ID

c) The x location of the top left corner of the box

d) The y location of the top left corner of the box

e) The height (h) of the box

f) The width (w) of the box

g) The class of the swimmer (if they are: on block, diving, swimming, underwater, turning, or finishing)

This will be most easily encoded by storing the data in a CSV file where each row is (frame number, tracklet ID, x, y, h, w, class). This is captured in table 4.20.

However other options do exist for storing data in case this proves insufficient/inconvenient (refer to Appendix B.4).

Swimmer Analytics Stroke Data

See section D.1 for details on what the swimmer position data looks like.

The general data (regarding that session) that needs to be stored is:

Spec ID	Specification Details	Req ID
D4.4.4.5 *Stage: beta*	The swimmer analytics stroke data will be stored in a CSV file.	None

Table 4.21.: Swimmer analytics stroke data requirements

Spec ID	Specification Details	Req ID
D4.4.4.6 *Stage: beta*	The swimmer analytics position data will be stored in a CSV file.	None

Table 4.22.: Swimmer analytics position data requirements

a) Frame rate in frames per second

b) Length of the pool

For each frame we need to store:

a) Frame Number

We can store this in a CSV file as this is simple data. This is captured in table 4.21.

Swimmer Analytics Position Data

See section D.1 for details on what the swimmer position data looks like.
The general data (regarding that session) that needs to be stored is:

a) Frame rate in frames per second

b) Length of the pool

This however is already covered by 4.4.4.
For each frame we need to store:

a) Frame Number

b) The position of the swimmer relative to the starting point (starting blocks)

c) The position of the top of a stroke (if it is the top of the stroke or not)

d) The distance per stroke (derived from positioning mentioned above, frame rate, and frame numbers)

We can store this in a CSV file as this is simple data referring to a single swimmer.This is captured in table 4.22.

4.4.5. Swimmer Analytics Software

Swimmer analytics software analyzes the swimmer and produces the final data utilized by the athletes and coaches. Due to time constraints this section is not completed but its details can be observed in [28]. In future documents this will be completed.

Swimmer Stroke Estimation

This estimation method is applied in the sub-video. Assume we assign a stroke value, called the s-value, for each frame in the video. This is done by fitting Range [0, 1] to the sine curve of each stroke cycle by any given sub video. s value of 1 represents the top of a stroke and 0 represents the bottom of the stroke. Also, the s value is set to be 0.5 for not swimming status. In addition, for each frame, a flag will be given to specify whether The swimmer in the picture is taking a swimming class. lesson.[28].

Swimmer Stroke models

The swimmer stroke models is trained through a large amount of video sources. The model is trained to output the value of s for each input frame. More details can be observed in [28]. In future documents this will be completed.

4.4.6. Swimmer Positioning Software

Overall Swimming Positioning Pipeline

Swimming positioning is a key factor in the swimmer tracking system. The pipeline for determining the position of a swimmer is as below

a) Create a high-quality panorama by given a video as an input source

b) Match each frame in the panorama for later calculation

c) Detect the edge of the pool in the entire panorama picture

d) Calculate out the estimated position of the swimmer with known length of the pool

Automatic Panorama Creation

To obtain the panorama, the system is given a non-static video with tracking data of one swimmer from a single camera as one source of input. There is a lot of necessary steps that need to be done for creating the panorama:

a) Input the video source file and convert the particular video into pictures

b) Delete some extra pictures especially some blurry pictures

c) Input 10-20 pictures as a small group for stitching to improve the quality of the entire panorama picture

d) Delete some poor quality pictures in which the control point value is below 2.4

e) Restitching the panorama again for better result

In Panorama Frame Matching

To determine the location of the swimmer we need to match the particular frame with the panorama

a) Input the video source file and convert the video into different frames

b) Looking for the particular frame that we need

c) Match the frame with entire panorama

d) Get the sequence number of the frame out of the total frames

Pool Edge Detection

For detecting the edge of the pool, there are two options available

a) Scale-invariant feature transform

b) create another supporting design to increase the productivity and usability of the pool edge detection.

The SIFT method make use of points matching between different views of a 3-D scene and view-based object recognition In this case, we extract the edge of the swimming pool which consists of multiple points and straight lines as key point localisation and compare these feature descriptions with some pool edge pictures in the database to determine if this object is the edge of the swimming pool.[29]

Position Estimation

Once we know the sequence number of the frame out of the total frames and the total length of the pool we can calculate out the estimated position and the ground truth position of a swimmer.

4.4.7. Deep Learning Utilities Software

Smart Swim has deep learning system for the automated swimming analytics. It is designed with You-Only-Look-Once (YOLO) object detection. YOLO is the neural network object detection algorithm. It is able to detect objects and train model by the deep learning detection [6].

The benefits of YOLO is extremely fast processing time, and it outperforms other object detection algorithms by prediction-based detection instead of region-based detection. Figure 4.7 illustrates how the detection can be predicted. It divides the image into grids and for each grid cell predicts bounding boxes, confidence for those boxes, and class probabilities. These predictions are encoded as an $S \times S \times (5B + C)$ tensor [6].

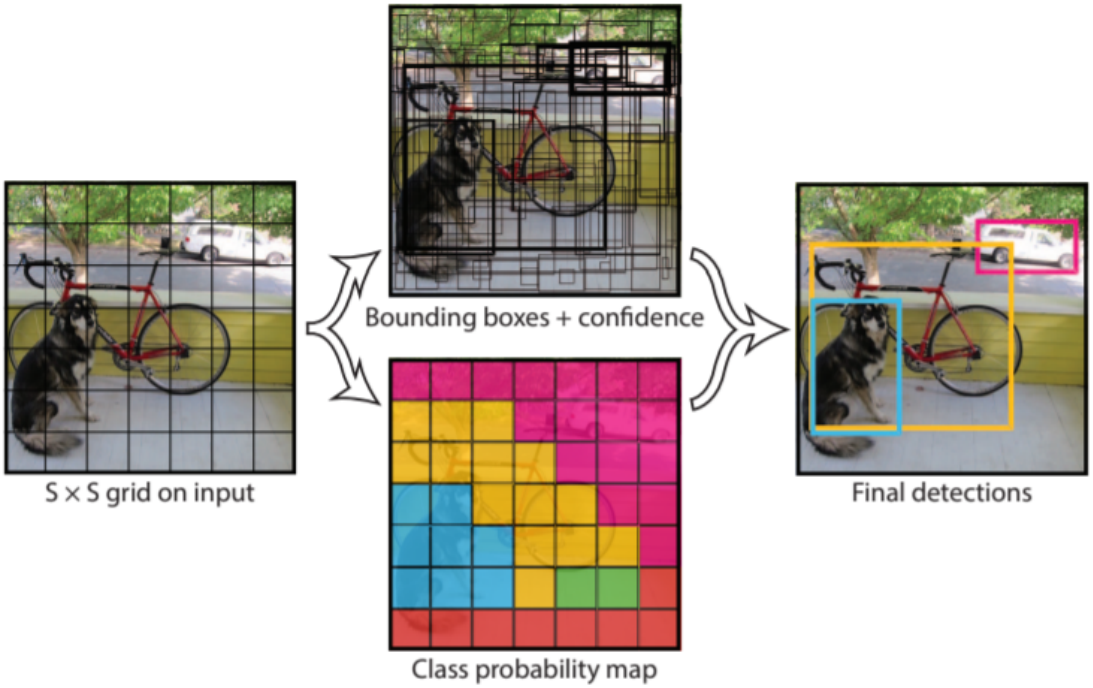


Figure 4.7.: YOLO Detection Process. Source: [6]

The limitation of YOLO is detection of tiny objects. As Smart Swim is to detect swimmers and their swimming phases, which are relatively not samll, it is negligible limitation.

Figure 4.8 is the experimental comparison done with Microsoft Common Objects in Context between YOLO and other object detection algorithms [7]. The x label is the batch time taken. The y label is the accuracy of object detection. It shows YOLO is faster than other algorithms and the scaled YOLO is more accurate.

Darknet, the open source framework, is used to implement YOLO. Darknet supports fast computation leveraging CUDA, which is a parallel computing platform and programming model of NVIDIA GPU. As Darknet is the open source framework, it is easy to install and enables to customize software as per the project problem. Darknet requires GTX980-equivalent or higher specifications to run [30].

Model Training

Model training is the process for a machine to learn swimming phases through detecting objects from images. Based on Pareto principles, annotated datasets are to be divided into 80/20 groups between training and testing [31]. 80% of data will be used for model training.

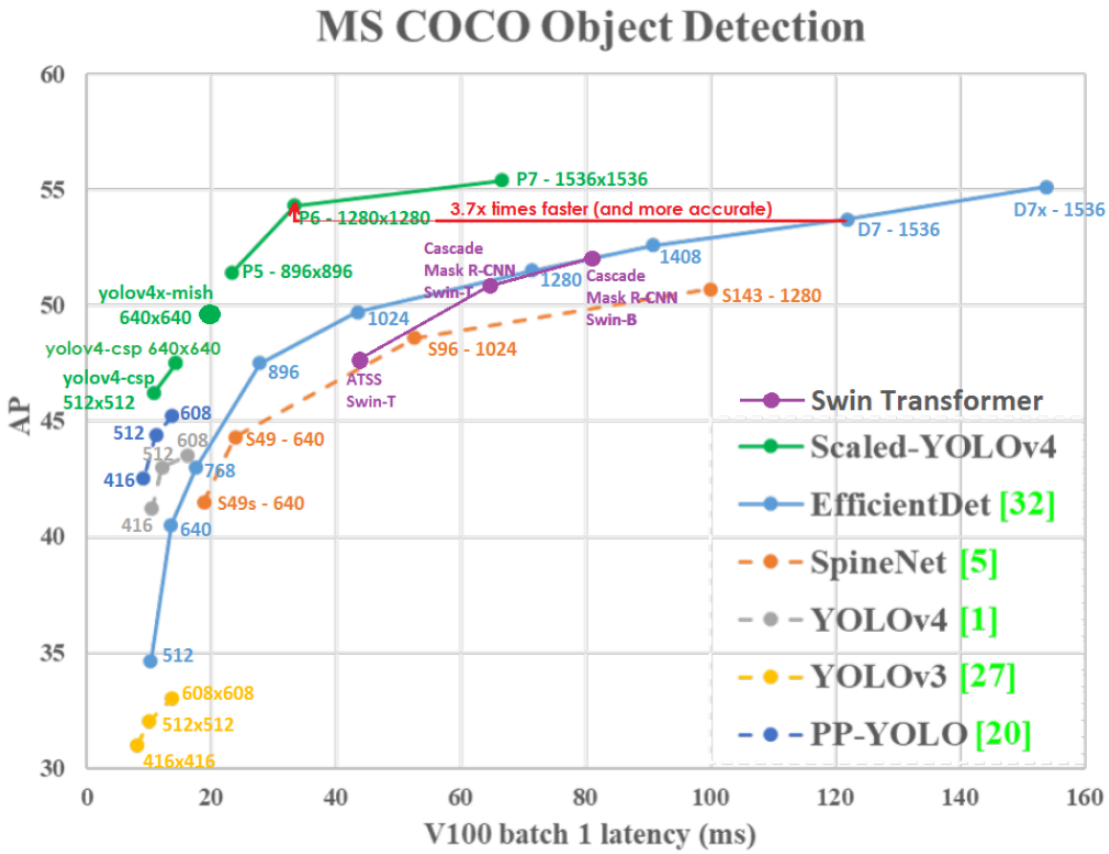


Figure 4.8.: YOLO vs. Other Algorithms. Source: [7]

The deep learning model is trained on the Compute Canada Cedar clusters to borrow NVIDIA GPU with CUDA.

nodes	cores	available memory	CPU	storage	GPU
576	32	125G or 128000M	2 x Intel E5-2683 v4 Broadwell @ 2.1Ghz	2 x 480G SSD	-
96	32	250G or 257000M	2 x Intel E5-2683 v4 Broadwell @ 2.1Ghz	2 x 480G SSD	-
24	32	502G or 515000M	2 x Intel E5-2683 v4 Broadwell @ 2.1Ghz	2 x 480G SSD	-
24	32	1510G or 1547000M	2 x Intel E5-2683 v4 Broadwell @ 2.1Ghz	2 x 480G SSD	-
4	32	3022G or 3095000M	4 x Intel E7-4809 v4 Broadwell @ 2.1Ghz	2 x 480G SSD	-
114	24	125G or 128000M	2 x Intel E5-2650 v4 Broadwell @ 2.2GHz	1 x 800G SSD	4 x NVIDIA P100 Pascal (12G HBM2 memory)
32	24	250G or 257000M	2 x Intel E5-2650 v4 Broadwell @ 2.2GHz	1 x 800G SSD	4 x NVIDIA P100 Pascal (16G HBM2 memory)
192	32	187G or 192000M	2 x Intel Silver 4216 Cascade Lake @ 2.1GHz	1 x 480G SSD	4 x NVIDIA V100 Volta (32G HBM2 memory)
640	48	187G or 192000M	2 x Intel Platinum 8160F Skylake @ 2.1Ghz	2 x 480G SSD	-
768	48	187G or 192000M	2 x Intel Platinum 8260 Cascade Lake @ 2.4Ghz	2 x 480G SSD	-

Figure 4.9.: Cedar Resources Availability. Source: [8]

Figure 4.9 shows the available computing resources of Cedar. According to it, Cedar

lets the system do deep learning with up to 192×4 GPU cores. Compute Canada Cedar can be accessed by SSH with the research credentials.

```
[yolo]
mask = 0,1,2
anchors = 10,13,  16,30,  33,23,  30,61,  62,45,  59,119,  116,90,  156,198,  373,326
classes=6
num=9
jitter=.3
ignore_thresh = .7
truth_thresh = 1
random=1
```

Figure 4.10.: YOLO Deep Learning Definition

Figure 4.10 and 4.11 show how Smart Swim is configured for deep learning.
classes tells Smart Swim is designed to analyze swimming by grouping behaviours into six classes: starting, diving, swimming, underwater, turning, finishing.
max_batches determines the number of batches to run. It is recommended to set the value at least 2000 $\times$ **classes** to guarantee the accuracy of training.
anchors and **num** define that the detection starts with anchor boxes at **anchors** coordinates to divide images into **num** blocks, then adjust box locations depending on the overlapped objects on images.
batch determines the number of images to process in a single batch.
subdivisions determines the number of parallelism.
Images are resized into **width** $\times$ **height**.
channels determines into how many blocks images are divided and reorganized during training. [7]
Model training can be customized by modifying these configurations for the best accuracy.

Model Augmentation

Model augmentation is the process to randomize given datasets and train model with different variations of data. This makes the model learn more thoroughly with all possible qualities of data. Darknet has the ability to do model augmentation together with model.

Figure 4.10 and 4.11 show how the model augmentation is configured.
random requests to perform model augmentation as many as **random** value every 10 batchs of training.
jitter controls to randomly change size of images to the ratio between $(1 - 2 \times$ **jitter**$)$ and $(1 + 2 \times$ **jitter**$)$.
angle controls to randomly rotate images by **angle** in degree.
saturation and **exposure** control to randomly change the staturation and brightness by the rates.
hue controls to randomly change the color of images by the rate [7].

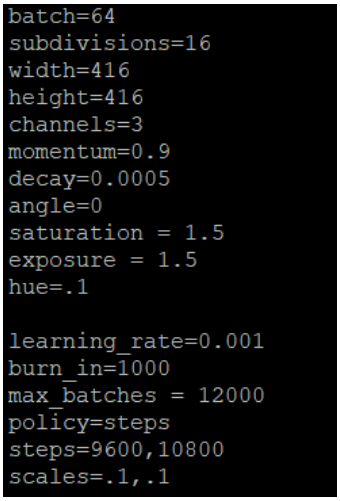


Figure 4.11.: YOLO Deep Learning Configuration

Model Testing

Model testing is the process to verify accuracy of the trained model and adjust the accuracy if needed. The remaining 20% data is to be used as testing datasets for model testing.

Model Data Packaging

Data collection produces semi-structured annotated data paired with corresponding images. Darknet expects each annotated frame data to be present in individual files. Data curating to restructure the data is to be handled with Python. The curated files are used to train and test model. The trained and tested model is to be available between Compute Canada and NVIDIA Jetson via SSH. If SSH is not a feasible option, the model can be transferred to NVIDIA Jetson to use the model on a single platform, which does not require communications between two platforms.

4.4.8. System Testing Software

Due to time constraints this section has not been completed. This is a release design specification. In future documents this will be completed.

Spec ID	**Specification Details**	**Corresponding Requirements ID**
D4.4.7 - Model Training/Augmentation *Stage: Alpha*	Model can be trained by learning from detecting images and annotated data. Model is trained with variety of data from augmentations.	R4.2.1.1 R4.2.1.3
D4.4.7 - Model Testing *Stage: Beta*	The model can be tested with testing datasets to validate the trained model	R4.2.1.2 R4.2.1.4

Table 4.23.: Deep Learning Design Specifications.

5. Conclusion

The design needs and possibilities to build Smart Swim have been outlined in this document, which is a living document. The following has been outlined in this document:

1. System Placement Design 4.1:
    - Optimal placement of system to maximize performance is recommended in this section
2. Mechanical Design 4.2
    - Embedded systems housing being waterproof to minimize damage risks from water
    - Pan and Tilt functionality hardware details
    - System power supply needs to ensure proper functionality are detailed
3. Hardware Design 4.3
    - Specifications of the embedded GPU processor are detailed
    - Specifications of the Raspberry Pie controlling the tilt pan assembly are detailed
4. Software Design 4.4
    - Requirements of the firmware for the Raspberry Pie controlling the tilt pan assembly are detailed
    - Use of OpenCV to perform real-time detection and tracking
    - Use of an existing API (courtesy of Tim Woinoski) and combining with QT to optimize user experience
5. Database needs for the various data streams
6. Swimmer Positioning Software 4.4.6
    - The use of a panorama to determine a swimmer's position
    - The methods of detecting pool edges and position estimation in use of a panorama
7. Defined the use of CUDA and GPU requirements for model building and augmentation

Bibliography

[1] W. Woinoski, "Automated swimming analytics using deep neural networks," *SFU Thesis Library*, 2020, unpublished.

[2] RaceTeck, "Raceteck," 2018, [Online], Available: https://racetek.ca/ [Accessed: Nov 19 2019].

[3] H. RCD, "Hs-422 deluxe standard servo," 1999, [Online], Available: https://hitecrcd.com/products/servos/analog/sport-2/hs-422/product [Accessed: July 10 2021].

[4] lady ada, "Introducing the raspberry pi 2 - model b," https://learn.adafruit.com/introducing-the-raspberry-pi-2-model-b?view=all#what-to-watch-out-for [Accessed: July 10 2021], n.d.

[5] Gadgetoid, "Raspberry pi pinout," https://pinout.xyz/ [Accessed: July 10 2021], n.d.

[6] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[7] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," 2020.

[8] C. Canada, "Compute canada cedar wiki," 2017, [Online], Available: https://docs.computecanada.ca/wiki/Cedar [Accessed: July 10 2021].

[9] tjwoinosk, "swim_annotator," 2021, [Online], Available: https://github.com/tjwoinosk/swim_annotator [Accessed: Jun 20 2021].

[10] Lynxmotion, "Announced specification of hs-422 standard deluxe servo," http://www.lynxmotion.com/images/data/hs422.pdf [Accessed: July 10 2021], n.d.

[11] T. Flir, "Firefly dl," 2019, [Online], Available: https://www.flir.ca/products/firefly-dl/ [Accessed: July 10 2021].

[12] N. Corporation, "Harness ai at the edge with the jetson tx2 developer kit," 2017, [Online], Available: https://developer.nvidia.com/embedded/jetson-tx2-developer-kit [Accessed: July 10 2021].

[13] NVIDIA, "Embedded system with jetson," 2014, [Online], Available: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/ [Accessed: June 10 2021].

[14] F. Form Swim, "Form," 2019, [Online], Available: https://www.formswim.com/ [Accessed: Nov 19 2019].

[15] Tritonwear, "Tritonwear, take the guesswork out of swimming faster," 2015, [Online], Available: https://www.tritonwear.com/ [Accessed: Nov 19 2019].

[16] C. Hudson, "Automated tracking of swimmers in the clean swimming phase of a race," Ph.D. dissertation, Sheffield Hallam University, 2015.

[17] A. Hall, B. Victor, Z. He, M. Langer, M. Elipot, A. Nibali, and S. Morgan, "The detection, tracking, and temporal action localisation of swimmers for automated analysis," *Neural Comput. Appl.*, pp. 1–19, 2020.

[18] J.-P. Eisenbarth, "Srs-ieee," 2016, [Online], Available: https://github.com/jpeisenbarth/SRS-Tex [Accessed: June 06 2021].

[19] R. FOUNDATION, "Raspberry pi 2 model b," 2015, [Online], Available: https://www.raspberrypi.org/products/raspberry-pi-2-model-b/ [Accessed: July 10 2021].

[20] Lee and A. C. Labs, "Arduino vs. raspberry pi: 9 crucial differences you need to know," 2019, [Online], Available: https://appcodelabs.com/arduino-vs-raspberry-pi-9-crucial-differences [Accessed: July 10 2021].

[21] A. Rosebrock, "Unifying picamera and cv2.videocapture into a single class with opencv," 2016, [Online], Available: https://www.pyimagesearch.com/2016/01/04/unifying-picamera-and-cv2-videocapture-into-a-single-class-with-opencv/ [Accessed: July 10 2021].

[22] ——, "Real-time object detection with deep learning and opencv," 2017, [Online], Available: https://www.pyimagesearch.com/2017/09/18/real-time-object-detection-with-deep-learning-and-opencv/ [Accessed: July 10 2021].

[23] OpenCV, "Opencv," 2021, [Online], Available: https://opencv.org/ [Accessed: Jun 20 2021].

[24] B. Jacob, G. Guennebaud, and et al, "Eigen," 2021, [Online], Available: https://eigen.tuxfamily.org/index.php?title=Main_Page [Accessed: Jun 20 2021].

[25] T. Q. Company, "Qt," 2021, [Online], Available: https://www.qt.io/ [Accessed: Jul 10 2021].

[26] U. Rathmann and J. Wilgen, "Qwt user's guide," n.d., [Online], Available: https://qwt.sourceforge.io/index.html [Accessed: Jul 10 2021].

[27] MongoDB, "Mongodb," 2021, [Online], Available: https://www.mongodb.com/ [Accessed: Jul 10 2021].

[28] T. Woinoski and I. V. Bajić, "Swimmer stroke rate estimation from overhead race video," in *2021 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*. IEEE, 2021, pp. 1–6.

[29] H. Senaratne, A. Mobasheri, A. L. Ali, C. Capineri, and M. Haklay, "A review of volunteered geographic information quality assessment methods," *International Journal of Geographical Information Science*, vol. 31, no. 1, pp. 139–167, 2017.

[30] J. Redmon, "Darknet: Open source neural networks in c," http://pjreddie.com/darknet/, 2013–2016.

[31] R. Dunford, Q. Su, and E. Tamang, "The pareto principle," *The Plymouth Student Scientist*, 2014.

[32] wxwidgets, "wxwidgets: Cross-platform gui library," 2021, [Online], Available: https://www.wxwidgets.org/l [Accessed: Jul 10 2021].

[33] Oracle, "Mysql," 2021, [Online], Available: https://www.mysql.com/ [Accessed: Jul 10 2021].

[34] ISO and Comity, "Information technology — artificial intelligence — overview of trustworthiness in artificial intelligence," International Organization for Standardization, Geneva, Switzerland, Standard, 2020, [Online], Available: https://www.iso.org/standard/77608.html [Accessed: Jun 13 2021].

[35] ——, "Artificial intelligence (ai) — assessment of the robustness of neural networks," International Organization for Standardization, Geneva, Switzerland, Standard, 2021, [Online], Available: https://www.iso.org/standard/77609.html [Accessed: Jun 13 2021].

[36] ——, "Information technology — security techniques — information security management systems — requirements," International Organization for Standardization, Geneva, Switzerland, Standard, 2013, [Online], Available: https://www.iso.org/standard/54534.html [Accessed: Jun 13 2021].

[37] ——, "Ergonomics of human-system interaction — part 161: Guidance on visual user-interface elements," International Organization for Standardization, Geneva, Switzerland, Standard, 2016, [Online], Available: https://www.iso.org/standard/60476.html [Accessed: Jul 10 2021].

[38] A. B. Craig, W. L. Boomer, and J. F. Gibbons, "Use of stroke rate, distance per stroke and velocity relationships during training for competitive swimming," in *International Symposium of Biomechanics in Swimming*, J. Terauds and E. W. Bedingfield, Eds., vol. 3. Baltimore University Park Press 1979, 1979, pp. 265–274.

[39] FINA, "Fina facilities rules." [Online]. Available: https://www.fina.org/sites/default/files/2017_2021_facilities_06102017_full_medium.pdf[Accessed:2020-07-22]

A. Supporting Test Plans

Table A.1 details the acceptance test plan to be evaluated at the end of the ENSC405W semester. The first row details a system of interest and the other details the necessary items each system must complete in order to be considered done.

Table A.1

Test Plan	**Acceptance Criteria**
Validate the swimmer detection data annotation system	- A user is able to open the application on Windows - A user is able to open a video to do data annotation with - A user is able to set the annotation frame rate after opening the video - A user is able to change and track swimmer's position and classes in the video during the annotation - The annotated data is produced upon completion
Validate the swimmer stroke data collection system	- A user can successfully annotate the position of a swimmers stroke as the video plays - A user is able to move forward and backwards through the video at an adjustable speed while annotating - A user can successfully clear annotations done on a video when entering edit mode while data exists for that video - When annotating, a user can specify if a swimmer is swimming or not as well as which stroke is being performed - For an annotated video, a user can view the graph of the sinusoidal signal and access it's frequency
Validate the model training system	- A user is able to input data-sets and the API has the ability to read them - After reading the input source given from the user, the API is able to distinguish training and test data-sets - After reading the input given source, The API has the ability to produce analytic results with testing data-sets - The API has the ability to train the model with given training data-sets

Continuation of Table A.1	
Test Plan	**Acceptance Criteria**
Validate the camera control	- The camera should perform tilt functionality - The camera should perform pan functionality - The camera should perform real-time image capturing and video recording functionality - The camera should perform correct movements corresponding to the commands from the tracking system - The error of the movements should be in a reasonable calculation range
Validate the real-time system	- The real-time tracking system should let the user to select a swimmer of interest to be tracked - The detection model should classify different state of the swimmer in the frame in real-time, and comparing the result with correct sample - The detection model should create box around the swimmer in the frame in real-time, and comparing the result with correct sample - The tracking model should send correct commands to the camera in order for performing correct movement - The system should generate various correct output files with unique names

Continuation of Table A.1	
Test Plan	**Acceptance Criteria**
Validate the positioning system	- A user is able to follow the swimmer's position during the whole process - A user is able to view the position where the swimmer standing at the diving platform. - A user is able to view the position where the swimmer diving into water - A user is able to view the position where the swimmer is under the water surface in the midway - A user is able to view the position where the swimmer is beyond the water surface in the midway - A user is able to give non-static video with tracking data of one swimmer from a single camera as one source of input - After reading the input source, the system is able to represent a post processing sequence of the position of the swimmer that removes impossibilities such as non-continuous velocity and acceleration - After evaluating input source data, the system is able to output a tendency analysis, like velocity, acceleration and deceleration rate of swimmer for performance analysis - After evaluating input source data, the system is able to output the accuracy of the system given the ground truth position of a swimmer
End of Table A.1	

Table A.1.: The table for the acceptance test plan to be presented at the end of the ENSC405W

B. Supporting Design Options

This section outlines backup Design options for the proposed options given in the main document.

B.1. User Interface API Backup Option

If QT does not work out for us, another popular option we can use to build the GUI is wxWidgets [32], which is a C++ library that will allow us to build the GUI cross-platform. This is a secondary option because it is also well known (we have a variety of sources to learn from), and it is mature [32].

B.2. Embedded GPU Processor Options

Table B.1 give different options for different embedded GPUs.

The GPU we will use to process the software is the NVIDIA Jetson TX2 module[12]. It is capable of handling 1.33 Trillion FLOPS which meets the YOLO V3 requirement. We decided to use this GPU firstly, due to its supercomputer graphic processing power with 2 Denver 64-bit CPUs and a Quad-Core A57 Complex, which is ideal for our application and secondly, because we have access to it through Simon Fraser University. The Jetson board will be used as the main computer to run the detection and tracking software and to communicate with the tilt pan controller to position the camera as required by the tracking model.

B.3. Communication Between Tilt-Pan Controller and GPU Design Options

In the main document communication between Tilt-Pan Controller and GPU is required. Table B.2 gives options on how this communication will be completed.

Assuming the use of a Raspberry Pie as a controller and a NVIDIA Jetson TX2 as a GPU. The Jetson board will communicate with the Pi through Ethernet. The 100 Base Ethernet port on the Raspberry Pi can be connected to the 10/100/1000BASE-T Ethernet port on the Jetson Board for Ethernet communication. In addition serial connections can be made between each boards GPIO pins.

Design Option	Specification
NVIDIA Jetson TX2	Jetson TX2 is a 7.5-watt supercomputer on a module that brings true AI computing at the edge. It's built around an NVIDIA Pascal-family GPU and loaded with 8 GB of memory and 59.7 GB/s of memory bandwidth. It features a variety of standard hardware interfaces that make it easy to integrate into a wide range of products and form factors.
NVIDIA Jetson Nano	The NVIDIA Jetson Nano Developer Kit is a small, powerful computer that lets you run multiple neural networks in parallel for applications like image classification, object detection, segmentation, and speech processing. All in an easy-to-use platform that runs in as little as 5 watts. It's the ideal solution for prototyping your next AI-based product and bringing it to market fast.
NVIDIA Jetson Xavier NX	Smaller than a credit card (70x45 mm), the energy-efficient Jetson Xavier NX module delivers server-class performance—21 TOPS (at 15 W) or up to 14 TOPS (at 10 W). It can run multiple modern neural networks in parallel and process data from multiple high-resolution sensors—a requirement for full AI systems. Available cloud-native support makes it easier than ever to develop and deploy AI software to edge devices.

Table B.1.: Embedded GPU Processor Options. Information from [12], [13]

Design Option	Specification
Ethernet Connection	Create scripts in the controller that polls a file waiting for updates, executing there contents, and deleting them once complete
Serial Connection	Set up a serial connection between GPIO pins which the controller can poll and execute as they come in

Table B.2.: Tilt-Pan controller and GPU communication Options

B.4. Database Backup Options

B.4.1. Detection Data Backup Options

If we have troubles with JSON we can use XML instead.

Another option is to use MySQL [33] instead of MongoDB [27] and create tables. One table would consist of the columns: frame number, x, y, h, w, c, l, **videoID**. There would be a second table with columns: FPS, RES, TNF, skip rate, **videoID**. The key would be videoID.

Lastly, in the worst case scenario we can store everything in CSV files. Though this may not be ideal since the data structure is a bit complicated for CSV, it is possible and a simple alternative.

B.4.2. Stroke Data Backup Options

The same options in B.4.1 apply here. However, regarding MySQL the following would apply. There would be one table with columns: frame number, y, swimming (yes or no), stroke, **videoID**. There would be a second table with columns: FPS, RES, TNF, **videoID**. The key would be videoID.

B.4.3. Swimmer Analytics Tracking Data Backup Options

If CSV is unsufficient for our purposes, we can also use XML or JSON for storing the data.

B.4.4. Wall and Lane Detection Data

The second method to wall and lane detection is to give all the pixels a class from the following list:

- Class one: pixel contains lane divider
- Class two: pixel contains wall edge
- Class three: empty class, contains no walls or edges

As the image is the union of classes one, two and three, it would be easiest to encode this data as the pixel locations of all pixels labelled as class one or class two for every frame that is used as data.So we could store in a JSON file the following:

- Frame Number
- Pixel location
- Pixel class

Additional ways of storing are as XML, or if those two fail then as a CSV file.

C. User Interface Design

This section outlines the proposed graphical user interface and application programming interface design for the above proposed systems.

C.1. Introduction/Background

Smart Swim Analytics aims to provide a product that requires little training and efficient in setup. Smart Swim involves two main UI interfaces: annotation and footage collection. We aim for them to be intuitive so that the annotation process can be done as quickly as possible, and the footage collection process can be run smoothly given a swimming pool is already a busy venue, especially during races.

C.2. User Analysis

This product will be used by the following user classes: Athletes, Sports Analysts, Swimmer Data Collection Specialists, and Footage Collectors.

C.2.1. Athletes and Coaches

While this product directly benefits the Athletes, they are the least important to satisfy in terms of product development as they are only interested in the resulting data produced by the system; they do not have to interact with the system.

C.2.2. Data Collection Specialist

The Swimmer Data Collection specialist mainly interacts with the Data collection system, thus they are moderately important. They are required to annotate the swimmers in a new competition venue. They must be able to create high quality data for the model training system to learn from. The Data Collection specialist must be trained no how to annotate footage of swimmers such that they are consistent with the already existing data.

C.2.3. Footage Collectors

The Footage Collectors must place the footage collection system in a location for which the swimmer models have been trained on, thus they are moderately important. They must make sure the Real-time system is working correctly and select the swimmer of interest for each race. Foremost, they must be trained on the setup of the tracking system.

Additionally they should know how to select the swimmer of interest and troubleshoot any issues that arise.

C.2.4. Sports Analysts

The Sports Analysts is the most important. They must orchestrate the users of the data collection system and the footage collectors. They also utilize the collected data from the data specialist to train or update swimmer models for the stroke rate estimation system and positioning systems. They may also have to interact with output data in order to get it in a format that is convenient for them. They must be trained on how to update models and install them into the new system when necessary. They must also be trained on how to extract data from the system and how to interpret it. Often they are or work very closely with swim coaches.

C.3. Technical Analysis

Analysis in the appendix takes into account the "Seven Elements of UI Interaction" (discoverability, feedback, conceptual models, affordances, signifiers, mappings, constraints) outlined in the ENSC 405W lectures and Don Norman's text (The Design of Everyday Things). Analysis encompasses both hardware interfaces and software interfaces.

C.3.1. Data Annotation

In terms of discoverability regarding the API the focus would be on ensuring there are simple visuals and as few options as possible to select from. While the ideal situation is to use keyboard shortcuts, there will be clear buttons to use in case the user is not comfortable with the shortcuts yet. Key information will be displayed (ex. frame number, lane number being worked on, action being performed) so that users understand what is being performed and to which swimmer.

In terms of feedback the focus is on ensuring users know when a change has occurred. For the API this entails having pop up messages whenever a file is being updated/created or ensuring it is clear that the action will update a file before the user confirms if they wish to continue. This would also include messages whenever something has gone wrong. Further, when the user is annotating, important information (ex. frame number, lane number, file name, etc) will be displayed so the user knows what they are annotating.

Signifiers in the API will be the buttons/icons and messages (to provide feedback).

In terms of conceptual models for the API the focus is on ensuring for each allowed operation presented on the main page (start annotations, start stroke counting, etc) the flow of actions and options presented clearly match and only focus on that specific operation. So after the user selects an operation (ex. start annotations) there will be a new view presented that will have all the related fields to the selected operation (ex. select video file). After all the fields have been filled a third and final view to allow the user to perform the operation (ex. screen with frame that is being annotated and options for annotating).

Affordability in terms of the API refers to what the user will be allowed to do and not do with for data collection. In this case, all input will be done via buttons or drawing (boxes), which helps guide users and limits their actions to what is allowed.

Mappings in terms of the API refers to what actions will result when users activate a button or interacting with input fields/messages. To ensure the mappings are clear, related fields will be grouped, everything will be sectioned (ex. after selecting to annotate a swimmer will the user be presented with options only related to this action).

Constraints regarding the API will mostly be semantic and logical. Semantic constraints referring to the meaning, this would be regarding the information present in the API, as well as usage of icons matching what users would expect that icon to refer to (though this could also be a cultural constraint). Logical constraints would be in the flow of actions when users click through buttons or are annotating. The groupings of the information/actions allowed and the way they are presented must be in a logical fashion.

C.3.2. Footage Collection

This will be similar to C.3.1 except this API is much simpler.

In terms of discoverability the API will focus on ensuring there are as few options and as simple options to choose from. There will not be many items to input and simple actions to perform.

In terms of feedback the API will focus on informing the user if the input was successfully recorded or not (error), and if the analysis is in progress or not via messages.

Signifiers in the API will be the buttons/icons and messages (to provide feedback).

In terms of affordability the API will be mostly input via buttons, except for the length of the pool (for which there will be checks on obviously incorrect values).

In terms of mappings there are mostly buttons. The flow of actions will be simple enough and grouped accordingly that everything will be mapped as a user would likely expect it to be.

In terms of constraints the API will need to handle, these will be semantic and logical. Semantic constraints here would be regarding appropriate use of icons (also possibly a cultural constraint). Logical constraints would be in the flow of actions being as a user likely would expect based on the information provided and order of operations.

In terms of conceptual models for the API the focus is on ensuring for each allowed operation presented on the main page (start analysis, update models, etc) the resulting flow of actions are specific to that operation. After the user selects an operation (ex. start analysis) there will be a new view with all the related fields to the operation (ex. select swimmer). After all required input is filled a message and, if appropriate, a status view will be present (ex. message that analysis has started, and a view with the option to cancel).

C.4. Engineering Standards

The product, Swimming Analytics, is to be implemented based on following engineering standards.

- **ISO/IEC TR 24028:2020 Information technology** — Artificial intelligence — Overview of trustworthiness in artificial intelligence [34]
- **ISO/IEC TR 24029:2021 Artificial Intelligence (AI)** — Assessment of the robustness of neural networks [35]
- **ISO/IEC 27001:2013 Information technology** — Security techniques — Information security management systems — Requirements [36]

C.4.1. Data Annotation

Specific to data annotation, we have the following standards:

- **ISO 9241-161:2016** - Ergonomics of human-system interaction — Part 161: Guidance on visual user-interface elements [37]

C.4.2. Footage collection

Specific to footage collection, we have the following standards:

- **ISO 9241-161:2016** - Ergonomics of human-system interaction — Part 161: Guidance on visual user-interface elements [37]

C.5. Analytical Usability Testing

The following two sections detail the testing to be performed by the designers to ensure the designs are intuitive.

C.5.1. Data Annotation

1. Upon launching the app the user is presented with a screen from which they can choose the main action to perform (annotate swimmer, count strokes, etc)
2. The user is informed when files cannot be found or successfully saved/updated
3. For each flow (ex. annotate swimmer) the user is presented with related options in the form of buttons, frames (to view the video), and other visual ques whose purpose are clear

C.5.2. Footage collection

1. Upon launching the app the user is presented with a screen from which they can choose the main action to perform (start analysis, upload model, etc)
2. The user is informed when files cannot be found for uploading a new model
3. The user is informed when the analysis is started or stopped (by user or errors)
4. For each flow (ex. start analysis) the user is presented with related options in the form of buttons, input text fields, and other visual ques whose purpose are clear

C.6. Empirical Usability Testing

C.6.1. Data Annotation

1. Action: User launches app. Expectation: User can see a variety of options on what actions to perform
2. Action: User selects option to annotate box. Expectation:
   - User is led through flow where they pick the video to annotate. If previous annotations exist those will be loaded automatically (assuming the file is present)
   - User is taken to screen where they can perform annotations via buttons. The screen includes displaying the current frame on which they may draw the boxes.
   - User is able to save changes
3. Action: User selects option to count strokes. Expectation:
   - User is led through flow where they pick the video to count strokes on. If previous annotations exist those will be loaded automatically (assuming the file is present)
   - User is taken to screen where they can perform annotations via buttons. The screen includes displaying the current frame being worked on, as well as the sinusoidal created thus far
   - User is able to save changes
4. Action: User selects option to create YOLO data. Expectation:
   - User can choose to create without JPEG, in which case the file with representations of the annotations in a format recognizable by YOLO will be created
   - User can choose to create with JPEG, in which case the file with representations of the annotations in a format recognizable by YOLO will be created as well as JPEG images of the annotations
5. Action: User selects option to create sub-videos. Expectation:
   - User is led through flow where they pick the video to create sub-videos form
   - If annotations exist, sub-videos will be successfully generated (of individual swimmers)
   - User is able to play the sub-videos successfully

C.6.2. Footage collection

C.6.3. Data Annotation

1. Action: User launches app. Expectation: User can see a variety of options on what actions to perform
2. Action: User selects option to upload model. Expectation:
   - User is led through flow where they pick the file
   - User is informed if model is uploaded successfully or not
   - User is asked to upload right file again if the model is not uploaded successfully
3. Action: User selects option to start analysis. Expectation:
   - User is led through flow where they input the length of the pool, select the specified swimmer to analyse, and where to save the output files
   - User is informed when the analysis starts
   - User is able to cancel the analysis at anytime if necessary
   - User is informed when the analysis is complete
   - User is informed if the analysis failed due to errors
   - user is able to back to the main page to start reanalysis if the analysis process failed
4. Action: User has completed a successful analysis. Expectation: User can view output files in the specified location.

C.7. Graphical Presentation

C.7.1. Data Annotation

In Figure C.1 is the current view of how swimmer annotations look from [9]. We will keep the same look, or if adjustments are needed we will aim for as close to this as possible. This is because the method provided is simple enough for users while being appropriate (in format) to work with the proposed system.

In Figure C.1 is the current view of how swimmer annotations look from [9]. While the current graph is not bad, we aim to improve upon it.

We plan to add various buttons to facilitate the flow of actions, which will minimize confusion and possibilities of error. In Figure C.3 is an example of the simplicity we will aim for.

C.7.2. Footage collection

We plan to have various buttons and simple plain text input fields to facilitate the flow of actions, which will minimize confusion and possibilities of error. In Figure C.4 is an example of the simplicity we will aim for.

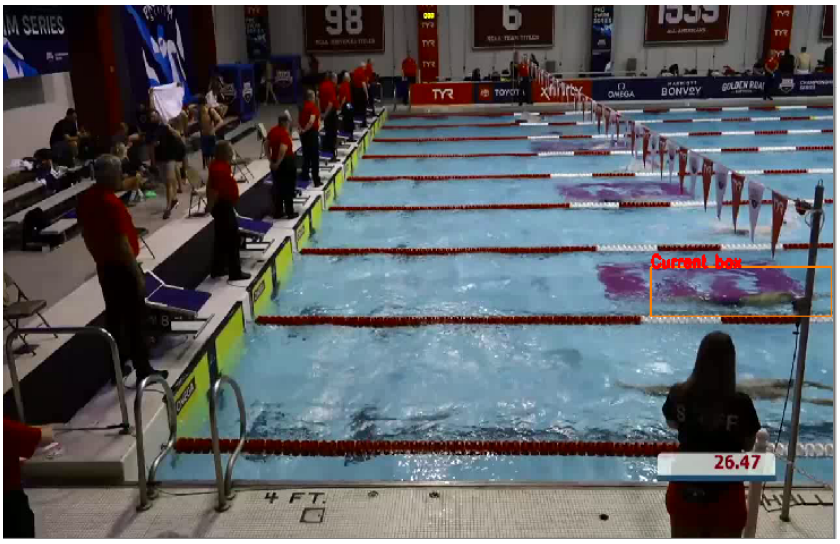


Figure C.1.: A screenshot of an annotation from use of [9]

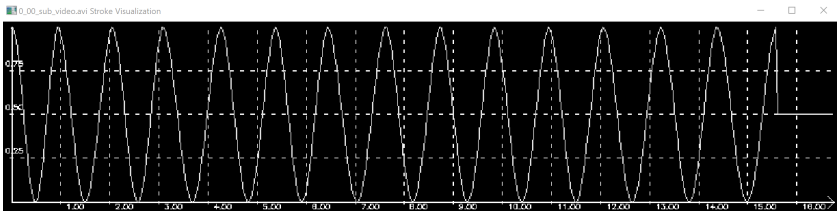


Figure C.2.: A screenshot of stroke annotations from use of [9]

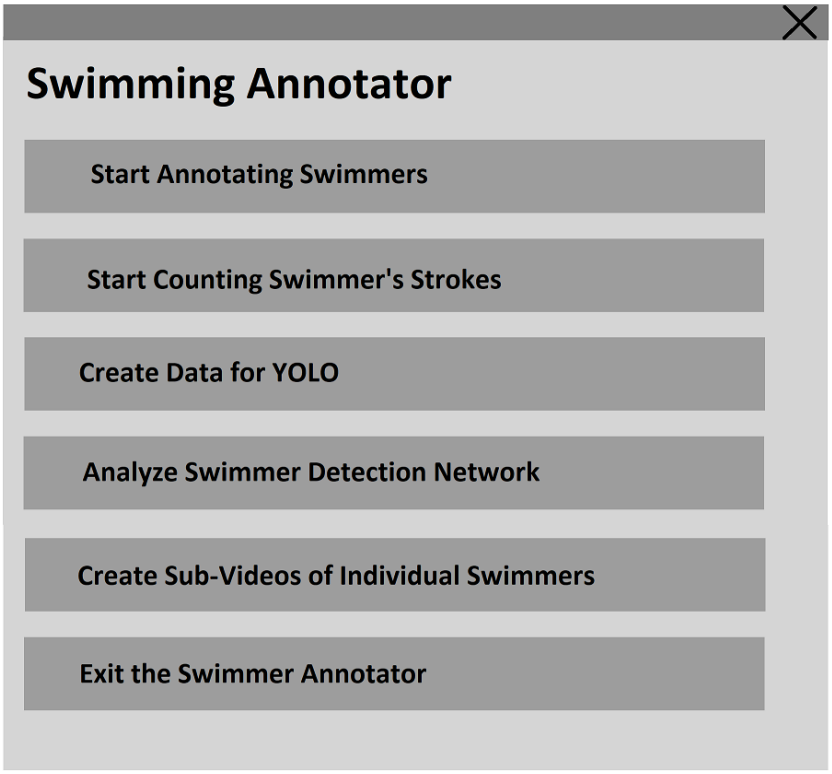


Figure C.3.: A rough design of a single view as planned updates to the API

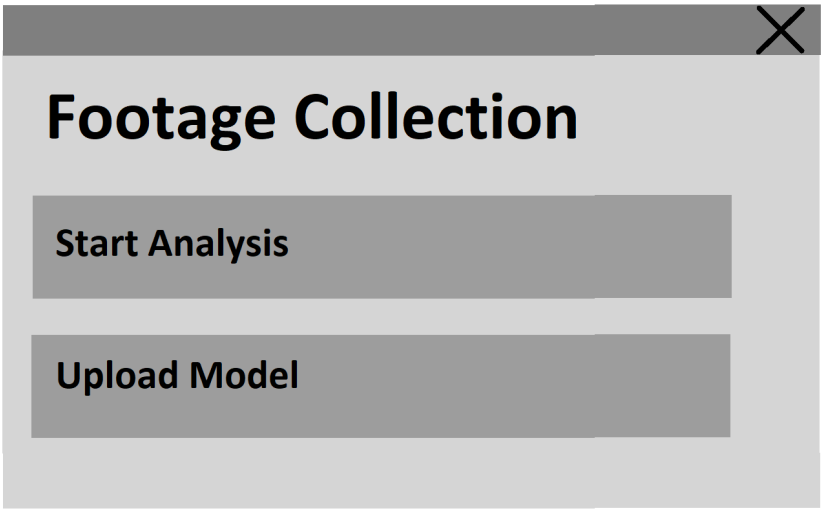


Figure C.4.: A rough design of a view in the Footage Collection API

C.8. Conclusion

Currently all user interfaces are in the alpha phase at this point. However the analysis in this section will be used to guide is construction in a way the allows users to most effectively use its functionality.

D. Background on Swimming

This background is based off the work in [1]. Understanding this subject matter will prove to be useful when considering the issues being solved in this work. Understanding how a race occurs, or how a swimmer swim, allows one to understand the challenges being faced. This section will continuously be referred to throughout this work when exploring challenges and solutions.

This section is broken down into the following sections.

- Section D.1: In which the Strokes Per Minute (SPM) and Distance Per Stroke (DPS) metrics are defined.
- Section D.2: In which a background on pools is given which is important for understanding the swimmer detection, tracking and positioning problems.
- Section D.3: In which a introduction to swimming races is given, which is important for swimmer tracking and positioning.
- Section D.4: In which the styles of swimming are more adequately defined, which is required for defining SPM.

For the purpose of this work, the aspect of swimming we are concerned with is speed and the strokes the swimmers take. The swimmer who can legally swim the specified race distance in the least time possible is the winner of the race. Swimming races are very controlled and thus a competition can take place where many races of the same event can occur in sequence. In order to compare two athletes, the time taken to complete the event is all that is required to rank the two swimmers. In fact swimming is so controlled that swimmers may be ranked across all occurring races at any given time and at any pool around the world and in some cases even when the pool is not the same length. This is in contrast to other sports like marathon events or ski jumping in which the ability of two athletes can only be quantified in a consistent manner when the athletes are at the same competition. Due to the very controlled environment swimming competitions require, many assumptions can be made in terms of pool size and swimmer position.

D.1. Swimmer Velocity Model

This section details the swimmer metrics that will be automatically obtained by the Smart Swim system. The following signals and values will be defined.

- L the length of the pool.

Figure D.1.: Common watch used by swim coaches to estimate swimmer SPM

- f the frame rate in frames per second (FPS) of the video footage of swimmers.
- n is a frame number in a video sequence sampled at a constant frame rate f.
- $x[n]$ is a signal which defines the position of the swimmer relative to the starting blocks at frame n, as such $L \geq d[n] \geq 0, \forall n$.
- $x'[n]$ is a signal which defines the velocity of the swimmer relative to the starting blocks at frame.
- $s[n]$ is a signal which defines the position of the top of a stroke (top of stroke is defined in section D.4). When a frame n contains a swimmer at the top of there stroke, the function takes a value of 1, if not, it takes a value of 0. In other words $s[n] \in \{0, 1\}$.
- $s'[n]$ which is the Strokes Per Minute of the swimmer as a function of n. This value stays the same between the top of each stroke the swimmer takes and changes after the end of a new stroke, that is $s'[n] \geq 0$ and only changes values when $s[n] = 1$.
- $d[n]$ is a signal which defines the distance per stroke (DPS) of a swimmer as a function of n. This value stays the same between the top of each stroke the swimmer takes and changes after the end of a new stroke, that is, $d[n] \geq 0$ and changes when $s[n] = 1$.

D.1.1. Strokes Per Minute

Strokes Per Minute or SPM is a measurement of the rate at which a swimmer takes strokes. The units are more generally in in **stroke cycles per minute**. This is due to the fact that the four swimmer strokes contain two strokes with so called "half strokes"; see Section D.4 for more details. When calculating SPM coaches often take a three stroke (cycle) average which they utilize a watch seen in Figure D.1. Beyond this, there seem to be no standard SPM collection method for all swim coaches. For the purposes of our work SPM is the output of the function $s'[n]$ divided by two when SPM is required for asymmetric strokes, for the symmetric strokes it is simply the output of $s'[n]$. Calculating $s'[n]$ is completed in the following manner. Firstly, the difference in

frames Δf_k for stroke k in frame n is found by taking the difference in l and m of frames such that $s[l] = 1$, $s[m] = 1$, $l < n < m$ and $\nexists p$ such that $s[p] = 1$ and $l < p < m$. The reciprocal of this value is taken and multiplied by 60 (as there is 60 seconds in a minute) and divided by the reciprocal of f, the frame rate of the footage in question. This is summarized in Equation D.1. If there exists n, such that there is no value l where $s[l] = 1$ and $l < n < m$ then $s'[n] = 0$. Conversely, if there exists n, such that there is no value m where $s[m] = 1$ and $l < n < m$ then $s'[n] = 0$. Note that this equation does not account for different strokes and thus if there are half strokes or not.

$$s'[n] = \frac{60}{f^{-1}\Delta f_k} \tag{D.1}$$

D.1.2. Distance Per Stroke

Distance Per Stroke or DPS is a measurement of the distance a swimmer moves given a single stroke, rather than stroke cycle (reference Section D.4 for more details on the difference between stroke and stroke cycle). DPS is generally measured in meters as most pools are measured in meters, however this is not a standard as it is common in the United States to have pools that are measured in Yards. **In this work, all DPS will be given in Meters**. For the purposes of our work DPS is the output of the function $d[n]$. $d[n]$ is calculated in a similar manner to SPM, values l and m for frame n are found such that $s[l] = 1$, $s[m] = 1$, $l < n < m$ and $\nexists p$ such that $s[p] = 1$ and $l < p < m$. The absolute difference between $x[l]$ and $x[m]$ is the value of $d[n]$. Again, if there exists n, such that there is no value l where $s[l] = 1$ and $l < n < m$ then $d[n] = 0$. Conversely, if there exists n, such that there is no value m where $s[m] = 1$ and $l < n < m$ then $d[n] = 0$. As such, this is summarized in Equation D.2.

$$d[n] = |x[m] - x[l]| \tag{D.2}$$

D.1.3. Swimmer Velocity

Swimmer velocity is the time rate of change of the position of a swimmer, also known as the derivative of their position. As such a simple manner for solving for a swimmers velocity is to differentiate, in time, the position of a swimmer. This can be approximated for each n by the well known difference equations D.3 - D.5. However, this method can become unstable if $x[n+1] - x[n]$ takes a value of zero due to a lac of precision of measurement. If this is the case, then more advance methods must be taken to find the velocity of a swimmer.

$$x'[n] = \frac{x[n+1] - x[n-1]}{2f^{-1}} \tag{D.3}$$

$$x'[n] = \frac{x[n+1] - x[n]}{f^{-1}} \tag{D.4}$$

$$x'[n] = \frac{x[n] - x[n-1]}{f^{-1}} \tag{D.5}$$

Another method to finding a swimmers velocity is to use the swimmer model [38]. This model says that a swimmer velocity is equal to SPM multiplied by distance per stroke to obtain meters per minute. To change to meters per second the result is divided by 60. This formula allows us to estimate $x'[n]$ by multiplying $d[n]$ by $s'[n]$ for each n and divide by 60, this is summarized by Equation D.6.

$$x'[n] = \frac{d[n]s'[n]}{60} \tag{D.6}$$

D.2. Pools

Swim competitions happen in many different pools and environments. The course of a pool refers to the distance the swimmers must travel in order to complete one pool length. For example, in a long course meter (LCM) pool, swimmers must complete fifty meters before they encounter a wall. Around the world, there are three main competition pool lengths in swimming, long course meters (LCM), short course meters (SCM) and short course yards (SCY). LCM pools are 50 meters long, SCM pools are 25 meters long and SCY pools are 25 yards long. Furthermore, pools can have different numbers of lanes. In each race every lane does not always contain a swimmers. More about pools can found in [39].

Different pools can have different numbers of lanes. Typically, there is an even number of lanes, ranging from six to ten. However, in some situations, competitions can be held in LCM pools but raced as SCY or SCM. This can result in up to a twenty-lane race. Causing even more confusion, some competitions have swimmers racing in one half of the pool, but warming up in the other half. An example would be having lanes 0 to 9 with swimmers racing while having lanes 10 through 19 open to swimmers who are not racing.

Lane ropes are the division between lanes. They run parallel to the pool edges and stop a swimmer from getting too close to another swimmer while also dampening waves. On the world stage there are very strict rules on how lane ropes may look and what color pattern they must adhere to. In general competition however, pools will use whatever lanes ropes are available and thus their detailed structures are of little use for swimmer detection. They are useful in a broad sense as they are fairly straight and define areas where swimmers can and can not be. In a training set, it would be desirable to have a wide variety of examples of pools with different styles of lane ropes.

D.3. Races

Swim racing has four strokes: butterfly (fly), backstroke (back), breaststroke (breast), and freestyle (free). These strokes all fall under the category of swimming. They are relatively easy to distinguish on a camera and can easily be identified by a human.

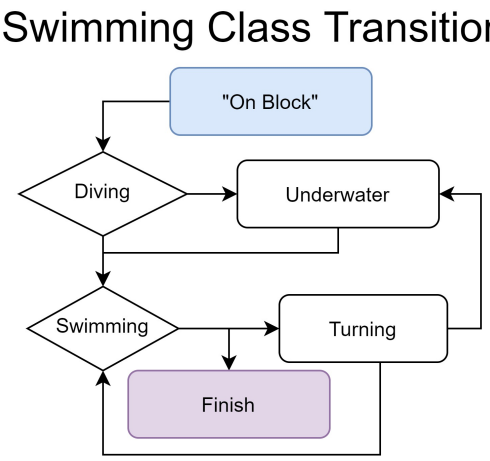


Figure D.2.: Order of swimming operations in a race

Each race starts with a dive. This is when the swimmer is positioned on an elevated starting block and then jumps through the air to enter the pool. This section is generally a very small fraction of a race as it only happens once and takes very little time. This action can be troublesome for automation as the swimmers position on the blocks (bent over) is very different than when they are in the water (stretched out) and due to high speed of the swimmer.

The next part of a race is the underwater sections. In the early 1980s it was shown that swimmers can be much faster underwater than on the surface, especially when pushing off the wall during a turn and after the dive. A considerable portion of a swimming race involves the swimmer underwater. This part of a race is problematic for automated swimming analytics. When swimmers are underwater they can be very difficult to see from an above-water camera. This is due do to the phenomenon of refraction, where light is bent because of the difference in densities between two substrates. This phenomenon can make a swimmer invisible for periods of time, creating what is known as occlusion. Another occluder of swimmers underwater is lane ropes – the plastic floating ropes separating swimmers in a race.

The next portion of the race is the turn action. The goal of a turn is to transition back to swimming when the swimmer reaches the end of a length. Turns can take many forms based on the event being preformed. After a turn, a swimmer usually ducks underwater and starts a new underwater section. The turn is problematic as the white water produced by the swimmers causes occlusions of the boundaries of the swimmer. This can make it difficult to localize the swimmer's position.

Between the underwater portion and turning portions of the race is the swimming. As mentioned before, there are four strokes that can occur, but only once per length. Possibly one of the easiest parts of the race to identify a swimmer is when they are swimming; it is also the most prevalent. There are a few reasons why a swimmer would

be hard to recognize in this portion, but it is mainly due to occlusions. These are caused by people or the sets of ropes hanging over the pool designed to warn a swimmer of an approaching wall.

Finally, a race ends with a finish. This simple action transitions the swimmer from swimming to finish and is easy to identify. Once the swimmer is finished they generally stay in the same place until the rest of the swimmers finish, marking the end of a race.

In general, the order of operation in a races is as follows: on-blocks, dive, underwater, swim, turn, underwater, swim and finish. However, there are varying numbers of swim, turn, underwater and finishes depending on the race category (event) – see Figure D.2 for a complete state transition diagram.

In addition to the technicalities of swimming, events can have different distances and styles. Such distances range from lengths of 50 meters to lengths of 1,500 meters. Some styles have swimmers preform all four strokes in one race, such as the individual medley (IM), or only one stroke at a time. Swimming has 18 different styles of individual events and another 5 team events known as relays. All the different positions, transitions, actions, pool lengths, pool environments and pool lane numbers for each event must be taken into account for when creating a fully functioning automated swimming analytics system.

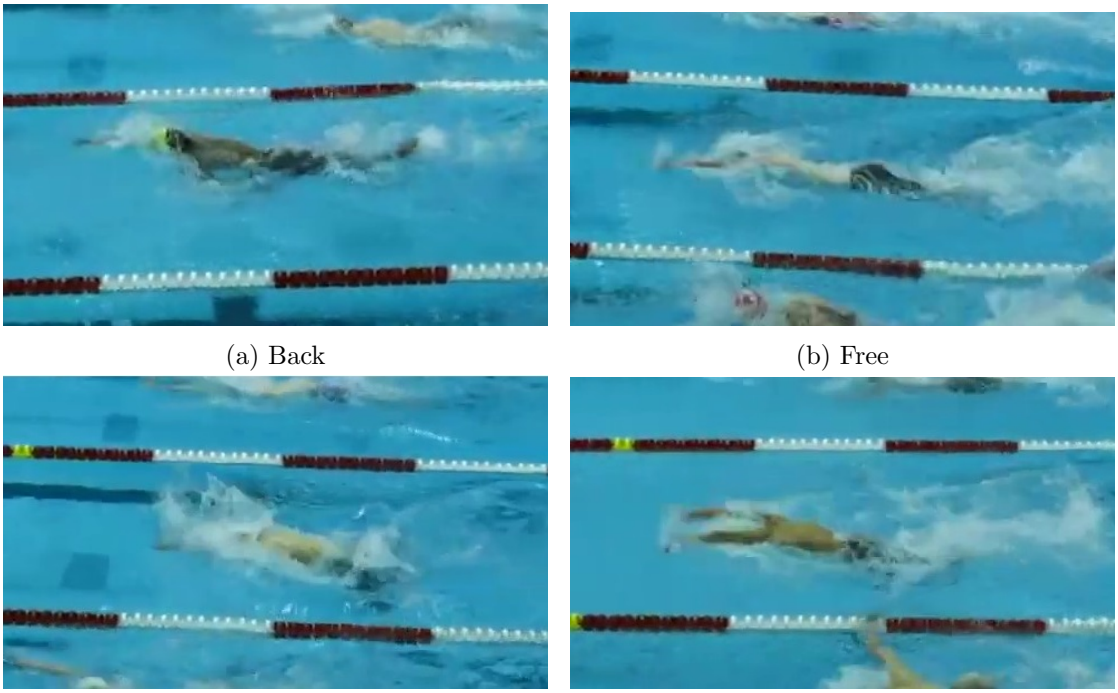
D.4. Stroke Styles Definitions

As mentioned in Section D.3 there are four strokes: fly, back, breast, and free. Each of the four strokes produce propulsion, which moves the swimmer though the water. The propulsion is caused by both the arms and the legs of the swimmer. While the legs produce considerable power and increase the buoyancy of a swimmer in the water, swimming analytics, at this point, is mainly concerned with the arms of a swimmer. This is because arms are more straightforward to correlate to faster swimming and also because the arms are generally responsible for approximately 80% of a strokes propulsion. Furthermore, for each of the four strokes, a stroke is counted as a stroke cycle. Because each stroke has its own definition of a stroke cycle, we shall briefly go over stroke cycle definitions.

D.4.1. Stroke Cycle Definitions

In terms of stroke cycles, the four strokes can be split into two categories for the purposes of stroke recognition. The first category is asymmetric strokes, seen in Figure D.3, which are the back D.3a and free D.3b styles of swimming. They are denoted as asymmetric as the stroke is not symmetric through the human sagittal plane. In terms of a stroke cycle, one arm pulls past the body while the other is recovering above the water allowing for strokes to be taken by each arm individually in quick succession. Thus, stroke cycles for free and back are completed when one of each arm is pulled and recovered past the body.

The second category of strokes is symmetric, seen in Figure D.3, which consists of the fly D.3d and breast D.3c styles of swimming. This is because these strokes are symmetric



(a) Back (b) Free

(c) Breast (d) Fly

Figure D.3.: An example of the top of each of the four strokes

across the human sagittal plane. Stroke cycles are more straightforward in this category because in each stroke, both arms are pulled past the body and recovered in unison. So, stroke cycles for fly and breast are counted for each stroke pull and recovery.

D.4.2. Top of Stroke

It was noted that an when $s[n] = 1$ the "top of a stroke" was occurring. In this section the definition of the "top of a stroke" for each of the four strokes will be given. Unfortunately each stroke has a unique definition for its top. Figure D.3 gives pictorial examples for each of the strokes "top of a stroke". For the asymmetrical strokes, free D.3b and back D.3a, in order to remove confusion the "top of a stroke" occurs twice per stroke cycle. This happens when either the left or right hand enters the water from the stroke's recovery. For the symmetrical strokes, Fly D.3d and breast D.3c. The top of the stroke is counted when both hands enter the water in the stroke recovery. As such, for fly and back, the "top of the stroke" occurs only once per cycle.