University of Brasilia, Brazil

Faculdade de Tecnologia

Deep Hybrid Model for Region of Interest Detection in Omnidirectional Videos

Relatório Científico Primary do projeto na modalidade Auxílio à Pesquisa Regular, fomentado pela Fundação de Amparo à Pesquisa do Estado de São Paulo.

Projeto FAPESP #2021/05972-7

Pesquisador Responsável: Sana Alamgeer

Brasilia, 17 de dezembro de 2024

1 Abstract

The main goal of the project is to design a new model that predicts regions of interest in 360° videos. The region of interest (ROI) plays an important role in 360° video streaming. For example, ROIs are used to predict view-ports, intelligently cut the videos for live streaming, etc so that less bandwidth is used. Detecting view-ports in advance helps reduce the movement of the head while streaming and watching a video via the head-mounted device. Whereas, intelligent cuts of the videos help improve the efficiency of streaming the video to users and enhance the quality of their viewing experience. This report illustrates the secondary task to identify ROIs, in which, we design, train, and test a hybrid saliency model. In this work, we refer to saliency regions to represent the regions of interest. The method includes the processes as follows: preprocessing the video to obtain frames, developing a hybrid saliency model for predicting the region of interest, and finally post-processing the output predictions of the hybrid saliency model to obtain the output region of interest for each frame. Then, we compare the performance of the proposed method with the subjective annotations of the 360RAT dataset. The proposed method including the experimental setup and results is described in section 2.

2 Methodology

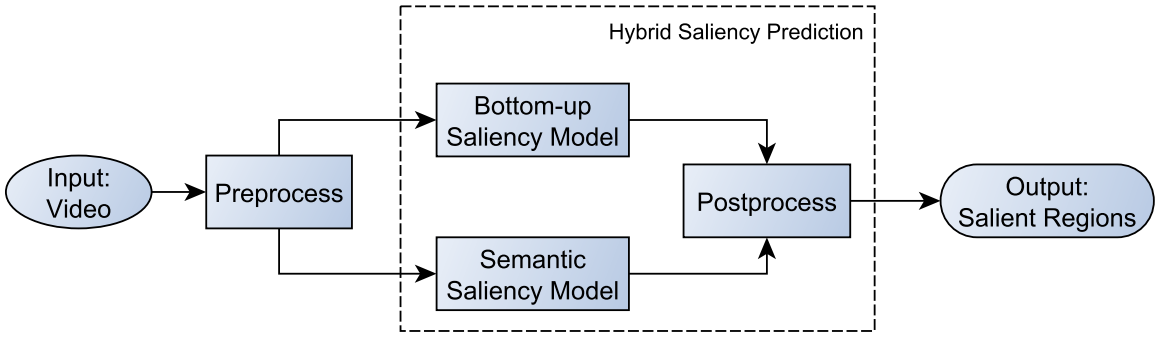

Figura 1: Flowchart of the proposed method of predicting region of interest in 360° videos.

In this section, we discuss the proposed methodology as shown in Figure 1. The method includes the processes as follows: preprocessing the video to obtain frames, developing a hybrid saliency model for predicting the region of interest, and finally post-processing the output predictions of the hybrid saliency model to obtain the output region of interest for each frame.

2.1 Input Preparation

To prepare the input, we parsed a .mp4 video into frames. Then, for every frame, we generated two types of inputs A and B. Input A is a segmented frame, i.e., we use a

threshold T to penalize bright pixels (regions of sky or light bulbs) and replace them with black pixels. The algorithm1 shows a set of instructions to generate input A for every frame. In this work, we have used value 9 for the T threshold.

Algorithm 1 An algorithm to generate input A.

Input: Let I be an input image.

1: Let T be the threshold value.
2: $IG \leftarrow gaussian(I)$ with $\sigma = 1$
3: $mask = IG < T$
4: Let $\tilde{a}$ be the BITWISE NOT of $mask$
5: $I_T \leftarrow I\,[\tilde{a}] = 0$

Output: I_T

To obtain input B, we computed optical flow maps. More specifically, we used the optical flow algorithm (to generate motion vector maps) implemented by Farneback [1], which performs well even when there are luminance changes and the scene has a lot of edges [2]. For illustration, Figure 2(a) shows sample frame of 360° video, Figure 2(b) shows the I_T image of frame, and Figure 2(c) shows the optical flow map of frame.

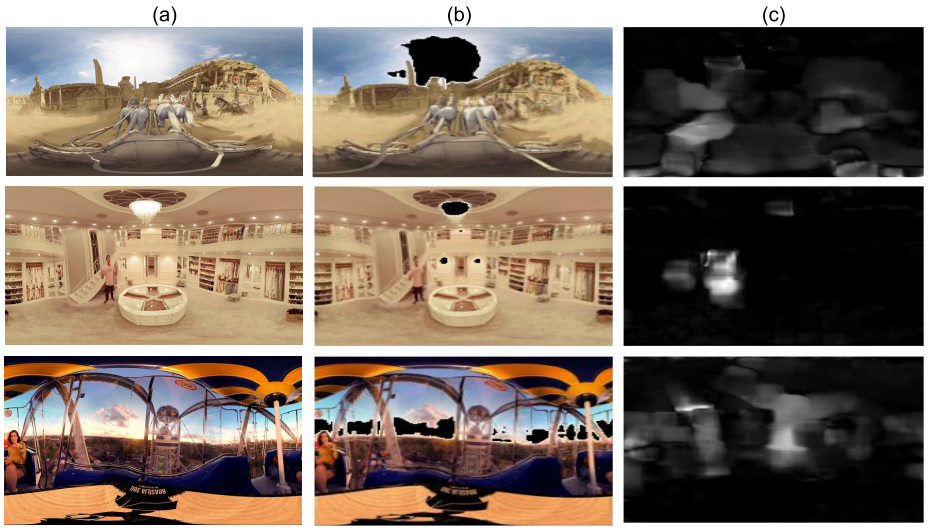

Figura 2: Illustration of inputs A and B for a random frame of a 360° video. (a) Reference frame, (b) I_T image of (a), and (c) Optical flow map of (a).

2.2 Bottom-up Saliency Model

Figure 3 shows the block diagram of the proposed bottom-up saliency model. Notice that the model has two streams, each composed of an independent architecture composed of a Convolutional Neural Network (CNN) and an Atrous Convolutional layer (ACL). The

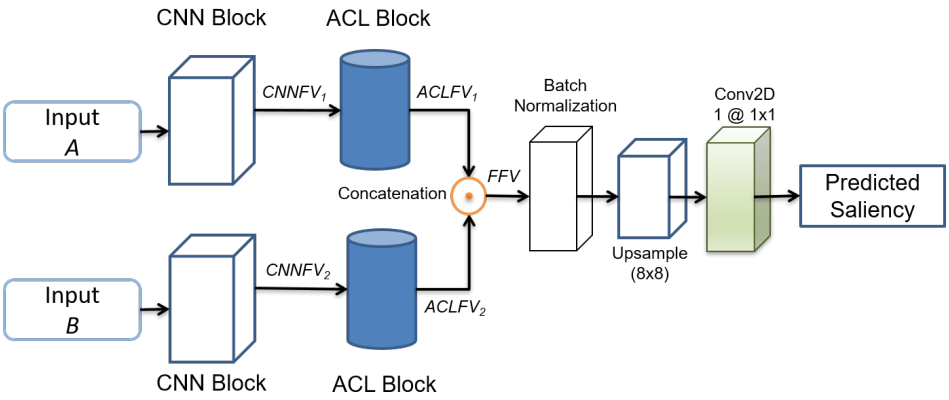

Figura 3: Block diagram of the proposed bottom-up saliency model.

processing streams in the architecture take inputs A and B individually for processing. Both streams first extract primary features of inputs using CNNs, and then extract high-level features using the block of ACLs. More specifically, input A is fed to the 1st stream, while input B is fed to the 2nd stream. Finally, the output from both streams is concatenated and fed into the regression block that generates the predicted saliency map. Next, we describe each of the stages in Figure 3.

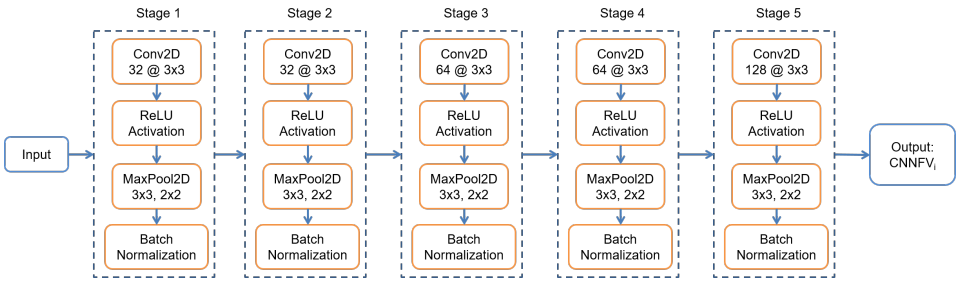

Figura 4: Illustration of CNN Block in the proposed bottom-up saliency model.

Figure 4 shows the CNN block, which contains 5 stages that extract primary features from an input RGB image. All stages have a similar set of layers, with the first layer being a 2D convolution layer with a different number of output filters and 3×3 kernels, the second layer a ReLU activation layer, the third layer a 2D max pooling layer with a 3×3 pool and a 2×2 stride, and the fourth layer a Batch Normalization layer. More specifically, stages 1 and 2 generate 32 output filters, stages 3 and 4 generate 64 output filters, and stage 5 generates 128 output filters. The output of CNN block in every processing stream is the CNN feature vector - CNNFV$_i$, in which i represents the corresponding stream.

The Atrous Convolution layer (ACL) can be used to effectively enlarge the receptive field of the network which in turn captures dense features, without increasing the number of parameters. More specifically, ACL is a dilated convolution where zeros are added between the weights of the convolution kernel. In the simpler case of a 1D Atrous

convolution, the output of the signal is defined as follows [3]:

$$y[i] = \sum_{k=1}^{K} x[i + rK] \cdot \omega[k] \tag{1}$$

where r is the dilation rate (or Atrous rate), $\omega[k]$ is the filter of length K, $x[i]$ is the input, and $y[i]$ is the output of a pixel.

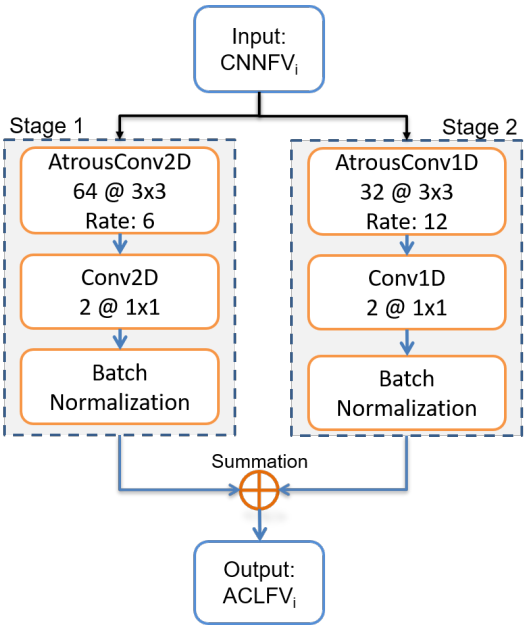

Figura 5: Illustration of ACL Block in the proposed bottom-up saliency model.

Figure 5 illustrates the architecture of the proposed ACL block consisting of 2 stages. Both stages have a similar set of layers, with the first layer a 2D Atrous Convolution layer with different output filters and Atrous rates with 3×3 kernels, the second layer a 2D convolution layer with 2 output filters and 1×1 kernels, and the third layer a Batch Normalization layer. More specifically, in stage 1, 2D ACL has an Atrous rate of 6 and 64 output filters, while in stage 2, 2D ACL has an Atrous rate of 12 and 32 output filters. Both ACL and CNN layers are activated by the ReLU activation function. From the ACL block, we obtain an ACL feature vector by the summation of the outputs y_i from each i-th stage, as follows:

$$ACLFV_i = y_1 + y_2, \tag{2}$$

where the i index corresponds to the stream ($i = 1, 2$) of the proposed method (see Figure 3).

As shown in Figure 3, the outputs of both streams are concatenated as follows:

$$FFV = ACLFV_1 \oplus ACLFV_2, \tag{3}$$

where FFV represents the concatenated feature vector, and '$\oplus$' represents the concatenation operation. The concatenated feature vector is fed into the regression block, which is composed of one Batch Normalization layer, one 2D Up-Sampling layer of size 8×8, and one 2D convolution layer with 1 output filter and a 1×1 kernel activated. The 2D Up-Sampling sampling uses Bilinear interpolation to increase the resolution of output filters. the final 2D convolution layer is activated by the Sigmoid activation function, and the output of this layer is a 2D activation map that represents the predicted saliency information.

2.3 Semantic Saliency Model

To generate semantic saliency information, we have used the Multi-projection variant of the YOLOv3 (MPYOLO) model [4]. The model has incorporated a multi-projection approach with a wide Field of View (FOV), and soft selection as an alternative to the standard non-maximum suppression (NMS) to select the detections produced by multiple windows of projections.

Figura 6: Sample outputs generated by MPYOLO [4] model. Every pair of images shows the input frame (left) and the frame with predicted bounding boxes (right).

Specifically, in the first step of MPYOLO, four stereographic projections are generated for which their horizontal and vertical spans are 180°. The horizontal overlap between the two neighbor projections is 90°. Then each sub-projection is separately processed by the YOLOv3 object detector. In the post-processing part, the bounding box adjustment part fixes distorted detection boxes, and soft-selection re-arranges the scores of redundant boxes in overlapping areas. Figure 6 shows examples of outputs generated by the MPYOLO model. These outputs are later used as semantic saliency maps in the proposed hybrid saliency model.

2.4 Hybrid Saliency Model

Once we obtain predictions from bottom-up and semantic saliency models, in the next step, we perform post-processing on these predictions. In this work, from now on, we refer to the predictions as bounding boxes. Let X be a set of bounding boxes $\{1, 2, 3, ..., x\}$

predicted by the bottom-up saliency map, Y be a set of bounding boxes $\{1, 2, 3, ..., y\}$ predicted by the semantic saliency model, and H be a set of final bounding boxes $\{1, 2, 3, ..., h\}$based on X and Y sets. Then, we perform post-processing by following the steps as described in the algorithm 2.

Algorithm 2 An algorithm to generate a set H of final bounding boxes as the output of hybrid saliency model.

Input: X containing the bounding boxes $\{1, 2, 3, ..., x\}$.
Input: Y containing the bounding boxes $\{1, 2, 3, ..., y\}$.
Input: Let $d()$ be the function to compute Euclidean Distance.
Input: Let t be the minimum threshold for euclidean distance.
Input: Let H be an empty set.

while $x \neq 0$ **do**
    $h \leftarrow 0$
    while $y \neq 0$ **do**
        $\|p\| = d(x, y)$
        if $\|p\| < t$ **then**
            $h \leftarrow x, y$
        end if
        if size of h is > 1 **then**
            while $h \neq 0$ **do**
                $u \leftarrow$ union-intersection based bounding boxes of h.
                $H \leftarrow u$
            end while
        end if
    end while
end while
if size of H is < 1 **then** ▷ Setting priority when the final set of bounding boxes if found empty.
    $H \leftarrow y$
else
    $H \leftarrow x$
end if

Output: H

To further elaborate on the steps involved in algorithms 2, Figure 7 shows examples of output bounding boxes. In this figure, the bottom-up saliency predictions represent the bounding boxes obtained from the bottom-up saliency model, the semantic saliency predictions represent the bounding boxes obtained from the semantic saliency model. The third column represents the bounding boxes obtained after merging the predictions of the first and second columns. This merging is performed using only the euclidean distance. The fourth column represents the bounding boxes that are obtained after performing union/intersection on the predictions of the third column.

It is worth pointing out that, if we obtain an empty set H with no bounding boxes, then we set priorities, i.e., we assign the Y set of bounding boxes to H, if Y is not empty.

Otherwise, we give priority to the X set and assign the bounding boxes to H. These priorities are set in steps 16 to 20 of algorithm 2.

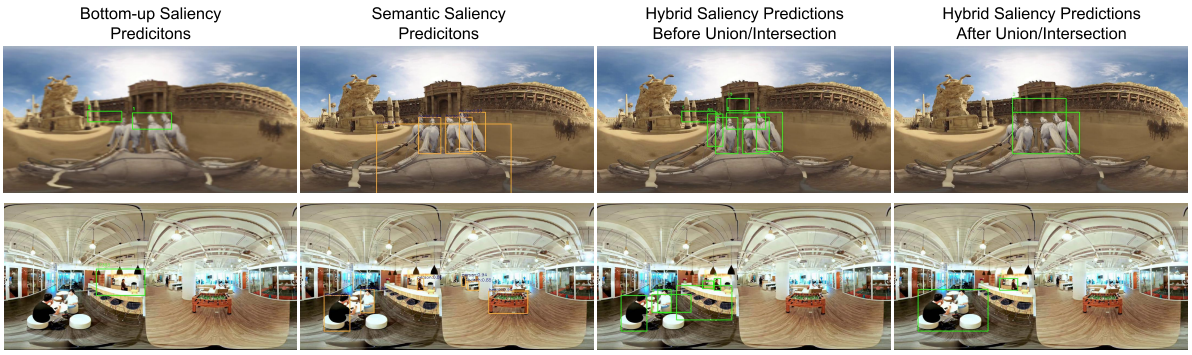

Figura 7: Examples of output bounding boxes obtained from different steps elaborated in the algorithm 2.

3 Experimental Setup

For training and testing the bottom-up saliency model, we used 360RAT [5] dataset of 360° videos. The dataset has eleven 360° videos, and each video has 9 subjective annotations. We used this dataset because of the diversity and popularity of the visual content. To prepare the input, we averaged 9 annotations for every corresponding frame. We also performed augmentation on input samples, and the algorithms include a Gaussian filter, horizontal and vertical flips, and rotations (90° clockwise, 180°, and 270°). It is worth mentioning that we also downsampled the spatial resolution of the frame to the size of 200×200 using the Image.CUBIC() function of Pillow library in Python. For performance evaluation, we used 3 metrics: Area Under Curve by Judd (AUC_Judd), Cross-Correlation (CC), and Similarity (SIM). Metric values closer to '1' indicate a closer agreement of the maps.

To train and test the bottom-up saliency model, we randomly divided the 360RAT dataset into three content-independent training, validation, and test subsets. Specifically, 80% of the dataset is randomly selected for training, 10% for validation, and the remaining 10% is used for testing. We report the AUC_Judd, CC, and SIM values only for the test subset. We trained the model using 1 Batch, 10 epochs, and Mean Square Error (MSE) as the training loss. In addition, we used the Stochastic Gradient Descent (SGD) optimizer [6] to minimize the loss function with a learning rate of 0.001. We implemented the proposed method using Keras [7] library of Python, and trained and tested on 25GB GPU in a LINUX environment.

Tabela 1: Statistical comparison between the proposed Bottom-up saliency model and state-of-the-art saliency models for 360° videos. The results are the values obtained by evaluation metrics AUC_Judd, CC, and SIM on the 360RAT dataset.

Saliency Model	Evaluation		
	AUC_Judd	CC	SIM
CubePadding360	0.5434	0.0910	0.1529
Panoramic360	0.5657	0.1284	0.1751
Proposed	**0.8290**	**0.5438**	**0.4450**

4 Results and Discussion

In the analysis of the results, first, we compared the performance of the proposed bottom-up saliency model with the state-of-the-art saliency models of 360° videos. We chose two saliency models CubePadding360 [8] and Panoramic360 [9]. These salience models are based on convolution neural network (CNN) architectures. Table 1 describes a statistical comparison in which the bold values represent the highest performance. In this table, we can see that the prediction performance of the proposed bottom-up saliency model has strong correlations with subjective annotations. Whereas, the performance of CubePadding360 and Panoramic360 models show low values for low AUC_Judd, CC, and SIM metrics. Figure 8 represents the performance of the proposed bottom-up saliency model during training and validation steps for 10 epochs. From this figure, we analyze that the model converges well from epoch 1 and keeps decreasing the training and validation losses up to the last epoch.

Figura 8: Training vs validation loss of the proposed bottom-up saliency model.

Next, we performed a comparative analysis among bottom-up, semantic and hybrid saliency models. Table 2 show statistical comparison with respect to AUC_Judd, CC, and

Tabela 2: Statistical comparison between the proposed Bottom-up, Semantic, and Hybrid saliency models and state-of-the-art saliency models for 360° videos. The results are the values obtained by evaluation metrics AUC_Judd, CC, and SIM on the 360RAT dataset.

Type	Saliency Models	Evaluation		
		AUC_Judd	CC	SIM
State-of-the-art	CubePadding360	0.5434	0.0910	0.1529
	Panoramic360	0.5657	0.1284	0.1751
Proposed	Bottom-up	**0.8290**	**0.5438**	**0.4450**
	Semantic	0.6875	0.2179	0.0330
	Hybrid	0.7990	0.4411	0.3918

Tabela 3: Results of ablation test in which we tested the performance of bottom-up saliency model using different values of threshold T. The results are the values obtained by evaluation metrics AUC_Judd, CC, and SIM on the 360RAT dataset.

T	Evaluation		
	AUC_Judd	CC	SIM
No Threshold	0.7324	0.4335	0.3289
0.5	0.7833	0.4559	0.4081
0.6	0.7990	0.4284	0.3755
0.7	0.7684	0.4342	0.3903
0.8	0.8038	0.4877	0.4064
0.9 (Proposed)	**0.8290**	**0.5438**	**0.4450**

SIM on the 360RAT dataset in which the bold values represent the highest performance. In this table, we see that the hybrid saliency model has shown slightly better performance in comparison with the semantic saliency model. But, the overall performance of both semantic and hybrid models is better than the state-of-the-art saliency models. The proposed bottom-up saliency model has shown the best performance in terms of predicting regions of saliency or regions of interest (ROI) that are in agreement with the subjective annotations of the 360RAT dataset.

For the ablation test, we performed two types of tests. In the first test, we evaluated the performance of the bottom-up saliency model by changing the threshold t value. More specifically, we tested 0.5, 0.6, 0.7, 0.8, and 0.9 values for T. We also tested the values between 0.1 and 0.4, but the network did not converge using these values, and therefore, we did not include these values in the analysis of the results. Table 3 shows the results obtained for this ablation test. In this table, we see that the best performance of the proposed bottom-up saliency model is obtained when the threshold value 0.9 was used. The model obtained the second-best performance using the 0.8 threshold value, while the worst performance deteriorated when no threshold was used.

In the second test, we evaluated the performance of the bottom-up saliency model using different Atrous rates in the ACL block (see Figure 5). Specifically, we tested the values 1, and a set of 4 and 8 for Atrous rates but kept the number of stages similar to the proposed method which is 2. When the Atrous rate is set to 1, which is a default value in a traditional convolutional neural network, then no dilation is performed in the kernels. Table 4 shows the results of this ablation test in which we see that the model

Tabela 4: Results of ablation test in which we tested the performance of bottom-up saliency model using different values of threshold T. The results are the values obtained by evaluation metrics AUC_Judd, CC, and SIM on the 360RAT dataset.

Atrous Rate	Evaluation		
	AUC_Judd	CC	SIM
1	0.5303	0.1448	0.1794
4 and 8	0.5000	0.2030	0.1764
6 and 12 (Proposed)	**0.8290**	**0.5438**	**0.4450**

performed almost similarly when Atrous rates were either set to 1, or 4 8 for two stages. But the highest performance of the model is achieved when Atrous rates 6 and 12 were used in the ACL block.

Table 5 presents the time required to train and test/run the proposed method including bottom-up, semantic, and hybrid saliency models using the 360RAT dataset. Notice that the time of pre-processing for semantic and hybrid saliency models is blank because the data pre-processed for the bottom-up saliency model was also used by semantic and hybrid saliency models. Because the bottom-up saliency model is the only model that we trained from scratch, the training time is blank for semantic and hybrid saliency models. The table shows the consumption of time for semantic and hybrid models only for post-processing because these models are generated as a part of post-processing in the proposed method (see Figure 1). In post-processing, the semantic model consumed 4 minutes to generate the predictions, while the hybrid model took approximately 10 minutes to merge and generate final predictions based on bottom-up and semantic saliency predictions.

Tabela 5: The time consumption of the proposed method using the 360RAT dataset.

Saliency Model	Pre-Processing (hours)	Training (minutes)	Testing (seconds)	Post-Processing (minutes)
Bottom-up	2.71	40.40	6	-
Semantic	-	-	-	4
Hybrid	-	-	-	10.21

Referências

[1] FARNEBÄCK, G. Two-frame motion estimation based on polynomial expansion. Springer Berlin Heidelberg, Berlin, Heidelberg, p. 363–370, 2003.

[2] ALAMGEER, S.; IRSHAD, M.; FARIAS, M. C. Q. CNN-based no-reference video quality assessment method using a spatiotemporal saliency patch selection procedure. *Journal of Electronic Imaging*, SPIE, v. 30, n. 6, p. 063001, 2021. Disponível em: <https://doi.org/10.1117/1.JEI.30.6.063001>.

[3] CHEN, L.-C. et al. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP, 06 2016.

[4] YANG, W. et al. Object detection in equirectangular panorama. *CoRR*, abs/1805.08009, 2018. Disponível em: <http://arxiv.org/abs/1805.08009>.

[5] PRADO, M. et al. 360rat: A tool for annotating regions of interest in 360-degree videos. In: *Proceedings of the Brazilian Symposium on Multimedia and the Web*. New York, NY, USA: Association for Computing Machinery, 2022. (WebMedia '22), p. 272–280. ISBN 9781450394093. Disponível em: <https://doi.org/10.1145/3539637.3557930>.

[6] SUTSKEVER, I. et al. On the importance of initialization and momentum in deep learning. In: *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*. [S.l.]: JMLR.org, 2013. (ICML'13), p. III–1139–III–1147.

[7] CHOLLET, F. *Keras*. 2021. Disponível em: <https://keras.io/>.

[8] CHENG, H.-T. et al. Cube padding for weakly-supervised saliency prediction in 360° videos. In: . [S.l.: s.n.], 2018. p. 1420–1429.

[9] MARTIN, D.; SERRANO, A.; MASIÁ, B. Panoramic convolutions for 360o single-image saliency prediction. In: . [S.l.: s.n.], 2020.