An Artificial Intelligence forecast system for strategic positioning in a professional competitive E-Sport.

by

Joaquim Augusto Silva Vieira

Dissertation submitted in fulfilment of the requirements for the degree

MASTER of ENGINEERING in ELECTRICAL ENGINEERING (M_ENGE)

In the Department of Electrical, Electronic and Computer Systems Engineering

Faculty of Engineering and Information Technology

Central University of Technology, Free State

Supervisor: Dr N.J Luwes

2021

Declaration

I, Joaquim Augusto Silva Vieira (Identification Number _____, Student Number _____) do hereby declare that the research conducted and submitted to the Central University of Technology of Bloemfontein, for the degree of Master of Engineering: Electrical Engineering, is my own independent work; and complies with the Code of Academic Integrity, as well as other relevant policies, procedures, rules and regulations of the Central University of Technology; and has not been submitted before by any other person in fulfilment and or partial fulfilment of the requirements for the attainment of any qualification or publication.

Signed: _  Date: __12 January 2021_____

Acknowledgements

The author would like to acknowledge the following individuals and institute, without whom, the completion of this dissertation would not have been possible:

- God – for the strength and perseverance to complete my dissertation.
- Dr N. Luwes – for an endless source of information and recommendations in the field of Machine Vision and Neural Networks.
- The Central University of Technology – for providing me with the opportunity and resources to complete a life goal.

Abstract

The gaming industry is one of the largest contributors in the entertainment industry, boasting an estimated US$152.2 billion of revenue for 2018, with electronic sports (E-sports) being one of the main contributors to this astronomical revenue. One such E-sport is Counter-Strike Global Offensive (CSGO). The purpose of the study is to develop an automatic strategy predictor (ASP) for Counter-Strike Global Offensive (CSGO) that will be able to determine a specific team's strategy accurately. The ASP achieved this by developing an MVS to analyse and capture historical matches and generate a player position stencil (PPS). This PPS is then processed further to obtain the necessary information to train an artificial neural network (ANN). The development of the ASP consists of intercommunicating multifaceted software components.

The first part is an image-processing section that captures the in-game mini-map within CSGO and by doing so is able to capture the player positions at a specific point in time for every playable round. This image-processing part consists of two components. The first is the player position stencil producer (PPSP); the second is the stencil processor (SP). The PPSP captures the in-game mini-map using the image acquisition techniques and the SP processes the stencils captured by the PPSP and produces the player position information (PPI). An information processing system (IPS) is developed in order to process the PPI. The ISP consists of three components. The first of these was an information concatenator (IC). The IC would concatenate the PPI, captured by the SP, with the additional information obtained from the CSGO Demo Manager. The CSGO Demo Manager is a software package that produces information about every round within a match. The additional information obtained from the CSGO Demo Manager contains information such as the team's economy, their utility, whether or not the in game device was planted or not and if the round was won or not. This is concatenated the PPI with the additional information, a comprehensive

understanding of the round can be generated. Once concatenated by the IC, the information is called the overall round strategy information (ORSI). The second and third software components form part of the information pre-processor (IPp). The purpose of the first IPp software component is to retrieve the ORSI from memory and perform pre-processing on the ORSI. The purpose of the pre-processing of the ORSI is to ensure that the ORSI was in the correct format for the training of an artificial neural network (ANN). The second IPp software component separates and arranges all the information by rounds. The information of each round is separated and stored as the information file for training and is called the overall round strategy information 2 (ORSI2) file. Once each round has an ORSI2, the fourth milestone is met. Each round's ORSI2 file is used to train an ANN. The ANN of each round is called the automatic strategy predictor (ASP). Once all the ASPs have been trained, five unseen matches are processed by the MVS, the IPS and the IPp in order to obtain five unseen ORSI2 files. These ORSI2 files are used as inputs to determine the accuracy of the ASP when predicting unseen matches. The outputs of each rounds ASP are compared to the actual player positions retrospectively.

The results show that the suite of software developed in the MVS was able to capture the in-game mini-map, as well as process the PPS that was captured. The information obtained by processing the PPS was successfully processed into a format that can be used for training an ANN. The training of the ASPs was successful, as the average R values obtained for each round's ASP, were greater than 0.98. With all the APS's trained, five unseen matches were processed once more in order to obtain five new ORSI2s. The new ORSI2s were used to determine the accuracy of each round's ASP when predicting the strategy used by the team within that specific round. The second round's ASP was able to achieve an accuracy of 67%; this is expected, as there is only one prior round of information for the ASP to make an accurate prediction. The later ASPs all achieved accuracies well

over 95% prediction success. The trained ASP delivered instant predictions once an input was given to the ASP and could be utilised by a professional E-sports team to study an opposing team, or by E-sports analysts and discussion panels.

Table of Contents

List of Tables

List of Equations

List of Figures

Chapter 1

Introduction and Hypothesis

1.1 Background to the study

Gaming as a sport and artificial intelligence (AI) has gone hand in hand for many years. During the mid-1990s, IBM Research developed an AI named Deep Blue. The researchers developed Deep Blue in order to beat the best chess player in the world at the time, Garry Kasparov. Deep Blue was able to defeat Kasparov in a 6-game match in 1997 [1]. A few years earlier, in 1972, the first ever video game competition was held at Stanford University's Artificial Intelligence lab in which students competed in the game Spacewar. This was followed by the first competitive gaming tournament held by Atari where participants had a chance to become world champions in Space Invaders, and have grown exponentially since [2][3].

Currently, one of the most popular E-sports is Counter-Strike Global Offensive (CSGO) [4]. CSGO is a multiplayer first-person shooter game developed by Valve and was released in August 2012. In 2017 CSGO set a world record at the e-league major tournament, with a prize pool of US$1 million for the winning team. The tournament was streamed on the internet, with the grand final televised worldwide. It is believed that a possible 1.2 million viewers watched via the internet and more than a million on television, totalling of 2.2 million concurrent viewers [5]. This proves to be comparable to other major sports such as cricket and soccer. The most widely viewed cricket match was recorded in 2015 at the Melbourne Cricket Club where they hosted a crowd of 121 696 people. In 2014 the FIFA World Cup had a televised audience of 34 650 000 people [6]. E-sports is viewed on various platforms:, Twitch.tv, YouTube, FaceBook live and even DSTV, to name but a few.

The re-viewing of opponents' previous matches, typically led by a team's coach, has been a common practice among sports teams. Re-viewing previous matches is also common in the E-sports

realm. In doing so, one gains valuable knowledge of the opposing team, such as bad habits that they may demonstrate, how they respond to aggressive pressure, et cetera. With this information one can exploit these points to gain an advantage on the opponents [7]. Re-viewing matches of the opposing team is also common practice in CSGO – the problem, however, being that this process is time consuming as the duration of each match is at least an hour. By having a machine vision system (MVS), the matches can be reviewed automatically.

An MVS would significantly reduce the time it would take to obtain sufficient information to study an opposing team. The output of an MVS could be fed into a Neural Net in order to automatically study the opposing teams' bad habits and style of play, thus being able to predict what strategy they may use.

1.2 Aim and objectives

In the professional E-sport CSGO, it is important to know the strategy of an opposing team. This could be used by professional teams when preparing for a match as well as for discussion panels and E-sports analysts. The players' positions are what dictate the strategy to be followed in a round. The position the players take in can be studied in order to formulate a counter-strategy to combat the positions of the enemy. The aim, therefore is to develop an automatic strategy predictor (ASP), a number of software components of a machine vision system (MVS), as well as the information processing system (IPS), with the resultant information being fed into an ANN. In order to achieve this, the following objectives were devised:

- undertaking a comprehensive review of the literature that will underpin this study;
- capturing and digitising the in-game mini-map with the use of real-time image acquisition with the use of an MVS;

- processing the captured images of the in-game mini-map in order to obtain the necessary information through image processing with the use of an MVS;

- processing the information that is obtained from the MVS using an IPS and

- developing and training the Artificial Neural Net (ANN), using the information obtained from the IPS, and evaluating the accuracy of the ANN.

1.3 Hypothesis

An automatic strategy predicator (ASP), using a number of software components of a machine vision system (MVS) as well as the information processing system (IPS) outputs being fed into an Artificial Neural Network (ANN), is able to predict the player positions of a professional CSGO team and thus determine the strategy that will be employed within the specified round.

1.4 The layout of the dissertation

This thesis comprises six chapters covering the following topics:

Chapter 1: Introduction

Chapter 1 contains a synopsis, problem statement, motivation, as well as the aim and objectives of the study.

Chapter 2: A literature review

Chapter 2 contains an extensive review of various literature of the gaming industry and electronic sports (E-sports), as well as the development and implementation of Artificial Neural Networks (ANNs) for the predicting of sporting outcomes. Furthermore, machine vision systems and the manner in which these technologies are implemented, are also reviewed in this chapter.

Chapter 3: Study design

In Chapter 3 presents the study design and a description of each phase within the study, as well as the milestones that need to be met throughout the study.

Chapter 4: Development of the Software Suite

In Chapter 4 the development of the suite of software needed for the study is communicated. It also includes three milestones needed to be achieved. These milestones were achieved through the development of software for the software suite:

- **Milestone 2:** Image Processing: ***Player position stencil producer (PPSP), Stencil processor (SP)***
- **Milestone 3:** Information Concatenation**: *Information concatenator***
- **Milestone 4:** Information Pre-processing: ***Information pre-processor***

Within this chapter a brief literature review is given on the various aspects that needed to be taken into account when developing each software within the suite. Furthermore, an extensive explanation of the development process is provided along with the results obtained from the development of each software.

Chapter 5: Development and Evaluation of the Automatic Strategy Predictor

Chapter 5 presents the development and the evaluation of each round's automatic strategy predictor (ASP). This includes completion of the following:

- **Milestone 5:** Development and training of each round's ANN
- **Phase 3:** Evaluation of the each round's ASP

Chapter 6: Evaluation of the each round's ASP

Chapter 7: Discussion and conclusions

In this concluding chapter the key findings of the study are presented. It is shown how these findings are integrated into existing knowledge, and it also contains a discussion of the challenges and further development of the ASP.

Chapter 2

Literature Review

2.1 Introduction

Team strategising is an essential aspect of competitive team sports that require analysis to ensure the best outcome when competing. Before competing, teams select the best players from a pool of available players in order to have the best talent representing the team. The process of selecting players is done by assessing each player's set of skills. Once the best players are chosen by a coach or talent scout, the player attributes are often the determining factor in the layout of the team, and the beginning of the strategising process [8]. Powerful big data analytic algorithms in conjunction with high-definition cameras are being used in the scouting process by major sports organisation such as Arsenal Football Club. An example of such a system is shown in Figure 1.

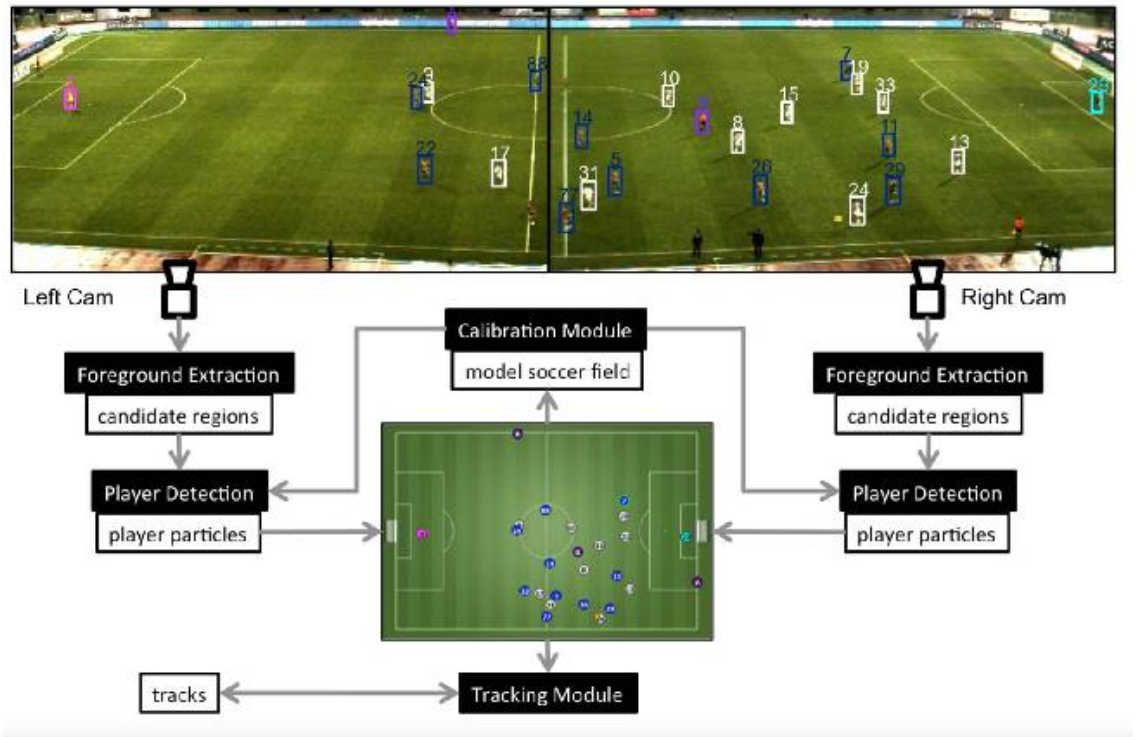


Figure 1 *An overview of the Sentioscope system using two cameras for real-time player tracking*

Furthermore, these systems are used by coaches and aid them in predicting the performance of players as well as the outcomes of games to gain a competitive edge over competitors in the English Premier League (EPL) [8]. These analytic algorithms analyse and predict the outcome of both a player or a match by reviewing historical data. The system reviews both videos and images of a player's activity and various other data points in order to make predictions [9]. Team organisations are planning similar systems for implementation in professional tennis as well as Formula 1.

2.1.1 The gaming industry

Video games have become a multi-million dollar part of the entertainment industry and have grown rapidly over recent years. From the early years of gaming, video games have captivated the minds and interests of millions of people with their stories and complex narratives, and more recently with the dawn of online and professional gaming, in E-sports.

2.1.2 History of gaming

A video game, by definition, is an electronic game that involves interaction with a user interface in order to generate visual feedback on two or more three-dimensional video display devices such as a screen or a virtual reality headset. The first game, called "Spacewar!", was developed in 1962 at the Massachusetts Institute of Technology (MIT) by Steve Russel in collaboration with Mart Graetz and others [10]. "Spacewar!" was played on the DEC PDP-1 microcomputer and featured two players controlling two spaceships –one called "the wedge" and the other "the needle" – that engaged in a dogfight while manoeuvring in the gravity well of a star.

While "Spacewar!" was the first-ever video game, the first video game to be *sold commercially,* was "Computer Space" in the early 1970s. It was developed and created by Ted Dabney and Nolan Bushnell. "Computer Space" used a black-and-white television set for a display; the computer system made use of 74 series TTL chips [11]. Later on similar systems were created, such as the

"Brown Box" by Ralf Baer [12], as well as gaming giant Atari's "Pong"[13]. From that point on, the video game industry snowballed with the advent of coin-fed arcade machines and games, such as Taito's Space Invaders becoming classics overnight.

"Spacewar!" being the first-ever video game also acquired the title as the first-ever game to host a competition. The first video game competition was held in 1972 by staff at Stanford University's Artificial Intelligence lab where students competed in the game "Spacewar!" [2]. Although "Spacewar!" held the title of the first video game competition, the first-ever *competitive* gaming tournament was held by Atari. Participants of the Atari tournament had a chance to become world champions in Space Invaders [3].

2.1.3 Online gaming

The 1970s were not only crucial for the development of the gaming industry, but also marked the advent of the internet. In the early 1960s, Joseph Licklider, a computer scientist at MIT, popularised the idea of an "intergalactic network" of computers. Later scientists developed a new technology called "packet switching". Packet switching was a method of transmitting electronic data more efficiently and would become the building blocks of what we today call the internet.

A simple example of packet switching over a network is shown in Figure 2 [14].

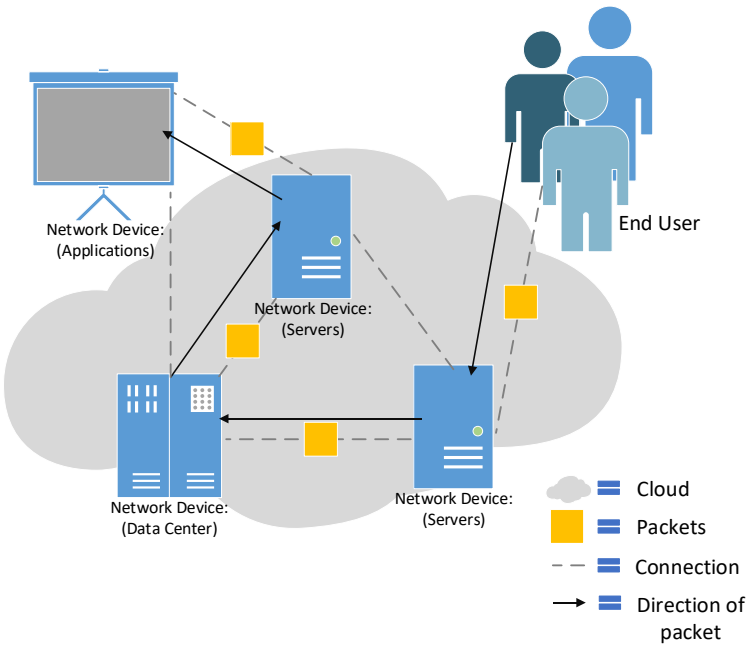


Figure 2 *Illustration of a packet-switching network*

On 29 October 1969 the first-ever message was sent from one computer to another over a network, and throughout the 1970s scientists Vinton Cerf and Robert Khan developed the Transmission Control and Internet Protocols (TCP and IP), which became the communication standards and the way in which data would be transmitted between networks [15].

With the development of video games and the internet sharing the same birth date, so to say, it was inevitable that their paths would cross. And they did for the first time with the first-ever online video game called “Island of Kesmai”. Island of Kesmai was an online role-playing game and was developed by computer science classmates Kelton Flinn and John Taylor in 1985 [16].

2.1.4 Electronic sports and electronic sports tournaments

Since the success of Island of Kesmai, online video gaming and the gaming industry as a whole have exploded as one of the most significant contributors in the entertainment industry. Forbes journalists believe that it will reach over $300 billion by 2025 [17]. The success of the video game industry is thanks to major gaming competitions being held around the world every year for games such as, *inter alia*, Fortnite, Counter-Strike Global Offensive (CSGO), DOTA 2, Call Of Duty. In 2017 CSGO held the World Electronic Sports Games with a prize pool of US$1.5 million. In 2019 Fortnite held a Fortnite World Cup, with a grand prize of US$30 million. These tournaments were streamed live on various platforms to millions of viewers.

The video games played at these tournament,s were labelled as electronic sports (E-sports). An E-sport is a multiplayer video game that is played competitively for spectators, and is typically played by professional gamers. The professional gamers in these competing teams are often referred to as electronic athletes (e-athletes) or cyber-athletes [18].

2.2 Counter-Strike Global Offensive

As mentioned in the section above, Counter Strike Global Offensive (CSGO) has become one of the largest E-sports, and the focus of this study is the application of an Artificial Neural Network (ANN) as a predictor of the strategic positioning of a professional CSGO team. The section below will discuss the history as well as the dominant concepts that dictate strategy in the E-sport, CSGO.

2.2.1 History

Counter-Strike was released to the public on 19 June 1999, but was first developed by Minh Le, James Ohlen, Jess Cliffe and Richard Gray as a mod for the popular game Half-Life. The developers of Counter-Strike refined the classic first-person shooter (FPS) recipe and created a dedicated

community of players that has followed the game for over a decade. Later Counter-Strike was acquired by the parent company of Half-Life, Valve, and given the name Counter-Strike. There have been several different Counter-Strike games over the years, with the latest iteration being Counter-Strike Global Offensive (CSGO), released in August 2012.

2.2.2 Concept of CSGO

CSGO is a fast-paced FPS where two factions, five players on each side, battle each other on a round-by-round basis, with one side being terrorist (T) and the other side counter-terrorist (CT). The objective of the Ts is to plant a destructive device in one of two plant sites, while the CTs need to either prevent the planting of the device or remove said device after it has been planted by the T-side. A round can be won by eliminating the entire enemy team or by completing the objective.

A competitive CSGO match has 30 rounds within regulation. A competitive CSGO match has two halves. A half is reached when 15 rounds have been played. After the 15 rounds, the teams will switch sides, much like conventional sports teams switch sides at half-time. Unlike conventional sports, the winner of a match is not decided at the end of a specified period of time. Instead, a winner is determined by the first team to have won sixteen rounds.

Each round starts with all five players of the team starting in their respective spawn spots on the map. Each map within CSGO has similar layouts.

The layout of these maps have a three-lane design (Figure 3): a left lane, a middle lane and a right lane, as well as two plant sites, Site A and Site B.

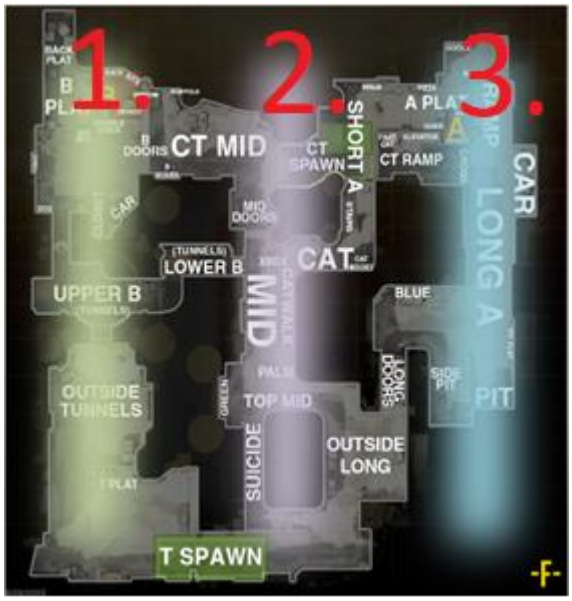


Figure 3 *A top-down image of the map Dust 2 found within Counter-Strike Global Offensive*

At the core of this competitive mode, is the game's economy. The economic factor of CSGO is what sets it apart from other first-person shooters E-sports. At the start of each round each player needs to purchase a weapon, armour and utilities (utilities being smoke grenades, flashbang grenades or explosive grenades). A player can earn money by eliminating enemy players or by winning the round.

Each weapon that can be purchased has differing economic returns for eliminating players. A player can earn more money by using a lesser weapon such as a pistol, as compared to a sniper rifle, with the trade-off being risk over reward. A summarised spreadsheet of the manners in which players can earn money can be seen in Figure 4 [19]. The economic structure of CSGO is often the basis of a team's strategy when preparing for a match.

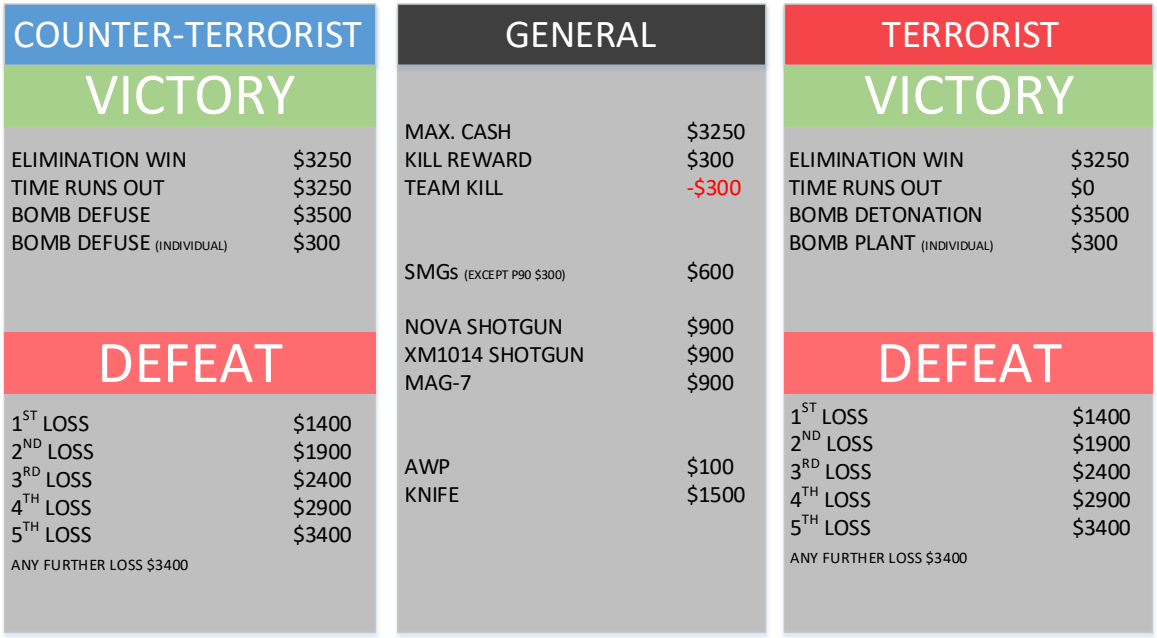


Figure 4 *Breakdown of Counter-Strike Global Offensive's economy*

2.3 Machine vision systems

An artificial neural network (ANN) developed for the purpose of predicating the strategy of a professional CSGO, will implement a machine vision system (MVS). This section will discuss the history of machine visions systems as well as various concepts within machine vision, such as image acquisition, resolution and frame rate. Furthermore, the image processing aspects of an MVS are discussed, which includes, image segmentation, region of interest, colour thresholding and feature extraction methods.

2.3.1 History of machine vision systems

A group of researchers at the Hitachi Central Research Laboratory coined the term "machine vision" in the early 1960s. Factory automation was the driving force for the development of these machine vision systems that made use of various image acquisition and image processing techniques [20]. The basis of all machine vision systems is to extract information from images that are captured in an automated manner. The information captured by the system, is then analysed to provide information. The process of analysing the captured images is automated as well [21][22].

The application of a machine vision system is unique to the problem trying to be solved. As there are countless applications for machine vision systems, the design and implementation techniques of the machine vision systems differ significantly from one application to another. Although the applications, designs and implementation techniques may differ, each machine vision system has common characteristics. All machine vision systems have three main components, viz image acquisition, image processing and information analysis. A flow diagram of a typical machine vision system is shown in Figure 5 [23].

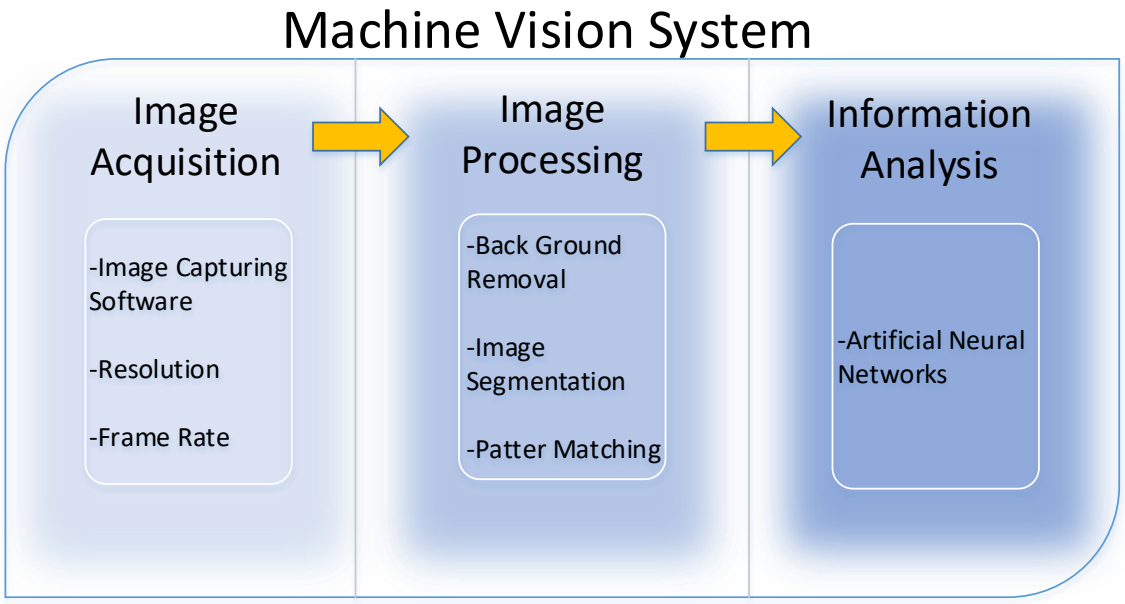


Figure 5 *Logical diagram of a typical machine vision system*

2.3.2 Image acquisition

Image acquisition is the beginning subsystem of any machine vision system. The image acquisition subsystem of a machine vision system depends on the goals set out for the machine vision system. Furthermore, the selection and design of the image acquisition subsystem entail the selection of the correct light source, sensors and optics concerning the object being imaged by the MVS. Often overlooked, are the specifications of the image acquisition subsystem, such as the rate at which the image acquisition subsystem is acquiring the image, the frame rate and the resolution [24][25][26].

2.4.3 Resolution

The resolution at which an image is acquired by the image acquisition subsystem is an aspect that is often overlooked when capturing images. When developing an image acquisition subsystem, a higher resolution is always desired, as more detail and more information can be obtained from a high-resolution image. Low-resolution images often seem “pixilated” and thus could result in information being lost. Figure 6 is an illustration of a direct comparison between a high-resolution image (A) and a low-resolution image (B). The most significant pitfall of having high-resolution imaging is that more memory space will be taken up. Furthermore, as the resolution of the image increases, so does the processing power of the machine needed to process the image after the acquisition [27].

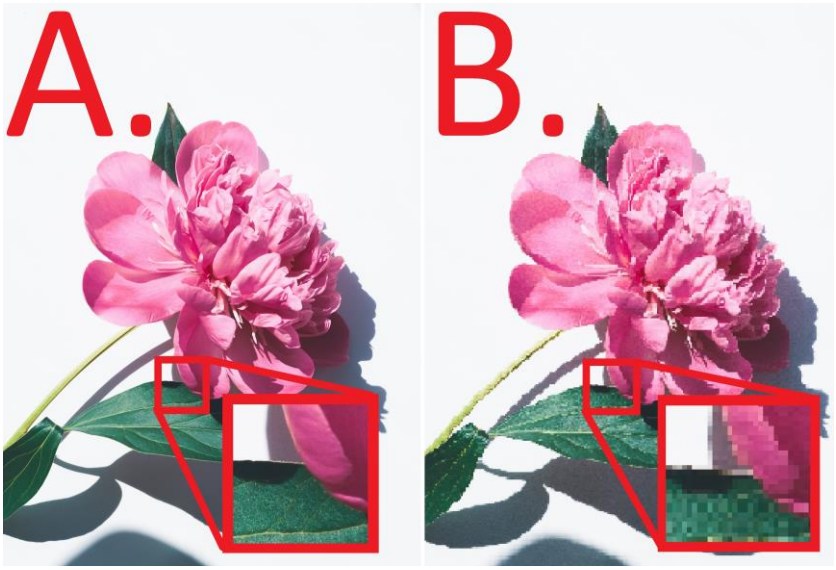


Figure 6 *Direct comparison between a high-resolution image, A. and a low-resolution image, B.*

2.4.4 Frame rate

The rate at which a video is captured, is known as the frame rate. The frame rate is determined by counting the number of frames captured per second, and is measured in frames per second (FPS). Figure 7 is an illustration of FPS. Image acquisition subsystems with higher frame rates can capture

more information per second, thus making high frame rates preferable. It is highly beneficial to have a high frame rate, but the down side is thata system that can store large amounts of data is needed. The videos captured, often take up hundreds of gigabytes of storage. Furthermore, the devices that capture the video at high frame rates are also under much computational strain, as many of these devices are overclocked, i.e. a process of boosting the performance of a processor in order to capture video at high frame rates [28][29].

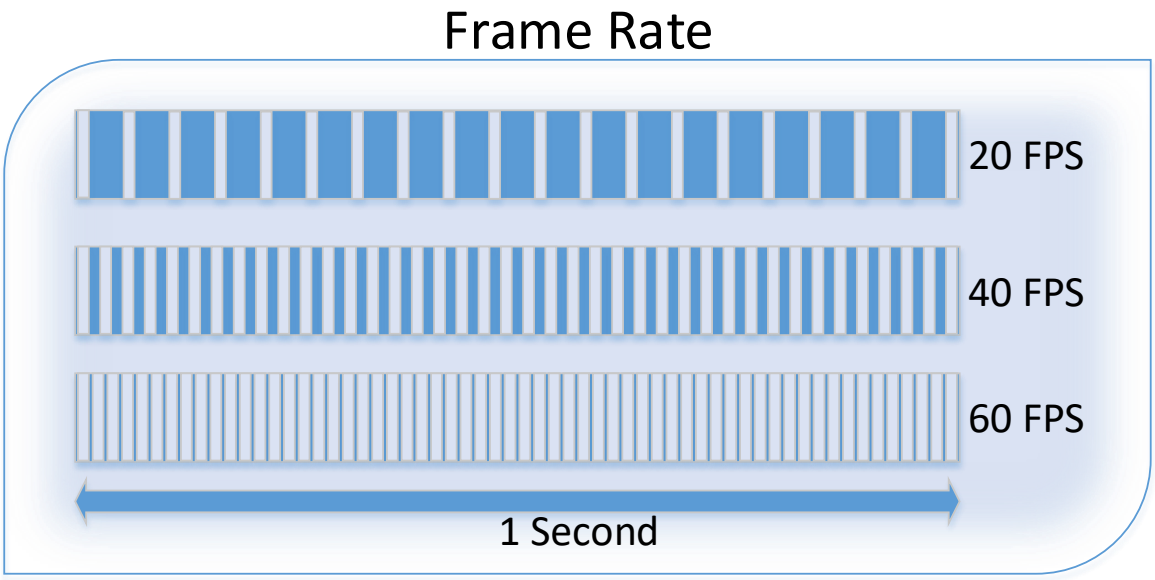


Figure 7 *Illustration of frame rate per second*

2.5 Image processing

Image processing is a method of performing and applying various algorithms and operations to an image in order to enhance or extract useful information from it. The images acquired by the image acquisition subsystem can be enhanced and manipulated using various image processing algorithms to extract useful information and features [30][31]. Image processing algorithms include algorithms for segmentation through thresholding or the application of a region of interest, and feature extraction by applying filtering algorithms as well as morphological algorithms.

2.5.1 Image segmentation

Segmentation of the captured image minimises the performance impact on the machine performing various image-processing algorithms. Segmentation of the image can be done with the use of a region of interest (ROI) and colour thresholding. Thresholding has the added benefit of being a process for extracting features from images.

2.5.2 Region of interest (ROI)

A region of interest (ROI) can be defined as removing unnecessary parts of an image that need not be processed by the image processing software [32]. These extracted, application-specific features are then easier to manipulate and be analysed by the software to generate useful data. The useful data can be used by image processing techniques to manipulate or improve images [33].

2.5.3 Colour thresholding

Colour thresholding is an image processing algorithm that is applied to an image in order to isolate a specific colour from an image. Colour thresholding is applied to an image by designating separate thresholds for each of the Red, Green and Blue (RGB) components of the image. Once the colour thresholding algorithm has processed an image, it converts the image into a binary image. These binary images are structured as a mask consisting of 1s and 0s as its pixels [34]. An illustration of colour thresholding by masking is shown in Figure 8.

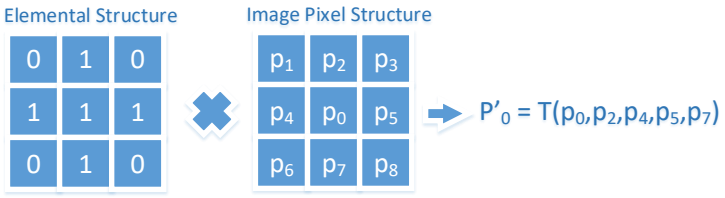


Figure 8 *Image binary masking*

Colour has often been used in machine vision for various tasks such as segmentation and recognition. Making use of colour for segmentation has proven to be superior to the use of geometric information for segmentation in tasks such as visual robustness under partial conditions, rotation in-depth, scale changes and rotation changes. Furthermore, colour thresholding as a form of image segmentation has also proven to utilise more efficient algorithms yielding real-time performance on standard hardware [35].

2.5.4 Feature extraction

Feature extraction is the process of removing unwanted features from the captured image to reduce the features found in the image, so that only application-specific features remain. Isolating important features and discarding unwanted features simplify the process of extracting and generating useful information from the image. Feature extraction can be performed by colour thresholding erosion, low-pass filtering and the implementation of convex hull morphological functions [30].

2.5.4.1 Erosion

Erosion eliminates pixels isolated in the background and erodes the contour of particles according to the template defined by the structuring element. For a given pixel P0, the structuring element is centred on P0. The pixels masked by a coefficient of the structuring element equal to 1 are then referred to as Pi. If the value of one pixel Pi is equal to 0, then P0 is set to 0, else P0 is set to 1. If AND(Pi) = 1, then P0 = 1, else P0 = 0. The figure below illustrates the effects of erosion and dilation [36].

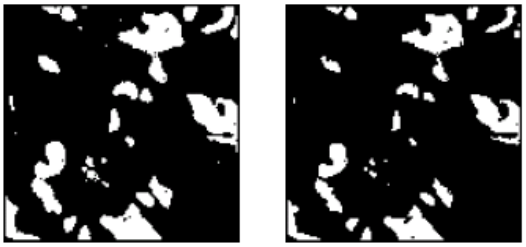


Figure 9 *Erosion of binary image A, and resulting in binary image B*

As seen above, A is the binary source image. B represents the source image after erosion.

2.5.4.2 Low-pass filters

Low-pass filters are used to filter out small particles that are created once an image has been processed, and is based on the widths of these particles. The widths of the particles are used to define the filter size. Having a filter given a size of N, the low-pass filter will eliminate all particles whose widths are less than or equal to N - 1 pixels. These particles disappear after (N – 1) / 2 erosions. Low-pass filters use erosions to select the particles to be removed. As the filters cannot distinguish between particles with widths of 2x pixels from particles with widths of 2x-1 pixels, one erosion cycle will eliminate both particles that are 2 pixels wide and 1 pixel wide as well. Both the high-pass and low-pass morphological filters use erosions to select particles for removal. Since erosions or filters cannot discriminate particles with widths of 2x pixels from particles with widths of 2x – 1 pixels, a single erosion eliminates both particles that are 2 pixels wide and 1 pixel wide. Table 1 shows the effect of low-pass filtering.

Table 1 *The effect of low-pass and high-pass filtering*

Filter Size (N)	Low pass Filter
N is an even number (N = 2x)	Removes particles with a width less than or equal to 2x – 1 Uses x – 1 erosions
N is an odd number (N = 2x + 1)	Removes particles with a width less than or equal to 2x Uses x erosions

2.5.4.3 Convex hull function

The convex hull morphological function is often used to enclose particles that need to be measured. Convex hull functions are often applied to binary images once they have been processed through a filter. Filters often create unwanted holes in the particles, causing the detecting, counting and tracking of these particles to be problematic. The convex hull function calculates the convex envelope around each particle. Convex hull functions effectively close the particle by filling all holes created by a filter, and thus making the process of generating a particle report on the binary image accurate when detecting, counting and tracking the particles found within the image. Figure 10 A shows the original labelled image used in this example. Figure 10 B shows the results after an MVS applies the convex hull function to the image.

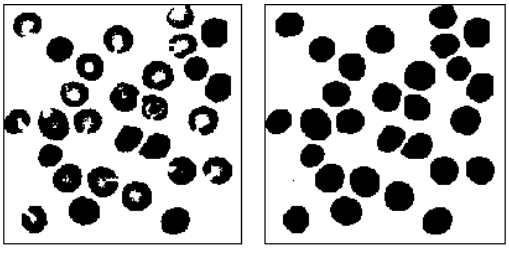


Figure 10 *Application of a convex hull function on a binary image. A represents a binary image after applying a low-pass filter. B represents the same binary image with a convex hull function applied to it* [37].

Having applied a convex hull algorithm to the binary image, measurements such as distance between features, size of features, number of features and location of features, can be extracted by image processing software through the generation of a particle report.

2.6 Pattern matching

Pattern matching is the technique used for quickly locating known reference patterns within an image. Pattern matching can provide applications with information about the presence or absence and number and location of a known reference within an image. There are three main areas of application of pattern matching; alignment, inspection and gauging [38]. Various methods can be implemented in the tracking of objects, viz. mean-shift and EM-based mean-shift [39]. Human motion analysis has been at the attention of computer vision for many years, with applications in various sports, such as basketball [40][41] and soccer [42]. Table 2 shows the various applications of pattern matching.

Table 2 *Applications of pattern matching and descriptions*

Applications of Pattern Matching and Descriptions	
Application	**Description**
Alignment:	Alignment is used to determine the position and orientation of a known object by locating its features within an image. These features are used as points of reference to locate the object.
Inspection:	Inspection, as the name implies, is used to find simple flaws on an object that is being produced, and thus grading the object. This is mainly used for quality control.
Gauging:	Gauging is a method of pattern matching used to measure the lengths, diameters and angles of an object. If the measurements fall below the accepted tolerance levels, the object is rejected.

Many software environments have been developed over the years to develop machine vision systems. When developing an MVS, various features need to be taken into consideration when selecting the software environment that is best suited. The following are key features that can be taken into consideration when selecting the best software environment. In Table 3, the features that should be considered when selecting the development software for an MVS.

Table 3 *Features that should be considered when selecting the development software for an MVS*

Feature	Selection consideration
Performance	Does the software meet all the requirements of the application in terms of speed and accuracy?
Learning	Are there software packages with which engineers arefamiliar? How much time will it take engineers to learn to use unfamiliar software packages effectively to accomplish the required tasks?
Ease of use	Does the software package offer drag-and-drop functionality, comprehensive help files and a digital community to provide support (blogs, social media)?
Reliability	Is the software stable, and does it function properly under all conditions?
Support	How much technical support is available, and would the software package still be supported in the near future?
Availability	Which software packages are freely available?

2.7 Information analysis

Information analysis within an MVS involves the processing of captured images and the extraction of useful information. Once the information that is deemed useful is captured during the image-processing step of an MVS, the information is processed to produce a result as required by the specific application of the MVS. Before machine learning, purpose-specific software was developed in order to process the information obtained by the MVS, but more recently machine learning is

used. Machine learning in the form of artificial neural networks (ANNs) has proven to be a robust alternative to the purpose-specific software, as ANNs can recognise intricate patterns captured within the information. Although ANNs are still programmed manually, modern software suits are used to develop a functional ANN in order to process information [30][31][43].

2.8 Modern methods of machine vision systems

From the early days at the Hitachi Central Research Laboratory in the early 1960s, MVSs have found their way into many aspects of our modern everyday life:

Table 4 *Examples of MVS applications*

Field	Application
Sports entertainment	Tracking and trajectory prediction of a ball in motion [44].
Sports strategy	Automatic player tracking and strategy determination in sports [45].
Gaming and movies	The capturing of the natural movements of objects in a 3D space such as human bodies [46].
Law enforcement	Automatic number plate recognition, traffic control, surveillance and safety [47].
Medical and pharmaceutical	Sorting and counting tablets, blister pack inspection, blood cell counting [21].
Material science	Surface inspection, particle analysis and counting [21].
Robotics	Automatic part assembly, robot guidance, area mapping [21].

2.9 Artificial neural networks

Artificial neural networks (ANNs) are computational networks that attempt to simulate the networks of a neuron of the biological central nervous system. By simulating the biological neural network, it can then be viewed as a novel computer architecture as well as a novel algorithmisation architecture to achieve basic computational operations, such as addition, multiplication and other fundamental logic operations, to solve otherwise complex problems [48]. An ANN becomes functional when the artificial neurons are linked together by weights that are automatically adjusted through learning when an activation function is applied [22][49].

2.9.1 ANN topologies

Before developing an ANN, there are several factors that need to be taken into consideration before one can choose the correct ANN topology for a specific application. In order to determine what the best topology is for an application, the following questions should be asked [49]:

- What is the accepted level of accuracy of the ANN?
- How efficiently does the network topology need to be concerning the application?
- How many layers are needed?
- How many neurons should be in each layer?
- How much information is needed in order to obtain an acceptable level of accuracy?

Each of the topologies of ANNs differs, depending on the intended application. The most common topologies are seen in Figure 11.1 the Feed-Forward Single Layer Perceptron (FF-SLP) network topology, Figure 11.2 the Feed-Forward Multi-Layer Perceptron (FF-MLP) network topology and Figure 11.3 the Feed-Forward Back-Propagation (FF-BP) network topology.

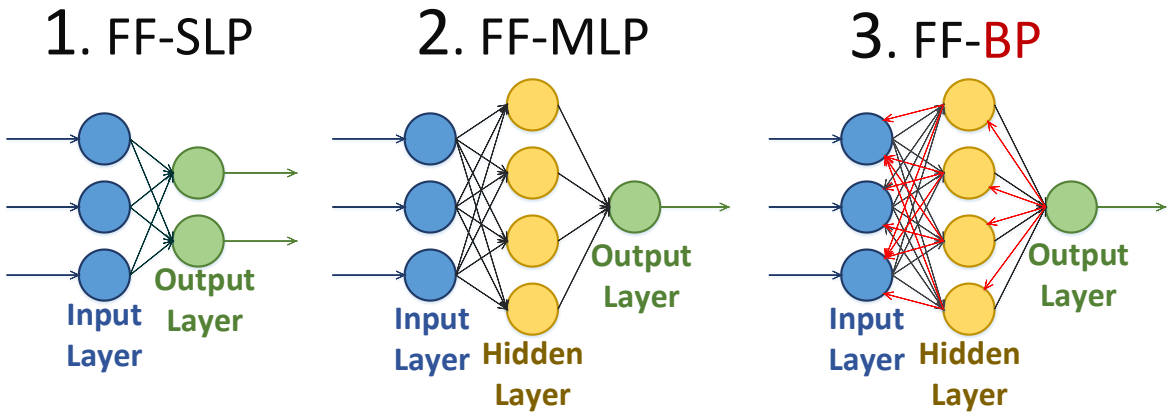


Figure 11 *Examples of ANN topologies. 1. Feed-Forward Single Layer Perception. 2. Feed-Forward Multi-Layer Perception. 3. Feed-Forward Back-Propagation.*

As the field of AI has grown over recent years, the efficiency of ANNs to process large amounts of information has also improved. The efficiency of ANNs to process large amounts of information has subsequently garnered the attention of large industries. ANNs have thus been used in industries such as city planning and energy supply, where it is used to determine the most efficient manner of energy supply coverage [49][50]. ANNs are also used for real-time language translation and image recognition. In the aerospace industry ANNs are also used to improve the current autopilot functionality within modern aircraft [51]. In the automotive industry there has recently been a drive to achieve self-driving cars. These self-driving cars make use of ANNs in order to perform automated obstacle detection and avoidance through image recognition [52].

One of the leading developers of ANN technologies has been Google. Google has successfully implemented ANNs across their products and services, such as search-specific advertisement placements, as well as suggested videos according to the videos watched by viewers on their YouTube Video streaming platform [53][54]. Google makes use of the millions of images uploaded to the internet to train their highly sophisticated image recognition, AI, to improve the accuracy of image search results when using their search engine.

2.9.2 ANN training

Before an ANN can be applied to execute a function, it needs to be trained [49][55][56]. The ability of ANNs to recognise and learn patterns from sequential training by self-adapting to input information is the main feature driving the implementation of ANNs across a variety of industries. A back-propagation algorithm adjusts the weights within each neuron in the neural network to achieve the self-adaptation process [57]. The process of adjusting the weights of the artificial neurons until the output target is reached, is defined as the training of an ANN [51].

Two types of training methods can be used when training an ANN [56][58]. These two training methods are unsupervised and supervised training, as depicted in Figure 12. Unsupervised learning is a class of ANNs that attempt to find a pattern within an information set. Unsupervised learning makes use of only input information and no corresponding output information. Unsupervised learning is typically used to determine the underlying structure or distribution within an information set in order to learn more about the set. Clustering is the most common way of performing unsupervised learning. Clustering automatically divides the information set into groups according to similarities found within the set. Unsupervised learning is often used when the information set is extensive, such as in the case of Big Data.

Supervised training is achieved by supplying an ANN with all the input variables (X), as well as all the output variables (Y), followed by applying a learning algorithm and mapping function based on these inputs to achieve the desired outputs (Y=f(X)). The two types of supervised learning are classification and regression. Classification predictive modelling is the task of approximating a mapping function (f) from the input variables to discrete output variables. A classification ANN is typically attempting to draw a conclusion from the observed values. Regression is predictive modelling used for approximating a mapping function from input variables to a continuous output variable. Regression is commonly applied when an output variable is a real or continuous value,

such as weight or mass. Currently, most developers make use of the supervised training method when training an ANN. Figure 12 shows an illustration of the subcategories of machine learning.

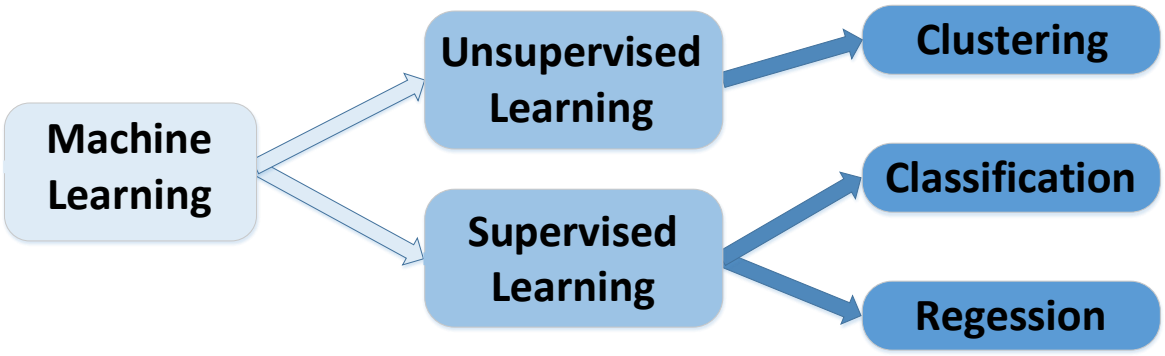


Figure 12 *Tree-diagram illustrating the subcategories of machine learning*

When a supervised network is being trained, it compares the current output to the desired output and applies the back-propagation algorithm until the desired output is achieved. The back-propagation algorithm adjusts the weights of each input neuron during the training, which will influence the behaviour and the output of the network [59]. How these adjustments are made, is decided by the learning algorithm chosen to train the network [57][60]. There are many training algorithms to choose from. The three most commonly used algorithms are the Levenberg-Marquardt algorithm, Bayesian regularisation algorithm and the Scale conjugate gradient algorithm [58][61].

When choosing a training algorithm, various training algorithms should first be evaluated in order to produce the most accurate output. When training an ANN, there are manners of determining the accuracy of an ANN. In order to determine the accuracy of an ANN, one can evaluate the mean square error (MSE) and the regression (R) values. Once an ANN is trained, the software within, which the ANN developed, would produce both the MSE and the R-values of the ANN. The MSE value is inversely proportional to the accuracy of the ANN. An MSE value of zero would indicate that there is no difference between the output of the ANN and the desired target output of the ANN.

Furthermore, an R-value of one indicates a close correlation and an R-value of zero indicates no correlation between the output of the ANN and the target output. When the obtained values of an ANNs MES and R-values are favourable, the weights of each artificial neuron is saved, and thus the ANN can be implemented [49][58][62][63].

2.9.3 Machine vision systems and artificial neural networks combination systems

A machine vision system (MVS) is developed to create digital models from captured images. The digital model is then processed, employing an image processing subsystem within the MVS in order to obtain information from the acquired images to achieve a specific goal. In order to achieve the goal, MVS is combined with machine learning algorithms to analyse and process the information obtained by the MVS. The application of an MVS and machine learning has resulted in considerable technological advances in several fields, of which some examples are shown in Table 5.

Table 5 *Applications of MVS*

Field	Applications
City planning	Efficient energy supply
Construction	Analysis of soil composition
Aerospace	High-performance autopilots for aircraft
Automotive	Automatic guidance systems for automobiles
Electronics	The development of ntegrated circuit chip design and circuit board fault finding
Entertainment	The creation of animation and special effects
Manufacturing	Process control and quality checks
Medical	Electroencephalogram (EEG) and electrocardiogram (ECG) analysis

There have been several implementations of Artificial Intelligence (AI) in the gaming industry. During the mid-1990s a team of researchers at IBM Research developed a competitive chess AI, called Deep Blue. Deep Blue was able to defeat the then reigning world chess champion, Garry Kasparov, in a six-game match in 1997 [1]. More recently researchers, funded by Elon Musk, developed an AI that beat a team of professional Dota-2 players in a show match in 2018 [64]. Furthermore, researchers at Wroclaw University in Poland implemented an AI in non-playable characters (NPC) to improve and increase the realism of these NPCs [65].

AI has also been implemented in gaming in order to make gaming environments look more realistic and more beautiful to improve player immersion. The AI was developed by Nvidia to upscale images of lower resolution to be displayed at a higher resolution. The AI upscaled the images by determining the best colour for each pixel to benefit small images that are upscaled or images that are zoomed in on. The AI developed by Nvidia, achieved this by implementing a Deep Neural Network (DNN) [66].

Game developers have also made use of AI when creating in-game worlds and environments. The development of a video game world depends fundamentally on the developers' designing skills, the time given to develop the game world, as well as the effort needed to complete and obtain a favourable result. AI algorithms have thus been implemented in order to generate game worlds. When an AI generates a game world, it is referred to as a procedurally generated world [67][68][69][70].

There have also been various attempts at integrating machine vision systems (MVS) into the world of gaming. One of the first attempts of integrating an MVS in gaming was in 1996. Researchers from the Mitsubishi Electric Research Laboratory (MERL), proposed a system that was able to make use of image processing algorithms and an input device to allow players to interact with a game, using their bodies and hand gestures. More recently Microsoft developers developed a motion tracking

system for their gaming console, known as the Xbox Kinect [71]. The Xbox Kinect can accurately track player movement and implement this movement as player input within video games. The Xbox Kinect sensor has also been implemented in various research studies. an Xbox Kinect was, for example, used to investigate the effects of additional virtual reality training using Xbox Kinect on upper extremity function, including the range of motion, motor function and gross manual dexterity, in stroke survivors with hemiplegia [72].

MVS and AI have been used in various sports studies as well. Researchers Davoodi and Khanteymoori, for example, made use of an MVS to capture and process various horse races over 100 races, and an AI to determine which of the horses would be most likely to win with the next encounter. The neural net developed by the researchers, achieved an accuracy of 77% in predicting the correct horse [73]. Researchers Tax and Joustra, who used information of a Dutch football competition over the past 13 years to predict the results of football matches, followed a similar approach. The neural network developed by Tax and Jouster achieved an accuracy of 56.1% when predicting the results of the Dutch football competition [74]. In 2008 researchers attempted to predict the results of four different sports –NFL (Rugby League), AFL (Australian Rules Football), Super Rugby (Rugby Union) and the English Premier League Football (EPL) –using data from 2002 up to 2008, and achieved an average accuracy of 67.5% [75]. As for strategising, researchers Cheshire and Halasz, as well as Liu and Carr, used an MVS to track players and analyse the strategies teams used on a basketball court, as well as on an indoor hockey field used by [45][76]. The studies mentioned above show that an MVS and an ANN can accurately predict the outcomes of sports, as well as the tracking of players, but this has yet to be implemented in the world of electronic sports. For this study an automatic strategy predictor (ASP) was envisaged by combining an MVS and an ANN for the capturing and processing of processional Counter-Strike Global Offensive (CSGO) e-matches, to learn their strategies from previous encounters and subsequently predict the strategies that the team would implement in a match.

2.12 Conclusion

In conclusion, literature illustrates that strategising in E-sports is as important as in conventional sports to be successful. In conventional sports the process of strategising has been modernised with the implementation of MVS to obtain Big Data to train an ANN successfully. Literature furthermore shows that the implementation of an MVS and ANNs has proven to be successful in various industries, including the E-sports industry, but has yet to be implemented in the same manner as they are implemented in conventional sports. Thus if an MVS is successfully combined with an ANN to create an automatic strategy predictor (ASP), it would have the potential to contribute a significantly to the professional E-sports industry by providing an accurate strategy predicting system which will reduces the time spent on reviewing thousands of hours of historical information of competing teams in preparation for matches.

Chapter 3

Study Design

3.1 Introduction

"If you know the enemy and know yourself, you need not fear the result of a hundred battles."— Sun Tzu, The Art of War

Strategising has been the cornerstone of all environments where two sides are pitted to determine the victor. Such strategising actions are most applicable in the military field, which dates back to the ancient Greek and Chinese empires. These ancient civilisations used well-planned methods and actions, based on their studies of their enemies before entering into a battle [77].

Board games were used as tools to help military trainers recruit and train leaders alike in order to understand the fundamental principles of warfare. The Prussian military required officers to play a board game, called “Kriegsspiel”. Having the military officers play Kriegsspiel, high command realised that, while individual officers might understand the principals of combat, they might not know how to apply them when facing an adversary. By analysing what happened after a game of Kriegsspiel, they may see which factors that mattered and how their choices influenced the outcome of the game [78]. Throughout the 1920s and 1930s, the United States Navy made use of war games to design military plans against potential adversaries [79]. More recently, war gaming continues to offer opportunities for scholars when they attempt to better understand security dynamics of a Russian invasion if the Baltics were to happen [80].

Thus, the dogmas of military strategies have permeated competitive gaming. These military dogmas are nowhere more prominent than in electronic sports (E-sports). As with Kriegsspiel, professional E-sports teams often review their previous encounters with other teams in order to understand the

mistakes that were made during a match and learn from them, as well as better understand their adversaries. By examining an opposing team, one would be able to determine their habits, strengths, weaknesses, leadership traits, and aggressive and defensive responses they demonstrate. By doing so, a counter-strategy is developed to exploit the adversary's weaknesses and defensive responses, much like generals in a war room preparing for battle [81].

When a professional Counter-Strike Global Offensive (CSGO) team is preparing for a match, the coach or captain obtains as much information about their adversary as possible. The process of obtaining this information is often performed by reviewing previous matches of an opposing team and noting all relevant information. The process of reviewing these matches is time consuming, as teams often play numerous matches per day during a tournament. Each of these matches are often an hour long, excluding overtime.

This study aimed at producing a system that was able to study the legacy information of a specific professional CSGO team on a specific map they play on to reduce the time taken to prepare for a match against such a team. The development of such a system involved the implementing the image-acquisition and image-processing capabilities of the LabVIEW software suite, and the artificial neural net programming capabilities of MatLab. The system is called the automatic strategy predictor (ASP).

3.2 Study design

The conceptual framework of this study takes on a flow diagram of chevron shapes. The first of these chevron shapes flows into the next, and so on. Each chevron shape represents a different milestone; and each milestone has a title as well as a deliverable. The proceeding stage cannot begin without the previous milestone's deliverable; thus, each needs to be completed sequentially. Figure 13, is the conceptual framework of the study.

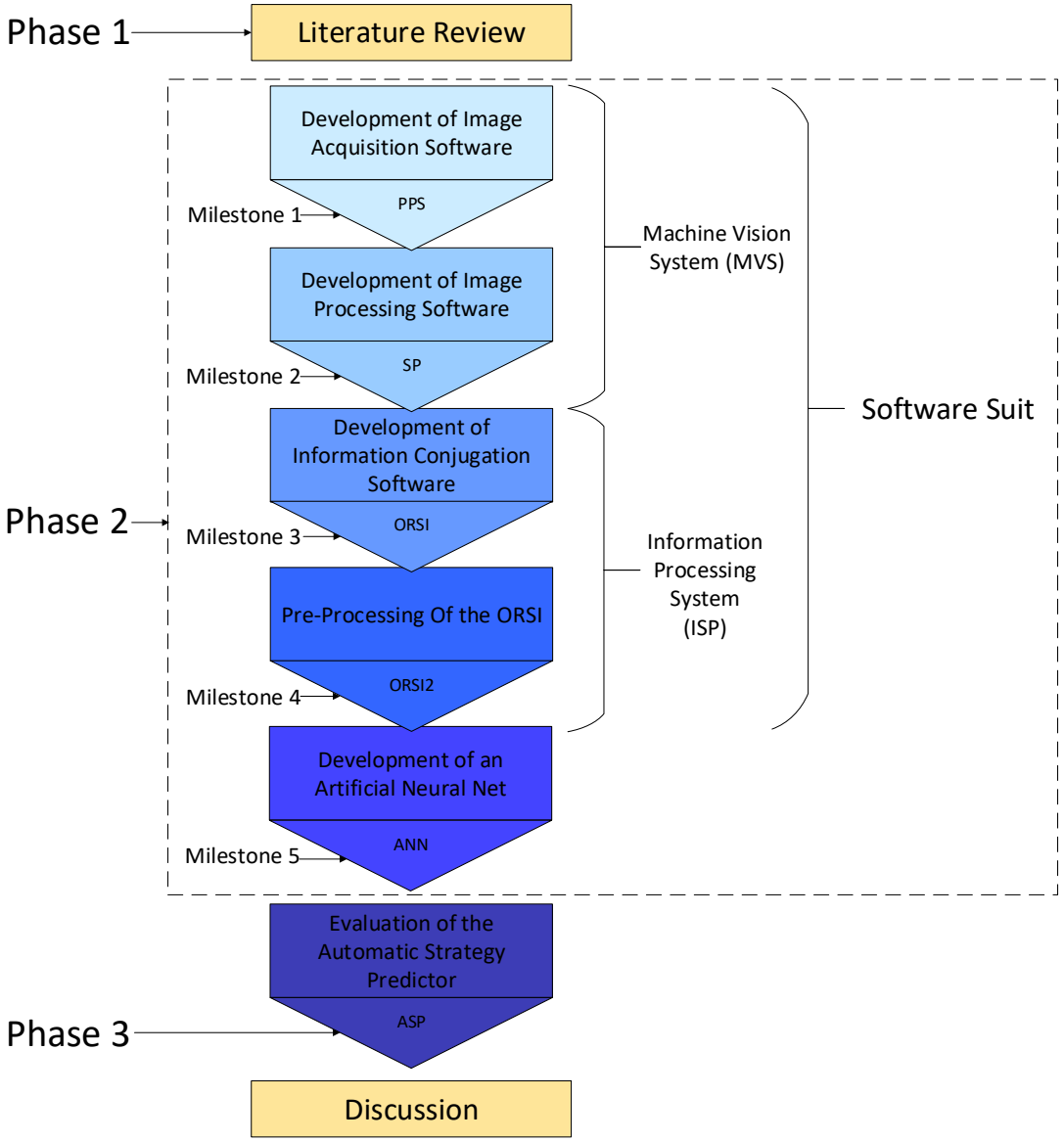


Figure 13 *Logical diagram of the development of the study*

Phase 2 of the study is broken down into five milestones that were achieved. Each milestone that was achieved, marks the completion of the development of the various software components within the machine vision system, the information processing system and the artificial neural network. Table 6 provides a descriptive summary of Figure 13.

Table 6 *Description of each Phase and Milestone of the study*

Phase 1: Literature Review	
Chevron	**Description**
Phase 1 Literature Review	The first phase of the study entails an extensive review of various literature of the gaming industry and electronic sports as the development of machine vision, and machine learning systems are implemented within various fields in engineering and gaming.
Phase 2: Development of the Software Suit	
Chevron	**Description**
Milestone 1: Player Position Stencil (PPS)	The aim of this milestone is to produce the player position stencil (PPS). The production of the PPS involves the use of image processing and image acquisition. By performing image acquisition and image processing, the player position stencil (PPS) is produced by the first software of the image processing software suite. The PPS is the deliverable needed to proceed to the next milestone.
Milestone 2: Stencil Processor (SP)	The aim of this milestone is to produce the stencil processor (SP). The SP then retrieves the stencils produced by the PPS, and further processes the stencils in order to extract the relevant player position information that was needed form the stencils. This player position information is then stored in a memory to be processed further by the next milestone within the study.
Milestone 3: Overall Round Strategy Information (ORSI)	The aim of this milestone is to produce the overall round strategy information (ORSI). In order to create the OSRI, two sets of information are concatenated by the information concatenator (IC). The IC concatenates the first set of information, the player PPI that was generated by the SP, to the second information set. The second information set is the end-of-round information. The end-of-round information will be referred to as the additional info. Counter-Strike Demo Manager obtained the additional info. The concatenated file of the two information sets is referred to as the overall round strategy information (ORSI) file and is the deliverable of the third milestone. The ORSI file contains the information of very round, of every match, of every year, for team Fnatic on the map de_Mirage.

Milestone 4: Overall Round Strategy Information 2 (ORSI2).	The aim of this milestone is to produce the overall round strategy information 2 (ORSI 2). The production of the ORSI2 involves the pre-processing of the ORSI. The pre-processing of the ORSI is performed by two software packages developed within the LabVIEW 2018 software suite. The objective of the first pre-processing software is to firstly remove all unnecessary information from the ORSI and, secondly, to re-arrange the ORSI. The objective of the second pre-processing software is to separate the ORSI into the training information sets for each round. Each round would thus have an ORSI2 file for the training of an artificial neural net (ANN).
Milestone 5: Artificial Neural Net (ANN)	The aim of this milestone is to produce an ANN for each round within one half of a match. In this milestone, the ORSI2 is prepared to train an ANN. The preparation of the ORSI2 is conducted within the MatLab Software suite. Each ANN is the automatic strategy predictor (ASP) of each round.
Phase 3: Evaluation of the ASP	
Chevron	**Description**
Phase 3: Evaluation of The ASP	Each ASP developed in the last milestone of the previous phase, is exported to the MatLab Simulink environment. By exporting each ASP to Simulink, they can be evaluated to determine the prediction accuracy of each ASP.

3.3 Discussion

Phase 1 presents the study and review of previous work of researches in the fields of gaming, machine vision and machine learning. The steps undertaken to construct the MVS (Phase 2, Milestones 1 and 2) are presented in Chapter 4. The steps undertaken to develop the IPS (Phase 2, Milestones 3 and 4) are presented in Chapter 4. The steps taken during the development of the ASP (Phase 2, Milestone 5) are presented in Chapter 5. Furthermore, the evaluation of the results obtained from the ASP (Phase 3) is presented in Chapter 5. In this chapter an overall discussion of all the results are presented and integrated with current knowledge of this topic.

Chapter 4

Development of the Software Suite

4.1 Introduction

The structure of the study comprises three phases. The second of the three phases involves the development of a software suite that would perform all the processing and information collection in order to develop an automatic strategy predictor (ASP). The software suite consists of six software programs. The first two programs that weredeveloped, is defined as the machine vision system (MVS). The third, fourth and fifth software programs form the information processing software of the suite. The conceptual framework of the software suite of the study is found in Figure 14.

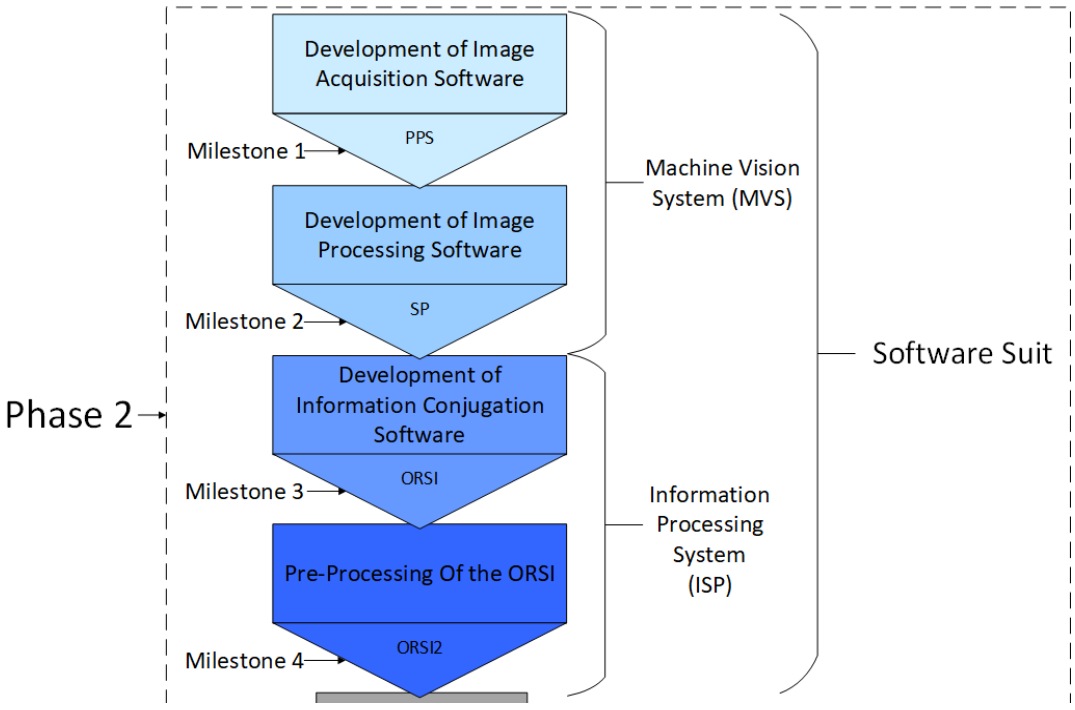


Figure 14 *Conceptual frame workflow diagram, highlighting the software suite development process*

The first MVS software is called the player position stencil producer (PPSP). The PPSP achieved the first milestone by producing the PPS. The second MVS software package is the Stencil Processor

(SP). The second milestone is achieved by the SP by processing the PPS. The product of the SP is thus the player position information (PPI). The information processing system (IPS) concatenated the PPI and the miscellaneous round information. Figure 15 depicts a logical flow diagram of the MVS.

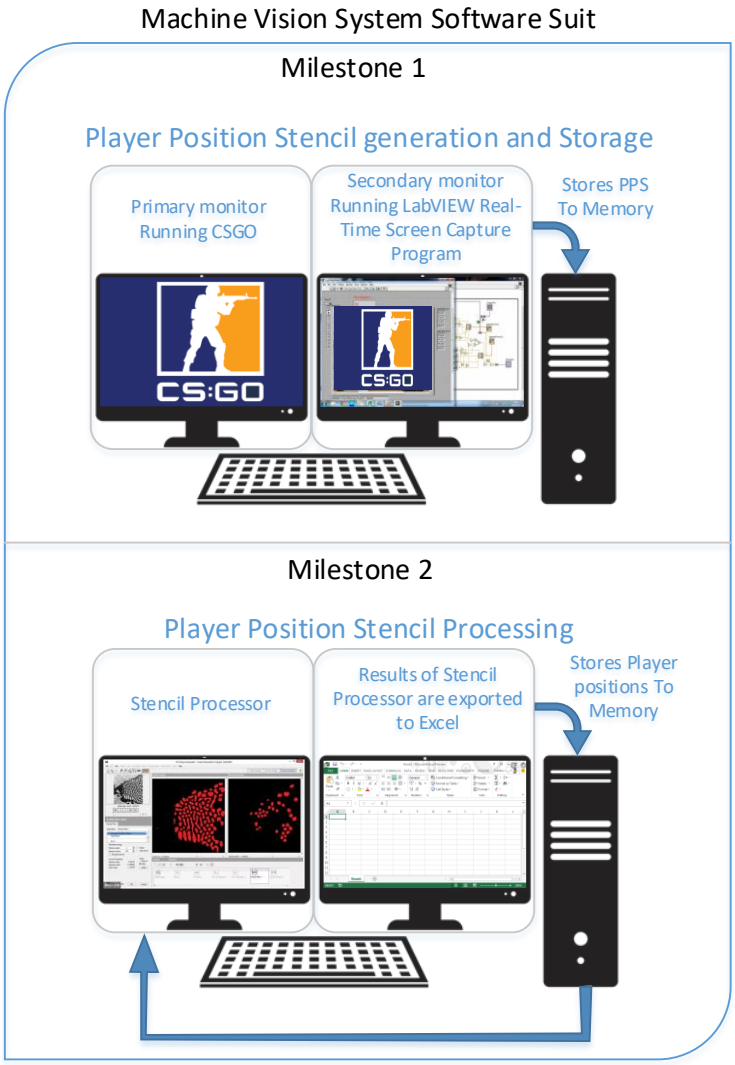


Figure 15 *Logical flow diagram of the machine vision system of the software suite*

Developing the third, fourth and fifth software packages in LabVIEW 2018, completes the information pre-processing portion of the software suite. Processing the information obtained by the MVS, by the IPS, would allow an artificial neural net to be created and trained within the MatLab software suite.

The first of the IPS software is the Information Concatenator (IC). The IC achieves the third milestone. Achieving the third milestone entails the concatenation of the information obtained by the MVS and the miscellaneous round information obtained through the Counter-Strike Demo Manager software. The overall round strategy information (ORSI) was the name given to the output obtained from the IC. The second and third units of IPS are the information pre-processors (IPp). The IPp achieves the fourth milestone. Achieving the fourth milestone entails the processing of the ORSI into a format needed to train an artificial neural net. The overall round strategy information 2 (ORSI2) is the name given to the output of the IPp. Figure 16 is a logical flow diagram of the IPS.

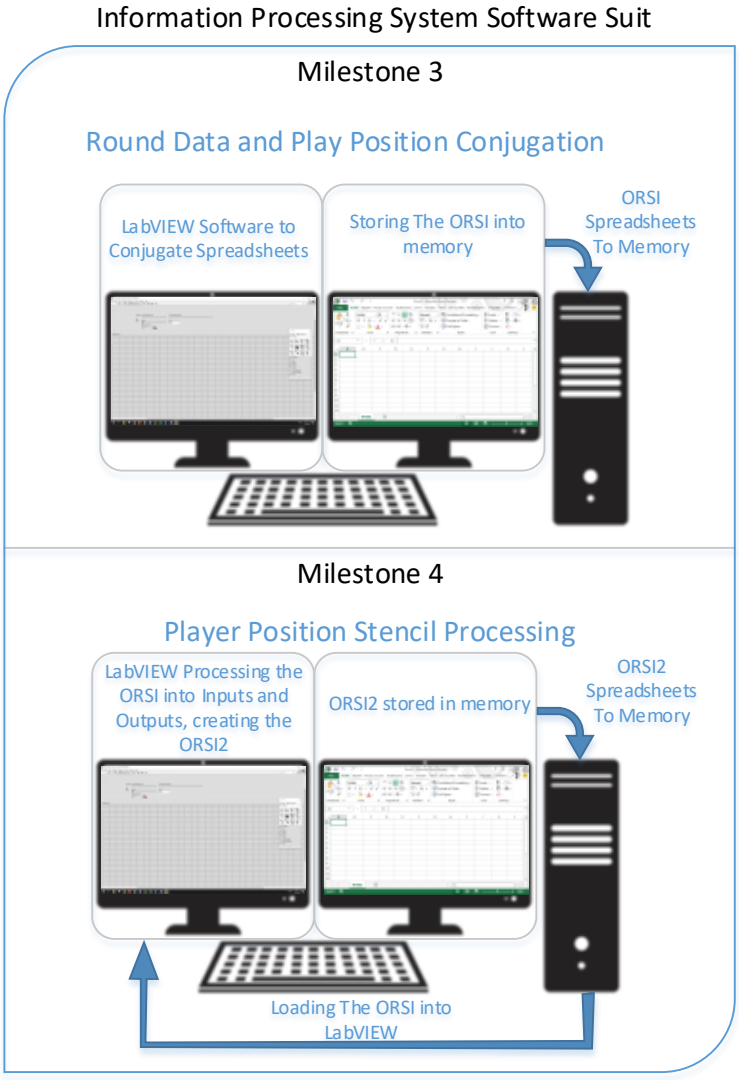


Figure 16 *Logical flow diagram of the information processing system of the software suite*

4.2 Software development environment: LabVIEW

LabVIEW 2018 software suite has been the tool for the development of the second software of all four of the software units within the software suite. The LabVIEW software suite is a block-orientated programming environment with a drag and drop style of programming. The LabVIEW programming environment consists of two windows, viz. the front panel window (FPW) and the back panel window (BPW). Figure 17A depicts a basic BPW, where Figure 17B depicts a basic FPW. The logical programming development of the four software programs was executed in the BPW. The FPW contains all the user-interface controls, logical indicators and relevant information.

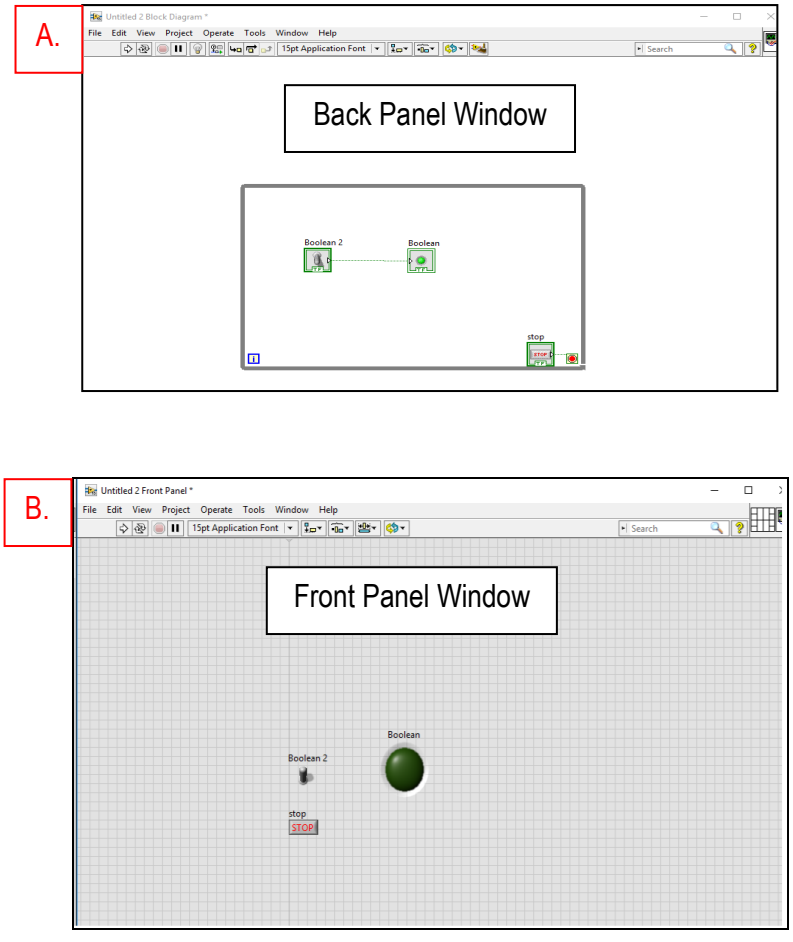
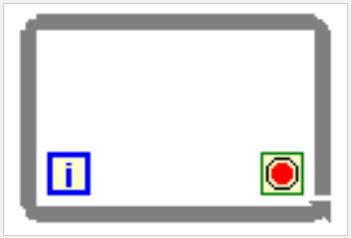
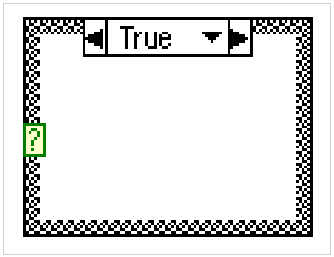
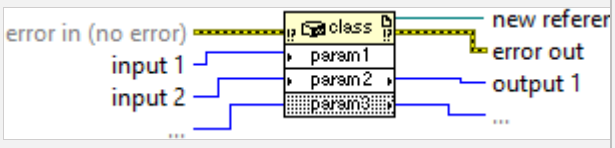
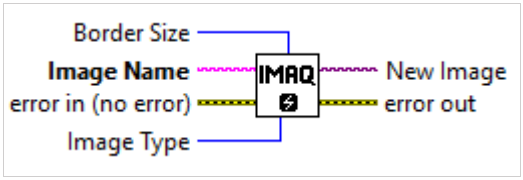
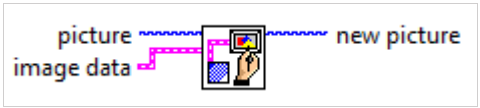
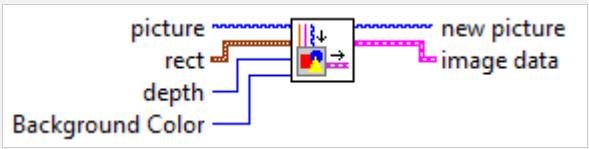


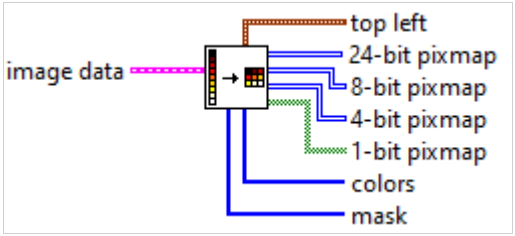
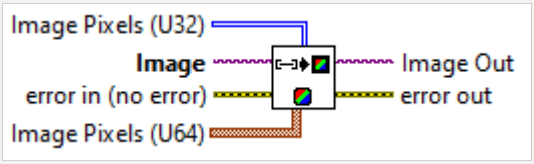
Figure 17 *Windows of the LabVIEW development environment. A. BDW and B. FPW*

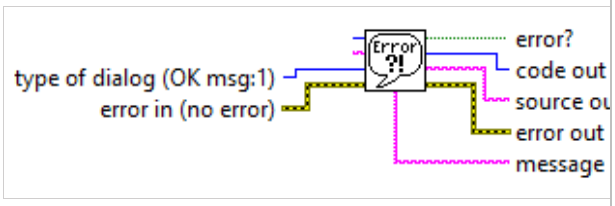
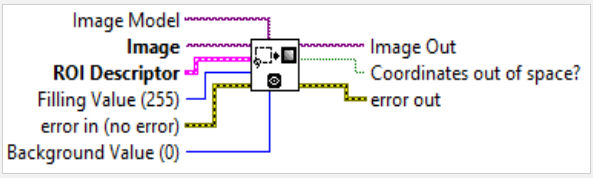
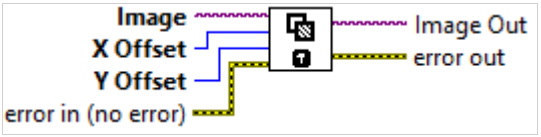
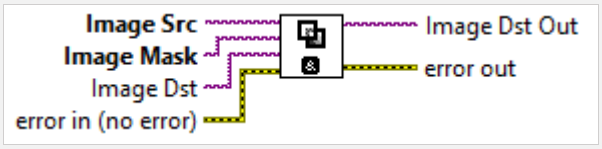
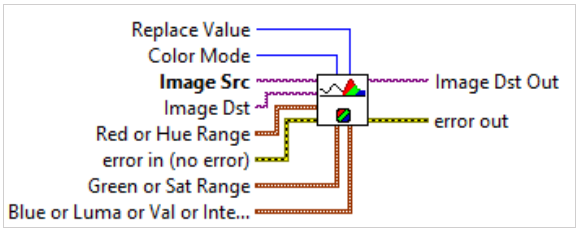
The LabVIEW software suite is host to a collection of pre-programmed code blocks, called virtual instruments (VIs). VIs are then dragged out of their respective tabs in the tool pallet and dropped on the BPW and connected by “wires” that transfer information between VIs. Each VI found in the tool pallet has its specific function, input type and output type, that needs to be taken into consideration when programming a LabVIEW VI. The outputs of one VI can be connected to the inputs of another VI by wires. The output can also be displayed on the FPW for the user by using indicators. A comprehensive list of the VIs used in the development of each LabVIEW software is shown in Table 7.

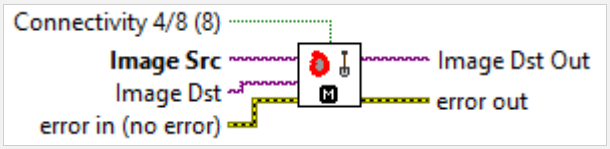
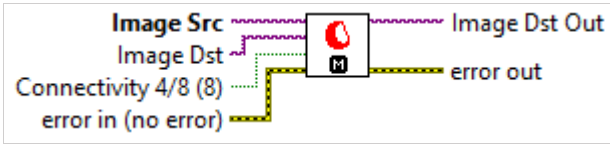
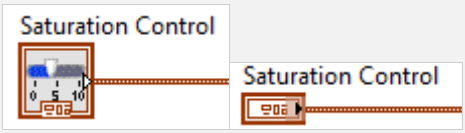
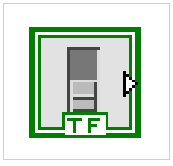
Table 7 *A comprehensive list of the VIs used in the development of each unit of LabVIEW software*

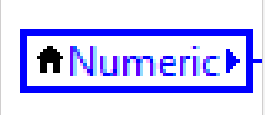
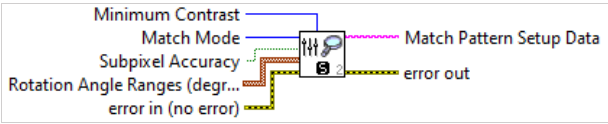
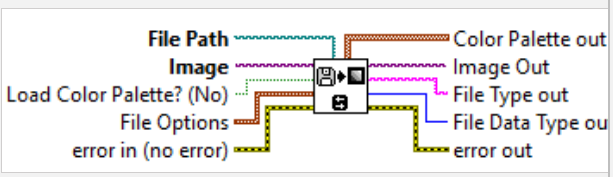
Identifier	Name of VI	Block image	Description
A	While Loop		A while loop is a sub-VI that will execute multiple times until specific conduction is met. Without a condition, the loop will run at least once.
B	Case Structure		Contains one or more sub-diagrams, or cases, exactly one of which executes when the structure executes. The value wired to the case selector determines which case is to be executed.

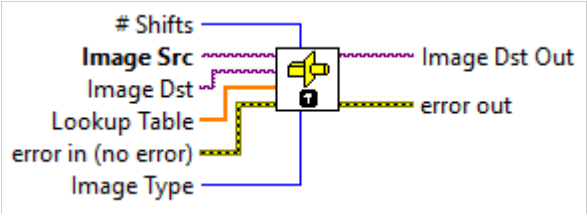
C	Invoke Node		The invoke node is used to perform a method on a VI using a VI server. Invoke nodes are not bound by the VI's connector panes and are free to call any VI.
D	IMAQ Create VI		This VI creates a temporary memory location for a captured image by the Vision Acquisition VI.
E	PNG Data to LV Image VI		The png data to LV Image. VI converts a PNG binary stream to a LabVIEW image.
F	Draw Flattened Pixmap VI		The Draw Flattened Pixmap VI draws a 1, 4 or 8-bit pixmap or a 24-bit RGB pixmap into a picture. This VI takes a 1D array of bytes as input, and assumes the user completes all packing and padding.
G	are to Pixmap VI		Converts a picture to a cluster of image data you can use to perform

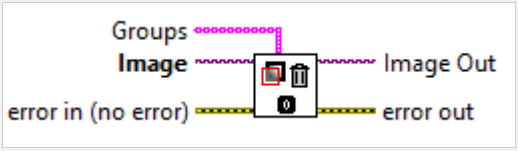
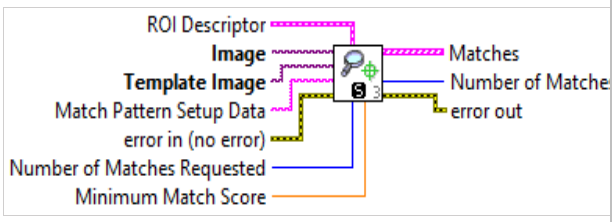
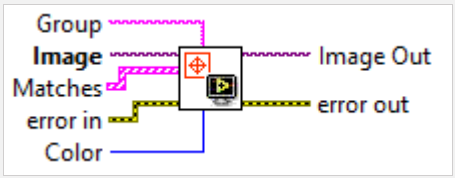
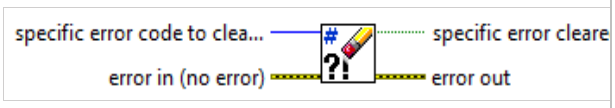
			certain tasks with the image, such as save it to a file using the Graphics Formats VIs.
H	Unflatten Pixmap VI		The Unflatten Pixmap VI converts a cluster of image data into a 2D array.
I	IMAQ Array To Colour Image VI		It creates a colour image from a 2D array. This VI receives the values as a 2D array of either unsigned 32-bit integers or clusters of four unsigned 16-bit integers, depending on the bit depth of Image.
J	IMAQ Display Image VI		The IMAQ Display Image VI is used to display an image.
K	IMAQ Dispose VI		The IMAQ Dispose VI clears the memory space reserved by the IMAQ Create VI for the image that was captured by the Vison Acquisition VI.

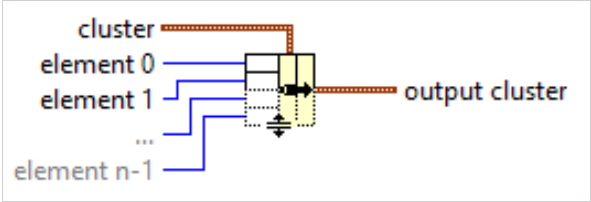
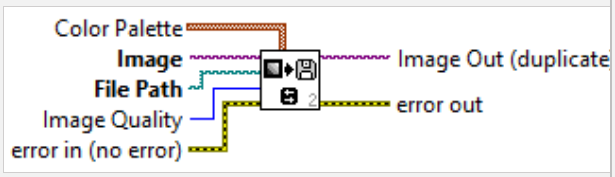
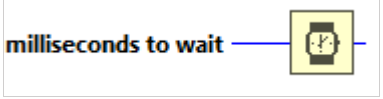
L	Simple Error Handler VI		Simple Error Handler VI indicates whether an error occurred. If an error occurred, this VI returns a description of the error and optionally displays a dialogue box.
M	IMAQ ROI To Mask 2 VI		The IMAQ ROI To Mask 2 VI transforms a region of interest to a mask.
N	IMAQ Set Offset VI		The IMAQ Set Offset VI defines the position of an image mask about the origin of the co-ordinate system (0, 0).
O	IMAQ Mask VI		IMAQ Mask VI recopies the image source into the image destination. If a pixel value is 0 in the image mask, the corresponding pixel in image destination is set to 0.
P	IMAQ Color Threshold VI		IMAQ Color Threshold VI applies a threshold to the three planes of an RGB or HSL image and places the result into an 8-bit image.

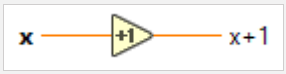
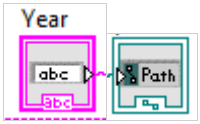
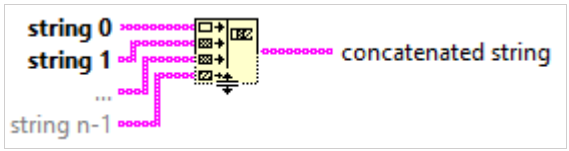
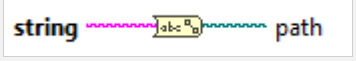
Q	IMAQ Fill Hole VI		IMAQ Fill Hole VI fills the holes found in a particle. The holes are filled with a pixel value of 1. The source image must be an 8-bit binary image. The holes found in contact with the image border are never filled, because it is impossible to determine whether or not these holes are part of a particle.
R	IMAQ Convex Hull VI		IMAQ Convex Hull VI draws the convex hull for each particle in the image
S	Numeric Controls and Indicators		Use numeric controls and indicators on the front panel to enter and display numeric data in LabVIEW applications.
T	Boolean Controls and Indicators		Use Boolean controls and indicators located on the Boolean subpalettes to enter and display Boolean (TRUE/FALSE) values with objects such as buttons, switches, and LED lights.

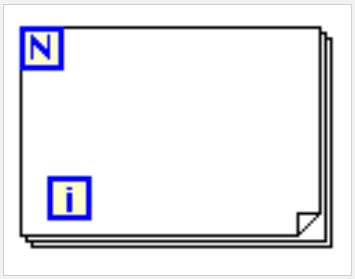
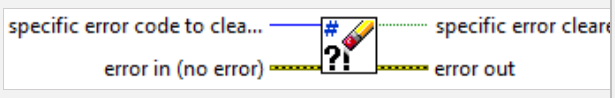
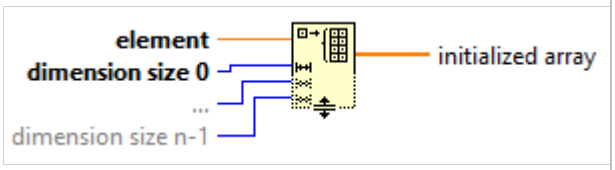
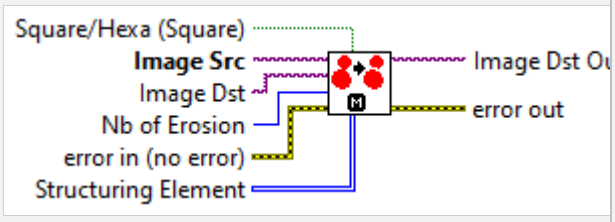
U	Local Variable		Create local variables for front panel objects in a single VI when you do not have access to a front panel object, or when you need to pass data between block diagram nodes. When you create a local variable, it appears on the block diagram, but it does not appear on the front panel.
V	IMAQ Setup Match Pattern 2 VI		IMAQ Setup Match Pattern 2 VI sets parameters that are used during the matching phase of pattern matching.
W	IMAQ Read File VI		The IMAQ Read File VI reads an image file. The file format can be a standard format (BMP, TIFF, JPEG, JPEG2000, PNG, and AIPD) or a non-standard format known to the user. In all cases, the read pixels are automatically converted into the image type passed by Image.

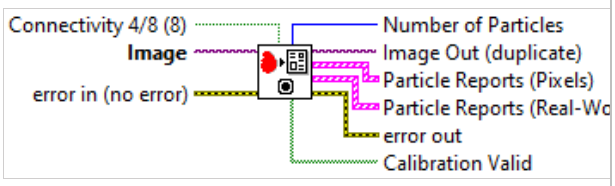
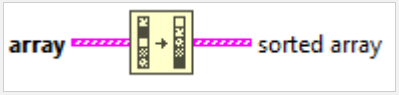
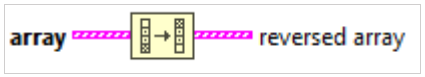
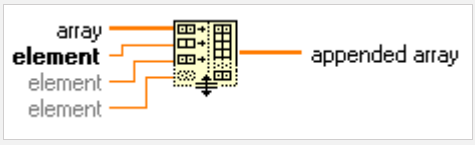
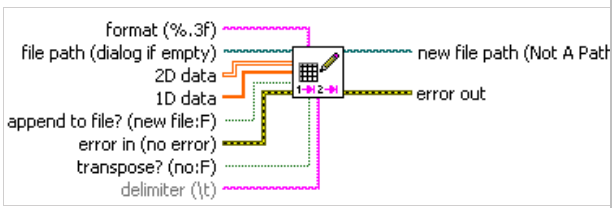
X	IMAQ Cast Image VI		The IMAQ Cast Image VI converts the current image type to the image type specified by Image Type. If you specify a lookup table, the IMAQ Cast Image VI converts the image using a lookup table. If converting from a 16-bit image to an 8-bit image, the VI executes this conversion by shifting the 16-bit pixel values to the right by the specified number of shift operations and then truncating to get an 8-bit value.
Y	Not Equal To 0? Function		This function returns TRUE if x is not equal to 0. Otherwise, it returns FALSE.
Z	Or Function		The Or Function computes the logical OR of the inputs. Both inputs must be Boolean values, numeric values, or error clusters. If both inputs are FALSE, the function returns FALSE. Otherwise, it returns

			TRUE.
AA	IMAQ Clear Overlay VI		IMAQ Clear Overlay VI clears the image overlay.
AB	IMAQ Match Pattern 3 VI		This VI searches for a pattern or template image in an inspection image. Run the IMAQ Learn Pattern 5 VI before this VI to configure the template image for the matching phase.
AC	Overlay Pattern Match Results		This function creates an overlay on the image when a match is found on the template that is provided.
AD	Clear Errors VI		The Clear Errors VI resets the error status to no error, code to 0, and source to an empty string. Use this VI when you want to ignore an error.
AE	Not Equal? Function		This function returns TRUE if x ≠ y. Otherwise, this function returns FALSE. You can

			change the comparison mode of this function.
AF	Bundle Function		This function assembles a cluster from individual elements.
AG	Greater Or Equal? Function		This function returns TRUE if x is greater than or equal to y. Otherwise, this function returns FALSE. You can change the comparison mode of this function.
AH	Round To Nearest Function		This function rounds the input to the nearest integer. If the value of the input is midway between two integers, the function returns the nearest even integer.
AI	IMAQ Write File 2 VI		This VI writes the image to a file in the selected format.
AJ	Wait (ms) Function		This function waits the specified number of milliseconds and returns the value of the millisecond timer.

AK	Increment Function		This function adds 1 to the input value.
AL	String and Path Controls and Indicators		The File path indicator displays a file path on the UI.
AM	Number To Decimal String Function		This function converts numbers to a string of decimal digits at least width characters wide or more extensive if necessary. If the number is floating-point or fixed-point, it is rounded to a 64-bit integer before conversion.
AN	Concatenate Strings Function		This function concatenates input strings and 1D arrays of strings into a single output string. For array inputs, this function concatenates each element of the array.
AO	String To Path Function		This function converts a string, describing a path in the standard format for the current platform, to a path.

AP	Unbundle By Name Function		This function returns the cluster elements whose names you specify.
AQ	For Loop		It executes its subdiagram n times, where n is the value wired to the count (N) terminal. The iteration (i) terminal provides the current loop iteration count, which ranges from 0 to n-1.
AR	Value indicator		The Value indicator displays a variable on the UI.
AS	Clear Errors VI		This VI resets the error status to no error, code to 0, and source to an empty string.
AT	Initialize Array Function		This function creates an n-dimensional array in which every element is initialised to the value of the element.
AU	IMAQ Separation VI		This VI separates touching particles, particularly small isthmuses found

			between particles.
AV	IMAQ Particle Analysis Report VI		This VI returns the number of particles detected in a binary image and an array of reports containing the most commonly used particle measurements.
AW	Sort 1D Array Function		This function returns a sorted version of an array with the elements arranged in ascending order.
AX	Reverse 1D Array Function		This function reverses the order of the elements in the array, where the array is of any type.
AY	Build Array Function		This function concatenates multiple arrays or appends elements to an n-dimensional array.
AZ	Write Delimited Spreadsheet VI		This VI converts a 2D or 1D array of strings, signed integers, or double-precision numbers to a text string and writes the string to a

			new byte stream file or appends the string to an existing file.

4.3 Software development environment: MATLAB

MATLAB is a programming platform primarily used by engineers and scientists. The basis of MATLAB is its programming language. The MATLAB programming language is matrix-based, allowing natural computational mathematics. The utilisation of the MATLAB software varies in a number of ways, from the analysis of data, the creation of models and simulation applications and the development of algorithms to the development of deep learning and machine learning applications.

4.4 Milestone 1: Image Acquisition

LabVIEW 2018 software suite has been the development platform for the development of the second software of the IPS. The first software developed for the MVS, was named the player position stencil producer (PPSP). The PPSP consisted of three subroutines and produced the player position stencil (PPS). The three subroutines of the PPSP and their functions were as follows:

- Subroutine 1: ***Screen Capture.*** The primary function of this subroutine was to capture the user interface(UI) of Counter-Strike Global Offensive (CSGO) that was being displayed by the primary screen. The CSGO UI displayed a demo of a previous match as it was being played by the two teams.

- Subroutine 2: ***Background Removal.*** The Background removal subroutine created an ROI around the in-game minimap, and thus player positions were effectively

captured by performing feature extraction on the ROI. The ROI minimised the impact of such a process on the machine. The feature extraction portion of the subroutine isolated only one team's colour out of the ROI using colour thresholding.

- Subroutine 3: ***Pattern Matching.*** Pattern matching software matches a template arrangement of pixels to an input image. When the PPSP pattern matching software finds a match between the template and the in-game round timer, the thresholded image was stored to memory by the PPSP. The PPSP software template was a snapshot of the in-game round timer at a specific time.

4.4.1 PPSP Subroutine 1: Screen Capture

Image acquisition is an integral part of any machine vision system (MVS). Image acquisition is performed in various ways by software, such as screen capture, with or without camera equipment. When performing image acquisition, such as screen capture, one has to take multiple factors into consideration. Factors include the resolution of the acquired image and the frame rate at which the machine will be performing the image acquisition.

Resolution: Resolution refers to the number of pixels an image consists of, typically denoted as 1024 x 768. The first value denotes the horizontal number of pixels, and the second denotes the vertical number of pixels. The the image therefore is 1024 pixels wide and 768 pixels high.

Furthermore, in any MVS the resolution of the image is crucial, as it directly influences the data that can be obtained from the image by image processing software. If the image that is acquired is of a low resolution, it will result in pixelation. Pixelation will result in loss of data. Subsequently, the highest possible resolution was used by the PPSP software, when performing image acquisition to

obtain the player position stencil (PPS). The PPS was captured entirely by software, the PPSP, as no cameras were needed. The PPSP is a screen-capturing program developed in LabVIEW to capture the CSGO user interface (UI). The PPSP captured the CSGO UI at 1920 X 1080 resolution to ensure no information was lost. Figure 18.

Figure 18 *Counter-Strike Global Offensive User Interface*

4.4.2 PPSP Subroutine 2: Background Removal

When performing image processing, removing the background of an acquired image is often necessary in order to extract specific features that need further processing. Image processing was performed by stencil processing (SP) software and colour thresholding. Colour thresholding isolates a specific colour from an image and, in doing so, it turns the image into a binary image. How these binary images are structured, is a mask consisting of 1s and 0s as its pixels. Figure 19 depicts an example of this masking [34].

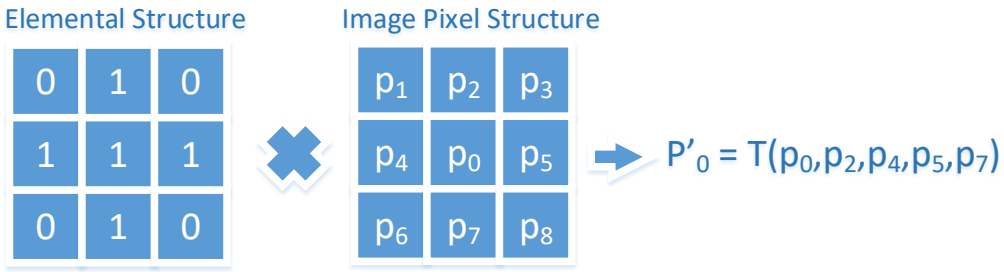


Figure 19 *Image binary masking*

In Figure 20 A can be seen as a typical capture of the in-game mini-map in full colour, as it was captured from the game**.** Once the PPSP captured the in-game mini-map, thresholding was applied by the PPSP to extract only the five players of the attacking team. Once the image was thresholded, it was converted into a binary image by the PPSP. The thresholded image is shown in B.

Figure 20 A. *Captured image of the CSGO in-game mini-map in full colour,* ***B.*** *Thresholded image of the captured image of the CSGO in-game mini-map.*

Frame Rate: The definition of a video is a visual media source that combines a sequence of images and displays them at a specific rate. Frame rate is the number of frames per second (FPS). The

higher the FPS, the smoother the video will appear, whereas if the FPS was low, the video will appear more jerky.

When screen capturing, the frame rate needs to be taken into consideration as well. When the frame rate is too low, loss of information will occur. The only benefit of having a low frame rate is that the processor will have less information to be processed per second. Low frame rate is juxtaposed to having a high frame rate, a fact one would consider to be beneficia,l as little to no data will be lost. Capturing images at a high frame rate will push a processor to its limits, and thus a stable middle ground was found in ten frames per second. Figure 21 is an illustration of FPS [82].

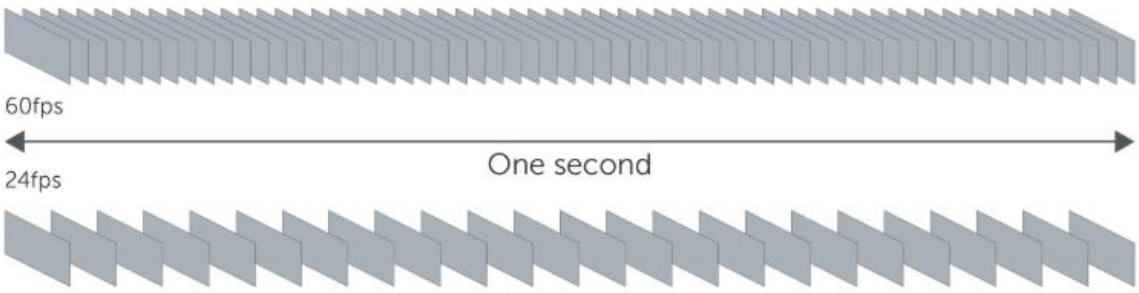


Figure 21 *Illustration of frames per second*

4.4.3 PPSP Subroutine 3: Pattern matching

Pattern matching in an MVS is several techniques which enable the localisation of a template pattern in a sample image. A template pattern could be a specific feature with known characteristics, e.g. a facial feature on an image of a face. When trying to find a specific pattern template in a larger image using an MVS, it could be extremely taxing on the machine performing the processing. By restricting the search area, the performance of a software can be greatly improved. Once the software finds a match, a score is given to the match by the software. The score given by the software, is a percentage of how closely the pixels in the template match the pixels of the captured image. Pattern

matching has many applications throughout the industry. There are three main applications of pattern matching, which can be seen in Table 8.

Table 8 ***Applications of Pattern Matching and Descriptions***

Applications of Pattern Matching and Descriptions	
Application	**Description**
Alignment:	Alignment is used to determine the position and orientation of a known object by locating its features within an image. These features are used as points of reference to locate the object.
Inspection:	Inspection, as the name implies, is used to find simple flaws on an object that is being produced and thus grade the object. This is mainly used for quality control.
Gauging:	Gauging is a method of pattern matching that is used to measures the lengths, diameters and angles of an object. If the measurements fall below the accepted tolerance levels, the object is rejected.

4.4.4 Software development: Development of image acquisition software

The image acquisition software was developed in the LabVIEW software suit BPW. The image acquisition software consists of three subroutines, namely the screen capturing subroutine, the background removal subroutine and the pattern matching subroutine.

The screen capturing subroutine captured the primary screen of the computer, which ran the Counter-Strike Global Offensive (CSGO) user interface (UI). CSGO was run at the maximum screen resolution in order to maximise the amount of detail that could be captured.

The process of capturing images could be very taxing on a computer, and thus when capturing the CSGO UI, a region of interest (ROI) was established. The ROI was set to be 920 pixels in height, and 800 pixels in width – the same size as the in-game mini-map, as this was the only element of the screen that needed to be captured to create the player position stencil (PPS). As the ROI typically is set at the origin of the screen (the origin is the top left of the image with the coordinates of (0;0)), an offset was set to capture the in-game mini-map, as well as the in-game round timer. The offset was calculated to be 575 pixels across from the origin and 20 pixels down from the origin. The in-game mini-map that was captured, was then stored in memory and used in the second subroutine with the aid of global variables. The subroutine that was developed can be seen in Figure 22.

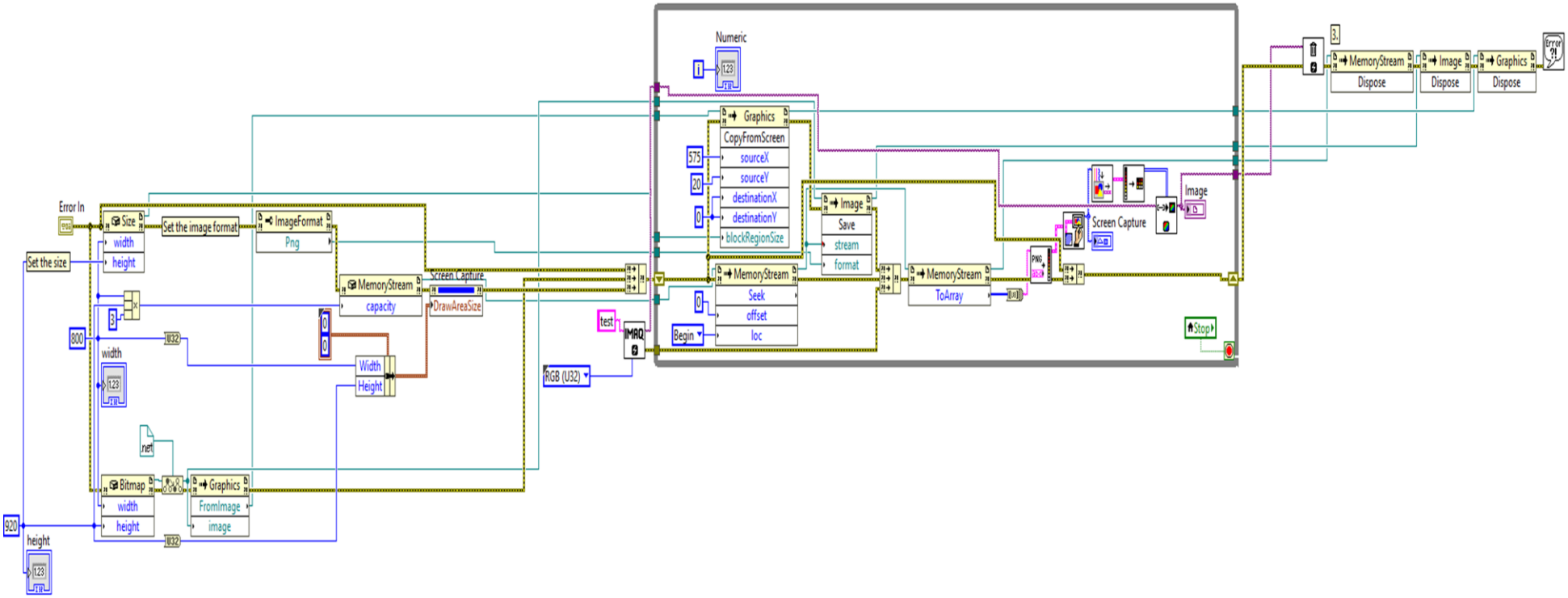


Figure 22 *Screen capturing subroutine of the image acquisition software*

The second of the three subroutines had the purpose of removing the background of the in-game mini-map in order to isolate the players on the map. This process was done using thresholding as the technique to remove all unwanted information from the captured image. Thresholding was achieved by isolating pixels with specific Hue, Saturation or Luma values that range between 0 and 255. Through testing, it was determined that the optimal settings of the thresholding subroutine were as follows; Hue: 0 – 66, Saturation: 3 – 255, Luma: 195 - 255. Once the image that was captured was thresholded, it was converted into a binary image, and transferred to the third and final of the subroutines as a global variable. The subroutine that was developed, can be seen in Figure 23.

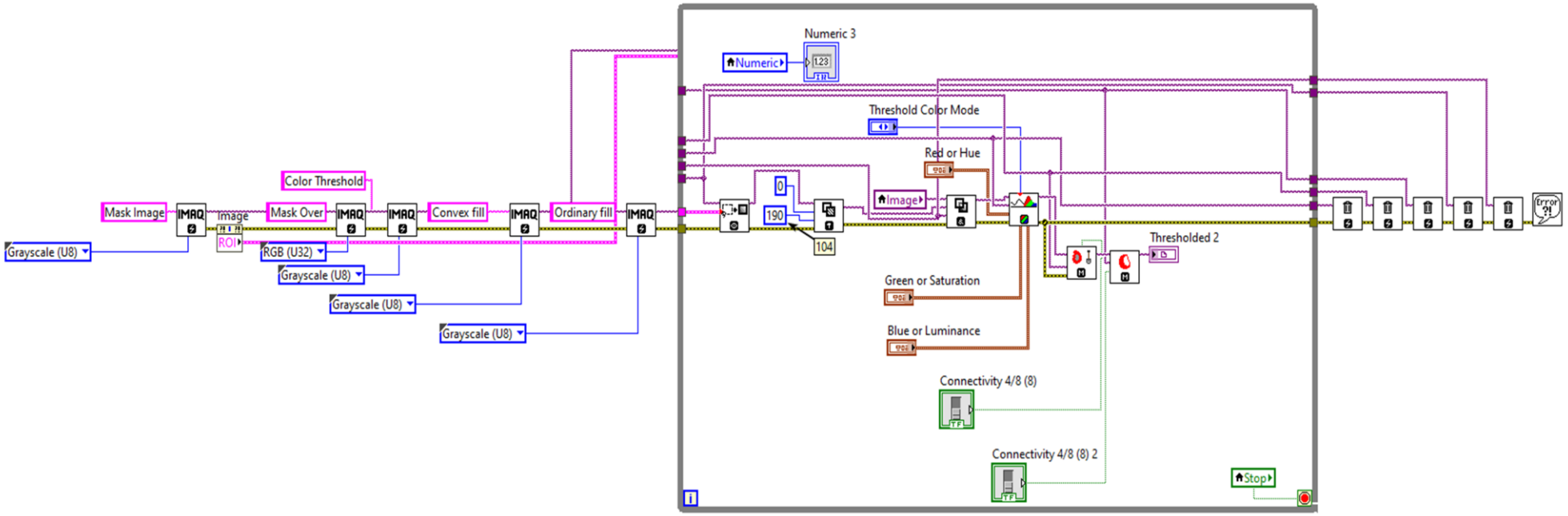


Figure 23 *Background removal subroutine of the image acquisition software*

The third subroutine was developed to perform pattern matching. Pattern matching was not performed on the whole image that was captured in the first subroutine, as this would be very resource intensive. The same ROI that was used in the first subroutine therefore was applied to the pattern matching subroutine. The pattern matching subroutine would be the control of the entire image acquisition software, as it would be the one to determine when an image is stored in memory or not.

As with any pattern matching program, a template is needed to test the images that are captured. The template used, was an image of the round timer, set at one minute and forty-five seconds. Once a match was found between the template and the image that was captured in the ROI, it would store the binary image that was created in the second subroutine in the memory. This binary image that was stored is called the PPS and was created for every round of every match. The subroutine that was developed, can be seen in Figure 24.

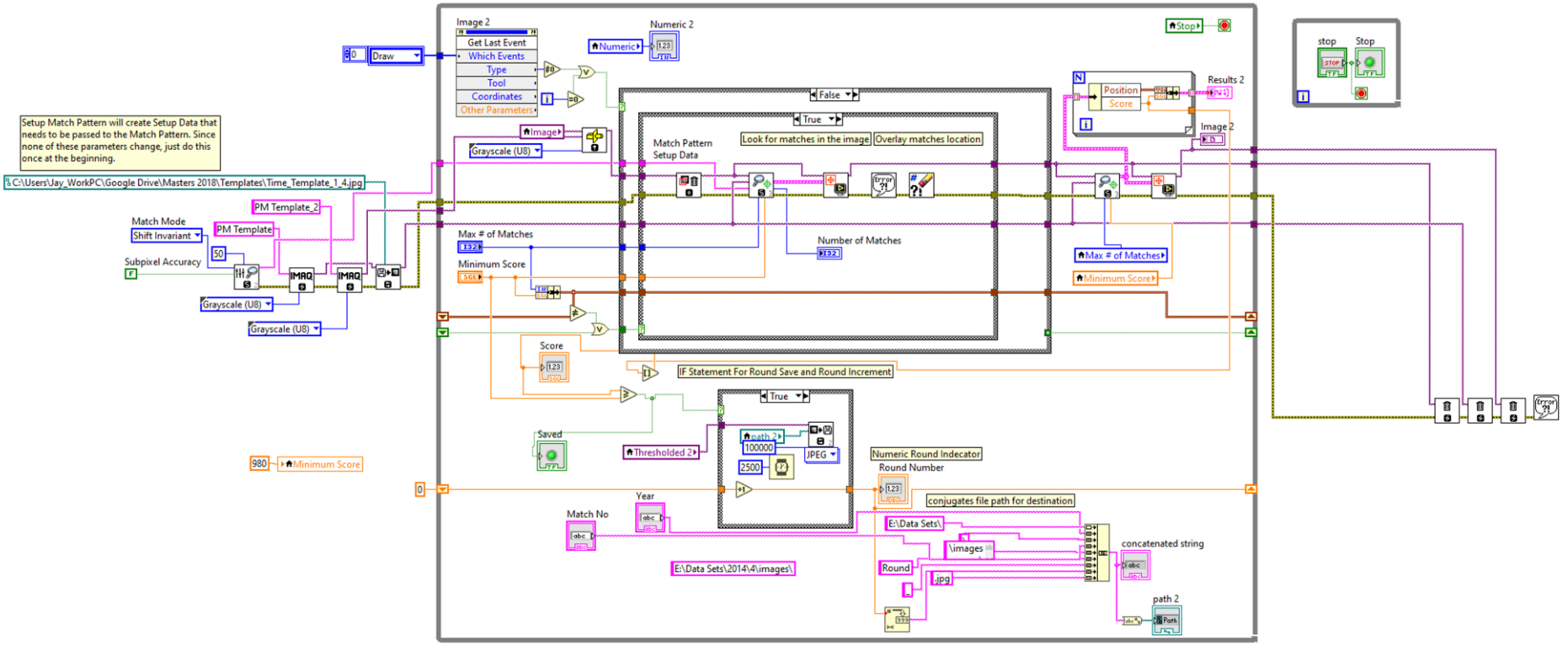


Figure 24 *Pattern matching subroutine of the image acquisition software*

4.4.5 Milestone 2: Image Processing

The LabVIEW 2018 software suite has been the tool used for developing the software of the image processing software of the MVS. The software was named the stencil processor (SP). The SP consisted of three subroutines. A sequence structure was the basis of the SP, as it would ensure that each of the subroutines would run sequentially and produce the player position information (PPI). The three subroutines and their functions were as follows:

- **Subroutine 1:** *PPS Retriever.* The function of this subroutine was to retrieve the player position stencil (PPS) generated by the PPSP and stored in memory by the image acquisition software of the MVS.
- **Subroutine 2:** *PPS Processor.* In order to obtain the player positions from the PPS, the PPS needed to be processed, using morphological processes. The function of this subroutine was to process the PPS and generate a report of the player X and Y co-ordinates within each stencil. The player co-ordinates obtained by SP, were named the player position information (PPI).
- **Subroutine 3:** *PPI Structuring and Storage.* In order to generate the Overall round strategy information, the PPI was captured sequentially to preserve the overall round strategy information (ORSI). Once the ORSI was captured by the SP, it was stored in memory for further processing.

4.4.6 SP Subroutine 1: PPS Retriever

Image processing is an integral part of any machine vision system (MVS). Image processing can be performed by software either as the image is acquired or afterwards. In order to process images after they have been acquired, the images would first have to be retrieved from memory by the software. In order to locate an image in memory, a file path is needed. The use of a path string enables the location of a file in memory. This path string would be the file path needed to locate the image file in memory. The path string would then be used with the Read Bitmap File VI, Read JPEG File VI, or Read PNG File VI that can be found in the graphics and sounds palette in the back panel

window (BPW). In order to view the retrieved image files from memory, one would have to connect the output of the Read VI to the input of a Draw Flattened Pixmap VI, which would then display the image on the FPW. The PPS retriever subroutine was thus used to retrieve PPS captured in the image acquisition software of the MVS. Figure 25 is an example of a simplified subroutine to retrieve and display an image.

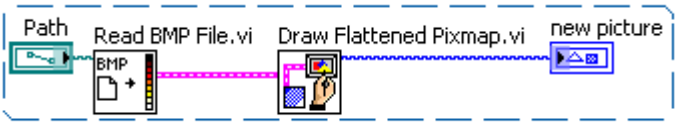


Figure 25 *Basic example of an Image Retrieval VI in LabVIEW*

4.4.7 SP Subroutine 2: PPS Processor

The second subroutine, the PPS Processor, processed the PPS that was retrieved from memory by the first subroutine. Image processing can be performed in various ways within the LabVIEW software suite. The image processing subroutine makes use of blob analysis as its primary form of image processing. Blob analysis is a method of morphological image processing. BLOB is an acronym for Binary Large OBject. A BLOB is a large area of pixels that are touching. Once an image is thresholded, all the pixels in a foreground state will be a solid colour with the pixel binary value of 1. All other pixels of the background are black with a pixel binary value of 0. One could use BLOB analysis to find statistical information such as the number of BLOBs, the co-ordinates of these BLOBs, the size of these BLOBs, or the presence or absence of BLOBs in a specific region within an image [83].

4.4.8 SP Subroutine 3: PPI Structuring and Storage

There are various ways in which to achieve this When storing information to memory in the LabVIEW software suite. The general manner of storing information is in a spreadsheet. The PPI

structuring and storage subroutine makes use of a Write To Spreadsheet File VI to store information. The Write To Spreadsheet File VI creates a file with various data values, separated by tab characters. Figure 26 depicts a Write To Spreadsheet File VI.

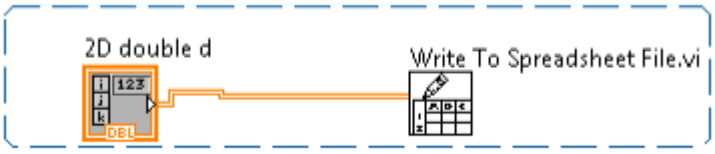


Figure 26 *Basic example of Write To Spreadsheet VI in LabVIEW*

4.4.9 Software development: Development of Image Processing software

The image processing software of the MVS was developed in the LabVIEW 2018 software suite BPW, and divided into three subroutines, viz. the PPS retriever, the PPS processor and finally the PPI structuring and storage.

The PPS retriever subroutine was developed to ensure that the PPS of each round of each match would be opened sequentially to be retrieved and then processed. The process of retrieving the correct PPS was done by selecting a specific year and match, to begin with. These parameters could be set on the FPW of the SP software. These parameters can be seen in Figure 27.

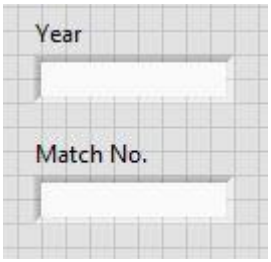


Figure 27 *User inputs on the FPW of the image processing software*

Once the year and match number parameters were set, the software was developed to count the number of rounds in the “match” folder that was selected. The number of rounds would serve to be the number of times the processing and storage subroutine would run. The number of rounds would decrease until every round’s PPS was processed. Once all the PPSs were processed, it was coded

that the image processing program would then increment the match number, and proceed to begin the process of counting the number of rounds once again and continue to processes each PPS of each round. The process of retrieving every PPS was done for every match within the selected year set at the start.

Once all the matches within the specified year were processed, the programme was coded to increment the year number and start the process of counting each match and processing each of the matches once again. This would be repeated up and until each PPS of every round, of every match, of every year was processed and stored into memory as the PPI. The PPS retrieval subroutine can be seen in Figure 28.

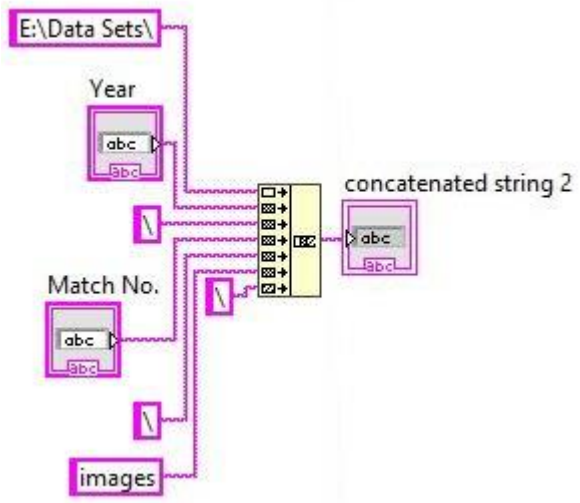


Figure 28 *PPS retrieval subroutine of the image processing software*

Once the PPS was retrieved from memory, the PPS was then processed, using two morphological VIs found in the LabVIEW 2018 software suit. The two morphological VIs were the IMAQ Separation VI and the IMAQ Particle Analysis VI. Figure 29 A. is the Separation VI, and Figure 29 B. is the IMAQ Particle Analysis VI.

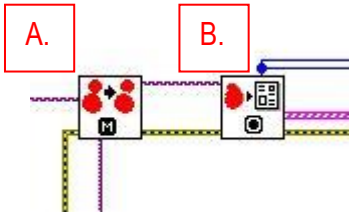


Figure 29 *A. The IMAQ separation VI alongside B. the IMAQ particle analysis VI*

The function of the separation process VI was to separate any particles found within the PPS that were too close to one another. With the particles of the PPS separated, the output of the IMAQ Separation VI was then wired into the input of the IMAQ Particle Analysis VI. The output of the IMAQ Particle Analysis VI was then presented in a table. This table included the X and Y co-ordinates of the particles as well as their sizes. The table produced by the particle report was found on the FPW of the SP software and can be seen in Figure 30.

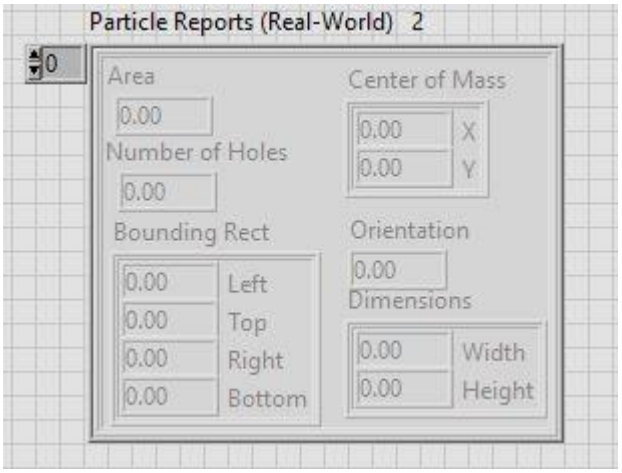


Figure 30 *The IMAQ Particle Analysis VI output found on the FPW*

Using the IMAQ Particle Analysis VI, small particles were filtered out, as a low pass threshold was established within this VI. By filtering out all the small particles, only the players within the PPS would be detected, and their respective X and Y co-ordinates would be recorded in a table. With the player X and Y co-ordinates now extracted from the PPS, it was essential to store these coordinates in the same order in which they were captured.

The sequential storage of these co-ordinates was achieved by creating a self-indexing array, which was the primary feature of the SPs software subroutine. In order to create a self-indexing array, the Insert into Array VI was used. The output of the Insert into Array VI was then fed into the input of the same Insert into Array VI. By doing so, the new player co-ordinates would be stored after the prior co-ordinates, thus maintaining the sequence in which the PPS's are processed. The array containing each player's co-ordinates for each round was then stored to memory using the Write to Delimited Spreadsheet VI. The spreadsheet that was stored in memory by the spreadsheet was then called the player position information (PPI). The complete BPW of the SP can be seen in Figure 31.

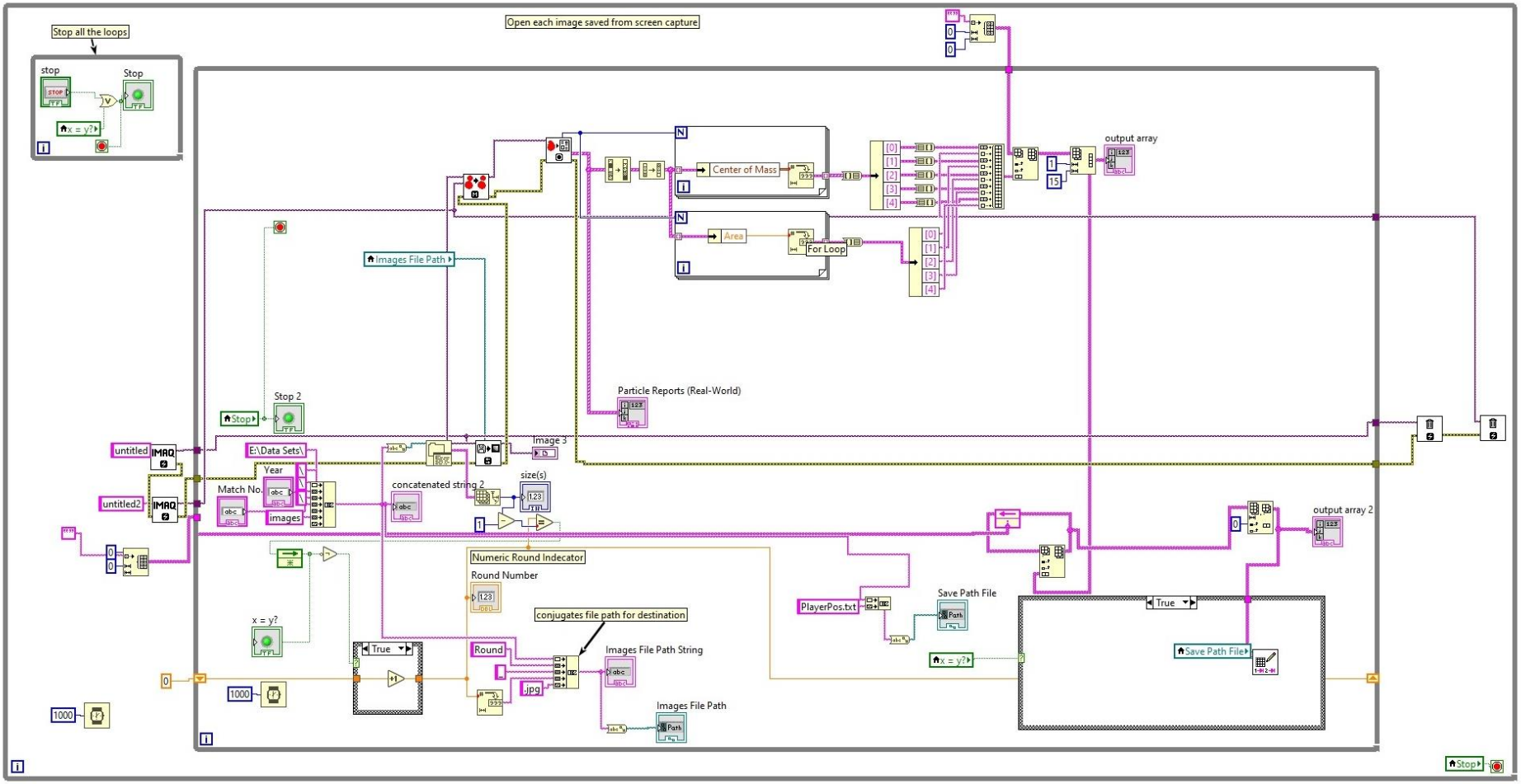


Figure 31 *The complete BPW of the SP*

4.5 Milestone 3: Information Concatenation

LabVIEW 2018 Software suite has been the tool for the development of the three software units of the information processing system (IPS). The LabVIEW software suite is a block-orientated programming environment that is best characterised by its drag and drop style of programming, as mentioned before. The development of the first software was necessary to concatenate the player position information (PPI) with the miscellaneous information obtained through the CSGO Demo Manager software. The first software of the IPS was the Information Concatenator (IC). The product of the IC would be the overall round strategy information (ORSI). The IC consisted of three subroutines. These three subroutines were as follows:

- **Subroutine 1:** *PPI and Additional Info Retriever.* The primary function of the PPI and Additional Info Retriever was to retrieve the PPI that was generated from the stencil processor (SP), as well as retrieve the miscellaneous round information generated by the Counter-Strike Demo Manager (Additional Info).
- **Subroutine 2:** *Spreadsheet Manipulator.* The primary function of the Spreadsheet Manipulator was to remove all unwanted information from the additional info that was retrieved by the first subroutine. Unwanted information was removed by the subroutine, using the string manipulation capabilities of the LabVIEW software suite.
- **Subroutine 3:** *Concatenator.* The Concatenator subroutine of the IC achieved the concatenation of the information.

4.5.1 IC Subroutine 1: PPI and Additional Round Information Retriever

The retrieval of stored information is often necessary for an IPS. The retrieval of information from memory is similar to the retrieval of images from memory in an MVS. Firstly, a file path is needed in order to retrieve a file from memory. The use of a path string enabled the locating of a file in memory. This path string is connected to a Read Delimited Spreadsheet VI by connecting wires. The

Read Delimited Spreadsheet VI reads the information of a file at the specified file location, beginning at a specified character offset, and converts the information it reads into a two-dimensional, double-precision array of numbers, strings or integers. The figure below is a representation of a Read Delimited Spreadsheet VI. Figure 32 A. represents a text file that contains two columns of data that is delimited by a comma. Figure 32 B. represents the BPW of the Read Delimited Spreadsheet VI. Figure 32 C. represents the FPW displaying the information that was retrieved by the Read Delimited Spreadsheet VI in the BPW.

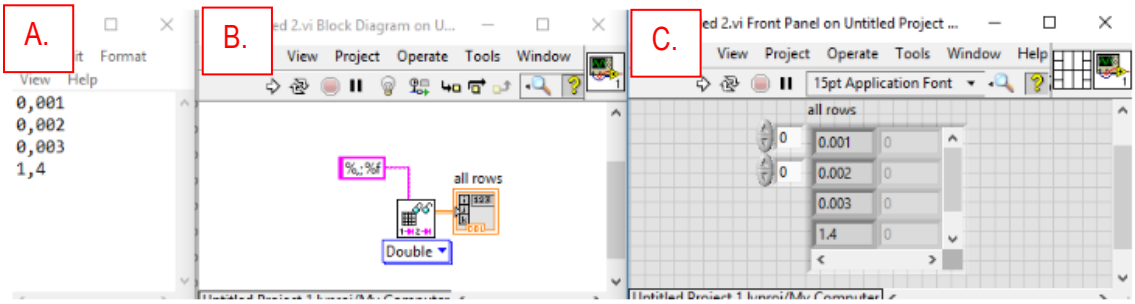


Figure 32 *A. A text file created with values, delimited by commas. B. The BPW with the VI responsible for retrieving the text file. C. The output of the retrieved file on the FPW.*

4.5.2 IC Subroutine 2: Spreadsheet Manipulator

When developing any system, communicating with instruments and storing information are commonplace. The information is often stored as strings by software. A string is a sequence of displayable or non-displayable ASCII characters [84]. Applications of strings vary from creating simple text messages to controlling instruments connected to a system or storage of information collected by a system. Tables are the most common form of storing strings containing information. Tables allow the information to be indexed easily. Figure 33 represents an example of a table containing strings of information.

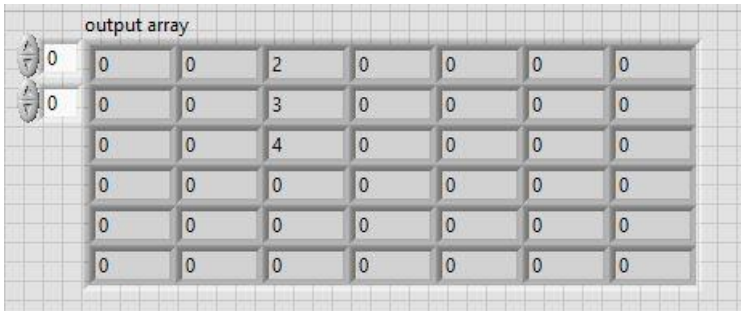


Figure 33 *Table containing Strings on the FPW of LabVIEW.*

When using a table in LabVIEW, the software can perform various functions on the information within the table. These functions are: deleting information from an index; finding and retrieving information from a specific index; searching for, retrieving or replacing characters within a string or a substring in a specified index; changing all the text within a string to upper or lower case; rotating or reversing a string in a specified index or concatenating two or more strings together. These functions can be found in the String Palette in the BPW. Figure 34 illustrates all the string functions found in the Strings Function Pallet.

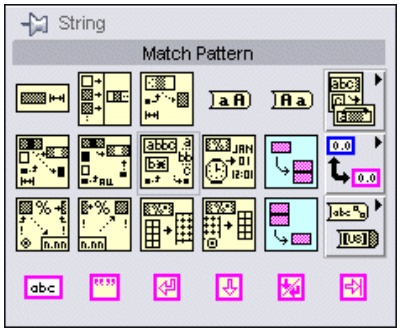


Figure 34 *String function pallet found in the LabVIEW BPW.*

4.5.3 IC Subroutine 3: Concatenator

When developing systems that produce multiple information spreadsheets, these spreadsheets need to be concatenated in order to be analysed. The indexing feature of arrays within the LabVIEW software suite made the concatenation of various spreadsheets possible. Once the concatenator

subroutine completed concatenation of the PPI and additional information arrays, the new array was referred to as the ORSI. The ORSI was stored to memory by the Write to Spreadsheet File VI. The Write To Spreadsheet File VI creates a file with the concatenated information, separated by tab characters.

4.5.4 IC Software development: Development of Information Concatenator software

The information concatenator (IC) software of the IPS was developed using the LabVIEW 2018 software suite BPW. The IC software was divided into three subroutines, viz. the PPI and additional info retriever, the spreadsheet manipulator and finally the concatenator.

Both the PPI and additional info retriever subroutine were developed to ensure the correct PPI was concatenated to the correct additional info. The process of retrieving the PPI and additional info was developed correspondingly to that of the PPS retriever subroutine of the SP. The process of retrieving the correct PPI and additional info was done by selecting a specific year and match, to begin with. These parameters could be set on the FPW of the IC software. These parameters can be seen in Figure 35.

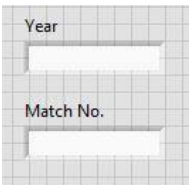


Figure 35 *User inputs on the FPW of the image processing software*

Once the year and match numbers were set, the software was developed to count the number of rounds selected in the “match” folder. The number of rounds would serve to be the number of times the Spreadsheet Manipulator subroutine would run.

The Spreadsheet manipulator was developed to remove all unwanted information found in the additional info a file that was retrieved from memory before concatenating the additional info with the PPI. The unnecessary information was all the counter-terrorist and overtime rounds that were played by Team Fnatic. The index array VI was used to locatedinstances of “Fnatic” or “fnatic” in the terrorist column within the additional info file. The index array VI portion of the subroutine can be seen in Figure 36. The index array VI portions of the subroutine were bound by a for loop with a count of 15. By looping this portion of the subroutine 15 times, it would ensure that only the rounds played within regulation would be captured and concatenated with the PPI. The for loop can be seen in Figure 36 A.

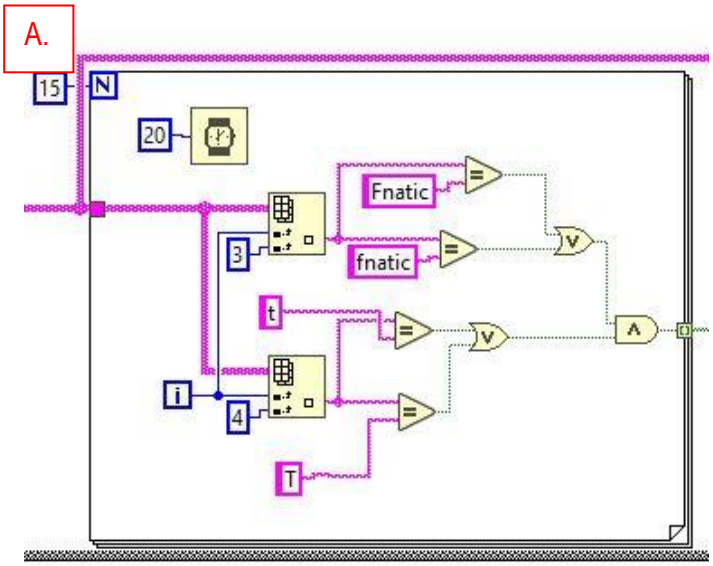


Figure 36 *Subroutine used to find and remove all unnecessary information from the additional info spreadsheet.*

The PPI and the additional info arrays were concatenated once all the unnecessary information of the additional info array was removed. The inputs of the Insert Into Array VI were set to 16 as the column number, with the delimited information of the PPI as the information input.

By doing so, all the additional information would populate the first 16 columns within the new array, with the remaining 11 columns populated by the PPI, and in doing so concatenating the two information arrays. The Appended.txt file is stored back into memory. The complete BPW of the IC software can be found in Figure 37.

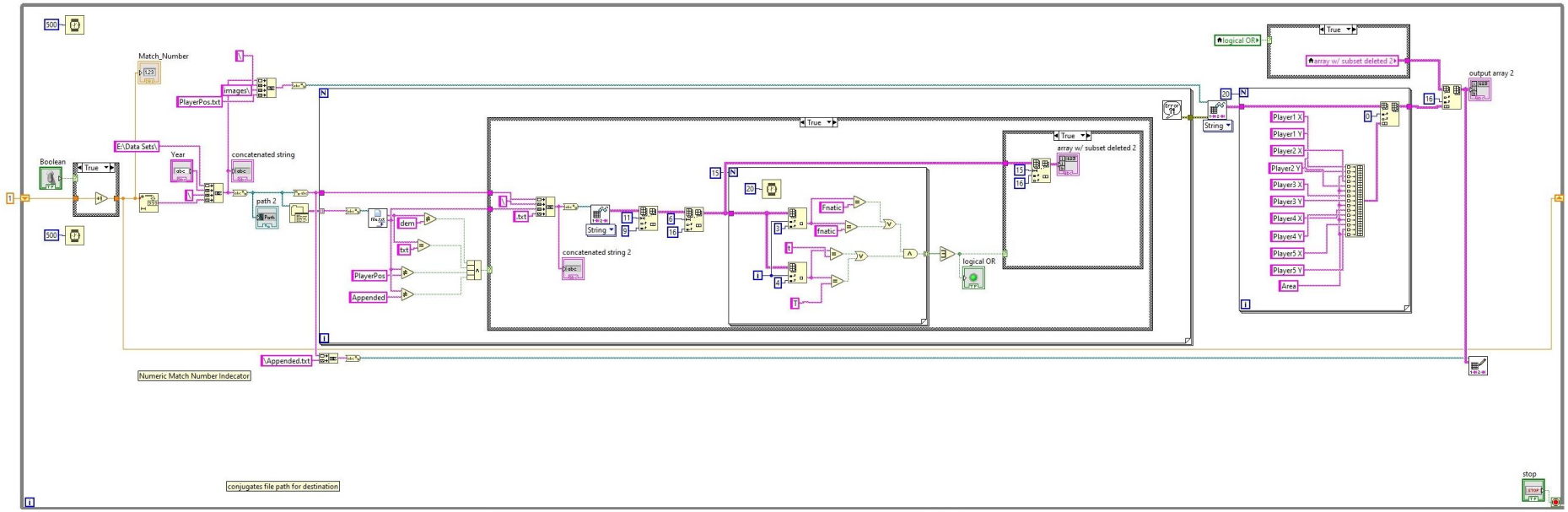


Figure 37 *The complete BPW of the IC software*

Once the Appended.txt was stored in memory, each of the Appended.txt files was imported to Microsoft Excel, thus creating the ORSI. Before the ORSI was processed, the player positions were encoded to simplify the processing by the neural net. The Microsoft Excel software provided the macro functionality where the script to encode the player positions. The written script was as follows:

```
Option Explicit

Public Function PtInPoly(Xcoord As Double, Ycoord As Double, Polygon As Variant) As Variant
  Dim x As Long, m As Double, b As Double, Poly As Variant
  Dim LB1 As Long, LB2 As Long, UB1 As Long, UB2 As Long, NumSidesCrossed As Long
  Poly = Polygon
  If Not (Poly(LBound(Poly), 1) = Poly(UBound(Poly), 1) And _
        Poly(LBound(Poly), 2) = Poly(UBound(Poly), 2)) Then
    If TypeOf Application.Caller Is Range Then
      PtInPoly = "#UnclosedPolygon!"
    Else
      Err.Raise 998, , "Polygon Does Not Close!"
    End If
    Exit Function
  ElseIf UBound(Poly, 2) - LBound(Poly, 2) <> 1 Then
    If TypeOf Application.Caller Is Range Then
      PtInPoly = "#WrongNumberOfCoordinates!"
    Else
      Err.Raise 999, , "Array Has Wrong Number Of Coordinates!"
    End If
    Exit Function
  End If
  For x = LBound(Poly) To UBound(Poly) - 1
    If Poly(x, 1) > Xcoord Xor Poly(x + 1, 1) > Xcoord Then
      m = (Poly(x + 1, 2) - Poly(x, 2)) / (Poly(x + 1, 1) - Poly(x, 1))
      b = (Poly(x, 2) * Poly(x + 1, 1) - Poly(x, 1) * Poly(x + 1, 2)) / (Poly(x + 1, 1) - Poly(x, 1))
      If m * Xcoord + b > Ycoord Then NumSidesCrossed = NumSidesCrossed + 1
    End If
  Next
  PtInPoly = CBool(NumSidesCrossed Mod 2)
End Function
```

Figure 38 *Macros script written in Microsoft Excel to locate the player coordinates within the pre-allocated areas*

The script above tested the player positions against the pre-allocated areas on the map. Figure 39 A illustrates the pre-allocated areas used in the testing of the player co-ordinates with Figure 39 B illustrating the areas of the map in a graphical format.

A.

b-aps	
X	Y
135	238
602	238
602	284
522	284
488	318
318	318
318	360
280	360
280	296
135	296
135	238

lowers	
x	y
318	320
486	320
486	351
350	351
350	496
318	496
318	320

cat	
x	y
350	394
385	394
502	461
502	496
350	496
350	394

mid	
x	y
522	284
602	284
602	580
434	580
434	555
318	555
318	496
507	496
507	464
522	464
522	284

a site	
x	y
343	621
545	621
545	721
522	721
522	745
500	745
500	888
343	888
343	621

connector	
x	y
343	555
434	555
434	621
343	621
343	555

T Ramp	
x	y
545	645
594	645
594	600
620	600
620	584
780	584
780	638
657	638
657	710
545	710
545	645

palace	
x	y
657	638
780	638
780	672
740	672
740	836
500	836
500	745
657	745
657	638

spawn	
x	y
601	238
780	238
780	584
689	584
689	377
662	347
601	347
601	238

B.

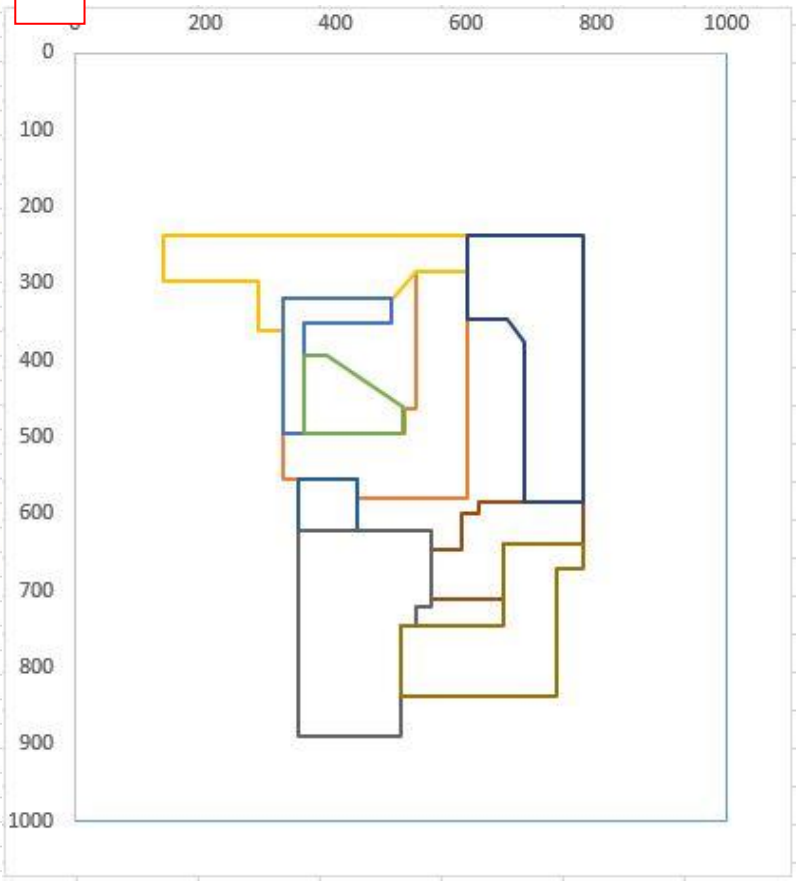


Figure 39 *A. Pre-allocated areas. B. Graphical representation of the pre-allocated areas*

4.6 Milestone 4: Information Pre-processing

The LabVIEW 2018 Software suite has been the tool for the development of the second and third software units of the IPS. The development of both software units was necessary in order to process the overall round strategy information (ORSI) in the correct format to train an artificial neural net (ANN). Information pre-processor (IPp) was the name given to both software units. The first IPp was responsible for the retrieval of the OSRI and the manipulation of the information in the ORSI into the correct format needed to train an ANN. The software of the first IPp consisted of one routine, divided into three parts as follows:

- **Part 1:** *File Retrieval.* The primary function of the first part was to retrieve the ORSI generated by the Information Concatenator (IC). The retrieval of the ORSI, by this software, is similar to that of the PPI and additional info retriever subroutine of the IC software.
- **Part 2:** *ORSI Re-arranging.* The primary function of the second part was to ensure that the ORSI is in the correct format to train an ANN. The second part of the routine placed the encoded player co-ordinates of the next round with the current round, and divided the ORSI arrays. This was necessary, as the previous round's information determined the player positions of the next round. For example, if one were to train a neural net on the strategy of round three, the ANN would need to know what the player positions were in round two.
- **Part 3:** *File Storage.* The primary function of the third and last part of the routine was to store the arrays created in Part 2 as *round number selected*.txt file and the *round number selected* – targets.txt files.

The second software unit of the Information Pre-processor (IPp) consisted of one routine. The responsibility of the second IPp software was to join all the respective *round number selected*.txt and *round number selected* –targ.txt files, as well as all previous rounds together. If one were to train an ANN on the strategy for round five, all previous rounds need to be within the data set for training.

Similarly, if you were to train an ANN on a strategy for round ten, all previous rounds need to be within that data set for training. The second software of the IPp consisted of one routine, divided into three parts. These parts were as follows:

- **Part 1:** *File Retrieval.* The primary function of the first part was to retrieve the "*round number selected"*.txt and *"round number selected" – targ*.txt generated by the First IPp software. The retrieval of the "*round number selected - round and targ.txt"* file is similar to that of the PPI, and Additional Info Retriever subroutine found within the IC Software.
- **Part 2:** *Round Arrangement.* The first part of the purpose of this routine was to retrieve the data and target files in the memory of a specific round, the user-specified round on the UI in the FPW of the software. With the round number specified, the data and target files of that round would be retrieved. Upon retrieval, they would be conducted together by this part of the routine.
- **Part 3: File Storage.** The primary function of the second and last part of the routine was to store the newly created file. The newly created file would be the ORSI2. Subsequently, each round would have an ORSI2 file.

4.6.1 IPp Software 1, Part 1: File Retrieval

For brevity sake, the process of retrieving files remains largely similar as mentioned in the PPI. The additional info retriever subroutine and the retrieval of information from memory are similar to the retrieval of images from memory in an MVS as mentioned above. Firstly, a file path is needed by the software to retrieve a file from memory. The use of a path string VI enabled the locating of a file in memory. The path string is connected to a Read Delimited Spreadsheet VI by connecting wires.

4.6.2 IPp Software 1, Part 2: ORSI Re-arranging

The process of re-arranging the ORSI remains similar to the spreadsheet manipulator subroutine of the IC software. A more in-depth explanation of spreadsheet and string manipulation can be found above in Subroutine 2 of the IC.

4.6.3 IPp Software 1, Part 3: File Storage

One can refer back to Subroutine 3 of the SP, as it discusses the storage of spreadsheet files at length.

4.6.4 IPp Software 2, Part 1: File Retrieval

The process of retrieving the files in the second software remained the same as with the first software. For brevity sake, referred to 1.6.1 IPp Software 1, Part 1: File Retrieval.

4.6.5 IPp Software 2, Part 2: Round Arrangement

The process of round arrangement remains similar to the spreadsheet manipulator subroutine of the IC software. A more in-depth explanation of spreadsheet and string manipulation can be found above in Subroutine 2 of the IC.

4.6.6 IPp Software 2, Part 3: File Storage

One can refer back to Subroutine 3 of the SP, as it discusses the storage of spreadsheet files at length.

4.6.7 Software development: Development of Information Pre-processor software 1

Before any programming could be done, a number of files needed to be created in memory. These files were the #.txt file and the # - targ.txt. The # represents the round number the user would input

in the front panel window of the LabVIEW software. Fifteen of these files were created, one for each round within a half of a Counter-Strike Global Offensive match.

The first Information Pre-processing (IPp) software was developed in the LabVIEW 2018 software suite BPW. The first software consisted of one routine that was divided into three parts, with each part being responsible for executing a specific function. As with every other LabVIEW program, the programming of the first IPp software began with the introduction of a While Loop VI on the back panel window (BPW) that is controlled by a stop button on the front panel window (FPW). By controlling the while loop with a stop button, the user had control over the number of times the software would run.

Inside the first IPp software While Loop VI, a file retrieval method was programmed into the routine. The file retrieval method made use of path string VI that was built to retrieve specific files from memory. The process of building a path string was done twice, as two files needed to be retrieved from memory. The exact pair of these files would be determined by the user with the string input VI on the FPW labelled round number. The number inserted by the user would be wired into a Build String VI that would create the path string. Once the user enters a number, the first IPp software would retrieve the specified round numbers pair of files, using the path string created. A second file retrieval part was programmed into the software with the goal of retrieving the ORSI created by the SP software, seen in Figure 40 Part 1.

Once all the necessary files were retrieved from memory, the information retrieved would be processed by the ORSI re-arranging part of of the program. The ORSI re-arranging part of the software would firstly remove the encoded player position information from the additional info. The splitting of the information was achieved through the use of the indexing capabilities of arrays within

LabVIEW. ORSI re-arranged was achieved by subtracting the indexed row of additional info to that of the PPI, seen in Figure 40 Part 2.

Once the information was separated, the new Additional info was stored as ""*round number selected*".txt", and the PPI was stored as ""*round number selected*"-targ.txt". The storage of the two files was made possible with the use of the Write to Delimited Spreadsheet VI provided by LabVIEW. This can be seen in Figure 40 Part 3.

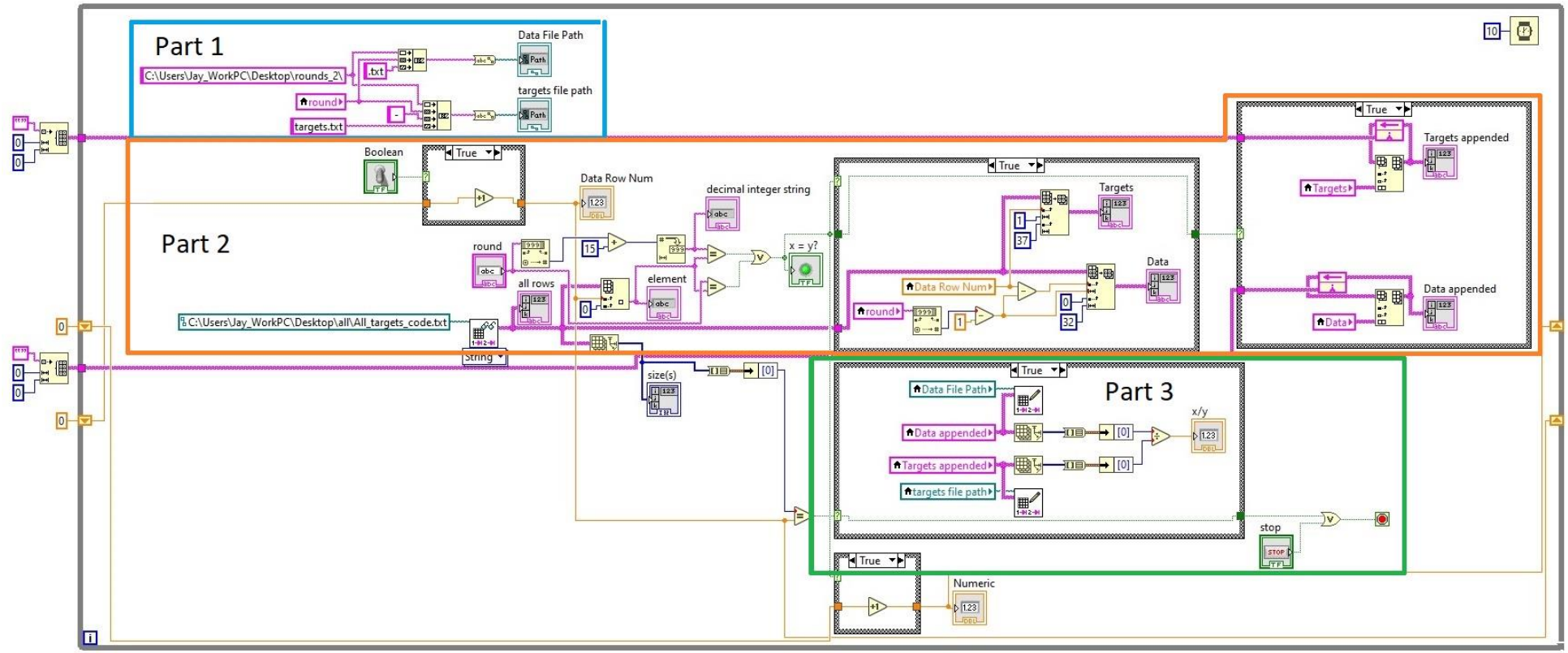


Figure 40 *BPW of the first IPp.*

4.6.8 Software development: Development of Information Pre-processor software (IPp) 2

The second IPp) was developed in the LabVIEW 2018 software suite BPW. It consisted of one routine divided up into three parts, with each part being responsible for executing a specific function. The second IPp also consisted of only one while loop on the back panel window (BPW), controlled by a stop button on the front panel window (FPW).

Much like the first IPp while loop, a file retrieval method was programmed into the routine. A build string VI was used to construct the file path string to retrieve the specified files from memory. The build string VI made use of a text input on the FPW to select the correct file. The user specified the round number before any processing would take place. This can be seen in Figure 41, *Part 1.1, 1.2, 1.3*.

Once the files were retrieved, the manner in which they were processed, was that all previous rounds would be stored, prior to the round specified by the user. If the user would therefore select "5" as the input the "#.txt" and "# – trag.txt" files of all round 1s, 2s, 3s and 4s would be concatenated to be stored as the information set for "5 – round and targ.txt". By doing so, all historical information leading up to round 5 would be captured in one file. The round arrangement was performed for every single round up round 15, excluding round 1, as there is no historical information for round one. The round arrangement part of the routine can be seen in *Part 2* in Figure 41.

Once a round had been processed, the new array that was created would then be stored in memory with the name "# -round and targ.txt". The "# -round and targ.txt" file is referred to as the ORSI2. The method of storing these files was the same as with the first IPp. The method of storing the "# -round and targ.txt" file can be seen in *Part 3* in Figure 41.

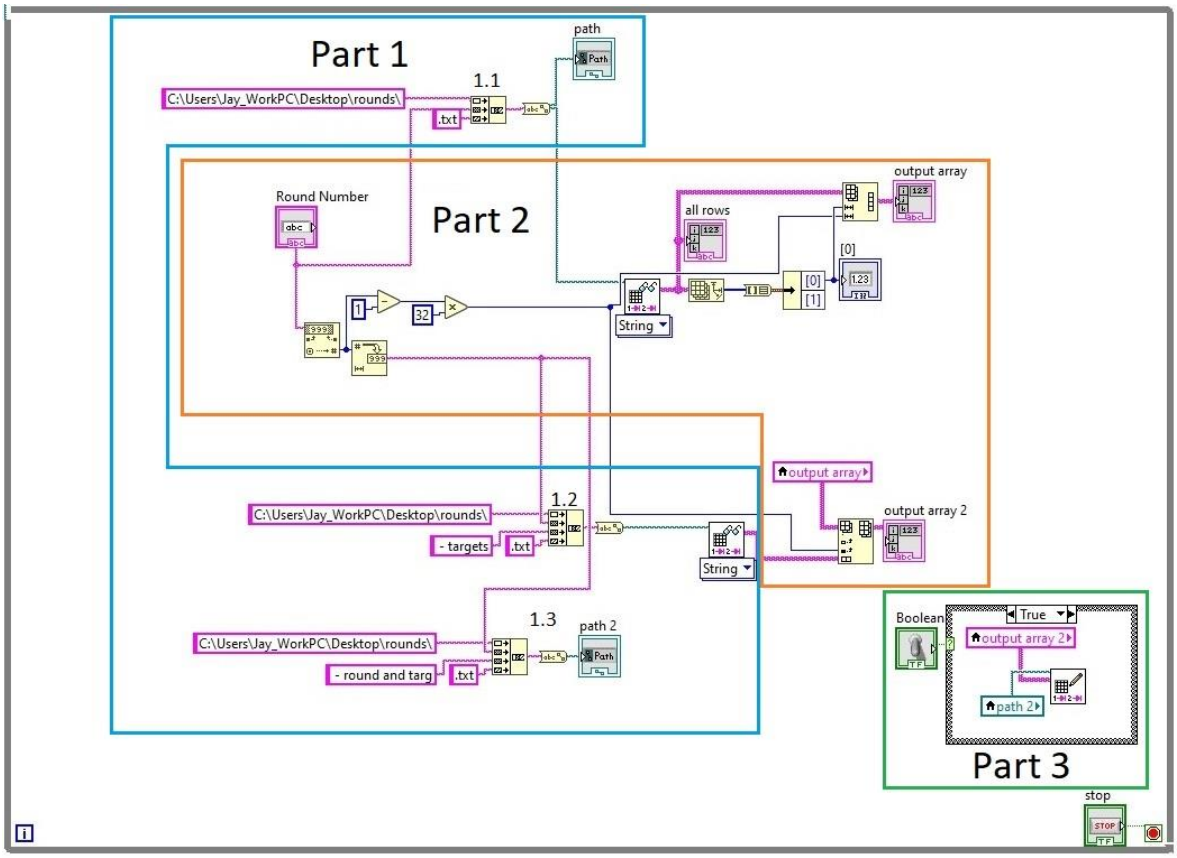


Figure 41 *BPW of the second IPp software*

4.7 Discussion

The purpose of the MVS was to acquire accurate images of the in-game mini-map found on the CSGO UI, and to store them as binary images. The binary images, captured by the MVS, were then successfully processed in order to capture the player positions across the map. This was achieved by selecting the correct frame rate to capture the images, the resolution of the images captured, and the colour threshold needed to extract the terrorist team's players from the in-game mini-map. The first and second milestones of Phase two of the study were achieved by the development of the PPSP and SP software of the MVS software suite. The third and fourth milestones of Phase two of the study were achieved by developing the software to perform the processing of the information obtained in Milestone two. The software developed, was the IC and the IPp. Once the PPI was successfully concatenated to the miscellaneous information obtained from the CSGO Demo

Manager and correctly pre-processed to be used to train an ANN, the development and training of each rounds automatic strategy predictor (ASP) was undertaken in Phase 2, Milestone 5 as laid out in Chapter 5.

Chapter 5

Development of each of the Automated Strategy Predictors

5.1 Introduction

Chapter 5 describes Phase 2, Milestone 5 of this study and provides details of the development of each of the Automated Strategy Predictors (ASPs). The purpose of the ASP is to predict the strategy that would be used by Team Fnatic on the terrorist half of a match for every proceeding round to the current round. The ASP makes use of the overall round strategy 2 (ORSI2) file generated by the second IPp to train an artificial neural network (ANN). Each round will have an ANN trained, using its corresponding ORSI2 file, thus achieving Phase 2, Milestone 5. Figure 42 provides a conceptual flow diagram of the overall study, highlighting Phase 2, Milestone 5, which involved the development of the 15 ANNs.

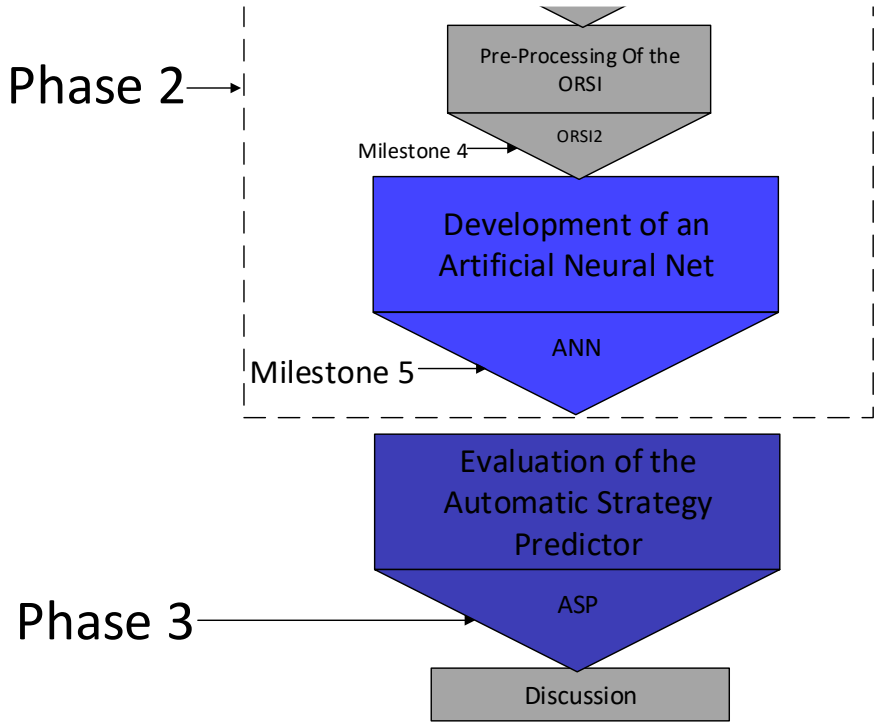


Figure 42 *Conceptual framework flow diagram, highlighting the development and evaluation of the artificial neural net*

5.2 Phase 2: Preparation for the Development of the ANN

Before the completion of Phase 2 and achieving Milestone 5, several steps were performed before the development of the ANNs. These steps included the importing and preparation of each round's ORSI2 file into the MATLAB programming environment for the development and training of the ANN. Phase 3 involved the implementation and evaluation of the ANN in the Simulink environment. A logical flow diagram of Phase 2, Milestone 5 and Phase 3 can be seen in Figure 43.

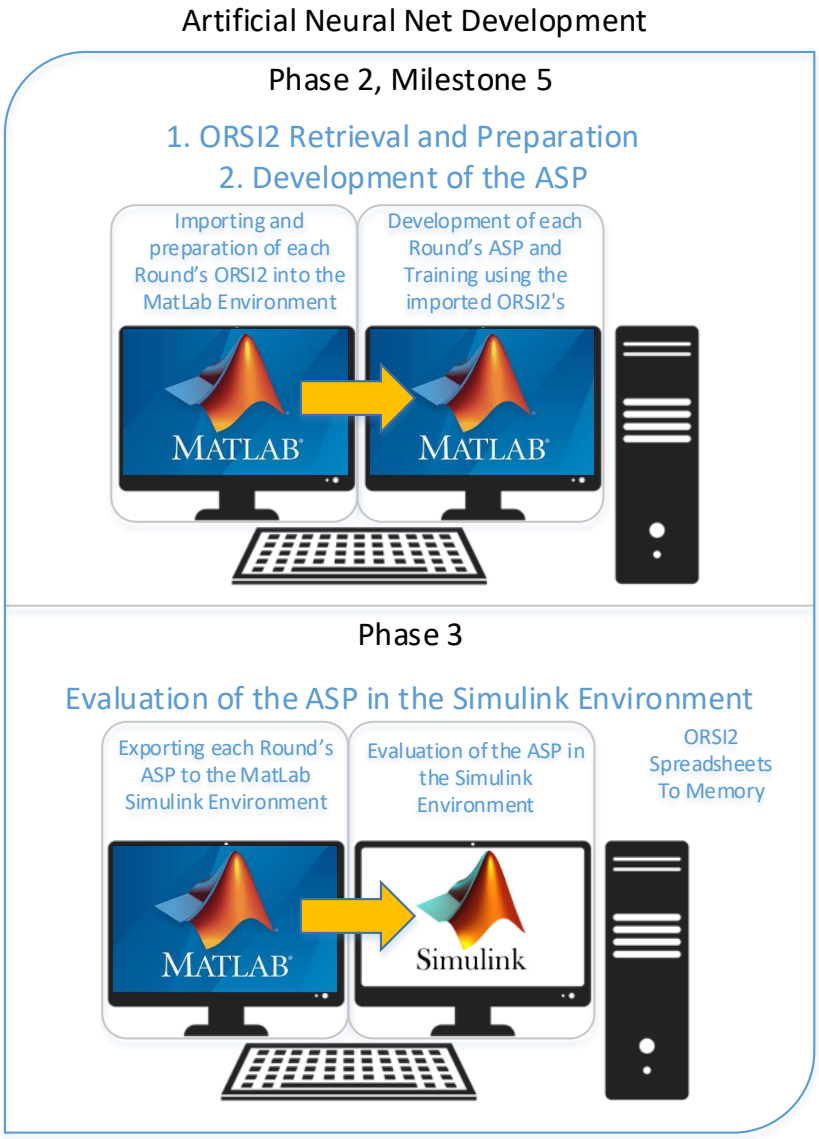


Figure 43 *Logical flow diagram of the development and evaluation of the artificial neural nets*

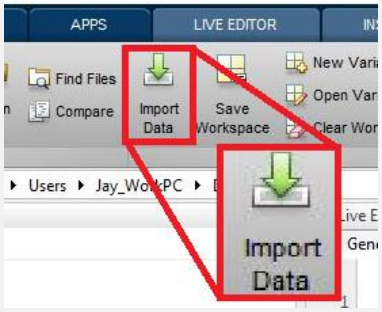
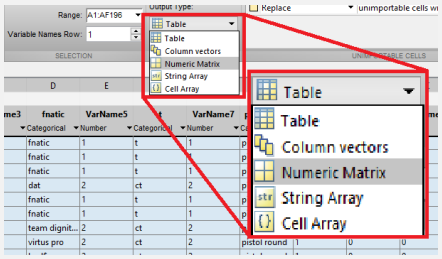
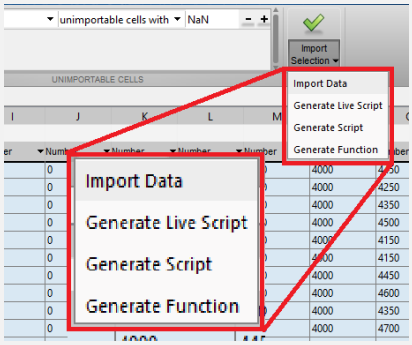
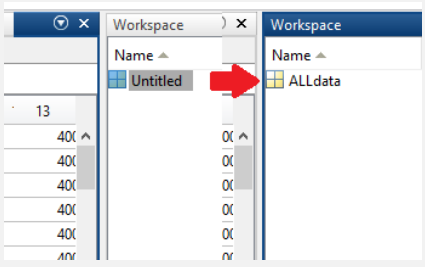
5.2.1 Software Development Environment: MatLab

The software chosen for the development and implementation of the artificial neural net (ANN) were MATLAB and MATLAB Simulink. MATLAB is a programming platform that is primarily used by engineers and scientists. The basis of MATLAB is its programming language. The MATLAB programming language is matrix-based, allowing natural computational mathematics. The utilisation of the MATLAB software varies in a number of ways, from analysing data, creating models and simulating applications, and developing algorithms to developing deep-learning and machine-learning applications. MATLAB contains a neural net development toolbox that was used to develop and train the ANNs that made up the Automatic Strategy Predictor (ASP).

5.2.2 Importation of each ORSI2 file into the MATLAB workspace

Before the ANNs of the ASP could be trained, each round's ORSI2 file containing the information needed to be imported into the MATLAB workspace. Importing the information was a two-step process, as described in Table 9. The data of each round were imported separately. Upon the importation of the ORSI2 of a round into the MATLAB workspace, the ORSI2 was manipulated to provide compatible information to train the ANN, using the ANN toolbox.

Table 9 *Steps Needed to import the ORSI2 into the MatLab Workspace*

Description of Step 1	Software window
One would begin the process of importing the ORSI2 file into MATLAB by clicking on the import data button found on the home ribbon of the MATLAB UI and selecting the ORSI2 file in memory.	
Description of Step 2	**Software window**
Once the Import window has opened, a drop-down menu is found in the import tab of the ribbon. By selecting the Numeric Matrix from the drop-down, all text will be ignored and converted to a NaN value.	
Description of Step 3	**Software window**
Once the information is converted to a numeric matrix, the Import Selection drop-down is found on the far right section of the Import tab of the ribbon. From the drop-down, Import Data was selected and thus imported the ORSI2 file as a numeric matrix to the MATLAB workspace.	
Description of Sep 4	**Software window**
Once imported, the matrix was renamed from "Untitled" to "ALLdata"	

5.2.3 Pre-processing of each ORSI2 file for the training of the ANN

In order to develop and train each of the ANNs, the overall round strategy information 2 (ORSI2) file was imported into the MATLAB Workspace and had to be converted to the correct format. The format of the information was determined by the type of training selected to train the ANN. This study utilised supervised training.

Supervised training is the method of training an ANN where both the input and output information is provided to the ANN as specified by the ANN toolbox found in MATLAB. The inputs of the ANN was the ORSI2 file that was generated by the second IPp. The targets of the ANN were the encoded player positions, generated by the second IPp. By performing supervised training, the ANN would "learn" which targets (encoded player positions) correlate with the inputs (ORSI2) it was provided with. In turn. The ANN would thus be able to predict the positions the players would use in the next round of the match. A conceptual diagram of the information format needed to train the ANN is found in Table 10.

Table 10 *Example of the data format needed by the ANN Toolbox*

ORSI2				
ValueA1	ValueA2	ValueA3	ValueA4	...
ValueB1	ValueB2	ValueB3	ValueB4	...
ValueC1	ValueC2	ValueC3	ValueC4	...
...	...	...	...	...

=

Encoded Player Positions				
ValueA1	ValueA2	ValueA3	ValueA4	...
ValueB1	ValueB2	ValueB3	ValueB4	...
ValueC1	ValueC2	ValueC3	ValueC4	...
...	...	...	...	...

The process of importing the ORSI2 file, as well as the targets, was performed 15 times, once for each round within a half of a match, with the exception of round 1, as there is no prior information to round 1 for the ANN to be trained from. By importing the respective ORSI2 and targets file for a

specific round, the ANN would be trained to predict what the player positions would be for the coming round.

Once the ORSI2 and targets were imported into the MATLAB workspace, it was realised that the information contained within the ORSI2 file would be inadequate to perform the training of the ANN, and thus a manner of generating additional information was devised.

A MATLAB script was written to generate additional information. The first step in the script would open the original ORSI2 file, named “ALLdata”, now in matrix form. The second step of the script was to generate a variance of values between -3% and 3%. The variance generated ,would then also be stored in a temporary matrix, named “RandPercent”.

The third step of the script was to multiply the original matrix, “ALLdata”, with the new “RandPercent” matrix in order to generate data with the variance between -3% and 3%. The newly created data with variance was stored in a matrix named “TempData”. The fourth and fifth step was to remove any zeros from the data, as it could result in a zero weight of a neuron when training an ANN. All 0s were therefore replaced with 1s to negate the zeroing of weights when training.

The sixth step then multiplied the zero-less variance set with the “TempData” matrix and store the result back to the same matrix. The seventh step in the script was to add the variance, that was generated and stored in the “TempData” matrix, to the original “ALLdata” matrix and store it in a matrix called “FinalData”. The final step was to append the “FinalData” matrix to the “ALLdata” matrix, thus increasing the information for training. The script commands are shown in Table 11.

Table 11 *Script commands to generate additional data*

Command	Command description
open('ALLdata');	Open original dataset
RandPercent = randi([0,30],[1176,28])/1000	Create Random Percentage of values between 0 and 3%
TempData = ALLdata.*RandPercent	Create random values using Random Variance
Plusminus = Randi([-1;1],[1176,28])	Add or subtract?
Plusminus(Plusminus==0)=1	Remove 0s and replace with 1s
TempData = TempData.*Plusminus	Add or subtract!
FinalData = ALLdata + TempData	Add the variance to the original dataset
ALLDATA = [ALLdata;FinalData]	Append the variated data to the original dataset

The "ALLDATA" matrix produced by the script would thus be the input data for the ANN. The process of generating the additional data was performed on each round's ANN's input data prior to the ANN being trained.

5.2.4 Milestone 5: Development and Training of each round's ANN

Each round's ANN was developed and compiled in the MATLAB Software suit with the aid of the MATLAB neural net toolbox. The development of each ANN consisted of various steps that needed to be followed in order to train and evaluate the accuracy of the ANN. A neural net is seen as a series of algorithms used to recognise relationships in a set of data through a process that mimics the discovery of patterns within a set of data when viewed by a human brain [85][86]. Thus, an algorithm was first chosen prior to training the ANN with the neural net toolbox. A table of the possible ANN algorithms provided by MATLAB is shown.

Table 12 *Three ANN algorithm found within the ANN Toolbox*

Algorithm	Description
The Levenberg-Marquart algorithm	The Levenberg-Marquart algorithm is able to process the data fast, but is resource-intensive. Furthermore, it stops training automatically when generalisation stops improving.
The Bayesian Regularisation algorithm	The Bayesian Regularisation algorithm typically requires more time, but is best used for data that contains small and noisy datasets. Training typically stops according to adaptive weight minimisation (regularisation).
The Scaled Conjugate Gradient algorithm	The Scaled Conjugate Gradient algorithm requires less memory and automatically stops training once generalisation stops improving.

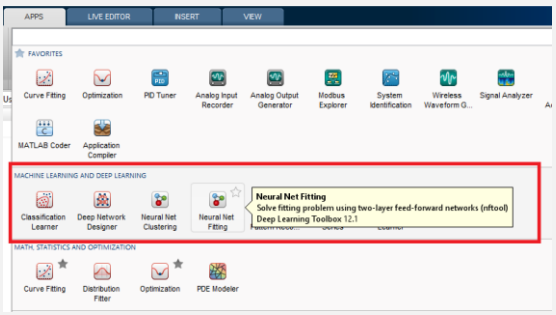
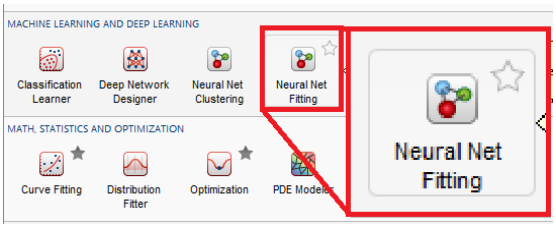
The Bayesian Regularisation algorithm was chosen for this study, as accuracy was more important than the time it would take to get results – with the added benefit that fewer resources would be required to perform the training of the ANNs.

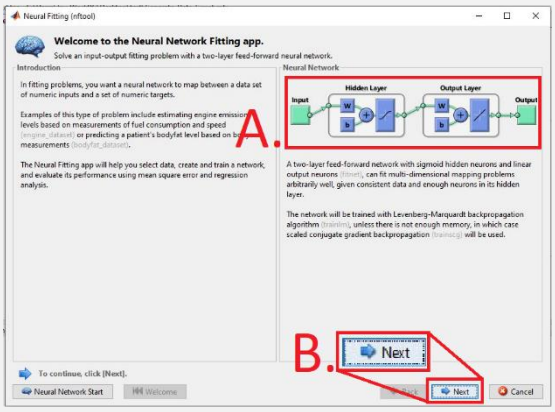
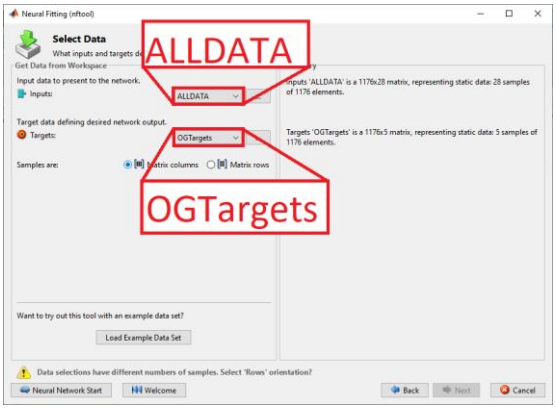
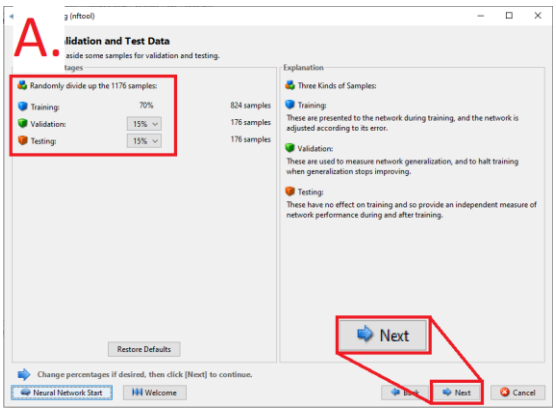
Once each round's ORSI2 files were imported into MATLAB and prepared for training using the MATLAB toolbox, each roun'ds ANN was trained one by one using the Bayesian Regularisation algorithm. When the ANN had successfully been trained, the results of the trained ANN would be evaluated in order to determine the accuracy of the ANN.

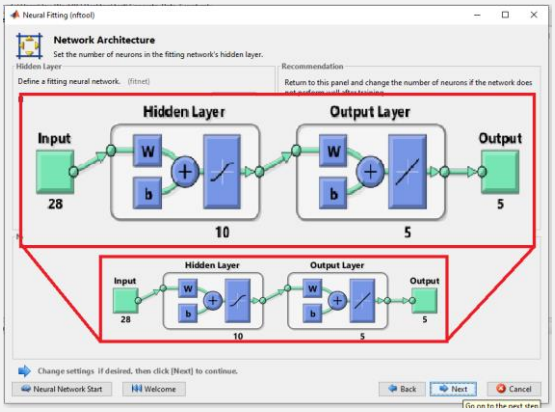
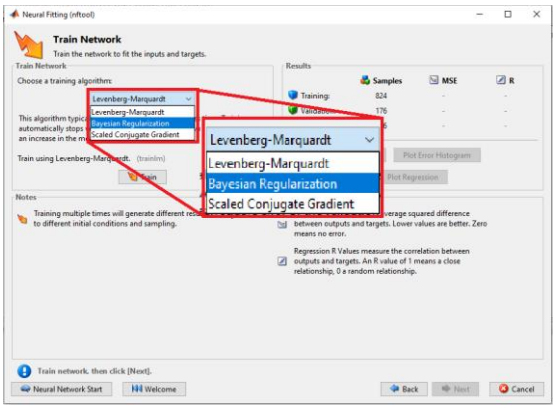
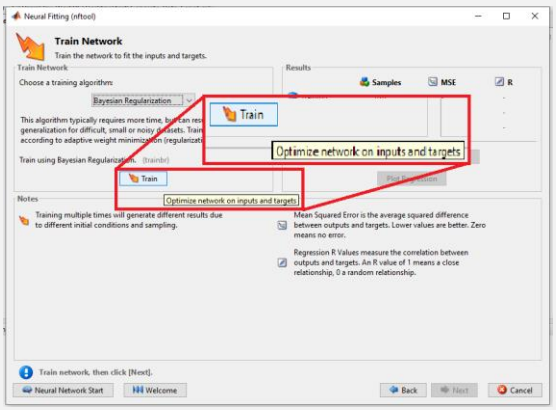
Each ANN that was trained, produced a report containing two parameters that allow one to evaluate the ANN. The parameters that were presented, are the Mean Square Error (MSE) and the

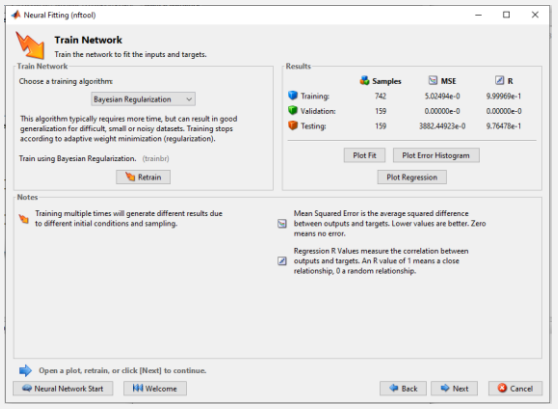
Regression (R). The MSE and R values are displayed on the results window of the ANN, once training was completed. For the ANN to be considered as accurate, the MSE-value needed to be as close to 0 as possible. The R-value measured the correlation between the outputs, being the answer the ANN produced, while the targets are the actual value. The closer the R-value is to 1, the better the correlation between the outputs and the targets, and thus the higher the accuracy. If the R-value were closer to 0 than 1, it would be indicative that the outputs and the targets have little to no correlation. The process of setting up and training each round's ANN can be found in Table 13.

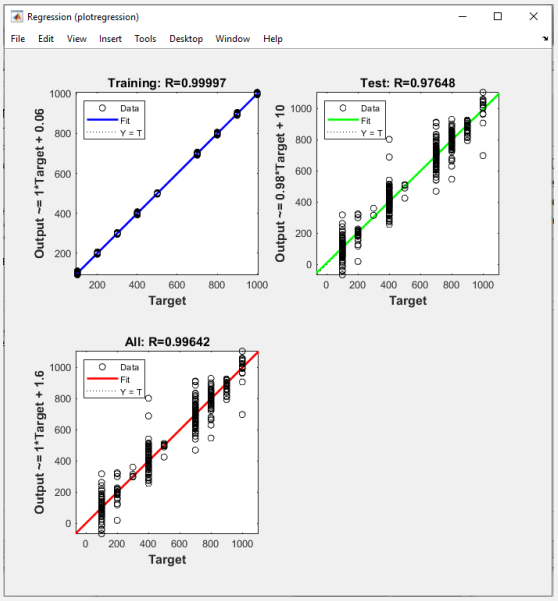
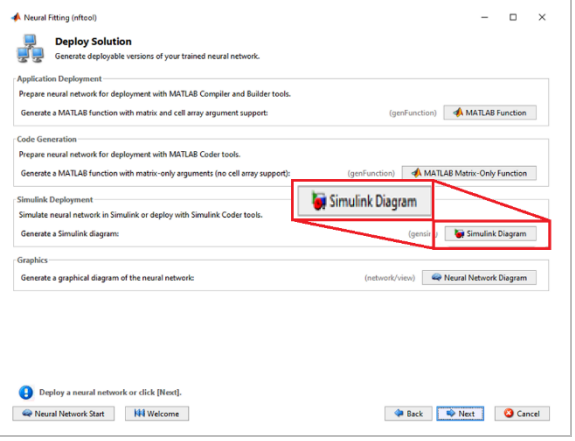
Table 13 *Steps taken to train and evaluate an ANN using the MatLab ANN Toolbox*

Step #	Description of steps	Software window
Steps to setup ANN		
1	By clicking on the "APPS" tab in the MATLAB ribbon, various different apps become available. An extensive list is provided by clicking on the drop-down arrow on the right-hand side of the ribbon.	
2	Once the drop-down is clicked, a list of various apps will be displayed under their respective categories.	
3	By locating the "Machine and Deep Learning" category, the "Neural Net Fitting" app is selected.	

4	Once selected, the Neural Net Fitting Toolbox will appear. The window contains basic information about the Neural Net Fitting App, as well as a diagram of a typical Fitting Neural Net diagram; this can be seen at A. The "Next" button, located bottom right of the window, is pressed to proceed with the setting up of the Neural Net. The "Next" button can be seen at B.	
5	Once the "Next" button is clicked, the next phase in the setting up of the Neural Net is opened. This phase allows the user to select the "Inputs" and "Targets" for the training of the Neural Net. The "ALLDATA" matrix was selected as the Inputs, and the "OGTargets" matrix was selected as the Targets for the Neural Network.	
6	With the Inputs and targets selected, one more setting was changed before clicking on the Next button bottom right of the window. The Radio button to select "Matrix Rows" was also selected, as leaving it on, the default "Matrix Columns" causes an error and will leave us unable to proceed.	
7	The next phase of setting up the Neural Net was to select the percentage of data the Neural Net would use to train, validate and test. These were left in their default values of 70% training data, 15% validation data and 15% testing data as seen in A. The "Next" button bottom right was clicked once again to proceed.	

8	With the data set divided into the appropriate amounts, the Neural Net Toolbox automatically generates a neural net topography based on the settings selected prior to this step. The Neural Net Toolbox automatically determined the input to be 28, with ten hidden layers, five output layers and five outputs. The inputs increased as the round number increased, with the outputs being constant throughout.	
9	In this step one is able to select the type of algorithm that will be used when training the Neural Net. As previously mentioned, the Bayesian Regularisation algorithm was selected for the Neural Net.	
10	With the algorithm selected, the Neural Net can be trained by clicking the “Train” button found in the middle of the “Train Network” window.	
	Training step	

11	When the Neural Net starts to train, the Neural Net Training window will appear displaying the progress of the training along with other information.	
	Evaluation of the training step	
12	Once training was completed, and successful, the “Train Network” window will reappear, displaying the results of the neural net. The “Train Network” window also allows the neural net to be retrained if the results are dissatisfactory.	

13	By clicking on the "Plot Regression" button located to the right centre of the "Train Network" window, more information on the accuracy of the neural net can be found by analysing the regression graphs produced.	
14	If the results were satisfactory, the neural net was exported to Simulink and saved by clicking Next, and opening the "Deploy Solution" window, and selecting "Simulink Diagram".	

Once the ANN was deployed to MATLAB Simulink, it was that specific round's automatic strategy predictor (ASP). The ASP consists of three function blocks found in the Simulink workspace. The first function block is labelled "x1 Constant". The "x1 Constant" function block consists of the inputs of the Neural Net. The "x1 Constant" function block is shown in Figure 44 A. The second block in the Simulink workspace is the neural net black box, labelled "Function Fitting Neural Net". The "Function Fitting Neural Net" is the neural net algorithm that takes inputs and produces the predicted outputs. It is depicted in Figure 44 B. The third block in the Simulink workspace, labelled "y1", is a scope function block. The function block serves as a display output for the "Function Fitting Neural Net" function block output. The "y1" function block can be seen in Figure 44 C., as depicted.

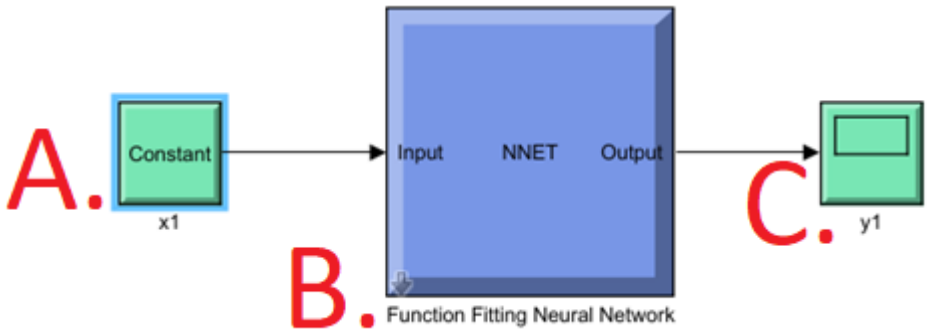


Figure 44 Simulink block diagram of an exported ANN.

In order to obtain a numerical output from the "Function Fitting Neural Net" function block, adjustments were made to the Simulink block diagram. A sink function block was added to the "Output" of the "Function Fitting Neural Net" block. The sink function block serves as a numerical indicator of the output of the "Function Fitting Neural Net". One would thus be able to see the predicted player positions based on the inputs given to the "Function Fitting Neural Net". The sink function block can be seen in Figure 45 A. Once the sink function block was added to the ANN, it was saved once again.

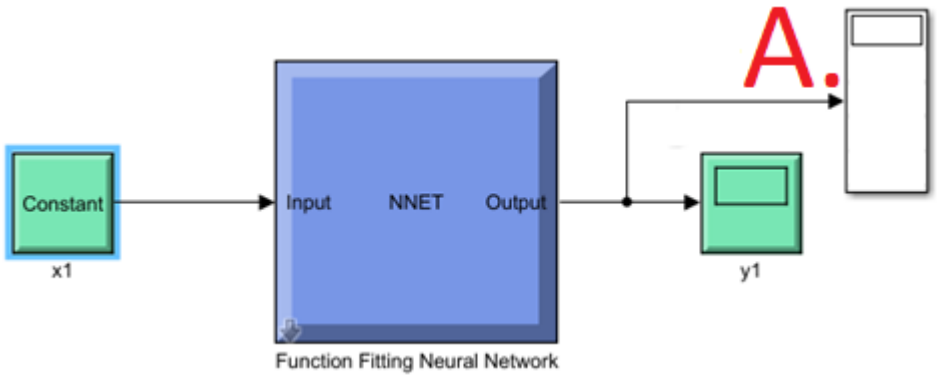


Figure 45 A. *Sink function block for displaying decimal values.*

Steps 12 and 13 from Table 13 were repeated five times. By repeating these steps five times for each rRound's ANN, the ANN that scored an R-value of 0.98 or higher of the five was selected to

determine the accuracy of each round's ANN. By finding the best ANN for each round, the fifth and final milestone was reached.

5.3 Discussion

In Phase 2, Milestone 5, 75 ANNs were developed, i.e. 5 ANNs per round in one half of a CSGO match. Of the five ANNs developed per round, the one ANN with an R-value greater or equal to 0.98 was kept; the remaining four were removed, thus leaving one ANN per round as the ASP for the specified round.

Chapter 6

Evaluation of each of the Automated Strategy Predictor

6.1 Introduction

Phase 3 involved the implementation of all 15 ANNs and the evaluation of the ASP. In order to evaluate the ability of each round's ASP to predict strategies, five unseen matches were selected. The five unseen matches were downloaded from HLTV.org. HLTV.org – the online database of all professional matches played across the world. The five matches are from January and February 2020. Figure 46 provides a conceptual flow diagram of the overall study, highlighting Phase 3, which involved the evaluation of the 15 ANNs.

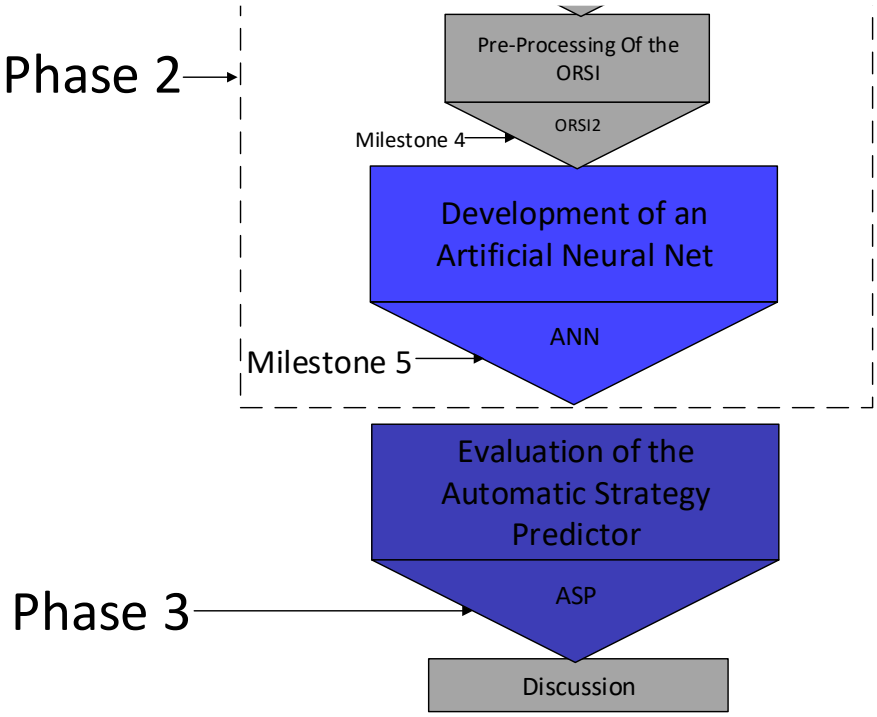
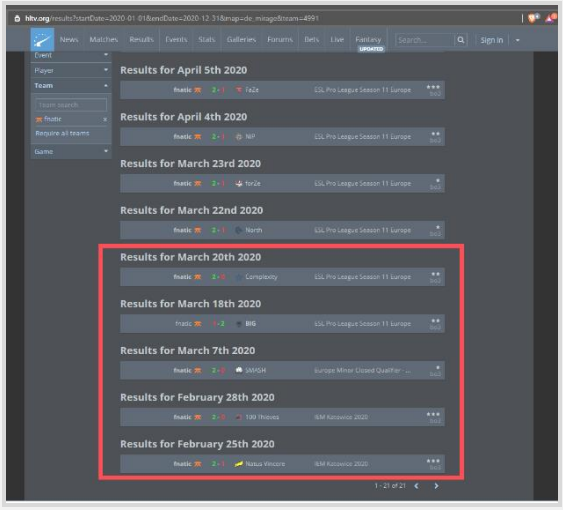


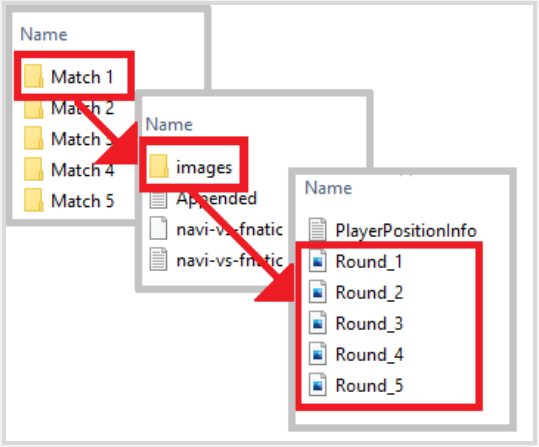
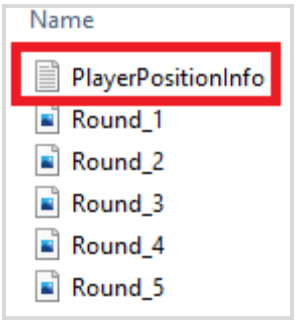
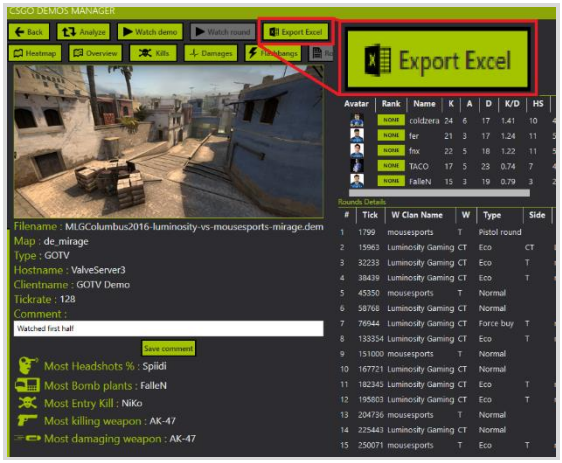
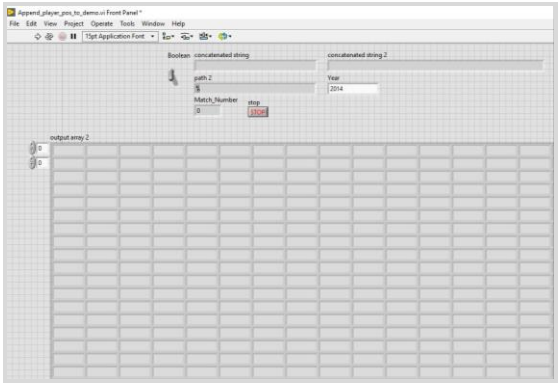
Figure 46 *Conceptual framework flow diagram, highlighting the development and evaluation of the artificial neural net*

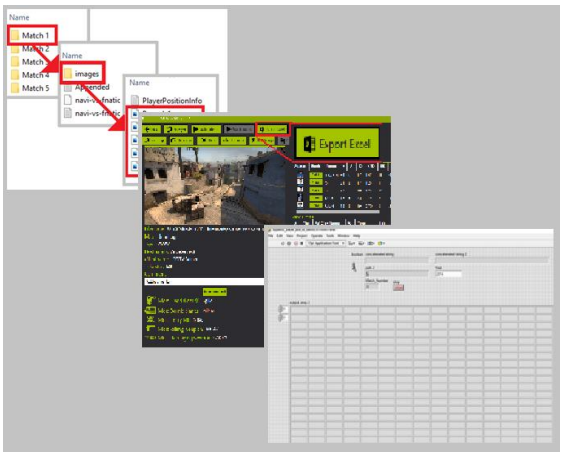
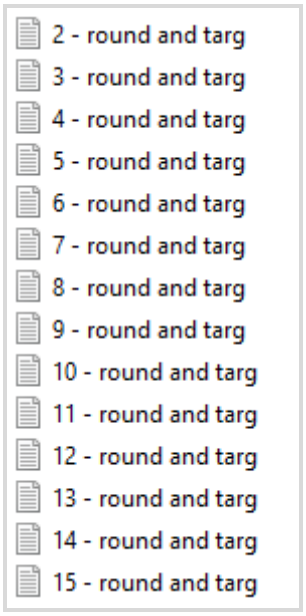
6.2 Phase 3: Evaluation of the each round's Automatic Strategy Predictor

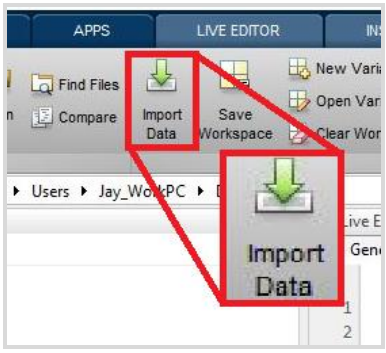
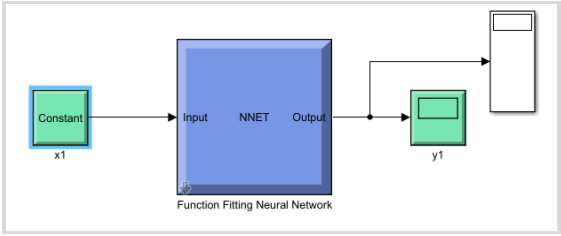
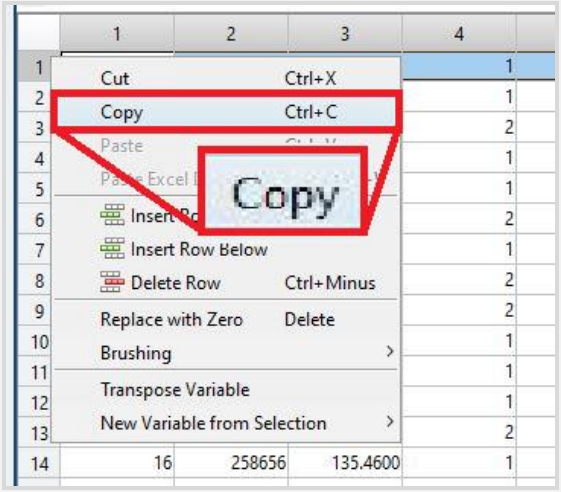
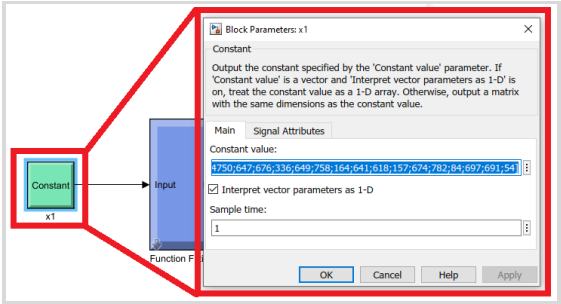
The purpose of Phase 3 of the study was to determine the accuracy of each round's automatic strategy predictor (ASP). To determine the accuracy of each round's ASP, five random matches were to be selected and used as inputs for each round's ASP. Firstly, each match was run through the machine vision software (MVS) suite, as well as the information processing system (ISP) in order for these matches to be used as inputs for each round's ASP. Once the results were obtained from each ASP, they were tabulated and saved. The tabulated results from the ASPs were compared to the results obtained through the manual review of the same five matches. Table 14 illustrates the process followed in preparing for the evaluation of Round N's ASP(N being the round number), using the ORSI2 file obtained from one match. The steps described are repeated for each of the 15 ASPs and all 5 matches.

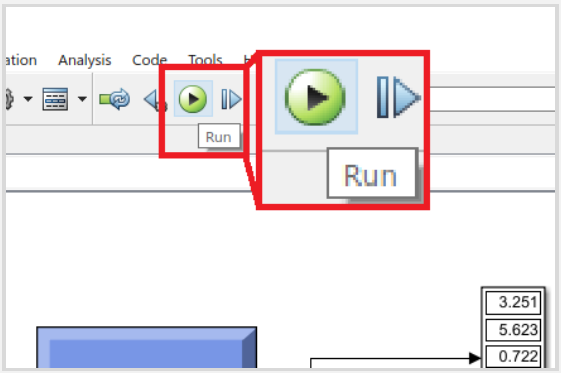
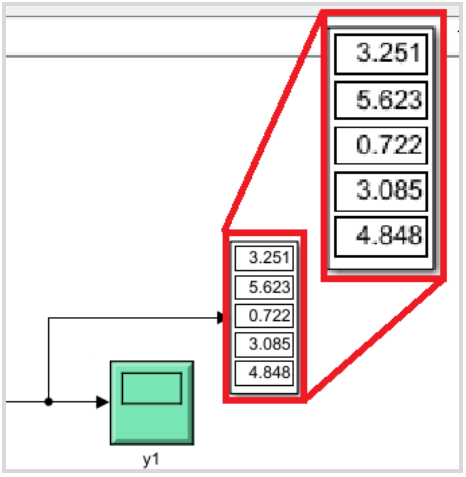
Table 14 *Steps taken to evaluate each round's ASP*

Phase 3: Evaluation of each round's ASP		
Step #	**Description of steps**	**Software window**
Steps to obtain unseen ORSI2 files		
Step 1	The first step to obtaining the overall round strategy information 2 (ORSI2) files needed to evaluate each round's ASP, was selecting the demos of five Team Fnatic matches on the map Mirage. These demos were obtained from HTLV.org.	

Step 2	Once all five match demos were obtained, they were separated into their respective folders. Another folder was created within the match folder to store all the images captured by the player position stencil producer (PPSP) for each round. Each PPSP is named after its respective round.	
Step 3	Having obtained each round's PPSs, they were fed into the stencil processor (SP) software. The SP software processed each round's PPS and produced the player position information (PPI) file. The PPI file produced, was a text file that contained the PPI for each round of the entire match.	
Step 4	The miscellaneous information of each match's demo was obtained through the CSGO Demo Manager software. The miscellaneous information file was stored by clicking on the "Export Excel" button located at the top of the UI.	
Step 5	With both the PPI and the miscellaneous information files obtained, the PPI and the miscellaneous information files were concatenated. The information concatenator (IC) software of the information processing (IP) software performed the concatenation of the PPI and the miscellaneous information files.	

	Once the PPI and the miscellaneous information files were concatenated, they were stored back into the folder containing the match demo. The user interface (UI) of the IC can be seen in the software window. Once concatenated, the file produced was the overall round strategy information (ORSI) file.	
Step 6	Steps 2, 3, 4 and 5 were repeated until all five matches had their own ORSI files.	
Step 7	Each ORSI file was then fed into the information pre-processing (IPp) software of the IPS. Once all ORSI files were processed, the IPp software produced an ORSI2 file for each round. By doing so, putting each round's information into the same format as the information used to train each round's ASP.	

	Phase 3: Evaluation of each round's ASP	
Step 8	Round N's ORSI2 file was imported into the MATLAB workspace. This was achieved by firstly locating Round N's folder. Once located, Round N's ORSI2 was imported into the MATLAB workspace and pre-processed into the correct format for the input of Round M's ASP.	
Step 9	As each round has its own ASP, Round N's ASP was opened using the MATLAB Simulink user interface.	
Step 10	In order to evaluate the accuracy of Round N's ASP, an input from Round N's ORIS2 file in the MATLAB workspace was copied.	
Step 11	The input copied from the MATLAB workspace was then pasted into the "x1 Constant" function block.	

Step 12	Once the input is inserted into Round N's ASP, the ASP was run in order to produce a prediction for the player positions in Round N of that specific match. This was performed by clicking on the "Run" button on the function ribbon.	
Step 13	By having clicked the "Run" button, the ASP produced a prediction as to what the strategy would be for the round.	
Step 14	The output of Round N's ASP is then tabulated and stored. In the software window alongside, the player positions that were observed through the manual review, is highlighted in blue and the prediction produced by the ASP is highlighted in red.	

	Manual Review Player Position				
Round N	Player 1	Player 2	Player 3	Player 4	Player 5
Match1	9	1	9	1	9
	Predicted Player Position				
Round N	Player 1	Player 2	Player 3	Player 4	Player 5
Match1	9.0	1.0	9.0	1.1	8.9

Once each match has been manually reviewed and once each ASP has been run with all the results tabulated, the accuracy of each round's ASP was calculated. In order to calculate the accuracy of each round's ASP, each of the predicted player positions is divided by the manually reviewed player position. The quotients are added together, and the absolute value of the sum of each quotient is then subtracted from 100. The difference between the summed quotient and 100 is then subtracted

from 100 once again to produce a value of 100. The final difference is the accuracy of the prediction. The equation used for the calculation of the prediction accuracy is seen in Equation 1:

Equation 1 *Prediction Accuracy*

$$PA\% = 100 - [100 - (|\sum_{x=1}^{n} \frac{Prx}{Px}|)]$$

Where: *n = 5*
PA = Prediction Accuracy
Prx = Predicted Player Position
Px = Manual Review Player Position

The full spreadsheet of all the player positions obtained through manual review, as well as all the predicted player positions for each round's ASP, is shown in Table 16.

Table 15 *All results and each ASP's Precision Accuracy*

	Manual Review Position					Predicted Results					Accuracy					%					Average	Absolute	Accuracy in %
Round 2	Player 1	Player 2	Player 3	Player 4	Player 5	Player 1	Player 2	Player 3	Player 4	Player 5	Predicted Position/Manual Review Position					Predicted Position/Manual Review Position X 100							
Match1	4	9	1	4	4	3.3	5.3	0.7	3.1	4.8	0.8	0.6	0.7	0.8	1.2	81	59	72	77	121	82.19		
Match2	9	1	4	1	4	4.4	3.4	4.9	5.0	1.4	0.5	3.4	1.2	5.0	0.4	49	343	123	499	36	209.93		
Match3	7	9	4	8	7	6.4	7.3	5.2	5.8	6.6	0.9	0.8	1.3	0.7	0.9	92	81	130	72	94	93.79		
Match4	1	1	1	1	1	2.1	1.2	1.7	1.5	2.7	2.1	1.2	1.7	1.5	2.7	212	121	175	146	274	185.50		
Match5	4	4	4	1	7	5.1	3.8	3.0	2.5	4.0	1.3	0.9	0.8	2.5	0.6	129	95	76	247	57	120.54		
																					138.39	38.39	61.61
Round 3	Player 1	Player 2	Player 3	Player 4	Player 5	Player 1	Player 2	Player 3	Player 4	Player 5													
Match1	4	7	1	4	7	4.9	8.4	2.6	3.5	7.4	1.2	1.2	2.6	0.9	1.1	121	120	263	89	105	139.71		
Match2	7	8	4	4	8	6.6	8.4	5.5	4.2	7.5	0.9	1.1	1.4	1.1	0.9	94	106	137	106	94	107.34		
Match3	1	8	1	1	1	1.1	7.6	0.6	2.2	1.6	1.1	1.0	0.6	2.2	1.6	107	95	57	219	160	127.60		
Match4	1	1	1	1	1	1.3	0.9	2.1	2.4	0.4	1.3	0.9	2.1	2.4	0.4	130	92	211	242	40	142.84		
Match5	1	4	1	4	4	1.9	4.2	0.8	4.4	4.2	1.9	1.1	0.8	1.1	1.0	194	106	80	109	105	118.89		
																					127.28	27.28	72.72
Round 4	Player 1	Player 2	Player 3	Player 4	Player 5	Player 1	Player 2	Player 3	Player 4	Player 5													
Match1	4	4	1	4	1	3.9	4.1	0.9	4.0	0.6	1.0	1.0	0.9	1.0	0.6	99	103	94	99	56	90.07		
Match2	1	1	1	8	4	0.4	0.2	1.1	7.9	4.2	0.4	0.2	1.1	1.0	1.1	35	21	114	99	106	75.01		
Match3	4	9	7	9	7	4.0	9.4	6.3	9.3	7.0	1.0	1.0	0.9	1.0	1.0	100	105	89	104	100	99.60		
Match4	4	4	8	1	1	4.1	4.0	8.4	0.4	0.9	1.0	1.0	1.0	0.4	0.9	102	99	105	45	93	88.76		
Match5	4	1	4	1	10	3.9	1.4	3.7	1.0	9.7	1.0	1.4	0.9	1.0	1.0	98	136	93	98	97	104.32		
																					91.55	8.45	91.55
Round 5	Player 1	Player 2	Player 3	Player 4	Player 5	Player 1	Player 2	Player 3	Player 4	Player 5													
Match1	4	1	4	8	1	4.1	1.0	3.9	7.8	1.1	1.0	1.0	1.0	1.0	1.1	102	104	98	98	111	102.57		
Match2	7	4	1	4	1	7.0	4.0	1.0	3.9	1.0	1.0	1.0	1.0	1.0	1.0	100	100	95	97	101	98.62		
Match3	4	1	4	1	4	3.9	1.0	3.9	0.9	3.9	1.0	1.0	1.0	0.9	1.0	97	97	98	87	97	95.25		
Match4	4	9	7	8	4	4.1	9.0	7.0	7.8	4.0	1.0	1.0	1.0	1.0	1.0	103	100	100	97	101	100.29		
Match5	4	2	1	4	4	4.0	2.3	0.9	3.7	4.1	1.0	1.1	0.9	0.9	1.0	100	115	88	92	103	99.48		
																					99.24	0.76	99.24
Round 6	Player 1	Player 2	Player 3	Player 4	Player 5	Player 1	Player 2	Player 3	Player 4	Player 5													
Match1	4	8	3	4	4	4.0	8.0	3.2	4.1	4.2	1.0	1.0	1.1	1.0	1.0	101	100	106	101	104	102.45		
Match2	4	4	1	4	4	3.9	3.9	1.2	4.0	4.2	1.0	1.0	1.2	1.0	1.1	99	98	119	100	105	104.24		
Match3	4	4	8	1	1	3.9	3.9	7.8	1.0	1.0	1.0	1.0	1.0	1.0	1.0	99	98	97	103	97	98.74		
Match4	7	4	8	9	7	7.1	4.1	8.1	9.0	7.0	1.0	1.0	1.0	1.0	1.0	101	102	101	100	100	100.90		
Match5	7	4	8	7	8	7.1	3.9	8.1	7.1	8.1	1.0	1.0	1.0	1.0	1.0	102	97	102	101	102	100.73		
																					101.41	1.41	98.59
Round 7	Player 1	Player 2	Player 3	Player 4	Player 5	Player 1	Player 2	Player 3	Player 4	Player 5													
Match1	7	8	7	7	7	7.1	8.0	7.0	7.0	7.0	1.0	1.0	1.0	1.0	1.0	101	100	100	100	100	100.08		
Match2	7	7	8	7	7	7.0	7.0	7.9	7.0	6.7	1.0	1.0	1.0	1.0	1.0	100	100	98	100	96	98.91		
Match3	9	4	4	1	4	9.0	4.1	3.9	1.0	4.0	1.0	1.0	1.0	1.0	1.0	100	102	98	101	101	100.50		
Match4	8	7	4	1	10	8.0	7.1	4.0	1.0	10.1	1.0	1.0	1.0	1.0	1.0	100	101	101	100	101	100.54		
Match5	4	9	4	1	4	4.0	9.1	4.0	0.9	4.1	1.0	1.0	1.0	0.9	1.0	99	101	100	87	103	97.79		
																					99.56	0.44	99.56
Round 8	Player 1	Player 2	Player 3	Player 4	Player 5	Player 1	Player 2	Player 3	Player 4	Player 5													
Match1	8	4	8	7	7	7.9	4.0	8.2	7.1	7.1	1.0	1.0	1.0	1.0	1.0	99	101	102	101	101	100.93		
Match2	8	8	7	7	8	8.0	8.0	6.8	7.0	7.9	1.0	1.0	1.0	1.0	1.0	100	101	98	100	99	99.36		
Match3	4	1	8	2	1	4.0	1.0	8.0	2.2	0.9	1.0	1.0	1.0	1.1	0.9	99	97	100	110	87	98.60		
Match4	4	4	1	4	1	4.0	4.1	1.0	4.0	1.0	1.0	1.0	1.0	1.0	1.0	100	102	100	100	98	100.16		
Match5	10	1	4	4	4	9.9	0.9	3.9	3.9	4.0	1.0	0.9	1.0	1.0	1.0	99	88	97	98	99	96.24		
																					99.06	0.94	99.06
Round 9	Player 1	Player 2	Player 3	Player 4	Player 5	Player 1	Player 2	Player 3	Player 4	Player 5													
Match1	4	4	7	9	10	4.1	4.0	7.1	8.9	10.1	1.0	1.0	1.0	1.0	1.0	102	99	101	99	101	100.10		
Match2	4	2	1	7	7	3.8	2.1	0.8	7.0	7.1	1.0	1.1	0.8	1.0	1.0	96	105	81	100	102	96.94		
Match3	4	7	4	8	7	4.0	7.1	4.0	8.1	6.9	1.0	1.0	1.0	1.0	1.0	100	102	101	101	99	100.37		
Match4	1	4	7	4	8	1.0	4.0	7.0	4.1	8.0	1.0	1.0	1.0	1.0	1.0	101	101	99	103	101	100.86		
Match5	1	1	7	1	8	1.1	1.0	7.0	1.0	8.0	1.1	1.0	1.0	1.0	1.0	106	95	100	100	100	100.13		
																					99.68	0.32	99.68
Round 10	Player 1	Player 2	Player 3	Player 4	Player 5	Player 1	Player 2	Player 3	Player 4	Player 5													
Match1	4	4	2	9	9	3.9	4.0	2.0	8.9	9.0	1.0	1.0	1.0	1.0	1.0	98	101	99	99	100	99.43		
Match2	4	2	4	1	9	4.1	2.0	4.0	1.1	9.0	1.0	1.0	1.0	1.1	1.0	101	99	99	107	100	101.35		
Match3	4	1	1	4	4	4.0	1.1	1.0	4.0	4.1	1.0	1.1	1.0	1.0	1.0	101	107	98	101	102	101.82		
Match4	1	9	2	7	1	1.1	9.0	2.1	7.0	1.0	1.1	1.0	1.0	1.0	1.0	106	100	103	101	103	102.36		
Match5	1	8	8	1	8	1.0	8.0	8.0	1.1	8.0	1.0	1.0	1.0	1.1	1.0	103	99	100	108	100	101.99		
																					101.39	1.39	98.61
Round 11	Player 1	Player 2	Player 3	Player 4	Player 5	Player 1	Player 2	Player 3	Player 4	Player 5													
Match1	9	1	9	1	9	9.0	1.0	9.0	1.1	8.9	1.0	1.0	1.0	1.1	1.0	100	100	100	107	99	101.01		
Match2	1	1	1	4	1	1.1	1.0	0.9	4.1	1.1	1.1	1.0	0.9	1.0	1.1	106	96	92	103	108	101.03		
Match3	1	7	4	1	1	0.9	6.9	4.0	1.0	1.1	0.9	1.0	1.0	1.0	1.1	90	98	99	99	107	98.68		
Match4	4	8	1	1	2	4.0	8.0	1.0	1.0	2.0	1.0	1.0	1.0	1.0	1.0	100	100	104	102	100	101.16		
Match5	4	8	1	7	1	4.0	8.1	4.0	7.0	1.0	1.0	1.0	4.0	1.0	1.0	100	101	405	100	104	161.89		
																					112.75	12.75	87.25

Round 12	Player 1	Player 2	Player 3	Player 4	Player 5		Player 1	Player 2	Player 3	Player 4	Player 5														
Match1	9	3	4	4	7		9.0	2.8	4.0	4.0	7.0		1.0	0.9	1.0	1.0	1.0		100	94	100	100	100		98.84		
Match2	4	1	1	7	4		4.0	1.1	1.0	6.9	4.0		1.0	1.1	1.0	1.0	1.0		101	107	104	99	99		102.08		
Match3	1	1	2	1	7		1.0	1.0	2.0	1.0	7.0		1.0	1.0	1.0	1.0	1.0		98	102	101	100	100		100.14		
Match4	7	7	7	8	7		7.0	7.0	7.0	8.0	7.1		1.0	1.0	1.0	1.0	1.0		100	100	100	100	101		100.01		
Match5	4	4	4	9	9		4.0	4.0	4.0	9.0	9.0		1.0	1.0	1.0	1.0	1.0		101	100	100	100	100		100.04		
																									100.22	0.22	99.78
Round 13	Player 1	Player 2	Player 3	Player 4	Player 5		Player 1	Player 2	Player 3	Player 4	Player 5														
Match1	4	2	4	4	1		4.0	2.0	4.0	4.0	1.0		1.0	1.0	1.0	1.0	1.0		99	100	99	100	99		99.47		
Match2	4	4	4	8	4		4.0	4.0	4.0	8.0	4.0		1.0	1.0	1.0	1.0	1.0		100	99	100	101	99		99.92		
Match3	2	2	4	9	1		2.0	2.0	3.9	8.9	1.0		1.0	1.0	1.0	1.0	1.0		99	102	99	99	95		98.96		
Match4	7	4	7	5	7		7.0	4.0	7.0	5.0	7.0		1.0	1.0	1.0	1.0	1.0		100	99	100	101	100		99.99		
Match5	7	1	4	4	1		7.0	0.9	4.0	4.0	0.9		1.0	0.9	1.0	1.0	0.9		99	95	100	101	92		97.40		
																									99.15	0.85	99.15
Round 14	Player 1	Player 2	Player 3	Player 4	Player 5		Player 1	Player 2	Player 3	Player 4	Player 5														
Match1	4	8	4	7	4		4.0	8.0	4.0	7.0	4.0		1.0	1.0	1.0	1.0	1.0		100	100	100	100	101		100.14		
Match2	7	7	9	8	8		6.6	7.0	9.0	8.0	8.0		0.9	1.0	1.0	1.0	1.0		95	101	100	100	99		99.04		
Match3	4	1	4	9	7		4.0	1.0	4.0	9.0	7.0		1.0	1.0	1.0	1.0	1.0		100	100	100	100	100		99.82		
Match4	4	1	1	8	4		4.0	1.0	1.0	8.0	4.0		1.0	1.0	1.0	1.0	1.0		99	101	102	100	100		100.48		
Match5	1	1	2	8	1		1.1	1.1	2.0	8.0	1.0		1.1	1.1	1.0	1.0	1.0		114	115	101	100	95		104.97		
																									100.89	0.89	99.11
Round 15	Player 1	Player 2	Player 3	Player 4	Player 5		Player 1	Player 2	Player 3	Player 4	Player 5														
Match1	7	4	4	4	8		7.0	4.0	4.0	4.0	8.0		1.0	1.0	1.0	1.0	1.0		100	100	99	99	100		99.65		
Match2	7	7	2	5	5		7.0	7.0	2.0	4.9	5.0		1.0	1.0	1.0	1.0	1.0		100	100	99	99	101		99.80		
Match3	7	8	7	7	7		7.0	8.0	7.0	7.0	7.0		1.0	1.0	1.0	1.0	1.0		100	100	100	101	100		100.16		
Match4	1	7	1	7	8		1.0	7.0	1.1	7.0	8.0		1.0	1.0	1.1	1.0	1.0		97	99	105	100	100		100.21		
Match5	1	8	4	4	1		1.0	8.0	4.0	4.0	1.0		1.0	1.0	1.0	1.0	1.0		100	100	99	101	98		99.66		
																									99.89	0.11	99.89

By calculating the average of each, an overall accuracy average was determined. Seen in Table 16 , the accuracy of round 2s ASP was the lowest, achieving only a 62% accuracy. The low accuracy of round 2's ASP was expected, since there is little prior information. Furthermore, each ASP after that improved accuracy; this, too, was expected. The process of reviewing the five unseen matches using the ASP took 1 hour and 13 minutes, where as the manual reviewing process of the same five matches took 6 hours and 39 minutes to complete.

Table 16 *Each round's ASP calculated accuracy*

Round N ASP	Accuracy (%)
ROUND 2 ASP	61.61
ROUND 3 ASP	72.72
ROUND 4 ASP	91.55
ROUND 5 ASP	99.24
ROUND 6 ASP	98.59
ROUND 7 ASP	99.56
ROUND 8 ASP	99.06
ROUND 9 ASP	99.68
ROUND 10 ASP	98.61
ROUND 11 ASP	87.25
ROUND 12 ASP	99.78
ROUND 13 ASP	99.15
ROUND 14 ASP	99.11
ROUND 15 ASP	99.89

6.3 Discussion

Phase 3 of the study compared the accuracy and the duration of the player position predictions of each round's ASP and the manual review of five unseen matches. The results of Phase 3 proved the ASP of the second and third rounds' accuracy was lower than the ASP of the rounds that followed. Furthermore, the overall process of analysing 5 unseen matches with the ASP 1 hour and 13 minutes, whereas the manual review of the same five matches took 6 hours and 39 minutes.

Chapter 7

Discussion and Conclusion

7.1 Introduction

In the professional sports industry one of the major considerations often overlooked is the strategising aspect when performing at the top echelon. Strategising is the cornerstone of competitive battle, both on the battlefield and in the sports industry. With machine vision and big data, technology has been used to study and develop strategies [8][9].

In previous years games also played a vital role in the military field, being used as instructional tools to teach military officers about the principles of combat. Furthermore, games have been used by the United States Navy to play war games that depict possible encounters with various adversaries and determine what the best strategies would be to defeat their adversaries in the likelihood that they would be at war with one another [77][78][79].

Subsequently much of the processes of strategising used in the military has permeated the professional gaming scene, specifically in the E-sport's Counter-Strike Global Offensive. Professional CSGO teams make use of previous encounters with various adversaries to determine the best strategy to counter and defeat their adversaries, similar to the manner in which "Kriegsspiel" was used to determine what faults were made when playing the board game [81]. In recent times the use of machine vision, as well as machine learning, has been implemented in conventional professional sports, such as soccer, to determine the best players and strategies by reviewing previous encounters against adversaries. A recent example of this being the Arsenal Football Club making use of closed circuit television (CCTV) cameras to predict players' performance, as well as the outcome of games to gain a competitive edge over competitors in the English Premier League

(EPL) [8]. Such systems have also been implemented in Formula 1, as well as in basketball and indoor hockey [9][45][75][76]. Yet nothing similar has been implemented in the E-sports arena.

The main limitation of reviewing previous encounters with adversaries, is that it is time-consuming. A professional CSGO match can be an hour long within regulation. Furthermore, an individual performing the review could miss multiple nuances. Therefore this study was undertaken to circumvent these limitations by developing an automatic strategy predictor (APS). The successfully developed ASP automatically predicts the player positions of all five players of the Team Fnatic Professional CSGO team in the proceeding round by making use of the information obtained within the current round, by combining machine vision (MV), information processing (IP) and artificial neural network (ANN) technologies. In the ASP, an MVS generates a player position stencil (PPS) by capturing the in-game mini-map at a pre-determined time within each round. The PPS for each round is processed by the MVS to produce the player position information (PPI) for each round. The information processing system (IPS) processes the PPI in order to produce an overall round strategy information 2 (ORSI2) file for each round. An ANN is then trained for using the ORSI2 file obtained from the IPS. This resulted in fifteen ANNs being developed and trained, as there are fifteen rounds in a half. Each ANN then predicts each of the five players' positions within that round.

7.2 The Automatic Strategy Predictor

The primary challenge of this study was to acquire a high fidelity, greyscale, binary image of the in-game mini-map of each round in the first half of every game ever played by the professional CSGO team, Team Fnatic. The images that were captured of the in-game mini-map, were then used to create unique player position stencils (PPS) for each round of every game. A machine vision system (MVS) was developed for the capturing of these PPSs of each round. The MVS made use of pattern matching, regions of interest (ROI), erosion, and various morphological algorithms to capture the PPS at the correct point within each round of every match ever played by Team Fnatic. The

morphological algorithms of thresholding were key to the removal of all unwanted information from the in-game mini-map consistently and for the reduction of noise within the captured images. The process of reviewing each match was automated, as over 200 matches have been played on the map de_mirage by Team Fnatic since they began playing CSGO.

A second challenge was to process the information obtained from the MVS to train each round's ANN. The information processing system (IPS) software suite was developed to produce the overall round strategy 2 (ORSI2) file for each round. This proved to be challenging, as only one half, the Terrorist (T) half, of each match was desired, and each round leading up to the current round would be needed to train the ANN for that specified round. The IPS would would therefore arrange each round from each match in an ascending sequence. For example, when creating the ORSI2 file for the fourth round's ANN, information from the first, second and third rounds of each match would be placed in order; for the ANN of the eighth round; each round from the first to the seventh would be placed in order. Arranging the information of each round in the correct order is essential for the training of an ANN. As there are no rounds prior to the first round of the match, the decision was taken that an ANN would be developed for the second round onwards and thus there was no ORSI2 file for the first round.

The final challenge in the study was to predict each player's position for each round in the T-half, based on information from all previous rounds. In order to achieve this goal, an ANN was trained for each round, with the exception of round one, by their respective ORSI2 file. The ORSI2 file is the data set used to train each of the ANNs. In order to ensure an accurate prediction of the player positions, large volumes of information were necessary to train each round's ANN. This was achieved by enlarging the original data set of each round by adding variance to all values of each of the original data set. A variance of 3% was added to each value of each of the original data set. The enlarged data sets were then used to train the ANN of each round. Once a round's ANN was trained,

an R-value was produced. The R-value was then noted and stored. The process of training each round's ANN and reviewing the R-values was repeated five times. The ANN which produced an R-value greater than or equal to 0.98 was then kept as the final ANN for the specified round, and would be the ASP for that round.

Finally, the overall player position prediction ability of each round's ASP was compared to a manual review of five never-before-seen matches. The five matches were run through both the MVS and IPS software suites in order to produce an ORSI2 file for each round of the five matches to be used as input for the testing of each round's ASP. The same five rounds were manually reviewed, and the player positions were noted down prior to the evaluation of the prediction accuracy of each round's ASP. The process of manually reviewing each of the five matches and noting down the player positions at a given time within each round, took 6 hours and 39 minutes to complete. Once each match was reviewed, each round's ASP was run for the five matches with their respective ORSI2 inputs in the Simulink UI. Once an input was placed in the input, the ASP was run in order to produce an output, and the output was stored. Each ASP was tested five times, as there were five matches. Once the ASP produced an output, the output was tabulated and stored. The process of testing each round's ASP for each of the five matches took 1 hour and 13 minutes. With both the ASP player positions predicated as well as the manual review of the player positions for each round tabulated, the accuracy of each prediction was calculated. The accuracy of the second and third rounds' ASP would be noticeably lower than the ASPs of the rounds that followed. The ASP for the second round achieved the lowest accuracy, scoring only 61.61% and 72.72%. The ASP of Round 4 was the first to successfully achieve a prediction accuracy higher than 91%. All other ASPs achieved a prediction accuracy higher than 98%, with the exception of the ASP of Round 11, as it only scored 87.25%. The overall accuracy of all the ASPs was 93.27%. Furthermore, by making use of the ASPs, the time it took to review five matches, was reduced by 5 hours and 23 minutes.

7.3 Concluding Remarks and Future Prospects

By designing and developing an ASP for each round, the challenges posed by the manual reviewing of CSGO matches was circumvented. As the manual reviewing processes of only five matches took 6 hours and 39 minutes, manually reviewing all matches ever played by a professional team could take years. The successfully developed ASPs could thus be great value to the professional CSGO E-sports teams by providing a timesaving and accurate manner of predicting an opposing team's strategy prior to a match or tournament. These ASPs could also be used to develop anti-strategies against opposing teams.

As the development of the ASPs was achieved by making use of historical matches of only one professional CSGO E-sports team, Team Fnatic, it is necessary to perform further investigations to determine whether the ASP is as accurate when predicting the strategies of other professional CSGO teams. As Team Fnatic is a Swedish gaming organisation, further studies can be performed to determine if there are similarities in the strategies used by other Swedish CSGO teams. The ASP could further be used to study the top professional CSGO teams of the various regions around the world in order to determine if the accuracy of the ASP is consistent across the various regions.

Furthermore, all the match demos of Team Fnatic that were selected for the study were played on the map de_mirage, and as such further study could be performed to determine whether the accuracy of the ASP would be similar when applied to other maps found within the game.

References

[1] M. Campbell, A.J. Hoane Jr. and F. Hsu, “Deep Blue,” Artif. Intell., vol. 134, no. 1–2, pp. 57–83, 2002, doi: 10.1016/S0004-3702(01)00129-1.

[2] C. Clark, Good Luck Have Fun: The Rise of eSports., vol. 112, no. 18. Skyhorse Publishing, 2016.

[3] “No Title,” Elecrronic Games, pp. 35–45, Mar. 1985.

[4] S.M. Janosik, NASPA Journal, 2005. https://www.pcgamesn.com/counter-strike-global-offensive/eleague-csgo-major-twitch-viewers-channel-record.

[5] M. IQBAL, “No Title.” https://www.businessofapps.com/data/twitch-statistics/.

[6] FiFa, “No Title.” https://www.fifa.com/worldcup/news/2014-fifa-world-cuptm-reached-3-2-billion-viewers-one-billion-watched--2745519.

[7] G.M.Arastey, “No Title.” https://www.sportperformanceanalysis.com/article/what-is-a-performance-analyst-in-sport.

[8] V. Rao and A. Shrivastava, “Team strategizing using a machine learning approach,” Proc. Int. Conf. Inven. Comput. Informatics, ICICI 2017, no. Icici, pp. 1032–1035, 2018, doi: 10.1109/ICICI.2017.8365296.

[9] https://analyticsindiamag.com/big-data-analytics-a-big-game-changer-in-sports/.

[10] L. King, Game on: The History and Culture of Video Games. Laurence King Pub, 2008.

[11] “Console Portraits: A 40-Year Pictorial History of Gaming.” https://www.wired.com/2007/06/gallery-game-history/.

[12] “Welcome to Poing-Story,” 2018. http://www.pong-story.com/intro.htm.

[13] “History of Gaming,” 200AD.

https://www.pbs.org/kcts/videogamerevolution/history/timeline_flash.html.

[14] R.M. Metcalfe and D.R. Boggs, “Ethernet: Distributed Packet Switching for Local Computer Networks,” Commun. ACM, vol. 19, no. 7, pp. 395–404, 1976, doi: 10.1145/360248.360253.

[15] R L. Pease, “A brief history of the NSREC,” IEEE Trans. Nucl. Sci., vol. 60, no. 3, pp. 1668–1673, 2013, doi: 10.1109/TNS.2013.2239661.

[16] R.A. Bartle, “From MUDs to MMORPGs: The History of Virtual Worlds,” Int. Handb. Internet Res., pp. 23–39, [Online]. Available: https://link.springer.com/chapter/10.1007/978-1-4020-9789-8_2.

[17] I. Koksal, “Video Gaming Industry & Its Revenue Shift.” https://www.forbes.com/sites/ilkerkoksal/2019/11/08/video-gaming-industry--its-revenue-shift/#360706bc663e.

[18] B. Hutchins, “Signs of meta-change in second modernity: The growth of e-sport and the World Cyber Games,” New Media Soc., vol. 10, no. 6, pp. 851–869, 2008, doi: 10.1177/1461444808096248.

[19] Kaze, “New CS:GO Round loss bonus,” 2019. https://www.hltv.org/forums/threads/560146/new-csgo-round-loss-bonus.

[20] M. Ejiri, “Machine Vision in Early Days: Japan’s Pioneering Contributions,” Asian Conf. Comput. Vis., pp. 35–53, 2007, [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-540-76386-4_3.

[21] P.B. Prasad, “Machine Vision Systems and Image Processing with Applications,” J. Innov. Comput. Sci. Innov., vol. 3, no. December, pp. 1–4, 2015, [Online]. Available: https://www.researchgate.net/publication/236872855_Machine_Vision_Systems_and_Image_Processing_with_Applications.

[22] N. Luwes, “Artificial Intelligence Machine Vision Grading System,” Central University of Technology, Free State, 2010.

[23] J. Radcliffe, J. Cox, and D.M. Bulanon, “Machine vision for orchard navigation,” Comput. Ind., vol. 98, pp. 165–171, 2018, doi: 10.1016/j.compind.2018.03.008.

[24] M.W. Burke, “Image Acquisition: Handbook of machine vision engineering: Volume 1,” in Image Acquisition: Handbook of machine vision engineering: Volume 1, Springer Science & Business Media, pp. 739–776.

[25] E. Adel, M. Elmogy, and H. Elbakry, “Image Stitching based on Feature Extraction Techniques: A Survey,” Int. J. Comput. Appl., vol. 99, no. 6, pp. 1–8, 2014, doi: 10.5120/17374-7818.

[26] J.J. Donnelly, “Selection of Image Acquisition Methods MDIS System,” vol. 1444, pp. 351–356, 1991.

[27] P.E.M. Phynes, “United States Patent (19),” no. 19, 1986.

[28] Y. Lv, A. Xu, C. Chong and M. Cao, “Application Analysis of the Linear CCD in the Path Recognition System,” 2007 IEEE Int. Conf. Integr. Technol., pp. 151–154, 2007.

[29] Z. Wang and Y. Ye, “Image recognition technology and optimized control algorithm of the self-tracing smart car based on CMOS camera sensor,” 2008.

[30] O. Semeniuta, S. Dransfeld, K. Martinsen and P. Falkman, “Towards Increased Intelligence and Automatic Improvement in Andustrial Vision Systems,” Procedia CIRP, vol. 67, pp. 256–261, 2018.

[31] K. Utai, M. Nagle, S. Hämmerle, W. Spreer, B. Mahayothee and J. Müller, “Mass estimation of mango fruits (Mangifera indica L., cv. ‘Nam Dokmai’) by linking image processing and artificial neural network,” Eng. Agric. Environ. Food, vol. 12, no. 1, pp. 103–110, 2019.

[32] L. Sun, Z. Fan, X. Ding, Y. Huang and J.W. Paisley, “Region-of-interest undersampled MRI reconstruction: A deep convolutional neural network approach.,” Magn. Reson. Imaging, 2019.

[33] S.-H. Lee, J. Moon and M. Lee, “A Region of Interest Based Image Segmentation Method

using a Biologically Motivated Selective Attention Model," 2006, pp. 1413–1420, doi: 10.1109/IJCNN.2006.246859.

[34] M. Meysenburg, "Thresholding," 2017. https://datacarpentry.org/image-processing/07-thresholding/.

[35] S.J. McKenna, Y. Raja and S. Gong, "Tracking colour objects using adaptive mixture models," Image Vis. Comput., vol. 17, no. 3–4, pp. 225–231, 1999, doi: 10.1016/s0262-8856(98)00104-8.

[36] N. Instruments, "Primary Morphology Operations," 2019. http://zone.ni.com/reference/en-XX/help/370281AG-01/nivisionconcepts/primary_morphology_operations/#erosionAndDilation.

[37] N. Instruments, "Advanced Morphology Operations," 2019. http://zone.ni.com/reference/en-XX/help/370281AG-01/nivisionconcepts/advanced_morphology_operations/.

[38] N. Instruments, "Color Pattern Matching," 2019. http://zone.ni.com/reference/en-XX/help/370281AG-01/nivisionconcepts/color_pattern_matching/.

[39] N. Instruments, "Object Tracking Techniques," 2015. http://zone.ni.com/reference/en-XX/help/372916T-01/nivisionconcepts/object_tracking_techniques/.

[40] J. Perš and S. Kovačič, "Computer vision system for tracking players in sports games," Int. Symp. Image Signal Process. Anal. ISPA, vol. 2000-Janua, pp. 177–182, 2000, doi: 10.1109/ISPA.2000.914910.

[41] J. Liu, P. Carr, R.T. Collins and Y. Liu, "Tracking sports players with context-conditioned motion models," Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., pp. 1830–1837, 2013, doi: 10.1109/CVPR.2013.239.

[42] A. Ribeiro, "Players Tracking in Football Game," p. 9, 2009.

[43] X. Zhang and W. Dahu, "Application of artificial intelligence algorithms in image proc[1] X. Zhang and W. Dahu, 'Application of artificial intelligence algorithms in image processing,' J.

Vis. Commun. Image Represent., vol. 61, pp. 42–49, May 2019.essing," J. Vis. Commun. Image Represent., vol. 61, pp. 42–49, 2019.

[44] B. Bal and G. Dureja, "Hawk Eye: A Logical Innovative Technology Use in Sports for Effective Decision Making," Sport Sci. Rev., vol. 21, no. 1–2, pp. 107–119, 2012, doi: 10.2478/v10237-012-0006-6.

[45] E. Cheshire, C. Halasz and J. K. Perin, "Player Tracking and Analysis of Basketball Plays," 2015.

[46] S. Yonemoto, A. Matsumoto, D. Arita and R. Taniguchi, "A real-time motion capture system with multiple camera fusion," Proc. 10th Int. Conf. Image Anal. Process., pp. 600–605, 1999.

[47] R. Waranusast, N. Bundon, V. Timtong, C. Tangnoi and P. Pattanathaburt, "Machine vision techniques for motorcycle safety helmet detection," in 28th International Conference on Image and Vision Computing New Zealand, {IVCNZ} 2013, Wellington, New Zealand, November 27-29, 2013, 2013, pp. 35–40, doi: 10.1109/IVCNZ.2013.6726989.

[48] D.B.M.J. et al., "Objective acoustic-phonetic speech analysis in patients treated for oral or oropharyngeal cancer," Folia Phoniatr. Logop., vol. 61, no. 3, pp. 180–187, 2009, doi: 10.1159/000219953 LK - http://WT3CF4ET2L.search.serialssolutions.com?sid=EMBASE&issn=10217762&id=doi:10.1159%2F000219953&atitle=Objective+acoustic-phonetic+speech+analysis+in+patients+treated+for+oral+or+oropharyngeal+cancer&stitle=Folia+Phoniatr.+Logop.&title=Folia+Phoniatrica+et+Logopaedica&volume=61&issue=3&spage=180&epage=187&aulast=De+Bruijn&aufirst=Marieke+J.&auinit=M.J.&aufull=De+Bruijn+M.J.&coden=FPLOE&isbn=&pages=180-187&date=2009&auinit1=M&auinitm=J.

[49] L.B. Nigrini, "Developing a Neural Network Model to Predict the Electrical Load Demand in the Mangaung Municipal Area," Central University of Technology, Free State., 2012.

[50] I.-A. Yeo and J.-J. Yee, "A proposal for a site location planning model of environmentally friendly urban energy supply plants using an environment and energy geographical information system (E-GIS) database (DB) and an artificial neural network (ANN)," Appl. Energy, vol. 119, pp. 99–117, 2014, doi: https://doi.org/10.1016/j.apenergy.2013.12.060.

[51] M. Jimenez Martinez and M. Alfaro Ponce, “Fatigue damage effect approach by artificial neural network,” Int. J. Fatigue, vol. 124, 2019, doi: 10.1016/j.ijfatigue.2019.02.043.

[52] J. Kim and C. Park, “End-To-End Ego Lane Estimation Based on Sequential Transfer Learning for Self-Driving Cars,” IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Work., vol. 2017-July, pp. 1194–1202, 2017, doi: 10.1109/CVPRW.2017.158.

[53] B. Marr, “The Amazing Ways Google Uses Deep Learning AI,” Enterprise Tech. https://www.forbes.com/sites/bernardmarr/2017/08/08/the-amazing-ways-how-google-uses-deep-learning-ai/#65ad5f953204.

[54] X. Ma, H. Wang, H. Li, J. Liu and H. Jiang, “Exploring sharing patterns for video recommendation on YouTube-like social media,” Multimed. Syst., vol. 20, no. 6, pp. 675–691, 2014, doi: 10.1007/s00530-013-0309-1.

[55] M. Bahrami, M. Akbari, S.A. Bagherzadeh, A. Karimipour, M. Afrand and M. Goodarzi, “Develop 24 dissimilar ANNs by suitable architectures & training algorithms via sensitivity analysis to better statistical presentation: Measure MSEs between targets & ANN for Fe–CuO/Eg–Water nanofluid,” Phys. A Stat. Mech. its Appl., vol. 519, pp. 159–168, 2019.

[56] L.B. Nigrini, “Developing a Neural Network Model to Predict the Electrical Load Demand in the Mangaung Municipal Area.” Doctoral thesis of the School of Electrical and Computer Systems Engineering, Central University of Technology, Free State, 2012.

[57] M.H. Taheri, M. Abbasi, and M. Khaki Jamei, “Using artificial neural network for computing the development length of MHD channel flows,” Mech. Res. Commun., vol. 99, pp. 8–14, 2019, doi: https://doi.org/10.1016/j.mechrescom.2019.06.003.

[58] A. Shahsavar, S. Khanmohammadi, D. Toghraie and H. Salihepour, “Experimental investigation and develop ANNs by introducing the suitable architectures and training algorithms supported by sensitivity analysis: Measure thermal conductivity and viscosity for liquid paraffin based nanofluid containing Al2O3 nanoparticles,” J. Mol. Liq., vol. 276, pp. 850–860, 2019, doi: https://doi.org/10.1016/j.molliq.2018.12.055.

[59] E. Reingold and J. Nightingale, “Artificial Neural Networks Technology,” 1999.

http://www2.psych.utoronto.ca/users/reingold/courses/ai/cache/neural3.html .

[60] B. Heung, H.C. (Derrick) Ho, J. Zhang, A. Knudby, C. Bulmer and M. Schmidt, “An overview and comparison of machine-learning techniques for classification purposes in digital soil mapping,” Geoderma, vol. 265, pp. 62–77, 2016, doi: 10.1016/j.geoderma.2015.11.014.

[61] T. Awolusi, O. Oke, O. Akinkurolere, A. Sojobi and O. Aluko, “Performance comparison of neural network training algorithms in the modeling properties of steel fiber reinforced concrete,” Heliyon, vol. e01115, pp. 1–27, 2019, doi: 10.1016/j.heliyon.2018.e01115.

[62] A. Plumb, R. Rowe, P. York and M. Brown, “Optimisation of the predictive ability of artificial neural network (ANN) models: A comparison of three ANN programs and four classes of training algorithm,” Eur. J. Pharm. Sci., vol. 25, pp. 395–405, 2005, doi: 10.1016/j.ejps.2005.04.010.

[63] K. Mohi Alden, M. Omid, A. Rajabipour, B. Tajeddin and M. Soltani Firouz, “Quality and shelf-life prediction of cauliflower under modified atmosphere packaging by using artificial neural networks and image processing,” Comput. Electron. Agric., vol. 163, no. May, 2019.

[64] A. Djalilian, “Leveling the Playing Field,” Ocul. Surf., vol. 16, no. 4, p. 393, 2018, doi: 10.1016/j.jtos.2018.08.007.

[65] M. Kopel and T. Hajas, “Implementing AI for Non-player Characters in 3D Video Games,” Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 10751 LNAI. pp. 610–619, 2018, doi: 10.1007/978-3-319-75417-8_57.

[66] T.M. Lee, Y.J. Oh, and I.-K. Lee, “Efficient Cloth Simulation using Miniature Cloth and Upscaling Deep Neural Networks,” vol. 1, no. 1, pp. 1–10, 2019, [Online]. Available: http://arxiv.org/abs/1907.03953.

[67] A. El Rhalibi, K.W. Wong, and M. Price, “Artificial intelligence for computer games,” Int. J. Comput. Games Technol., vol. 2009, no. 1, 2009, doi: 10.1155/2009/251652.

[68] G.M. Costa, “Procedural terrain generator for platform games using Markov chain,” pp. 671–

674, 2018.

[69] T.J. Rose and A.G. Bakaoukas, “Algorithms and Approaches for Procedural Terrain Generation - A Brief Review of Current Techniques,” in 2016 8th International Conference on Games and Virtual Worlds for Serious Applications (VS-GAMES), 2016, pp. 1–2, doi: 10.1109/VS-GAMES.2016.7590336.

[70] J. Pers and S. Kovacic, “A System for Tracking Players in Sports Games,” Electrotech. Rev., vol. 67, no. 5, pp. 281–288, 2000.

[71] W.T. Freeman, K. Tanaka, J. Ohta and K. Kyuma, “Computer vision for computer games,” Proc. Int. Conf. Autom. Face Gesture Recognit., pp. 100–105, 1996, doi: 10.1109/afgr.1996.557250.

[72] H. Sin and G. Lee, “Additional virtual reality training using Xbox kinect in stroke survivors with hemiplegia,” Am. J. Phys. Med. Rehabil., vol. 92, no. 10, pp. 871–880, 2013, doi: 10.1097/PHM.0b013e3182a38e40.

[73] E. Davoodi and A.R. Khanteymoori, “Horse racing prediction using Artificial Neural Networks,” Proc. 11th WSEAS Int. Conf. Neural Networks, NN ’10, Proc. 11th WSEAS Int. Conf. Evol. Comput. EC ’10, Proc. 11th WSEAS Int. Conf. Fuzzy Syst. FS ’10, pp. 155–160, 2010.

[74] M. Herold, F. Goes, S. Nopp, P. Bauer, C. Thompson and T. Meyer, “Machine learning in men’s professional football: Current applications and future directions for improving attacking play,” Int. J. Sport. Sci. Coach., vol. 14, no. 6, pp. 798–817, 2019, doi: 10.1177/1747954119879350.

[75] A. McCabe and J. Trevathan, “Artificial intelligence in sports prediction,” Proc. - Int. Conf. Inf. Technol. New Gener. ITNG 2008, pp. 1194–1197, 2008, doi: 10.1109/ITNG.2008.203.

[76] J. Liu, P. Carr, R.T. Collins and Y. Liu, “Tracking sports players with context-conditioned motion models,” Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., pp. 1830–1837, 2013, doi: 10.1109/CVPR.2013.239.

[77] "Strategy and Tactics, Military." https://www.scholastic.com/teachers/articles/teaching-content/strategy-and-tactics-military/.

[78] J. Wintjes, "Europe ' s Earliest Kriegsspiel? Book Seven of Reinhard Graf zu Solms ' Kriegsregierung and the ' Prehistory ' of Professional War Gaming," Br. J. Mil. Hist., vol. 1870, no. 1, pp. 15–33, 2015, [Online]. Available: http://bjmh.org.uk/index.php/bjmh/article/view/68.

[79] K. Mizokami, "A Brief History of Naval Wargames." https://news.usni.org/2013/09/24/brief-history-naval-wargames.

[80] D. Shlapak and M. Johnson, "Reinforcing Deterrence on NATO's Eastern Flank: Wargaming the Defense of the Baltics," Reinf. Deterrence NATO's East. Flank Wargaming Def. Balt., 2016, doi: 10.7249/rr1253.

[81] H. Raley, "War and sports have a lot of similarities." https://www.galvnews.com/lifestyle/article_ed5e6c97-e68f-55a8-b846-fe60143f1cb6.html.

[82] D. Brunner, "Frame Rate." https://www.techsmith.com/blog/frame-rate-beginners-guide/.

[83] N. Instruments, "Image Processing with NI Vision Development Module." https://www.ni.com/en-za/innovations/white-papers/06/image-processing-with-ni-vision-development-module.html.

[84] J. Hieronymus, "ASCII phonetic symbols for the world's languages: Worldbet," J. Int. Phon. Assoc., 1993, doi: 10.1002/9780470114735.hawley09038.

[85] A. Zheng and J. Jin, "Using AI to Make Predictions on Stock Market," pp. 1–6, 2017.

[86] J.CHEN,"NeuralNetwork." https://www.investopedia.com/terms/n/neuralnetwork.asp#:~:text=A neural network is a,way the human brain operates.&text=Neural networks can adapt to,to redesign the output criteria.

[87] S. Baysal and P. Duygulu, "Sentioscope: A Soccer Player Tracking System Using Model

Field Particles," IEEE Trans. Circuits Syst. Video Technol., vol. 26, no. 7, pp. 1350–1362, 2016, doi: 10.1109/TCSVT.2015.2455713.