Saliency Detection in 360° Videos

Ziheng Zhang, Yanyu Xu, Jingyi Yu, and Shenghua Gao(✉)

ShanghaiTech University, Shanghai, China
{zhangzh,xuyy2,yujingyi,gaoshh}@shanghaitech.edu.cn

Abstract. This paper presents a novel spherical convolutional neural network based scheme for saliency detection for 360° videos. Specifically, in our spherical convolution neural network definition, kernel is defined on a spherical crown, and the convolution involves the rotation of the kernel along the sphere. Considering that the 360° videos are usually stored with equirectangular panorama, we propose to implement the spherical convolution on panorama by stretching and rotating the kernel based on the location of patch to be convolved. Compared with existing spherical convolution, our definition has the parameter sharing property, which would greatly reduce the parameters to be learned. We further take the temporal coherence of the viewing process into consideration, and propose a sequential saliency detection by leveraging a spherical U-Net. To validate our approach, we construct a large-scale 360° videos saliency detection benchmark that consists of 104 360° videos viewed by 20+ human subjects. Comprehensive experiments validate the effectiveness of our spherical U-net for 360° video saliency detection.

Keywords: Spherical convolution · Video saliency detection
360° VR videos

1 Introduction

Visual attention prediction, commonly known as saliency detection, is the task of inferring the objects or regions that attract human's attention in a scene. It is an important way to mimic human's perception and has numerous applications in computer vision [1–3]. By far, almost all existing works focus on image or video saliency detection, where the participants are asked to look at images or videos with a limited field-of-view (FoV). And an eye tracker is adopted to record their eye fixations. The process, however, differs from the natural way human eyes perceive the 3D world: in real world, a participant actually **actively** explores the environment by rotating the head, seeking an omnidirectional understanding of the scene. In this paper, we propose to mimic this process by exploring the saliency detection problem on 360° videos.

Despite significant progresses in Convolutional Neural Networks (CNN) [4] for saliency detection in images/videos [5,6], there are very little, if few, studies on panoramic saliency detection. The brute-force approach of warping the

Z. Zhang and Y. Xu—equal contributions.

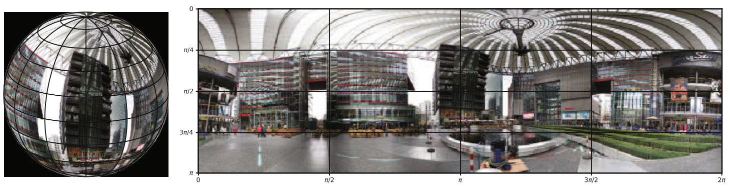


Fig. 1. Distortion introduced by equirectangular projection. **Left:** 360° image on sphere; **Right:** 360° image on equirectangular panorama

panoramic contents onto perspective views is neither efficient nor robust: partitioning the panoramas into smaller tiles and project the results using local perspective projection can lead to high computational overhead as saliency detection would need to be applied on each tile. Directly applying perspective based saliency detection onto the panorama images is also problematic: panoramic images exhibit geometric distortion where many useful saliency cues are not valid. Some latest approaches attempt to employ tailored convolutional networks for the spherical panoramic data. Yet, they either focus on coping with spherical data with radius components while ignoring distortions caused by equiangular cube-sphere representation [7], or dynamically stretching the kernels to fit the local content and therefore cannot achieve kernel parameter sharing [8]. Actually, when human explores the 360° contents, our brain uses the same mechanism to detect saliency when changing the view angle or FOV. In other words, if we leverage the CNN for 360° video saliency detection, the convolution operation corresponding to different view angle/FOV should maintain the same kernels.

To better cope the 360° video saliency detection task, we propose a new type of spherical convolutional neural networks. Specifically, we define the kernel on a spherical crown, and the convolution corresponds to the rotation of kernel on sphere. This definition has the parameter sharing property. Further, considering that the 360° videos are usually stored with equirectangular panorama, we propose to extend the spherical convolution to the panorama case by re-sampling the kernel based on the position of the patch to be convolved. We further propose a spherical mean square loss to compensate the distortion effect caused by equirectangular projection. In 360° videos, the participants search the environment continuously. This implies that the gaze in the previous frame affects the gaze in the subsequent frames. Then we propose to leverage such temporal coherence for efficient saliency detection by instantiating the spherical convolutional neural networks with a novel spherical U-Net [9]. Experiments validate the effectiveness of our scheme.

By far, nearly all saliency detection datasets are based on narrow FoV perspective images while only a few datasets on 360° images. To validate our approach, we construct a large-scale 360° videos saliency detection benchmark that consists of 104 360° videos viewed by 20+ human subjects. The duration of each video ranges from 20 s to 60 s. We use the a Glass eye tracker to track gaze. Figure 2 shows some example frames on several 360° videos in our dataset. We compare our spherical convolutional neural network with several state-of-the-

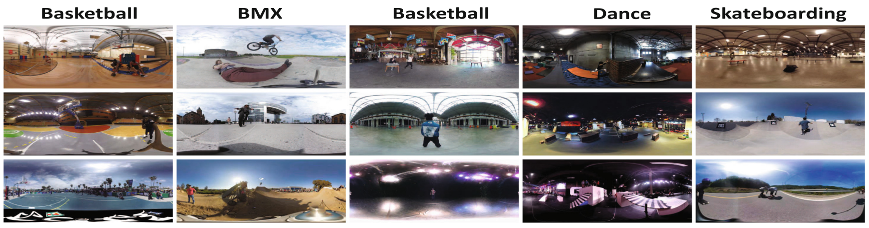


Fig. 2. The examples of five domains in our 360° video dataset

art techniques on this new data and show our technique significantly outperforms prior art in accuracy and robustness.

The contributions of this paper are summarized as follows: (i) A new type of spherical convolutional neural networks is defined, and the kernels are shared across all patches on the sphere. Thus our definition is more natural and useful for spherical saliency detection. We further extend it to panorama case; (ii) We propose a sequential saliency detection scheme and instantiate the spherical convolutional neural networks with a spherical U-net architecture for frame-wise saliency detection; (iii) We build a large-scale 360° video saliency detection dataset which would facilitate the evaluation of saliency detection in 360° videos. The dataset and code have been released to facilitate further research on 360° video saliency detection[1].

2 Related Work

2.1 Convolution Neural Networks for Spherical Data

Though CNN has demonstrated their effectiveness for many computer vision tasks [10,11], the data fed into traditional CNN are perspective images. To tackle the spherical data, methods in [12,13] firstly project a spherical image with equirectangular projection, then standard CNN is applied. However, such equirectangular projects introduce distortion in Fig. 1. The patches of the same size on sphere may correspond to regions of different shapes based on their coordinates (θ, ϕ). So the standard convolution with the shared kernels is no longer perceptually meaningful. To solve this problem, [8] propose to stretch kernels in standard CNN to fit the shape of the patch on the equirectangular plane in convolution. This can avoid the distortion problem to some extent. But the filters shape in their solution depends on the longitude of the patch on the sphere, consequently, the kernels in their method are not shared, which introduces the expensive computational and storage costs. Further, Boomsma et al.[7] propose to adopt the equiangular cubed-sphere representation for sphere data representation, then concentric cubed-sphere convolution is applied. But their solution is proposed for sphere data with radii components (like a solid ball), which differs from the data type of ours. Besides, equiangular cubed-sphere representation still

[1] GitHub: https://github.com/xuyanyu-shh/Saliency-detection-in-360-video.

introduce distortion in each facet, which damages to the accuracy of convolution. Different from these work, in [14,15], the spherical image is repeatedly projected to tangent planes at all locations, and the convolution is conducted on these planes. Although such solution improves accuracy, it also brings expensive computational costs. Further, the disjoint projection planes make the intermediate representation cannot be shared for higher layer convolution. Recently, Cohen et al.[16] propose a new type of Spherical CNN on SO(3) manifold, and their solution is expressive and rotation-equivariant. With Fast Fourier Transform, their solution can be greatly accelerated. However, the concept of SO(3) CNN is not so adhere to our intuition to process 2D spherical images and quite distincts from the concept of planner CNN.

Though many CNN models have been proposed for spherical data, none are customized for 360° videos. Actually, when we change the FOV in 360° videos, our brain actually uses the same mechanism for environment exploration. In other words, the kernels used for saliency detection should be shared across all views. This motivates us to design a new type of spherical CNN: we define kernels with the spherical crown shape, we rotate and convolve the kernel with patches on the sphere-polar coordinate system. [2] In this way, the kernel can be shared. So our solution is more natural and more interpretable for saliency detection in 360° videos.

2.2 Video Saliency Detection

Many efforts have been done to study the video saliency detection, either hand-crafted features based methods [17–20], or deep learning based methods [6,21–24], yet the study of video saliency detection in 360° videos is still at its primitive stage. [12,25] are two pioneer work along this direction, but the 360° data used in these work are static images. Actually, the videos with dynamic contents are more common in real applications. To understand the behavior of human in dynamic 360° videos, especially 360° sports videos, Hu et al.propose to predict the salient objects by feeding projected panorama images into CNN directly. However, the distortion of projection is not considered, which would reduce the accuracy. In addition, the salient objects are manually labeled, which does not necessarily reflect the real behavior of human visual attention. In order to better understand the users' behavior in VR scenes, we propose to study the eye fixation in 360° videos. To the best of our knowledge, this is the first work that works on eye fixation prediction in 360° videos. We also build a dataset to facilitate the evaluation of our work.

[2] Since the sphere image is usually stored with planar format, we actually project spherical image on the Euclidean plane with equirectangular projection, then we re-sample kernel based on the shape of the projected patch to be convolved, after that we convolved the target patch with the transformed kernel.

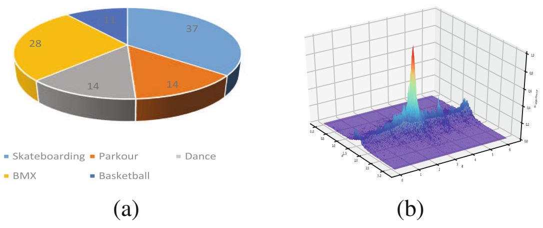


Fig. 3. Dataset Analysis: (a) the distribution of the five sports domains based on the number of videos; (b) the distribution of eye fixations on equirectangular panorama. (Best viewed in color)

3 360° Dynamic Video Saliency Detection Dataset

3.1 Data Collection

We collect the 360° videos from Sports-360 dataset [13], and remove the video clips whose length is less than 20 s[3], and use the remaining 104 video clips as the data used for saliency detection in 360° videos. The video contents involve five sports (*i.e.* basketball, parkour, BMX, skateboarding, and dance), and the duration of each video is between 20 and 60 s. Figure 3(a) shows the distribution of five sports videos. Then we display the videos with an HTC VIVE HMD and a '*7invensun a-Glass*' eye tracker is embedded into the HMD to record the eye fixation points of the participants when they watching videos.

We recruit 27 volunteers (between 20 and 24 years) to participant in the experiments. All 104 videos are divided into 3 sessions and each session contains 35 360° videos. Volunteers are asked to watch 360° videos at a fixed starting location ($\theta = 90, \phi = 180$) in random orders. We set a shorter break (20 s) between 2 videos and a longer break (3 min) after watching 15 videos. We also calibrate the system after the long break. In the end, each video is watched by at least 20 volunteers. The total time used for data collection is about 2000 min.

Distribution of Gaze Points Figure 3 (b) shows the distribution of all eye fixation angle in θ, ϕ of all participants over all the videos on equirectangular panorama. The peak in the center of panorama ($\theta = 90, \phi = 180$) is because all participants explore the environment with the fixed starting location. Further, we can see that the eye fixation points mostly centered around the equator, which means the volunteers tend to explore the environment along the horizontal direction, and they seldom look down/up. There are almost no eye fixation points around north/south pole.

[3] We only use videos longer than 20 s rather than entire Sports-360 dataset because in [12] Sitzmann et al.evaluate the exploration time for a given still scene, and show that 'on average, users fully explored each scene after about 19 s'.

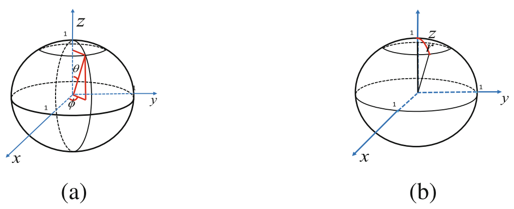


Fig. 4. (a) Spherical coordinate system: ϕ is the angle between X axis and the orthogonal projection of the line on the XOY plane, and θ is the angle between the Z axis and the line; (b) Spherical crown kernel: the red line represents the radius r. (Best viewed in color)

4 Spherical Convolutional Neural Networks

In this section, we introduce our spherical convolution on sphere and its extension on panorama. A new spherical mean square error (sphere MSE) loss is also introduced for spherical convolution on panorama. Note that the convolution operation in deep learning usually refers to correlation in math.

The spherical convolution. Spherical convolution is an operation on feature map f and kernel k on sphere manifold S^2. S^2 is defined as the set of points $x \in \mathbb{R}^3$ with norm 1, which can be parameterized by spherical coordinates $\theta \in [0, \pi]$ and $\phi \in [0, 2\pi]$ as shown in Fig. 4(a). To simplify the the notation, here we model spherical images and filters as continuous functions $f : S^2 \to \mathbb{R}^K$, where K is the number of channels. Then the spherical convolution is formulated as [26]:

$$[f * k](x) = \int_{S^2} f(Rn)k(R^{-1}x)dR \tag{1}$$

where n is the north pole and R is rotations on sphere represented by 3×3 matrices.

In this paper, filter k only have non-zero values in spherical crown centered at north pole, whose size is parameterized by r_k, which corresponds to orthodromic distance between north pole and borderline of the crown as shown in Fig. 4 (b). So the radius r_k controls the number of parameters in kernel k and the size of local receptive field. Larger r_k means there are more parameters in k and the local receptive field is larger.

Convolution on equirectangular panorama. The spherical images or videos are usually stored as 2-D panorama through equirectangular projection represented by (θ, ϕ) ($\theta \in [0, \pi]$ and $\phi \in [0, 2\pi]$). So we extend Eq. (1) to the convolution between feature map f and kernel k on projected panorama as

$$[f * k](\theta, \phi) = \iint f(\theta', \phi')k(\theta' - \theta, \phi' - \phi) \sin\theta' d\theta' d\phi' \tag{2}$$

There are several differences between spherical convolution on equirectangular and convolution on perspective image. Firstly, we denote the points set for the kernel centered at (θ_0, ϕ_0) as $\mathcal{D}_k(\theta_0, \phi_0) = \{(\theta, \phi) | g(\theta, \phi, \theta_0, \phi_0) \leq 0\}$, where $g(\theta, \phi, \theta_0, \phi_0) \leq 0$ corresponds to the equation of sphere crown (kernel) centered at (θ_0, ϕ_0). The behavior of $\mathcal{D}_k$ is different from that in standard convolution for perspective image. Specifically, when we move the kernel and when its center is $(\theta_0 + \Delta_\theta, \phi_0 + \Delta_\phi)$, the points set for the moved kernel cannot be directly obtained by simply moving the $\mathcal{D}_k(\theta_0, \phi_0)$ by $(\Delta_\theta, \Delta_\phi)$, which can be mathematically written as follows:

$$\begin{aligned} \mathcal{D}_k(\theta_0 + \Delta_\theta, \phi_0 + \Delta_\phi) &= \{(\theta, \phi) | g(\theta - \Delta_\theta, \phi - \Delta_\phi, \theta_0, \phi_0) \leq 0\} \\ &\neq \{(\theta + \Delta_\theta, \phi + \Delta_\phi) | g(\theta, \phi, \theta_0, \phi_0) \leq 0\} \end{aligned} \tag{3}$$

Secondly, there is a $\sin\theta'$ term in the integrand in spherical convolution in Eq. (2), which causes the different behavior of spherical convolution compared with the convolution for projective images. Thirdly, there does not exist padding in spherical convolution in Eq. (1) owing to the omnibearing view of 360° images. But, it indeed needs padding in convolution on equirectangular panorama in Eq. (2), owing to the storage format. The padding needed here is circle shift. For example, when kernel locates at the far left region, it needs to borrow some pixels from far right region for computing convolution. To simplify the notation, we also term the convolution on equirectangular panorama as spherical convolution. In this way, we can implement convolution on sphere by using convolution on equirectangular panorama.

We define sample rate on equirectangular panorama as the number of pixels per rad. So sample rate of panorama is $sr^\theta = H/\pi, sr^\phi = W/2\pi$ along θ and ϕ direction, respectively. Here H, W is the height and width of panorama. As a special case, for the kernel with radius equalling r_k, when the kernel is centered at north pole, its projection on equirectangular panorama would be a rectangular, whose size is denoted as $W_k \times H_k$, and the sample rate along θ and ϕ direction are given by $sr_k^\theta = H_k/r_k, sr_k^\phi = W_k/2\pi$.

Implement Details. For discrete spherical convolution on panorama, we set the kernel parameters to be learnt as the equirectangular projection of kernel centered at the north pole ($\theta \leq r_k$). So the kernel projected on equirectangular panorama corresponds to a rectangular of size $W_k \times H_k$. It is worth noting we can also set the kernels at other positions rather than north pole, yet sample grid of the kernel will change accordingly. The discrete spherical convolution on equirectangular panorama includes the following steps: determine non-zero region of kernel centered at (θ_i, ϕ_i) and obtain the set of points fallen into the kernel area $\mathcal{D}_k(\theta_i, \phi_i)$, rotate those points back to $\mathcal{D}_k(0, 0)$, re-sample the original kernel to find values for each sampling points in $\mathcal{D}_k(\theta_i, \phi_i)$.

Determine the points fallen into $\mathcal{D}_k(\theta_i, \phi_i)$. For the spherical crown kernel with radius r_k centered at (θ_i, ϕ_i), the points fallen into this kernel area with coordinates (θ, ϕ) satisfy the following equation:

$$\sin\theta_i \cos\phi_i \sin\theta \cos\phi + \sin\theta_i \sin\phi_i \sin\theta \sin\phi + \cos\theta_i \cos\theta = \cos\theta_k \tag{4}$$

which can be simplified as

$$\sin(\phi + \psi) = C \tag{5}$$

Algorithm 1. Obtain the set of grid points on the kernel area

```
Input: kernel radius $r_k$, kernel location $\theta_k, \phi_k$.
Output: The set of grid points on the kernel area $S$
 1: $S \leftarrow \emptyset$
 2: Calculate range of $\Theta$:
 3: $\Theta \in [\max(0, \theta_k - r_k), \min(\theta_k + r_k, \pi)]$
 4: Calculate range of $\Phi$ for each $\theta \in \Theta$:
 5: for Each $\theta \in \Theta$ do
 6:     Find the solution of equation Eq. 5
 7:     if There exists infinitely many solutions then
 8:         $\phi \in [0, \pi]$
 9:     else if There exists two solutions $\phi_1 < \phi_2$ then
10:         $\phi \in [\phi_1, \phi_2]$
11:     else if There exists no solutions then
12:         No grid points $(\theta, \phi)$ on the kernel area
13:     end if
14:     Add $(\theta, \phi)$ to $S$.
15: end for
```

where $\sin\psi = \frac{a}{\sqrt{a^2+b^2}}$, $\cos\psi = \frac{b}{\sqrt{a^2+b^2}}$, $C = \frac{d-c}{\sqrt{a^2+b^2}}$ and $a = \sin\theta_i \cos\phi_i \sin\theta$, $b = \sin\theta_i \sin\phi_i \sin\theta$, $c = \cos\theta_i \cos\theta$ and $d = \cos\theta_k$.

Once the kernel area on sphere is determined, we can sample the corresponding points on panorama to obtain the points needed for convolution. We list the main steps for such stage in Algorithm 1.

Rotate the set of Sampled Points Back to North Pole. Now that we have the sampled points of current kernel area, we also need to determine the correspondences between them to those kernel values centered at the north pole. To do this, we rotate these points back to original kernel region by leveraging matrix multiplication between their cartesian coordinate representations and rotation matrix along Y as well as Z axis. Note that after rotation, the sampled points might lie between original kernel points centered at the north pole.

Re-sample the Original Kernel. In order to obtain kernel values for sampled points that lie between original kernel points centered at the north pole, we use grid sampling technique as used in Spatial Transform Network [27], which is basically a general interpolation method for such re-sampling problem on 2D image. The third row in Fig. 5 shows the sampling grid corresponding to kernel located at $\theta = 0, \pi/4, \pi/2, 3\pi/4, \pi$ and $\phi = \pi$.

Finally, the result of spherical convolution is given by sum of element-wise multiplication between re-sampled kernel points and corresponding panorama points, divided by the total number of re-sampled kernel points.

Properties of spherical convolution. The sphere convolution has the following three properties: sparse interactions, parameter sharing, and equivariant representations.

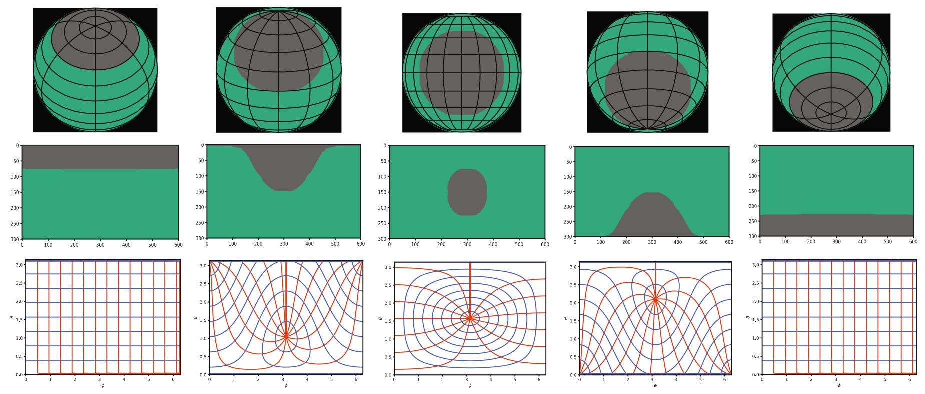


Fig. 5. Parameter sharing. This figure indicates how spherical crown kernel changes on sphere and projected panorama from north pole to south pole with angle interval equalling $\pi/4$. The first raw is the region of the spherical crown kernel in sphere. The second raw shows the region of spherical crown kernel in the projected panorama. The third row shows sampling grid corresponding to each kernel location. Red curve represents θ sampling grid and blue curve represents ϕ sampling grid

- **Sparse interactions.** Standard CNNs usually have sparse interactions, which is accomplished by making the kernel smaller than the input. Our proposed spherical CNN also have this important property. Such sparse connection greatly reduces the number of parameters to be learned. Further, the convolution in higher layers corresponds to gradually larger local receptive field, which allows the network to efficiently model the complicated interactions between input and output.
- **Parameter sharing.** Similar to the standard convolution for perspective image, the parameters of spherical crown kernel is the same everywhere on the sphere, which means the kernel is shared. This would greatly reduce the storage requirements of the model as well as the number of parameters to be learned. Kernel at different location is re-sampled as shown in Fig. 5.
- **Equivariant representations.** In both standard convolutions for perspective image and spherical convolution, the parameter sharing makes the layers with equivariance to translation property, which means if the input changes, the output changes in the same way.

4.1 Spherical Mean Squared Error (MSE) Loss

Mean Squared Error (MSE) loss function is widely used in perspective image based CNN. However, the standard MSE is designed for perspective image. For perspective image, discretization is performed homogeneously in space, which differs from the case of panorama. To adopt MSE loss to panorama, we weight the square error for each pixel on panorama with it's solid angle on sphere before average them.

We define the solid angle in steradian equals the area of a segment of a unit sphere in the same way a planar angle in radians equals the length of an arc of

Table 1. The architecture of the CNN

Layer	Operation	Input size	Input source	Output size	Kernel radius	Kernel size
0	Input	-	-	$3 \times 150 \times 300$	-	-
1	Spherical Conv	$3 \times 150 \times 300$	Layer 0	$64 \times 150 \times 300$	$\pi/32$	(8,16)
2	Max pooling	$64 \times 150 \times 300$	Layer 1	$64 \times 75 \times 150$	$\pi/32$	(8,16)
3	Spherical conv	$64 \times 75 \times 150$	Layer 2	$128 \times 75 \times 150$	$\pi/16$	(4,8)
4	Max pooling	$128 \times 75 \times 150$	Layer 3	$128 \times 38 \times 75$	$\pi/16$	(4,8)
5	Spherical conv	$128 \times 38 \times 75$	Layer 4	$256 \times 38 \times 75$	$\pi/4$	(4,8)
6	Max pooling	$256 \times 38 \times 75$	Layer 5	$256 \times 19 \times 38$	$\pi/4$	(4,8)
7	Spherical conv	$256 \times 19 \times 38$	Layer 6	$256 \times 19 \times 38$	π	(8,16)
8	Up-sampling	$256 \times 19 \times 38$	Layer 7	$256 \times 38 \times 75$	$\pi/4$	(4,8)
9	Spherical conv	$(256 + 256) \times 38 \times 75$	Layer 8 & 6	$128 \times 38 \times 75$	$\pi/4$	(4,8)
10	Up-sampling	$128 \times 38 \times 75$	Layer 9	$128 \times 75 \times 150$	$\pi/16$	(4,8)
11	Spherical conv	$(128 + 128) \times 75 \times 150$	Layer 10 & 5	$64 \times 75 \times 150$	$\pi/16$	(4,8)
12	Up-sampling	$64 \times 75 \times 150$	Layer 11	$64 \times 150 \times 300$	$\pi/32$	(4,8)
13	Spherical conv	$(64 + 64) \times 150 \times 300$	Layer 12 & 3	$1 \times 150 \times 300$	$\pi/32$	(4,8)

a unit circle, which is the following ratio: $\Omega = A/r^2$, where A is the spherical surface area and r is the radius of the considered sphere. Thus, for a unit sphere, the solid angle of the spherical crown with radius r and centered at the north pole is given as: $\Omega = 2\pi(1 - \cos r)$. It is desirable that the image patch corresponding to the same solid angle would have the same weight for sphere MSE, thus we arrive at the following objective function:

$$\mathcal{L} = \frac{1}{n} \sum_{k=1}^{n} \sum_{\theta=0,\phi=0}^{\Theta,\Phi} w_{\theta,\phi} (S_{\theta,\phi}^{(k)} - \hat{S}_{\theta,\phi}^{(k)})^2 \tag{6}$$

where $\hat{S}^{(k)}$ and $S^{(k)}$ are the ground truth saliency map and predicted saliency map for the k^{th} image, and $w_{\theta,\phi}$ is weights for each points which is proportional to its solid angle and $w_{i,j} \propto \Omega(\theta, \phi)$. $\Omega(\theta, \phi)$ is solid angle corresponding to the sampled area on saliency map located at (θ, ϕ). In our implementation, we just set $w_{\theta,\phi} = \Omega(\theta, \phi)/4\pi$ (4π is the solid angle of the unit ball).

5 Spherical U-Net Based 360° Video Saliency Detection

5.1 Problem Formulation

Given a sequence of frames $V = \{v_1, v_2, \ldots, v_T\}$, our goal is to predict the saliency map corresponding to this video $S = \{s_2, s_2, s_3, \ldots, s_T\}$. So deep learning based 360° video saliency detection aims at learning a mapping G that maps input V to S. However, different from perspective videos whose saliency merely depends on the video contents, where a participant look at in 360° videos also depends on the starting position of the participant. We define s_0 as the eye fixation map at the starting position, which is the saliency map corresponding to the starting point, then the video saliency detection can be formulated as

$$G^* = \arg\min_F \|S - G(V, s_0)\|_F^2 \tag{7}$$

In practice, we can initialize s_0 with a Gaussian kernel centered at the staring position. Further, the participants actually watch the 360° video in a frame-by-frame manner, and the eye fixation in previous frame helps the prediction of that in next frame. So we can encode such prior into the objective and arrive at the following objective :

$$F^* = \arg\min_F \sum_{t=1}^{T} \|s_t - F(v_t, s_{t-1})\|^2 \tag{8}$$

Here F is the prediction function which takes the saliency map of previous frame and video frame at current moment as input for saliency prediction of current moment. Inspired by the success of U-Net [9] we propose to adapted it with a Spherical U-Net as F for the frame-wise saliency detection.

5.2 Spherical U-Net

The network architecture of Spherical U-Net is listed in Table 1. The input to the network is projected spherical images v_t at time t and project spherical saliency map s_{t-1} at time $t-1$. Similar to U-net [9], our spherical U-net also consists of a contracting path (left side) and an expansive path (right side). In the contracting path, there are three spherical convolutional layers followed by rectified linear unit (ReLU) and a 2×2 spherical max pooling is adopted to down-sample the data. The expansive path consists of three spherical convolutional layers followed by ReLU and up-sampling. For the last three spherical convolutional layer, their inputs are concatenation of the outputs of their previous layer and the corresponding layer with the same output size from the contracting path. A spherical convolution ins final layer is used to map each feature vector to a saliency map. In total the network has 7 spherical convolutional layers.

6 Experiments

6.1 Experimental Setup

We implement our spherical U-Net with the PyTorch framework. We train our network with the following hyper-parameters setting: mini-batch size (32), learning rate (3e-1), momentum (0.9), weight decay (0.00005), and number of iterations (4000).

Datasets. We evaluate our proposed spherical CNN model on both Salient360! dtaset and our video saliency dataset. Salient360! [28] is comprised of 98 different 360° images viewed by 63 observers. Our video saliency dataset consists of 104 360° videos viewed by 20 observers. For Salient360, we follow the standard training/testing split provided in [28]. For the experiments on our dataset, 80 videos are randomly selected as training data, and the remaining 24 videos are used for testing. All baseline methods are compared based on the same training/testing split. For image saliency, we regress the saliency maps directly from the RGB 360° images.

Metrics. We create the ground truth saliency maps through a way similar to spherical convolution using a crown Gaussian kernel with sigma equaling to 3.34°. Owing to the distortion during projection, it does not make sense to directly compare two panorama saliency maps like typical 2D saliency maps. Therefore, we utilize the metrics including CC, AUC-judd, and NSS introduced in [28] to measure the errors between the predicted saliency maps and the ground truth.

Baselines. We compare our proposed spherical U-Net with the following state-of-the-art: image saliency detection methods, including LDS [29], Sal-Net [5] and SALICON [30], video saliency detection methods, including GBVS [31] and a more recent dynamic saliency [32] and the 360° image saliency models [33]. Of all these methods, Sal-Net and SALICON are deep learning based methods, and we retrain the models with panorama images on the datasets used in our paper for performance comparison. We also design the following baselines to validate the effectiveness of our spherical model.

- *Standard U-Net.* Compared with our spherical U-Net method, the CNN and MSE loss in such baseline is conventional CNN and standard MSE loss.
- *Spherical U-Net w.o. sal.* Compared with our spherical U-Net method, the only difference is that the saliency of previous frame is not considered for the saliency prediction of current frame.

In addition, we measure human performance by following the strategies in [34]:

- *Baseline-one human*: It measures the difference between the saliency map viewed by one observer and the average saliency map viewed by the rest observers.
- *Baseline-infinite humans*: It measures the difference between the average saliency map viewed by a subset of viewers and the average saliency map viewed by the remaining observers.

Recent work has employed several top-down cues for saliency detection. Previous work [35] shows that human face boosts saliency detection. Therefore, we also design a baseline *Top-down cue (face)* to use human face as cue, and post-process saliency map following [35].

6.2 Performance Evaluation

We compare our method with all baselines in Table 2. It shows that our method outperforms all baseline methods on our video saliency dataset, which validates the effectiveness of our scheme for 360° video saliency detection.

In order to show the rotation equivariant in θ and ϕ direction, we rotate 60°, 120° along ϕ direction and 30°, 60° along θ direction on test data, and do data augmentation by rotating random degree on both direction on training set. The results are shown in Fig. 7. We can see that compared to rotation in ϕ direction,

Table 2. The performance comparison of state-of-the-art methods with our spherical U-Net on our video saliency dataset

	LDS [29]	Sal-Net [5]	SALICON [30]	GBVS [31]	Wang *et al.* [32]	SaltiNet [33]	Spherical U-Net w.o. sal	Standard U-Net	Top-down cue (face)	Ours	Baseline-one Human	Baseline-infinite Humans
CC	0.2727	0.2404	0.2171	0.1254	0.2929	0.2582	0.3716	0.2457	0.6250	0.6246	0.7641	0.7035
NSS	1.6589	1.3958	1.3178	0.8003	1.5869	1.4470	2.2050	1.3034	3.5339	3.5340	5.4339	5.6504
AUC-judd	0.8169	0.8266	0.8074	0.7799	0.7906	0.8579	0.8464	0.8300	0.8985	0.8977	0.7585	0.8634

that in θ direction slightly changes because of the change of sample grid when rotating along θ direction.

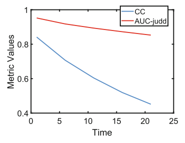
To evaluate the performance of our method on the Salient360! dataset, we have modified our model to directly predict the saliency map for a static 360° image. In Fig. 6, we can see that our method outperforms all baselines, which validates the effectiveness of our method for static 360° image.

6.3 Evaluation of Different Components in Spherical U-Net

We conduct ablation study by replacing the spherical CNN with standard CNN (*Ours w. standard conv*) and replacing spherical MSE with standard MSE (*Ours w. standard MSE*), respectively. The performance of these baselines is listed in table in Fig. 6. We can see that both spherical CNN and Spherical MSE contributes the performance.

Methods	CC	AUC-judd	NSS
LDS [29]	0.3134	0.6186	0.6703
Sal-Net [5]	0.2998	0.6397	0.6792
SALICON [30]	0.3233	0.6511	0.6918
Ours	0.4087	0.6594	0.6989

Baselines	CC	AUC-judd	NSS
Ours w. standard MSE	0.6189	3.1620	0.8685
Ours w. standard conv	0.2877	1.6145	0.8520
Ours w. smaller kernel	0.6023	3.219	0.8593
Spherical pooling	0.6253	3.5333	0.8980
Our Spherical U-Net	0.6246	3.5340	0.8977



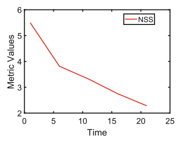


Fig. 6. The First: The performance comparison of state-of-the-art methods with our spherical CNN on Salient360 [36] dataset. **The Second:** The performance comparison of different components on our method on our video saliency dataset. **The Third** and **The Forth:** the performance of saliency map prediction for a longer time based on CC, AUC-judd, and NSS metrics respectively

We also evaluate spherical U-Net with different spherical kernel sizes and the comparison with spherical U-Net with smaller kernel sizes than our spherical U-Net (*Ours w. smaller kernel*) is shown in table in Fig. 6. We can see that larger kernel leads to better performance. One possible reason is that a larger spherical kernel could involve more parameters, which could increase the capability of our network. Another reason is that larger kernel increases kernel sample rate, which might improve the accuracy when re-sample the kernel.

6.4 Spherical Pooling

We do comparison between planner pooling and spherical pooling in the table in Fig. 6. In this paper, spherical pooling could be regarded as a special spherical convolution, similar to the relationship between planner ones. The spherical pooling outperforms planner pooling, responsible for consistency between receptive field of kernel with spherical feature maps. To note that, [16] also uses planner (3D) pooling to downsample feature maps. Since planner pooling achieves similar performance as spherical pooling and has lower computational cost, following [16], currently we use planner pooling.

6.5 Saliency Prediction for a Longer Time

Middle and right figures in Fig. 6 show the results when our model predicts saliency maps for a longer time based on CC, NSS, and AUC-judd metric. We can see that the performance of saliency prediction degenerates for as time elapse. One possible reason is that as time goes longer, the prediction of previous frame becomes less accurate, which consequently would affect the saliency detection of current frame.

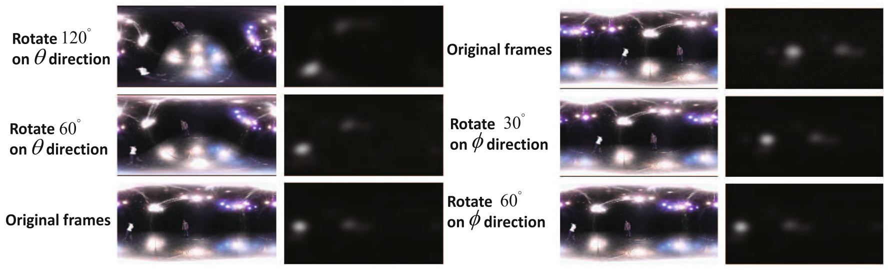

Rotate 120° on θ direction
Rotate 60° on θ direction
Original frames
Original frames
Rotate 30° on ϕ direction
Rotate 60° on ϕ direction

Fig. 7. The rotation equivariant in θ and ϕ direction: The first and third columns are rotated frames and the second and forth columns are our predictions

6.6 Time and Memory Costs

Our model is trained on four NVIDIA Tesla P40 GPUs. We calculate the average running time for each image batch. The average running time of our model is 5.1 s/iter. The spherical U-Net listed in Table 1 has about 6.07 M parameters, and it consumes 21×4 GB of memory when batch size is 32 when training. It takes about 36 h to train the model on the our video saliency dataset (the total number of iterations is 4000.).

7 Conclusion and Discussions

Our work attempts to exploit the saliency detection in dynamic 360° videos. To this end, we introduce a new type of spherical CNN where the kernels are shared across all image patches on the sphere. Considering that the 360° videos are stored with panorama, we extent spherical CNN to the panorama case, and we propose to re-sample kernel based on its location for spherical convolution on panorama. Then we propose a spherical U-Net for 360° video saliency detection. We also build a large-scale 360° video saliency dataset for performance evaluation. Extensive experiments validate the effectiveness of our proposed method. It is worth noting our spherical CNN is a general framework, it can also be applied to other tasks involving 360° video/image.

There still exists some space to improve our method for video saliency prediction. Currently, to simplify the problem, we only consider the saliency map of the previous frame for the prediction of current frame. Considering the saliency map over a longer time range may boost the performance, for example, we can also combine our spherical U-Net with LSTM. The combination of spherical CNN with other types of deep neural network is beyond the study scope of this paper, and we will leave them for future work.

Acknowledgement. This project is supported by the NSFC (No. 61502304).

References

1. Itti, L.: Automatic foveation for video compression using a neurobiological model of visual attention. IEEE Trans. Image Process. **13**(10), 1304–1318 (2004)
2. Setlur, V., Takagi, S., Raskar, R., Gleicher, M., Gooch, B.: Automatic image retargeting. In: Proceedings of the 4th International Conference on Mobile and Ubiquitous Multimedia, pp. 59–68. ACM (2005)
3. Chang, M.M.L., Ong, S.K., Nee, A.Y.C.: Automatic information positioning scheme in ar-assisted maintenance based on visual saliency. In: De Paolis, L.T., Mongelli, A. (eds.) AVR 2016. LNCS, vol. 9768, pp. 453–462. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-40621-3_33
4. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
5. Pan, J., Sayrol, E., Giroinieto, X., Mcguinness, K., Oconnor, N.E.: Shallow and deep convolutional networks for saliency prediction, pp. 598–606 (2016)
6. Bazzani, L., Larochelle, H., Torresani, L.: Recurrent mixture density network for spatiotemporal visual attention. arXiv preprint arXiv:1603.08199 (2016)
7. Boomsma, W., Frellsen, J.: Spherical convolutions and their application in molecular modelling. In: Advances in Neural Information Processing Systems, pp. 3436–3446 (2017)
8. Su, Y.C., Grauman, K.: Learning spherical convolution for fast features from 360 imagery. In: Advances in Neural Information Processing Systems, pp. 529–539 (2017)
9. Ronneberger, O., Fischer, P., Brox, T.: U-Net: convolutional networks for biomedical image segmentation. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (eds.) MICCAI 2015. LNCS, vol. 9351, pp. 234–241. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24574-4_28
10. Schroff, F., Kalenichenko, D., Philbin, J.: FaceNet: a unified embedding for face recognition and clustering. In: Proceedings of the IEEE Conference on Computer Vision And Pattern Recognition, pp. 815–823 (2015)
11. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems, pp. 91–99 (2015)

12. Sitzmann, V., Serrano, A., Pavel, A., Agrawala, M., Gutierrez, D., Wetzstein, G.: Saliency in VR: how do people explore virtual environments? (2016)
13. Hu, H.N., Lin, Y.C., Liu, M.Y., Cheng, H.T., Chang, Y.J., Sun, M.: Deep 360 pilot: learning a deep agent for piloting through 360deg sports video (2017)
14. Su, Y.C., Grauman, K.: Making 360 video watchable in 2D: learning videography for click free viewing. arXiv preprint (2017)
15. Su, Y.C., Jayaraman, D., Grauman, K.: Pano2vid: automatic cinematography for watching 360° videos. In: Asian Conference on Computer Vision, pp. 154–171 (2016)
16. Cohen, T.S., Geiger, M., Koehler, J., Welling, M.: Spherical CNNs. arXiv preprint arXiv:1801.10130 (2018)
17. Zhong, S.h., Liu, Y., Ren, F., Zhang, J., Ren, T.: Video saliency detection via dynamic consistent spatio-temporal attention modelling. In: AAAI, pp. 1063–1069 (2013)
18. Zhou, F., Bing Kang, S., Cohen, M.F.: Time-mapping using space-time saliency. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3358–3365 (2014)
19. Itti, L., Dhavale, N., Pighin, F.: Realistic avatar eye and head animation using a neurobiological model of visual attention. In: Applications and Science of Neural Networks, Fuzzy Systems, and Evolutionary Computation VI, vol. 5200, pp. pp. 64–79. International Society for Optics and Photonics (2003)
20. Ren, Z., Gao, S., Chia, L.T., Rajan, D.: Regularized feature reconstruction for spatio-temporal saliency detection. IEEE Trans. Image Process. **22**(8), 3120–3132 (2013)
21. Bak, Ç., Erdem, A., Erdem, E.: Two-stream convolutional networks for dynamic saliency prediction. arXiv preprint arXiv:1607.04730 (2016)
22. Wang, W., Shen, J., Shao, L.: Video salient object detection via fully convolutional networks. IEEE Trans. Image Process. **27**(1), 38–49 (2018)
23. Chaabouni, S., Benois-Pineau, J., Hadar, O., Amar, C.B.: Deep learning for saliency prediction in natural video (2016)
24. Liu, Y., Zhang, S., Xu, M., He, X.: Predicting salient face in multipleface videos. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4420–4428 (2017)
25. Ruhland, K., et al.: A review of eye gaze in virtual agents, social robotics and HCI: Behaviour generation, user interaction and perception. Comput. Graph. Forum **34**(6), 299–326 (2015)
26. Driscoll, J.R., Healy, D.M.: Computing fourier transforms and convolutions on the 2-sphere. Adv. Appl. Math. **15**(2), 202–250 (1994)
27. Jaderberg, M., Simonyan, K., Zisserman, A., et al.: Spatial transformer networks. In: Advances in Neural Information Processing Systems, pp. 2017–2025 (2015)
28. Rai, Y., Gutiérrez, J., Le Callet, P.: A dataset of head and eye movements for 360 degree images. In: Proceedings of the 8th ACM on Multimedia Systems Conference, pp. 205–210. ACM (2017)
29. Fang, S., Li, J., Tian, Y., Huang, T., Chen, X.: Learning discriminative subspaces on random contrasts for image saliency analysis. IEEE Trans. Neural Netw. Learn. Syst. **28**(5), 1095–1108 (2017)
30. Huang, X., Shen, C., Boix, X., Zhao, Q.: SALICON: reducing the semantic gap in saliency prediction by adapting deep neural networks. In: IEEE International Conference on Computer Vision, pp. 262–270 (2015)
31. Harel, J., Koch, C., Perona, P.: Graph-based visual saliency. In: Advances in Neural Information Processing Systems, pp. 545–552 (2007)

32. Wang, W., Shen, J., Guo, F., Cheng, M.M., Borji, A.: Revisiting video saliency: a large-scale benchmark and a new model (2018)
33. Assens, M., Mcguinness, K., Giroinieto, X., O'Connor, N.E.: SaltiNet: scan-path prediction on 360 degree images using saliency volumes (2017)
34. Judd, T., Durand, F., Torralba, A.: A benchmark of computational models of saliency to predict human fixations. In: MIT Technical report (2012)
35. Goferman, S., Zelnikmanor, L., Tal, A.: Context-aware saliency detection. IEEE Trans. Pattern Anal. Mach. Intell. **34**(10), 1915 (2012)
36. Rai, Y., Callet, P.L.: A dataset of head and eye movements for 360 degree images. In: ACM on Multimedia Systems Conference, pp. 205–210 (2017)