FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Prediction of visual behaviour in immersive contents

Nuno Rodrigues de Castro Santos Silva

Mestrado em Engenharia Informática e Computação

Supervisor: Maria Teresa Andrade

Second Supervisor: Tiago Soares da Costa

June 27, 2022

Prediction of visual behaviour in immersive contents

Nuno Rodrigues de Castro Santos Silva

Mestrado em Engenharia Informática e Computação

Approved by . . . :

June 27, 2022

Abstract

In the world of broadcasting and streaming, multi-view video provides the ability to present multiple perspectives of the same video sequence, giving the viewer a sense of immersion in the real-world scene.

It can be compared to VR and 360° video, although there are significant differences, notably in the way that images are acquired: instead of placing the user at the center, presenting the scene around the user in a 360° circle, it uses multiple cameras placed in a 360° circle around the real-world scene of interest, capturing all of the possible perspectives of that scene. Additionally, in relation to VR, it uses natural video sequences and displays.

On another note, the issue that plagues content streaming of all kinds is the bandwidth requirement, which, particularly in VR and multi-view applications, leads to an increase of the required data transmission rate.

A possible solution to lower the required bandwidth would be to limit the number of views to be streamed fully, focusing on those surrounding the area at which the user is keeping his gaze. This is proposed in SmoothMV, a multi-view system that uses a non-intrusive head tracking approach to enhance the viewer's navigation and Quality of Experience (QoE). This system relies on a novel "Hot&Cold" matrix concept to translate head positioning data into viewing angle selections.

The main focus of this dissertation is on transforming and storing the gaze data acquired into datasets. These will be used as training data for a proposed Neural Network, fully integrated within SmoothMV, with the purpose of predicting the user's interest points on the screen during the playback of multi-view content.

The objective behind this effort is to predict possible user viewing interests in the near future and optimize bandwidth usage through buffering of adjacent views which could possibly be requested by the user. After concluding the development of this dataset, advancements in this dissertation will focus on the formulation of a solution to present generated heatmaps of the most viewed areas per video, previously captured using the solution developed in this work.

Keywords: Multi-View, Head-Tracking, Intel RealSense, Neural Networks, SmoothMV, Datasets, Adaptive Streaming, Multimedia, View Prediction

Resumo

No mundo da transmissão e *streaming*, o vídeo *multi-view* oferece a possibilidade de apresentar múltiplas perspectivas da mesma sequência de vídeo, proporcionando ao espectador uma sensação de imersão na cena do mundo real.

É possível estabelecer uma comparação entre Realidade Virtual e vídeos em 360º, apesar de haverem diferenças significativas, nomeadamente na maneira como os vídeos são capturados: em vez de colocar o utilizador no centro e apresentar a cena ao redor do utilizador, num círculo de 360º, o vídeo *multi-view* é capturado utilizando múltiplas câmaras colocadas à volta da cena de interesse do mundo real, capturando todas as perspectivas possíveis. Adicionalmente, em relação à realidade virtual, são utilizadas sequências de vídeo naturais.

Um problema que persegue *streaming* de todo tipo de conteúdos é o requisito de largura de banda que, principalmente em aplicações VR e de *multi-view*, se traduz num aumento da taxa de transmissão de dados necessária.

Uma possível solução para baixar a largura de banda requerida seria limitar o número de *views* a serem transmitidas, concentrando as *views* em redor da área que o utilizador visualiza. Isto é proposto pelo *SmoothMV*, um sistema *multi-view* que utiliza um método de rastreio facial não intrusivo, para melhorar a navegação e a qualidade de experiencia (QoE) do utilizador. Este sistema conta com um novo conceito de matriz *Hot&Cold*, que traduz a informação de posicionamento facial em seleções do ângulo de visão.

O objetivo principal desta dissertação é a transformação e armazenamento de dados adquiridos em *datasets*. Estes serão utilizados como dados de treino para uma rede neuronal proposta, completamente integrada com o *SmoothMV*, com o intuito de prever pontos de interesse no monitor do utilizador, durante a reprodução de conteúdo *multi-view*.

O propósito da obtenção destes dados será prever possíveis interesses visuais do utilizador num futuro próximo e otimizar a utilização da largura de banda através do buffer de *views* adjacentes que podem possivelmente ser requisitadas pelo utilizador. Após a conclusão deste dataset, o trabalho nesta dissertação irá focar-se em formular uma solução para apresentar *heatmaps*, gerados a partir das zonas mais vistas de cada vídeo, capturadas anteriormente utilizando a solução desenvolvida neste trabalho.

Palavras Chave: Multi-Vista, Rastreio-Facial, Intel RealSense, Redes Neuronais, SmoothMV, Datasets, Transmissão Adaptável, Multimédia, Previsão de dados visuais

Acknowledgements

To my mom, who has supported and stood beside me all my life, for being the pillar throughout my academic journey, never once looking back on efforts, always believing in me. In loving memory of my father.

To my brother Diogo, aunts Lena and Ana and grandparents Ernestina, Antonio, Rita and Manuel, for always being present in all moments of my life.

To Sara, who spent countless hours helping me test and review work done, for all the patience and love.

To all my family and close friends, for believing in me when at times i wouldn't, and who were there to help with whatever they could. There was always a continuous stream of people ready to participate whenever possible without any hesitation, and for that, I am deeply grateful.

As this was a research involving a lot of data gathering efforts and a lot of dynamic interactions with people, I feel deeply appreciative of everyone that took time to participate and get involved with the process.

To Eng. Maria Teresa Andrade, the coordinator and supervisor, I'd like to thank for giving me the opportunity to work with her on this research and for her guidance and appraising. I would also like to thank Eng. Tiago Soares da Costa, the proponent and second supervisor of this study, who provided continuous support throughout this thesis, answering all the questions and doubts and being present and readily available to provide guidance and advice.

Last but not least, as this is the last phase of my academic life, I would like to thank all the professors I have come across. There were lessons learned from all of them and this knowledge is very appreciated.

Nuno Castro Silva

"Predicting the future isn't magic, it's artificial intelligence."

Dave Waters

Contents

List of Figures

List of Tables

List of Abbreviations and Acronyms

AI	Artificial Intelligence
CNN	Convolutional Neural Network
FoV	Field of View
HEVC	High Efficiency Video Coding
ML	Machine Learning
MVC	Multi-view Coding
P2P	Peer-to-Peer
POV	Point of View
QoE	Quality of Experience
RAM	Recurrent Attention Models
SDK	Software Development Kit
SVC	Scalable Video Coding
VOD	Video On Demand
VR	Virtual Reality
VS	Visual Studio

Chapter 1

Introduction

This chapter is the introduction to the dissertation. The context of the work is given first, followed by the motivations and objectives. The final section gives a general outline of this document with a brief description of each chapter and its context.

1.1 Context

Ever since the appearance of computers in work spaces, the use of multiple displays in order to enhance productivity [8][9] has been a common occurrence in many professions such as in graphic design, engineering, programming, video editing, among others. In the realm of computer games, since the early versions of *Doom*, multiple-monitor display modes have been supported [10]. The increased immersion and the perception of being physically present in a non-physical world can be increased substantially with the use of multi-monitor or multi-view solutions.

Many challenges are presented on the path to increasing immersive experiences, namely due to the significant amount of resources they usually consume. "Multi-view has the potential to offer immersive viewing experiences to users, as an alternative to 360º and Virtual Reality (VR) applications" [1].

In the *SmoothMV* application, the user has the ability to freely navigate within a multi-view scene. How it works is the following, the user has access to a single screen, in which a single view of a 360º scene is displayed, which represents the content captured by a single camera. While tracking the facial data, if the user faces the right corner of that view, the scene shown on the monitor will switch, presenting the camera's view to the right of the initial scene.

This application is what is proposed through *SmoothMV* [1], an alternative to an immersive experience for 360º content, using non-intrusive head-tracking technologies, namely Intel Realsense Cameras [5]. These can offer competitive prices [11] on older models, such as the one used, when compared to medium-to-high range Virtual Reality headsets [12], while still being compatible with modern hardware and drivers.

Given the multitude of possible views, however, the bandwidth required to load these fully can be a challenge, specifically when taking into account that the user is only observing one view at a time. Therefore, a possible solution is to restrict the number of views being buffered, focusing only on the view that the user is facing and those he will likely move to. This is, therefore, a prediction problem, which will use a Neural Network to attempt to predict the buffered views. The reason behind this work, consequently, is to provide a training set to be fed to the referred Neural Network.

1.2 Motivation and Objectives

The main objective behind this dissertation was to elaborate a solution to obtain datasets that consist of the visual/facial tracking behaviour of spectators of multi-view content, using already existing technologies [5], while keeping the overall cost low. After the application was developed, a public collection of information with different users was done in order to generate data.

In a later development stage, heatmaps were generated for each view, representing the visual focus of users. These datasets in the form of heatmaps will be tested in an existing Neural Network, and compared to other already existing datasets. They will then be used to train and validate a Neural Network integrated with an adaptive multi-view system, to predict user's visual focus during playback of multi-view content, thus optimizing the delivery of views for better efficiency and bandwidth usage.

The great majority of existing datasets related to visual behaviour focus on the usage of Head Mounted Devices (HMD's) and are usually expensive[12], complex, and slightly intrusive. Therefore, with this work, we will contribute towards creating datasets for immersive content, but maintaining low intrusion and expenses.

Personally, one of the reasons that led me to take an interest in this theme was my previous experience with open-source gaze tracking software in certain video games. Using a regular webcam recording and processing the user's gaze allowed the user to shift his PoV (Point of View) in game by moving their head, which can help tremendously for example, in flight simulation games. In this example[1], even though the user has his hands occupied on flight sticks and would otherwise not be able to move their mouse to change their in-game PoV, he can use his gaze instead.

1.3 Outline

This document is organized the following way.

Chapter 1: The current chapter presents the context of the dissertation, revealing the increase of popularity of 360º content, lightly introduces *SmoothMV*, and finally establishes the organization of the document.

[1]"Use of TrackIR for immersive flying in Arma 2" https://www.youtube.com/watch?v=9wXx3vMy_AQ&t=245s

Chapter 2: The second chapter presents more in-depth information related to multi-view video and *SmoothMV*, as well as providing the state of the art on the same domains explored during this dissertation, such as head-tracking prediction, datasets in immersive content, compression in multi-view content, and insight into human attention. The research on these papers' themes was used as inspiration and source for several decisions taken during the development of this work.

Chapter 3: The third chapter provides the methodology used in all steps of the dissertation. Here a detailed description is done of the proposed solution, giving insight into the structure of the proposed dataset, as well as into the processing of the obtained data. A brief summary of the working of Machine Learning algorithms and the importance of training Neural Networks for the prediction problem posed is also given.

Chapter 4: The fourth chapter provides information regarding the technologies used in this dissertation, as well as detailing the development of all systems proposed in the previous chapter. It describes all the steps in developing the solution, delineating challenges and solutions found, and the implemented features. Here is also shown the visual design of the obtained software.

Chapter 5: The fifth chapter lays out analyses, statistics and examples of the obtained dataset, as well as some information about the users that participated in the experiment.

Chapter 6: The sixth and final chapter concludes this document with a review and the conclusions for this dissertation, as well as pointing out a few limitations, and suggesting a few changes for future work.

Chapter 2

Background & Related Work

This chapter provides some important background knowledge related to multi-view content and *SmoothMV*, followed by an introduction to Machine Learning in the case of prediction problems, and the importance of datasets in the training of Neural Networks for a better comprehension of the themes discussed throughout this dissertation. Related work in the same domains as that of this dissertation is also shown in summary. The results and conclusions derived from the papers mentioned and summarized here will be used further below as the basis for certain decisions undertaken during the development of this dissertation.

2.1 Background

2.1.1 Multi-View Video

Multi-view video is an exciting application, as it enables users to watch a static or dynamic scene from different viewing angles. As such, it can be defined as a 3D video representation, requiring multiple synchronized video signals, which show the same scenery from different viewpoints[13]. Each viewpoint, or view, is recorded through a different camera, therefore multiple video cameras are used simultaneously in order to capture a complete multi-view video, and all their viewpoints combined are what defines a scene[1].

A multi-view scene can have the cameras in various positions, either all facing away from a center position, giving a 360º surround image, viewed in Figure 2.1, or having the cameras surrounding a point of interest, allowing a view of this point from all angles, as is the case of Figure 2.2.

Multi-view video is an alternative approach to immersive streaming. Given that each view is, in fact, a video of its own, the higher the quality of each video, the higher the resolution will be for the scene. Also, the number of cameras directly influences the number of videos, and multi-view content can have any number of cameras.

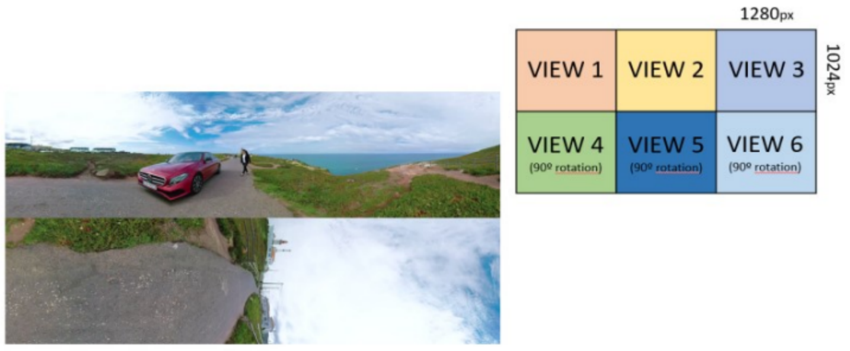


Figure 2.1: Example of a multi-view scene, divided in views [1]

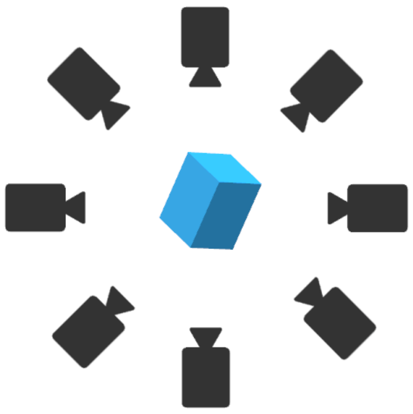


Figure 2.2: Example of the displacement of cameras surrounding a center point in multi-view scene

Therefore, when comparing a view-rich multi-view video, with a single video of the same duration, the former will have exponentially higher volume of data, depending on the video quality and number of cameras/views. This can present a few problems as multi-view media will have high bandwidth requirements for streaming or broadcasting.

A couple of approaches can be taken in order to minimize the required data bandwidth. The first involves minimizing the number of views to be streamed fully, as will be discussed further below in Section 2.1.2, and examples of other studies presented in Section 2.2.1.

The second approach concerns using different coding methods and techniques in order to compress the broadcast multi-view content. As mentioned in [14], "effectively coding multi-view visual content is an indispensable research topic because multi-view image and video that provide greatly enhanced viewing experiences often contain huge amounts of data"

Conventionally, predictive coding methods are used to address the compression by exploiting the temporal and interviewpoint redudancy existing in a multiview sequence. These are effective, yet, time-consuming, and as such different coding and compression methods are regularly used to allow for more efficient data transfer. A few of these strategies[15][16][14][17] are presented in Section 2.2.2.

In this dissertation, however, we won't go much deeper into this matter, as the main focus is the generation of a training set for a Neural Network, in order to minimize the number of broadcast views.

It is noteworthy that, within the most effective Head-Tracking solutions for lowering bandwidth requirements (Section 2.2.1), both approaches are used in order to achieve the maximum efficiency[18][1].

2.1.2 SmoothMV

SmoothMV [1] is a potential solution to lowering the bandwidth requirement in a multi-view streaming scenario. Instead of showing the full multi-view scene to the user, only one view is shown on the user's monitor. Using a non-intrusive head tracking approach with Intel Realsense Technology [5], the view presented to the user can be switched, depending on what part of the screen they are facing. This is achieved by using a novel *Hot&Cold* matrix concept, which translates head positioning data into viewing angle selections. If the user faces the right side of the screen, the presented view will switch to the view on the right of the original view, and so on for the rest of the views.

By itself, this approach already limits the number of views required to be streamed simultaneously, decreasing the bandwidth requirement a considerable amount. However, the process of switching between views, when a user focuses on a different part of the monitor, can cause latency in the event that the new view to be displayed has not been buffered yet. This is where *SmoothMV* utilizes a prediction solution to foresee the most likely view that the user will be interested in and face next. As such, the system pre-emptively buffers the most likely view so it can begin delivery beforehand, minimizing view-switching latency and maintaining a high-quality experience to the user.

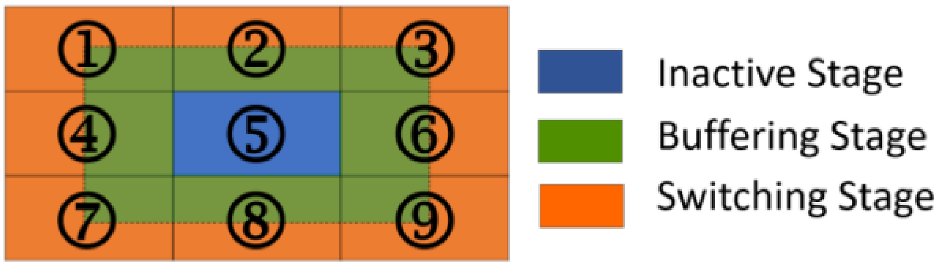


Figure 2.3: The Hot&Cold matrix, with its distinct 3 stages [1]

This is achieved by dividing the user's monitor, or the view he is facing, into 3 sections, as seen in Figure 2.3. When a user is facing the inactive region, no buffering of extra views occurs and the shown view doesn't change. When a user "is" in the buffering stage of a particular region, the system begins buffering the perspectives surrounding that region. And when a user "is" in the switching stage, the system switches the view to that which the user was facing. An example of this can be seen in Figure 2.4. Once a view-switching operation concludes, the system goes back to the inactive stage, given that the user returns to facing the center or the inactive stage.

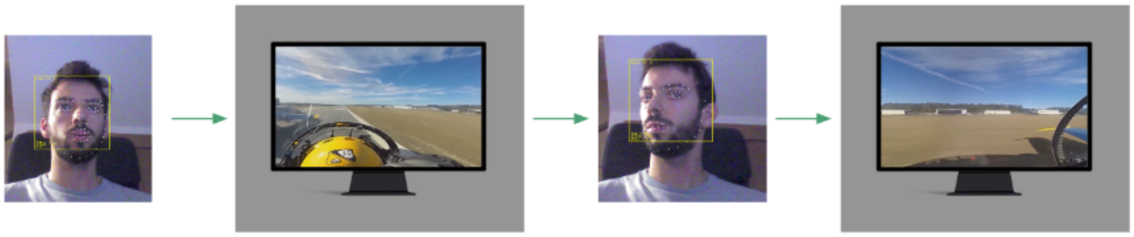


Figure 2.4: Example of a view-switching scenario with SmoothMV

Using *SmoothMV*, the accuracy of the system in selecting the next desired or correct view was above 90%, and concludes suggesting that, in future developments, user prediction will be studied through use of a Recurrent Attention Model for viewing angle prediction, requiring datasets containing multi-video content and head tracking data, which this dissertation will focus on creating.

2.1.3 Machine Learning in Prediction Problems

Machine Learning (ML) is one type of Artificial intelligence (AI) that permits software applications to more accurately predict outcomes without having to be explicitly programmed to do so. They achieve this by having ML algorithms that use historical data to predict new future output values[19].

Depending on who you ask, a different explanation will be given regarding the actual definition of ML. NVIDIA [20] defines it as “the practice of using algorithms to parse data, learn from it, and then make a determination or prediction about something in the world.” while Stanford [21] suggests that ML is “the science of getting computers to act without being explicitly programmed.”

Regardless of the chosen definition, at the most basic level, the goal behind ML is to make decisions and recommendations based on thousands of calculations and analyses, while adapting to new data independently. This is done by infusing AI machines with the capacity to learn from the data they are being fed. The system learns, identifies patterns and makes decisions with minimal human intervention. In an ideal scenario, machines increase efficiency and greatly reduce the possibility of human error.

There are several uses for ML problems, such as spam filtering, recommendation engines, real-time chatbot agents, dynamic pricing tactics, among others.

ML is very important as it allows to cut costs, mitigate risks, and improve overall quality of life for users. With the increase of access to data, and of computation power, ML is becoming ever-present everyday, and more integrated into several aspects of human life.

Several works[18][22][3][23], mentioned previously in Chapter 2, make use of ML and Neural Networks (NN) in order to predict, with high degrees of success, the future gaze location of users.

2.1.4 Training of Neural Networks

Neural Networks, in short, mimic the human brain through a sort of algorithms, allowing computer programs to recognize patterns and solve common problems in the fields of AI, ML and deep learning. In the grand scale of things, NN's are a subset of ML, which by itself is a subset of AI.

An Artificial Neural Network (ANN) is comprised of several node layers, from which there is an input layer and an output layer. Each node, or artificial neuron, in between the layers, connects one to another and has an associated weight and threshold. A simple example can be viewed in Figure 2.5. If the output of any individual node is above the specified threshold, then that node activates, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer.

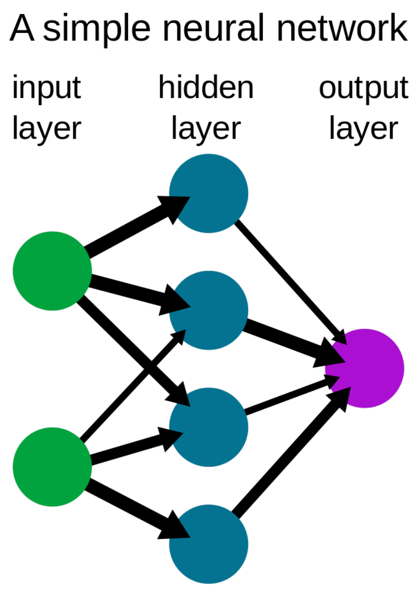


Figure 2.5: Example of a simple NN [2]

In order for these intelligent algorithms to be able to make predictions, they need to be fed training data. These training samples consist of measured data of some type combined with the solutions or results obtained, that help the NN generalize this information into a consistent input-output relationship.

A good metaphor for how a NN works is the following: "For example, let's say that you want your neural network to predict the eating quality of a tomato based on color, shape, and density. You have no idea how exactly the color, shape, and density are correlated with overall deliciousness, but you can measure color, shape, and density, and you do have taste buds. Thus, all you need to do is gather thousands upon thousands of tomatoes, record their relevant physical characteristics, taste each one (the best part), and then put all this information into a table"[24].

NN's can also be used to analyse images, or frames in a video, for example, Convoluted Neural Networks (CNN's) are most commonly applied to analyze visual imagery, but when images increase in size and quantity, CNN's become computationally expensive, due to the amount of computation scaling linearly with the number of image pixels.

A Recurrent Attention Model (RAM) is a NN that processes inputs sequentially, attending to different locations within an image one at a time, and incrementally combining information from these fixations to build up a dynamic internal representation of that image. This makes them ideal to apply to images, where CNN's falter.

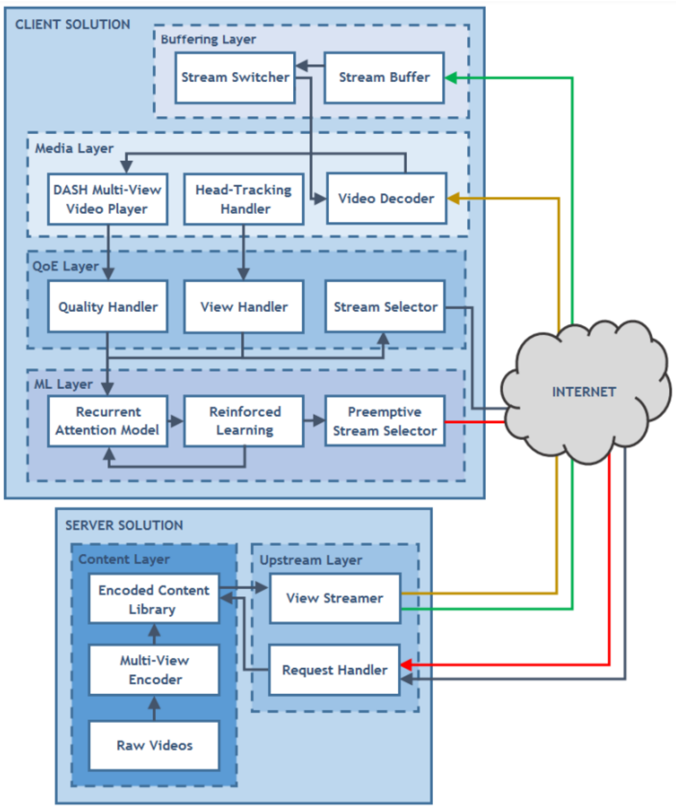


Figure 2.6: Architecture of interactive multi-view streaming system, with headtracking for view selection, ML-based content prediction and preemptive stream buffering [3]

As such, training data is very important for any kind of NN, and as aforementioned, the purpose of the dataset obtained in this dissertation is to further train ML models, namely Recurrent Attention Models (RAMs), a type of NN, suggested in the work mentioned in Section 2.2.1.3. The purpose of the referenced work was to reduce view switching latency, optimize network resources and minimize video artifacts occurring during transitioning from distinct views.

The Machine Learning techniques used involve real-time and historical head-tracking data and RAMs in order to collect streaming packets from more relevant viewing angles, or views, ahead of time. As such, possible stream selections will attempt to predict viewing angles, focus of attention and network conditions, using the current and past user head positioning.

The referenced approach, visualized in Figure 2.6 begins by collecting user head information and determining the current corresponding view, if real-time head tracking data is available.

Based on this information, the RAM's generate predictions, and the current user gaze is constantly compared with these predictions. This information, correct or incorrect, is then proceeded to a re-learning process, using reinforced learning to maximize the total sum of rewards obtained by the predictions, in order to improve the obtained knowledge and to validate the usefulness of the model.

With these predictions, we can figure the likely most suitable regions of interest, and these can be translated into a *Hot&Cold* matrix, and determine if a view change is required. If a predicted interest region is within a buffering stage (Figure 2.3), a buffering request establishes which view should be buffered for future user interactions, and if the users gaze is facing the switching stage (Figure 2.3) of the predicted interest region an immediate stream switching procedure is executed, as the data from the new view was already buffered.

It is worthy to mention that the input data to train the NN will consist in the generated heatmaps in the form of pictures (of the JPG or PNG format), where each correspond to a frame, as seen in Figure 4.16b, built from the obtained logs, along with the corresponding frame of the video it corresponds to.

With the use of Machine Learning techniques on multi-view streaming systems, the dynamic content buffering capabilities, through the use of view selection predictions, will improve the QoE of the user.

2.2 Related Work

2.2.1 Head-Tracking Prediction

2.2.1.1 Selective streaming of multi-view video for head-tracking 3d displays

Through this solution [18], a way to stream a set of dense multi-view sequences can be achieved while significantly reducing the bandwidth requirements.

This is achieved by tracking the user's head position and using live and past data to predict future positioning. With this predictive information, the client can request ahead of time from the server to fully transmit the necessary views that the client believes the user will be facing at, while at the same time minimizing delay, due to network transporting and stream switching. Highly compressed, lower-quality version of views are also requested in order to provide protection for the case of the incorrect view being predicted, and the user facing a way that was not expected.

This solution makes use of multi-view coding (MVC) and scalable video coding(SVC) concepts to obtain improved compression efficiency while also providing flexibility in bandwidth allocation to the selected views and utilizing Rate-distortion performance as a measure of evaluation. The proposed system outperforms the reference systems in the meaning of transmitted bits for general operating conditions, which is more efficient in some cases. It is also observed that the lower-quality views from adjacent visualization are worth the extra bandwidth cost since they allow for continuous play of the 3D video in cases where the predicted viewing angle differs from the actual current viewing angle.

2.2.1.2 TRACK: A Multi-Modal Deep Architecture for head motion prediction in 360º videos

TRACK [22] is a deep architecture that uses both data from past positions and knowledge of video content to broadcast only a portion or section of the screen at each point in time, which matches that of the user's predicted Field of View (FoV). *TRACK* differs from other prediction solutions by not using any live head tracking data (at the time of sending data from the server) and instead relying on data previously recorded of user's facial facing location, as well as that of the displayed video, and feeding them to a Deep Recurrent Network. In turn, a Deep Recurrent Network processes individually the time series of positions before merging the obtained embedding with visual content features, and another Recurrent Network processes the saliency map estimated from the content. Result-wise, *TRACK* achieves state-of-the-art performance on all datasets and horizons tested, from 0 to 5 seconds.

2.2.1.3 Predictive Multi-View Content Buffering applied to Interactive Streaming System

In this dissertation [3], from the same authors as, but predating, *SmoothMV* [1], the concept of Machine Learning (ML) is introduced in the field of attention focus prediction, using predictive stream buffering mechanisms based on ML techniques. The methodology consists of using Recurrent Attention Models (RAM) focusing on specific high-resolution regions and discarding non-relevant regions to reduce the total workload of other systems that use, for example, Convolutional Neural Network's (CNN).

The most suitable regions of interest can be determined for the user by utilizing live data collected from user's head positioning, and following a reinforced learning approach, to maximize the total sums of rewards obtained by the network's prediction, correlating live data and visual attention data from current frames. The propositions in this work are meant to be used in conjunction with *SmoothMV* [1], in addition to the datasets created in this dissertation.

2.2.1.4 LiveDeep: Online Viewport Prediction for Live Virtual Reality Streaming Using Lifelong Deep Learning

LiveDeep [23] proposes a way to reduce bandwidth necessities, but in this case, for Virtual Reality (VR) applications. This is achieved by predicting and streaming only the user's viewport of interest in high quality to the VR headset or device. The viewport can be defined as a device-specific viewing angle, which delimits horizontally the scene from the head direction center, called viewport center[25]. The issue here is that the majority of existing viewport prediction approaches focus only on the video-on-demand (VOD) use cases, requiring local offline processing of historical video and/or user data, not available in the live streaming scenario. In this work, however, a viewport prediction solution is achieved for live VR streaming, requiring only video content and user data in the current viewing session.

In order to address the issues of insufficient training data and real-time processing, a live VR-specific deep learning mechanism is proposed, named *LiveDeep*. *LiveDeep* employs a hybrid approach to address several challenges in live VR broadcasting by using alternative online data collection, labeling and training methods to make up for the sparse training data. After a series of tests using 48 users and 14 VR videos of different genres, the results indicated that by using *LiveDeep*, 90% of prediction accuracy and about 40% of bandwidth, savings can be achieved as well as reducing significantly processing times, translating into bandwidth and real-time requirements for successful live VR streaming.

2.2.2 Compression in Multi-View Content

2.2.2.1 A Comparative Analysis of Advance Three Dimensional Video Coding for Mobile Three Dimensional TV

Various techniques exist for encoding of 3D video content in order to solve limitations such as memory, bandwidth and processing power. In this research[15], three different techniques, H.264/MPEG-4 AVC simulcast, H.264/MVC and mixed resolution stereo coding (MRSC) are compared with each other at different bit rates, and the results shown between two parameters: peak signal to noise ratio (PSNR) and bit rate.

H.264/MPEG-4 AVC simulcast and H.264/MVC are based on a full left and right video content encoding. MVC is an extended form of H.264 standards where the depth map is generated for each view by the encoder, including some additional features to decrease decoder complexity and improve efficiency. H.264/MVC is also the precursor to High Efficiency Video Coding (HEVC), also known as H.265, used in other studies presented below[17][16] while H.264/MPEG-4 AVC is a predecessor to MPEG-H Part 2.

MRSC includes full left and a sub-sampled right video content encoding for improved coding efficiency and reduced decoding complexity. It compresses one of the outgoing generated data with the help of down-sampling of one view, while using the second view as full-length data for transmission.

The results in this study show MSRC having comparatively higher peak signal to noise ration for all the different type of standard video formats.

2.2.2.2 HEVC based Multi-View Video Codec using Frame Interleaving Technique

In this dissertation [17], HEVC (High Efficiency Video Coding) is presented for the use with multi-view video. To achieve this, the frames of multi-view videos are interleaved to create a monoscopic video sequence. The interleaving is directed in a way to increase the exploitation of the temporal and inter-views correlations.

The MV-HEVC (Multi-video HEVC) standard codec is configured to work as a single layered codec, which functions as a monoscopic HEVC codec with advanced video coding capabilities and is used to encode interleaved multi-view video frames. The performance of the suggested codec is compared with the anchor standard MV-HEVC codec by coding three standard multi-view video sequences, and experimental results show the proposed codec out performs the anchor standard M-HEVC codec in terms of bitrate and peak signal-to-noise ratio.

2.2.2.3 An Epipolar Geometry-Based Fast Disparity Estimation Algorithm for Multiview Image and Video Coding

Multiview image and video often contain huge amounts of data. In this paper[14], an alternative to conventional hybrid predictive-coding methodologies is presented, as their key yet time-consuming component, motion estimation, is not efficient in interviewpoint prediction or disparity estimation (DE).

The proposed alternative is a DE technique that accelerates the disparity search by employing epipolar geometry. With the use of optimal disparity vector distribution histograms and theoretical analysis, results show that the proposed epipolar geometry-based DE can greatly reduce search region and effectively track large and irregular disparity, which is typical in in convergent multi-view camera setups.

This technique can obtain a similar coding efficiency, as that of existing state-of-the-art fast motion estimation approaches, while achieving a significant speedup for interviewpoint prediction and coding. Moreover, a robustness study shows that the proposed DE algorithm is insensitive to the epipolar geometry estimation noise. Hence, its wide application for multi-view image and video coding is promising.

2.2.2.4 Multiview coding and compression for 3D video

In this phd thesis work[16], a multi-view video coding (MVC) is studied to solve several problems within its coding techniques that make use of visual similarities between views. MVC is the extended profile of H.265/AVC video codec, and offers compression efficiency improvements of over 50% over simulcast video coding. Better compression performance, however leads to higher random access (RA) complexity which can hamper MVC usage in applications, as well as degrading the viewers QoE due to the lack of interactivity.

In this work, two solutions to solve RA complexity are presented, by proposing faster interview prediction approaches.

The first approach aims to lower the cost of randomly accessing any picture at any point in time, with respect to the multi-view reference model JMVM and other state-of-the-art methods, achieving a RA gain of 20% relative to the reference MVC model.

The second approach achieves a RA gain of 53.33%, compared to the benchmark MVC standard. These evaluation methods were achieved by using a novel RA ability evaluation method, allowing for more accurate assessment by considering all picture types of the tested multi-view schemes.

2.2.3 Datasets in Immersive Content

2.2.3.1 A Dataset for Exploring User Behaviors in VR Spherical Video Streaming

Here [4] is an example of a dataset generated with user behaviour within immersive content. In this paper, 48 users, half male, half female, watched 18 sphere videos from 5 categories, through an HTC Vive headset, and were carefully recorded, from their head movement, to their direction of focus, and were even questioned as to what they could remember after each session.

In VR and immersive applications, head movement is one of the most important user behaviours, as it can reflect a user's visual attention, preference, and even unique motion pattern. From the generated dataset, it was shown that users share common patterns in VR spherical video streaming, which are different from conventional video streaming.

One of the reasons this dataset was generated was due to, as by the author's knowledge, no dataset containing this type of information was publicly available at the time of the publication.

Another reason for the creation of this dataset is that, with this kind of information and a better understanding of user behaviours, the development of a more efficient video transmission system or coding schemes can be developed.

Examples of the contents and graphs made through this data can be seen in Figure 2.7

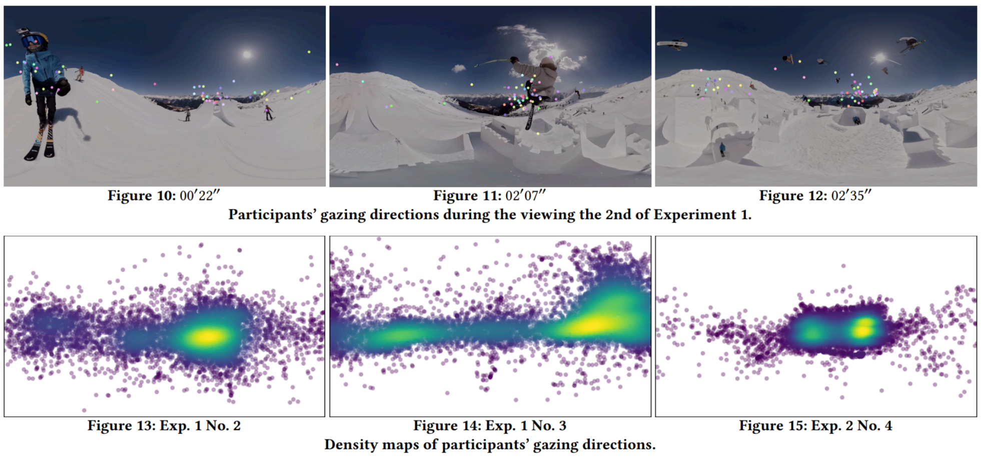


Figure 2.7: Participant's gazing directions from the dataset [4]

2.2.3.2 A Dataset of Head and Eye Movements for 360○ Videos

In this paper[26], the authors explain their methodology on creating a new dataset on visual attention in 360º contents.

Currently, a good portion of the currently published studies on the exploration of user behaviour in 360º using VR glasses consider only head movements, despite the importance of eye movements in the exploration of omnidirectional content. As such, the dataset in this work contains both eye and head movement data.

The main goal of this work is to provide a publicly available dataset to support research on visual attention for 360º content, in order to support and facilitate research activities of the community.

In order to develop this dataset, head and eye tracking was obtained from 57 users during viewing experiments.

2.2.4 An insight into human attention

2.2.4.1 Where people look when watching movies: Do all viewers look at the same place?

In this study[27], a group of 20 subjects with normal eyesight, each watched 6 different movie clips and their gaze was recorded.

It was concluded that for over half the watching time, the gaze of the subjects fell upon an area that was less than 12% of the movie scene, and that male and older subjects were more likely to look in the same direction than female and younger subjects, respectively.

The purpose of this study was to discover whether people with visual impairments that caused resolution loss, could be aided through the magnification around the most important point of a scene on a display.

2.2.4.2 Investigating Student Sustained Attention in a Guided Inquiry Lecture Course Using an Eye Tracker

For this study[28], an investigation was done on the common belief that students' attention begins to decline after the first 15 mins of class, by tracking the eyesight their eyesight during a guided inquiry physical science course.

Using Tobii Glasses, a portable eye-tracking solutions, the gaze of 17 students during different length classes was recorded. The attention of participants was divided between on-task and off-task for every second of data, and the analysis of the results showed that participants exhibited on-task vigilance percentages starting with 67% at the start of class and rising to an average of above 90% on-task vigilance at the 7 to 9-min mark with minor fluctuation and remaining high (over 80%) during the remaining length of the class.

This counters the popular belief that attention declines quickly during the run of a class, as the participants on-task gaze spans were more numerous than their off-task spans; and supports the conclusion that well-structured classes marked by student-student and instructor-student interactions can be an effective way of keeping student attention vigilance for the entirety of the class, and not just the initial 10 min.

Chapter 3

Methodology

In this chapter, the details of the solution developed in this dissertation are described. The proposed solution for tracking and data collection is presented, as well as the developed system for heatmap generation.

3.1 Proposed Solution

Performing accurate facial tracking with a common webcam from the ground up isn't easy. If there was an already existing alternative, relatively cheap and also offering accurate data, it would mean that starting from such a tested solution could result in a better starting point.

Therefore, the baseline solution made use of the available Intel Realsense F200 camera and its available source code from the Intel Realsense SDK. It's native face tracking solution was utilized to obtain the head movement data of the test users.

The software itself was developed in C#, using Visual Studio, making use of its capable Windows Form builder. Given the need to play media during user data capturing, the most straight forward solution was to use the embedded Windows Media Player SDK to be responsible for video display and managing.

One of the major advantages from using WMP, is the ability to use its native playlists to allow for several videos to be played sequentially, which in a scenario where there are several videos to be viewed, and each video is made from several different views, is very practical.

As mentioned before, since each multi-view video consists of several views, and given that only one is displayed at a time, the full-length required for a single test user to watch every view of the multi-view video, would take a great deal of time. As such, each video was divided into several segments from different views, allowing us to divide all parts of a scene into smaller parts, so that we could distribute through several playlists to test users for viewing.

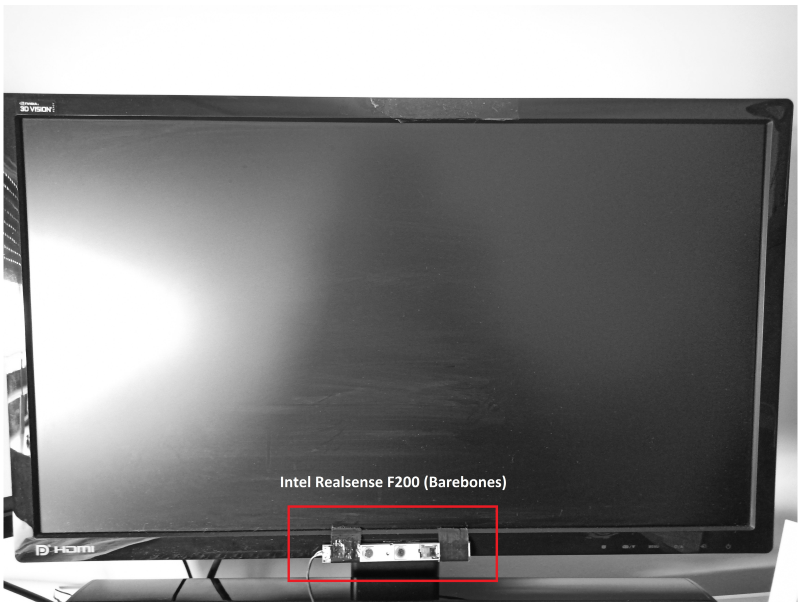


Figure 3.1: Barebones Intel Realsense F200 and monitor setup

Once the dataset generation software was done, the next step was the organization and set-up of a test bed, the division and preparation of multi-view content to be presented, and the sourcing of test users in order to put the software to test and generate the datasets themselves.

Once the dataset was completed, work consisted of developing a solution in Python to load the used videos and the obtained datasets, and from them, create heatmaps or saliency maps, which would be used to trained the proposed Neural Network, and scatter plots and then analyse the obtained data.

3.2 Features of the Dataset

The obtained dataset will need to have a list of parameters required for later processing, analysis and training. A dataset format we can take note from is the one available in *Salient360* [29], a website containing a wide range of datasets and toolboxes with the intent of training visual attention models for VR and 360º videos, or from *A Dataset of Head and Eye Movements for 360° Videos*[26]. Both of these use similar visual attention models, which can be taken as inspiration for this dissertations main goal.

Given this, such parameters must include: the Name of the user, unique to each dataset; a video timestamp, to know at what point in the video the test user was; and the X and Y coordinates of the location in the monitor at which the test facial orientation corresponds to. Other information that can be useful for later purposes are some head location attributes, such as yaw, pitch and roll, as well as the facial landmarks from pose tracking.

Regarding dependent attributes, within a view, we can divide the scene in two ways. Firstly we can divide the scene in 3 stages (Inactive Stage, Buffering Stage and Switching Stage), as viewed

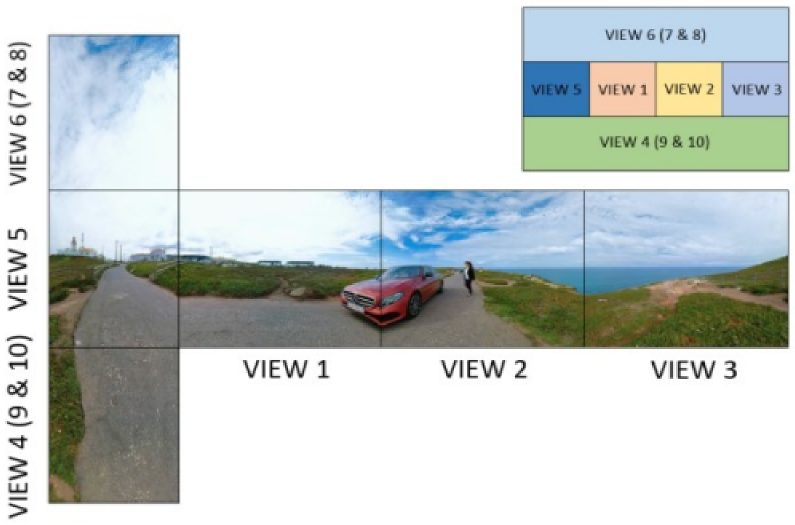


Figure 3.2: Example of the division of a multi-view video into different views [1]

in Figure 2.3 and explained in *SmoothMV*, which we refer to as semi grid position and can be numbered from 0 to 2.

Secondly, we can divide the scene in 9, from 1 to 9, as divided and numbered in Figure 2.3. This number is known as grid position.

Finally, it is necessary to know which multi-view video (or scene) the view belongs to, which can be defined as video file. Initially, height was also a parameter that was to be recorded, but was later discarded, as it didn't influence the viewing.

To clarify, the dataset will consist of several logs, where there is a log per viewing session. Each viewing session corresponds to a unique user's viewing of a playlist, containing several views, and that user's viewing data.

To put into perspective, the parameters were the following:

- User Name;
- User Age;
- User Gender;
- Video File;
- Timestamp;
- Grid Position;
- Semi Grid Position;
- Pitch;
- Yaw;
- Roll;

- X Visual Coordinate;
- Y Visual Coordinate;
- Face Landmarks.

One may consider that there is a redundant amount of information for the intended purpose of this study. In the case of the Neural Network for which this dataset is intended, truly there is an excess of parameters, where some won't end up being used for training. This is, however, on purpose, as this dataset is intended to increase the overall available datasets for training a multitude of multi-view content, where the implementer might make use of more parameters besides those used in *SmoothMV*.

3.3 Saliency/Heatmap Creation

The next step in development was the creation of a software that generates heatmaps of user's focus attention for each view, in similarity to what can be seen in this video [1]. The idea is to, for each video file representing each view, gather all corresponding user logs, construct focus attention saliency or heat maps and scatter plots, and overlay the heatmaps on top of the videos for later data analysis.

For the development of this solution, firstly, the obtained datasets, in the form of user logs, need to be read and processed and, according to which view we are generating a heatmap or scatter plot, fetch all the information from the logs containing that view.

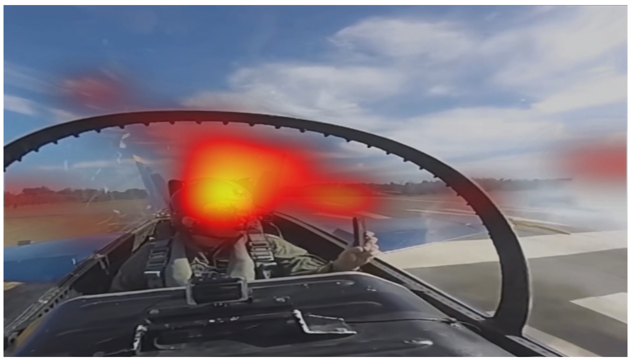


Figure 3.3: Example of a frame from the Blue Angels video with an overlaying heatmap

From here the software needs to count the number of compatible logs, and for each heatmap frame, place the correlated gaze information from the data, corresponding to a point in the screen for each user/log, allowing the generation of a heatmap.

Having all the different heatmap frames, we can go ahead and either analyse them individually, or create a heatmap video, which we can overlay on top of the corresponding view, obtaining a figure such as that in Figure 3.3.

[1]https://www.youtube.com/watch?v=8_l30wP2a2I

Chapter 4

Development

This chapter begins by providing information about the existing technologies that were used to develop the solution presented in this dissertation. Secondly, detailed insight is given about the development during different stages of progress, and how certain challenges were overcome. Finally, a description of the preparation of the videos to be displayed and the process of obtaining data is presented.

4.1 Technologies

4.1.1 Intel Realsense Technology

Intel Realsense Technology [5] is a wide product range of mainly depth and tracking technologies, allowing machines to have depth perception capabilities, such as the camera used for this dissertation, an F200 webcam (Figure 4.1).

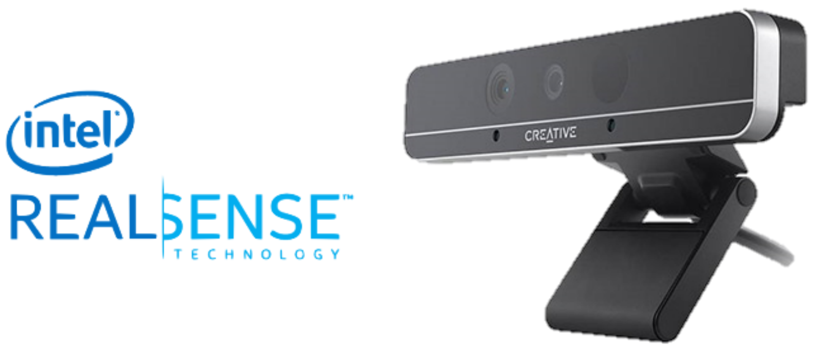


Figure 4.1: Intel Realsense F200 Developers Kit Camera [5]

Realsense products consist of Vision Processors, Depth and Tracking Modules, and Depth Cameras, supported by an open source, cross-platform SDK. The relatively inexpensive camera[11]

Table 4.1: Technical specifications table for Intel Realsense F200

Technical Specifications	
RGB video resolution:	Full HD 1080p (1920 x 1280)
IR Depth Resolution:	VGA (640 x 480)
Laser Projector:	Class 1 IR Laser Projector
Frame rate:	30 fps (RGB), 60 fps (IR depth)
FOV (Field-of-View)	77° (RGB), 90° (IR depth)
Range:	0.2m ∼1.2m
Microphone:	Dual-array microphones

and the available frameworks and documentation make it so that this technology is easy to implement onto the final solution.

In regards to the technical specifications, the camera is proved to be quite capable[30], and more importantly, requires the user to sit at about 0.2m 1.2m for accurate measurements.

4.1.2 C#

C# [31] is a general purpose, multi-paradigm, object-oriented programming language from Microsoft that combines the computing power of C++ [32] with the programming ease of Visual Basic [33]. C# is based on C++ and contains features that resemble those of Java [34]. The majority of the solution will be developed in C#, as that is the main programming language used in the base program that will be edited to create the solution.

4.1.3 Python

Python[35] is a high level, object oriented, general purpose programming language, with an emphasis on code readability. It will be used in the later stages of the project while developing a solution in order to obtain heatmaps for the most viewed section of each view, using the Matplotlib [36] library.

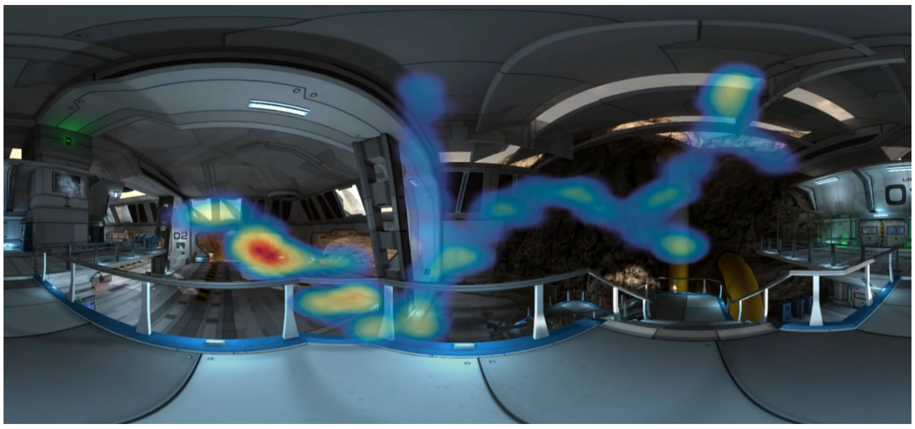


Figure 4.2: Example of a saliency/heatmap of visual tracking data overlaid on a video [6]

4.1.4 VirtualDub

VirtualDub [37] is a video capture/processing utility for Windows platforms. Although it cannot rival other more powerful video editors, it provides fast linear operations and capabilities for processing a large number of files and it is easily extended with third-party video filters. It is a great tool to trim and export videos with precision, which is why it was chosen to trim, scale and prepare views for user gaze tracking experiments.

4.1.5 Windows Media Player SDK

Windows Media Player Software Development Tool [38] is a framework that provides tools and programming technologies that can be used to extend the capabilities of Windows Media Player and Windows Media Player Mobile. Given its easy-to-use library, simple playlist management and native compatibility with the IDE used to create the program, Microsoft Visual Studio [39], this framework was chosen for both media management and playback. The only setback is the lack of ability to scale videos in order to stretch them to screen size, as it only displays videos on their original aspect ratio. However, using *VirtualDub*'s [37] video editing powers, all the videos to be used in the program are formerly edited and stretched to fit the screen.

4.2 Intel RealSense Face Tracking

The Intel RealSense SDK contains several input/output modules and algorithm modules for face tracking and other uses[7]. The *PXCFaceModule* interface of the SDK is the representation of the face tracking module, while the *PXCFaceData* interfaces abstract the face location detection, landmark detection, pose detection and expression detection data of a tracked face.

For the best results, the following points should be taken into notice:

- Good lighting is important for 2D RGB tracking;
- Shadows, back lighting or strong directional lighting - including sunlight are to be avoided;
- Slow movements work best for tracking;
- Usable range up to 1.2 meters;
- Do not tilt head outside of 30 degrees (on any axis) from the screen;
- Camera has a field of view so longest distance will be in center screen.

Each of the following subchapters present noteworthy face detection algorithms and provide a specific type of data.

4.2.1 Face Location Detection

Applications can locate one or more faces within a rectangle as viewed in Figure 4.3, useful to count how many faces are in the video sample, or to find their general locations.

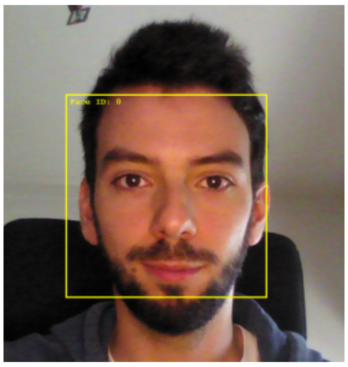


Figure 4.3: Example of a Face location Detection

4.2.2 Landmark Detection

This algorithm (as seen working in Figure 4.4) is able to pinpoint 78 key landmarks in order to further identify separate facial features (eyes, mouth, eyebrows, etc.) of the detected face. One example usage would be to determine the user's eye location in applications that change perspectives based on where the user is looking on the screen, or the creation of a face avatar.

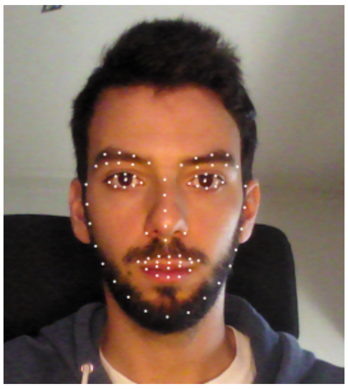


Figure 4.4: Example of a Landmark Detection

4.2.3 Pose Detection

Making use of the previous landmarks, the facial orientation algorithm, visualized in Figure 4.5, can be estimated, offering data in three ways: roll, pitch, and yaw. This can be used to, for example, change the perspective of a scene based on the user's face orientation to simulate a 3D effect.

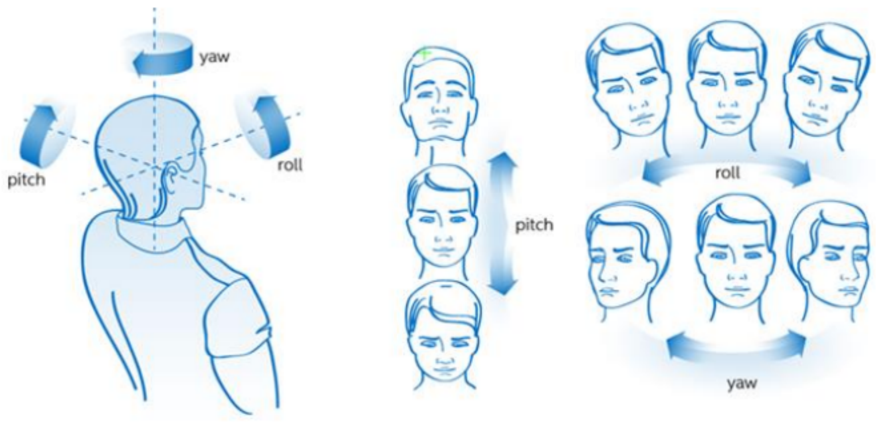
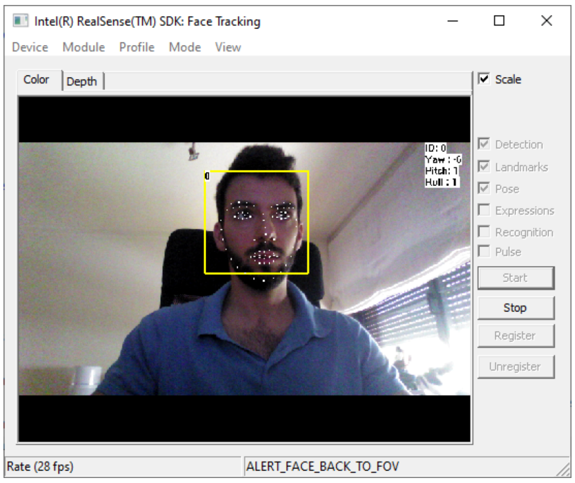


Figure 4.5: Example of a Pose Detection[7]

4.3 Solution Developed

The software developed was programmed in C#, using the Visual Studio IDE, making use of it's Window Form constructor. Initially, the coordinators provided the student with a software, viewed in Figure 4.7, which already integrated the pose detection module of Intel RealSense, which was heavily based on the Intel Realsense SDK Sample for Face tracking, Figure 4.6.



Rate (28 fps)

ALERT_FACE_BACK_TO_FOV

Figure 4.6: Intel RealSense SDK Face Tracking Sample[7]

In order for this software, viewed in Figure 4.7, to work as intended, the following changes would need to happen:

- Incorporate the WMP video player into the software;
- Allow the loading of both videos and playlists;
- Make the pose tracking match a point in the screen;

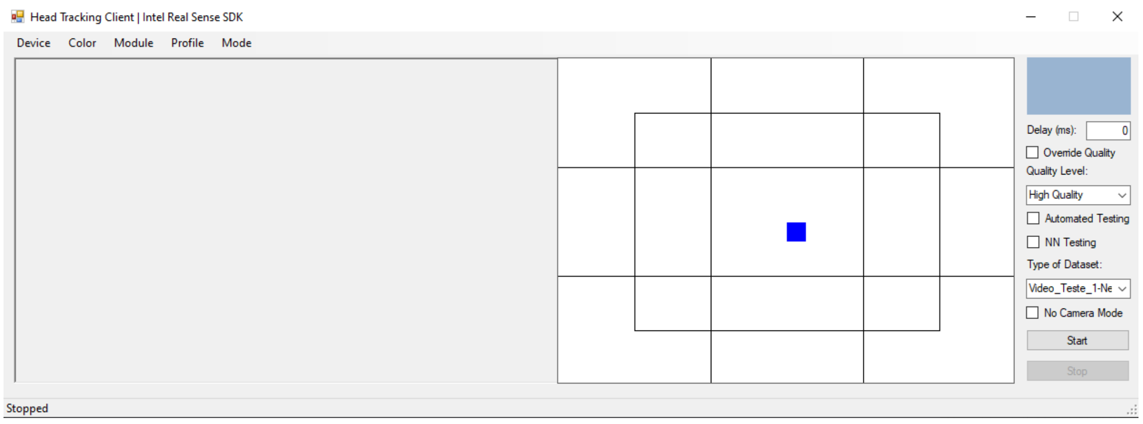


Figure 4.7: Prototype software provided by coordinators

- Implement a Calibration method;
- Create a text log with the necessary parameters required such as in Section 3.2.

4.3.1 WMP SDK Incorporation and media loading

To start, in order for WMP to work on a computer, it needs to be installed, and for the playing of .avi videos, a few codecs[1] and changes[2] need to be installed and done, respectively.

Following this, the incorporation of the WMP SDK[38] is fairly simple, as the SDK is native to VS, and it contains several very useful functions. Firstly the WMP form was placed into the Main Form of the application, and is initialized without an UI. Secondly, in the tool strip menu item, a Media button was introduced, that offers two options, "Open Video" or "Open Playlist".

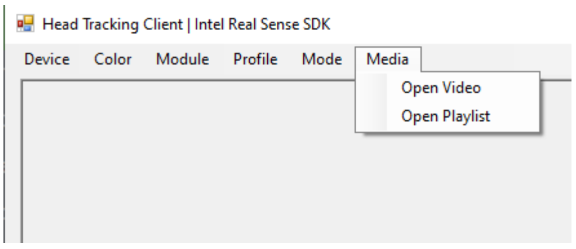


Figure 4.8: Media tool strip media options

Either of these options use an *OpenFileDialog* class to display a standard dialog box to allow the user to load their video or playlists, saving the path of the media. WMP allows either a video or a playlist file to be loaded through a path to said file.

Once the start button is pressed, if there is media loaded, the play media function is called, and the video is made to go into fullscreen. WMP SDK also supports requesting the current playing video's name, and timestamp, which will be useful later in logging, in Section 4.3.4.

[1]https://www.mediaplayercodecpack.com/
[2]https://thegeekpage.com/windows-media-player-not-working/

4.3.2 Face Tracking

Whenever the start or calibrate buttons are pressed, the camera and its software need to be initialized. As such, when either of these is pressed, a new *FaceTracking* form is created and the *SimplePipeline* method called for it, in *FaceTracking.cs* which is taken directly from the Sample, Figure 4.6.

To sum up how it works, initially a session needs to be created for Face Tracking. This is done by creating an instance of *PXCSenseManager* using *CreateInstance*. An instance of *PXCMStatus* is also created for error checking in the rest of the program.

Secondly, in order to initialize the pipeline, a few steps must be taken, such as enabling the face tracking with *EnableFace*, querying the face module instance using the *QueryFace*, initializing the pipeline, retrieving an active configuration, and finally setting the tracking mode using *SetTrackingMode*.

Once this is done, we can configure the type of detection we want (face detection, landmark, pose, seen in Section 4.2), and apply the configurations. The next step is to acquire each frame in a loop using *AcquireFrame(true)*, and in every iteration, update the face data, followed by creating all the necessary data structures to be streamed, such as, for example *PXCMFaceData::PoseEulerAngles*. At the end of every frame loop, *ReleaseFrame* is used to acquire a fresh frame in the next iteration.

On the side of the *MainForm.cs* application, queries can now be made to retrieve the needed information. In our case, Face Detection, Face Landmarks, and Face Pose are queried via *QueryDetection*, *QueryLandmarks* and *QueryPose*, respectively. From these, we get the following:

- *QueryDetection*: Retrieves a rectangle of the detected face (Figure 4.3);
- *QueryLandmarks*: Retrieves a set of facial landmarks (Figure 4.4);
- *QueryPose*: Retrieves the yaw, pitch and roll of the detected face (Figure 4.5).

From these queries, we obtain the needed facial data to later transform into screen coordinates.

4.3.3 User Gaze calibration and conversion to coordinates

On the original Face Tracking sample from Intel there is no kind of calibration, as there is no face to screen conversion, only data related with the person's facial position. As such, some kind of conversion from pose to coordinates is necessary.

As seen in the previous chapter, we have access to the yaw and pitch of a user. With these, we take inspiration from the calibration method from the Eye Tracking method of the Intel RealSense Samples, and have the user follow with their face, a red dot in the screen, which moves between 4 distinct points: up, down, left and right, and record the yaw and pitch in these movements.

From here, we can convert the pose values into X (from yaw) and Y (from pitch) coordinates. It was requested that the obtained values corresponded to a point within a 480 x 342 rectangle, as

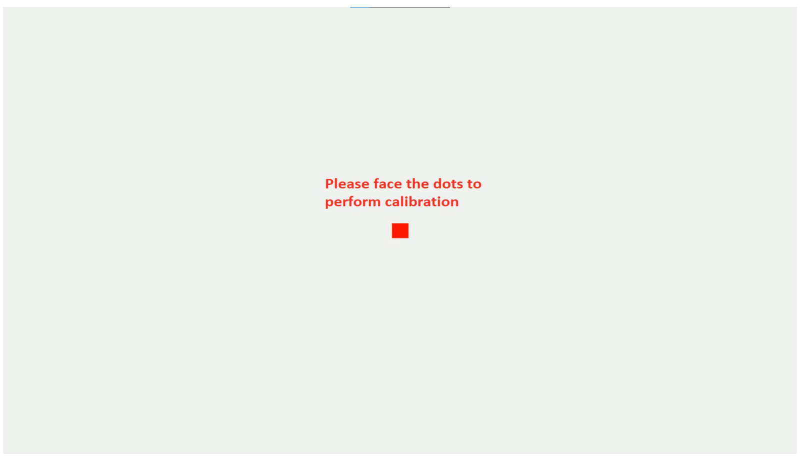


Figure 4.9: Beginning of the calibration video displayed to the user

is the size of the panel where we can see the lines dividing grid and semigrid, where the user's gaze corresponds to the blue dot in the center, as seen in Figure 4.10.

In order to convert the pose into coordinates, a rule of three is used to calculate the conversion factor. For example, knowing that horizontally, the panel's X value goes from 0 to 480, and knowing that the user's yaw value changes from -21, when facing left, to 23, when facing right, we can easily obtain a coordinate for other yaw values, when facing any other section of the screen.

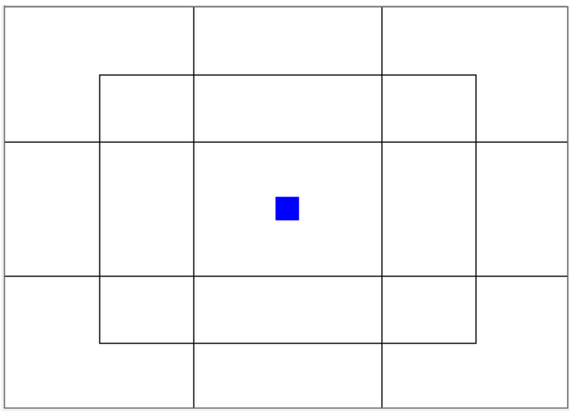
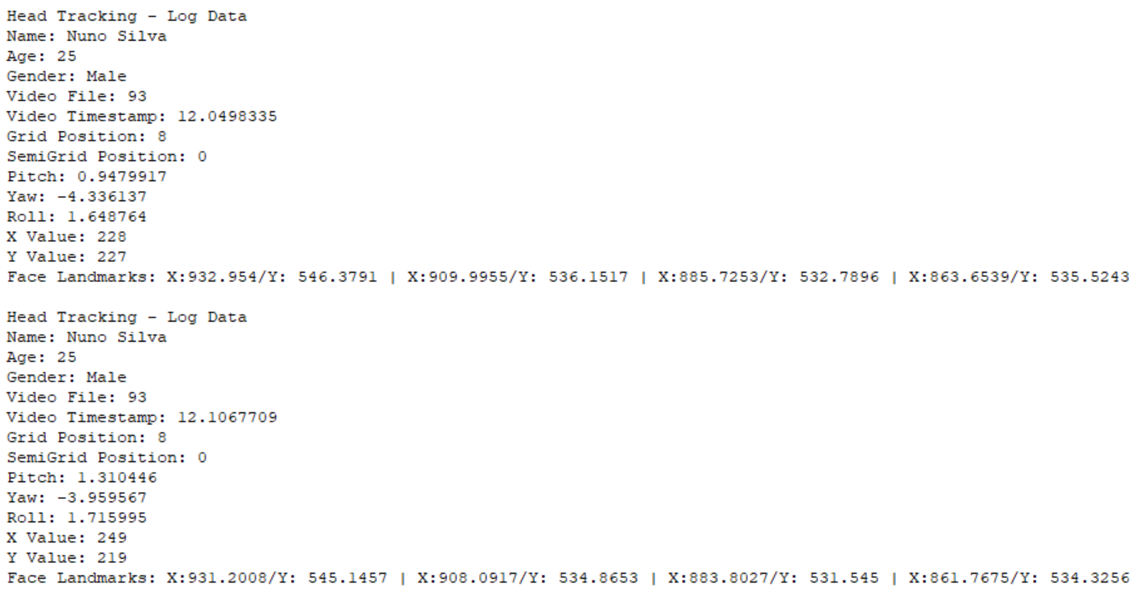


Figure 4.10: Application Panel where the blue dot corresponds to the gaze of the user, and the lines divide the panel's grids and semigrids

4.3.4 Logging

Logging consists of saving all needed parameters viewed in Section 3.2. To achieve this, every time a media is played, the *SocketData* function is called. Here, a .txt file is generated, whose name is of the format "log-file-client-yyyy-MM-dd-HH-mm-ss", according to the time and date the function was called. On this file, around every 0.05 seconds (at about 20 frames per second, or lower according to processor speed), a log file is written as seen in Figure 4.11.



```
Head Tracking - Log Data
Name: Nuno Silva
Age: 25
Gender: Male
Video File: 93
Video Timestamp: 12.0498335
Grid Position: 8
SemiGrid Position: 0
Pitch: 0.9479917
Yaw: -4.336137
Roll: 1.648764
X Value: 228
Y Value: 227
Face Landmarks: X:932.954/Y: 546.3791 | X:909.9955/Y: 536.1517 | X:885.7253/Y: 532.7896 | X:863.6539/Y: 535.5243

Head Tracking - Log Data
Name: Nuno Silva
Age: 25
Gender: Male
Video File: 93
Video Timestamp: 12.1067709
Grid Position: 8
SemiGrid Position: 0
Pitch: 1.310446
Yaw: -3.959567
Roll: 1.715995
X Value: 249
Y Value: 219
Face Landmarks: X:931.2008/Y: 545.1457 | X:908.0917/Y: 534.8653 | X:883.8027/Y: 531.545 | X:861.7675/Y: 534.3256
```

Figure 4.11: Example of two instances of a log

Here all the requested parameters in Section 3.2 are shown according to the user's gaze direction.

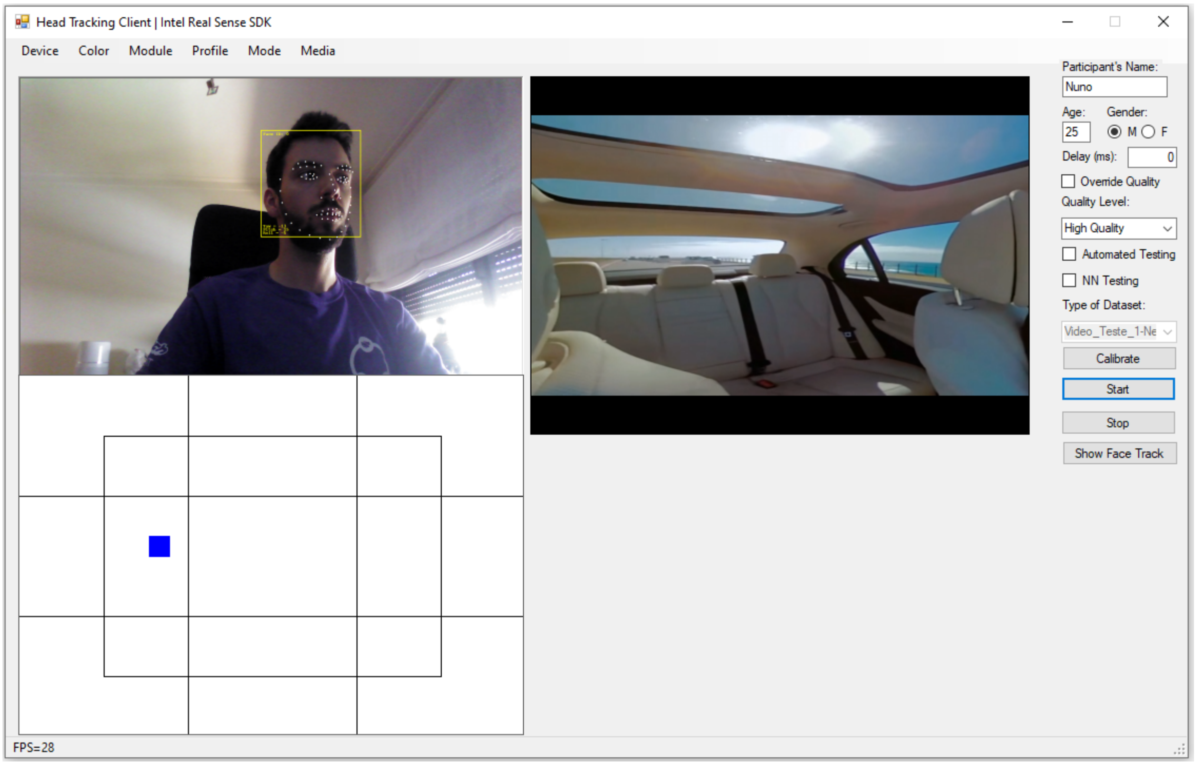
4.4 Visual Design

The developed solution, originated from editing the software visible in Figure 4.7 can be seen in Figure 4.13, and a running version of it is shown at Figure 4.12.

4.4.1 Options

From looking at Figure 4.13, starting with the upper tool strip menu, referenced by (**1**) we can view various options. The following are imported from the Sample face tracking software from Intel Realsense, which contain:

- Device: Allows choosing between available connected cameras;
- Color: Allows choosing between different color modes (YUV2, RGB24, RGB32) and different resolutions and framerates, for the camera's video;
- Module: Allows choosing between modules (only one available, Face Analysing);
- Profile: Allows choosing between different types of tracking. The default, which we use, is 3D tracking;
- Mode: Allows choosing between live, playback and record. The default, which we use, is live.



Participant's Name: Nuno
Age: 25
Gender: [✓ M] F
Delay (ms): 0
☐ Override Quality
Quality Level: High Quality
☐ Automated Testing
☐ NN Testing
Type of Dataset: Video_Teste_1-Ne

Figure 4.12: Final solution Running Example

The only button not included previously in the sample is the following:

- Media: Allows opening either a video or a playlist (Figure 4.8).

Moving on to the options referenced by (**2**) in Figure 4.13, here the user can insert their name, age and gender to appear in the log, as well as other options which will be used later in the training with neural networks.

In (**3**), we have the following options:

- Calibrate: Begins calibration Procedure;
- Start: If calibration is done, and media is loaded, begins playing media. If calibration isn't done, it begins the calibration procedure, and once it ends, begins playing media and logging;
- Stop: If media is playing, this option allows to stop all processes;
- Show Face track: If calibration is done, and media isn't yet playing, this option shows a dot on the screen which represents the user's gaze, allowing him to check if the calibration is up to par.

At the screen to the top left, (**4**), the view of the camera is shown by using Landmark, Face and Pose detection, allowing the user to check if the face tracking is working as intended.

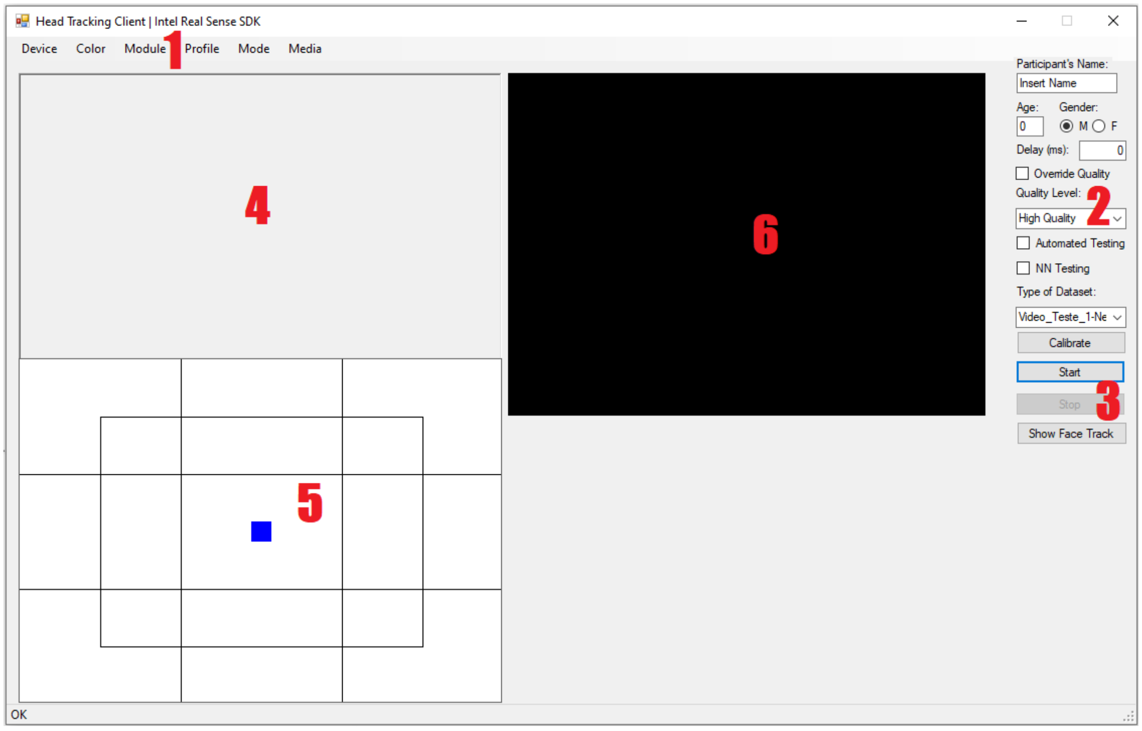


Figure 4.13: Final solution Design

In the bottom left, (**5**) the application panel is presented where the blue dot corresponds to the user's gaze, and the lines divide the panel's grids and semigrids, as viewed in Figure 4.10, updated in real time.

The screen represented by (**6**) corresponds to the video player screen. When the user presses start, the video automatically goes into full screen. However, if the user minimizes, the video will continue playing in this screen.

4.4.2 Running the application

To run the application recording gaze logs, the ensuing steps must be followed.

First, the user must fill in their name, age and gender (**2**), as well as load a media element, namely a playlist (**1**) in the "Media" option in the strip menu.

Secondly, the user must press "Calibrate" (**3**), and while the calibration video is playing (Figure 4.9), the user must follow the red dot with their gaze.

Once the calibration is done, the user can look at the application panel to view if their gaze has been well calibrated. The user can also press "Show Face Track" **3** to verify this.

Finally, to start media play and logging, the user must press "Start" (**3**), and the loaded media will go full screen.

When Visualization ends, the player will exit full screen mode, and the user can select "Stop" (**3**).

It is recommended to use the application in a multi-monitor scenario, in a way that the user can have the video playing in the main monitor, while in the side monitor, the user drags the software window in a way that both (**4**) and (**5**) are visible, in order to make sure that the software is tracking correctly, as can be seen happening in Figure 4.15.

4.5 Preparation of videos

The 360º videos provided by the coordinators for users to view for later training of the Neural Network were the following:

1. Mercedes-Benz E-Class drive in Lisbon[3];
2. Blue Angels Flight[4];
3. Amusement Ride[5];
4. Ancient City of Petra[6];
5. Carnival of Venice[7];
6. Real Madrid vs Juventus 2017 Champions League Final[8];
7. Rafting on Zambezi River[9].

Given that each video consists of a 360º image, each can be divided in 6 views representing an orientation of said video, as seen in Figure 2.1. This makes the total needed viewing length of each video to be substantially higher than if viewing a "normal" one view video(non 360º). Adding to this, it was requested that there would be at least 20 viewings per individual view, in order to have a large enough dataset on each view for later training the Neural network.

Were every individual view to be viewed by 20 different users, the necessary total number of users, respecting the previous conditions, and having each user viewing about 15m, we would get the numbers as seen in Table 4.2.

As seen in Table 4.2, with the previous named conditions, we would require 326 view sessions, and a total viewing time of 81h26m in order to fulfil the requested 20 viewings per view. This however, did not feel achievable during the time given to complete this dissertation. As such, several changes were suggested and made.

First of all, the *Real Madrid - Juventus Champions league final* video was discarded, namely as it was the longest video but also due to its visual content not being very interesting or noteworthy

[3]https://www.youtube.com/watch?v=Za78gJU5Tfc
[4]https://www.youtube.com/watch?v=H6SsB3JYqQg
[5]https://www.youtube.com/watch?v=a61sgtnhrjA
[6]https://www.youtube.com/watch?v=xSiv4TkfSOE
[7]https://www.youtube.com/watch?v=pwivE6bvD8w
[8]https://www.youtube.com/watch?v=JTWdUBIvOFY
[9]https://www.youtube.com/watch?v=2S3ufI6atDg

Table 4.2: Initial planned viewing strategy

Video	Length (min:s)	# of views	Length * # of views (m:s)
Mercedes-Benz Drive	3:18	6	19:48
Blue Angels Fight	8:12	6	49:12
Amusement Ride	6:06	6	36:36
Ancient City of Petra	6:58	6	41:48
Carnival of Venice	5:53	6	35:18
Real Madrid - Juventus	8:27	6	50:42
Rafting	1:49	6	10:54
	Sum of all Lengths:		244:18
	Sum of Lengths * 20:		4886:00
	Sum of Lenghs * 20 (h:m):		81:26
	Users needed (15m view time):		326

different from others, as only one of its views at any given time was facing the football field, while the camera position remained mostly static. This meant that the other views were facing: stadium benches with people (left and right and back views), the ground (bottom view) and the stadium ceiling (top view), for majority of the duration of the video.

Secondly, all the top views from all other videos, excluding the Blue Angels video were discarded, as most of the time the image showed nothing but sky without many visual elements, while in the exception, the Blue Angels video, given the various acrobatic movements the airplanes do, this view remained interesting and with varied visual content.

Thirdly, the bottom view of the Blue Angels video was discarded, as it showed nothing but the pilot seat, and a few segments of the Carnival of Venice bottom view were discarded, as they were usually static segments facing for example water, or ground.

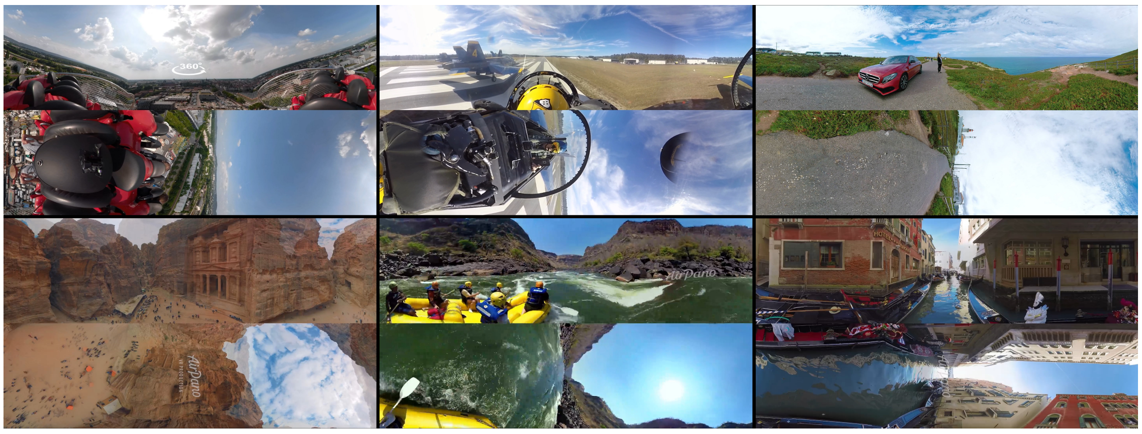


Figure 4.14: Examples of frame extracted from the 6 used 360º videos

Finally, about half of the Amusement Ride Length was cut, as the footage ended up being repetitive, given that the PoV is from within a seat in the ride, which consists on the user spinning around a center point. On top of this, it was decided to increase the view time of each user to

Table 4.3: Adjusted viewing strategy

Video	Length (min:s)	# of views	Length * # of views (m:s)
Mercedes-Benz Drive	3:18	4	13:12
Blue Angels Fight	8:12	5	41:00
Amusement Ride	6:06	3	18:18
Ancient City of Petra	6:58	5	34:50
Carnival of Venice	5:53	4	23:32
Real Madrid - Juventus	8:27	0	0:00
Rafting	1:49	5	9:05
	Sum of all Lengths:		139:57
	Sum of Lengths * 20:		2799:00
	Sum of Lengths * 20 (h:m):		46:39
	Users needed (20m view time):		140

20 mins, keeping in mind what was researched in a paper previously mentioned[28], not to over strain the users with videos with large lengths.

Having adjusted the number of views according to the previous mentioned changes, we get a more achievable number, 140, and a total run time of 47h. It is noteworthy that this number doesn't mean that 140 users are needed, but instead that 140 viewing sessions are needed, as there will be multiple videos, or playlists, throughout which the views will be organized.

As such, seven unique 20 minute playlists were created, consisting of views from each video stitched together, allowing for a user to theoretically watch all seven playlists without viewing the same content twice.

To make cutting segments easier, each video was divided in shorter segments, all around 1m10s long, and all videos were stretched to an aspect ratio of 16:9, as WMP doesn't support stretching or altering aspect ratio in any form, displaying all videos at their native resolutions, which caused the appearance of black bars, had they not been edited. All video edits and cuts were done through Virtual Dub (Section 4.1.4).

4.6 Data Capture Setup

Given that the data capture would happen both at the INESC TEC premises, as well as at the author's home, the setup required that the same conditions would be present in both locations.

As such, in order to keep the conditions ideal for recording, keeping in mind the points mentioned in Chapter 4.2, a ring light was placed behind the top of the monitor, to remove any possible shadows or difference in light. Also, the same 24 inch monitor was used in both locations, and all users remained less than the maximum distance away from the monitor, as seen in Table 4.1, at a comfortable distance. Finally, after some experimentation, the camera was placed on the lower side of the monitor, looking at the user from below, as visible in Figure 3.1, as it seemed to give the best results.

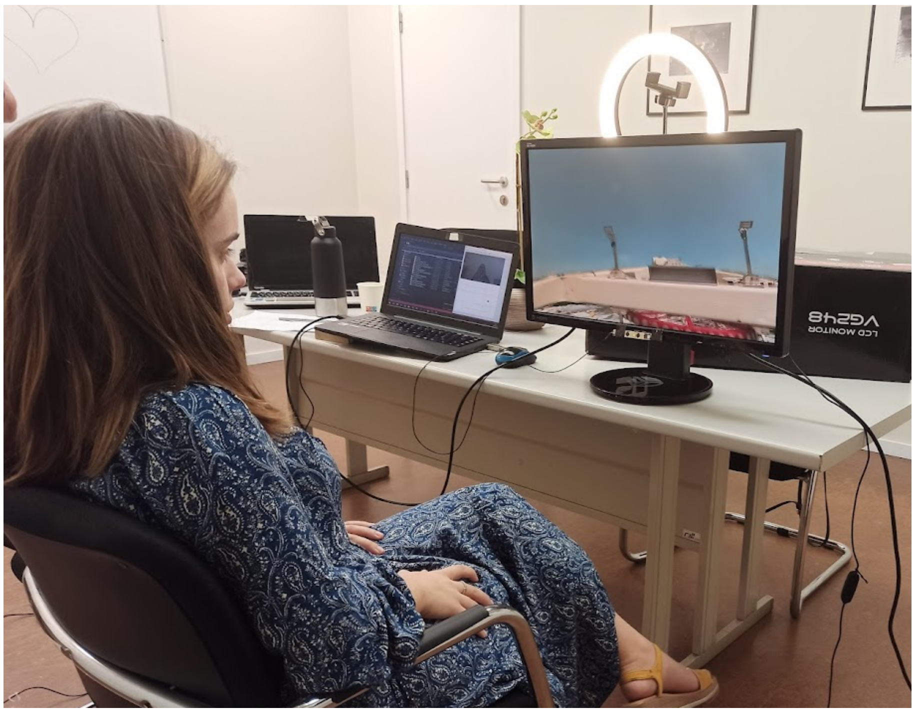


Figure 4.15: Testbed setup during data capture at INESC TEC

Left of the monitor is the laptop running the software and with a partialview of the head tracking program, so we can make sure the gaze tracking is working as intended, as visible in Figure 4.15.

4.7 Data Capture

As mentioned before in Section 4.5, a good amount of users would be needed to get the full 140 viewings. In order to gather volunteers, a mass email was sent to the INESC TEC community, explaining the premise of the study and inviting them to participate and help gather data.

Alongside this, friends, family and colleagues were also invited to take part. There was a need for the setup to be able to be mounted and dismounted fairly quickly, thus making it possible to assemble it in the INESC TEC office, where Figure 4.15 was taken, as to receive volunteers from the community and colleagues, as well as assemble the setup at the author's home in order to receive family and friends.

To manage the large number of volunteers, a schedule was created where users could select time blocks according to their availability, so that each person would get their opportunity to see as many videos as comfortably possible without feeling pressure of a line of people waiting for their turn.

At some times, namely when there were no scheduled viewings, researchers and students from INESC TEC's CTM (Centre for Telecommunications and Multimedia) department, where the testbed was located, were invited to go through the experiment. In some occasions, curious members of the INESC TEC community would question what the running experiments were, and would end up running the experiment as well, contributing to achieving the objective number of viewings.

In total, 46 volunteers contributed towards running the experimenting, with some viewers viewing multiple playlists, allowing the obtained dataset to be constructed.

4.8 Data Processing

All the data processing steps for the generation of scatter plots and heatmaps were done in Python and Jupyter Notebooks.

Each run of the developed solution creates both a heatmap and scatter plot video for a specified video. For ease of understanding, in this chapter, both of them will be referred to as plots.

Once the heatmaps, in video file, are formed, Sony Vegas was used to overlay them with their corresponding video, to make figures such as Figure 3.3. Each run of the solution creates videos of each plot for the corresponding video or view. The dataset is composed by 140 different logs and, as previously stated, contains certain parameters, visible in Section 3.2 and each of its logs has the format of a plain text file (TXT), as visible in Figure 4.11.

Initially, the following variables were loaded into the program: The video file name, for which the plots will be generated, the framerate we wanted our video plots to have, and the path to the location of the dataset logs.

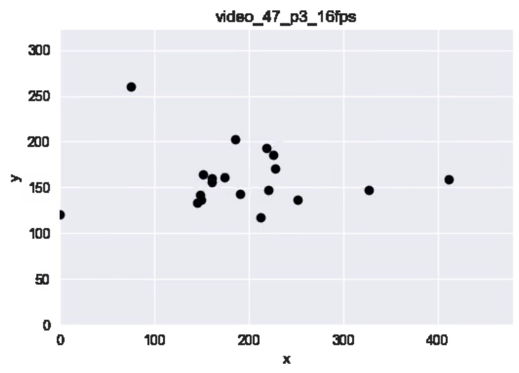
Following this, the program then creates a CSV file and goes through all the logs in the folder specified, reading only those that contain the intended video. The final CSV file contains the following data, regarding said video:

- Video Timestamp;
- X Value;
- Y Value;
- User Name.

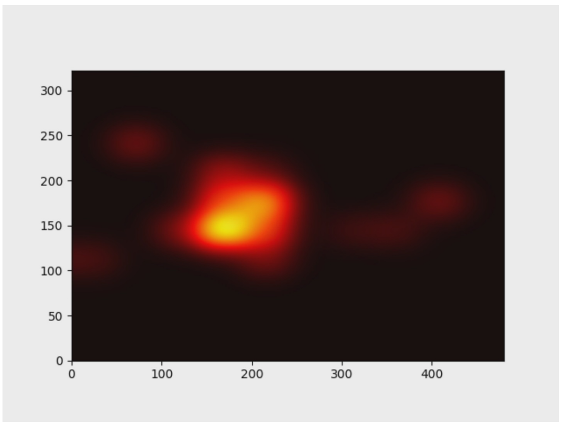
Afterwards, the CSV is converted onto a DataFrame format, from the Python pandas library, and the data is is ordered by timestamp. The reason behind the change of formats is to allow faster and easier access to the large volume of data.

Knowing the video's duration and the number of logs we have in the previous DataFrame, and keeping in mind the requested framerate, we are able to calculate the output video's amount of frames, and the time interval between two frames.

Following this, a list of lists is created, where each list within a list contains all X,Y points existing within the time interval between two frames, ordered by order of appearance. In other words, each list within a list contains all visual points in a frame. Figure 4.16a corresponds to a single frame where each dot coincides with a single user's gaze.



(a) Scatter Plot



(b) Heatmap

Figure 4.16: Example of the obtained plots from the Blue Angels video

With this data sorted, each frame of the scatter plot was created by generating a *matplotlib.pyplot.scatter* plot type, taking a list within the list previously mentioned, and all frames were merged together using the FFMpegWriter library, at the specified framerate. This generated a video as long as the original video.

For the heatmap or saliency map, a similar procedure was followed, where each frame was created using a *numpy.histogram2d*, and smoothed using *matplotlib's imshow* method, to gain the appearance visible in Figure 4.16b. For generating a video, *matplotlib.animation.FuncAnimation* was used.

To get images such as the one in Figure 3.3, Sony Vegas was used to display both the heatmap and the video simultaneously, stretching the heatmap over the video, and applying a chroma keyer effect over the heatmap's black background, in order to turn it transparent.

Chapter 5

Results

5.1 User Analyses

Overall, there were a total of 46 volunteers participating in the data gathering stage of this dissertation. The statistics can be seen in Table 5.1 and are described below.

The sample was very balanced, being that 24 volunteers were men and 22 were women. Thus, we can gather equal insight into what both gender groups find most appealing in the views that were presented to them.

Regarding the volunteer's ages, the range is between the oldest volunteer, being 82 years old, and the youngest volunteers being 22. The average age was 33, while the mode age was 25, having 12 users with this age. 30 of the 46 users (65%) were between 22 and 26 years old.

Additionally, each user watched on average 3 playlists, translating into an average watch time of 60 min per user, as the average playlist length was of about 20 mins.

Table 5.1: Statistics of participating users

Total # of users:	46
Male:	24
Female:	22
Avg. playlist duration:	20 min
Avg. playlists viewed/user:	3
Avg user view time:	60 min
Youngest user:	22 years
Oldest user:	82 years
Avg. user age:	32.74 years
Mode user age:	25 years

5.2 Result Analyses with overlaid heatmaps

At the end of data capture, we were able to reach the 140 viewings set out in Section 4.5. For every instance of every video, 20 gazes from different users can be observed.

After generating heatmaps and overlaying them onto their respectful videos, we were able to obtain a very eye friendly and easy to comprehend visualisation of the gazes for each scene, as seen in Figure 5.1.

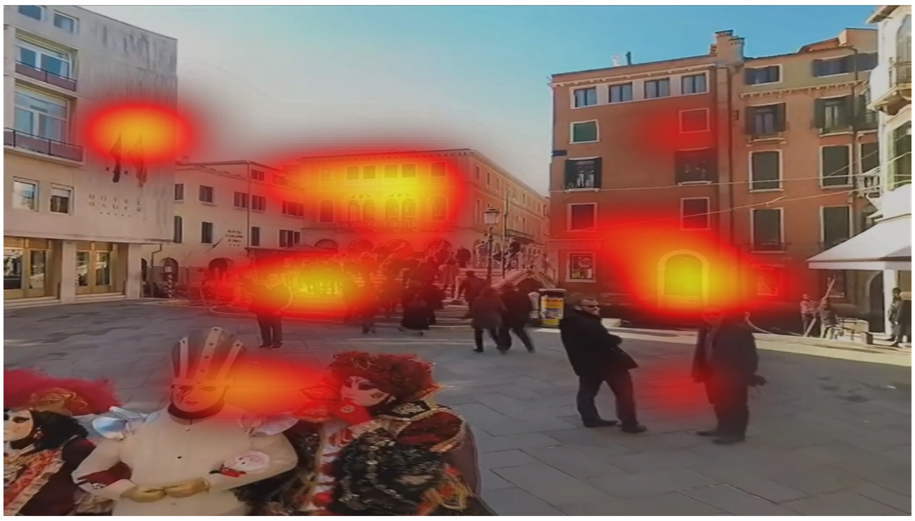
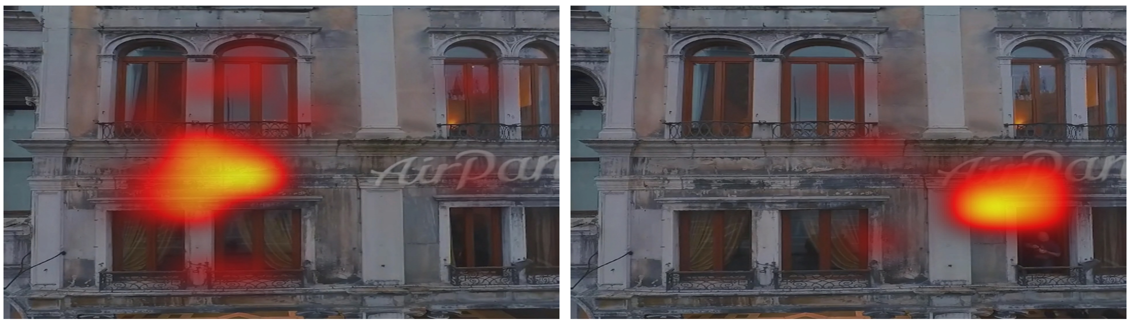


Figure 5.1: Example of a frame from the Carnival of Venice video with an overlaying heatmap

From a merely visual perspective, it was easy to take notice of what visual queues or movements caught the users' gaze. For example, in scenes that were initially mostly static, and where a movement or change happens, it is noticeable that the majority of gazes end up converging in the area where a change was introduced, such as in Figure 5.2. Initially, the scene is mostly static and user gazes are moving mostly randomly around it as viewed in Figure 5.2a. However, when a change is introduced, such as, in this case, a man appearing in a window, the majority of users' gaze tend to focus on the area where a change was introduced, as viewed in Figure 5.2b.



(a) Initial frame (b) Later frame

Figure 5.2: Comparison of the area of attention between two mostly static frames separated by a few seconds, where a change happens in the latter, in the Carnival of Venice video

Another noteworthy point is that users are usually drawn at the presence of people in a scene, as visible in Figure 5.3.

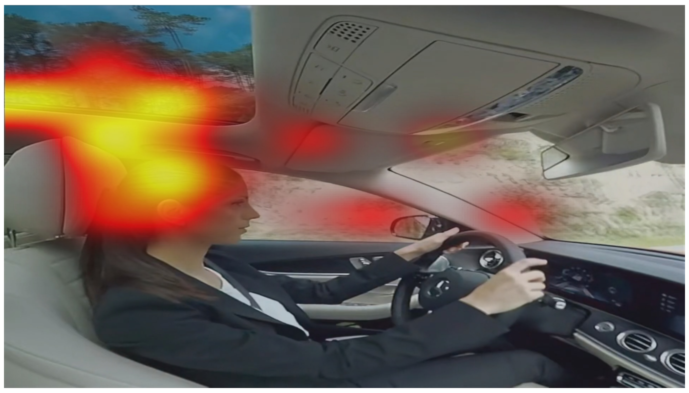
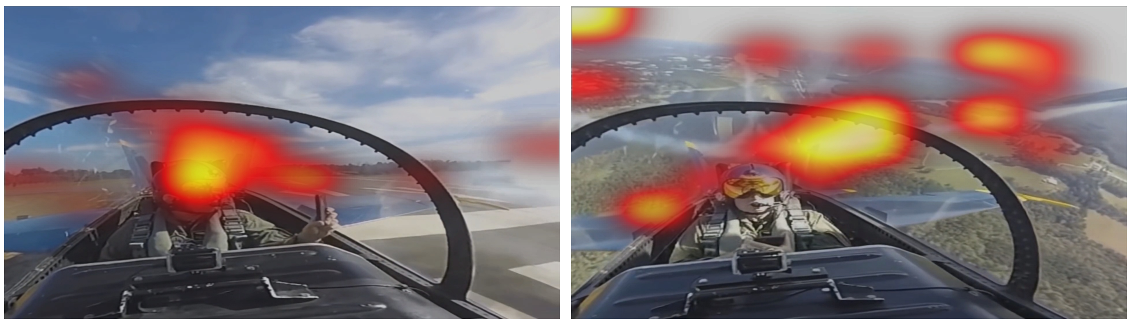


Figure 5.3: Example of a frame from the Mercedes-Benz Drive video with an overlaying heatmap

In all the videos that feature movement, or background changes, it is noticeable that users are usually drawn into these dynamic changes. For example, in the Blue angels video, on the view facing the co-pilot, initially the airplane is still on the runway, static, about to take off. Here it is prominent that the overall users' gaze focus on the man on the cockpit, as viewed in Figure 5.4a. Later on, however, as the airplane is already above-ground, the gaze from users tends to move in a more spread out fashion, namely focusing on the constantly changing background of the frame, as visible in Figure 5.4b.



(a) Initial frame (b) Later frame

Figure 5.4: Comparison of the area of attention between two frames separated by a few seconds in the Blue Angels video

Curiously, as noted in[27], it seems that throughout the videos, the general central point of interest tends to be similar among users. From looking at these segments and from the noticeable changes in users' gaze, we are confident that the results obtained in this work will serve as robust and useful training data for the proposed NN, or any other ML models that uses it as training set.

Chapter 6

Main Contributions and Conclusions

This chapter concludes this report. A discussion on the main contributions from this work is given, as well as an overall overview, conclusion, and insight into possible future work.

6.1 Main Contributions

The analysis of visual behaviour from users during content viewing sessions of media is gradually more appealing for the training and evaluation of Artificial Intelligence models. Their relevance is in such a way significant that there is a rising demand in the international market of data, in the form of datasets, for a wide array of purposes[1], usually meant for a betterment of a companies offerings and tailoring their products to their costumers.

The majority of existing datasets, however, as mentioned in Chapter 1.2, are associated to the use of Head Mounted Devices, such as VR goggles, which in turn are complex, expensive, and hardware demanding. The available datasets in the field of multi-view content, the type of content focused on this work, is relatively lacking. As such, the output of this dissertation will be contributing to increasing the overall available usable training data for multi-view content, in a market where similar work is scarce.

6.2 Conclusions

In this dissertation, a new dataset of multi-view content for 6 omnidirectional videos is presented, with the associated gaze fixation and head positioning data of 20 users for each view, in the form of saliency/heat maps, scatter plots, and regular logs. This data was obtained after processing the raw facial positioning data of 46 different users, on a setup testbed with an Intel Realsense F200 camera recording each user's gaze.

[1]https://www.youtube.com/watch?v=ConsSlIf6n4

Along with the dataset, the tools to record a dataset of this type for any video were developed, as well as solutions to transform the output into graphical data, in order to aid the research community wanting to expand the available multi-view content pool.

6.3 Limitations

The approach presented in this dissertation has a few limitations considering its mode of operation, as named below:

- User Behaviour: The performance and reliability of each user's gaze is dependent on that user's behaviour. The program records the user's head pose, not their eye-sight, therefore it is requested for the user to follow their eye sight with their facial position. The conduct of each user during their viewing experiment will depend on whether their gaze data is reliable.
- Length of used videos: Initially, it was planned for each experiment to run for 10-12 minutes, a relative short time span. With the extension of each experiment to an average of 20 minutes, and the length of some videos being relatively long, user behaviour might have some changes while watching a scene, depending if they have been watching for a long time previously, as that might make them stop losing focus and start wandering their gaze.
- Quality of used videos: The original 360º videos are of high quality, and when viewing them in a VR setting, quality is not an issue. However, with the division of each video in 6 views, stretching each from their original aspect ratios to 16:9 and then displaying one and all in a 24 inch full HD monitor, the lack of image quality is noticeable.
- Multitude of steps from recording to results: In order for the work developed to function and to obtain datasets, several different steps must be taken, from recording user gaze, to producing heatmaps, to overlaying them on top of the original video.

6.4 Future Work

Overall, the solution obtained in this work was quite satisfactory, with fairly reliable performance. However, as mentioned before, a few limitations exist in certain areas, allowing room for improvements to be made to the work developed in this dissertation. These are described below:

- Eye-tracking in opposition to pose tracking: As previously mentioned, this work depends on each user following their eye movement with their facial movement. As such, it should be considered to replace the pose tracking method with an eye-tracking solution, which would be a relatively manageable task, given the already available Intel Realsense SDK eye-tracking functions, compatible with the used camera. It is a fact that *SmoothMV* uses facial pose tracking in order for user's to switch their view in a multi-view content, however, in this data capture, the ability to switch views as proposed in *SmoothMV* is not present, therefore, it might make more sense to capture the eye-sight of the user for this data.

- Better training data: The videos displayed for this dataset contained a diverse and rich multitude of visual content. However, at times they could be considered too lengthy, displaying similar scenes for long periods of time. Using different videos that would still be visually rich, but that spanned for shorter periods of time could benefit not only in making the experiment more interesting and engaging for the user, but also use the saved time to increase the number of visual gazes per scene by having more, but shorter, experiments.
- Uniform Solution: For a greater QoE of the operating user, a one-program-does all could be developed, where both the gaze capture and heatmap generation would work sequentially.

References

[1] Tiago Soares da Costa, Maria Teresa Andrade, and Paulo Viana. SmoothMV: Seamless Content Adaptation through Head Tracking Analysis and View Prediction. *International Workshop on Immersive Mixed and Virtual Environment Systems (MMVE'21)*, 2021.

[2] Wiso. Neural network example. https://commons.wikimedia.org/wiki/File:Neural_network_example.svg/, October 2008.

[3] Tiago Costa, Maria Andrade, and Paula Viana. Predictive multi-view content buffering applied to interactive streaming system. *Electronics Letters*, 55, 05 2019.

[4] Chenglei Wu, Zhihao Tan, Zhi Wang, and Shiqiang Yang. A dataset for exploring user behaviors in vr spherical video streaming. *Proceedings of the 8th ACM on Multimedia Systems Conference*, 2017.

[5] Microsoft. Intel realsense technology. https://www.intel.com/content/www/us/en/architecture-and-technology/realsense-overview.html, 2015.

[6] Anthony Guay. Heatmap overlay on top of a live video example, retinad vr. https://www.youtube.com/watch?v=8_l30wP2a2I, 2015.

[7] Intel. Intel realsense face analysis tutorial. https://www.intel.com/content/dam/develop/external/us/en/documents/face-tracking-843462.pdf.

[8] Bill Gates. How i work: Bill gates. https://money.cnn.com/2006/03/30/news/newsmakers/gates_howiwork_fortune/index.htm, April 2006.

[9] Ivan Berger. The virtues of a second screen. https://www.nytimes.com/2006/04/20/technology/the-virtues-of-a-second-screen.html, April 2006.

[10] Hank Leukart. Official doom faq, section 9.2. https://www.gamers.org/docs/FAQ/doomfaq/sect2.html#9-2, 1994.

[11] Intel realsense f200 price on. https://pt.aliexpress.com/item/32800823462.html. Accessed: 17/06/2022.

[12] Tabitha Baker. How much is a vr headset. https://www.gamesradar.com/how-much-is-a-vr-headset/, June 2022.

[13] Jens-Rainer Ohm. Mpeg developments in multi-view video coding and 3d video. https://tech.ebu.ch/docs/events/3dtv09/presentations/ebu_3dtv09_ohm.pdf, April 2009.

[14] Jiangbo Lu, Hua Cai, Jian-Guang Lou, and Jiang Li. An epipolar geometry-based fast disparity estimation algorithm for multiview image and video coding. *Circuits and Systems for Video Technology, IEEE Transactions on*, 17:737 – 750, 07 2007.

[15] Shailendra Kumar, USMANI T, Syed Saeed, and Naimur Kidwai. A comparative analysis of advance three dimensional video coding for mobile three dimensional tv. *International Journal of Electronics and Communication Engineering (IJECE) ISSN (E) 2278-991X*, 3:59–64, 09 2014.

[16] Seif Allah El Mesloul Nasri. *Multiview coding and compression for 3D video*. Theses, Université Badji Mokhtar - Annaba (Algérie), July 2018.

[17] Bruhanth Mallik and Akbar Sheikh Akbari. Hevc based multi-view video codec using frame interleaving technique. In *2016 9th International Conference on Developments in eSystems Engineering (DeSE)*, pages 181–185, 2016.

[18] Engin Kurutepe, M. Reha Civanlar, and A. Murat Tekalp. Selective streaming of multiview video for head-tracking 3d displays. In *2007 IEEE International Conference on Image Processing*, volume 3, pages III – 77–III – 80, 2007.

[19] Ed Burns. In-depth guide to machine learning in the enterprise. https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML/, March 2021.

[20] MICHAEL COPELAND. What's the difference between artificial intelligence, machine learning and deep learning? https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/, July 2016.

[21] Andrew Ng. Stanford: Supervised machine learning: Regression and classification. https://www.coursera.org/learn/machine-learning/.

[22] Miguel Fabian Romero Rondon, Lucile Sassatelli, Ramon Aparicio-Pardo, and Frédéric Precioso. TRACK: A Multi-Modal Deep Architecture for Head Motion Prediction in 360-Degree Videos. In *ICIP 2020 - IEEE International Conference on Image Processing*, Abu Dhabi / Virtual, United Arab Emirates, October 2020.

[23] Xianglong Feng, Yao Liu, and Sheng Wei. Livedeep: Online viewport prediction for live virtual reality streaming using lifelong deep learning. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 800–808, 2020.

[24] Robert Keim. Training datasets for neural networks: How to train and validate a python neural network. https://www.allaboutcircuits.com/technical-articles/training-dataset-neural-network-how-to-train-validate-python-neural-network January 2020.

[25] Xavier Corbillon, Gwendal Simon, Alisa Devlic, and Jacob Chakareski. Viewport-adaptive navigable 360-degree video delivery. In *2017 IEEE International Conference on Communications (ICC)*, pages 1–7, 2017.

[26] Erwan J. David, Jesús Gutiérrez, Antoine Coutrot, Matthieu Perreira Da Silva, and Patrick Le Callet. A dataset of head and eye movements for 360° videos. In *Proceedings of the 9th ACM Multimedia Systems Conference*, MMSys '18, page 432–437, New York, NY, USA, 2018. Association for Computing Machinery.

[27] Robert Goldstein, Russell Woods, and Eli Peli. Where people look when watching movies: Do all viewers look at the same place? *Computers in biology and medicine*, 37:957–64, 08 2007.

[28] David Rosengrant, Doug Hearrington, and Jennifer L O'Brien. Investigating student sustained attention in a guided inquiry lecture course using an eye tracker. *Educational Psychology Review*, 33:11–26, 2020.

[29] Jesús Gutiérrez, Erwan J. David, Antoine Coutrot, Matthieu Perreira Da Silva, and Patrick Le Callet. Salient360! - visual attention modeling for 360º content. `https://salient360.ls2n.fr/datasets/training-dataset/`, 2018.

[30] Timen Ten Harkel, Caroline Speksnijder, F. Van der Heijden, Carien Beurskens, Koen Ingels, and Thomas Maal. Depth accuracy of the realsense f200: Low-cost 4d facial imaging. *Scientific Reports*, 7, 11 2017.

[31] Microsoft. C# programming language. `https://docs.microsoft.com/en-us/dotnet/csharp/`, 2000.

[32] Bjarne Stroustrup. C++ programming language. `https://www.cplusplus.com/`, 1985.

[33] Microsoft. Visual basic programming language. `https://docs.microsoft.com/en-us/dotnet/visual-basic/`, 1991.

[34] James Gosling. Java programming language. `https://docs.oracle.com/javase/7/docs/technotes/guides/language/`, 1995.

[35] Guido van Rossum. Python programming language. `https://www.python.org/`, 1991.

[36] John D. Hunter. Matplotlib python library for visualization. `https://matplotlib.org/`, 2003.

[37] Avery Lee. Virtualdub software. `https://www.virtualdub.org/`, 2000.

[38] Microsoft. Windows media player sdk. `https://docs.microsoft.com/en-us/windows/win32/wmp/windows-media-player-sdk`, 1998.

[39] Microsoft. Microsoft visual studio. `https://visualstudio.microsoft.com/`, 1997.