SalGCN: Saliency Prediction for 360-Degree Images Based on Spherical Graph Convolutional Networks

Haoran Lv, Qin Yang, Chenglin Li, Wenrui Dai, Junni Zou, Hongkai Xiong
Institute of Media, Information, and Network, Shanghai Jiao Tong University, Shanghai 200240, China
{lvhaoran,yangqin,lcl1985,daiwenrui,zoujunni,xionghongkai}@sjtu.edu.cn

ABSTRACT

The non-Euclidean geometry characteristic poses a challenge to the saliency prediction for 360-degree images. Since spherical data cannot be projected onto a single plane without distortion, existing saliency prediction methods based on traditional CNNs are inefficient. In this paper, we propose a saliency prediction framework for 360-degree images based on graph convolutional networks (SalGCN), which directly applies to the spherical graph signals. Specifically, we adopt the Geodesic ICOsahedral Pixelation (GICOPix) to construct a spherical graph signal from a spherical image in equirectangular projection (ERP) format. We then propose a graph saliency prediction network to directly extract the spherical features and generate the spherical graph saliency map, where we design an unpooling method suitable for spherical graph signals based on linear interpolation. The network training process is realized by modeling the node regression problem of the input and output spherical graph signals, where we further design a Kullback–Leibler (KL) divergence loss with sparse consistency to make the sparseness of the saliency map closer to the ground truth. Eventually, to obtain the ERP format saliency map for evaluation, we further propose a spherical crown-based (SCB) interpolation method to convert the output spherical graph saliency map into a saliency map in ERP format. Experiments show that our SalGCN can achieve comparable or even better saliency prediction performance both subjectively and objectively, with a much lower computation complexity.

CCS CONCEPTS

• **Computing methodologies** → **Computer vision**; **Interest point and salient region detections**.

KEYWORDS

Graph convolutional neural networks, saliency, 360 degree images, interpolation, graph unpooling.

ACM Reference Format:
Haoran Lv, Qin Yang, Chenglin Li, Wenrui Dai, Junni Zou, Hongkai Xiong. 2020. SalGCN: Saliency Prediction for 360-Degree Images Based on Spherical Graph Convolutional Networks. In *Proceedings of the 28th ACM International Conference on Multimedia (MM '20), October 12–16, 2020, Seattle, WA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3394171.3413733

MM '20, October 12–16, 2020, Seattle, WA, USA

ACM ISBN 978-1-4503-7988-5/20/10…$15.00
https://doi.org/10.1145/3394171.3413733

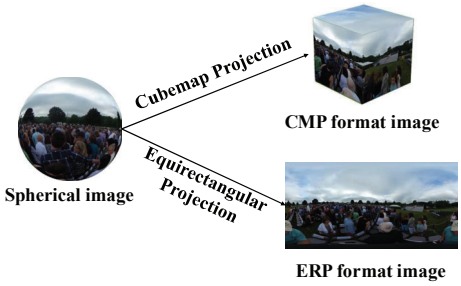

Figure 1: Different 2D projections of a spherical image.

1 INTRODUCTION

360-degree image/video that allows viewers to freely rotate their heads to acquire an immersive experience is a new multimedia form emerging in recent years. It has been widely applied to many fields, such as virtual reality (VR), augmented reality (AR), and autonomous cars [11]. This emerging trend of applications of 360-degree images/videos has also attracted research interests in modeling and predicting the visual attention for 360-degree images, which provides a fundamental support to facilitate different applications, such as tile-based 360-degree video streaming [13], automatic photography [19], and viewpoint prediction [23].

Visual saliency prediction aims to predict the image area of human interest by simulating human visual preference. Recently advanced saliency prediction models for images are mostly developed based on deep learning, where the development of convolutional neural networks (CNNs) enables these deep learning-based visual saliency prediction to achieve state-of-the-art performance in traditional 2D images/videos [7, 8, 14, 15]. However, these traditional convolution kernels with a regular grid design cannot be directly applied to deal with spherical features, which inevitably incurs difficulties in applying CNNs to process spherical data with a non-Euclidean geometry. To address this problem, many practical solutions have been proposed for processing 360-degree (or spherical) images. For example, some works proposed to convert 360-degree images into 2D images through some projection methods, which then enabled application of traditional 2D saliency prediction methods to process the projected 360-degree images [1, 12, 18]. Two commonly used projection method, i.e., equirectangular projection (ERP) and cubemap projection (CMP), are shown in Figure 1. Along another research direction, there have also been works dedicated to re-designing the convolution kernel to make it suitable for spherical images, which then allowed to directly perform saliency prediction on 360-degree images [25].

Although the above solutions provide two possible directions for spherical image saliency prediction, some problems are still

remaining and needed to be addressed. 1) With either the ERP or CMP projection method, a projection distortion will be inevitably incurred when projecting the spherical image onto a 2D plane. For example, the ERP projection introduces a large amount of image distortion at the two poles. In comparison, the CMP projection projects a spherical image onto 6 planes, which therefore incurs a lower distortion than the ERP projection. However, due to the need to deal with the six planes separately, the CMP projection introduces redundant image boundaries which makes the saliency map discontinuous at the stitching boundary [12]. 2) To further reduce the image distortion caused by CMP projection and thus alleviate the image boundary, some works use more projection planes to project a 360-degree image, which however increases greatly the computation complexity [1, 21]. 3) In order to re-design the convolution kernel to make it suitable for spherical images, current methods need to interpolate the feature maps during convolution, which would accumulate the interpolation error when the network deepens and eventually lead to a poor prediction accuracy [25].

To address the above problems, in this paper, we propose a saliency prediction architecture for 360-degree images based on graph convolutional networks (SalGCN). Specifically, we use the Geodesic ICOsahe-dral Pixelation (GICOPix)-based graph signal construction method to convert the original spherical images and their ground truth saliency maps in ERP format into corresponding spherical graph signal representations. We then propose a graph saliency prediction network that is able to directly extract the spherical features and build up the mapping between the input spherical graph signal representation of the original spherical image and the output spherical graph saliency map. The proposed graph saliency network adopts a basic encoder-decoder structure, where we design the spherical graph unpooling layer (SG-Unpooling) based on linear interpolation for upsampling of the spherical graph signals. The network training is realized by modeling the node regression problem of the input and output spherical graph signals, where we further design a KL divergence loss with sparse consistency to make the sparseness of the saliency map closer to the ground truth. Eventually, to obtain the ERP format saliency map for evaluation, we further propose a spherical crown-based (SCB) interpolation method to convert the output spherical graph saliency map into a saliency map in ERP format.

The main advantages of the proposed SalGCN is three-fold. 1) Our method of extracting spherical features using GCNs fundamentally addresses the projection distortion problem. 2) Since the proposed saliency prediction network processes directly on the spherical graph signals, the redundant image boundaries and repeated computations on different projection planes are avoided. 3) The entire graph convolution process will not interpolate the feature map, which then prevents interpolation errors from accumulating as the number of network layers increases. As a verification, experiments show that even at an input size that is 3 order-of-magnitude smaller than the other comparison methods, our SalGCN can still achieve a comparable performance with the best method (SalGAN360 [1]), with the running time also reduced by 3 order-of-magnitude.

The rest of this paper is organized as follows. Sections 2 and 3 describe current researches of saliency prediction models and preliminaries, respectively. Section 4 details the proposed framework. Section 5 shows the subjective and objective performance evaluation. Finally, concluding remarks are given in Section 6.

2 RELATED WORK

2.1 Extending 2D Saliency Prediction Models

Works within this branch extend the existing saliency prediction models from 2D to spherical images by projecting them onto a 2D plane through some projection methods and then applying these well-performed saliency prediction models proposed for 2D images. Lebreton *et al.* [9] proposed GBVS360 for saliency prediction of 360-degree images. The core idea is to leverage the graph-based visual saliency (GBVS) approach [6] (a heuristic approach for generating saliency map based on the Markov chain) for multiple viewport images generated by spherical images, and then project the viewport back to the ERP plane to obtain the saliency map. Ling *et al.* [10] proposed the color dictionary sparse representation (CDSR) approach to generate saliency maps, which used an over-complete color dictionary to sparsely represent multiple sub-images divided by the ERP format image, and finally stitching into the full saliency map. Monroy *et al.* [12] applied SalNet [5] to extract saliency for each face of the CMP format image. Different from that, Chao *et al.* [1] applied SalGAN [14] to the 486 cut planes projected by 360-degree image to achieve better performance.

In general, these works need to perform saliency prediction on multiple 2D images, resulting in an inevitable projection distortion and a very high computation complexity. Although the use of CMP projection can balance the amount of calculation and image distortion, it will additionally introduce redundant image boundaries, which causes discontinuities of the saliency map at stitching boundaries. In addition, because the same object may have different shapes at different projection angles, it is unreasonable to use the same convolution kernel to process different projection planes. In our work, instead of projecting a spherical image onto multiple 2D planes, a spherical graph signal representation is directly constructed from the original spherical image, and then spherical features are extracted through a graph convolutional network. This approach not only naturally possesses the graph convolution of weight sharing, but also avoids introducing redundant image boundaries and huge computation complexity. More importantly, it completely addresss the distortion problem caused by image projection.

2.2 Re-designing the Convolution Kernel

Alternatively, some works re-design the convolution kernel to make it suitable for processing the spherical image (or its 2D projection in ERP format), and then perform saliency prediction directly on this spherical image (or its ERP projection). Zhang *et al.* [25] proposed a spherical U-net [17] suitable for ERP format images for saliency prediction, where the convolution kernel of the spherical convolution was defined at the pole position, and the convolution operation was realized by continuously re-sampling the feature map to adapt the shape of the convolution kernel. However, since the pixel positions of feature map do not coincide with the convolution kernel, re-sampling feature maps will require interpolation. Su *et al.* [20] experimentally proved that the interpolation of feature maps would affect the performance of deep networks. At the same

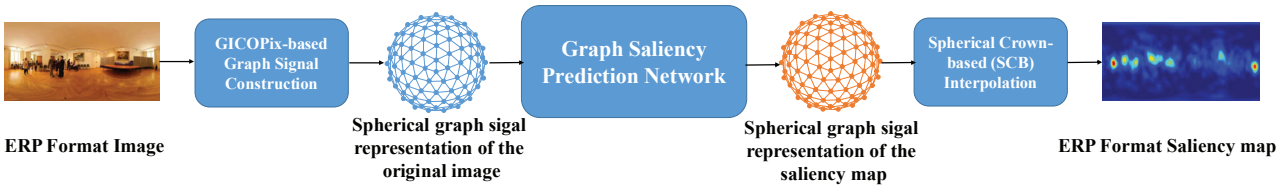

Figure 2: The proposed SalGCN framework with three modules: 1) the *GICOPix-based graph signal construction* converts an ERP format image to a spherical graph signal representation, 2) the *graph saliency prediction network* generates a spherical graph signal representation for the saliency map based on the input spherical graph signal, and 3) the *spherical crown-based (SCB) interpolation* finally reconstructs the resulting saliency map in ERP format from the spherical graph saliency signal.

time, spherical U-net desires a very large demand for computing resources (e.g., four NVIDIA Tesla P40 GPUs). And due to the influence on feature map interpolation, the spherical U-net can only achieve the same performance on the Salient360 [16] dataset as the traditional method directly applied to the ERP format image. Different from the spherical U-net, our proposed method uses graph convolution to extract spherical features, such that the interpolation operation of the feature map will not be introduced in the convolution process. In addition, due to the use of the ChebNet as the graph convolutional layer, our network does not rely on a large number of computation resources.

3 PRELIMINARIES

We define a spherical image as $S_i(\theta, \varphi)$, where $\theta \in [0, \pi]$ and $\varphi \in [0, 2\pi]$ represent the latitude and longitudes of a spherical point, respectively. Then, its ERP projection with size $W \times H$ can be denoted as $E_i(x, y) \in \mathbb{R}^{W \times H \times 3}$, where the last dimension corresponds to the three channels of the RGB image. We use the GICOPix scheme [22] to obtain an undirected and connected graph $G(\mathcal{V}, \mathcal{E}, W)$ from $E_i(x, y)$, where $\mathcal{V}$ represents a vertex set of size N, $\mathcal{E}$ represents an edge set, and W is a weighted adjacency matrix of size $N \times N$ with $w_{i,j}$ being the connection weight between the vertices v_i and v_j. In this paper, the connection weight of any two neighboring nodes $w_{i,j}$ is set to 1, and the normalized graph Laplacian is defined as $L = I_N - D^{-1/2} W D^{-1/2}$, where $D \in \mathbb{R}^{N \times N}$ is a diagonal degree matrix of the weight matrix with $d_{i,i} = \sum_{j=1}^{N} w_{i,j}$ and I_N is the identity matrix of order N.

In order to make the GCN have CNN-like localization characteristics, we use the same ChebNet as in SGCN [22] to approximate the convolution kernel by recursively computing a Chebyshev polynomial. For a graph signal x, the graph spectral convolution is defined as

$$y = (\sum_{k=0}^{K-1} \theta_k T_k(\widetilde{L}))x, \tag{1}$$

where K is the order of the Chebyshev polynomial, θ_k denotes the Chebyshev polynomial coefficient corresponding to the learnable parameter, and $\widetilde{L} = 2L/\lambda_{max} + I_N$ with λ_{max} denoting the largest eigenvalue of L. Thus, $T_k(\widetilde{L}) \in \mathbb{R}^{N \times N}$ represents the Chebyshev polynomial that can be computed by the recursive relation $T_k(\widetilde{L}) = 2\widetilde{L}T_{k-1}(\widetilde{L}) - T_{k-2}(\widetilde{L})$ with $T_0 = I_N$ and $T_1 = \widetilde{L}$. The main benefit of this approach is that it greatly reduces the learning complexity to $(O(K|\varepsilon|))$ and constrains the spectral convolution defined in Equation (1) to be K-localized, i.e., the spectral convolution operation relating to the center vertex only depends on all the vertex values and edge weights on a path of length $k < K$.

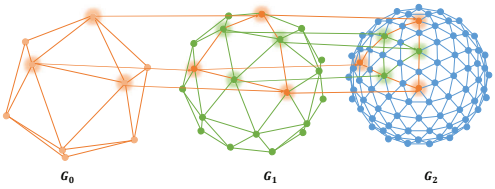

Figure 3: Graph signal construction process from G_0 to G_2.

4 PROPOSED METHOD

The overall framework of the proposed SalGCN is shown in Figure 2, which comprises three modules in cascade. First, the original spherical image in ERP format is converted to a spherical graph signal representation by the Geodesic ICOsahedral Pixelation (GICOPix)-based graph signal construction module. We then design a graph saliency prediction network to generate a spherical graph signal representation for the saliency map of the same size as the graph signal representation for the spherical image. Finally, we propose a spherical crown-based (SCB) interpolation module to reconstruct the ERP format saliency map from its spherical graph signal representation. In addition, we model the network training as a node regression problem in the graph signal, and propose a KL loss with sparse consistency for network training.

4.1 Problem Formulation

The proposed graph convolutional network-based image saliency prediction aims to build up a mapping between the spherical graph signal representation of the input spherical image and the spherical graph signal representation of the output saliency map. Here, we denote N_G as our deep graph convolutional network and *GICOPix* as the GICOPix-based graph signal construction operation, then the objective of saliency prediction for a 360-degree image can be expressed as:

$$\min_{\theta_k} \sum_{t=0}^{N} dist\left|GICOPix(E_{gt})[v_t], \mathrm{N}_G(GICOPix(E_i))[v_t]\right|, \tag{2}$$

where E_{gt} and E_i represent the ground truth of saliency map and the input RGB image in ERP format, respectively, and v_t represents the t-th node in the graph signal. Therefore, the entire optimization

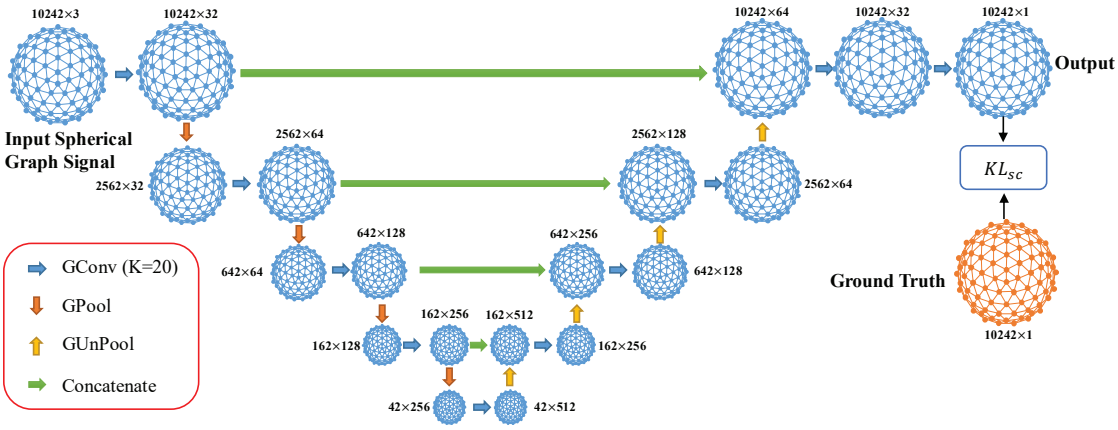

Figure 4: Architecture of the proposed graph saliency prediction network.

process actually completes the node regression task of the spherical graph signal representation.

4.2 GICOPix-based Graph Signal Construction

In the proposed SalGCN framework, we first leverage the GICOPix-based graph signal construction module [22] to construct a spherical graph signal representation from the original 360-degree image in ERP format. As shown in Figure 3, the workflow is as follows. 1) We first construct the largest internal geodesic icosahedron consisting of 20 equilateral triangles for the spherical signal S, then all 12 vertices of the icosahedron are used as the graph sampling points for the spherical signal to generate the level 0 graph signal G_0. 2) We further divide each face (isolateral triangle) of the initial geodesic icosahedron into 4 equilateral triangles. By connecting the sphere center and the newly generated vertices of these equilateral triangles and extending these connecting lines to the sphere, we then generate some new graph sampling points. The newly generated graph sampling points on the sphere are merged with G_0 to obtain the Level 1 graph signal G_1. 3) We repeat the previous step to generate more levels of graph signal, denoted as G_l, $l \in \mathbb{N}$. The relationship between the level index l and the number of vertices of the graph signal is $N_l = 10 \times 2^{2l} + 2$.

4.3 Graph Saliency Prediction Network

4.3.1 Network Structure. Inspired by U-net [17], we propose a deep graph convolutional network for graph saliency prediction by using the basic encoder-decoder structure, as illustrated in Figure 4. The input of the network comes from the output of the GICOPix-based graph signal construction module, which is the spherical graph signal representation converted from the orignal spherical image in ERP format. For the encoding part, we adopt five graph convolutional layers, with a rectified linear unit (ReLU) used as the activation function for each graph convolutional layer. In addition, the first four graph convolutional layers are followed with a graph pooling layer as in [22] to obtain hierarchical representations of the graph signal. For the decoding part, we also use ReLU as the activation function of five graph convolutional layers. In addition, we propose a graph unpooling layer to be added before the first four convolutional layers. In particular, the input for each of these four graph convolutional layers is obtained by concatenating the output of its previous unpooling layer and the output of the corresponding graph convolutional layer with the same output size from the encoding part.

In summary, the proposed graph saliency prediction network has a total of ten graph convolutional layers, four pooling layers, and four unpooling layers. The output of the network is a spherical graph signal representation for the predicted saliency map, which will be then fed into the SBC interpolation module to generate a saliency map in ERP format.

4.3.2 Graph Pooling Layer. Multi-scale feature maps are an integrated part of the network. In this paper, we adopte the rotation-equivariant pooling layer proposed in [22], which can be viewed as an inverse process of the graph construction, i.e., G_l is coarsened to G_{l-1} by discarding newly added vertices in the process from G_{l-1} to G_l. The change in the number of vertices in the pooling process can be approximated as $N_l/N_{l-1} \approx 4$, which can be considered as a parallel to the pooling operation with stride = 2 in the traditional CNN.

4.3.3 SG-Unpooling Layer. Deep learning-based image saliency prediction schemes usually use a fully convolutional network (FCN) for end-to-end training, which thus requires that the output size of the network is consistent with the input size. To satisfy this, a common practice is to use the unpooling layer to upsample the feature map. To the best of our knowledge, there is little existing work on the unpooling layer design for graph convolutional networks. In this paper, we propose a novel unpooling operation for spherical graph signals, which is named SG-unpooling.

Benefiting from the GICOPix-based method to construct the graph signal, we adopt a graph unpooling strategy by using linear interpolation. Specifically, using $\mathcal{V}_l$ to denote the vertex set of the spherical graph at the l-th level, the graph unpooling process is defined as

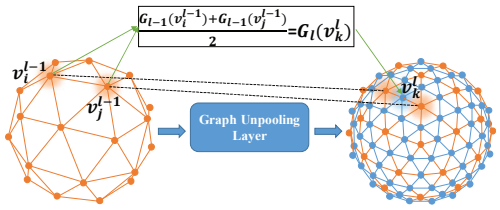

Figure 5: The proposed graph unpooling layer.

$$G_l(v_k^l) = \begin{cases} G_{l-1}(v_j^{l-1}), & if\ Pos(v_j^{l-1}) = Pos(v_k^l), \\ \frac{G_{l-1}(v_i^{l-1})+G_{l-1}(v_j^{l-1})}{2}, & if\ \mathrm{Nei}(v_k^l) = \{v_i^{l-1}, v_j^{l-1}\}, \end{cases} \tag{3}$$

where v_k^l represents the k-th vertex in the graph at the l-th level, the operation $Pos(\cdot)$ returns the coordinates of the vertex v on the sphere, and the operation $Nei(\cdot)$ returns the two neighboring vertices of vertex v_k^l in G_{l-1}. The specific unpooling operation is graphically illustrated in Figure 5.

The proposed graph unpooling operation does not change the value of the original vertex in the preceding graph level, but generates a value for the newly generated vertex in the current graph level by linear interpolation, which is different from interpolation of the feature maps during convolution and also achieves a better performance than simply setting the values of newly generated vertices to 0. This advantage has also been demonstrated through the results in Section 5.4.

4.3.4 KL Loss With Sparse Consistency. Although the KL divergence is widely used in regression tasks, traditional KL divergence can only measure the distance between distributions. For saliency prediction, if only the KL divergence is used to measure the difference between the ground truth of the spherical graph saliency map $G_{gt}(v_t)$ and the output spherical graph saliency map $G_s(v_t)$ of the network, the trained network will produce a denser saliency map. In Equation (5), for example, when $G_{gt}(v_t)$ is very small, even if the difference between $G_{gt}(v_t)$ and $G_s(v_t)$ is large, it will be given a very small weight by $G_{gt}(v_t)$, such that the large difference between $G_{gt}(v_t)$ and $G_s(v_t)$ is not reflected in the loss function, which would in turn affect the network training. In order to make the saliency map's sparseness closer to the real situation, we propose a KL divergence with sparse consistency, as

$$KL_{sc} = \frac{\lambda}{1+\lambda} KL_s + \frac{1}{1+\lambda} KL_{hist}, \tag{4}$$

$$KL_s = \sum_{t=1}^{N} G_{gt}(v_t) \log\left[\frac{G_{gt}(v_t)}{G_s(v_t)}\right], \tag{5}$$

$$KL_{hist} = \sum_{i=1}^{255} hist(G_{gt})(i) \log\left[\frac{hist(G_{gt})(i)}{hist(G_s)(i)}\right], \tag{6}$$

where λ is a hyper-parameter used to control the weights of two KL divergences. KL_s represents the KL divergence between the ground truth of the spherical graph saliency map G_{gt} and the output spherical graph saliency map G_s of the network. KL_{hist} represents the KL divergence between the histogram of G_{gt} and the histogram of G_s, where the number of groups in the histogram is 255. The entire loss function in Equation (4) can be understood as adding a penalty term to the traditional KL loss. Equation (5) makes the spatial distribution of the network output continuously approach the ground truth saliency map, while Equation (6) makes the numerical distribution of the network output continuously approach the ground truth saliency map.

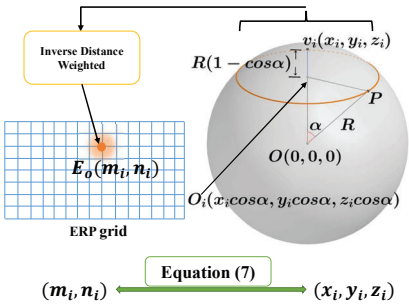

Figure 6: The spherical crown centered at (x_i, y_i, z_i).

4.4 Spherical Crown-based (SCB) Interpolation

The ultimate goal of 360-degree image saliency prediction is to obtain an ERP format saliency map. Therefore, we further propose a novel spherical crown-based (SCB) interpolation method to convert the output spherical graph saliency map of the graph saliency prediction network to the ERP format saliency map.

As shown in Figure 3, the constructed spherical graph signal contains a relatively uniform distribution of sample points on the sphere. Due to the distortion introduced by projecting from a non-Euclidean spherical space to the Euclidean 2D plane, however, these spherical points after projection will become non-uniformly distributed points in the ERP format image. Therefore, the proposed SCB interpolation method essentially completes the conversion from non-uniform sample points to uniform grid points in the ERP format image. Since the spherical graph signal has a relatively uniform distribution on the sphere, we implement the interpolation process on the sphere. Denoting $E_o(m_i, n_i)$ as the pixel value to be interpolated at position $(m_i, n_i), 1 \le i \le W \times H$ in the ERP grid, the Cartesian coordinate of the corresponding spherical point (x_i, y_i, z_i) can be determined by the inverse ERP projection and written as

$$\begin{cases} x_i = \sin(\frac{\lceil i/W \rceil}{H} \times \pi) \cos(\frac{i - \lfloor i/W \rfloor W}{W} \times 2\pi), \\ y_i = \sin(\frac{\lceil i/W \rceil}{H} \times \pi) \sin(\frac{i - \lfloor i/W \rfloor W}{W} \times 2\pi), \\ z_i = \cos(\frac{\lceil i/W \rceil}{H} \times \pi). \end{cases} \tag{7}$$

Then, $P_{E_o} = \{(x_i, y_i, z_i) | \forall 1 \le i \le W \times H\}$ represents the complete set of Cartesian coordinates of the spherical points corresponding to the set of ERP grid points. Furthermore, we denote the output of graph saliency prediction network as $G_s(\mathcal{V})$ where $|\mathcal{V}| = N$, and the corresponding Cartesian coordinate set as $P_{G_s} = \{(x_t, y_t, z_t) | \forall v_t \in G_s(\mathcal{V})\}$. As shown in Figure 6, on a sphere that is with a radius of R and centered at the origin, we fix $v_i = (x_i, y_i, z_i) \in P_{E_o}$ as the center point to construct a spherical crown with a height of $R[1 - \cos(\alpha)]$, and add the points in $G_s(\mathcal{V})$ within this spherical crown to the set $\mathcal{U}$. Without loss of generality, here we set $R = 1$

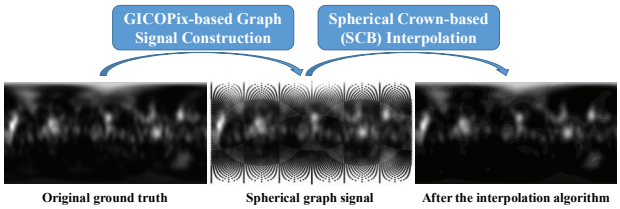

Figure 7: Effect of the SCB-Interpolation method.

and $\alpha = 2\pi/(W-1)$, and then get the bottom plane equation of the spherical crown centered at $v_i = (x_i, y_i, z_i)$ as

$$x_i(x - x_i\cos\alpha) + y_i(y - y_i\cos\alpha) + z_i(z - z_i\cos\alpha) = 0, \quad (8)$$

which is perpendicular to vector (x_i, y_i, z_i). If a point (x, y, z) is within this spherical crown centered at (x_i, y_i, z_i), then it should be separated with the origin by the plane defined in Equation (8), which mathematically can be expressed as

$$(\cos\alpha - x_i x - y_i y - z_i z)\cos\alpha \leq 0. \quad (9)$$

Finally, we interpolate the pixel value of $v_i = (x_i, y_i, z_i)$ by the inverse distance weighted averaging, and use this interpolated pixel value to represent $E_o(m_i, n_i)$, as

$$E_o(m_i, n_i) = \sum_{v_t \in \mathcal{U}} \frac{G_s(v_t)}{(D^2(v_t) + e)\sum_{v_u \in \mathcal{U}} \frac{1}{D^2(v_u)+e}}, \quad (10)$$

where $D(v_t)$ denotes the Euclidean distance between vertices v_t and v_i and e is a sufficiently small constant (set to $1e-8$ in this paper) to prevent the denominator from being zero. Equation (10) can be understood as first obtaining weights for all vertices in $\mathcal{U}$ according to their distances from $v_i = (x_i, y_i, z_i)$, and weighted averaging the corresponding vertex values of $v_t \in \mathcal{U}$ to obtain $E_o(m_i, n_i)$. We visualize the spherical graph saliency map as a scatter plot and show the interpolation effect of the proposed SCB interpolation method in Figure 7, demonstrating that it can recover the original saliency map in ERP format from the spherical graph saliency map.

5 EXPERIMENTS

5.1 Experiment Setup

Our graph saliency prediction network is implemented on the TensorFlow framework. To train and test this network, we use head-eye movements images from Salient360 dataset [16], including 40 training images and 25 test images in ERP format. For network training, we apply the GICOPix-based graph signal construction to obtain a spherical graph signal representation of 5-level (10242 points) for both the ERP format spherical images and ground truth saliency maps. We use the Adam optimizer to train the model for 20000 epochs at a fixed learning rate of 0.0001 and with a batch size 5. In order to make the model better trained, we choose $\lambda = 1$ in Equation 4. To prevent overfitting, we add a batch normalization layer and a dorpout rate of 0.9 after each convolutional layer, and add the L_2 regularization term with a weight of 5e-5 to the loss function. The entire training process is performed on a single Nvidia GTX1080Ti GPU (11GB).

Table 1: Performance evaluation of different methods on Salient360 dataset [16], where different index represents a different method: (1)-BMS, (2)-BMS360, (3)-GBVS360, (4)-SalNet360, (5)-SalGAN360, (6)-SalGCN (Ours). The bold number represents the best performance, and the underlined number represents the second-best performance. ↑ and ↓ show the direction of higher accuracy.

Metric	(1)	(2)	(3)	(4)	(5)	(6)
KL↓	0.554	0.526	0.560	0.494	<u>0.431</u>	**0.428**
CC↑	0.544	0.503	0.440	0.536	**0.642**	<u>0.589</u>
SIM↑	0.647	0.661	0.645	0.669	<u>0.681</u>	**0.686**
NSS↑	0.864	0.767	0.667	0.882	**1.122**	<u>0.945</u>
AUC↑	0.719	0.699	0.676	0.720	**0.772**	<u>0.736</u>
Param.	-	-	-	**7.26M**	31.78M	<u>8.71M</u>
Input.	0.08M	0.08M	-	<u>0.46M</u>	23.89M	**0.01M**
Time.	-	-	-	<u>8.55s</u>	738.92s	**0.63s**

5.2 Metrics

For 360-degree image saliency prediction, it is unreasonable to directly evaluate the predicted saliency map in ERP format, since latitudinal distortions are introduced by the ERP projection. Instead, we use the toolbox provided by [4] for the evaluation, where a saliency map is projected back to the sphere with uniformly sampled points and all metrics are calculated only at these sample points. Here, we select five metrics to evaluate our proposed SalGCN: Kullback-Leibler divergence (KL), linear correlation coefficient (CC), similarity measure (SIM), normalized scanpath saliency (NSS), and area under the curve (AUC). Except the KL metric, a larger value of all the remaining metrics would indicate a more accurate prediction for the saliency.

The evaluation of image saliency prediction usually requires a comparison between the predicted saliency map with the ground truth saliency map after being blurred by a Gaussian filter [7]. In this paper, we use the proposed SCB interpolation method to upsample the output ERP format saliency map to a resolution of 1024×2048, and then blur the saliency map with a Gaussian filter with kernel size of 64. This choice of Gaussian kernel size is a common setting, which also guarantees the best performance of the proposed SalGCN.

5.3 Comparison To State-of-the-arts

To verify the superiority of the proposed SalGCN, we compare its performance with the following state-of-the-art methods, including BMS [24], BMS360 [9], GBVS360 [9], SalNet360 [12], and SalGAN360 [1]. Among them, BMS, BMS360 and GBVS360 are heuristic methods, while SalNet360 and SalGan360 are deep learning-based methods. For all learning-based methods, we also compare the amount of parameters, model operation time, and the number of input pixels to the neural network.

Table 1 shows the performance evaluation of different methods on Salient360 dataset [16], where the bold and underlined numbers represent the best performance and the second-best performance, respectively. It can be seen that our proposed SalGCN achieves the best prediction performance in terms of KL and SIM, and the second-best prediction performance in terms of CC, NSS and AUC. From the

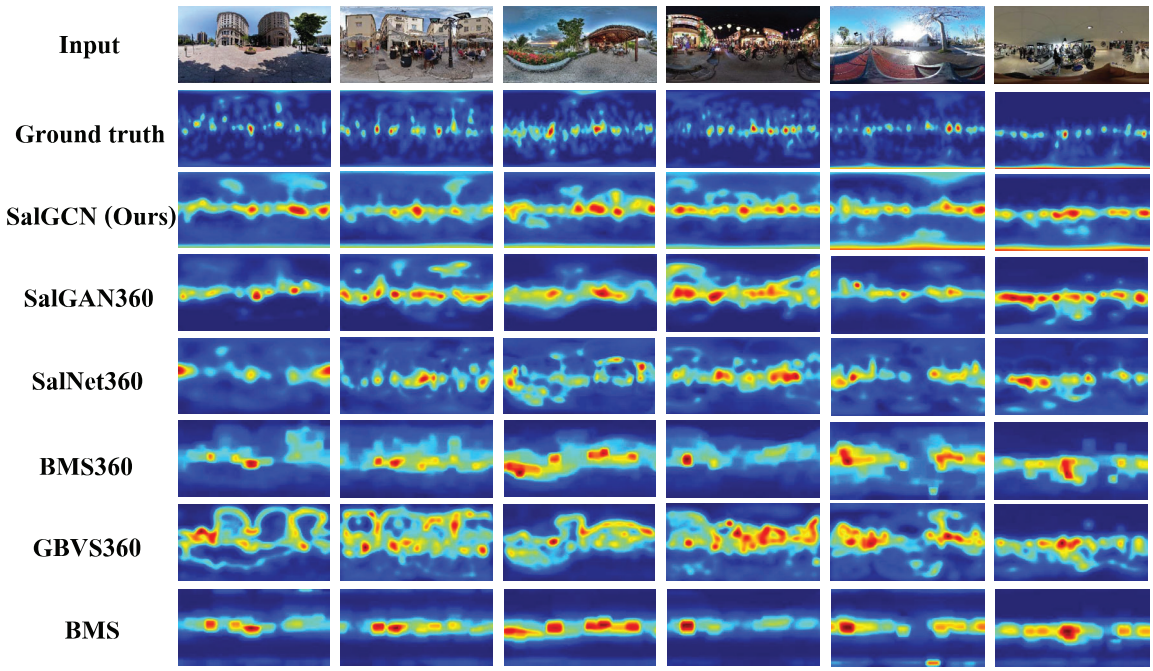

Figure 8: Saliency maps of 6 test images randomly selected from Salient360 dataset [16] and the prediction results.

perspective of model parameters, the amount of our SalGCN is similar to that of SalNet360 and far less than that of SalGAN360. These comparison results demonstrate that our SalGCN is not inferior to the existing deep neural network-based models in performance. In addition, it can be found that SalGCN achieves a better performance in distribution-based metrics (e.g., KL), but a slightly lower performance in location-based metrics (e.g., AUC, NSS). This is due to the limited research progress of the graph convolutional network itself, i.e., currently it is not possible to process an arbitrarily large graph signals with deep networks. However, it is worth noting that in this paper, we construct a graph with only 10242 nodes, while in SalNet360 and SalGAN360, 460800 ($6 \times 320 \times 240$) and 23937024 ($(9 \times 9 \times 6 + 1) \times 256 \times 192$) pixels are required as the input, respectively, to get the saliency map of a 360-degree image. In other words, the input size of our SalGCN is order-of-magnitude lower than those of SalNet360 and SalGAN360, which indicates a seriously unequal amount of information needed for prediction. Even though, our SalGCN outperforms SalNet360 in all metrics, and achieves the best performance among all models in the KL and SIM metrics, which demonstrates that SalGCN can fit the spherical distribution very well and has a great potential in the position-based metrics. In addition, thanks to the smaller network size and input size, and the direct processing of the entire spherical graph signal through the proposed saliency prediction network, SalGCN is much less computationally expensive than SalNet360 and SalGAN360. Experiments show that under the same setting of computation resources and test images, the average model operation time of SalGCN for each 360-degree image is only 0.63 second, while SalNet360 and SalGAN360 require 8.55 seconds and 738.92 seconds, respectively.

In order to visually compare the saliency maps generated by different methods, we randomly selected 6 test images from the Salient360 dataset, and show the ERP format saliency maps generated by different methods in Figure 8. It can be seen that our SalGCN can predict the saliency map very well. Specifically, as shown in the last two columns, our SalGCN can better capture the saliency in the high latitude regions.

5.4 Ablation Studies

To further verify the effectiveness of our SalSGCN, we design the following baselines to evaluate the contributions and gains introduced by various components in our framework. *(1) CNN based U-net:* We use CNN to build up a standard U-net, input 128×256 images in ERP format directly to the network, and use the mean square error (MSE) loss for network training. *(2) SphereNet [2] based U-net:* We change the convolutional layer in the above CNN based U-net to Spherenet, while other settings remain the same. *(3) SalGCN with gUnpool [3]:* We replace all the unpooling layers in our SalGCN with the gUnpool layer (that simply adds zero to the upsampling position) in [3], and keep the other settings unchanged. *(4) SalGCN with standard KL divergence:* We train our SalGCN using standard KL divergence in Equation (5). *(5) SalGCN with less input nodes:* We train our SalGCN using 2562 input nodes. *(6)SalGCN with more input nodes:* We train our SalGCN using 40962 input nodes.

The results of ablation studies are shown in Table 2. The performance of Baselines (1) and (2) are much worse than SalGCN in all metrics. In addition, we surprisingly find that Baseline (1) using CNN even outperforms Baseline (2) using SphereNet. We believe that the main reason for this phenomenon is that the convolutional layer of SphereNet needs to interpolate the feature map, which

Table 2: Performance evaluation under different baselines.

Baseline	KL↓	CC↑	SIM↑	NSS↑	AUC↑
(1)	0.620	0.525	0.663	0.791	0.698
(2)	0.790	0.403	0.626	0.609	0.654
(3) (Ours)	0.477	0.565	0.678	0.932	0.732
(4) (Ours)	0.462	0.573	0.680	0.902	0.729
(5) (Ours)	0.598	0.527	0.658	0.856	0.716
(6) (Ours)	**0.415**	**0.611**	**0.703**	**0.972**	**0.743**
SalGCN (Ours)	0.428	0.589	0.686	0.945	0.736

leads to the continuous accumulation of errors as the network deepens. At the same time, it also shows that the convolution operation that needs to interpolate the feature map is not suitable for deep networks. Our SalGCN not only takes into account the distortion problem caused by the projection of the spherical image onto the 2D plane, but also avoids introducing the interpolation operation of the feature map in the convolution, thereby achieving a much better performance than Baselines (1) and (2). Baseline (3) also presents a certain performance gap than our SalGCN. This is due to our graph unpooling operation that retains the position information of the old vertices and calculates the value of the newly generated vertices through linear interpolation. The results of Baseline (4) prove that our proposed KL_{sc} loss plays a positive role in network training. Finally, the results of Baselines (5) and (6) prove that the performance of the proposed SalGCN can be further improved by increasing the number of input nodes, which demonstrates a great potential of our SalGCN in terms of further improving the prediction accuracy.

5.5 Rotation Equivariance

Benefiting from the Chebyshev polynomial filters, SalGCN is able to achieve the rotation equivariance of the spherical signal. As shown in Figure 9, we traine SalGCN on the training data without rotation and test it on the test image with a certain rotation. Specirically, we rotate the test image by $\frac{\pi}{6}$, $\frac{\pi}{3}$, $\frac{\pi}{2}$ along the θ direction, respectively. The results show that SalGCN has a good adaptability to the rotation of the spherical images.

5.6 Stitching Boundary Comparison

It is also worth noting that because SalGCN directly processes the image on the spherical surface, the image boundary is not introduced in the convolution process, which fundamentally eliminates the problem of redundant boundaries generated by the traditional CNN method. In Figure 10, we show the performance comparison between our SalGCN and the traditional CNN-based method (SalNet360, SalGAN360) on the image stitching boundary. Among them, the orange solid lines in the bottom row represent the image boundary due to CMP projection. It can be clearly seen that because SalNet360 needs to project the spherical image onto multiple tangent planes and process them separately, the saliency map after stitching has very obvious image boundaries. For SalGAN360, due to the use of 486 directions of projection, the 2D images in adjacent projection directions have a lot of overlapping area in content, which makes the boundary problem alleviated. In contrast, our SalGCN has no such problems.

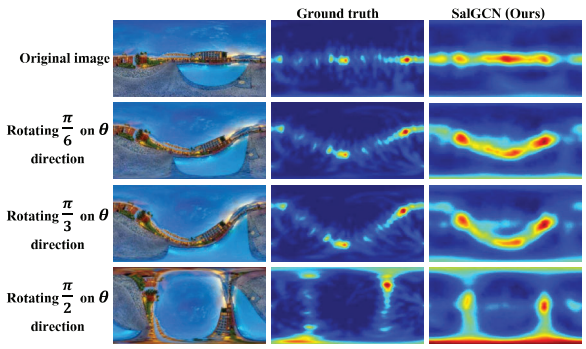

Figure 9: Rotation equivariance of the proposed SalGCN.

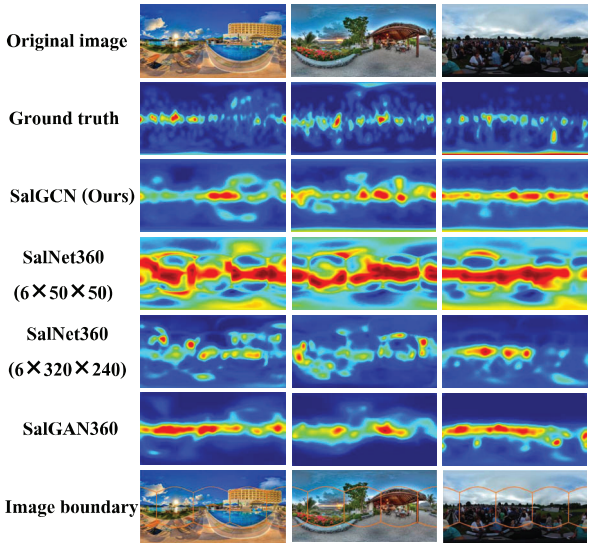

Figure 10: Comparison of image boundaries between SalGCN and SalNet360.

6 CONCLUSION

In this paper, we proposed a framework for predicting the saliency of spherical images using graph convolutional networks. We achieved saliency prediction by modeling the saliency prediction task of spherical images into the node regression task of spherical graph signals. We also designed the unpooling operation of the spherical graph signal based on linear interpolation to achieve the upsampling of the spherical graph signal. Finally, we proposed a spherical crown based interpolation method to convert the spherical graph signal into an ERP format saliency map. Experiments have shown that our SalGCN could achieve comparable or even better saliency prediction accuracy with a much less computation complexity.

ACKNOWLEDGMENTS

This work was supported in part by the National Natural Science Foundation of China under Grants 61871267, 61931023, 61972256, 91838303 and 61831018, and in part by the Shanghai Rising-Star Program under Grant 20QA1404600.

REFERENCES

[1] Fang-Yi Chao, Lu Zhang, Wassim Hamidouche, and Olivier Deforges. 2018. SAl-GAN360: visual saliency prediction on 360 degree images with generative adversarial networks. In *2018 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*. IEEE, 01–04.

[2] Benjamin Coors, Alexandru Paul Condurache, and Andreas Geiger. 2018. Spherenet: Learning spherical representations for detection and classification in omnidirectional images. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 518–533.

[3] Hongyang Gao and Shuiwang Ji. 2019. Graph u-nets. *arXiv preprint arXiv:1905.05178* (2019).

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

[5] Le Han, Xuelong Li, and Yongsheng Dong. 2018. SalNet: Edge constraint based end-to-end model for salient object detection. In *Chinese Conference on Pattern Recognition and Computer Vision (PRCV)*. Springer, 186–198.

[6] Jonathan Harel, Christof Koch, and Pietro Perona. 2007. Graph-based visual saliency. In *Advances in neural information processing systems*. 545–552.

[7] Xun Huang, Chengyao Shen, Xavier Boix, and Qi Zhao. 2015. Salicon: Reducing the semantic gap in saliency prediction by adapting deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*. 262–270.

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

[9] Pierre Lebreton and Alexander Raake. 2018. GBVS360, BMS360, ProSal: Extending existing saliency prediction models from 2D to omnidirectional images. *Signal Processing: Image Communication* 69 (2018), 69–78.

[10] Jing Ling, Kao Zhang, Yingxue Zhang, Daiqin Yang, and Zhenzhong Chen. 2018. A saliency prediction model on 360 degree images using color dictionary based sparse representation. *Signal Processing: Image Communication* 69 (2018), 60–68.

[11] Maxime Meilland, Andrew I Comport, and Patrick Rives. 2010. A spherical robot-centered representation for urban navigation. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 5196–5201.

[12] Rafael Monroy, Sebastian Lutz, Tejo Chalasani, and Aljosa Smolic. 2018. Salnet360: Saliency maps for omni-directional images with cnn. *Signal Processing: Image Communication* 69 (2018), 26–34.

[13] Cagri Ozcinar, Julián Cabrera, and Aljosa Smolic. 2019. Visual attention-aware omnidirectional video streaming using optimal tiles for virtual reality. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 9, 1 (2019), 217–230.

[14] Junting Pan, Cristian Canton Ferrer, Kevin McGuinness, Noel E O'Connor, Jordi Torres, Elisa Sayrol, and Xavier Giro-i Nieto. 2017. Salgan: Visual saliency prediction with generative adversarial networks. *arXiv preprint arXiv:1701.01081* (2017).

[15] Junting Pan, Elisa Sayrol, Xavier Giro-i Nieto, Kevin McGuinness, and Noel E O'Connor. 2016. Shallow and deep convolutional networks for saliency prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 598–606.

[16] Yashas Rai, Jesús Gutiérrez, and Patrick Le Callet. 2017. A dataset of head and eye movements for 360 degree images. In *Proceedings of the 8th ACM on Multimedia Systems Conference*. 205–210.

[17] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*. Springer, 234–241.

[18] Yu-Chuan Su and Kristen Grauman. 2017. Learning spherical convolution for fast features from 360 imagery. In *Advances in Neural Information Processing Systems*. 529–539.

[19] Yu-Chuan Su and Kristen Grauman. 2017. Making 360 video watchable in 2d: Learning videography for click free viewing. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 1368–1376.

[20] Yu-Chuan Su and Kristen Grauman. 2019. Kernel transformer networks for compact spherical convolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 9442–9451.

[21] Tatsuya Suzuki and Takao Yamanaka. 2018. Saliency map estimation for omni-directional image considering prior distributions. In *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2079–2084.

[22] Qin Yang, Chenglin Li, Wenrui Dai, Junni Zou, GuoJun Qi, and Hongkai Xiong. 2020. Rotation Equivariant Graph Convolutional Network for Spherical Image Classification. In *CVPR 2020*. IEEE.

[23] Qin Yang, Junni Zou, Kexin Tang, Chenglin Li, and Hongkai Xiong. 2019. Single and Sequential Viewports Prediction for 360-Degree Video Streaming. In *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 1–5.

[24] Jianming Zhang and Stan Sclaroff. 2013. Saliency detection: A boolean map approach. In *Proceedings of the IEEE international conference on computer vision*. 153–160.

[25] Ziheng Zhang, Yanyu Xu, Jingyi Yu, and Shenghua Gao. 2018. Saliency detection in 360 videos. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 488–503.