1 Contrast Preservation

As described in the main text, we use a contrast-preserving post-process kernel inspired by existing work in the area of image filtering and downsampling [Öztireli and Gross 2015; Grundland et al. 2006; Kim et al. 2011]. While our implementation is most closely related to the work by Grundland et al. [Grundland et al. 2006], it differs in the following ways:

- We vary the size of neighborhood for and magnitude of contrast enhancement with the degree of foveation, i.e. the contrast enhancement increases with retinal eccentricity.
- Since we don’t have an estimate for the actual loss of contrast (e.g. ratio of standard deviation), we use a user defined constant factor of enhancement.

In practice, our approach consists of a sequence of three postprocess passes. We first compute average color for each pixel using a radially increasing filter width. We implement a separable filter, so this step takes two passes. Next, we boost the contrast for each pixel. The following describes the final pass:

\[
p'_{ij} = \bar{p}_{ij} + f_e \cdot (1 + \sigma_{ij}) \cdot (p_{ij} - \bar{p}_{ij})
\]

Where, \(p_{ij}\) and \(p'_{ij}\) are the pixel colors before and after contrast enhancement, respectively; \(\bar{p}_{ij}\) is the pixel color averaged over a radially increasing region; \(\sigma_{ij}\) is the degree of foveation measured as the effective filter width (i.e. 0 in the fovea), and \(f_e\) is a user-defined constant (set to 0.2 in our implementation).

Following is a code snippet that implements our approach:

```cpp
vec3 enhanceContrast(vec3 pix, vec3 pmean, vec2 sigma) {
  // computer amount of contrast enhancement
  // based on degree of foveation (sigma)
  float cScale = 1.0f + length(sigma) * gParams.fe;
  vec3 scaledColor = pmean + (pix - pmean) * cScale;
  return clamp(scaledColor, vec3(0), vec3(1));
}
```

Note that we can optimize our implementation by removing the first two passes, and instead generating mipmaps of the input image. The final pass would then fetch from the appropriate mip level based on the retinal eccentricity. However, using mipmapping for contrast enhancement reduces image quality in a noticeable manner.

References


Figure 1: Psychometric functions for emulated foveated rendering using our HMD setup. Each column shows the psychometric functions for a different user. The top row shows the psychometric functions for Aliased foveation, the middle row shows the psychometric functions for Temporally Stable foveation, and bottom row shows the psychometric functions for contrast-preserving foveation. The x-axis indicates the stimulus level i.e. blur rate measured in arcminutes per degree. Higher is better.

Figure 2: Psychometric functions for emulated foveated rendering using our Desktop setup. Each column shows the psychometric functions for a different user. The top row shows the psychometric functions for Temporally Stable foveation, and bottom row shows the psychometric functions for contrast-preserving foveation. The x-axis indicates the stimulus level i.e. blur rate measured in arcminutes per degree. Higher is better.
Figure 3: Psychometric functions for our comparison between Guenter et al [Guenter et al. 2012] and our proposed rendering system. Each column shows the psychometric functions for a different user. The top row shows the psychometric functions for Guenter et al [Guenter et al. 2012], and bottom row shows the psychometric functions for our proposed renderer. The x-axis indicates the transition size in degrees. Lower is better.