A Benchmark of Four Methods for Generating 360° Saliency Maps from Eye Tracking Data

Brendan John*, Pallavi Raiturkar*, Olivier Le Meur†, Eakta Jain*
*University of Florida, Gainesville, US
†Univ Rennes, CNRS, IRISA, France

***Abstract*—Modeling and visualization of user attention in Virtual Reality is important for many applications, such as gaze prediction, robotics, retargeting, video compression, and rendering. Several methods have been proposed to model eye tracking data as saliency maps. We benchmark the performance of four such methods for 360° images. We provide a comprehensive analysis and implementations of these methods to assist researchers and practitioners. Finally, we make recommendations based on our benchmark analyses and the ease of implementation.**

***Index Terms*—Saliency, Visualization, Eye movements in VR, 360 images**

I. Introduction

With the explosive growth of commercial VR and AR systems, there is increased access to innovative 3D experiences. In particular, 360° images and videos are being used for entertainment, storytelling, and advertising. Content creators are actively working out techniques and building tools to guide user attention in these new media. A critical enabler for these efforts is measuring and visualizing eye tracking data. Eye trackers built into VR headsets serve as a reliable tool to understand how attention is allocated in 3D environments.

The study of attention and eye movements in 2D content is well established. Saliency maps highlight regions that attract the most visual attention and have applications in predicting gaze [1], compression [2], and selective rendering [3] to name a few. In 360° images, the user is surrounded by a photo-realistic virtual scene. Because only a fraction of the scene is viewed at once, the allocation of visual attention is different than 2D content. Moreover, due to the spherical nature of 360° content, novel saliency map generation methods are required.

To generate 2D saliency maps, eye tracking data is processed to identify fixations. Fixations are aggregated in a map that is convolved with a 2D Gaussian kernel. For 2D displays the number of pixels per visual degree is assumed to be the same in horizontal and vertical directions, so an isotropic Gaussian is used. The Kent distribution is an analog to a 2D Gaussian on the surface of a 3D sphere [4]. 360° images encode spherical data, and the natural extension is to process them using such a distribution. However, computing a Kent based saliency map is slow due to a spatially varying kernel. Fortunately, several approximate alternatives exist.

In this paper, we benchmark four alternative methods for generating 360° saliency maps. We report accuracy and runtime for each algorithm, and present pseudocode to implement them. Based on these analyses and ease of implementation, we identify the most favorable approach.

II. Background

Saliency maps aggregate data from multiple observers into a representative map of human attention. Typically, saliency maps are generated by summing fixations from each observer into a discrete fixation map,

$$F^i(\boldsymbol{x}) = \sum_{k=1}^{M} \delta(\boldsymbol{x} - \boldsymbol{x_k^i}), \tag{1}$$

where M is the number of fixations by the i^{th} observer, $\boldsymbol{x}$ represents two dimensional pixel coordinates in the image, and $\boldsymbol{x_k^i}$ represents the coordinates of the k^{th} fixation [5]. δ is the Dirac function, which evaluates to 1 at $\delta(1)$, and 0 everywhere else. All maps from N observers are averaged to produce a unique representative map,

$$F(\boldsymbol{x}) = \frac{1}{N} \sum_{i=1}^{N} F^i(\boldsymbol{x}) \tag{2}$$

This fixation map is convolved with a 2D Gaussian kernel to produce a continuous map highlighting salient regions instead of individual pixels. The Gaussian kernel accounts for noise in measurement, such as calibration error, as well as falloff in visual acuity outside the foveal region. The standard deviation parameter, σ, is typically set between 1° and 5° visual angle due to the size of the foveal region and eye tracker error [6].

For 360° content, saliency researchers need to translate existing 2D methodologies to immersive 3D environments. 2D metrics are well understood [5], but they must be reconsidered for 360° saliency maps and scanpaths. As saliency prediction models adapt to 360° [7]–[10], there is a need for a standardized method to evaluate them against ground truth. Several eyetracking datasets are available for 360° images and video, with their own method for generating saliency maps [11]–[13].

III. Methods To Generate Saliency Maps in 360°

Four methods of generating 360° saliency maps have been proposed recently: applying a Gaussian kernel to fixations on the viewport and projecting the resulting maps onto the surface of the viewing sphere [12], applying a Gaussian kernel to the face of a cubemap [9], modifying the Gaussian kernel based on row [14], and simply applying an isotropic Gaussian kernel on the equirectangular image [11]. These methods have been proposed in separate publications, with algorithmic details scattered between supplementary materials, or as part of datasets. Here, we collect these alternatives in one place,

clarify algorithmic details, and benchmark their performance using a Kent distribution based method as ground truth. We briefly summarize and present pseudocode for the Kent based method, and the four approximate methods. For each method the output is a saliency map, S, that is then normalized such that all of the values sum to one. A square kernel is used for ease of implementation, with $kernel\_size$ set to 12 times the number of pixels per degree rounded up to an odd number. This size is larger than usual to ensure that all non-zero values are included. Helper functions are defined in the Appendix.

Kent Distribution: The Kent distribution is an isotropic bivariate normal distribution defined on the surface of a three dimensional unit sphere [4]. While the complete 5 parameter distribution is anisotropic, we use the simplified isotropic form. The probability density function f is defined as

$$f(\vec{\boldsymbol{x}}, \kappa, \vec{\boldsymbol{\gamma}}) = \frac{\kappa e^{\kappa \vec{\boldsymbol{\gamma}} \cdot \vec{\boldsymbol{x}}}}{4\pi sinh(\kappa)}, \tag{3}$$

where $\vec{\boldsymbol{x}} \in \mathbb{R}^3$ is an input vector representing a point on the sphere. The parameter $\kappa > 0$ represents the concentration of the probability density function. The parameter $\vec{\boldsymbol{\gamma}} \in \mathbb{R}^3$ is the mean direction of the distribution, around which points are normally distributed. Pixels in the equirectangular image x, y map to azimuth and elevation angles θ, ϕ. These angles are converted from spherical coordinates to a 3D vector in cartesian coordinates $\vec{\boldsymbol{x}}$. We compute the kernel weights by inputting the pixel neighborhood $N(x, y, kernel\_size)$ as vectors into f, where $\vec{\boldsymbol{\gamma}}$ is a vector that represents the current pixel. Due to unequal sampling of the sphere in equirectangular images, the kernel must be recomputed for each row. Input to this method is a fixation map F, and parameter κ.

procedure KENT(F,κ)
  $S \leftarrow zeros(num\_rows, num\_cols)$
  for $r = 1$ to num_rows **do** ▷ Parallelizable loop
    $\phi \leftarrow \pi |r/num\_rows - 0.5|$
    $\theta \leftarrow 0$ ▷ θ is constant
    $c = num\_cols/2$ ▷ c is constant
    $\vec{\gamma} \leftarrow sph2cart(\theta, \phi, 1)$
    $K \leftarrow f(N(c, r, kernel\_size), \kappa, \vec{\gamma})$
    $Normalize\ K$ ▷ Kernel weights sum to 1
    $S_{row} \leftarrow F \sim K$ ▷ Optimized to only output row r
    $S(r) = S_{row}$

In a MATLAB implementation we found that using a κ value of 707 or higher produces a result too large to fit in a 64 bit floating point number. Typical values of κ will be larger than this for modeling 1° or less of visual angle in saliency map generation. HPF[1] for high precision floating point numbers is needed, which increases the amount of time needed to perform the operations invoked by f. The following methods either approximate the Kent distribution (modified Gaussian), or operate outside the spherical domain.

Isotropic Gaussian Method: For 2D map generation a spatially invariant kernel is used, filtering the image in seconds. While quick, this computation does not account for distortions in the equirectangular image. Input to this method are a fixation map F, and standard deviation σ.

procedure ISOTROPIC(F,σ)
  $G_\sigma \leftarrow 1d\_gaussian(\sigma, kernel\_size)$
  $K \leftarrow G_\sigma \cdot G_\sigma^T$
  $S \leftarrow F \sim K$ ▷ $\sim$ indicates 2D convolution

Modified Gaussian Method: Upenik & Ebrahimi [14] introduce a modified Gaussian kernel that accounts for equirectangular distortions near the poles. A scale factor of $\frac{1}{cos\phi}$ is computed for each elevation angle ϕ, to stretch an isotropic Gaussian kernel horizontally. A bivariate Gaussian kernel is computed for each row as the matrix product

$$K = G_{\sigma_y} \cdot G_{\sigma_x}^T, \tag{4}$$

where G_{σ_y} is a column vector representing a 1D Gaussian with standard deviation σ_y in pixels, and $G_{\sigma_x}^T$ is a row vector representing a 1D Gaussian where $\sigma_x = \frac{\sigma_y}{cos\phi}$. This method applies a different filter at each row, requiring many 2D convolutions. This method has a similar runtime and structure to the Kent distribution, but is much easier to implement.

procedure MODIFIEDGAUSSIAN(F,σ)
  $S \leftarrow zeros(num\_rows, num\_cols)$
  for $r = 1$ to num_rows **do** ▷ Parallelizable loop
    $\phi \leftarrow \pi |r/num\_rows - 0.5|$
    $G_{\sigma_y} \leftarrow 1d\_gaussian(\sigma, kernel\_size)$
    $G_{\sigma_x} \leftarrow 1d\_gaussian(\sigma/cos\phi, kernel\_size)$
    $K \leftarrow G_{\sigma_y} \cdot G_{\sigma_x}^T$
    $S_{row} \leftarrow F \sim K$ ▷ Optimized to only output row r
    $S(r) = S_{row}$

Cubemap Method: Cubemaps reduce image distortions by projecting the spherical image onto cube faces representing perspective views from within the sphere. This format allows each face to be filtered with an isotropic Gaussian, but introduces discontinuities at the borders. To reduce this effect, cubes at two orientations are aligned and combined in equirectangular format to generate one representative map for the image. Weights W_1 and W_2 are applied with an element-wise multiplication to reduce the contribution of pixels near the edges of each face, as described by [9]. Transforming high resolution images between formats is time consuming, and the method does not completely remove border discontinuities.

procedure CUBEMAP(F,σ)
  $F_{rot} \leftarrow rotatesphereXYZ(F, \frac{pi}{4}, 0, \frac{pi}{4})$
  $G_\sigma \leftarrow 1d\_gaussian(\sigma, kernel\_size)$
  $K \leftarrow G_\sigma \cdot G_\sigma^T$
  $Cube_1 \leftarrow equirect2cube(F) \sim K$
  $Cube_2 \leftarrow equirect2cube(F_{rot}) \sim K$
  $S_1 \leftarrow cube2equirect(Cube_1)$
  $S_2 \leftarrow cube2equirect(Cube_2)$
  $S \leftarrow W_1 . * S_1 + W_2 . * S_2$

Viewport Method: 360° content is realized as a projection of the spherical image onto a viewport determined by the observer's head orientation. A Gaussian kernel can then be

[1] https://www.mathworks.com/matlabcentral/fileexchange/36534-hpf-a-big-decimal-class

applied directly within the viewport, and projected back onto the equirectangular image [12]. These values are summed across the equirectangular saliency map for each fixation. This method's runtime scales with the number of fixations. It can produce projection errors due to interpolation. The process of projecting each viewport onto the image generally has the longest runtime with high resolution images, and requires the HMD's horizontal and vertical field of view, fov_x and fov_y. Input for this method is a list of fixations in HMD screen space, head rotation, and the horizontal and vertical standard deviations of the 2D Gaussian in pixels, σ_x and σ_y.

procedure VIEWPORT(FIXATIONS,ROTATIONS,σ_x,σ_y)
  $S \leftarrow zeros(num\_rows, num\_cols)$
  for $i = 1$ to M $fixations$ **do**
    $viewport \leftarrow zeros(viewport\_size)$
    $x, y \leftarrow fixation_i$
    $R \leftarrow rotation_i$
    $viewport(y, x) = 1$
    $G_{\sigma_y} \leftarrow 1d\_gaussian(\sigma_y, kernel\_size)$
    $G_{\sigma_x} \leftarrow 1d\_gaussian(\sigma_x, kernel\_size)$
    $K \leftarrow G_{\sigma_y} \cdot G_{\sigma_x}$
    $S_{viewport} \leftarrow viewport \sim K$
    $S_{equirect} \leftarrow vp2sphere(S_{viewport}, R, fov_x, fov_y)$
    $S \mathrel{+}= S_{equirect}$

IV. Benchmark Methodology & Results

We benchmark the isotropic Gaussian, Viewport, Cubemap, and modified Gaussian saliency map methods to evaluate accuracy and performance. The Kent based method serves as ground truth, taking hours to compute for each image at full resolution. The modified Gaussian method also has a long runtime as it performs a convolution at each row. Instead, we first reduce the resolution of the image to 5% it's original size using MATLAB's $imresize$, then apply the modified Gaussian method. The result is then returned to the original resolution using bicubic interpolation. 5% scale was selected to generate a map in several seconds, with less than 2% deviation from the full resolution output. While saliency map generation is typically an offline process, our goal is to select a method that is efficient to ease analysis for researchers, and for real time applications such as 360° video streaming to many clients.

We use publicly available data and metrics for evaluation [12]. The dataset contains 40 images, each with fixations from at least 40 different observers classified using velocity thresholding. The image resolution ranged from 5376x2688 to 18332x9166. For our benchmark the saliency maps for each method are compared with the Kent based maps, and results are averaged across all images. Head orientation data was not provided with the dataset, meaning we had to use the provided maps instead of generating our own for the Viewport method. The parameter σ=3.34° was used for each method to match the dataset. The parameter κ=430 was found to match σ=3.34° by comparing the mean and standard deviation of points generated on a sphere using the Kent distribution, and points from a projected 2D Gaussian. We have not yet derived a closed form relation between the two parameters. Results were generated on a PC with an 8 core AMD FX-8350 processor, and MATLAB R2017a. MATLAB routines for these methods have been made publicly available, including a parallelized implementation of the Kent and Modified Gaussian method. The serialized code was used for the benchmark.

Method	Modified Gaussian (5%)	Isotropic Gaussian	Cubemap	Viewport	Viewport
Dataset	[12]	[12]	[12]	[12]	Ours
CC (↑)	0.994	0.988	0.984	0.408	0.930
KLD (↓)	0.011	0.097	0.108	0.865	0.273
RMSE (↓)	0.018	0.022	0.024	0.145	0.044
Time(s)	5.16	8.80	280	X	1031

TABLE I: Computed metrics for each method compared with ground truth Kent saliency maps. We implemented the Viewport method for use with our own dataset.

The saliency map metrics Correlation Coefficient (CC), KL Divergence (KLD), and RMSE were used to measure similarity between each of these maps and the Kent method map for all images [5]. CC ranges from 0 to 1, with high values indicating better performance. KLD represents divergence between two probability distributions, meaning lower values are considered a good score. Similarly, a low RMSE indicates a better approximation. Table I shows the computed metrics averaged across each image in the dataset. The modified Gaussian at 5% scale has the highest CC value, lowest KLD, and lowest RMSE indicating it had the highest similarity to the Kent saliency maps. In addition to high accuracy, the modified Gaussian at 5% is also efficient with an average runtime of 5.16 seconds.

The Viewport method maps provided with the dataset have no runtime, as they were read from file. We found that these maps scored lowest of all evaluated methods. An explanation for this could be that an isotropic Gaussian was used within the viewport when generating saliency maps. The viewport size is based on the Oculus DK2 screen and field of view, which is 1920x1080 and 94°x105°, meaning the number of pixels per degree is different in horizontal and vertical directions. To avoid this limitation, we use pixels per degree in the horizontal and vertical directions and apply an anisotropic Gaussian.

To evaluate the effect of an anisotropic Gaussian kernel, we implemented the Viewport method to process our own data. We conducted an IRB approved experiment on 22 participants with a similar protocol to [12], using the same VR eye tracking equipment and images. We computed Kent and Viewport saliency maps for each image and evaluate them in the same manner, shown in Table I. We produced more accurate results, raising CC scores from 0.41 to 0.93, and reducing RMSE from 0.15 to 0.04 across the same images. This effect can be observed visually, as Figure 1 compares heat maps from both Viewport and Kent method outputs.

V. Discussion

We found that a subsampled version of the modified Gaussian method is sufficient for generating accurate 360° saliency maps efficiently. While a sub-sampled Kent method could also be used, the parameter κ is not as intuitive as σ, which