Universidad Zaragoza
1542

Master's Thesis

Modeling Human Visual Behavior in Dynamic 360º Environments

Author

Edurne Bernal Berdún

Supervisor

Daniel Martín Serrano

Co-supervisor

Belén Masiá Corcoy

Master in Robotics, Graphics and Computer Vision
Escuela de Ingeniería y Arquitectura
2022

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe remitirse a seceina@unizar.es dentro del plazo de depósito)

D./Dª. Edurne Bernal Berdún ,

en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de Estudios de la titulación de Máster Universitario en Robótica, Gráficos y Visión por Computador (Título del Trabajo)

Modeling human visual behavior in dynamic 360º environments

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 27/01/2022

BERNAL BERDUN EDURNE - 73134670Y

Firmado digitalmente por BERNAL BERDUN EDURNE - 73134670Y
Fecha: 2022.01.27 21:15:27 +01'00'

Fdo: Edurne Bernal Berdún

Abstract

Virtual reality (VR) is rapidly growing: Advances in hardware, together with the current high computational power, are driving this technology, which has the potential to change the way people consume content, and has been predicted to become the next big computing paradigm. However, although it has become accessible at a consumer level, much still remains unknown about the grammar and visual language in this medium. Understanding and predicting how humans behave in virtual environments remains an open problem, since the visual behavior known for traditional screen-based content does not hold for immersive VR environments: In VR, the user has total control of the camera, and therefore content creators cannot ensure where viewers' attention will be directed to. This understanding of visual behavior, however, can be crucial in many applications, such as novel compression and rendering techniques, content design, or virtual tourism, among others.

Some works have been devoted to analyzing and modeling human visual behavior. Most of them have focused on identifying the content's regions that attract the observers' visual attention, resorting to *saliency* as a topological measure of what part of a virtual scene might be of more interest. When consuming virtual reality content, which can be either static (i.e., 360° images) or dynamic (i.e., 360° videos), there are many factors that affect human visual behavior, which are mainly associated with the scene shown in the VR video or image (e.g., colors, shapes, movements, etc.), but also depend on the subjects observing it (their mood and background, the task being performed, previous knowledge, etc.). Therefore, all these variables affecting saliency make its prediction a challenging task.

This master thesis presents a novel saliency prediction model for VR videos based on a deep learning approach (DL). DL networks have shown outstanding results in image processing tasks, automatically inferring the most relevant information from images. The proposed model is the first to exploit the joint potential of convolutional (CNN) and recurrent (RNN) neural networks to extract and model the inherent spatio-temporal features from videos, employing RNNs to account for temporal information at the time of feature extraction, rather than to post-process spatial features as in previous works. It is also tailored to the particularities of dynamic VR videos, with the use of spherical convolutions and a novel spherical loss function for saliency prediction that work on a 3D space rather than in traditional image space. To facilitate spatio-temporal learning, this work is also the first in including the optical flow between 360° frames for saliency prediction, since movement is known to be a highly salient feature in dynamic content.

The proposed model was evaluated qualitatively and quantitatively, proving to outperform state-of-the-art works. Moreover, an exhaustive ablation study demonstrates the effectiveness of the different design decisions made throughout the development of the model.

Acknowledgements

I would like to thank my supervisors, Daniel Martín y Belén Masiá, for the time they have invested in this project and for giving me the opportunity to work on a research group. Your advice and support have been of vital help during all these months. I am looking forward to continue working and learning from you.

I would also like to thank all the members of *Graphics and Imaging Lab* for the warm welcome I have received, and for sharing with me the pleasant working atmosphere they have created.

Finally, I would like to thank my family for their unconditional support, guiding me in the most difficult moments, and celebrating with me every little victory.

Contents

List of Tables

List of Figures

Acronyms

CC	Correlation Coefficient
CNN	Convolutional Neural Network
ConvLSTM	Convolutional Long Short-Term Memory
DNN	Deep Neural Network
DL	Deep Learning
FoV	Field of View
GAN	Generative Adversarial Network
GB	Gigabyte
GCN	Graph Convolutional Network
GPU	Graphics Processing Unit
HMD	Head Mounted Display
KLDiv	Kullback-Leibler Divergence
LSTM	Long Short-Term Memory
NSS	Normalized Scanpath Saliency
MSE	Mean Squared Error
RNN	Recurrent Neural Network
RGB	Red Green Blue
SLIC	Simple Linear Iterative Clustering
SIM	Similarity Metric
SVM	Support Vector Machine
VR	Virtual Reality

1. Introduction

"Virtual reality is the use of computer technology to create the effect of an interactive three-dimensional world in which the objects have a sense of spatial presence" (Steve Bryson – NASA Ames). "In a virtual environment, patterned sensory impressions are delivered to the senses of the human participant through computer generated displays (visual, auditory, tactile and kinesthetic)" (Ellis, 1992). Nowadays, the most common virtual reality (VR) experience is the one provided by a head-mounted display (HMD). These devices create the notion of spatial presence by showing a slightly different image to each eye, simulating human's stereoscopic vision, and tracking head movement, allowing users to explore the whole 360° virtual environment that surrounds them. Besides the HMD, some controllers are often used as a tool to interact with the virtual environment, enhancing the final experience. Although VR was once relegated to a privileged minority, recent advances in computer graphics and computer vision, the technical improvement of displays, and the increasing computing power of devices, are fostering its development and allowing it to reach the consumer level.

Virtual reality, which has been usually associated with the entertainment industry, has proven to be a powerful tool in many other fields such as the manufacturing industry, online market, architecture, design, and education, among others. Although many advances have been made in VR, this new visual representation technology presents challenges and limitations that are still an open problem. Unlike traditional media, where the whole content is usually shown on a flat screen, VR content occupies the 360° space around the user. As in real life, only the part of the scene which falls into the observer's field of view is seen, and it is by eye and head movements that the remainder of the environment can be explored. Therefore, *the user takes control of the camera*, choosing what to observe. Due to this paradigm shift, most of the patterns and visual behaviors known for 2D images do not neccessarily hold for 360° immersive environments. This may be due to the feeling of presence that VR elicits, where the observer perceives and responds to VR simulations as if they were real.

Human visual behavior refers to the eye movements performed in response to a visual stimulus, which can be roughly categorized into fixations (i.e., the maintaining of the gaze on a single location), and saccades, (i.e., rapid jerky movements that take place between fixations). The two most common representations of visual behavior are: Saliency, the probability of each image's pixel to be observed; and scanpaths, the ordered gaze fixation of the observers' when shown an image. Usually, observers tend to fixate on the areas of an image that are of more interest, although each may present a radically different scanpath. Therefore, many approaches have focused on modeling which points in a scene are most likely to attract users' attention, resorting to *saliency* as a topological measure of the conspicuity of the elements of that scene, or in other words, the probability of each element to receive a fixation from the observer.

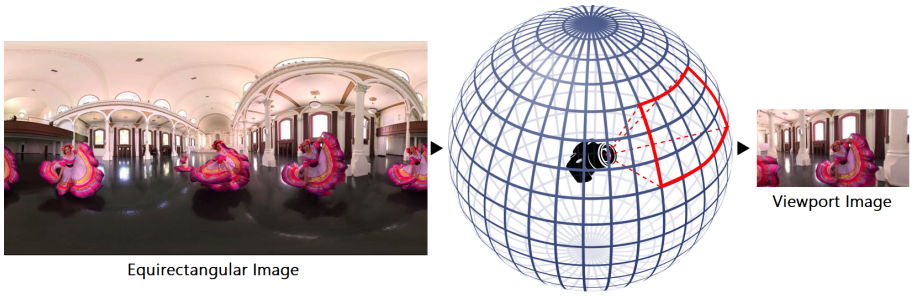


Figure 1.1: Example of a 360° scene projected into a 2D plane by an equirectangular projection (right) and a viewport projection (left) [4]. The viewport representation corresponds to the the reduce region of the whole 360° image observed at each instant (red contour in the sphere). Note the distortion introduced by the equirectangular projection, which is larger in the image regions corresponding to areas of the sphere that are close to the poles, and the practically null distortion in the viewport projection.

The study of fixation data from real observers gathered with eye-tracking devices has shown that there exists an inter- and intra- observer variability when exploring 360° visual stimuli [7]. This variability makes the task of visual human attention prediction challenging, but although the behavior of multiple observers in response to the same stimulus is rarely the same, they all share some common, inherent patterns [7]. Most of the observers are usually attracted by the most salient regions of a scene. Hence, modeling the saliency of an image can be considered a fundamental step towards understanding human visual behavior in VR environments. Modeling saliency in dynamic 360° content (i.e., videos) presents even additional challenges with respect to the static case, since features such as the movement of objects, actions, or the storyline of the video being watched have a huge impact on saliency, and therefore on human attention.

This project focuses on modeling saliency in dynamic cinematic content, which refers to 360° videos consisting of a succession of frames (i.e., 360° static images) that are intended to be shown in an HMD and usually pre-recorded. Many previous works have been devoted to saliency prediction in 360° content due to the numerous potential applications (see Martin et al's survey [8]:Sec.4), such as the assistance in VR content design: having a model that simulates human visual attention helps VR content creators get a sense of what the viewers' behavior will be like, adapting the experience to them. However, working with the spherical representation of this 360° content, either static or dynamic, can be cumbersome. Therefore, VR content is usually reprojected into 2D, facilitating visualization and manipulation (see Figure 1.1).

The aim of this master's thesis is to propose a novel saliency prediction model for dynamic 360° content. For this purpose, relevant state-of-the-art works devoted to this problem were reviewed, particularly those based on deep learning (DL). Deep learning techniques are rapidly growing in many disciplines, thanks to their ability to learn and model inherent characteristics of complex data. Previous works have shown that DL is able to learn spatio-temporal features from images, and have leveraged them to yield accurate saliency predictions. However, and after reviewing this body of literature, some limitations have been encountered.

Thus, the proposed model for saliency prediction in dynamic 360° content was specifically designed to to alleviate these limitations. It was evaluated over two different datasets in a quantitative and qualitative fashion, with results outperforming previous approaches in the state of the art. Additionally, an exhaustive ablation study supports the different design decisions made trough the development of the model.

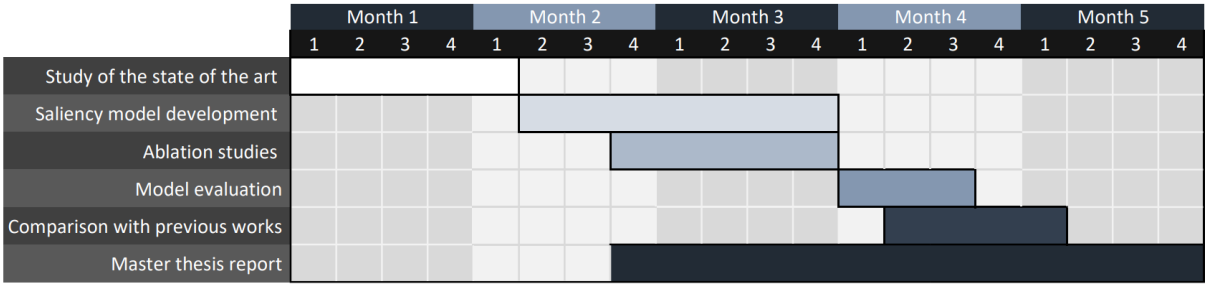


Figure 1.2: Gantt chart of the project schedule.

1.1 Objectives and Scope of the Project

The main objective of this master's thesis is the design, implementation, and evaluation of a saliency prediction model for dynamic 360° environments. To accomplish it, the following specific objectives are established:

- Study of the state of the art in saliency prediction in traditional and 360° content, for both static and dynamic content (Section 2).
- Design and implementation of a saliency prediction model for dynamic 360° content (Section 3).
- Evaluation of the model's performance and comparison to state-of-the-art models (Section 4.2 and 4.3).
- Ablation studies of the different modules that form the proposed model (Section 4.4).

This project is carried out in the *Graphics and Imaging Lab* research group, at the University of Zaragoza. The group's work focuses on *computer graphics*, conducting research in areas of physically realistic rendering, image processing, computational photography, virtual reality, or applied perception, among others. The group's work frequently involves the use of gaze tracking and deep learning techniques, as well as concepts such as saliency.

1.2 Planning and tools

The timeline followed to achieve the different objectives of this master thesis is shown in the Gantt chart of Figure 1.2. The time dedicated to the project is of 744 hours distributed over five months.

The saliency model was implemented in Python, using PyTorch, an open-source machine learning framework for research prototyping and production deployment. The datasets used for training and evaluation of the model were processed with the programming language and numeric computing environment MATLAB. The deep learning model was trained with a Quadro P5000 GPU with an integrated memory of 16 GB. GitHub was used as the version control tool.

2. Related Work

This section provides an overview of the state of the art in saliency prediction. It first presents a review of the literature about traditional 2D saliency, since it has established a strong basis later leveraged by models designed specifically for 360° content. Then, it focuses on the different existing approaches to address the particular challenges of saliency prediction in both static and dynamic 360° content.

2.1 Modeling visual attention in traditional 2D content

2.1.1 Heuristic approaches

In the last decades, many works have been devoted to saliency prediction in 2D content. The method proposed by Itti et al. [9] in 1998 can be considered the seminal work in this regard. They propose a heuristic approach in which they extract low-level features from the images, such as color, orientation, or high contrast areas, and linearly combine them to obtain a final saliency map. Several follow-up works [10, 11, 12] continued with this bottom-up strategy, exploring different hand-crafted features to improve the predicted saliency. Similarly, Haret et al. [13] develop a graph-based visual attention model that uses a Markovian approach to generate activation maps from feature vectors, and combine them into a single saliency map.

However, these bottom-up strategies based on low-level features often failed to capture the actual eye movements, thus novel approaches [14, 15, 16] arised to palliate this limitation. Judd et al. [15] proposed using low-level, middle, and high-level image features, combining them with a linear Suport Vector Machine (SVM). Borji et al. [14] combined the best previous bottom-up models with top-down semantic features (e.g., cars, animals, faces, etc.) using machine learning techniques such as SVM, regression, and AdaBoost classifiers.

2.1.2 Data-driven approaches

With the emergence of deep neural networks (DNNs), hand-crafted features were progressively replaced. DNNs, especially convolutional neural networks (CNNs), were yielding outstanding results in image classification tasks since they were able to extract automatically the most relevant features from the images. Numerous works [17, 18, 19, 20] followed these data-driven approaches in the field of saliency prediction, leading to unprecedentedly accurate models that

increasingly resembled human behavior.

One of the first models to exemplify the superiority of DNNs was DeepFix [17], a fully convolutional neural network that proved to surpass state-of-the-art results relying on previous bottom-up approaches based on hand-crafted features. Later, and considering that eye movements in free-viewing are related to both bottom-up and top-down visual factors, Liu et al. [18] presented a multiresolution convolutional neural network able to extract features at three different scales. However, Deep Learning techniques are usually limited by the enormous amount of data and time required for training. Some works like the ones from Junting et al. [20] or DeepGaze [19] alleviate these limitations by leveraging feature spaces obtained from networks specifically trained for other tasks such as image classification.

As new deep learning techniques have emerged, some of them have also been applied to saliency prediction. Particularly, architectures like generative adversarial networks (GANs) such as SalGAN [21], or long short-term memory (LSTM) recurrent networks, like SAM [22] and DSCLRCN [23], have achieved results outperforming other CNN approaches.

Although saliency prediction in 2D images has been extensively explored in recent years, the body of literature specifically addressing this topic in dynamic content (i.e., video), is quite narrow. Works such as the ones from Bak et al.[24, 25], or the one from Jianget et al. [26] also resorted to DNNs to extract spatial features, but those works were also designed to handle temporal information. Specifically, the two latter use recurrent neural networks, whereas the former obtains that information from optical flow estimation.

2.2 Modeling and predicting attention in static 360° content

The emergence and popularization of virtual reality have increased the interest, and even necessity, on understanding human behaviour in virtual environments. As detailed in the previous section, many works have focused on modeling human visual attention in 2D images, but these cannot be applied to this 360° content, since this content presents new challenges such as the distortion generated by reprojecting 3D content into 2D space, or the fact that observers cannot seen the whole 360° image at once. Nevertheless, the evolution of saliency prediction models for 360° content has been closely linked to that of traditional one, thus the same trends can be observed, where heuristic approaches established a baseline until the rise of deep neural networks.

2.2.1 Heuristic approaches

Heuristic approaches extract low and high hand-crafted features to determine areas of interest. Within them, it is possible to differentiate two approaches to deal with 360° images. The first one resorts to new methodologies to obtain those features in the new paradigm of 360° content, such as using viewport representations instead of the whole 360° image [27, 28], or segmenting the image into super-pixels by a simple linear iterative clustering (SLIC) before feature extraction [29].

The other approach is based on the application of 2D saliency models, such as those discussed in Section 2.1, to different reprojections of the 360° content (e.g., cube-map, equirectangular, etc.). The main limitation of this strategy lies in the distortion introduced by the sphere-to-plane projection. Works like Lebreton et al. [30] and Guilherme et al. [31] apply 2D saliency prediction approaches on the viewport images, and usually depends on the Field of View (FoV) of the used display. This representation presents little geometric distortion but limits the size of seen content. On the other hand, Startsev et al. [32] proposed the use of both equirectangular and cube-map projection to diminishing the distortion.

2.2.2 Data-driven approaches

Deep learning rapidly overtook heuristic approaches based on hand-crafted features as occurred with traditional image saliency prediction, due to their outstanding performance. Most of the works proposed for spherical images are adapted from, or strongly based on 2D image saliency models.

Assens et al. [33] propose an equirectangular projection and the model SaltiNet, which has the same structure as the CNN-based 2D image saliency network SalNet [20]. Except for the last layer which is intended for sampling scanpaths from the saliency maps, which are the succession of fixations that an observer would probably make when exploring the image.

On the other hand, Monroy et al. [34] also proposed a network whose core is based on SalNet [20] but using a cube-map representation. The obtained saliency, which is represented as cube-maps, jointly with the spherical coordinates of the pixels, are passed to a second CNN module that refines the prediction and provides the final equirectangular saliency map.

SalGAN [21] is adapted as well to 360° images by Chao et al. in SalGAN360 [35] by re-training part of the model with 360° images (i.e., fine-tuning). They resorted to a cube-map representation: Transferring each equirectangular image into multiple cube maps by rotating the center of the cube to multiple horizontal and vertical angles. SalGAN360 uses for training a new loss function that combines three typical saliency evaluation metrics: Kullback-Leibler divergence (KLD), Pearson's Correlation Coefficient (CC), and Normalized Scanpath Saliency (NSS).

Dealing with the distortion introduced by the sphere-to-plane projection has been one of the main challenges when using 360° content. The works aforementioned tried to diminish the effect of distortion in their final predictions by using representations such as cube-maps or viewport images, that yield lower distortion than equirectangular projections, but still deform the images. Nonetheless, representations with multiple images lead to discontinuities, redundant image boundaries, and repeated computations, which can hamper the prediction process.

Some state-of-the-art works [36, 5, 37] propose a different strategy to alleviate this issue. Instead of manipulating the projection of the content, they modified the neural network structure to account for 360° content particularities. Haoran et al. [36] represent the 360° images with a Geodesic ICOsahedral Pixelation (GICOPix) and present an alternative to CNNs with SalGCN, which is based on graph convolutional networks (GCNs). In SphereNet [5], rather than applying the conventional kernel directly in the equirectangular image, it is applied on spherical space, and then projected into de 2D plane. Consequently, the kernel used for convolution and pooling

presents the same distortion as the equirectangular image. Although initially conceived for classification and detection tasks, it has subsequently been used for saliency prediction in works such as the one from Martin et al. [3]. A similar approach is proposed by Yanyu et al. [37], but they use a circular crown kernel in the sphere. Additionally to their spherical convolutional and pooling layers, they also present a spherical Mean Squared Error (MSE) loss, which weights the MSE of each pixel with its solid angle, thus accounting for the distortion of equirectangular saliency maps.

2.3 Modeling and predicting attention in dynamic 360° content

Although dynamic content consists of a succession of images and could be considered an extension of saliency prediction in 360° images, the human visual behavior towards it differs greatly from the static case. Aspects such as the movement of the elements or the plot of the sequence influence the viewer's attention, causing the saliency of each frame to be influenced by previous frames.

For saliency prediction in dynamic 360° content, some models such as PanoSalNet [38], followed a naïve approach and presented a saliency CNN-based estimation network similar to those used for images. However, it just predicts an independent saliency map for each frame, thus not taking temporality into consideration. Due to this limitation, many posterior works resorted to a different approach, in which spatio-temporal features are extracted in order to make a more suitable prediction.

A frequent choice in state-of-the-art models is the use of Recurrent Neural Networks (RNNs), especially LSTMs, given their ability to retain temporal information. Cheng et al. [2] use a cube-map representation and propose a cube padding technique applicable to CNNs, which solves the image boundary problem. With it, they obtain the prior saliency maps of each frame with a CNN-based network, and then pass them through a convolutional LSTM (ConvLSTM) to capture relations between saliency in consecutive frames and get the definitive saliency prediction.

A more sophisticated architecture was proposed by Dahou et al. with AtSal [1], in which they combine the saliency maps obtained from two different streams (namely attention and expert streams). The core of the attention stream architecture is a convolutional encoder-decoder, which consist on consecutive convolutional layers that extract the image's features (i.e., encoder) followed by deconvolutional layers that recover the saliency map from them (i.e., decoder). This CNN encoder-decoder, jointly with an attention mechanism, are intended to extract global static saliency from the equirectangular frames. On the other hand, the expert stream tries to learn spatio-temporal saliency information from a cube-map representation. It is composed of two SalEMA [39] networks, whose architecture is based on CNNs and LSTMs, one for the poles and the other for the equator views of the cube map. Having a different network for the poles allows them to avoid overestimating fixations in these areas, as most of the saliency concentrates in the equator.

The network proposed by Yanyu et al. [37] for video saliency prediction includes spherical convolutions, pooling, and MSE loss as aforementioned, but also implements a spherical convolutional LSTM. The model's architecture is based on a U-Net [40], but with the inclusion of an LSTM in its bottleneck to capture temporal information.

This master thesis presents a data-driven approach for 360° video saliency prediction based on the spherical convolutions from SphereNet [5]. In line with state-of-the-art models like Cheng et al.'s [2], Yanyu et al.'s [37], or AtSal [1], LSTMs will be used to obtain temporal features. These previous works use traditional CNNs to extract the image features and then process them with LSTMs. However, traditional CNNs only base their predictions on the current frame, thus the features extracted could not be the most relevant ones given the time frame. To account for this limitation, the model presented in this master thesis proposes to use spherical convolutional LSTMs, namely Spherical ConvLSTMs, to directly learn the spatiotemporal features. Additionally, and taking as a reference works like the one by Bak et al. [24] for 2D video saliency, this work proposes using optical flow estimation together with the 360° images as input to the network, since it is considered to have a strong influence on visual human attention.

3. A Model for Saliency Prediction on 360° Videos

This section details the proposed model for saliency prediction on dynamic 360° content, which is based on an encoder-decoder architecture built over Spherical Convolutional LSTMs (Section 3.2). The input to the model is the RGB frames from the 360° video, and an estimation of the optical flow between frames to predict frame-wise saliency (Section 3.1.3). To further alleviate the effect of the distortion introduced by the equirectangular representation of 360° content, a novel spherical loss function is presented in Section 3.3.

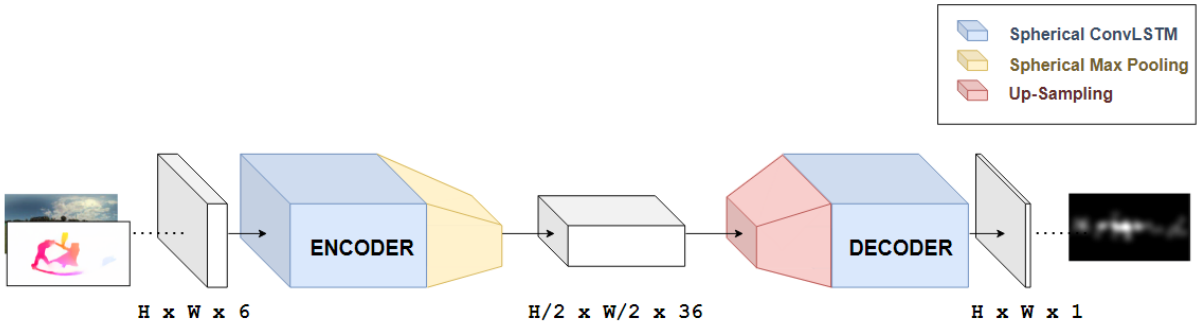
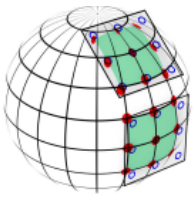


Figure 3.1: Network's encoder-decoder architecture: The encoder is composed of an Spherical ConvLSTM and a Spherical Max Pooling layer. It takes as input the concatenation of the RGB image and the optical flow of the frame. The decoder, formed by an Spherical ConvLSTM and an Up-Sampling layer, decodes the feature vector predicting the final saliency.

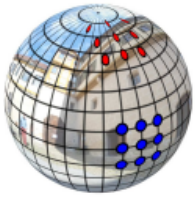
3.1 Theoretical Background

3.1.1 Spherical Convolutions

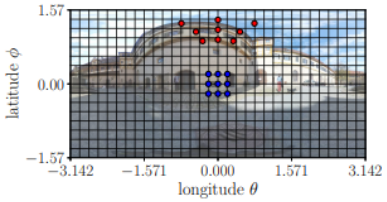
This work is built over the spherical convolutions specifically designed for equirectangular images presented by SphereNet [5], which are an adaptation of traditional convolutional layers (see Appendix A.3). These spherical convolutions were designed to handle the distortion of 360° images: The convolutional operations are projected from the 2D image domain to the surface of a sphere representing the 360° image. To account for real neighboring in spherical space, the kernel is defined as the projection on the sphere of a small patch tangent to its surface (Figure 3.2a). Thus, the kernel sampling pattern is distorted along with the image in equirectangular projection (Figures 3.2b and 3.2c).



(a) SphereNet Kernel tangent to the sphere.



(b) Kernel Pattern projected on the sphere. Kernel center on the equator (blue) and close to the pole (red).



(c) Equirectangular representation of the projected Kernel Pattern center on the equator (blue) and close to the pole (red).

Figure 3.2: Original figures from SphereNet [5] representing spherical convolutions.

3.1.2 Convolutional Long Short-Term Memory Cells

Dynamic videos contain elements that are likely to move, change, or disappear. These continuous variations in position, appearance, or illumination are considered temporal features which usually attract the observer's attention, being perceived as salient.

Convolutional Long Short-Term Memory (ConvLSTM) cells can extract and process temporal features from sequential data (e.g., videos) and infer temporal relations between them [41]. The fully-connected structures from traditional LSTMs (see Appendix A.4) are replaced in ConvLSTMs by convolutional layers, which are traditionally used in image processing due to their ability to infer spatial correlations from the input data. A ConvLSTM cell is composed of cell states c and hidden states h, which enable storing information through the processing of the points of the input sequence. The cell and hidden state at each timestep t are computed as follows:

$$\begin{aligned} i_t &= \sigma\left(W_i * [x_t, h_{t-1}] + b_i\right) \\ f_t &= \sigma\left(W_f * [x_t, h_{t-1}] + b_f\right) \\ o_t &= \sigma\left(W_o * [x_t, h_{t-1}] + b_o\right) \\ g_t &= \tanh\left(W_g * [x_t, h_{t-1}] + b_g\right) \\ c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\ h_t &= o_t \odot \tanh\left(c_t\right) \end{aligned} \tag{3.1}$$

where i, f, and o are the self-parameterized input, forget and output gates, W and b are learned weights and biases, x is an input data point, $*$ the convolutional operation, $\odot$ the Hadamard product, and σ the sigmoid activation function. The cell output is obtained from the last value of h. More details about the behaviour of LSTMs can be found in Appendix A.4.

3.1.3 Optical Flow Estimation

The movement of objects, people, or animals in 360° videos influences human visual attention [42, 43], and both their speed and direction can be relevant features to consider when predicting saliency. A representation of this movement is the optical flow, which in computer vision can be defined as a velocity field associated with image changes, yielding the relative movement

between objects and the camera.

The optical flow estimation between frames used in this master thesis is obtained with RAFT [44], a deep neural network trained on the Sintel Dataset [45], which has shown outstanding results on both synthetic and real images. RAFT stands out for its efficiency, even with high-resolution images, and strong generalization, posing a great advantage with respect to alternative methods for optical flow estimation. RAFT is inspired by optimization-based approaches. It is formed by an encoder that extracts a feature vector for each pixel of both frames. These feature vectors are passed through a correlation layer that computes the visual similarity between the pixels. Then, these similarities are used to update the optical flow estimation (which is initialized to zero) by means of a RNN-based network, mimicking the steps of an iterative optimization algorithm.

Although RAFT is designed to work on traditional 2D images, the provided optical flow estimation was evaluated and found to be sufficiently accurate (see Section 4.4.4), since most of the motion is going to be concentrated in the less distorted region of equirectangular images: the equator.

3.2 Architecture

The proposed model is based on an encoder-decoder architecture (see Appendix A.2). The encoder module extracts the spatiotemporal features from the frames of a 360° video, and the decoder module predicts each frame's saliency map from these spatiotemporal features.

Some state-of-the-art works for saliency prediction in 360° videos [37, 2, 38] have proposed similar encoder-decoder approaches using CNNs, which apply either traditional or spherical convolutions, for both the encoder and decoder. Additionally, they include a LSTM-based module after the encoder to infer the temporal relationships between encoded feature vectors. Although these approaches extract spatio-temporal features from the feature vectors, the encoding stage is done without any information from the previous steps.

In contrast, the model proposed in this thesis is based on the hypothesis that having temporal information at the time of feature extraction would allow obtaining more relevant features from dynamic content, leading to more accurate predictions. Therefore, the proposed encoder and decoder architectures are built over ConvLSTMs (see Section 3.1.2), but whose traditional convolutions have been replaced by the spherical convolutions proposed in SphereNet (see Section 3.1.1) to account for the distortion introduced by the equirectangular projection. The Spherical ConvLSTMs are used to encode and decode the feature vectors considering the information retained in their internal states (i.e., hidden and cell states) about previous frames, thus accounting for the temporal information at the time of feature extraction.

Additionally, the optical flow between frames (see Section 3.1.3), which is hypothesized to provide crucial information about the video saliency, is input to the network as an RGB image, where color encodes the displacement (direction and magnitude) of each pixel between consecutive frames.

Figure 3.1 shows the network's architecture diagram. The first module of the network's

architecture is the encoder, which is fed with the concatenation of a sequence $\mathcal{S}$ of frames f with length n, and the corresponding sequence $\mathcal{OF}$ of optical flow estimations o. Hence, the model's input $\mathcal{I}$ can be defined as:

$$\mathcal{I} = [\mathcal{S}, \mathcal{OF}] = \{f_0\ o_0, \cdots, f_t\ o_t, \cdots, f_n o_n\} \qquad t\ \epsilon\ [0, n] \tag{3.2}$$

where f and o are RGB equirectangular images of size WxHx3 (W and H are the width and height of the frames, and 3 the number of channels). The timestep of each element on the sequence is represented by t.

The encoder module is built with a Spherical ConvLSTM followed by a Spherical Max Pooling layer. The Spherical ConvLSTM takes at each timestep t the input vector $i_t = [f_t, o_t]$ with shape WxHx6, and outputs a hidden state with 36 channels (this size was empirically set). Then, to obtain the feature vector of the current frame, a Spherical Max Pooling layer downsamples the input along its spatial dimensions by taking the maximum value over a spherical kernel of size 3x3 for each of the 36 channels. This yields a final shape of the feature vector (i.e., bottleneck) of $\frac{W}{2}$x$\frac{H}{2}$x36.

The encoder is followed by the decoder module, built with a Spherical ConvLSTM and an Up-Sampling layer. The Spherical ConvLSTM decodes the feature vector into a single channel that represents the saliency map. Then, the Up-Sampling layer is applied to increase the spatial resolution to match the input spatial dimensions. Thus, the obtained final saliency map has a shape of WxHx1.

3.3 Loss Function

This model has been trained with a novel spherical weighted Kullback–Leibler Divergence (KLDiv) loss term, which refers to the KLDiv adapted to 360° content. The traditional Kullback–Leibler Divergence is a metric broadly used for saliency map comparisons, which measures the overall dissimilarity between two saliency maps that are considered as probability density functions and can be defined as follows:

$$KLDiv\,(G, P) = \sum_{i,j} G_{i,j}\ \log\left(\epsilon + \frac{G_{i,j}}{\epsilon + P_{i,j}}\right) \tag{3.3}$$

where $P_{i,j}$, $G_{i,j}$, are the predicted and ground truth saliency values at pixel (i, j), and ϵ is a regularization constant that determines how much zero-valued predictions are penalized.

However, this metric does not account for the distortion introduced by the equirectangular projection, obtaining inadequate scores for this type of content. This behavior is represented in the upper image of Figure 3.3, where for the same area on the sphere (i.e., yellow circle) different KLDiv values are obtained depending on its location. To overcome this limitation also present in the Mean Squared Error (MSE) loss proposed by Xu et al. [37], they applied a spherical weighting by the solid angle to compensate for the distortion. For an equirectangular image of shape m x n these weights can be computed as:

$$w_{i,j} = \frac{\sin(\Delta\varphi(j+1)) - \sin(\Delta\varphi j)}{\Delta\theta}$$
$$\text{where } \Delta\theta = \frac{\pi}{m} \; ; \; \Delta\varphi = \frac{\pi}{2n} \; ; \; i \in [0, n] \; ; \; j \in [0, m] \tag{3.4}$$

The proposed spherical weighted KLDiv loss employ to train the model applies this weighting to the traditional KLDiv, compensating for the distortion introduced by the equirectangular projection, since the contribution of each pixel to the KLDiv is proportional to its solid angle (see the bottom image of Figure 3.3). Therefore, the proposed spherical weighted KLDiv loss is defined as follows:

$$\mathcal{L}_{KLDiv}(G, P) = \sum_{i,j} w_{i,j} \; G_{i,j} \; \log\left(\epsilon + \frac{G_{i,j}}{\epsilon + P_{i,j}}\right) \tag{3.5}$$

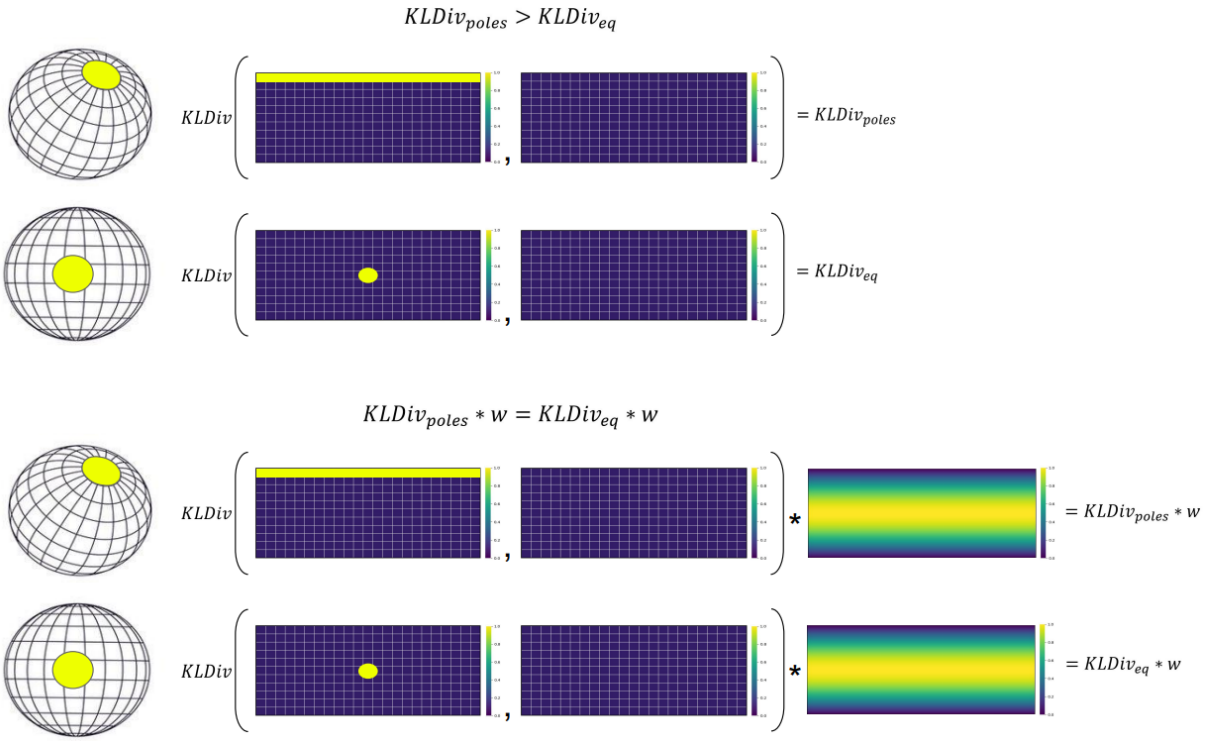


Figure 3.3: Graphical representation of the KLDiv metric behavior. The image represents the KLDiv and weighted KLDiv metrics obtained between two assumed saliency maps: the one obtained by projecting the spherical image with a salient area (yellow circle) and a map without any salient points. It can be seen that an equal area on the sphere (yellow circle) corresponds to different projected areas. Therefore, the traditional KLDiv (top image) is greater at the poles, since the projected yellow patch covers a larger number of pixels in the 2D representation, which contributes to the KLDiv score. However, when weighted by their solid angle, the contribution of each pixel to the KLDiv is compensated, obtaining the same value (bottom image).

3.4 Dataset and Training Details

The dataset used to train the proposed model was the VR-EyeTracking dataset [46], whose videos present a resolution of at least 4K (3840 pixels in width), a frame ratio of 25 frames per second (fps), duration ranging 20s to 60s, and varied content (e.g., indoor scenes, outdoor activities, or sports games). The fixations of 45 human observers, recorded with an eye-tracking device mounted on the HMD, are also provided together with the 360° videos. From the entire dataset of 76 360° videos, 32 that present simple scenes and a static camera are selected as training data.

To train the model, the selected 32 videos were down-sampled from 25 fps to 8 fps, and reshaped to a 320x240 resolution, which reduces memory, computation, and processing requirements. Each video is then divided into sequences of 20 frames, each one corresponding to 2.5s. The initial 20 frames are discarded to avoid a center bias present due to the capturing methodology (i.e., all participants were asked to look at the same point at the beginning of each video).

A second dataset proposed by Zhang et al. [47] will be used for an additional evaluation of the model and comparisons to state-of-the-art works. The dataset consists of 104 videos of the 342 from the Sports-360 dataset [48] whose duration was at least 20s. The videos show different sports activities such as dance, BMX, or skateboarding, and present challenging scenes with multiple elements and salient areas. They also include artistic cuts and artistic transitions (e.g., fade out, cross dissolve). Even though most of the videos were recorded with a static camera, some of them were captured while the camera was in motion. The recorded eye fixations of 20 observers are provided along with the videos as ground truth data. The Sports-360 dataset was processed as the VR-EyeTracking dataset.

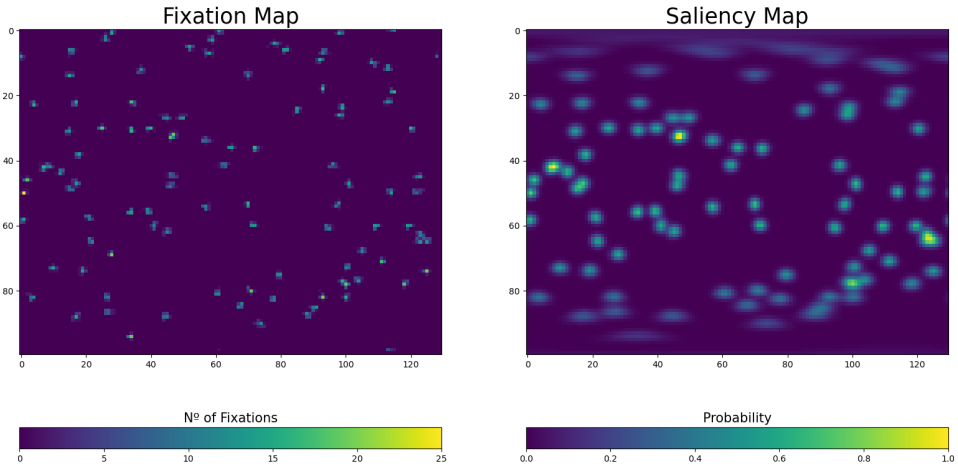


Figure 3.4: Example of a saliency map (right image) obtained from its fixation map (left image), after applying the Gaussian kernel, whose horizontal radius has been scaled to account for the distortion introduced by the equirectangular projection.

The ground truth saliency maps used for training and evaluation were obtained from the eye fixations provided with the datasets using the method adopted by Sitzman et al. [49]. Each value of the saliency map represents the probability that an observer will fix its gaze on that

pixel. Therefore, each frame's saliency map is built by counting the number of fixations that observers have made at each pixel location in the equirectangular image (see the left image of Figure 3.4). Then, these maps are convolved with a Gaussian with a standard deviation of 2° of visual angle to yield continuous saliency maps (see the right image of Figure 3.4). To account for the distortion introduced by the equirectangular projection, the Gaussian kernel's horizontal radius is scaled, increasing as it approaches the poles.

The training process took 10.93 hours on a Quadro P5000 with the following hyperparameters: an Adam optimizer [50], a learning rate of 0.001, a batch size of 3 sequences, and 175 epochs. Please refer to Appendix A.1 for more information about deep neural networks and their training process. A mean inference time of 125.795 milliseconds with a standard deviation of 1.266 milliseconds was obtained after performing 3000 model inferences with sequences of 20 frames.

4. Results and Evaluation

This section provides an in-depth analysis of the performance of the proposed model for dynamic 360° saliency prediction. Section 4.2 show some results obtained with the proposed model, and Section 4.3 presents a qualitative and quantitative comparison with state-of-the-art works. Furthermore, Section 4.4 offers a detailed ablation study that validates the effectiveness of the different elements included in the network architecture.

4.1 Saliency Metrics

To assess the performance of the developed model, and provide a meaningful comparison with state-of-the-art works, three different metrics commonly used in the literature for saliency maps comparison were chosen. They are computed following the implementation proposed by Gutiérrez et al. [51] in which each pixel i of a saliency map is weighted by the sine of its latitude, thus accounting for the distortion introduced by the equirectangular projection. The metrics used to measure the difference between a predicted saliency map P and its ground truth G obtained from real observers' fixations (see Section 3.4), are the following:

- Linear Correlation Coefficient (CC): It is a statistical method used to measure the correlation or dependence between two variables. In saliency prediction, CC measures the linear relationship between two saliency maps, where CC=0 means poor correlation, CC=1 perfect correlation, and CC=-1 perfect correlation but the distributions are opposite. Therefore, areas with similar magnitude values in both predicted and ground truth saliency maps will present high positive CC values. This metric can be obtained from the covariances σ of the saliency maps as follows:

$$CC\left(P,G\right)=\frac{\sigma\left(P,G\right)}{\sigma(P)\times\sigma\left(G\right)} \tag{4.1}$$

- Similarity Metric (SIM): It measures the similarity between two saliency maps represented as probability distributions. The SIM value is 1 when both saliency maps are the same, and 0 if they do not overlap. The SIM score is computed as follows:

$$\begin{gathered}\mathrm{SIM}\left(P,G\right)=\textstyle\sum_i\min\left(P_i,G_i\right)\\ \text{where }\textstyle\sum_i P_i=\sum_i G_i=1\end{gathered} \tag{4.2}$$

This metric is very sensitive to missing values, penalizing predictions that do not present the same density as the ground truth. Because of this, the standard deviation of the Gaussian kernel applied when obtaining the ground truth saliency maps (see Section 3.4) affects the SIM value, only reaching its maximum value when the standard deviation of the predicted and ground truth saliency maps is the same [52].

- Kullback-Leibler Divergence (KLDiv): It measures the difference between two saliency maps considered as probability distributions. KLDiv ranges from 0, where both saliency maps are the same, to infinity. The KLDiv value is obtained as follows:

$$KLDiv\left(P,G\right)=\sum_{i}G_{i}\log\left(\epsilon+\frac{G_{i}}{\epsilon+P_{i}}\right) \tag{4.3}$$

The SIM, KLDiv, and CC values presented as results in the comparisons represent the average mean and the average standard deviation of the measures obtained for all the videos. The mean values and their standard deviation for each video are computed by evaluating each video's frame with respect to its ground truth counterpart with the three metrics presented. Then, the values obtained for all the videos are averaged to obtain the final performance of a model.

There are a greater number of metrics for saliency map comparison than the three presented, and each of them considers different factors. The properties of each saliency map affect metric scores differently: how the ground truth is represented, whether the prediction includes dataset bias, whether the inputs are probabilistic or whether spatial deviations exist between the prediction and ground truth [52]. Therefore, several metrics and qualitative analyses must be used to draw conclusions about the similarity between two saliency maps, since there is no single metric that can address all the saliency maps' properties and provide a reliable similarity measure.

4.2 Results

A K-Fold Cross-Validation strategy has been used for the evaluation of the proposed model over the VR-EyeTracking dataset, due to the limited number of selected videos (see Section 3.4). K-Fold cross-validation is a procedure used for the evaluation of machine learning models when data availability is limited. This method divides the entire dataset into k groups, or folds, of approximately equal size. Then, k models are trained under identical conditions and hyperparameters, but each time a different fold is used as test data and the remaining k-1 as training data. The evaluation results of each k-model on the corresponding test fold are averaged to determine the overall network's performance. This evaluation method offers a better analysis of the network's accuracy than using a single test fold or split with a reduced number of samples, which are susceptible to being biased, leading to an overestimation, or underestimation of the network's performance.

The VR-EyeTracking dataset was divided into 5 different folds, with each fold representing approximately 20% of the dataset. Therefore, the proposed model was trained five times (i.e., five runs) for the evaluation with the VR-EyeTracking dataset, each of them with a different

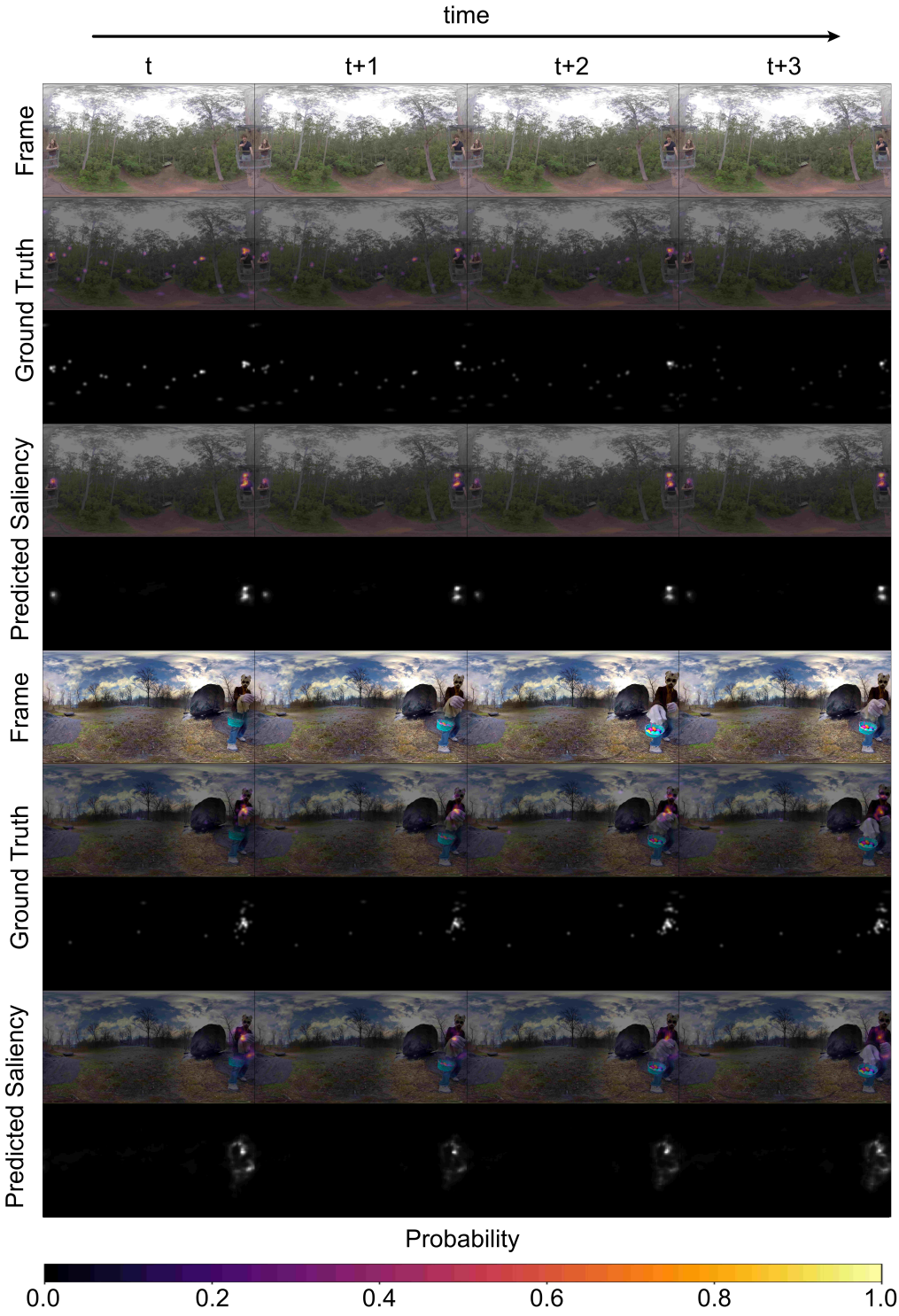


Figure 4.1: Results obtained with the proposed model for two sequences of the VR-EyeTracking dataset[1]. The horizontal axis represents time. The vertical axis shows the frames, the ground truth saliency, and the predicted saliency for two sequences. Saliency is represented as a heat map blended with the frame's image, where warm colors correspond to more salient areas. Note that the proposed model performs accurate predictions similar to the ground truth, both focusing in small, yet relevant regions of the scene.

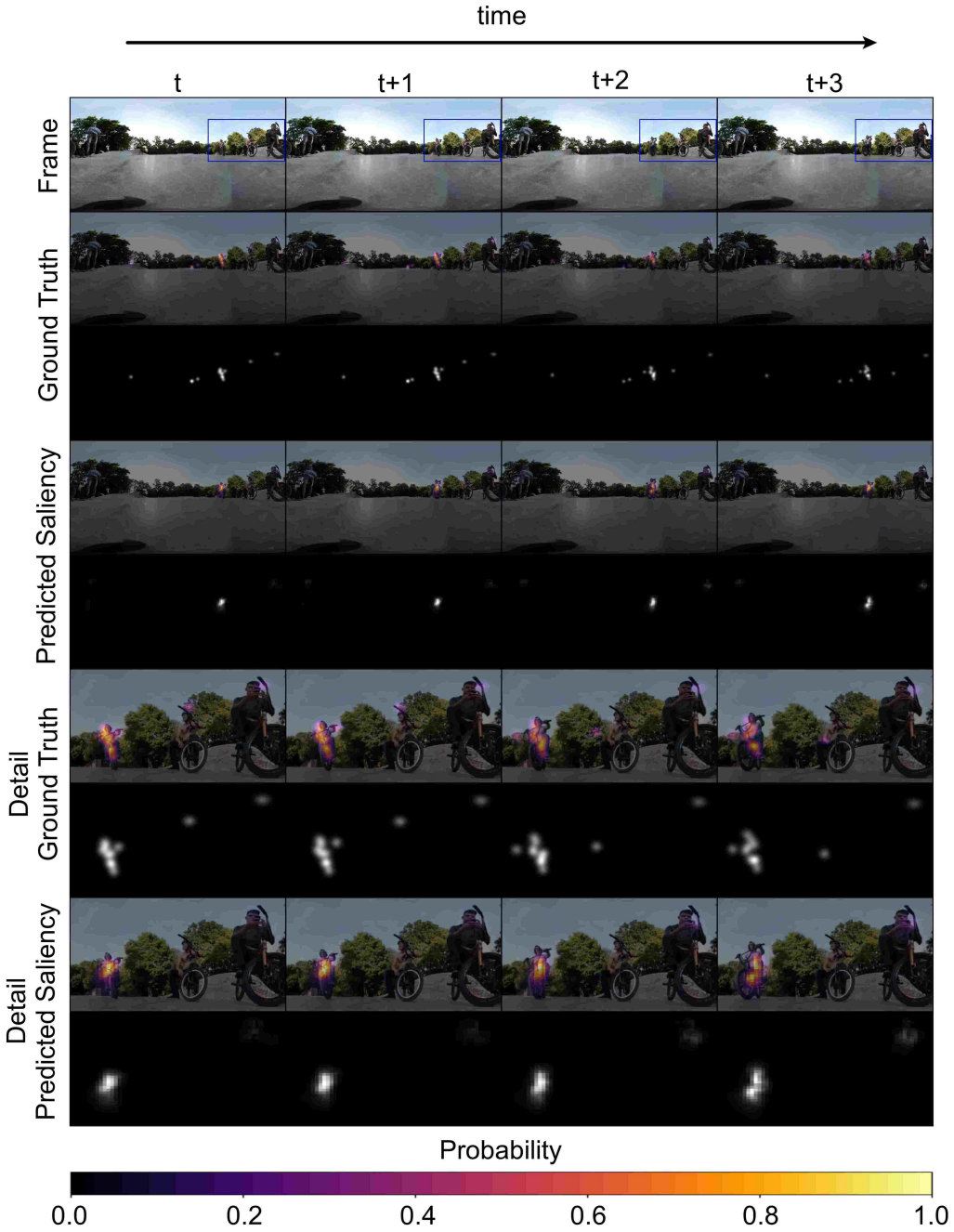


Figure 4.2: Results obtained with the proposed model for a sequence of the Sports-360 dataset[1]. The horizontal axis represents time. The vertical axis shows the frames, the ground truth saliency, and the predicted saliency for the sequence and the zoomed area within the blue rectangle. Saliency is represented as a heat map blended with the frame's image, where warm colors correspond to more salient areas. Note the model's ability to identify dynamic features in the sequence, where, although there are multiple salient elements (e.g., people on bicycles), it only focuses on the truly relevant one, since it has prior information about what happened earlier in the 360° video.

combination of folds. For the evaluation with the Sports-360 dataset, a single training was performed with all the 32 videos of the selected VR-EyeTracking dataset.

Figures 4.1 and 4.2 show some sample results obtained with the proposed model in the VR-EyeTracking and Sports-360 datasets.[1] For a set of videos, some consecutive RGB frames are shown, together with their ground truth saliency maps and the model's prediction, blended with the frame. It can be seen how the proposed model is able to perform accurate saliency predictions, which are close to the ground truth, both focusing on small, yet relevant regions of the scene. The ability of the model to handle dynamic features can also be appreciated in the Figure 4.2, where although there are multiple salient elements (i.e., people on bicycles) the network is able to detect and focus on which one is truly salient, thanks to the temporal information it has retained about what happened before in the 360° video.

Additionally, Table 4.1 provides a quantitative evaluation of the proposed model with the CC, SIM and KLDiv metrics obtained for the VR-EyeTracking and Sports datasets, showing promising results with respect to state-of-the-art works.

4.3 Comparison with Previous Methods

The proposed model was compared to three state-of-the-art works whose implementation is publicly available: one spherical deep neural network designed for saliency prediction in 360° images (i.e., Martin et al's [3]) to serve as a baseline, and to show that static approaches cannot perform well in dynamic scenarios, and two state-of-the-art models specialized on 360° videos, ATSal [1] and CP-360 [2]. More details on them can be found in Sections 2.2.2 and 2.3. To assess the models' performance, the metrics CC, SIM, and KLDiv (Section 4.1) were computed over the saliency predictions of the models for the VR-EyeTracking and Sports-360 datasets.

Table 4.1 shows that the proposed model outperforms previous approaches in all computed metrics. Martin et al.'s model was trained on a dataset of static 360° images, which hinders its ability to generalize to dynamic content, focusing on more low-level features such as contrast or edges. Both ATSal and CP-360 models were trained on dynamic content, and have a small LSTM-based module on their bottleneck to account for temporal relations. However, they neglect the fact that temporal information may be of importance even when extracting and encoding features, and are therefore limited. The proposed model, nevertheless, is able to handle temporal relations in all stages, and yields more precise results. Note that, although performance drops for all the models when evaluated on the Sports-360 datasets given its complex nature (i.e., moving cameras and several regions of interest), the proposed model is still able to outperform previous approaches, suggesting it is robust and able to generalize to different types of stimuli.

Additionally, Figure 4.3 shows a qualitative comparison between the models on a small subset of sequences from 360° videos from the VR-EyeTracking and Sports-360 datasets[1]. The proposed model produces plausible saliency predictions, identifying the regions where humans tend to direct their attention.

[1] Video results are available at:
https://drive.google.com/drive/folders/1d4I4hvDiZwLGXDbFwc5uFeeenoziKZcx?usp=sharing.

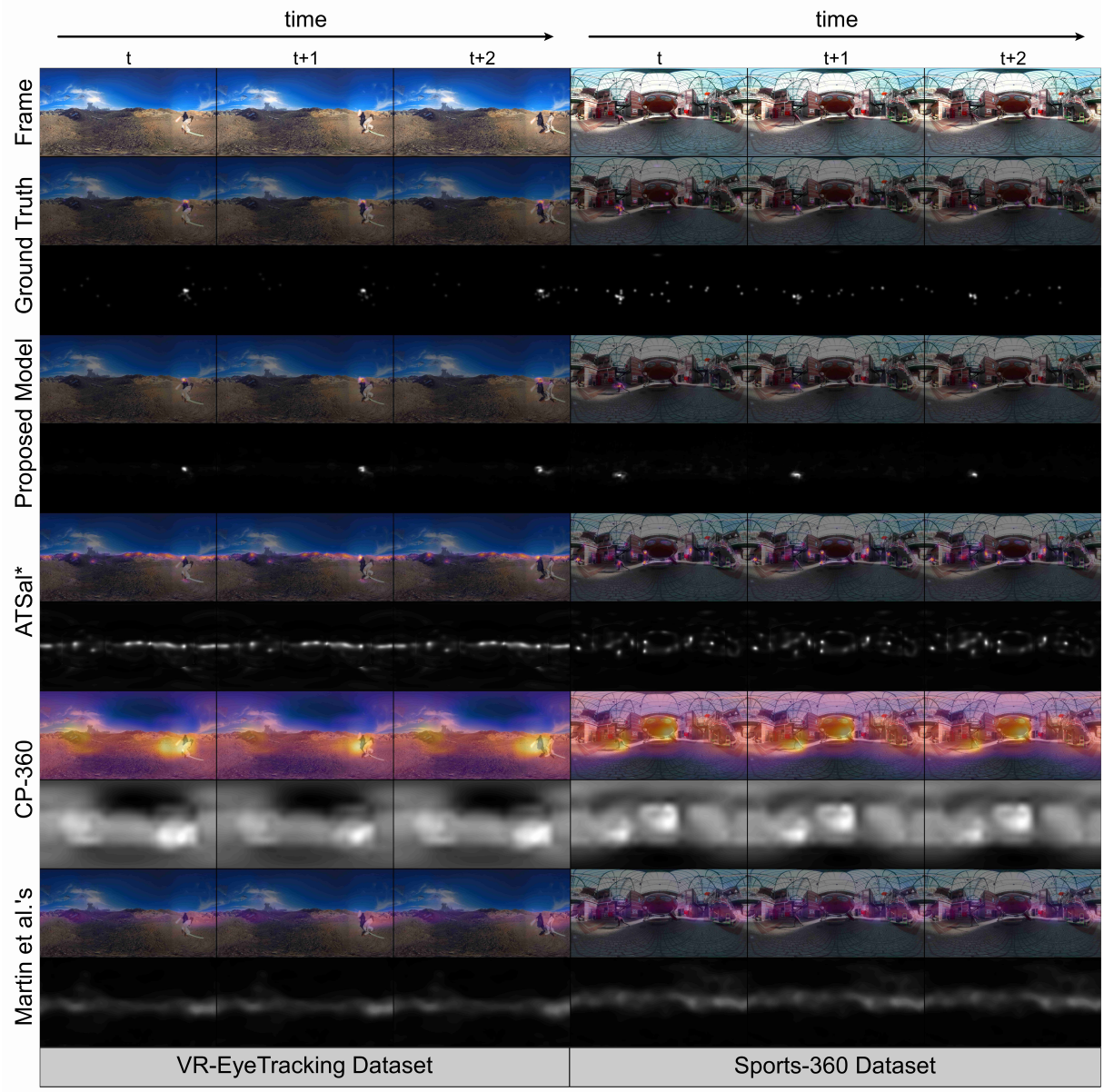


Figure 4.3: Qualitative comparison of the proposed model against ATSal [1], CP-360 [2], and Martin et al.'s [3][1]. The horizontal axis represents time. The vertical axis shows, from top to bottom: The RGB frame, the ground truth saliency, the proposed model's prediction, ATSal, CP-360, and Martin et al.'s saliency predictions. Saliency is shown both with a heat map blended with the frame's image (warm colors correspond to high salient areas) and with the saliency maps (black indicates zero probability of being observed). Note that the proposed model outperforms the state-of-the-art works since its saliency prediction is closer to the ground truth. Martin et al.'s, as a saliency prediction method for 360° images, tends to predict exploration saliency maps typical of this type of content. Both ATSal and CP-360 models were trained on dynamic content, but either fail to identify the truly salient elements of the 360° video or predict a high probability for most of the frame's regions, which does not resemble the ground truth behavior. ∗ATSal was trained with the VR-EyeTracking dataset; its results are included for completeness.

	VR-EyeTraking dataset			Sports-360 dataset		
	CC ↑	SIM ↑	KLDiv ↓	CC ↑	SIM ↑	KLDiv ↓
Proposed Model	**0.4075 (0.1539)**	**0.2186 (0.0653)**	**10.070 (1.5424)**	**0.2598 (0.1113)**	**0.1221 (0.0411)**	**12.501 (1.0902)**
ATSal*	0.2435 (0.0883)	0.1069 (0.0266)	12.398 (0.6667)	0.1792 (0.0743)	0.0785 (0.0226)	13.037 (0.6231)
CP-360	0.2577 (0.0502)	0.1516 (0.0310)	13.754 (0.7039)	0.2106 (0.0492)	0.1038 (0.0260)	14.773 (0.6089)
Martin et al.'s	0.1520 (0.0439)	0.0834 (0.0194)	13.607 (0.5245)	0.1444 (0.0460)	0.0657 (0.0175)	14.005 (0.4921)

Table 4.1: Quantitative comparisons of the proposed model against ATSal[1], CP-360[2], and Martin et al.'s [3], with both VR-EyeTracking and Sports-360 datasets. Arrows indicate whether higher or lower is better, and boldface highlights the best result for each metric. The values represent the mean score among the different videos in the dataset for each metric, and in brackets is shown the averaged standard deviation. The individual scores for each video can be found in the Appendix B.1. ∗ATSal was trained with the VR-EyeTracking dataset; its results are included for completeness.

The quantitative evaluation of Table 4.1 shows that 360° videos present a great variability in terms of saliency between frames, since the scenes can change completely from frame to frame, and between videos, due to their varied content (e.g., sports, films, indoor scenes, etc.). This causes noticeable differences among the CC and SIM values obtained for each frame of the same video, resulting in elevated standard deviations when using ATSal and the proposed model. In contrast, CP-360 and Martin et al.'s present significantly lower standard deviations. This low variability comes from the fact that CP-360 predicts similar high probability saliency maps for all videos and Martin et al.'s has a strong equator bias, failing to represent the natural variation of data. Despite this variability, the proposed model shows consistently better overall results, further confirmed by the qualitative assessment.

4.4 Ablation Study

Some ablation studies have been performed to endorse the effectiveness of the decisions taken through the design stage of the model's architecture. These studies analyze the influence of the input data resolution, the spherical convolutions, the loss function employed, the inclusion of optical flow and depth information, and the advantages reported by the Spherical ConvLSTMs.

The K-Fold Cross-Validation strategy, already mentioned in Section 4.2 has been used for the different tests. The results obtained for each experiment of the ablation study are shown in Table 4.2. The values represent the average of the five scores obtained with the metrics detailed in Section 4.1 for each fold.

4.4.1 Input Resolution

This first ablation study analyzes the influence of the resolution of the 360° videos on the ability of the network to predict an accurate saliency map. To this purpose, the proposed model detailed

in Section 3 was trained with two different setups: with the 360° videos of the dataset resize to a resolution of 208x106, and with a higher resolution of 320x240.

The results obtained for the three metrics in these experiments can be seen in the first two rows of Table 4.2. The mean score of the three metrics reports a slight improvement when using the higher resolution layout, which could indicate that the reduction of the frame's resolution, hindered the training process.

4.4.2 Spherical Convolutions

While the purpose and operation of spherical convolutions have already been explained in Section 3.1.1, this ablation study justifies in terms of the model's performance its inclusion on the ConvLSTM architecture. For its evaluation, the proposed model architecture is modified, replacing the spherical convolutions implemented in the Spherical ConvLSTMs of both encoder and decoder by traditional convolutions. Then, the non-spherical model is trained under the same conditions as the original spherical one. The results can be observed in the third row of Table 4.2.

Results show an improvement in the model's performance when spherical convolutions are used. Evidencing that the use of architectures that account for the peculiarities of the equirectangular representation is an advantageous approach when dealing with 360° content.

4.4.3 Alternative Loss Functions

The current spherical weighted KLDiv loss function used to train the model was compared against two alternative configurations. A traditional KLDiv (Equation 4.3), in which all pixels have equal weight in the metric, no matter where they are located, and a spherical sampled KLDiv, which presents a different strategy to account for the distortion introduced by the equirectangular projection. Instead of weighting each pixel by its solid angle, the equirectangular image's pixels are sampled following a uniform cosine sampling distribution. Therefore, the equator of the image will have more weight than the poles on the final value of the loss function, since a greater number of pixels belonging to this region will be used to compute the metric. The results obtained after training the model with the spherical weighted KLDiv, spherical sampled KLDiv, and traditional KLDiv metrics are shown in the second, fourth, and fifth rows of Table 4.2 respectively.

The results obtained with the different loss functions suggest that the best suited is the weighted KLDiv, as it achieves better mean accuracy for all the metrics. It was expected that the sampled KLDiv loss function, which accounts for the equirectangular projection, would report better performance than the traditional KLDiv loss. However, sampled KLDiv loss presents worse results for CC and SIM. This could be either because of the poor performance of randomness of the samples, or the fact that most of the relevant information of the videos are in the less distorted part, leading the traditional KLDiv loss to reach a higher performance.

4.4.4 Inclusion of Optical Flow Estimation

The motivation behind providing an optical flow estimation between consecutive frames to the model is to achieve more accurate saliency predictions, since moving elements tend to capture human visual attention. Therefore, the aim of this ablation study is to analyze the effect that providing the network with this information has on its performance. For this purpose, a variation of the model was trained in which it was deprived of the optic flow information. Therefore, the dimensions of the input data became 320x240x3, since only the RGB image of each frame is fed to the model. The results obtained with this configuration are shown in the sixth row of Table 4.2.

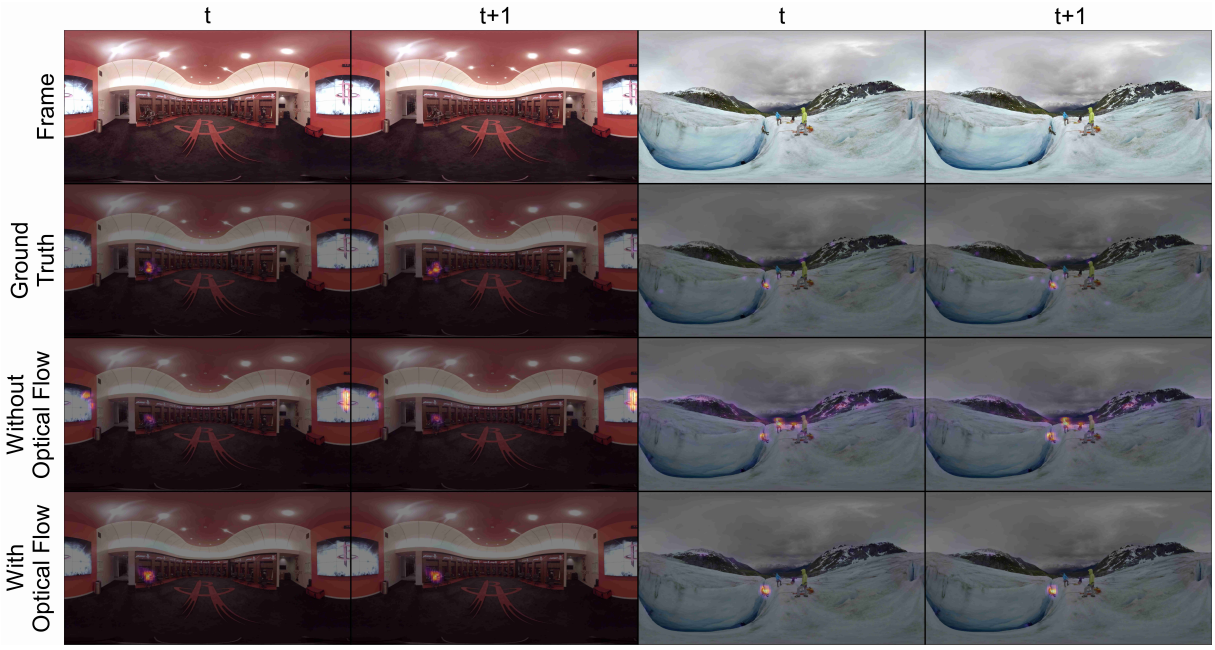


Figure 4.4: Comparison between saliency predictions obtained with and without optical flow estimations for two sequences of VR-EyeTracking dataset. The horizontal axis represents the time sequence. The vertical axis shows from top to bottom: the RGB frames, the ground truth saliency, the predicted saliency without optical flow and the predicted saliency with optical flow. Saliency is represented as a heat map blended with the frame's image, warm colors correspond to high salient areas. Note that when using an optical flow estimation the model performs more accurate predictions, reducing the saliency of high contrast areas, that although salient in static images, in the dynamic case they lose relevance in the face of motion.

The scores obtained with the three metrics show an improvement in performance when including the optical flow estimations. This improvement is even more noticeable when qualitatively analyzing the saliency predictions made by the model. Figure 4.4 shows two cases where the inclusion of this optical flow estimation is especially critical in identifying salient regions. Without optical flow estimation, the model tends to fixate on areas of high contrast, which while salient in static content, are unattractive in dynamic content due to the existence of motion and action.

208x106	320x240	Spherical ConvLSTM	Optical Flow	Depth	Weighted KLDiv	Sampled KLDiv	Traditional KLDiv	CC ↑	SIM ↑	KLDiv ↓
✓	-	✓	✓	-	✓	-	-	0.3975 (0.1560)	0.2118 (0.0656)	10.608 (1.6748)
-	✓	✓	✓	-	✓	-	-	**0.4075 (0.1539)**	**0.2186 (0.0653)**	**10.070 (1.5424)**
-	✓	-	✓	-	✓	-	-	0.3284 (0.1366)	0.1896 (0.0593)	10.739 (1.4757)
-	✓	✓	✓	-	-	✓	-	0.3822 (0.1373)	0.2088 (0.0576)	10.377 (1.3414)
-	✓	✓	✓	-	-	-	✓	0.3873 (0.1241)	0.2062 (0.0522)	10.498 (1.2233)
-	✓	✓	-	-	✓	-	-	0.3614 (0.1271)	0.1946 (0.0541)	10.781 (1.2942)
-	✓	✓	✓	✓	✓	-	-	0.3779 (0.1374)	0.2072 (0.0565)	10.456 (1.3540)
-	✓	-	✓	-	✓	-	-	0.2830 (0.1317)	0.1124 (0.0339)	12.339 (0.7856)

Table 4.2: Ablation Study: Comparison of the scores, mean and standard deviation, obtained with the different configurations. Check marks indicate which elements are used in the ablation study, and the shaded row is the model proposed in this master's thesis. The values represent the mean score among the different videos in the dataset for each metric, and in brackets is shown the averaged standard deviation. The individual scores for each video can be found in the Appendix B.2. Please refer to Section 4.4.1 to 4.4.6 for further details.

4.4.5 Influence of Spherical ConvLSTMs

The estimation of the optical flow provides the network with temporal information about the previous frame. The present study aims to determine if the temporal information provided by the motion flow estimation is sufficient, or if ConvLSTMS need to be included to achieve accurate predictions. To analyze the effect of using spherical ConvLSTMs without optical flow estimation versus using only the latter, a structure without spherical ConvLSTMs is trained. The model consists of an encoder and decoder formed only by spherical convolutional layers. The encoder accepts as input data the concatenation of the RGB frame and the RGB optical flow of size 320x240x6. The input's spatial size is reduced to 160x120x36 by means of a spherical convolution and a spherical Max Pool (SphereNet implementation), thus extracting the feature vector. The decoder returns the predicted saliency map of size 320x240x1 by processing the feature vector with another spherical convolution and an Up-sampling layer. The results obtained with this configuration can be seen in the eighth row of Table 4.2.

In both quantitative (Table 4.2) and qualitative (Figure 4.5) comparisons between using just optical flow and using just Spherical ConvLSMs, a drop in performance can be observed with the former one. It can be seen in Figure 4.5 that the saliency prediction provided by the model that only uses optical flow practically corresponds to this estimation plus some high contrast areas, which leads to an erroneous prediction since not all moving elements are salient. However, the poor accuracy reported may be due to the simplicity of the structure, which is built with only two spherical convolutional layers and deeper architectures are usually employed in DL networks for image processing. Nevertheless, this simple architecture provides a fair comparison with the proposed model, since with the same number of convolutions and equal feature vector's size it achieves significantly better saliency predictions.

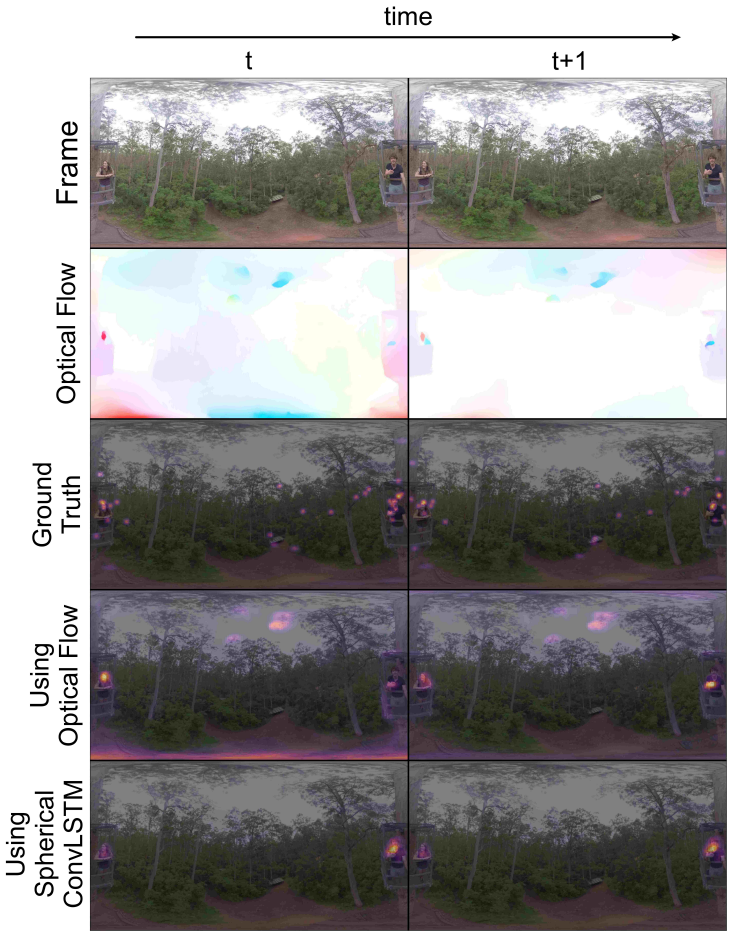


Figure 4.5: Comparison between saliency predictions using just Spherical ConvLSTMs and just optical flow for a sampled sequence of VR-EyeTracking dataset. The horizontal axis represents the time sequence. The vertical axis shows from top to bottom: the RGB frames, the optical flow estimation, the ground truth saliency, the predicted saliency with just optical flow, and the predicted saliency with just Spherical ConvLSTMs. Saliency is represented as a heat map blended with the frame's image, warm colors correspond to high salient areas. Note that when using just the optical flow estimation the model predicts high saliency to areas that are not salient, such as tree branches, which, although they have movement, do not attract humans' attention.

4.4.6 Inclusion of Depth Information

Not only does the movement present in a scene play an important role in its saliency, but depth also has a great influence on it [53, 54]. The closeness or remoteness of the elements on the scene can alter the way in which humans perceive them. Objects close to the observer are more likely to be looked at due to their proximity, but also deep areas such as corridors, horizons, or abysses tend to capture human attention on 360° content. Because of this, the possibility of providing the network with an estimation of the depth at each frame was studied. The proposed model was trained with input data of shape 320x240x7, composed of depth estimation, optical flow estimation, and the RGB image. Additionally, the size of the hidden state is increased from 36 to 49, since the inclusion of the depth estimation may require a larger feature vector to be able to collect the new input information. The results can be seen in the seventh row of Table 4.2.

Depth estimation was obtained with DPT [55], a deep neural network for saliency prediction in traditional 2D images. Although several works address the issue of depth prediction in 360° images [56, 57, 58], all of them focus on indoor scenes, due to a lack of labeled datasets of outdoor scenes. The choice of DPT over other depth estimation networks for traditional 2D images was mainly based on the depth estimations provided, which greatly surpassed those obtained with other methods such as Monodepth2 [59] or MiDaS [60]. The depth estimation is performed for each frame, obtaining a numerical depth value for each pixel, thus it is represented as a single channel matrix of the same size as the frame.

However, depth estimation seems to hinder the model's performance instead of boosting it. This may be due to the lack of precision of the depth estimations obtained with DPT, which could be confusing the model rather than enhancing its learning. It has been observed that the depth estimations present a low resolution (i.e., all objects beyond a distance threshold are considered to have the same depth, even if some are clearly much further away than others). This is especially relevant in outdoor scenes, which constitute a large part of the datasets, and where the difference between the depths of the scene elements is high. Therefore, this lack of resolution could result in salient areas due to their great depth being overlooked in preference to nearby regions.

To conclude, the present ablation study has shown that the use of optical flow estimation is a better approach, which combined with spherical convolutions, an encoder-decoder structure based on ConvLSTMs, and a spherical weighted loss function, allows to produce accurate saliency predictions that resemble human visual behavior.

5. Conclusions

This master's thesis has presented a DL-based approach for modeling human visual behavior in dynamic 360° environments through saliency prediction. The proposed saliency model has been tailored to dynamic 360° scenes, accounting for the spatio-temporal information and the particularities of the 360° equirectangular representation by its encoder-decoder architecture based on Spherical ConvLSTMs. The proposed model outperforms state-of-the-art works, obtaining accurate saliency predictions in a wide variety of 360° videos.

The work carried out in this master's thesis reveals that dynamic content requires models that account for temporal information. Saliency prediction models for static images are surpassed by architectures specifically designed for videos, whose awareness of the temporal relationships between frames seems to play a major role in the accuracy of the dynamic saliency prediction. Additionally, the availability of this temporal information at the time of feature extraction and its posterior decoding has proved to allow inferring the most relevant features, leading to more accurate predictions. The proposed model has been the first to account for temporal information through an encoder-decoder based on spherical ConvLSTMs, demonstrating to outperform previous state-of-the-art approaches based on traditional CNNs encoder-decoders, which only extract spatial features.

Based on the results obtained and the ablation study carried out, optical flow can be considered as a highly salient feature that should be contemplated in dynamic saliency prediction. The proposed model has benefited from the inclusion of the optical flow estimations between frames, contributing positively to the saliency prediction's accuracy.

It also has been shown that to work with 360° images, architectures must be adapted to their particularities. The use of spherical convolutions and the novel spherical weighted KLDiv loss function, which take into account the distortion present in the equirectangular representation, aid in the interpretation of 360° images, enhancing the quality of the modeled saliency.

5.1 Limitations and Future Work

The work carried out in this master's thesis aims to be a first step towards understanding and modeling human visual behavior in dynamic 360° environments, aiming to serve as a basis for future works in this field that is still an open problem. There is room for improvement in dynamic 360° saliency prediction and further work is expected.

The proposed model for saliency prediction in 360° videos has been trained with a selection of

videos from the VR-EyeTraking dataset without camera movement. Thus, saliency prediction on 360° videos presenting more challenging scenes remains interesting for future works. The proposed approach has focused on videos with a static camera due to the use of optical flow estimations. These videos present a clearer correlation between high optical flow and salient objects than videos with camera movement, where for example the background of the scenes usually shows high optical flow without being salient at all. Therefore, as camera movement was not contemplated during training, it is to be expected that the model does not adapt to these situations. To palliate this, the training dataset could be extended in future works with videos presenting camera movement, analyzing whether the network is able to establish the relationships between optical flow and saliency in both static and dynamic camera videos.

A limitation of the current model related to the LSTMs architecture is that, although these networks present longer memory than conventional RNNs, it is still limited. Recently, more powerful architectures are emerging to cope with this limitation (e.g., transformers [61]), thus an interesting future work could be to explore the implementation of these alternative architectures for the particular case of saliency prediction in dynamic 360° content.

Saliency models for 360° videos are also being held back by the lack of datasets with a large number of videos and ground truth data. Existing datasets present recorded gaze data of a small group of observers, resulting in sparse ground-truth saliency maps in which the common behavior is sometimes unclear, usually in complex scenes with multiple salient elements. Therefore, gaze data from a larger number of observers are needed to have a representative sample of human visual behavior, thus allowing saliency models to infer it.

Additionally, 360° dynamic environments are not only visual experiences, but they usually include directional acoustic stimuli surrounding the observers that affect how they perceive and explore the scenes. These acoustic cues greatly affect visual perception [62]. Therefore, the study of the inclusion of audio together with the 360° video is an interesting line for future work, but this is again limited by the lack of a dataset containing auditory information about the 360° videos.

Bibliography

[1] Yasser Dahou, Marouane Tliba, Kevin McGuinness, and Noel O'Connor. ATSal: An Attention Based Architecture for Saliency Prediction in 360 Videos. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12663 LNCS:305–320, nov 2020.

[2] Hsien-Tzu Cheng, Chun-Hung Chao, Jin-Dong Dong, Hao-Kai Wen, Tyng-Luh Liu, and Min Sun. Cube padding for weakly-supervised saliency prediction in 360° videos. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1420–1429, 2018.

[3] Daniel Martin, Ana Serrano, and Belen Masia. Panoramic convolutions for 360° single-image saliency prediction. In *CVPR Workshop on Computer Vision for Augmented and Virtual Reality*, 2020.

[4] Nitish S. Mutha. How to map equirectangular projection to rectilinear projection. `http://blog.nitishmutha.com/equirectangular/360degree/2017/06/12/How-to-project-Equirectangular-image-to-rectilinear-view.html`.

[5] Benjamin Coors, Alexandru Paul Condurache, and Andreas Geiger. Spherenet: Learning spherical representations for detection and classification in omnidirectional images. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, pages 525–541, Cham, 2018. Springer International Publishing.

[6] Omar Costilla Reyes, Ruben Vera-Rodriguez, Abdullah Alharthi, Syed Yunas, and Krikor Ozanyan. *Deep Learning in Gait Analysis for Security and Healthcare*, pages 299–334. 10 2019.

[7] Daniel Martin, Ana Serrano, Alexander W. Bergman, Gordon Wetzstein, and Belen Masia. Scangan360: A generative model of realistic scanpaths for 360° images. *arXiv preprint arXiv:2103.13922*, 2021.

[8] Daniel Martin, Sandra Malpica, Diego Gutierrez, Belen Masia, and Ana Serrano. Multimodality in vr: A survey. *ACM Comput. Surv.*, dec 2021. Just Accepted.

[9] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1254–1259, 1998.

[10] Laurent Itti and Christof Koch. A saliency-based search mechanism for overt and covert shifts of visual attention. *Vision Research*, 40:1489–1506, 2000.

[11] Erkut Erdem and Aykut Erdem. Visual saliency estimation by nonlinearly integrating features using region covariances. *Journal of vision*, 13 4:11, 2013.

[12] Neil Bruce and John Tsotsos. Saliency based on information maximization. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems*, volume 18. MIT Press, 2006.

[13] Jonathan Harel, Christof Koch, and Pietro Perona. Graph-based visual saliency. In *NIPS*, 2006.

[14] Ali Borji. Boosting bottom-up and top-down visual features for saliency estimation. *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 438–445, 2012.

[15] Tilke Judd, Krista A. Ehinger, Frédo Durand, and Antonio Torralba. Learning to predict where humans look. *2009 IEEE 12th International Conference on Computer Vision*, pages 2106–2113, 2009.

[16] Se-Ho Lee, Jin-Hwan Kim, Kwang Pyo Choi, Jae-Young Sim, and Chang-Su Kim. Video saliency detection based on spatiotemporal feature learning. *2014 IEEE International Conference on Image Processing (ICIP)*, pages 1120–1124, 2014.

[17] Srinivas S. S. Kruthiventi, Kumar Ayush, and R. Venkatesh Babu. Deepfix: A fully convolutional neural network for predicting human eye fixations. *IEEE Transactions on Image Processing*, 26(9):4446–4456, 2017.

[18] Nian Liu, Junwei Han, Dingwen Zhang, Shifeng Wen, and Tianming Liu. Predicting eye fixations using convolutional neural networks. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 362–370, 2015.

[19] Matthias Kümmerer, Lucas Theis, and Matthias Bethge. Deep gaze i: Boosting saliency prediction with feature maps trained on imagenet. 2015.

[20] Junting Pan, Elisa Sayrol, Xavier Giro-I-Nieto, Kevin McGuinness, and Noel E. O'Connor. Shallow and deep convolutional networks for saliency prediction. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 598–606, 2016.

[21] Junting Pan, Cristian Canton, Kevin McGuinness, Noel E. O'Connor, Jordi Torres, Elisa Sayrol, and Xavier and Giro-i Nieto. Salgan: Visual saliency prediction with generative adversarial networks. In *arXiv*, January 2017.

[22] Marcella Cornia, Lorenzo Baraldi, Giuseppe Serra, and Rita Cucchiara. Predicting human eye fixations via an lstm-based saliency attentive model. *IEEE Transactions on Image Processing*, 27(10):5142–5154, 2018.

[23] Nian Liu and Junwei Han. A deep spatial contextual long-term recurrent convolutional network for saliency detection. *IEEE Transactions on Image Processing*, 27(7):3264–3274, 2018.

[24] Çagdas Bak, Aykut Erdem, and Erkut Erdem. Two-stream convolutional networks for dynamic saliency prediction. *ArXiv*, abs/1607.04730, 2016.

[25] Cagdas Bak, Aysun Kocak, Erkut Erdem, and Aykut Erdem. Spatio-temporal saliency networks for dynamic saliency prediction. *IEEE Transactions on Multimedia*, 20(7):1688–1698, 2018.

[26] Lai Jiang, Mai Xu, Tie Liu, Minglang Qiao, and Zulin Wang. Deepvs: A deep learning based video saliency prediction approach. In *ECCV*, 2018.

[27] Yucheng Zhu, Guangtao Zhai, and Xiongkuo Min. The prediction of head and eye movement for 360 degree images. *Signal Process. Image Commun.*, 69:15–25, 2018.

[28] A feature-based approach for saliency estimation of omni-directional images. *Signal Processing: Image Communication*, 69:53–59, 2018. Salient360: Visual attention modeling for 360° Images.

[29] Yuming Fang, Xiaoqiang Zhang, and Nevrez Imamoglu. A novel superpixel-based saliency detection model for 360-degree images. *Signal Processing: Image Communication*, 69:1–7, 2018. Salient360: Visual attention modeling for 360° Images.

[30] Pierre Lebreton and Alexander Raake. Gbvs360, bms360, prosal: Extending existing saliency prediction models from 2d to omnidirectional images. *Signal Processing: Image Communication*, 69:69–78, 2018. Salient360: Visual attention modeling for 360° Images.

[31] Guilherme Luz, João Ascenso, Catarina Brites, and Fernando Pereira. Saliency-driven omnidirectional imaging adaptive coding: Modeling and assessment. In *2017 IEEE 19th International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–6, 2017.

[32] Mikhail Startsev and Michael Dorr. 360-aware saliency estimation with conventional image saliency predictors. *Signal Processing: Image Communication*, 69:43–52, 2018. Salient360: Visual attention modeling for 360° Images.

[33] Marc Assens, Xavier Giro-i Nieto, Kevin McGuinness, and Noel E. O'Connor. Saltinet: Scan-path prediction on 360 degree images using saliency volumes. In *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 2331–2338, 2017.

[34] Rafael Monroy, Sebastian Lutz, Tejo Chalasani, and Aljoscha Smolic. Salnet360: Saliency maps for omni-directional images with cnn. *ArXiv*, abs/1709.06505, 2018.

[35] Fang-Yi Chao, Lu Zhang, Wassim Hamidouche, and Olivier Déforges. Salgan360: Visual saliency prediction on 360 degree images with generative adversarial networks. *2018 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, pages 01–04, 2018.

[36] Haoran Lv, Qin Yang, Chenglin Li, Wenrui Dai, Junni Zou, and Hongkai Xiong. SalGCN: Saliency Prediction for 360-Degree Images Based on Spherical Graph Convolutional Networks. *MM 2020 - Proceedings of the 28th ACM International Conference on Multimedia*, pages 682–690, oct 2020.

[37] Yanyu Xu, Ziheng Zhang, and Shenghua Gao. Spherical DNNs and Their Applications in 360º Images and Videos. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[38] Anh Nguyen, Zhisheng Yan, and Klara Nahrstedt. Your attention is unique: Detecting 360-degree video saliency in head-mounted display for head movement prediction. *MM 2018 - Proceedings of the 2018 ACM Multimedia Conference*, pages 1190–1198, oct 2018.

[39] Panagiotis Linardos, Eva Mohedano, Juan José Nieto, Noel E. O'Connor, Xavier Giro i Nieto, and Kevin McGuinness. Simple vs complex temporal recurrences for video saliency prediction. In *BMVC*, 2019.

[40] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.

[41] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 802–810, Cambridge, MA, USA, 2015. MIT Press.

[42] Jay Pratt, Petre V. Radulescu, Ruo Mu Guo, and Richard A. Abrams. It's alive!: Animate motion captures visual attention. *Psychological Science*, 21(11):1724–1730, 2010. PMID: 20974713.

[43] Hauke Meyerhoff, Stephan Schwan, and Markus Huff. Perceptual animacy: Visual search for chasing objects among distractors. *Journal of experimental psychology. Human perception and performance*, 40, 12 2013.

[44] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. *European Conf. on Computer Vision (ECCV)*, 2020.

[45] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In A. Fitzgibbon et al. (Eds.), editor, *European Conf. on Computer Vision (ECCV)*, Part IV, LNCS 7577, pages 611–625. Springer-Verlag, October 2012.

[46] Yanyu Xu, Yanbing Dong, Junru Wu, Zhengzhong Sun, Zhiru Shi, Jingyi Yu, and Shenghua Gao. Gaze prediction in dynamic 360° immersive videos. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5333–5342, 2018.

[47] Ziheng Zhang, Yanyu Xu, Jingyi Yu, and Shenghua Gao. Saliency detection in 360° videos. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[48] Hou-Ning Hu, Yen-Chen Lin, Ming-Yu Liu, Hsien-Tzu Cheng, Yung-Ju Chang, and Min Sun. Deep 360 pilot: Learning a deep agent for piloting through 360º sports video. *30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-January:1396–1405, may 2017.

[49] Vincent Sitzmann, Ana Serrano, Amy Pavel, Maneesh Agrawala, Diego Gutierrez, Belen Masia, and Gordon Wetzstein. Saliency in VR: How Do People Explore Virtual Environments? *IEEE Transactions on Visualization and Computer Graphics*, 24(4):1633–1642, apr 2018.

[50] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[51] Jesús Gutiérrez, Erwan David, Yashas Rai, and Patrick Le Callet. Toolbox and dataset for the development of saliency and scanpath models for omnidirectional/360° still images. *Signal Processing: Image Communication*, 69:35–42, nov 2018.

[52] Zoya Bylinskii, Tilke Judd, Aude Oliva, Antonio Torralba, and Frédo Durand. What do different evaluation metrics tell us about saliency models? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(3):740–757, 2019.

[53] Miroslav Laco and Wanda Benesova. Depth in the visual attention modelling from the egocentric perspective of view. In *International Conference on Machine Vision*, 2019.

[54] Karthik Desingh, K. Madhava Krishna, Deepu Rajan, and C. V. Jawahar. Depth really matters: Improving visual salient region detection with depth. In *BMVC*, 2013.

[55] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. *ArXiv preprint*, 2021.

[56] Il Dong Yun, Hyuk-Jae Lee, and Chae-Eun Rhee. Improving 360 monocular depth estimation via non-local dense prediction transformer and joint supervised and self-supervised learning. *ArXiv*, abs/2109.10563, 2021.

[57] Giovanni Pintore, Marco Agus, Eva Almansa, Jens Schneider, and Enrico Gobbetti. Slicenet: deep dense depth estimation from a single indoor panorama using a slice-based representation. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11531–11540, 2021.

[58] Nikolaos Zioulis, Antonis Karakottas, Dimitrios Zarpalas, and Petros Daras. Omnidepth: Dense depth estimation for indoors spherical panoramas. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[59] Clement Godard, Oisin Mac Aodha, Michael Firman, and Gabriel Brostow. Digging into self-supervised monocular depth estimation. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3827–3837, 2019.

[60] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2020.

[61] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. NIPS'17, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc.

[62] Belen Masia, Javier Camon, Diego Gutierrez, and Ana Serrano. Influence of directional sound cues on users exploration across 360° movie cuts. *IEEE Computer Graphics and Applications*, 2021.

[63] Christopher Olah. Understanding lstm networks. `https://colah.github.io/posts/2015-08-Understanding-LSTMs`.

Appendix A. Deep Learning Background

A.1 Deep Neural Networks

Deep learning is a subset of machine learning algorithms, called Deep Neural Networks (DNNs) that attempt to match the ability of humans' brains to learn. Nowadays, DNNs are being used in multiple applications in everyday products and services such as digital assistants, self-driving cars, medical diagnosis, stock market, electricity demand prediction, or marketing.

DNNs are formed by several layers of artificial neurons: elementary processing units that interconnect with each other. Like those in the human brain, artificial neurons process information, recognize patterns and learn from data or examples provided. The interaction between neurons is defined by adjustable parameters (weights and biases) and by fixed operations (e.g., nonlinear functions, concatenations, multiplications, etc.). In the learning stage of the DNN, these parameters, are automatically adjusted to produce the expected response from the DNN.

Deep learning algorithms can be classified as supervised or unsupervised depending on how these parameters are learned. Supervised learning is the most common form of machine learning in which label data is used to learn to perform classification or prediction tasks. Labeled data refers to data points whose information to be extracted by the DNN is known. However, it is not intended that the DNNs 'memorize' the label of each data point, but that learn from them to generalize, determining which relationships, patterns, or characteristics must be identified in the input data. On the other hand, unsupervised learning is performed without labeled data by recognizing differences between data points and grouping them according to them (a technique known as clustering).

A.1.1 Development of a Deep Neural Network by Supervised Learning

The development of a DNN to perform a specific task with a supervised learning technique starts by collecting the labeled data. Then, a network architecture is designed, implemented, and trained with the labeled data. Finally, the trained DNN is tested to assess its performance when dealing with unseen data.

Data acquisition

A sufficiently large number of relevant examples must be collected to allow the network to correctly learn from them. These examples or data points need to be labeled as well, since the label or ground truth information is what the DNNs must learn from de data point. The totality of the labeled data points is called the dataset. The dataset is usually preprocessed to facilitate the network's learning. This processing can consist of a reduction of the data variables, the elimination of outliers, or the normalization of the data.

The dataset should be as large and varied as possible, otherwise, it could result in an insufficient adjustment of the model to the task to be performed (i.e., underfitting) or a 'memorization' of the dataset (i.e., overfitting). The ability of a DNN to generalize is strongly influenced by the variety of training data, thus the dataset should be a representative sample of all possible variations of the data. For example, in an image classification task between apples and pears images, a network trained with a dataset formed only by images of red apples will lead to erroneous classifications when dealing with images of green apples, as it has never seen this variation on the data.

The labeled dataset is generally divided into two different splits: training split, which usually represents 80-90% of the whole dataset, and test split.

Design of the Network's Architecture

The design of the DNN architecture will determine its ability to fit the specific task for which it is designed. The architecture design must be based on the input that is going to receive (i.e., the data points) and the output that should produce (i.e., the labels of the data points). The DNNs' architecture refers to the order, type, and quantity of the different modules that built the network, such as layers of neurons, operations, and activation functions.

The design of the architecture is frequently limited by the capabilities of the specific hardware used for training, since the number of operations needed for training a DNN is elevated, resulting in high computational and storage requirements. Architectural constraints could also result from the limited capabilities of not the training hardware, but the devices where the network is to be deployed

Supervised Training

DNNs learn to perform a specific task from the data of the training split. The adjustable parameters that constitute the network's architecture are modified during training to produce an output as similar as possible to the ground truth. The initial values of these parameters can be randomly initialized or sampled from a sampling distribution.

To train the model, each data sample in the training set, or group of data (i.e., batch), is passed through the network to produce an output. The generated output is compared with the ground truth with a loss function, which is a comparison metric (e.g., mean square error) that usually depends on the task being performed by the model. Once the error is known, it

is backpropagated, computing the gradient of the loss function with respect to the network's parameters. Then, optimization algorithms are used to reduce the gradients by adjusting the parameter's values, thus optimizing the loss function. The optimization algorithms have a parameter called the learning rate that multiplies the value of the gradients, which allows controlling the degree of updating of the network parameters.

This process of optimization is repeated iteratively for each batch, the number of iterations (i.e., steps) required to process the entire train split is called an epoch. Usually, a subset of the training data (around 10-20 % of the training split) is chosen as validation data. After each epoch, and once the model has been updated with the epoch's training split, the validation data is passed through the model to compute the loss function. The loss function value obtained with the validation test split after each epoch provides feedback about how well the model is learning to generalize. For example, a significant increase of the validation loss along with a continuous decrease of the training loss could indicate the overfitting of the network. The validation data is also employed to adjust the different hyperparameters (e.g., optimizer, loss function, number of epochs, etc.) to obtain the best performance of the model.

Training is usually terminated when the loss function obtained with the training data is no longer minimized with the optimizer (i.e., there is no gradient) indicating that a local minimum or global minimum has been reached. However, training may be terminated before this happens if there is a continued stall or rise in the loss function obtained with the validation data, which could indicate that the model is overfitting.

Evaluation

The evaluation is performed over the set of data initially separated from the total dataset, the test split, and therefore it has never been seen by the network. The test data is fed into the trained DNN to be processed, thereby obtaining the label predicted for it, this operation is known as inference. After performing the inference over the test split, the labels obtained are compared, either qualitative or quantitative, with the ground truth to obtain an evaluation of the network's performance.

A.2 Encoder-Decoder Architectures

These architectures are composed of two main modules, the encoder, and the decoder, through which the data is processed sequentially. These structures are typically employed in variable sequence processing tasks with recurrent neural networks (RNNs), such as in language processing tasks, and in image processing, where they are formed by convolutional layers.

The encoder is used to 'encode' the input data into a more suitable representation, called latent space or feature vector, extracting from it the most relevant information for the task to be performed. In image processing, the encoder based on convolutional layers reduces the spatial dimensionality of the input images at the expense of increasing their channel dimension, in which each channel represents an image's feature. Therefore, translating the images into a list of representative features.

These features are then processed by the decoder, which retrieves the target output representation (e.g., saliency map, semantic segmentation, etc.) by decoding them. The stage between the encoder and decoder, where the input's feature vector can be obtained, is called the bottleneck.

A.3 Convolutional Layers

The traditionally called convolutional layers are a type of transformation frequently used in deep learning models for image processing. Convolutional layers extract features from images by convolving the image with kernel filters, with are 2D tensors of weights. The convolutional layer is defined by the number of filters and their shape. When training a model that includes convolutional layers, the kernel's weight values are learned. Thus, they are optimized to extract the most relevant features for the specific task in which the model is being trained.

Given an input tensor I with shape mxnxch and a convolutional layer with k number of kernel filters, the output tensor O will have shape mxnxk. For each kernel filter, the convolution operation performs an element-wise product between each element of the 2D kernel and the input tensor at each tensor's location. The results are summed to obtain the output value in the corresponding position of the output tensor, called a feature map. This operation is exemplified in Figure A.1.

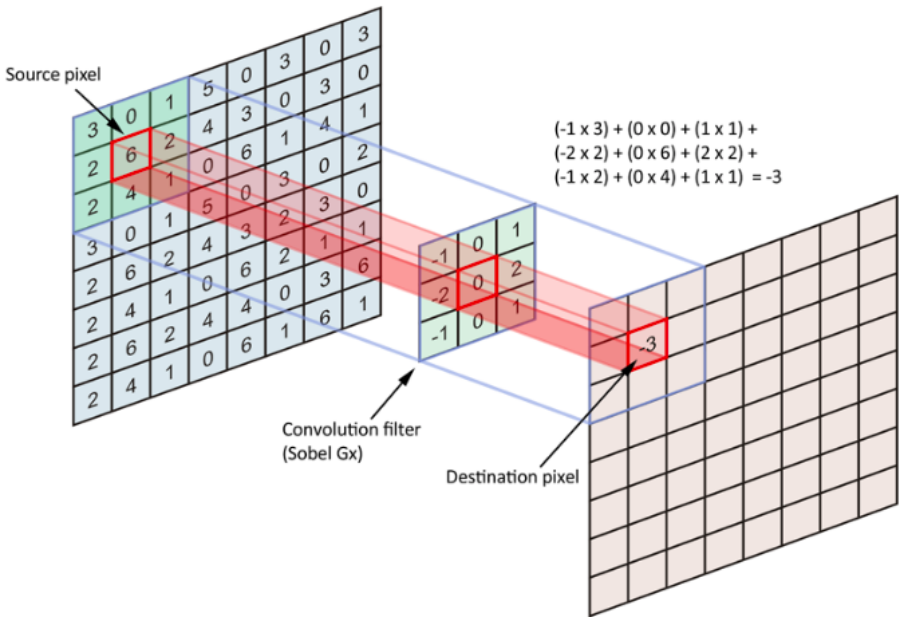


Figure A.1: Representation of the operations performed by a convolutional layer [6].

A.4 Long Short-Term Memory Cells

The Long Short-Term Memory (LSTMs) are used to process sequential data of variable length since they are capable to infer and store patterns between timesteps. They are recursive architectures that store previous information into the cell states, thus the output obtained for an input data point in a sequence is influenced by previous data points. The information outputted, stored, and forgotten by the LSTM is regulated by structures called gates.

An LSTM cell at each timestep t is defined by a hidden state h_t and a cell state c_t, whose values are determined by the response of its three self-parameterized forget, input, and output gates to the arrival of a new data point x_t. The LSTM architecture can be seen in Figure A.2.

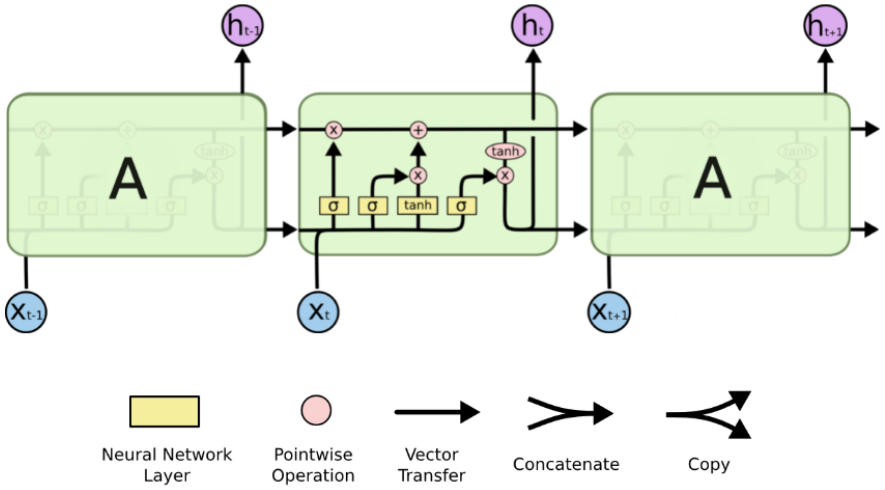


Figure A.2: Schematic representation of the operations performed on a LSTM cell (A) where t represents the timestamp, h the hidden state and c the cell state [63]

A.4.1 Cell state update

First, the forget gate f_t decides which information is going to be eliminated from the cell state c_t by multiplying each value of c_t by a number between 0 and 1. These values are obtained as shown in Equation A.1, where the input x_t is concatenated with the previous hidden state h_{t-1} and passed through a fully connected layer of neurons represented by its weights W_f and bias b_f. A sigma function σ is applied to the output of the fully connected layer to obtain the values between 0 and 1.

$$f_t = \sigma\left(W_f * [x_t, h_{t-1}] + b_f\right) \tag{A.1}$$

Then, the input gate i_t decides which values of the proposed new cell state g_t are input into the cell state c_t. The input gate also uses a sigmoid layer to produce a value between 0 and 1, which will be multiplied by each element of g_t, thus controlling the incoming information. Both are computed from the processing trough a fully-connected layer the concatenation of x_t and h_{t-1} as shown in Equations A.2 and A.3, where W and b are the weights and biases.

$$i_t = \sigma\left(W_i * [x_t, h_{t-1}] + b_i\right) \tag{A.2}$$

$$g_t = \tanh\left(W_g * [x_t, h_{t-1}] + b_g\right) \tag{A.3}$$

Therefore, the cell state at time t is:

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{A.4}$$

where $\odot$ is the Hadamard product.

A.4.2 Hidden state update

The hidden state h_t is considered the output of the LSTMs cell. The h_t is based on the current cell state, whose information is filtered by the output gate o_t, again using a sigmoid function to produce a value between 0 and 1. A hyperbolic tangent function is applied to c_t to help regulate the values flowing through the network, squishing them to always be between -1 and 1. Therefore, the hidden state and output gate values at time t are:

$$\begin{aligned} o_t &= \sigma\left(W_o * [x_t, h_{t-1}] + b_o\right) \\ h_t &= o_t \odot \tanh\left(c_t\right) \end{aligned} \tag{A.5}$$

where $\odot$ is the Hadamard product.

Appendix B. Quantitative Evaluation

This appendix presents the detailed results obtained in the quantitative evaluations performed in the comparison with previous works (Section 4.3) and the ablation study (Section 4.4). The scores obtained with the saliency metrics mentioned in Section 4.1 are presented in this appendix itemized by video, instead of averaged for all the videos belonging to the dataset.

The 360º videos in the VR-EyeTracking and Sports-360 datasets (Section 3.4) present varied content with different themes (e.g., films, sports, indoors, outdoors, etc.). This means that the performance of a model could greatly fluctuate from one video to another. Moreover, the model's performance also differs between frames, since as dynamic content, scenes change from frame to frame and can present significantly different scenarios within the same video. Therefore, the actual performance of a model for a specific video should not be directly extracted from the overall behavior, which is why this appendix provides the scores itemized by video.

B.1 Comparison with Previous Works

Tables B.1,B.2, B.3, and B.4 show the values obtained for each video in the VR-EyeTracking dataset for the proposed model, ATSal, CP-360, and Martin et al's [3] respectively (please refer to Section 4.3 for further details). The tables show the mean and the standard deviation of the CC, SIM, and KLDiv scores between each video's frames.

Tables B.5, B.6, B.7, and B.8 show the values obtained for each video in the Sports-360 dataset for the proposed model, ATSal, CP-360, and Martin et al's [3] respectively (please refer to Section 4.3 for further details). The tables show the mean and the standard deviation of the CC, SIM, and KLDiv scores between each video's frames.

B.2 Ablation Study

The ablation study performed in this master's thesis consisted of eight studies of the model using different setups, please refer to Section 4.4 for more information. Table B.9 indicates which elements were employed in each study, and Tables from B.10 to B.17 show the scores obtained with the saliency metrics itemized by video for each study. The different model configurations were evaluated with the VR-EyeTracking dataset (Section 3.4).

	Proposed Model					
	CC		SIM		KLDiv	
Video	Mean	Std	Mean	Std	Mean	Std
003	0.4285	0.1675	0.2386	0.0734	8.9560	1.6844
005	0.1773	0.0798	0.1410	0.0335	11.7037	1.0806
013	0.4063	0.1675	0.2162	0.0702	10.3207	1.6347
015	0.4183	0.1665	0.2213	0.0743	9.9883	1.7530
018	0.6395	0.1039	0.3571	0.0592	6.9805	1.1273
020	0.2923	0.1490	0.1642	0.0531	11.3344	1.4902
022	0.3316	0.1290	0.2033	0.0545	10.2527	1.2515
027	0.3956	0.2008	0.1812	0.0848	11.0605	1.7852
028	0.4142	0.1675	0.1851	0.0614	11.1041	1.3615
029	0.2787	0.1080	0.1630	0.0441	11.5675	1.0321
033	0.5941	0.2028	0.2633	0.0724	9.6076	1.4615
040	0.3211	0.1250	0.1689	0.0482	11.3080	1.2077
043	0.4011	0.1559	0.1899	0.0441	10.5007	1.1457
047	0.6503	0.0802	0.3131	0.0587	8.4859	1.2580
052	0.5075	0.1535	0.2248	0.0579	10.2931	1.0854
053	0.3483	0.1084	0.1929	0.0493	10.7719	1.1996
059	0.5506	0.1533	0.2393	0.0612	9.8382	1.2028
066	0.4175	0.1301	0.1780	0.0482	11.1533	1.0962
067	0.3557	0.1544	0.1869	0.0591	10.9474	1.3800
072	0.5171	0.2392	0.2735	0.1186	8.7108	2.3296
075	0.2392	0.1954	0.1730	0.0854	11.1876	2.2367
080	0.4335	0.2240	0.2514	0.0869	9.0619	1.8688
087	0.1805	0.0753	0.1672	0.0571	10.7468	1.9489
094	0.2633	0.1837	0.1909	0.0886	10.9085	2.2807
103	0.5027	0.2264	0.2618	0.0977	9.1019	1.7980
110	0.2573	0.1197	0.2020	0.0695	10.4149	1.9247
159	0.3843	0.1817	0.1854	0.0558	10.6763	1.4442
160	0.2976	0.0835	0.2467	0.0388	8.7627	1.5503
170	0.4291	0.1691	0.2456	0.0603	8.8430	1.1250
184	0.6546	0.2063	0.3143	0.0818	8.6354	1.6064
197	0.3664	0.1541	0.2028	0.0567	9.8454	1.5238
213	0.3165	0.1641	0.2500	0.0857	8.1738	2.4840

Table B.1: Mean and standard deviation of the CC, SIM, and KLDiv scores for each video in the VR-EyeTracking dataset with the proposed model.

	ATSal*					
	CC		SIM		KLDiv	
Video	Mean	Std	Mean	Std	Mean	Std
003	0.2947	0.1214	0.1268	0.0247	11.9346	0.6449
005	0.1681	0.0604	0.1385	0.0311	12.3208	1.0607
013	0.1935	0.1006	0.0968	0.0314	12.6177	0.7186
015	0.3298	0.1646	0.0862	0.0248	12.8471	0.6825
018	0.3025	0.1132	0.1299	0.0298	11.8355	0.6162
020	0.2602	0.0820	0.1091	0.0187	12.1810	0.4846
022	0.1469	0.0486	0.0919	0.0240	12.6511	0.5803
027	0.1335	0.0913	0.0560	0.0179	13.2455	0.4691
028	0.2516	0.0903	0.1168	0.0383	12.4079	0.7890
029	0.2487	0.0618	0.1253	0.0194	12.2819	0.5459
033	0.2353	0.0378	0.1007	0.0247	12.6018	0.6174
040	0.1423	0.0560	0.0769	0.0212	13.2113	0.5866
043	0.2361	0.0714	0.1249	0.0258	11.9620	0.6048
047	0.3086	0.1133	0.1253	0.0272	12.0116	0.5778
052	0.2316	0.0671	0.1165	0.0266	12.1139	0.7120
053	0.3840	0.1217	0.1303	0.0257	11.9343	0.5410
059	0.2721	0.1116	0.1129	0.0225	12.5219	0.5845
066	0.2333	0.0698	0.1026	0.0202	12.3047	0.4377
067	0.1279	0.0893	0.0763	0.0344	13.0386	0.9585
072	0.1395	0.0514	0.0662	0.0176	13.1789	0.4595
075	0.2133	0.0559	0.0975	0.0319	12.5272	0.6614
080	0.3369	0.1453	0.1286	0.0343	11.7417	0.8122
087	0.1773	0.0808	0.1433	0.0440	11.9137	1.4179
094	0.1176	0.0602	0.0548	0.0127	13.5021	0.3660
103	0.2855	0.1576	0.0935	0.0403	13.0494	0.8911
110	0.1623	0.0869	0.1215	0.0474	12.2149	1.1925
159	0.2309	0.0880	0.1106	0.0233	12.5973	0.6109
160	0.3130	0.0819	0.1703	0.0303	10.8138	0.6412
170	0.1881	0.0671	0.0837	0.0120	12.5898	0.2944
184	0.4936	0.0590	0.1247	0.0177	11.9306	0.4354
197	0.2403	0.0800	0.1108	0.0213	12.2938	0.6041
213	0.2738	0.1415	0.1170	0.0310	12.0061	0.7388

Table B.2: Mean and standard deviation of the CC, SIM, and KLDiv scores for each video in the VR-EyeTracking dataset with ATSal [1].∗ATSal was trained with the VR-EyeTracking dataset; its results are include for completeness.

	CP-360					
	CC		SIM		KLDiv	
Video	Mean	Std	Mean	Std	Mean	Std
003	0.3184	0.0555	0.1898	0.0230	12.8568	0.5042
005	0.3369	0.0489	0.2208	0.0268	12.1090	0.6250
013	0.2464	0.0378	0.1237	0.0291	14.3389	0.6777
015	0.2189	0.0489	0.1377	0.0348	14.1146	0.7745
018	0.3189	0.0509	0.1692	0.0335	13.3593	0.7103
020	0.2891	0.0424	0.1937	0.0258	12.7515	0.6356
022	0.2699	0.0573	0.1534	0.0361	13.6605	0.8330
027	0.2566	0.0599	0.1378	0.0390	14.0419	0.8801
028	0.2183	0.0675	0.1352	0.0429	14.1765	0.9702
029	0.1977	0.0540	0.1341	0.0418	14.1288	0.9798
033	0.1765	0.0539	0.1244	0.0379	14.4549	0.8380
040	0.2553	0.0575	0.1370	0.0393	14.0789	0.8995
043	0.2467	0.0488	0.1475	0.0295	13.7643	0.6951
047	0.3106	0.0477	0.1582	0.0240	13.5622	0.5129
052	0.2778	0.0545	0.1674	0.0294	13.3441	0.6772
053	0.3117	0.0382	0.1818	0.0275	13.1080	0.6022
059	0.2336	0.0409	0.1382	0.0223	14.0712	0.5179
066	0.1865	0.0319	0.1332	0.0300	14.2647	0.6543
067	0.2427	0.0496	0.1253	0.0281	14.3536	0.6488
072	0.1840	0.0383	0.1033	0.0282	14.9416	0.6354
075	0.2647	0.0613	0.1630	0.0460	13.5338	1.0572
080	0.2474	0.0635	0.1624	0.0323	13.5086	0.8219
087	0.3282	0.0653	0.2077	0.0362	12.4042	0.7741
094	0.2290	0.0459	0.1225	0.0279	14.4121	0.6360
103	0.2322	0.0522	0.1247	0.0463	14.4171	1.0202
110	0.2876	0.0724	0.1580	0.0345	13.4846	0.7401
159	0.2520	0.0469	0.1715	0.0219	13.3417	0.4882
160	0.3818	0.0470	0.1949	0.0230	12.6614	0.5254
170	0.2611	0.0379	0.1536	0.0209	13.7146	0.4700
184	0.1938	0.0443	0.1349	0.0244	14.2191	0.5542
197	0.2844	0.0447	0.1864	0.0284	12.9461	0.6408
213	0.3017	0.0428	0.1430	0.0241	13.9530	0.5268

Table B.3: Mean and standard deviation of the CC, SIM, and KLDiv scores for each video in the VR-EyeTracking dataset with CP-360 [2].

	Martin et al.'s [3]					
	CC		SIM		KLDiv	
Video	Mean	Std	Mean	Std	Mean	Std
003	0.1230	0.0268	0.0798	0.0140	13.5773	0.3375
005	0.1047	0.0445	0.0851	0.0154	13.4557	0.4721
013	0.2006	0.0620	0.0955	0.0252	13.2854	0.6353
015	0.1996	0.0555	0.0885	0.0245	13.4217	0.6105
018	0.2201	0.0312	0.1025	0.0129	13.0230	0.3199
020	0.0925	0.0250	0.0733	0.0105	13.7371	0.2726
022	0.1114	0.0481	0.0746	0.0211	13.7778	0.5786
027	0.1383	0.0455	0.0767	0.0192	13.7194	0.4694
028	0.0988	0.0779	0.0662	0.0291	14.2312	0.8689
029	0.1079	0.0596	0.0733	0.0288	13.9019	0.7536
033	0.1697	0.0414	0.0898	0.0242	13.5312	0.6092
040	0.1291	0.0380	0.0711	0.0237	14.0416	0.5830
043	0.2298	0.0347	0.1122	0.0172	12.8290	0.4253
047	0.2139	0.0390	0.0954	0.0118	13.1761	0.2962
052	0.2106	0.0525	0.1111	0.0208	12.8858	0.5271
053	0.0722	0.0373	0.0675	0.0148	14.2077	0.4376
059	0.1393	0.0254	0.0790	0.0107	13.6377	0.2820
066	0.1800	0.0308	0.0791	0.0167	13.5794	0.3971
067	0.1035	0.0354	0.0611	0.0211	14.2183	0.5846
072	0.1076	0.0501	0.0545	0.0180	14.3833	0.5081
075	0.0976	0.0420	0.0679	0.0233	13.9161	0.6492
080	0.1257	0.0874	0.0913	0.0342	13.6006	1.0241
087	0.1278	0.0678	0.0918	0.0274	13.4743	0.9450
094	0.1066	0.0322	0.0619	0.0150	14.1244	0.3873
103	0.1249	0.0419	0.0729	0.0288	13.9684	0.8004
110	0.1812	0.0671	0.1008	0.0290	13.2135	0.7335
159	0.1893	0.0338	0.0950	0.0151	13.1928	0.3880
160	0.2749	0.0281	0.1440	0.0128	12.2057	0.3809
170	0.1911	0.0357	0.1015	0.0148	13.1171	0.3782
184	0.2367	0.0269	0.0925	0.0160	13.3010	0.3739
197	0.1726	0.0246	0.0838	0.0125	13.3574	0.3134
213	0.1171	0.0575	0.0583	0.0149	14.2842	0.4443

Table B.4: Mean and standard deviation of the CC, SIM, and KLDiv scores for each video in the VR-EyeTracking dataset with Martin et al.'s [3].

	Proposed Model							Proposed Model					
Video	CC		SIM		KLDiv		Video	CC		SIM		KLDiv	
217	0.3171	0.1559	0.1296	0.0569	12.1710	1.3809	269	0.0118	0.0155	0.0283	0.0183	15.4930	0.9625
218	0.2063	0.0849	0.0947	0.0237	13.0641	0.6229	270	0.3242	0.1877	0.1049	0.0535	12.7899	1.2350
219	0.1351	0.0845	0.0580	0.0243	13.9510	0.6625	271	0.3321	0.1252	0.1464	0.0273	11.8987	0.5378
220	0.2608	0.1239	0.1524	0.0567	11.6985	1.4605	272	0.3031	0.1468	0.1295	0.0318	12.0845	0.6910
221	0.1940	0.0918	0.1084	0.0351	12.9689	0.8242	273	0.3856	0.1381	0.1510	0.0430	11.6894	0.9407
222	0.4912	0.1800	0.1887	0.0682	10.8263	1.5238	274	0.0720	0.0510	0.0426	0.0132	14.2998	0.4035
223	0.3788	0.1592	0.1987	0.0639	10.6089	1.4157	275	0.0895	0.0747	0.0771	0.0495	13.8101	1.5601
224	0.1824	0.1049	0.1110	0.0391	12.7950	1.0357	276	0.3291	0.1500	0.2115	0.0685	10.1173	1.6259
225	0.1569	0.1030	0.0887	0.0362	13.4349	0.9642	277	0.4169	0.2022	0.1337	0.0578	12.1699	1.2037
226	0.2997	0.1963	0.0924	0.0446	12.9954	0.9987	278	0.0027	0.0111	0.0176	0.0122	15.7344	0.9457
227	0.0513	0.0464	0.0488	0.0180	14.4725	0.6737	279	0.2639	0.1244	0.1488	0.0518	11.8563	1.2658
228	0.1360	0.0371	0.0866	0.0173	13.2667	0.4577	280	0.4215	0.1547	0.1806	0.0659	10.6553	1.9776
229	0.0517	0.0412	0.0490	0.0206	14.2593	0.7301	281	0.2668	0.0597	0.1512	0.0236	11.5533	0.6622
230	0.2868	0.1338	0.1171	0.0371	12.4623	0.9362	282	0.3195	0.1857	0.1345	0.0554	11.9555	1.4528
231	0.4228	0.1896	0.1616	0.0394	11.4523	0.8681	283	0.1691	0.1051	0.1131	0.0434	13.0476	1.0736
232	0.1104	0.0441	0.0738	0.0251	13.6332	0.6557	284	0.0707	0.0444	0.0618	0.0217	14.1279	0.7241
233	0.3628	0.1544	0.1355	0.0469	12.1155	0.9586	285	0.1286	0.0763	0.0581	0.0293	13.9529	0.8357
234	0.2556	0.1109	0.1284	0.0440	12.3032	1.0327	286	0.0114	0.0239	0.0307	0.0202	15.2546	1.1105
235	0.3527	0.1538	0.1231	0.0402	12.3539	0.8592	287	0.2227	0.1130	0.1290	0.0371	12.4527	0.9351
236	0.1758	0.0994	0.0924	0.0320	13.0787	0.8446	288	0.2854	0.1242	0.1240	0.0375	12.3255	0.9084
237	0.2524	0.1010	0.1554	0.0524	11.5770	1.2971	289	0.2400	0.1777	0.1055	0.0450	13.0275	1.1501
238	0.1886	0.0752	0.0966	0.0240	12.9992	0.6726	290	0.0938	0.0643	0.0866	0.0365	13.4401	1.2318
239	0.3021	0.0975	0.1446	0.0305	11.8375	0.7795	291	0.3103	0.1559	0.1060	0.0466	12.6361	1.1217
240	0.4043	0.1464	0.1646	0.0554	11.3388	1.5093	292	0.1051	0.0591	0.0796	0.0257	13.6677	0.7211
241	0.1456	0.0697	0.0627	0.0178	13.8784	0.5176	293	0.2422	0.1211	0.1261	0.0418	12.4056	0.9340
242	0.2884	0.1787	0.1740	0.0898	11.0400	2.2297	294	0.2043	0.0580	0.1065	0.0248	12.6850	0.6656
243	0.3841	0.1226	0.1754	0.0329	11.2069	0.7489	295	0.3160	0.1117	0.1475	0.0369	11.9317	0.7974
244	0.2253	0.1023	0.1244	0.0430	12.1798	1.2431	296	0.0919	0.0714	0.0686	0.0406	13.9617	1.2456
245	0.1077	0.0551	0.0764	0.0267	13.5466	0.7708	297	0.2723	0.1973	0.1208	0.0650	12.4236	1.6231
246	0.1753	0.1286	0.1240	0.0644	12.6434	1.8907	298	0.2065	0.1415	0.1411	0.0610	11.7646	1.9794
247	0.2830	0.1459	0.1350	0.0534	12.1157	1.4111	299	0.3209	0.0922	0.1505	0.0474	11.7125	1.1624
248	0.0427	0.0509	0.0396	0.0245	14.7307	1.0662	300	0.1089	0.0644	0.0784	0.0276	13.5475	0.7564
249	0.0295	0.0344	0.0412	0.0185	14.7312	0.6496	301	0.3887	0.1466	0.1615	0.0471	11.4403	1.0287
250	0.3479	0.1232	0.1641	0.0329	11.4165	0.8306	302	0.0945	0.0590	0.0554	0.0216	14.1144	0.6839
251	0.3245	0.1270	0.1892	0.0511	10.5540	1.4367	303	0.4186	0.1763	0.1489	0.0541	11.7598	1.1735
252	-0.0039	0.0136	0.0196	0.0125	16.0198	0.6759	304	0.2947	0.1247	0.1881	0.0556	10.6246	1.6169
253	0.0511	0.0588	0.0410	0.0250	14.8519	1.1520	305	0.0946	0.0689	0.0820	0.0239	13.7676	0.7921
254	0.3429	0.1383	0.1871	0.0408	10.6921	1.1608	306	0.4126	0.0879	0.1713	0.0355	11.1323	0.8192
255	0.0058	0.0166	0.0312	0.0213	15.5804	0.9055	307	0.3659	0.1634	0.1656	0.0510	11.3613	1.0798
256	0.0079	0.0164	0.0260	0.0189	15.6081	0.9718	308	0.0090	0.0302	0.0256	0.0241	15.4164	1.1526
257	0.5594	0.1538	0.1973	0.0516	10.7550	0.9945	309	0.1303	0.0690	0.0758	0.0232	13.6253	0.7139
258	0.4163	0.1841	0.1735	0.0505	11.0916	1.2331	310	0.3856	0.1091	0.1464	0.0387	11.6998	0.9372
259	0.5139	0.1945	0.2025	0.0771	10.7654	1.6627	311	0.3491	0.1371	0.1279	0.0423	12.1598	0.9766
260	0.3416	0.1187	0.1558	0.0370	11.6983	0.8219	312	0.2876	0.1158	0.1864	0.0549	10.5434	1.4978
261	0.3227	0.2048	0.1680	0.0784	11.0109	2.1668	313	0.3986	0.1334	0.1477	0.0392	11.7560	0.9326
262	0.4894	0.2293	0.2027	0.0804	10.8192	1.5204	314	0.3628	0.2203	0.1931	0.0924	10.6921	2.3499
263	0.2278	0.1076	0.1572	0.0519	11.4131	1.5216	315	0.0958	0.0456	0.0734	0.0204	13.6603	0.6277
264	0.1700	0.0911	0.0892	0.0497	13.1952	1.2049	316	0.4208	0.2079	0.1889	0.0799	10.9946	1.4456
265	0.1916	0.1053	0.0908	0.0275	13.0897	0.7367	317	0.2287	0.1202	0.1354	0.0573	12.2514	1.3489
266	0.2881	0.1234	0.1415	0.0437	11.9960	1.0699	318	0.2070	0.1425	0.1246	0.0668	12.3415	1.7890
267	0.0642	0.0647	0.0773	0.0371	13.7009	1.6177	319	0.3614	0.1538	0.1560	0.0426	11.6033	0.9530
268	0.1789	0.1000	0.1032	0.0301	12.8524	0.9459	320	0.2058	0.0690	0.1137	0.0288	12.6145	0.7007

Table B.5: Mean and standard deviation of the CC, SIM, and KLDiv scores for each video in the Sports-360 dataset with the proposed model.

	ATSal							ATSal					
Video	CC		SIM		KLDiv		Video	CC		SIM		KLDiv	
217	0.1636	0.0690	0.0705	0.0252	13.0839	0.6402	269	0.2827	0.1004	0.0745	0.0195	12.8182	0.4422
218	0.1303	0.0459	0.0688	0.0255	13.2000	0.5824	270	0.1900	0.0853	0.0690	0.0241	13.0137	0.5498
219	0.1293	0.0837	0.0604	0.0218	13.5311	0.6613	271	0.1110	0.0463	0.0749	0.0202	13.1688	0.5690
220	0.1330	0.0620	0.0693	0.0226	13.1941	0.5993	272	0.1461	0.0365	0.0851	0.0174	12.8305	0.4604
221	0.2018	0.0552	0.0782	0.0177	12.8762	0.4565	273	0.0569	0.0274	0.0507	0.0173	13.7715	0.5136
222	0.1891	0.0710	0.0623	0.0145	13.3681	0.4386	274	0.0473	0.0430	0.0371	0.0139	14.3024	0.5895
223	0.3319	0.1148	0.1210	0.0230	12.6429	0.6761	275	0.1265	0.0573	0.0679	0.0318	13.2417	0.7809
224	0.1650	0.0829	0.0924	0.0327	12.7619	0.7993	276	0.1069	0.0385	0.0688	0.0166	13.3106	0.4421
225	0.0995	0.0527	0.0648	0.0195	13.5428	0.6096	277	0.1040	0.0711	0.0493	0.0189	13.8308	0.6888
226	0.1424	0.0550	0.0428	0.0131	13.6177	0.3908	278	0.2485	0.1330	0.0651	0.0161	13.0031	0.4132
227	0.3030	0.0795	0.1083	0.0157	12.4033	0.4797	279	0.1486	0.0525	0.0799	0.0171	12.9974	0.4324
228	0.1360	0.0491	0.0909	0.0196	12.8616	0.5685	280	0.2133	0.0722	0.1055	0.0246	12.5222	0.5772
229	0.1143	0.0435	0.0621	0.0123	13.2037	0.4502	281	0.2135	0.0588	0.1253	0.0229	11.8533	0.5934
230	0.1379	0.0829	0.0640	0.0219	13.5814	0.6293	282	0.1227	0.0651	0.0685	0.0196	13.4770	0.6208
231	0.0755	0.0452	0.0543	0.0206	13.6593	0.5502	283	0.2990	0.0474	0.0909	0.0105	12.6387	0.2460
232	0.1260	0.0557	0.0718	0.0294	13.1125	0.7689	284	0.2891	0.1220	0.1081	0.0316	12.5312	0.8492
233	0.1791	0.0538	0.0746	0.0154	12.8874	0.3829	285	0.2115	0.0900	0.0711	0.0268	13.0539	0.6856
234	0.1878	0.0684	0.1033	0.0277	12.5439	0.6377	286	0.1963	0.0890	0.0901	0.0265	12.7078	0.5824
235	0.1380	0.0912	0.0715	0.0328	13.4169	0.9141	287	0.1719	0.0434	0.1087	0.0225	12.4248	0.6397
236	0.1400	0.0686	0.0686	0.0190	13.1571	0.5497	288	0.2942	0.1153	0.0819	0.0274	12.6877	0.6216
237	0.2015	0.0775	0.1035	0.0222	12.4500	0.4710	289	0.1418	0.1114	0.0658	0.0344	14.5963	2.0640
238	0.0723	0.0341	0.0555	0.0192	13.7344	0.6393	290	0.1330	0.0517	0.0675	0.0237	13.1357	0.5751
239	0.1249	0.0543	0.0712	0.0203	13.1904	0.6042	291	0.0464	0.0331	0.0361	0.0165	14.0323	0.5320
240	0.1697	0.0672	0.0916	0.0219	12.7739	0.5604	292	0.1571	0.0457	0.0890	0.0177	12.7178	0.4951
241	0.1348	0.0640	0.0662	0.0220	13.3704	0.5676	293	0.1882	0.0505	0.0846	0.0236	12.9264	0.5655
242	0.1948	0.1058	0.0901	0.0315	12.8989	0.8486	294	0.0959	0.0386	0.0719	0.0181	13.3243	0.5326
243	0.1526	0.0651	0.0916	0.0226	13.1507	0.5936	295	0.2657	0.0677	0.1218	0.0253	12.1217	0.6250
244	0.1978	0.0749	0.0827	0.0187	12.7897	0.5047	296	0.1418	0.0812	0.0671	0.0189	13.3032	0.5193
245	0.1357	0.0753	0.0781	0.0264	13.0839	0.6530	297	0.1177	0.0715	0.0478	0.0141	13.7299	0.4773
246	0.1817	0.1192	0.0764	0.0321	13.2186	1.1646	298	0.1784	0.0751	0.0967	0.0224	12.7227	0.5806
247	0.2011	0.0824	0.0916	0.0279	13.0439	0.9089	299	0.1318	0.0996	0.0786	0.0375	13.1645	1.0288
248	0.1916	0.1572	0.0566	0.0207	13.3018	0.5771	300	0.1790	0.0952	0.0879	0.0347	12.9639	0.9684
249	0.0559	0.0467	0.0425	0.0118	13.6480	0.3666	301	0.1530	0.0619	0.0706	0.0204	13.2346	0.5548
250	0.2676	0.1057	0.1143	0.0296	12.3693	0.6965	302	0.1653	0.0852	0.0661	0.0197	13.3409	0.5468
251	0.2273	0.0525	0.1258	0.0222	11.9448	0.5896	303	0.0866	0.0810	0.0415	0.0247	13.8327	0.8641
252	0.1659	0.0672	0.0744	0.0130	12.8438	0.3218	304	0.1732	0.0641	0.1024	0.0210	12.6846	0.5291
253	0.3489	0.1398	0.0886	0.0179	12.5563	0.4180	305	0.1942	0.0985	0.1050	0.0447	12.5045	1.1925
254	0.2352	0.0766	0.1247	0.0282	12.0422	0.7228	306	0.3933	0.1007	0.1130	0.0221	11.9805	0.4620
255	0.2248	0.0653	0.1008	0.0212	12.3885	0.4832	307	0.0943	0.0369	0.0643	0.0120	13.3262	0.3298
256	0.2996	0.1489	0.0822	0.0276	12.7050	0.5622	308	0.1049	0.0741	0.0513	0.0192	13.7546	0.6420
257	0.4986	0.1441	0.1156	0.0164	12.1272	0.4693	309	0.2003	0.0566	0.0825	0.0100	12.7460	0.2592
258	0.1628	0.0552	0.0774	0.0197	13.0344	0.5355	310	0.2019	0.0807	0.1015	0.0246	12.6994	0.6571
259	0.3601	0.1631	0.1072	0.0318	12.2475	0.6182	311	0.2011	0.0686	0.0789	0.0207	12.7249	0.4970
260	0.1669	0.0632	0.0814	0.0208	12.7711	0.5111	312	0.2492	0.0416	0.0910	0.0208	12.7351	0.5413
261	0.1509	0.0892	0.0759	0.0323	13.2135	1.0322	313	0.2044	0.0849	0.0939	0.0197	12.8113	0.5684
262	0.1034	0.0457	0.0627	0.0266	13.3804	0.7040	314	0.2193	0.1501	0.0806	0.0289	12.8635	0.6986
263	0.2005	0.0725	0.1240	0.0377	12.1000	1.0229	315	0.1111	0.0389	0.0720	0.0148	13.1812	0.4153
264	0.3416	0.1050	0.0974	0.0280	12.4280	0.5999	316	0.2177	0.0829	0.0799	0.0202	12.6848	0.4910
265	0.1712	0.0548	0.0710	0.0155	13.0778	0.3721	317	0.1859	0.0878	0.0876	0.0221	12.9216	0.5945
266	0.1428	0.0733	0.0832	0.0313	13.1295	0.8174	318	0.1070	0.0673	0.0596	0.0183	13.8050	0.6603
267	0.2191	0.0929	0.1161	0.0303	12.2355	0.9080	319	0.1458	0.0605	0.0669	0.0175	13.1358	0.4483
268	0.1193	0.0764	0.0863	0.0438	13.4598	1.3017	320	0.1485	0.0537	0.0870	0.0252	12.8467	0.6605

Table B.6: Mean and standard deviation of the CC, SIM, and KLDiv scores for each video in the Sports-360 dataset with ATSal [1].

	CP-360							CP-360					
Video	CC		SIM		KLDiv		Video	CC		SIM		KLDiv	
217	0.1953	0.0516	0.0860	0.0250	15.1481	0.5900	269	0.2103	0.0545	0.0987	0.0273	14.9292	0.6309
218	0.2618	0.0617	0.1186	0.0359	14.4409	0.8452	270	0.2121	0.0535	0.0948	0.0311	14.9771	0.7372
219	0.1926	0.0452	0.0739	0.0237	15.4690	0.5617	271	0.2089	0.0557	0.1190	0.0236	14.4439	0.5565
220	0.1846	0.0529	0.0842	0.0258	15.1836	0.6295	272	0.2295	0.0407	0.1311	0.0242	14.1629	0.5559
221	0.2089	0.0505	0.1098	0.0274	14.6888	0.6417	273	0.2089	0.0411	0.1026	0.0204	14.8064	0.4634
222	0.1578	0.0273	0.0797	0.0196	15.3620	0.4480	274	0.1530	0.0453	0.0823	0.0278	15.2860	0.6751
223	0.2590	0.0414	0.1305	0.0250	14.1984	0.5722	275	0.1827	0.0566	0.0861	0.0387	15.2205	0.9171
224	0.2147	0.0649	0.1162	0.0267	14.5051	0.5997	276	0.2236	0.0377	0.1074	0.0174	14.6710	0.3831
225	0.1859	0.0541	0.1126	0.0281	14.5418	0.6485	277	0.1676	0.0444	0.0735	0.0219	15.5124	0.5021
226	0.1013	0.0370	0.0571	0.0217	15.9875	0.5101	278	0.2267	0.0340	0.1084	0.0325	14.6942	0.7118
227	0.2221	0.0398	0.1180	0.0200	14.4039	0.4565	279	0.2535	0.0380	0.1089	0.0195	14.6455	0.4357
228	0.2912	0.0318	0.1625	0.0185	13.3394	0.4238	280	0.2129	0.0335	0.1204	0.0215	14.4610	0.5016
229	0.2760	0.0529	0.1253	0.0213	14.2369	0.4871	281	0.3115	0.0367	0.1838	0.0180	12.9518	0.4301
230	0.2167	0.0505	0.0864	0.0276	15.1621	0.6610	282	0.1258	0.0368	0.0688	0.0245	15.6331	0.5918
231	0.2028	0.0700	0.1038	0.0406	14.7721	0.9704	283	0.1733	0.0278	0.1052	0.0140	14.8893	0.3065
232	0.2136	0.0661	0.1079	0.0427	14.6700	0.9987	284	0.1487	0.0391	0.0825	0.0163	15.3229	0.3999
233	0.1968	0.0410	0.0975	0.0203	14.9273	0.4528	285	0.1792	0.0564	0.0805	0.0275	15.3387	0.6893
234	0.2355	0.0587	0.1124	0.0348	14.5645	0.8472	286	0.2515	0.0431	0.1242	0.0248	14.3337	0.5547
235	0.1926	0.0345	0.0832	0.0230	15.2684	0.4905	287	0.2563	0.0411	0.1328	0.0180	14.1122	0.3868
236	0.2547	0.0585	0.1174	0.0257	14.4189	0.5833	288	0.2280	0.0616	0.0981	0.0323	14.9029	0.7667
237	0.2744	0.0447	0.1278	0.0310	14.2253	0.7468	289	0.1567	0.0365	0.0855	0.0233	15.1464	0.5547
238	0.1274	0.0382	0.0840	0.0245	15.2822	0.5960	290	0.2227	0.0468	0.1163	0.0280	14.4532	0.6915
239	0.2157	0.0466	0.1115	0.0221	14.5481	0.4575	291	0.1525	0.0553	0.0827	0.0299	15.2722	0.7165
240	0.1836	0.0478	0.0961	0.0192	14.9279	0.4447	292	0.2344	0.0394	0.1228	0.0206	14.3333	0.4672
241	0.1877	0.0510	0.0844	0.0200	15.2315	0.4653	293	0.1873	0.0312	0.0932	0.0200	15.0212	0.4715
242	0.2377	0.0650	0.1108	0.0209	14.5572	0.4903	294	0.2413	0.0361	0.1263	0.0193	14.2092	0.4343
243	0.2241	0.0456	0.1115	0.0236	14.6038	0.5539	295	0.1965	0.0323	0.1071	0.0198	14.7521	0.4298
244	0.2785	0.0439	0.1174	0.0201	14.3451	0.4236	296	0.1878	0.0770	0.0929	0.0354	15.0809	0.8380
245	0.2492	0.0480	0.1323	0.0266	14.0760	0.6233	297	0.1938	0.0642	0.0890	0.0256	15.1262	0.6147
246	0.1773	0.0609	0.0915	0.0311	15.0376	0.7456	298	0.2174	0.0476	0.1129	0.0265	14.5344	0.6222
247	0.2338	0.0363	0.1203	0.0243	14.3802	0.5658	299	0.2058	0.0753	0.1004	0.0368	14.8522	0.9283
248	0.1486	0.0832	0.0870	0.0350	15.2207	0.7922	300	0.1647	0.0576	0.0937	0.0316	15.0126	0.7659
249	0.1096	0.0829	0.1179	0.0372	14.5292	1.0090	301	0.2185	0.0349	0.1088	0.0295	14.6559	0.6459
250	0.2825	0.0494	0.1436	0.0275	13.8826	0.6245	302	0.1880	0.0305	0.0810	0.0216	15.2444	0.4857
251	0.2719	0.0334	0.1420	0.0194	13.8524	0.4441	303	0.1680	0.0652	0.0697	0.0262	15.5248	0.6664
252	0.2059	0.0501	0.1231	0.0235	14.3561	0.5436	304	0.2116	0.0519	0.1063	0.0232	14.7037	0.5372
253	0.2079	0.0366	0.1033	0.0189	14.8329	0.4275	305	0.1785	0.0475	0.0947	0.0298	14.9717	0.7049
254	0.2308	0.0615	0.1334	0.0267	14.0786	0.6218	306	0.2996	0.0510	0.1102	0.0201	14.5536	0.4652
255	0.2695	0.0468	0.1376	0.0274	14.0115	0.6304	307	0.2558	0.0384	0.1168	0.0215	14.4626	0.4495
256	0.2140	0.0586	0.1175	0.0290	14.5532	0.6465	308	0.1882	0.0628	0.0861	0.0300	15.1580	0.7129
257	0.2812	0.0400	0.1087	0.0231	14.6100	0.5188	309	0.2316	0.0416	0.1156	0.0222	14.4528	0.5129
258	0.2111	0.0405	0.1146	0.0239	14.5349	0.5499	310	0.2512	0.0355	0.1003	0.0203	14.8235	0.4406
259	0.2318	0.0387	0.1114	0.0272	14.6497	0.5970	311	0.2011	0.0749	0.1065	0.0318	14.7490	0.7874
260	0.1873	0.0465	0.0976	0.0307	14.9259	0.7266	312	0.2086	0.0348	0.1048	0.0206	14.7709	0.4541
261	0.1948	0.0671	0.1154	0.0336	14.5505	0.8092	313	0.2335	0.0750	0.1051	0.0286	14.7425	0.6911
262	0.2014	0.0631	0.1019	0.0357	14.8562	0.8137	314	0.2209	0.0584	0.0792	0.0287	15.2493	0.7378
263	0.2959	0.0427	0.1372	0.0272	13.9832	0.6197	315	0.2120	0.0399	0.1130	0.0177	14.5342	0.4024
264	0.2534	0.0588	0.0976	0.0521	14.8804	1.1759	316	0.2444	0.0698	0.1075	0.0382	14.6633	0.8990
265	0.1925	0.0462	0.1095	0.0320	14.6786	0.7451	317	0.1952	0.0579	0.0959	0.0228	14.9388	0.5489
266	0.2279	0.0465	0.1197	0.0278	14.4053	0.6148	318	0.1876	0.0571	0.1013	0.0303	14.8134	0.7109
267	0.2357	0.0499	0.1256	0.0199	14.2654	0.4599	319	0.1957	0.0391	0.0956	0.0193	14.9597	0.4765
268	0.1691	0.0562	0.0962	0.0280	14.9906	0.7003	320	0.2360	0.0664	0.1129	0.0285	14.5579	0.6750

Table B.7: Mean and standard deviation of the CC, SIM, and KLDiv scores for each video in the Sports-360 dataset with CP-360 [2].

	Martin et al.'s							Martin et al.'s					
Video	CC		SIM		KLDiv		Video	CC		SIM		KLDiv	
217	0.1996	0.0456	0.0750	0.0160	13.7369	0.4104	269	0.2005	0.0487	0.0711	0.0132	13.7733	0.3291
218	0.1758	0.0442	0.0710	0.0198	13.8260	0.5181	270	0.1025	0.0468	0.0455	0.0154	14.4540	0.4656
219	0.1232	0.0509	0.0496	0.0173	14.4719	0.5141	271	0.1045	0.0276	0.0715	0.0187	13.8960	0.5301
220	0.1521	0.0484	0.0609	0.0179	14.0305	0.5059	272	0.1988	0.0418	0.0863	0.0136	13.3899	0.3430
221	0.1292	0.0489	0.0658	0.0199	14.0260	0.5688	273	0.1411	0.0311	0.0636	0.0125	13.9969	0.3316
222	0.1400	0.0661	0.0519	0.0176	14.3408	0.5269	274	0.0345	0.0342	0.0317	0.0121	14.9858	0.4232
223	0.1597	0.0296	0.0797	0.0140	13.7468	0.3551	275	0.1715	0.0702	0.0740	0.0325	13.8803	0.8898
224	0.1658	0.0585	0.0826	0.0250	13.6844	0.6618	276	0.0885	0.0533	0.0485	0.0155	14.4803	0.5139
225	0.0453	0.0290	0.0434	0.0118	14.6183	0.3756	277	0.1504	0.0486	0.0577	0.0182	14.2805	0.5687
226	0.1345	0.0487	0.0453	0.0161	14.4509	0.4984	278	0.1661	0.0574	0.0603	0.0169	14.0517	0.4523
227	0.0893	0.0319	0.0557	0.0102	14.2226	0.3111	279	0.0962	0.0263	0.0609	0.0112	14.2646	0.3115
228	0.1142	0.0319	0.0742	0.0137	13.8523	0.4046	280	0.2327	0.0396	0.0974	0.0226	13.3057	0.5505
229	0.1128	0.0329	0.0600	0.0127	14.1171	0.3556	281	0.2400	0.0318	0.1158	0.0132	12.6514	0.3195
230	0.1701	0.0518	0.0570	0.0181	14.1571	0.4826	282	0.1554	0.0588	0.0610	0.0250	14.1966	0.6974
231	0.1520	0.0542	0.0711	0.0259	13.8655	0.6927	283	0.0991	0.0136	0.0470	0.0067	14.4298	0.1744
232	0.1072	0.0563	0.0618	0.0247	14.2395	0.7735	284	0.1482	0.0409	0.0617	0.0151	14.0805	0.4417
233	0.2003	0.0454	0.0873	0.0143	13.5292	0.3754	285	0.1440	0.0537	0.0529	0.0209	14.2566	0.6041
234	0.1948	0.0465	0.0873	0.0237	13.5330	0.6218	286	0.1629	0.0560	0.0706	0.0180	13.8159	0.4268
235	0.1036	0.0546	0.0483	0.0171	14.5503	0.5401	287	0.1938	0.0469	0.0980	0.0179	13.2417	0.4644
236	0.1051	0.0503	0.0624	0.0150	14.1973	0.5477	288	0.1110	0.0410	0.0435	0.0145	14.4259	0.4399
237	0.1852	0.0650	0.1036	0.0289	13.2813	0.8077	289	0.1551	0.0799	0.0701	0.0231	13.8279	0.5283
238	0.1022	0.0182	0.0539	0.0104	14.2556	0.2957	290	0.1666	0.0353	0.0746	0.0130	13.7281	0.3387
239	0.1481	0.0461	0.0689	0.0128	13.8224	0.3656	291	0.0333	0.0346	0.0320	0.0157	14.8558	0.5623
240	0.2029	0.0654	0.0870	0.0172	13.4785	0.4424	292	0.1837	0.0423	0.0848	0.0167	13.5202	0.4222
241	0.1503	0.0432	0.0533	0.0141	14.2942	0.3893	293	0.1099	0.0192	0.0609	0.0102	14.2036	0.2906
242	0.1498	0.0604	0.0687	0.0211	13.9087	0.5729	294	0.0982	0.0383	0.0613	0.0139	14.1312	0.4005
243	0.1815	0.0505	0.0865	0.0223	13.5982	0.5870	295	0.2517	0.0467	0.1050	0.0191	13.0797	0.4683
244	0.0983	0.0717	0.0589	0.0228	14.4620	0.7858	296	0.1196	0.0247	0.0464	0.0135	14.4755	0.3883
245	0.1357	0.0565	0.0683	0.0198	13.9344	0.5198	297	0.1419	0.0465	0.0517	0.0130	14.2188	0.3780
246	0.1161	0.0679	0.0514	0.0188	14.3576	0.5509	298	0.1613	0.0448	0.0739	0.0160	13.8488	0.3914
247	0.1511	0.0344	0.0718	0.0191	13.7702	0.5109	299	0.2011	0.0723	0.0831	0.0288	13.5642	0.7603
248	0.0905	0.0715	0.0481	0.0250	14.4821	0.7864	300	0.1141	0.0527	0.0648	0.0208	14.1176	0.5112
249	0.0919	0.0373	0.0561	0.0150	14.2107	0.4103	301	0.1115	0.0507	0.0582	0.0172	14.2238	0.5047
250	0.1805	0.0439	0.0864	0.0153	13.5543	0.4201	302	0.1722	0.0603	0.0665	0.0161	13.9134	0.4147
251	0.2115	0.0382	0.0905	0.0138	13.2672	0.3650	303	0.1059	0.0901	0.0456	0.0258	14.5685	0.8185
252	0.1222	0.0257	0.0588	0.0093	14.0436	0.2338	304	0.2206	0.0327	0.0875	0.0146	13.4025	0.3655
253	0.1740	0.0394	0.0693	0.0126	13.8781	0.3319	305	0.1411	0.0685	0.0720	0.0265	14.0995	0.9308
254	0.2162	0.0409	0.1046	0.0159	13.0413	0.4049	306	0.1637	0.0365	0.0631	0.0130	13.8898	0.3456
255	0.1894	0.0422	0.0876	0.0176	13.5007	0.4484	307	0.1402	0.0374	0.0786	0.0141	13.8140	0.3692
256	0.1646	0.0466	0.0671	0.0148	13.9063	0.3844	308	0.1260	0.0557	0.0507	0.0180	14.2803	0.4902
257	0.1425	0.0253	0.0607	0.0145	14.0072	0.3905	309	0.1508	0.0500	0.0720	0.0146	13.8173	0.4162
258	0.1183	0.0496	0.0632	0.0186	14.0935	0.5589	310	0.1679	0.0260	0.0555	0.0098	14.1743	0.2868
259	0.0819	0.0289	0.0499	0.0150	14.3976	0.4178	311	0.2211	0.0545	0.0899	0.0248	13.3792	0.6469
260	0.1581	0.0485	0.0744	0.0210	13.7728	0.5563	312	0.1894	0.0354	0.0819	0.0136	13.5538	0.3595
261	0.1237	0.0615	0.0685	0.0244	13.9742	0.6704	313	0.1372	0.0558	0.0683	0.0241	14.0190	0.6522
262	0.1149	0.0542	0.0554	0.0238	14.2670	0.6332	314	0.1124	0.0713	0.0495	0.0239	14.5082	0.7244
263	0.1896	0.0431	0.0969	0.0194	13.2001	0.4998	315	0.1762	0.0370	0.0821	0.0149	13.6134	0.3988
264	0.1592	0.0371	0.0551	0.0258	14.1253	0.6874	316	0.1706	0.0403	0.0669	0.0201	13.8609	0.5225
265	0.1466	0.0246	0.0655	0.0150	14.0066	0.3963	317	0.1515	0.0475	0.0699	0.0153	13.9347	0.4424
266	0.1647	0.0343	0.0849	0.0200	13.4611	0.4324	318	0.1064	0.0562	0.0544	0.0187	14.3220	0.5520
267	0.0889	0.0396	0.0639	0.0120	14.2063	0.2701	319	0.1513	0.0285	0.0754	0.0127	13.8162	0.3226
268	0.0956	0.0760	0.0618	0.0332	14.8713	1.3454	320	0.1685	0.0334	0.0765	0.0155	13.7096	0.3885

Table B.8: Mean and standard deviation of the CC, SIM, and KLDiv scores for each video in the Sports-360 dataset with Martin et al.'s [3].

208x106	320x240	Spherical ConvLSTM	Optical Flow	Depth	Weighted KLDiv	Sampled KLDiv	Traditional KLDiv	Ablation Study
✓	-	✓	✓	-	✓	-	-	1
-	✓	✓	✓	-	✓	-	-	2
-	✓	-	✓	-	✓	-	-	3
-	✓	✓	✓	-	-	✓	-	4
-	✓	✓	✓	-	-	-	✓	5
-	✓	✓	-	-	✓	-	-	6
-	✓	✓	✓	✓	✓	-	-	7
-	✓	-	✓	-	✓	-	-	8

Table B.9: Configuration of the model in each ablation study. Check marks indicate which elements are present in each ablation study.

	Ablation Study 1					
Video	CC		SIM		KLDiv	
003	0.4345	0.1723	0.2384	0.0705	8.7390	1.5158
005	0.2107	0.0610	0.1543	0.0267	11.9484	0.7031
013	0.3296	0.1537	0.1776	0.0651	11.5945	1.5386
015	0.4554	0.1673	0.2114	0.0608	10.8022	1.4834
018	0.6483	0.0946	0.3526	0.0710	7.2721	1.8653
020	0.3839	0.2134	0.2155	0.0913	10.2025	2.4971
022	0.2319	0.1211	0.1878	0.0610	11.0768	1.7895
027	0.3721	0.1855	0.1691	0.0666	11.8268	1.5465
028	0.3639	0.1904	0.1576	0.0521	12.1867	1.2288
029	0.3422	0.1512	0.1805	0.0510	11.7253	1.0877
033	0.5481	0.2242	0.2418	0.0891	10.4648	1.8409
040	0.3706	0.1520	0.2078	0.0590	10.6692	1.5706
043	0.3943	0.1461	0.1883	0.0526	10.9798	1.4100
047	0.6648	0.0822	0.3595	0.0668	7.5687	1.5664
052	0.5289	0.1614	0.2858	0.0688	9.0579	1.4576
053	0.2460	0.1203	0.1629	0.0486	11.9058	1.2887
059	0.4285	0.1383	0.1990	0.0461	10.9906	1.1944
066	0.4670	0.2092	0.2127	0.0684	10.4878	1.7887
067	0.3610	0.1629	0.2036	0.0669	11.0499	1.6113
072	0.5768	0.2413	0.2355	0.1042	10.4950	2.1040
075	0.2247	0.1518	0.1705	0.0766	11.3049	2.2822
080	0.4058	0.2045	0.2246	0.0626	10.0976	1.7362
087	0.1511	0.0633	0.1510	0.0505	11.4328	1.8603
094	0.2997	0.1849	0.1853	0.0845	11.7099	2.0353
103	0.4323	0.1927	0.2337	0.0894	9.6814	2.6591
110	0.2582	0.1379	0.2015	0.0800	10.9673	2.2877
159	0.3379	0.1570	0.1580	0.0348	11.8857	0.9080
160	0.2655	0.1359	0.2042	0.0660	10.5136	2.0911
170	0.4300	0.1826	0.2360	0.0822	9.7508	1.7099
184	0.6743	0.1519	0.2826	0.0716	9.6040	1.4469
197	0.3326	0.1446	0.1909	0.0506	10.6856	1.4329
213	0.2980	0.1366	0.2332	0.0647	9.1352	2.0560

Table B.10: Mean and standard deviation of the CC, SIM, and KLDiv scores for each video in the VR-EyeTracking dataset in the ablation study number 1.

	Ablation Study 2					
Video	CC		SIM		KLDiv	
003	0.4285	0.1675	0.2386	0.0734	8.9560	1.6844
005	0.1773	0.0798	0.1410	0.0335	11.7037	1.0806
013	0.4063	0.1675	0.2162	0.0702	10.3207	1.6347
015	0.4183	0.1665	0.2213	0.0743	9.9883	1.7530
018	0.6395	0.1039	0.3571	0.0592	6.9805	1.1273
020	0.2923	0.1490	0.1642	0.0531	11.3344	1.4902
022	0.3316	0.1290	0.2033	0.0545	10.2527	1.2515
027	0.3956	0.2008	0.1812	0.0848	11.0605	1.7852
028	0.4142	0.1675	0.1851	0.0614	11.1041	1.3615
029	0.2787	0.1080	0.1630	0.0441	11.5675	1.0321
033	0.5941	0.2028	0.2633	0.0724	9.6076	1.4615
040	0.3211	0.1250	0.1689	0.0482	11.3080	1.2077
043	0.4011	0.1559	0.1899	0.0441	10.5007	1.1457
047	0.6503	0.0802	0.3131	0.0587	8.4859	1.2580
052	0.5075	0.1535	0.2248	0.0579	10.2931	1.0854
053	0.3483	0.1084	0.1929	0.0493	10.7719	1.1996
059	0.5506	0.1533	0.2393	0.0612	9.8382	1.2028
066	0.4175	0.1301	0.1780	0.0482	11.1533	1.0962
067	0.3557	0.1544	0.1869	0.0591	10.9474	1.3800
072	0.5171	0.2392	0.2735	0.1186	8.7108	2.3296
075	0.2392	0.1954	0.1730	0.0854	11.1876	2.2367
080	0.4335	0.2240	0.2514	0.0869	9.0619	1.8688
087	0.1805	0.0753	0.1672	0.0571	10.7468	1.9489
094	0.2633	0.1837	0.1909	0.0886	10.9085	2.2807
103	0.5027	0.2264	0.2618	0.0977	9.1019	1.7980
110	0.2573	0.1197	0.2020	0.0695	10.4149	1.9247
159	0.3843	0.1817	0.1854	0.0558	10.6763	1.4442
160	0.2976	0.0835	0.2467	0.0388	8.7627	1.5503
170	0.4291	0.1691	0.2456	0.0603	8.8430	1.1250
184	0.6546	0.2063	0.3143	0.0818	8.6354	1.6064
197	0.3664	0.1541	0.2028	0.0567	9.8454	1.5238
213	0.3165	0.1641	0.2500	0.0857	8.1738	2.4840

Table B.11: Mean and standard deviation of the CC, SIM, and KLDiv scores for each video in the VR-EyeTracking dataset in the ablation study number 2.

	Ablation Study 3					
Video	CC		SIM		KLDiv	
003	0.2484	0.1385	0.1262	0.0410	11.9548	1.2006
005	0.1699	0.0901	0.1245	0.0316	11.6964	0.9213
013	0.2910	0.0950	0.1447	0.0497	11.7224	1.1636
015	0.2201	0.1279	0.0997	0.0345	12.7946	0.8975
018	0.5329	0.1181	0.2760	0.0581	7.8631	1.6325
020	0.1780	0.0684	0.1198	0.0247	12.2093	0.5850
022	0.2528	0.1270	0.1677	0.0494	10.9144	1.2693
027	0.3786	0.2558	0.1957	0.1207	10.4894	2.8161
028	0.2962	0.1690	0.1583	0.0698	11.8308	1.6890
029	0.3289	0.1284	0.2019	0.0585	10.3989	1.5041
033	0.1854	0.1066	0.1433	0.0556	12.2970	1.4806
040	0.3380	0.1397	0.1877	0.0620	10.8252	1.4316
043	0.2917	0.1363	0.1508	0.0479	11.5510	1.0634
047	0.6096	0.0792	0.3321	0.0437	7.6796	1.0443
052	0.4826	0.1700	0.2617	0.0581	9.2402	1.2089
053	0.3481	0.1526	0.1978	0.0601	10.8517	1.4108
059	0.4544	0.0936	0.2212	0.0399	10.3810	0.9272
066	0.4121	0.1447	0.2174	0.0633	10.5390	1.4895
067	0.3295	0.1681	0.2065	0.0903	10.6979	2.1780
072	0.4509	0.2355	0.2052	0.0998	10.5637	2.1744
075	0.1960	0.1324	0.1481	0.0630	11.4837	2.3462
080	0.3537	0.2030	0.2032	0.0770	10.5342	1.6529
087	0.1567	0.0510	0.1627	0.0311	11.0696	1.1405
094	0.1728	0.0834	0.1467	0.0532	11.7948	1.4862
103	0.4407	0.2407	0.2727	0.1289	8.4647	2.7913
110	0.2098	0.1115	0.1723	0.0659	11.2744	1.7242
159	0.3285	0.1670	0.1861	0.0706	10.6238	1.8364
160	0.2526	0.1088	0.2049	0.0367	9.6807	1.1306
170	0.3466	0.1345	0.1809	0.0325	11.3185	0.6867
184	0.4651	0.1523	0.2425	0.0652	9.6285	1.4836
197	0.1612	0.0734	0.1138	0.0235	12.6480	0.5504
213	0.3765	0.1706	0.2862	0.0939	7.6619	2.3086

Table B.12: Mean and standard deviation of the CC, SIM, and KLDiv scores for each video in the VR-EyeTracking dataset in the ablation study number 3.

	Ablation Study 4					
Video	CC		SIM		KLDiv	
003	0.3802	0.1258	0.2013	0.0506	10.4207	1.1657
005	0.1298	0.0592	0.1271	0.0286	12.0698	0.9042
013	0.2936	0.1218	0.1702	0.0574	11.6152	1.2330
015	0.3562	0.1576	0.1838	0.0523	10.9646	1.2277
018	0.5280	0.0762	0.2674	0.0337	9.1603	0.6526
020	0.2394	0.1305	0.1506	0.0440	11.9751	1.0284
022	0.3322	0.1230	0.2151	0.0546	9.8111	1.2816
027	0.3945	0.2097	0.1890	0.0915	10.8932	1.9283
028	0.3755	0.1804	0.1846	0.0641	11.1645	1.4119
029	0.2690	0.1165	0.1548	0.0409	11.8809	0.8940
033	0.5444	0.1562	0.2412	0.0523	10.1560	1.0298
040	0.3150	0.1433	0.1714	0.0539	11.3927	1.2051
043	0.4185	0.1522	0.2266	0.0570	9.7535	1.2470
047	0.5635	0.0639	0.2683	0.0326	9.3003	0.7096
052	0.4743	0.1725	0.2596	0.0629	9.2690	1.3842
053	0.3556	0.1099	0.1903	0.0503	10.9761	1.1185
059	0.5020	0.1074	0.2314	0.0453	10.0961	1.0043
066	0.4589	0.0925	0.2206	0.0506	10.2871	1.1163
067	0.3632	0.1522	0.2172	0.0732	10.3639	1.8059
072	0.5099	0.2240	0.2691	0.1043	8.8827	2.0103
075	0.2284	0.1530	0.1669	0.0749	11.3623	1.9633
080	0.3994	0.2079	0.2191	0.0671	10.0131	1.4621
087	0.1378	0.0639	0.1481	0.0427	11.1923	1.6034
094	0.2716	0.1568	0.1937	0.0830	10.9199	2.0869
103	0.3877	0.1847	0.2366	0.0944	8.9986	2.3391
110	0.2646	0.1432	0.2037	0.0822	10.4741	2.0718
159	0.4435	0.1846	0.2377	0.0706	9.0815	1.4658
160	0.2539	0.0820	0.2156	0.0286	9.4930	1.0105
170	0.4428	0.1634	0.2420	0.0574	9.4184	1.0237
184	0.5697	0.1391	0.2295	0.0464	10.3193	0.9270
197	0.2717	0.0870	0.1426	0.0257	11.7785	0.7229
213	0.3573	0.1541	0.2522	0.0721	8.9015	1.8902

Table B.13: Mean and standard deviation of the CC, SIM, and KLDiv scores for each video in the VR-EyeTracking dataset in the ablation study number 4.

	Ablation Study 5					
Video	CC		SIM		KLDiv	
003	0.2855	0.1239	0.1609	0.0467	11.2830	1.1980
005	0.1803	0.0845	0.1353	0.0317	11.8439	0.8929
013	0.3414	0.1108	0.2095	0.0452	10.5140	1.1160
015	0.3161	0.1179	0.1596	0.0378	11.4911	0.8864
018	0.4742	0.0849	0.2771	0.0445	8.6932	1.0318
020	0.2728	0.1341	0.1457	0.0425	11.7907	1.0279
022	0.4027	0.1186	0.2134	0.0507	10.4394	1.1282
027	0.4331	0.2110	0.1968	0.0898	10.7921	1.9002
028	0.3635	0.1499	0.1787	0.0537	11.3793	1.1393
029	0.3492	0.1217	0.1792	0.0452	11.2649	1.0370
033	0.5492	0.1400	0.2351	0.0508	10.2041	1.0090
040	0.3555	0.1275	0.1881	0.0519	11.0780	1.2058
043	0.4061	0.1488	0.2053	0.0487	10.3180	1.0620
047	0.5354	0.0912	0.2887	0.0404	8.6285	1.0290
052	0.3965	0.1680	0.2002	0.0447	10.7919	0.9916
053	0.3669	0.1069	0.2022	0.0434	10.7280	0.9799
059	0.4905	0.0974	0.2343	0.0452	10.0094	0.9788
066	0.4615	0.0671	0.2169	0.0359	10.3829	0.9179
067	0.3638	0.1514	0.2139	0.0733	10.5349	1.7203
072	0.5661	0.1997	0.2632	0.1140	8.9785	2.0575
075	0.2434	0.1632	0.1707	0.0717	11.3438	1.7952
080	0.3927	0.1263	0.2085	0.0418	10.3927	0.9318
087	0.1744	0.0682	0.1677	0.0434	10.9531	1.5178
094	0.2470	0.1307	0.1923	0.0731	11.1149	1.8085
103	0.4166	0.2196	0.2403	0.1020	9.2181	2.5530
110	0.3017	0.1402	0.2107	0.0749	10.2810	1.8784
159	0.4522	0.1694	0.2405	0.0585	9.3252	1.2296
160	0.2717	0.0776	0.2178	0.0407	9.9024	1.1058
170	0.4770	0.0776	0.2042	0.0291	10.7002	0.6625
184	0.4473	0.0653	0.2039	0.0349	10.6666	0.8056
197	0.3208	0.0730	0.1562	0.0285	11.7218	0.6605
213	0.4443	0.1049	0.2462	0.0378	9.5776	0.8879

Table B.14: Mean and standard deviation of the CC, SIM, and KLDiv scores for each video in the VR-EyeTracking dataset in the ablation study number 5.

	Ablation Study 6					
Video	CC		SIM		KLDiv	
003	0.3609	0.1437	0.1600	0.0454	11.4296	1.1093
005	0.2081	0.0684	0.1459	0.0295	11.6792	0.7737
013	0.3094	0.1292	0.1571	0.0537	11.6391	1.2012
015	0.2800	0.1400	0.1550	0.0464	11.5143	1.0556
018	0.4625	0.0961	0.2213	0.0355	10.1580	0.8602
020	0.2359	0.1159	0.1407	0.0341	12.2300	0.7875
022	0.3451	0.0980	0.2117	0.0499	10.5636	1.1901
027	0.3506	0.1860	0.1651	0.0740	11.3954	1.6802
028	0.2951	0.1694	0.1701	0.0738	11.6266	1.7515
029	0.2765	0.1048	0.1619	0.0473	11.6622	1.0901
033	0.5459	0.1835	0.2431	0.0792	9.9187	1.6757
040	0.3019	0.1105	0.1870	0.0579	11.0425	1.4277
043	0.4049	0.1181	0.1761	0.0394	11.0906	0.9320
047	0.4765	0.1078	0.2371	0.0376	9.8643	0.8818
052	0.3928	0.0924	0.1677	0.0408	11.2999	1.0086
053	0.3968	0.0952	0.2064	0.0391	10.7021	0.8958
059	0.4299	0.0849	0.1890	0.0364	10.9120	0.8065
066	0.4481	0.1514	0.2045	0.0645	10.7124	1.3771
067	0.4092	0.1097	0.2172	0.0561	10.4411	1.3329
072	0.4519	0.1952	0.2134	0.0835	10.2876	1.9104
075	0.2168	0.1343	0.1571	0.0660	11.7905	1.8888
080	0.4360	0.1990	0.2301	0.0615	9.9019	1.3773
087	0.1986	0.0662	0.1847	0.0467	10.6663	1.4756
094	0.2592	0.1091	0.1650	0.0539	11.6309	1.3775
103	0.4671	0.2275	0.2714	0.1111	8.9606	2.5339
110	0.2924	0.1062	0.2131	0.0661	10.5680	1.8107
159	0.2867	0.1175	0.1696	0.0480	11.3085	1.2725
160	0.3218	0.0866	0.2367	0.0396	9.2186	1.0648
170	0.4382	0.1549	0.2284	0.0592	9.9713	1.0773
184	0.3398	0.1505	0.1897	0.0665	10.7825	1.5695
197	0.2628	0.1025	0.1364	0.0338	12.1772	0.8308
213	0.4272	0.1144	0.2978	0.0552	7.7520	1.3903

Table B.15: Mean and standard deviation of the CC, SIM, and KLDiv scores for each video in the VR-EyeTracking dataset in the ablation study number 6.

	Ablation Study 7					
Video	CC		SIM		KLDiv	
003	0.3148	0.1609	0.1649	0.0637	11.3670	1.4918
005	0.2052	0.0736	0.1460	0.0295	11.7209	0.8047
013	0.3508	0.1377	0.2054	0.0543	10.5253	1.3365
015	0.2902	0.1566	0.1871	0.0635	10.5702	1.8053
018	0.5816	0.1043	0.3148	0.0481	7.9540	1.0664
020	0.1664	0.0659	0.1192	0.0248	12.5475	0.6634
022	0.3491	0.1111	0.2289	0.0569	9.7304	1.4016
027	0.4249	0.2324	0.1866	0.0820	11.0072	1.7460
028	0.1180	0.1055	0.0843	0.0468	13.8292	1.3668
029	0.3823	0.1174	0.1918	0.0469	10.8834	1.0910
033	0.5085	0.2063	0.2328	0.0743	10.3115	1.4127
040	0.3994	0.1427	0.2183	0.0593	10.1242	1.5422
043	0.3221	0.1305	0.1784	0.0446	11.0343	1.1572
047	0.6305	0.0871	0.3389	0.0442	7.5890	0.9926
052	0.4969	0.1899	0.2513	0.0682	9.2826	1.4538
053	0.3682	0.1354	0.2266	0.0552	10.0281	1.2670
059	0.5252	0.1272	0.2688	0.0534	9.0940	1.0925
066	0.5099	0.1431	0.2323	0.0698	10.1259	1.5276
067	0.3196	0.1472	0.1953	0.0521	10.7467	1.2676
072	0.5545	0.2060	0.2233	0.0981	10.1431	2.1525
075	0.2163	0.1379	0.1495	0.0589	11.6772	1.6357
080	0.4118	0.2042	0.2386	0.0649	9.7733	1.3414
087	0.1673	0.0720	0.1483	0.0384	11.6076	1.1192
094	0.0742	0.0665	0.0822	0.0465	13.6030	1.2505
103	0.4085	0.1931	0.2249	0.0851	9.5288	2.0606
110	0.2370	0.1273	0.1641	0.0663	11.5711	1.7172
159	0.3946	0.1804	0.1785	0.0455	11.0496	1.1972
160	0.2418	0.0836	0.2080	0.0368	10.0331	1.2192
170	0.3935	0.1622	0.2154	0.0602	9.9397	1.2362
184	0.6526	0.1049	0.3027	0.0497	8.7139	1.0583
197	0.3951	0.1316	0.2154	0.0445	9.9223	0.9877
213	0.3905	0.1540	0.2775	0.0760	8.1864	1.8685

Table B.16: Mean and standard deviation of the CC, SIM, and KLDiv scores for each video in the VR-EyeTracking dataset in the ablation study number 7.

	Ablation Study 8					
Video	CC		SIM		KLDiv	
003	0.2325	0.1317	0.1008	0.0264	12.4718	0.6044
005	0.1507	0.0848	0.1190	0.0336	12.0119	0.9630
013	0.2662	0.1121	0.1431	0.0445	11.8636	1.0386
015	0.2510	0.1345	0.0987	0.0282	12.6502	0.6469
018	0.4050	0.1403	0.1912	0.0481	10.6238	1.0591
020	0.1674	0.0712	0.0928	0.0155	12.6865	0.3707
022	0.2174	0.1146	0.0922	0.0302	12.8203	0.6872
027	0.3126	0.2422	0.0928	0.0551	12.7813	1.2312
028	0.2973	0.1681	0.0903	0.0325	12.8364	0.7611
029	0.3313	0.1154	0.1029	0.0236	12.4537	0.4859
033	0.3819	0.1824	0.1022	0.0413	12.4955	0.9251
040	0.2884	0.1215	0.1115	0.0327	12.4311	0.7387
043	0.3372	0.1445	0.0976	0.0335	12.5155	0.7454
047	0.4215	0.1542	0.1787	0.0517	10.8888	1.1722
052	0.3309	0.1925	0.1348	0.0337	11.8684	0.7797
053	0.1579	0.0862	0.1107	0.0343	12.6356	0.8666
059	0.2772	0.1450	0.1011	0.0316	12.5691	0.7206
066	0.3029	0.1411	0.1042	0.0298	12.4939	0.6881
067	0.2473	0.1326	0.1253	0.0481	12.2707	1.1479
072	0.3350	0.1860	0.0718	0.0274	13.1715	0.6752
075	0.1866	0.1467	0.1090	0.0432	12.5517	1.1030
080	0.2333	0.1229	0.1099	0.0328	12.3549	0.7751
087	0.1736	0.0484	0.1202	0.0262	11.9921	0.6382
094	0.1719	0.1232	0.0875	0.0352	13.0676	0.9046
103	0.3554	0.1995	0.1108	0.0520	12.3958	1.1460
110	0.3523	0.0957	0.1837	0.0338	10.7752	0.6859
159	0.3060	0.1385	0.1204	0.0299	12.0666	0.6838
160	0.2717	0.0727	0.1480	0.0340	11.4493	0.7770
170	0.3474	0.1373	0.1171	0.0246	12.1955	0.5196
184	0.2639	0.1236	0.0888	0.0238	12.8122	0.5807
197	0.2177	0.0643	0.0841	0.0163	12.8404	0.3746
213	0.4242	0.1440	0.1538	0.0323	11.3888	0.6460

Table B.17: Mean and standard deviation of the CC, SIM, and KLDiv scores for each video in the VR-EyeTracking dataset in the ablation study number 8.