Supplementary Materials: Instance-Level Panoramic Audio-Visual Saliency Detection and Ranking

Anonymous Authors

1 CUBEMAP-PROJECTION SAMPLING

Given the input image feature map $\mathbf{f}^{\mathbf{V}}$, let p_i be a pixel in the feature map, namely central point, and p_{ij} refers to its neighbor point. We sample a small set of neighbor points around each central point by leveraging cubemap projection. It contains the following steps:

i) equirectangular-to-cube transformation. Let the side length of a cube map be w. As the field-of-view (FoV) of each face is 90°, each face can be treated as a perspective camera with a focal length of $\frac{w}{2}$, and they all share the same center point in the world coordinate system. Due to the fixed viewing direction in cubemap projection, a rotation matrix R_h can represent the extrinsic matrix of each camera. For a pixel p_i on equirectangular map, we can transform it into the coordinate on the certain cube face h. This process is detailed in Equation (1) and Algorithm 2.

$$q_i^x = \sin(\theta) \cdot \cos(\phi); q_i^y = \sin(\phi); q_i^z = \cos(\theta) \cdot \cos(\phi)$$
$$K = \begin{bmatrix} w/2 & 0 & w/2 \\ 0 & w/2 & w/2 \\ 0 & 0 & 1 \end{bmatrix}; \hat{p}_i = K \cdot R_h^T \cdot q_i \tag{1}$$

where θ and ϕ represent the longitude and latitude of point p_i on the sphere. The range of θ spans from $-\pi$ to $+\pi$, while the range of ϕ spans from -0.5π to $+0.5\pi$. The x, y, and z components of vector q_i are represented as q_i^x, q_i^y, and q_i^z.

ii) uniform sampling on the cube map. Similar to equirectangular sampling, we select the eight nearest neighbor pixels of each pixel on the equirectangular projection as neighbor points. The process is given in Equation (2):

$$p_{ij} = p_i(x \pm a, y \pm b), \{a, b = 0, 1; a, b = 0, 2\} \tag{2}$$

iii) cube-to-equirectangular transformation. All these neighbors are projected back to the equirectangular domain. Given a neighboring point p_{ij} on a specific face h, we can perform a coordinate transformation to map it onto the ER projection, as illustrated in Equation (3) and Algorithm 1.

$$\begin{aligned} q_{ij} &= R_i \cdot K^{-1} \cdot \hat{p_{ij}} \\ \theta &= \arctan\left(q_{ij}^x / q_{ij}^z\right) \\ \phi &= \arcsin\left(q_{ij}^y / |q_{ij}|\right) \end{aligned} \tag{3}$$

Algorithm 1: Cube-to-Equirectangular

Input: x_c and y_c: x and y coordinates of the input cube map; *side*: the face of the input cube map; w_c: width of the input cube map.

Output: x_e and y_e: x and y coordinates of the output equirectangular map

// 1. define the transformation function from 3D Cartesian coordinate into spherical coordinates;

Function GetThetaPhi(x, y, z)**:**
  $dv \leftarrow \sqrt{x \cdot x + y \cdot y + z \cdot z}$;
  $x' \leftarrow x/dv$;
  $y' \leftarrow y/dv$;
  $z' \leftarrow z/dv$;
  $\theta \leftarrow \arctan 2(y', x')$;
  $\phi \leftarrow \arcsin(z')$;
  return θ, ϕ;

// 2. compute the spherical coordinates θ and ϕ of the output equirectangular map;

if *side* == *"front"* **then**
  $\theta, \phi \leftarrow$ GetThetaPhi($1, x, y$);
end
else if *side* == *"right"* **then**
  $\theta, \phi \leftarrow$ GetThetaPhi($-x, 1, y$);
end
else if *side* == *"left"* **then**
  $\theta, \phi \leftarrow$ GetThetaPhi($x, -1, y$);
end
else if *side* == *"back"* **then**
  $\theta, \phi \leftarrow$ GetThetaPhi($-1, -x, y$);
end
else if *side* == *"bottom"* **then**
  $\theta, \phi \leftarrow$ GetThetaPhi($-y, x, 1$);
end
else if *side* == *"top"* **then**
  $\theta, \phi \leftarrow$ GetThetaPhi($y, x, -1$);
end

// 3. map spherical coordinates to 2D coordinates;

$x\_c \leftarrow 0.5 + 0.5 \cdot (\theta/\pi)$;
$y\_c \leftarrow 0.5 + \phi/\pi$;

Algorithm 2: Equirectangular-to-Cube

Input : x_e and y_e: x and y coordinates of the input equirectangular map; h_e and w_e: height and width of the input equirectangular map; w_c: width of the output cube map.
Output : x_c and y_c: x and y coordinates of the output cube map; $side$: the face of the output cube map.

// 1. compute spherical coordinates θ and ϕ;
$\theta \leftarrow (x\_e/w\_e - 0.5) \cdot 2 \cdot \pi$;
$\phi \leftarrow (y\_e/h\_e - 0.5) \cdot \pi$;
// 2. compute 3D Cartesian coordinates x, y, and z;
$x \leftarrow \cos(\phi) \cdot \sin(\theta)$;
$y \leftarrow \sin(\phi)$;
$z \leftarrow \cos(\phi) \cdot \cos(\theta)$;
// 3. compute the absolute value of x, y, and z;
$x\_abs \leftarrow |x|$;
$y\_abs \leftarrow |y|$;
$z\_abs \leftarrow |z|$;
// 4. compute 3D cube coordinates of the six faces;
if $x\_abs \geq y\_abs$ ***and*** $x\_abs \geq z\_abs$ **then**
  if $x > 0$ **then**
    $x\_c, y\_c, side \leftarrow -z, y, \text{“}right\text{”}$;
  end
  else
    $x\_c, y\_c, side \leftarrow z, y, \text{“}left\text{”}$;
  end
  $max\_axis \leftarrow x\_abs$;
end
else if $y\_abs \geq x\_abs$ ***and*** $y\_abs \geq z\_abs$ **then**
  if $y > 0$ **then**
    $x\_c, y\_c, side \leftarrow x, -z, \text{“}bottom\text{”}$;
  end
  else
    $x\_c, y\_c, side \leftarrow x, z, \text{“}top\text{”}$;
  end
  $max\_axis \leftarrow y\_abs$;
end
else
  if $z > 0$ **then**
    $x\_c, y\_c, side \leftarrow x, y, \text{“}front\text{”}$;
  end
  else
    $x\_c, y\_c, side \leftarrow -x, y, \text{“}back\text{”}$;
  end
  $max\_axis \leftarrow z\_abs$;
end
// 5. map 3D coordinates to 2D coordinates;
$x\_c \leftarrow x\_c/max\_axis$;
$y\_c \leftarrow y\_c/max\_axis$;
$x\_c \leftarrow (x\_c + 1)\ /2 \cdot cw$;
$y\_c \leftarrow (y\_c + 1)\ /2 \cdot cw$;