FOCAS: Practical Video Super Resolution using Foveated Rendering

Lingdong Wang
University of Massachusetts Amherst
Amherst, Massachusetts, USA
lingdongwang@umass.edu

Mohammad Hajiesmaili
University of Massachusetts Amherst
Amherst, Massachusetts, USA
hajiesmaili@cs.umass.edu

Ramesh K. Sitaraman
University of Massachusetts Amherst
Amherst, Massachusetts, USA
ramesh@cs.umass.edu

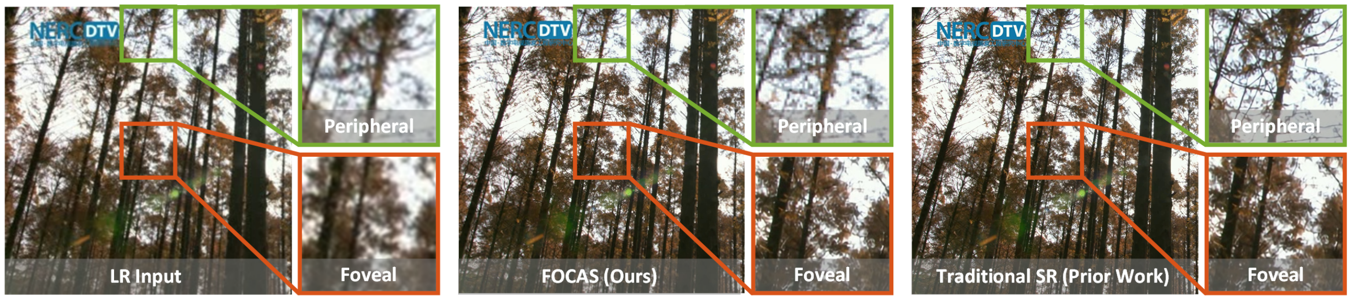

Figure 1: Left: low-resolution (LR) input; Middle: FOCAS takes 24 ms to construct a foveated high-resolution (HR) frame; Right: Traditional (non-foveated) super-resolution (SR) takes 64ms to construct a full HR frame. Traditional SR upgrades the entire frame of the LR input to HR, incurring large computational costs, high latencies, and low frame rates, making it unsuitable for real-time video streaming. However, human vision is more sensitive to video quality in the central foveal region, and less sensitive in the periphery. FOCAS upgrades only the central foveal region of each frame to HR, resulting in reduced computational cost, lower latency, and higher frame rates, while nearly matching the perceptual video quality of traditional SR.

ABSTRACT

Super-resolution (SR) is a well-studied technique for reconstructing high-resolu- tion (HR) images from low-resolution (LR) ones. SR holds great promise for video streaming since an LR video segment can be transmitted from the video server to the client that then reconstructs the HR version using SR, resulting in a significant reduction in network bandwidth. However, SR is seldom used in practice for real-time video streaming, because the computational overhead of frame reconstruction results in large latency and low frame rate.

To reduce the computational overhead and make SR practical, we propose a deep-learning-based SR method called **Fo**veated **Cas**caded Video Super Resolution (FOCAS). FOCAS relies on the fact that human eyes only have high acuity in a tiny central foveal region of the retina. FOCAS uses more neural network blocks in the foveal region to provide higher video quality, while using fewer blocks in the periphery as lower quality is sufficient. To optimize the computational resources and reduce reconstruction latency, FOCAS formulates and solves a convex optimization problem to decide the number of neural network blocks to use in each region of the frame. Using extensive experiments, we show that FOCAS reduces the latency by 50% – 70% while maintaining comparable visual quality as traditional (non-foveated) SR. Further, FOCAS provides a 12 – 16× reduction in the client-to-server network bandwidth in comparison with sending the full HR video segments.

CCS CONCEPTS

• **Computing methodologies** → **Computer vision**; **Neural networks**; **Perception**.

KEYWORDS

super resolution, foveated rendering, deep learning, latency

ACM Reference Format:
Lingdong Wang, Mohammad Hajiesmaili, and Ramesh K. Sitaraman. 2021. FOCAS: Practical Video Super Resolution using Foveated Rendering. In *Proceedings of the 29th ACM International Conference on Multimedia (MM '21), October 20–24, 2021, Virtual Event, China.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3474085.3475673

1 INTRODUCTION

Super-resolution (SR) is a well-studied mechanism to reconstruct high-resolution (HR) frames of a video from low-resolution (LR) ones. A key benefit of SR is that it reduces the network bandwidth required for video streaming, since a video client can download the LR version of a video segment from the video server and reconstruct the HR version using SR. Thus, SR alleviates the network bandwidth bottleneck of video streaming at the cost of additional computation at the video client [19]. However, state-of-the-art SR methods incur a large computational overhead, resulting in high latency to construct each frame, leading to unacceptably low frame rates. So, despite its potential, SR is seldom used for real-world real-time video streaming.

With the goal of making SR practical, we propose **Fo**veated **Cas**caded Video Super Resolution (FOCAS), a real-time foveated video SR method based on deep learning. Human eyes only possess high acuity in the fovea, the central 5.2° region of the retina [3]. FOCAS uses this property to render the foveal region around the eye fixation in high quality, while rendering the peripheral region in low quality. Specifically, FOCAS downloads LR video segments and uses neural network to perform SR. But unlike a traditional SR, FOCAS uses more neural network blocks to produce a higher quality image in the foveal region, while using fewer blocks in the peripheral region where a lower quality will suffice. Thus, FOCAS offers similar visual quality as traditional SR, but using far fewer computation resources. FOCAS can also be contrasted with traditional foveated rendering that requires high-resolution (HR) content to fill in the foveal region. Traditional foveated rendering requires the server to download HR content, thus consuming significantly more network bandwidth than FOCAS that downloads only LR content.

The main technical challenge of foveated SR is meeting the stringent real-time latency requirement of 30–50 ms [2] to perform SR on each frame, so that an adequate frame rate can be maintained. Note that the latency requirement imposes a budget on the computational cycles available for performing SR. FOCAS optimally allocates the computational budget by spending more computational cycles for performing SR in the foveal region where higher quality is desired, and significantly fewer computational cycles in the peripheral region where a lower quality is sufficient.

Our Contributions. We now list our key contributions.

(1) We propose the novel idea of foveated SR that combines the best aspects of foveated rendering and traditional SR. Foveated SR achieves the same reduction in bandwidth as traditional SR by enabling the client to download only LR video segments from the server. However, foveated SR has significantly less computational overhead than traditional SR, allowing it to achieve similar video quality with smaller latency. Likewise, foveated SR is similar to traditional foveated rendering in that the foveal region is rendered in high quality. However, traditional foveated rendering does not reduce bandwidth significantly as HR versions of the video segment will still need to be downloaded for display in the foveal region.

(2) We design FOCAS that implements foveated SR using a deep learning approach. FOCAS shrinks the intermediate feature map in a cascaded manner during the inference phase. It can be easily applied to any traditional SR model to perform foveated SR.

(3) FOCAS formulates the problem of performing foveated SR within a latency budget as a convex optimization problem that can then be solved efficiently. The optimization determines how many neural network blocks to use in which parts of the frame.

(4) We finally conduct extensive experiments and show that FOCAS can reduce the latency by 50% – 70%, leading to a $2\times - 3\times$ frame rate improvement, while achieving visual quality that is comparable to traditional SR. Also, by downloading LR video segments instead of HR ones, FOCAS achieves $12 - 16\times$ bandwidth saving. Our implementation is available at github.com/UMass-LIDS/focas.

2 BACKGROUND AND RELATED WORK

2.1 Foveated Rendering

Human eyes have high sensitivity in the fovea, the central 5.2° region of retina. But the acuity of human eye rapidly decreases outside fovea towards the periphery [3]. To utilize this property, foveated rendering only renders the foveal region with high quality, while rendering the peripheral region with low quality. In this way, the computation required for rendering is largely reduced, as 5.2° covers only 0.8% of total pixels on a regular display [9].

Multiresolution [9] proposes to render a frame in three concentric regions with different qualities, while each region's resolution is assigned according to a linear model of human visual acuity. [23] proposes a foveated rendering system that varies pixel shading rate based on its eccentricity. DeepFovea [16] proposes a deep-learning-based video reconstruction method. But DeepFovea reconstructs HR frames from discrete pixels instead of LR frames as in our case.

Prior foveated rendering methods assume that the HR versions of the video are readily available in the video client. While foveated rendering also has wide applications in video streaming where video segments are transferred through the wide-area network (WAN). A foveated video streaming strategy is proposed in [25] that divides one frame into multiple tiles, sends HR tiles for the foveal region and LR tiles for the periphery. This method uses a commodity webcam to trace eye movement instead of professional eye trackers. [13] utilizes foveated video streaming for cloud gaming. [24] develops a foveated streaming system for 360° video. [20] streams 360° video using both a foveated scheme and semantic saliency.

2.2 Super Resolution

Single-image super-resolution (SISR) aims at improving the resolution of one single image. The first deep-learning model for SISR is SRCNN [6], a 3-layer convolutional neural network (CNN). VDSR [17] then proposes a deep network and introduces residual learning. ESPCN [27] designs sub-pixel convolution, an efficient way to upscale the feature map while maintaining locality. Later, DBPN [10], HAN [22], CAR [28] and many other methods further improve the performance of SISR.

By extending SISR, video super-resolution (VSR) tries to enhance the resolution of the video. The key difference of VSR from SISR is that VSR can utilize temporal information from other frames, as scenes in a video typically have a strong correlation and locality. From a technical standpoint, the literature on VSR could be divided into two categories: non-recurrent methods and recurrent methods.

Non-recurrent VSR methods are usually based on 2D or 3D CNN. Viewing each frame as a 2D tensor, EDVR [30] uses stacked 2D deformable convolution layers to conduct VSR. TOFlow [31] performs optical flow estimation and warp operation between frames, then aggregates frames to generate high-quality output. But inaccurate motion estimation and motion compensation (MC&ME) might incur artifacts. To avoid explicit MC&ME, PFNL [33] utilizes progressive fusion and non-local operation to capture features. From another aspect, considering time as the third dimension, DUF [15] adopts 3D convolution to generate dynamic upsampling filters to capture spatio-temporal features.

Recurrent VSR methods adopt recurrent neural network (RNN) to capture historical semantic information from previous frames.

FRVSR [26] improves frame resolution by using MC&ME to align adjacent frames in a recurrent manner. RBPN [11] extends back-projection operation from SISR to VSR and performs it recurrently. RLSP [7] propagates both previous SR output and previous feature, and processes them with stacked convolutional layers. Following RLSP, RRN [14] uses stacked residual blocks and achieves a better SR quality. We take RRN as the base model for our work because of its straightforward structure and good performance.

In traditional video SR, the full-size input will go through the whole model, resulting in an output with uniform quality everywhere. However, since human eye pays much more attention to the foveal region, this is an inefficient approach to allocate computational resources. As a result, when traditional SR methods are applied to scenarios like real-time video streaming, they largely sacrifice model capacity and SR performance to meet the stringent latency requirement [4, 32, 34]. In contrast, FOCAS allocates quality only where it matters to the human visual system, achieving lower latencies without a noticeable quality degradation.

3 FOCAS SYSTEM DESIGN

FOCAS includes three phases as shown in Fig. 2. In the first training phase, we obtain a trained CNN model. The cornerstone of FOCAS is the second phase, where through an optimization process, we find the feature depth and region size to maximize foveated SR quality given the latency constraint. Finally, we apply the settings to model, and generate foveated SR results in the inference phase.

In what follows, we first present the model structure in Section 3.1. In Section 3.2, we explain how we construct a customized inference phase of FOCAS that takes into account the foveated SR. Finally, we introduce how to train a model as required in the inference phase in Section 3.3. Note that we present the details of the quality allocation phase in Section 4.

3.1 Model Structure

The architecture of FOCAS is built based on RRN [14] as shown in Fig. 3. Here we briefly introduce it and refer to [14] for details.

FOCAS adopts a recurrent structure. In recurrent iteration t, it takes the current and previous frames I_t and I_{t-1}, previous feature H_{t-1} and previous SR output O_{t-1} as the inputs. At the end of iteration, the model will output H_t and O_t. The output H_t is a deep learning feature map carrying global information of the video. O_t is the result of SR, an upsampled image with higher resolution.

At first, the image O_{t-1} is reshaped by a Pixel Unshuffle layer. Pixel Unshuffle layer moves data from the spatial dimension to the depth dimension, transforming the tensor shape from $C \times sH \times sW$ to $s^2C \times H \times W$, where s is an upscale factor. Pixel Unshuffle layer is the inverse operation of Pixel Shuffle layer [27], which will be used to transform a deep feature back to an image.

Later, input data are concatenated and fed into a convolution (Conv) layer followed by ReLU [8]. Then the feature enters some stacked residual blocks (ResBlock) [12]. ResBlock is composed of Conv-ReLU-Conv layers, with a skip connection adding the input to the output. In one branch, the feature from ResBlocks is sent into a Conv-ReLU layer to be the output feature H_t. In the other branch, the feature is processed by a Conv layer, reshaped by a Shuffle layer, and added with the bicubic-upsampled I_t to be the SR result O_t.

Despite its recurrent structure, FOCAS can simulate non-recurrent video SR methods by not using recurrent states. Specifically, by setting the recurrent state O_{t-1}, H_{t-1} as null and still inputting $[I_{t-1}, I_t]$, the model only receives recent frames without global knowledge. As the state-of-art video SR methods may adopt recurrent or non-recurrent structure, we examine the performance of both types of FOCAS model for comprehensiveness.

3.2 Inference Phase

The idea of FOCAS inference comes from the observation that each ResBlock in the model will introduce finer details to the feature map, hence improve the visual quality of SR output. On the other hand, human eyes only identify high visual quality in a small foveal region, and endure low quality in the large peripheral region.

Leveraging these properties, FOCAS only allows the feature map for foveal region to go through more ResBlocks and gain higher quality, while let the feature map for peripheral region early exit to decrease the overall latency. As a result, the foveal region of SR output is high-quality, while the peripheral region ends up with lower but acceptable quality. Such a foveated SR strategy achieves comparable visual quality with full-size SR, but greatly reduces the latency by spending less computation on the periphery.

Following the common practice in foveated rendering [9, 25], we adopt a three-region quality distribution. That is, FOCAS will output an image consisting of three regions with decreasing visual qualities. The foveal region around the eye fixation will have the highest quality, the peripheral region will have the lowest quality, and the blending region in the middle will have a medium quality.

The inference process of such a three-region FOCAS model is illustrated in the third phase of Fig. 2, where the eye fixation is assumed to be the center. At *first*, the input data $I_t, I_{t-1}, O_{t-1}, H_{t-1}$ are concatenated into a feature map and processed by a Conv layer. Then, the feature map is processed by several ResBlocks to be a low-quality feature, which can be interpreted as a low-quality image. *Secondly*, we crop a sub-feature around the eye fixation from the low-quality feature. Due to the locality of convolution operations, the feature map has a spatial correspondence with the output image. Hence this sub-feature can be viewed as the blending region in the final output. Only this sub-feature is sent into successive ResBlocks and enhanced to achieve middle quality, while the rest part is no longer computed. *Finally*, a small feature map for the foveal region is cropped out of middle-quality feature, goes through several additional ResBlocks, and achieves the highest quality.

After the above steps, we have a large low-quality feature map for the periphery, middle-size medium-quality feature for the blending region, and a small high-quality feature for the foveal region. We then stack these three features together centered at the eye fixation. Finally, we send this mixed-quality feature map into the last Conv layers, and obtain the output image and feature, both with decreasing quality from the gaze point to periphery.

In the inference phase, the size of intermediate feature map shrinks in a cascaded way. Therefore our method is named as Foveated **Cascaded** Video Super Resolution. We argue that the design of cascade inference suits the sequential architecture of modern CNN. Therefore, FOCAS can be pervasively applied to existing SR methods, transforming them into their foveated SR counterparts.

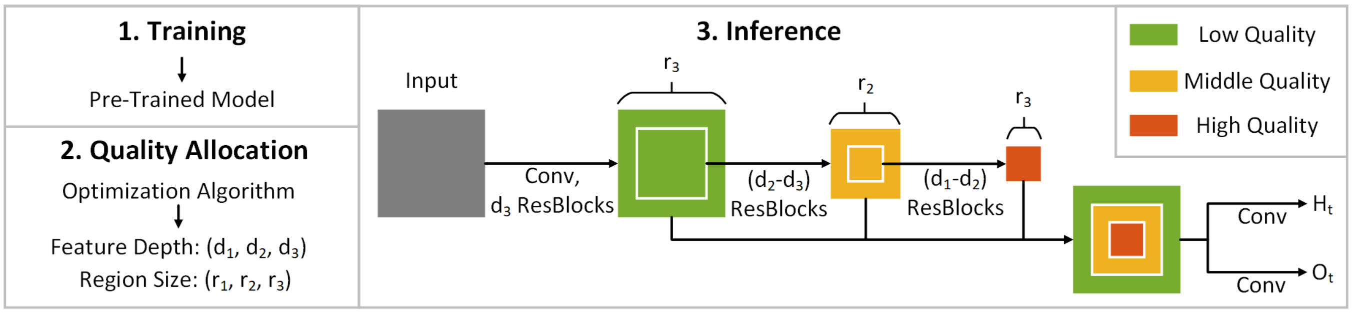

Figure 2: Overview of FOCAS.

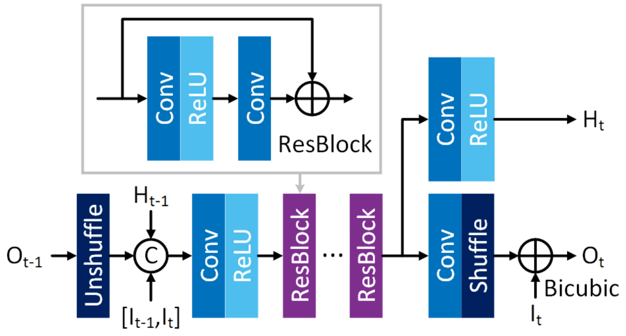

Figure 3: Model structure of FOCAS.

3.3 Training Phase

In contrast to the inference phase, our model is trained without cascade. One may argue that the training for foveated SR could be based on different quality levels. But we claim that a cascaded training strategy is highly inefficient, since only a small region of input data goes deeply into the model. As a result, the input data is not fully utilized, and the deeper ResBlocks are not fully trained.

Another challenge is the need for adaptability of "interpreter" layers, the last Conv layers before outputs working as the interpreter of feature maps. As required in the inference, the interpreter must adapt to features of different qualities from different ResBlocks. One can force the interpreter to adapt by feeding these features to it in the training phase. But, it degrades the performance since the model wastes its capacity on peripheral features, instead of focusing on learning high-quality feature for the foveal region.

The solution to both problems is the same — train the end-to-end model just like a regular RRN model without cascade. In this way, the complete input data goes through the full model, leading to full data utilization and full model training. Besides, the residual connections will bypass features from all ResBlocks to the interpreter layers. So the interpreter can eventually handle features of different qualities, even if they are never directly received in training.

Since FOCAS shares the same model structure and training process with RRN, it can also share the same model parameter with RRN. In other words, FOCAS can be derived from a pre-trained RRN model, transforming a traditional SR model to a foveated SR one. It proves that FOCAS is easy-to-use and can be applied to many existing models even without re-training.

4 OPTIMUM QUALITY ALLOCATION

In this section, we focus on the optimization process of FOCAS, where we make two decisions for each region— how large the region size should be, and what is the right feature depth, i.e., how many ResBlocks should each region's feature go through. These two decisions jointly determine the amount of computational resources we allocate for each region, leading to different visual quality and different latency. It is vital to allocate quality to each region wisely, so that we can achieve the maximum visual quality under a limited amount of resources. We formulate the problem as an optimization problem in Section 4.1 and show how to solve it in Section 4.2.

4.1 Problem Modeling and Formulation

We first provide a model for visual quality. Then, we introduce visual importance to model human vision. Next, we present the estimation method of inference latency. Finally, we formulate the optimization problem in Eq. (4) using all these concepts.

4.1.1 Visual quality and its relation to feature depth. It is challenging to model the visual quality for two reasons. Firstly, there is no perfect measurement of video quality in human vision. Here we just adopt normalized PSNR as the metric of visual quality. Secondly, the visual quality of an SR result has a complicated relationship with the content of the input image, recurrent state, and the feature depth. When we have access to the target video or the target dataset, an intuitive solution is to measure and record the mapping between these factors and the visual quality of our pre-trained model over the target data. Our goal in FOCAS, however, is to be generic and adaptive to unknown input data. Hence, we model visual quality as being content-independent and related only to feature depth. In other words, we model all feature maps going through the same number of ResBlocks result as having the same visual quality.

We now define a function $q(d)$ that maps feature depth d to visual quality. We use Vimeo-90K dataset [31] as our training dataset, assume that it has a generic data distribution, and use it as the input to the optimization module of FOCAS. We then feed the output features from all ResBlocks to the last Conv layer, and obtain output images with different qualities. Finally, we measure the PSNR scores of these images and normalize them to $[0, 1]$. These normalized PSNR scores are used as visual qualities for different feature depths as shown in Fig. 4, where the x-coordinate is feature depth d, and the y-coordinate is the visual quality $q(d) \in [0, 1]$. Note that Fig. 4 represents the visual quality of FOCAS with recurrent states.

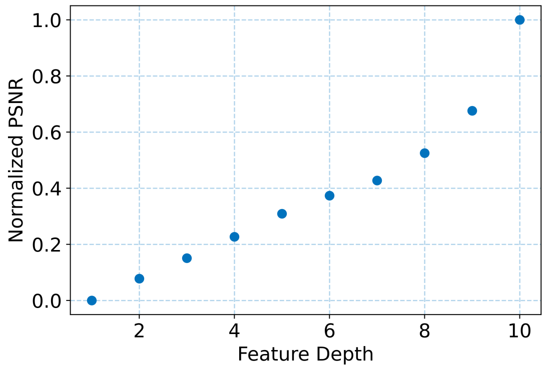

Figure 4: Visual quality as a function of feature depth.

FOCAS divides each frame into concentric squares centered at the position of eye fixation as shown in Figure 5. The smallest center square and the annular regions between two consecutive squares form *quality regions* that are shown in different colors. FOCAS assigns the same feature depth to pixels within the same region, resulting in similar video quality. Note that the center square would receive the highest feature depth and the feature depth decreases as moving towards the periphery.

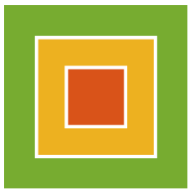

Figure 5: Quality regions.

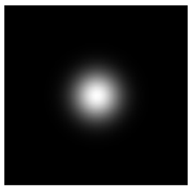

Figure 6: Visual importance.

4.1.2 Visual Importance. We adopt the model in [21] to assign a weight to each pixel according to its importance in human vision, and the weight follows a normalized 2D Gaussian function centered at the gaze point. The mathematical formula is shown as follows:

$$w_{u,v} = \frac{1}{2\pi\sigma_u\sigma_v} e^{-\left(\frac{(u-u_e)^2}{2\sigma_u^2} + \frac{(v-v_e)^2}{2\sigma_v^2}\right)}, \tag{1}$$

where $w_{u,v}$ is the weight for position (u, v), (u_e, v_e) is the position of eye fixation, and both σ_u, σ_v are set to be the foveal size 64 pixels (2°) as in [21]. Denoting $\sigma = \sigma_u = \sigma_v$, we can rewrite Eq. (1) into

$$w(x) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2}{2\sigma^2}}, \tag{2}$$

where $x = \sqrt{(u-u_e)^2 + (v-v_e)^2}$ represents the distance from a pixel to the position of eye fixation. An illustration of the visual importance weight mask is shown in Fig. 6, where eye fixation is the center and brightness means a higher value.

4.1.3 Inference Latency. The inference latency of the FOCAS model is dominated by the computation time of ResBlocks. Since we assign intermediate features with the same number of channels, the runtime of a single ResBlock is proportional to the area (height × width) of the input feature. So, the runtime of FOCAS for one region is proportional to the number of ResBlocks and the area of input feature. We model the total inference latency as a linear function of the sum of latency for each region, i.e.,

$$t(\boldsymbol{d}, \boldsymbol{r}) = A + B \sum\nolimits_{i=1}^{N} (d_i - d_{i+1}) r_i^2. \tag{3}$$

where A, B are the linear coefficients. For a region i, the area of input feature is the square of region size r_i. Because a higher-quality feature is computed based on a lower-quality feature, the feature for a new region only need to go through $d_i - d_{i+1}$ ResBlocks, where d_i is the current region's feature depth and d_{i+1} is the outer region's. Overall, $(d_i - d_{i+1})r_i^2$ represents the latency of region i. To measure the coefficients A and B in Eq. (3), we randomly generate 108 models with different feature depths as well as input areas, and then measure their latency. As shown in Fig. 7, we can obtain relatively accurate A and B using linear regression.

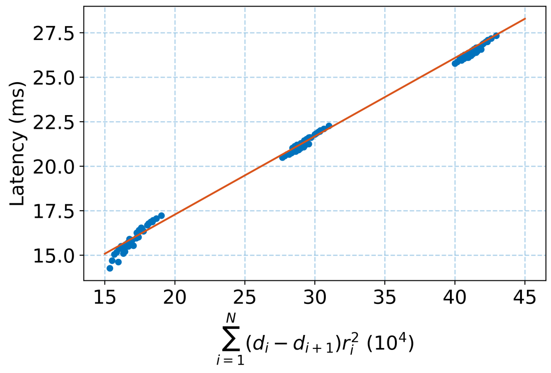

Figure 7: Linear fitting for latency estimation.

4.1.4 Optimization Problem Formulation. The objective of the optimization is to maximize the overall visual quality under a foveated view by assigning region size and feature depth for each region. Meanwhile, the latency of the process must obey a specified time bound. Putting them together, this problem is formulated as

$$\begin{aligned} \max_{\boldsymbol{d},\boldsymbol{r}} \ & \sum_{i=1}^{N} \sum_{x=r_{i-1}}^{r_i} x e^{-\frac{x^2}{2\sigma^2}} q(d_i) \\ \text{s.t. } & t(\boldsymbol{d}, \boldsymbol{r}) \le T, \\ & r_{i-1} \le r_i, \ \forall i, \\ & d_{i-1} \ge d_i, \ \forall i, \end{aligned} \tag{4}$$

where the optimization variables are $\boldsymbol{d}$ and $\boldsymbol{r}$. More specifically, variable $\boldsymbol{d} = (d_1, d_2, \cdots, d_N)$ represents the vector of feature depths with element d_i as the feature depth of region i. Further, $\boldsymbol{r} = (r_1, r_2, \cdots, r_{N-1})$ represents the region sizes. Note that the outermost region is always the full input, so $r_N = L$, where L is the input feature size. In this work we have $L = 270$, 4× downscaled from the image height 1080. We set $r_0 = 0$ and $d_{N+1} = 0$ for consistency.

We note that the closed-form representation of the objective function in Problem (4) is derived by substituting $w(x)$ in Eq. (2) into the following equation:

$$2\pi x w(x) q(d_i) = \frac{1}{\sigma^2} x e^{-\frac{x^2}{2\sigma^2}} q(d_i). \tag{5}$$

We now further explain the derivation of the above objective function. The foveated visual quality of SR output is the dot product of its visual quality and the visual importance. Here we assume that the gaze point is always at the center of the image. Then an intuitive interpretation of foveated visual quality is to stack the visual quality distribution in Fig. 5 with the visual importance weight mask in Fig. 6, and sum up the values in Fig. 5 weighted by Fig. 6. To do so, we further assume that regions are circular for mathematical tractability. Then, the objective function in Eq. (5) could be interpreted as follows. There are N regions and x is the distance from a pixel to the eye fixation. For each region i, $1 \leq i \leq N$, x varies from the inner boundary r_{i-1} to the outer boundary r_i. At a given distance x, there are $2\pi x$ pixels on the ring, all with visual importance $w(x)$. While within the same region i, all pixels have the same visual quality $q(d_i)$. In other words, we traverse all pixels, sum up their visual quality scores weighted by their visual importance scores. A sanity check is that, the full-size SR result has the highest quality 1 everywhere, and the sum of $w(\cdot)$ over all pixels equals 1, so full-size SR has the highest possible foveated visual quality 1. In the end, by removing the constant factors in Eq. (5), we obtain an equivalent but simplified objective function in Eq. (4).

Finally, we explain the three constraints in Problem (4). Firstly, the inference latency defined in Eq. (3) is no more than a pre-specified time limit T. As discussed in Section 2.1, T should be between 30-50 ms. Our experiments show that $\sim$ 20 ms is enough for FOCAS to achieve performance near the full SR approach. The second constraint enforces that the radii are monotonically non-decreasing. Thirdly, feature depth of the inner region is no less than the outer region. The third constraint comes from the intuition that q is monotonically increasing with d, and we want to assign a higher visual quality to inner regions compared to the outer regions.

4.2 Solving the Optimization Problem

In this section, we develop our solution approach to solve Problem (4). To facilitate our algorithm design, we first simplify Problem (4) to deal with one optimization variable. To do so, we leverage the fact that there are only 10 ResBlocks in the FOCAS model, so the value of feature depth is between 1 to 10, i.e., $d_i \in \{1, 2, \cdots, 10\}$. Besides, the number of region is also small (3 in FOCAS). As a result, the search space of $\boldsymbol{d}$ is limited and can be fully traversed by an exhaustive search. In this way, we can view $\boldsymbol{d}$ as a pre-specified constant and focus on optimizing the region sizes $\boldsymbol{r}$.

As the second step, we relax the objective function in Eq. (4) by viewing discrete pixels as continuous as follows:

$$\begin{aligned}&\sum_{i=1}^{N}\sum_{x=r_{i-1}}^{r_i} x e^{-\frac{x^2}{2\sigma^2}} q(d_i) \approx \sum_{i=1}^{N}\int_{r_{i-1}}^{r_i} x e^{-\frac{x^2}{2\sigma^2}} q(d_i)\, dx\\ &= \sigma^2 [q(d_1) e^{-\frac{r_0^2}{2\sigma^2}} + \sum_{i=1}^{N-1}(q(d_{i+1}) - q(d_i)) e^{-\frac{r_i^2}{2\sigma^2}} - q(d_N) e^{-\frac{r_N^2}{2\sigma^2}}],\end{aligned} \tag{6}$$

where the second line in above equation is obtained by solving the integral. We then replace the objective function in Eq. (4) with Eq. (6) to obtain an approximate problem. Since $\sigma, r_0, r_N, q(d_i)$ are all constants, we only keep the middle term for simplicity without loss of generality. Finally, we reverse the maximization problem

Algorithm 1 Quality Allocation of FOCAS

```
**Input:** Time limit $T$. Number of regions $N$.
**Output:** Optimal region sizes $\boldsymbol{r}^*$. Optimal feature depths $\boldsymbol{d}^*$.
  **for** every $\boldsymbol{d} = (d_1, d_2, \cdots, d_N)$ s.t. $d_1 \geq d_2 \geq d_N$ **do**
    Solve problem Eq. (8) with $N, T, \boldsymbol{d}$ to get the result $\boldsymbol{a}$.
    $\boldsymbol{r} \leftarrow (\lceil\sqrt{a_1}\rceil, \lceil\sqrt{a_2}\rceil, \cdots, \lceil\sqrt{a_{N-1}}\rceil)$ .
    Compute the score of objective function in Eq. (5) with $\boldsymbol{d}, \boldsymbol{r}$.
  **end for**
  **return** $\boldsymbol{d}, \boldsymbol{r}$ with the highest score.
```

into a minimization one as follows:

$$\begin{aligned}\min_{\boldsymbol{r}=(r_1,\cdots,r_{N-1})} &\sum_{i=1}^{N-1}(q(d_i) - q(d_{i+1})) e^{-\frac{r_i^2}{2\sigma^2}}\\ \text{s.t. } & A + B\sum_{i=1}^{N}(d_i - d_{i+1}) r_i^2 \leq T,\\ & r_{i-1} \leq r_i,\ \forall i.\end{aligned} \tag{7}$$

We then transform the constraint $r_{i-1} \leq r_i$, into an equivalent one by squaring both sides, hence, the variables all become square, i.e., in the format of r_i^2. Then, we can replace the optimization variable with $a_i = r_i^2$, and re-write the problem as

$$\begin{aligned}\min_{\boldsymbol{a}=(a_1,\cdots,a_{N-1})} &\sum_{i=1}^{N-1}(q(d_i) - q(d_{i+1})) e^{-\frac{a_i}{2\sigma^2}}\\ \text{s.t. } & A + B\sum_{i=1}^{N}(d_i - d_{i+1}) a_i \leq T,\\ & 0 \leq a_1 \leq a_2 \leq \cdots \leq a_{N-1} \leq L^2.\end{aligned} \tag{8}$$

Theorem 4.1. *Problem (8) is a convex optimization problem.*

Proof. Denote the objective function in Eq. (8) as f, whose Hessian matrix is $\nabla^2 f = \text{diag}((\cdots, \frac{1}{4\sigma^4}(q(d_i) - q(d_{i+1})) e^{-a_i}, \cdots))$. Since visual quality of the inner region is always higher than the outer, we have $q(d_i) - q(d_{i+1}) > 0$, which means $\nabla^2 f \geq 0$. Hence, the objective function f is convex. Meanwhile, all the constraints are linear. Therefore, Problem (8) is a convex optimization problem. □

Since Problem (8) is convex, we can solve it optimally using convex solvers. We adopt CVXPY [1, 5], a Python framework for convex optimization, to solve the problem optimally.

In summary, the algorithm to set region sizes and feature depths for a FOCAS model is shown in Alg. 1. The user needs to indicate the latency requirement T and the number of regions N. The algorithm will exhaustively search every $\boldsymbol{d}$, and then compute the corresponding $\boldsymbol{r}$ by solving a convex optimization problem. At the end, the combination of $\boldsymbol{d}, \boldsymbol{r}$ with the highest foveated visual quality is returned as the solution.

5 EXPERIMENTAL RESULTS

5.1 Overview and Setup

5.1.1 Dataset. Our training dataset is Vimeo-90K [31], a common training dataset for SR containing 91,701 7-frame video segments without eye trace. We train the FOCAS model on the training set of Vimeo-90K, and then measure the visual quality w.r.t. feature depth in Fig. 4 on its testing set. All frames are originally in 448×256 resolution, and randomly cropped into 256×256 tiles to be the ground truth. Targeting at 4× SR task, we downsample ground-truth examples 4× to be 64×64 as input. Downsampling is realized by Gaussian blur with $\sigma = 1.6$ as used in [14]. We use horizontal and vertical flips with probability 0.5 for data augmentation.

We adopt [29] as our testing dataset, which contains videos and eye traces from human observers. We use 37 HD videos in the dataset. All videos have 1920×1080 resolution and around 300 frames, with contents varying from people, buildings to natural scenes. We further divide each video into 50-frame (2-second) clips as input examples. For each video, there are 34 eye traces. We use the average position of left eye and right eye's gaze points as the eye fixation. Only the first eye fixation during a frame is considered, and the eye fixation is set to the center when losing track.

5.1.2 Performance Metrics. We adopt two performance metrics to measure visual quality. Both metrics are applied to the luminance channel, that is, Y channel of a YCbCr-format image.

The first metric is Peak Signal to Noise Ratio (PSNR). It is a non-foveated metric that treats everywhere equally, representing the visual quality under a global view.

The second metric is Eye-Weighted PSNR (EWPSNR) [21]. EW-PSNR is similar to PSNR, but assigns weights to pixels according to a 2D-Gaussian model of human vision. The EWPSNR metrics is obtained as follows:

$$\text{EWPSNR} = 10 \log \left(\frac{(2^n - 1)^2}{\text{EWMSE}} \right), \tag{9}$$

where

$$\text{EWMSE} = \frac{\sum_{u=1}^{U} \sum_{v=1}^{V} \left(w_{u,v} \cdot (I'_{u,v} - I_{u,v})^2 \right)}{UV \sum_{u=1}^{U} \sum_{v=1}^{V} w_{u,v}}, \tag{10}$$

U and V are the image height and width, (u, v) is a position, $w_{u,v}$ is defined in Eq. (1), and n is the number of bits per sample set to 8. EWPSNR is a foveated metric that cares more about the foveal region but less about the periphery, measuring the foveated visual quality in human vision.

5.1.3 Implementation. We implement FOCAS using the PyTorch framework. We adopt CVXPY [1, 5] as the convex optimization solver for Alg. 1. Experiments are all conducted on GTX 2080Ti GPU. We follow the same hyper-parameters and training setting as RRN [14]. Specifically, intermediate features have 128 channels. The learning rate is 10^{-4}, and then decreases to 10^{-5} at epoch 60, with a total of 70 epochs. The model is optimized by the Adam [18] optimizer towards ℓ_1 loss function. Lastly, the settings of the Adam optimizer are $\beta_1 = 0.9, \beta_2 = 0.999$ and 5×10^{-4} weight decay.

5.1.4 Stability. SR methods might incur artifacts or noises in the video, which severely harms the visual quality. Here we develop two strategies to prevent artifacts and improve the SR stability.

In the inference phase, directly stacking feature maps of different qualities causes obvious artifacts at region boundaries. To solve this problem, we perform a 4-pixel linear interpolation at the boundary, blending the features of adjacent regions.

In recurrent FOCAS, the feature depth for one position may vary among frames, resulting in the possibility of small reconstruction error. The impact of this small error, however, could be accumulated and amplified since it is sent into future steps together with recurrent states. As a result, it spreads out and becomes a major artifact. This problem only appears when FOCAS runs for too long. Hence, we solve it by alternating FOCAS inference and full-size SR inference. The full-size inference will correct the small error in recurrent states and prevent it from accumulating, while we can still adopt foveated inference to gain latency reduction for the remaining frames.

Table 1: FOCAS with Different Quality Allocation Settings

Name	T	d_1	d_2	d_3	$4r_1$	$4r_2$	L (ms)	Score
FOCAS-15	15	10	4	1	128	224	14.12	0.5474
FOCAS-16	16	10	8	1	224	256	14.82	0.8800
FOCAS-17	17	10	4	1	288	368	16.11	0.9603
FOCAS-20	20	10	8	1	416	448	18.47	0.9984
FOCAS-25	25	10	3	1	544	824	23.64	0.9999

5.2 Results

5.2.1 Quality Allocation. Since the visual quality function measured on FOCAS with and without recurrent state are similar, we use the function in Fig. 4 for both experiments. We adopt a three-region foveated scheme, i.e., $N = 3$. Then, by setting different latency constraint T, we can derive different versions of FOCAS from Alg. 1 with different performance-latency trade-offs. Here we report 5 variants of quality allocation settings in Table 1.

In this table, T is the latency limit in ms, and we name the corresponding FOCAS version after their T. With 3 regions, d_i represents the feature depth of region i and r_i is the region size. Note that r_3 is always the full input size so we omit it in Table 1. Beyond these, column L refers to the actual inference latency in runtime. And "Score" represents the value of objective function described in Eq. (5). Targeting at 4× SR task, all these models gain 16× theoretic bandwidth saving since the size of uncompressed LR resource to download is $\frac{1}{16}$ of its HR origin. With the MPEG4 compression codec, FOCAS achieves 12.85× bandwidth saving on our testing dataset.

The input feature to ResBlocks are 4× down-scaled in width and height by the Pixel Unshuffle layer. For an intuitive comparison with the ground-truth image of size 1080×1920, we show $4r_i$ in the table. Taking the architecture of FOCAS-20 as an example. Firstly, the input of size 1080×1920 (actually 270×480) goes through $d_3 = 1$ ResBlock, achieving low quality as the peripheral region. Then, a blending region of $r_2 = 448 \times 448$ is cropped and goes through 7 more blocks to achieve depth $d_2 = 8$. Finally, the foveal region of $r_1 = 416 \times 416$ reaches depth $d_1 = 10$ for the highest quality.

The results show that runtime inference latency always satisfies the latency limit T. As for the SR performance, FOCAS-15 performs poorly due to its stringent latency requirement. But FOCAS-20 performs satisfyingly with 0.9984 score in 18.47 ms, and FOCAS-25 achieves 0.9999 in 23.64 ms. As the latency increases, the objective score rises rapidly at first but gradually saturates. This diminishing return observation is reasonable since we can easily improve the visual quality by enhancing the quality of foveal region when the score is low. But once it reaches the limit, we can barely benefit from increasing the peripheral quality.

Table 2: Comparison of Non-Recurrent SR Methods

Method	Foveated	Latency(ms)	PSNR	EWPSNR
Bicubic		1.14	27.77	91.00
RRN-10L		63.55	**34.13**	**97.72**
RRN-3L		24.01	33.30	96.90
FOCAS-15	✓	14.12	30.86	95.70
FOCAS-16	✓	14.82	30.90	96.90
FOCAS-17	✓	16.11	30.92	97.42
FOCAS-20	✓	18.47	31.04	97.71
FOCAS-25	✓	23.64	31.23	**97.72**

Table 3: Comparison of Recurrent SR Methods

Method	RRN/FOCAS	Latency(ms)	PSNR	EWPSNR
Bicubic	-	1.14	27.77	91.00
RRN-10L	-	63.55	**34.88**	**98.50**
RRN-5L	-	35.17	30.78	97.30
FOCAS-15	3/4	35.30	32.30	96.82
FOCAS-16	3/4	35.70	32.33	97.69
FOCAS-17	3/4	36.44	32.36	98.11
FOCAS-20	2/3	36.50	32.27	98.37
FOCAS-25	1/3	33.61	31.95	98.41

5.2.2 Non-recurrent FOCAS. We compare the performance of non-recurrent FOCAS with other non-recurrent SR methods in Table 2. The full-size SR baselines are bicubic upsampling and non-recurrent RRN [14]. We report the 10-block RRN (RRN-10L), which has the highest visual quality, as well as 3-block RRN (RRN-3L), which has similar latency with FOCAS-25.

The results show that FOCAS-20 and FOCAS-25 achieve comparable foveated visual quality (EWPSNR) with RRN-10L but gaining 71% and 63% latency reduction, respectively. Comparing with RRN-3L, FOCAS-25 has similar latency and lower PSNR, but significantly higher EWPSNR. This means that FOCAS is worse than RRN-3L from a global view but looks better in human eyes. Note that RRN-3L can be viewed as letting the whole image go through 3 ResBlocks, allocating uniform quality to the image. In contrast, FOCAS will let the foveal region go through 10 blocks but let the periphery go through just 1 block. Verified by the experiments, we conclude that foveated SR performs better than traditional SR in a non-recurrent setting.

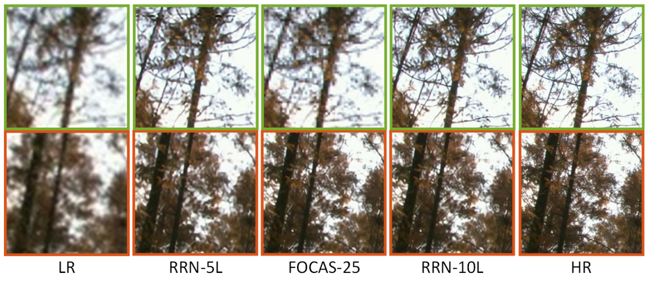

Figure 8: Results of recurrent SR methods. Top: peripheral region. Bottom: foveal region.

5.2.3 Recurrent FOCAS. For recurrent SR methods, we take bicubic upsampling, 10-block RRN, and 5-block RRN as baselines and show the experimental results in Table 3. As discussed in Section. 5.1.4, to make recurrent FOCAS stable, we need to alternatively use full-size SR and FOCAS inference. 'RRN/FOCAS' = 1/3 means that one out of four frames is processed by RRN-10L and rest are by FOCAS. We deliberately set the ratio so that all FOCAS models have similar latency with RRN-5L for fair comparisons.

As T increases, FOCAS model will perform increasingly better in terms of foveated visual quality (EWPSNR). FOCAS-16 to FOCAS-25 suppress the EWPSNR score of RRN-5L with similar inference time. The best one among them, FOCAS-25, can achieve comparable EWPSNR with RRN-10L in only a half of runtime. The same observation is also illustrated in Fig. 8. The performance of FOCAS-25 is very close to RRN-10L and clearly better than RRN-5L in the foveal region, at the cost of peripheral quality.

Different from the perfect performance of non-recurrent FOCAS, recurrent FOCAS performs slightly worse than full-size SR and gains less latency reduction. This is because the greedy nature of FOCAS conflicts with its recurrent manner. Uniform-quality SR will enhance somewhere not being watched at the moment, which is wasteful in non-recurrent scenarios. But in recurrent scenarios, the computation of an unseen position will contribute to the future through recurrent states. In contrast, FOCAS never invests in the future. FOCAS greedily allocates resources according to the current gaze point, leading to performance loss in the long run. Despite this, by comparing FOCAS-25 with RRN-5L, we conclude that foveated SR also outperforms full-size SR in a recurrent setting.

6 CONCLUSION

Exploiting the fact that human visual acuity rapidly decreases outside the fovea, we proposed the idea of foveated SR to reduce the inference latency without noticeable visual quality degradation. Toward this, we designed FOCAS that generates HR videos out of LR ones by reconstructing the foveal region with more computational resources, and paying less attention to the periphery. In this way, FOCAS reduces the inference latency significantly while maintaining similar visual quality. The unique technical challenge in FOCAS is to find the right size and quality of the foveated regions. We addressed this challenge by formulating and optimally solving a convex optimization problem that determines feature depths and region sizes of all regions. As proven by extensive experiments, non-recurrent FOCAS achieves the same foveated visual quality while saving about 70% latency. Meanwhile, recurrent FOCAS achieves visual quality that is comparable to the baseline, while also attaining a 50 – 70% latency reduction, 2 – 3× frame rate improvement, and 12 – 16× bandwidth saving. Thus, FOCAS is a practical way to employ foveated SR in real-time video streaming.

ACKNOWLEDGMENTS

We acknowledge the support from the U.S. National Science Foundation (NSF) under grant numbers CNS-2102963, CAREER-2045641, CNS-1763617, CNS-1901137.

REFERENCES

[1] Akshay Agrawal, Robin Verschueren, Steven Diamond, and Stephen Boyd. 2018. A rewriting system for convex optimization problems. *Journal of Control and Decision* 5, 1 (2018), 42–60.

[2] Rachel Albert, Anjul Patney, David Luebke, and Joohwan Kim. 2017. Latency Requirements for Foveated Rendering in Virtual Reality. *ACM Trans. Appl. Percept.* 14, 4, Article 25 (Sept. 2017), 13 pages. https://doi.org/10.1145/3127589

[3] Christine A. Curcio, Kenneth R. Sloan, Robert E. Kalina, and Anita E. Hendrickson. 1990. Human photoreceptor topography. *Journal of Comparative Neurology* 292, 4 (1990), 497–523. https://doi.org/10.1002/cne.902920402 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/cne.902920402

[4] Mallesham Dasari, Arani Bhattacharya, Santiago Vargas, Pranjal Sahu, Aruna Balasubramanian, and Samir R. Das. 2020. Streaming 360-Degree Videos Using Super-Resolution. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications.* 1977–1986. https://doi.org/10.1109/INFOCOM41043.2020.9155477

[5] Steven Diamond and Stephen Boyd. 2016. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research* 17, 83 (2016), 1–5.

[6] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. 2014. Learning a Deep Convolutional Network for Image Super-Resolution. In *Computer Vision – ECCV 2014*, David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars (Eds.). Springer International Publishing, Cham, 184–199.

[7] D. Fuoli, S. Gu, and R. Timofte. 2019. Efficient Video Super-Resolution through Recurrent Latent Space Propagation. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW).* 3476–3485. https://doi.org/10.1109/ICCVW.2019.00431

[8] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep Sparse Rectifier Neural Networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 15)*, Geoffrey Gordon, David Dunson, and Miroslav Dudík (Eds.). JMLR Workshop and Conference Proceedings, Fort Lauderdale, FL, USA, 315–323. http://proceedings.mlr.press/v15/glorot11a.html

[9] Brian Guenter, Mark Finch, Steven Drucker, Desney Tan, and John Snyder. 2012. Foveated 3D Graphics. *ACM Trans. Graph.* 31, 6, Article 164 (Nov. 2012), 10 pages. https://doi.org/10.1145/2366145.2366183

[10] Muhammad Haris, Gregory Shakhnarovich, and Norimichi Ukita. 2018. Deep Back-Projection Networks for Super-Resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[11] Muhammad Haris, Gregory Shakhnarovich, and Norimichi Ukita. 2019. Recurrent Back-Projection Network for Video Super-Resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).*

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[13] Gazi Karam Illahi, Thomas Van Gemert, Matti Siekkinen, Enrico Masala, Antti Oulasvirta, and Antti Ylä-Jääski. 2020. Cloud Gaming with Foveated Video Encoding. *ACM Trans. Multimedia Comput. Commun. Appl.* 16, 1, Article 7 (Feb. 2020), 24 pages. https://doi.org/10.1145/3369110

[14] Takashi Isobe, Fang Zhu, Xu Jia, and Shengjin Wang. 2020. Revisiting Temporal Modeling for Video Super-resolution. arXiv:2008.05765 [eess.IV]

[15] Younghyun Jo, Seoung Wug Oh, Jaeyeon Kang, and Seon Joo Kim. 2018. Deep Video Super-Resolution Network Using Dynamic Upsampling Filters Without Explicit Motion Compensation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[16] Anton S. Kaplanyan, Anton Sochenov, Thomas Leimkühler, Mikhail Okunev, Todd Goodall, and Gizem Rufo. 2019. DeepFovea: Neural Reconstruction for Foveated Rendering and Video Compression Using Learned Statistics of Natural Videos. *ACM Trans. Graph.* 38, 6, Article 212 (Nov. 2019), 13 pages. https://doi.org/10.1145/3355089.3356557

[17] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. 2016. Accurate Image Super-Resolution Using Very Deep Convolutional Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[18] Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs.LG]

[19] Royson Lee, Stylianos I. Venieris, and Nicholas D. Lane. 2020. Neural Enhancement in Content Delivery Systems: The State-of-the-Art and Future Directions *(DistributedML'20).* Association for Computing Machinery, New York, NY, USA, 34–41. https://doi.org/10.1145/3426745.3431336

[20] Wei-Tse Lee, Hsin-I Chen, Ming-Shiuan Chen, I-Chao Shen, and Bing-Yu Chen. 2017. High-resolution 360 Video Foveated Stitching for Real-time VR. *Computer Graphics Forum* 36, 7 (2017), 115–123. https://doi.org/10.1111/cgf.13277

[21] Zhicheng Li, Shiyin Qin, and Laurent Itti. 2011. Visual attention guided bit allocation in video compression. *Image and Vision Computing* 29, 1 (2011), 1–14. https://doi.org/10.1016/j.imavis.2010.07.001

[22] Ben Niu, Weilei Wen, Wenqi Ren, Xiangde Zhang, Lianping Yang, Shuzhen Wang, Kaihao Zhang, Xiaochun Cao, and Haifeng Shen. 2020. Single Image Super-Resolution via a Holistic Attention Network. In *Computer Vision – ECCV 2020*, Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (Eds.). Springer International Publishing, Cham, 191–207.

[23] Anjul Patney, Marco Salvi, Joohwan Kim, Anton Kaplanyan, Chris Wyman, Nir Benty, David Luebke, and Aaron Lefohn. 2016. Towards Foveated Rendering for Gaze-Tracked Virtual Reality. *ACM Trans. Graph.* 35, 6, Article 179 (Nov. 2016), 12 pages. https://doi.org/10.1145/2980179.2980246

[24] Miguel Fabian Romero-Rondón, Lucile Sassatelli, Frédéric Precioso, and Ramon Aparicio-Pardo. 2018. Foveated Streaming of Virtual Reality Videos. In *Proceedings of the 9th ACM Multimedia Systems Conference* (Amsterdam, Netherlands) *(MMSys '18).* Association for Computing Machinery, New York, NY, USA, 494–497. https://doi.org/10.1145/3204949.3208114

[25] Jihoon Ryoo, Kiwon Yun, Dimitris Samaras, Samir R. Das, and Gregory Zelinsky. 2016. Design and Evaluation of a Foveated Video Streaming Service for Commodity Client Devices *(MMSys '16).* Association for Computing Machinery, New York, NY, USA, Article 6, 11 pages. https://doi.org/10.1145/2910017.2910592

[26] Mehdi S. M. Sajjadi, Raviteja Vemulapalli, and Matthew Brown. 2018. Frame-Recurrent Video Super-Resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[27] Wenzhe Shi, Jose Caballero, Ferenc Huszar, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. 2016. Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[28] W. Sun and Z. Chen. 2020. Learned Image Downscaling for Upscaling Using Content Adaptive Resampler. *IEEE Transactions on Image Processing* 29 (2020), 4027–4040. https://doi.org/10.1109/TIP.2020.2970248

[29] Toinon Vigier, Josselin Rousseau, Matthieu Perreira Da Silva, and Patrick Le Callet. 2016. A New HD and UHD Video Eye Tracking Dataset. In *Proceedings of the 7th International Conference on Multimedia Systems* (Klagenfurt, Austria) *(MMSys '16).* Association for Computing Machinery, New York, NY, USA, Article 48, 6 pages. https://doi.org/10.1145/2910017.2910622

[30] Xintao Wang, Kelvin C.K. Chan, Ke Yu, Chao Dong, and Chen Change Loy. 2019. EDVR: Video Restoration With Enhanced Deformable Convolutional Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops.*

[31] Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T Freeman. 2019. Video Enhancement with Task-Oriented Flow. *International Journal of Computer Vision (IJCV)* 127, 8 (2019), 1106–1125.

[32] Hyunho Yeo, Youngmok Jung, Jaehong Kim, Jinwoo Shin, and Dongsu Han. 2018. Neural Adaptive Content-aware Internet Video Delivery. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18).* USENIX Association, Carlsbad, CA, 645–661. https://www.usenix.org/conference/osdi18/presentation/yeo

[33] Peng Yi, Zhongyuan Wang, Kui Jiang, Junjun Jiang, and Jiayi Ma. 2019. Progressive Fusion Video Super-Resolution Network via Exploiting Non-Local Spatio-Temporal Correlations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV).*

[34] Anlan Zhang, Chendong Wang, Bo Han, and Feng Qian. 2021. Efficient Volumetric Video Streaming Through Super Resolution. In *Proceedings of the 22nd International Workshop on Mobile Computing Systems and Applications* (Virtual, United Kingdom) *(HotMobile '21).* Association for Computing Machinery, New York, NY, USA, 106–111. https://doi.org/10.1145/3446382.3448663