Final Report –Tsotsos Lab Summer 2015
Omar Abid
Research Assistant

Retinal Filter Foveation

Input is the following constructor:

```
RetinalFilter (int w, int h, float pixelsPerDegree = 10, float fovealRange = 4, bool luminance =
true, float perRange_Begin = 10, float perRange_End = 11);
```

Note that at this time the 'w' and 'h' variables are not used since the image width and height are taken directly from the matrix with the convolve function. It was kept at this time since it's parent 'Filter.cpp' required the constructor have these arguments. It should be removed in a feature revision. Also note that the fovealRange, perRange_Begin, perRange_end refer to the foveal range for which the density is greatest and the begin and end point for which the peripheral rod density is at it's greatest respectively. These arguments are given in degrees. The 'luminance' flag is set to true when applying the rod receptor density function and false when applying the cone receptor density function.

Output is an image with the density function applied. See figure 2.

The following files were updated : *RetinalFilter.cpp* and *RetinalFilter.h*. These files were initially blank templates and then updated with a Rod and Cone Photo Density equation. It works using a blurring effect:

1. Overrides the 'convolve' function in the 'Filter.cpp'
2. Generates the foveation function with respect to the input image size.
3. Performs the foveation by applying this function to the input image.

<u>Overrides the 'convolve' function in 'Filter.cpp'</u>
Although not really a convolution, access to the input and output image matrix was needed and this function allowed us to do this. Subsequent function calls relied on these matrix pointers to generate the output image. The constructor for this is given as:

```
RetinalFilter::convolve (Matrix* input, Matrix* output, bool scale)              (1)
```

<u>Generates the Foveation function with respect to the input image size</u>
The foveation function below returns the 'densityFunction'; the density of either the rods or the cones and further detail of this will be given in later sections.

```
returnPhotoDensityFunction (densityFunction, imgW,imgH)                 (2)
```

The image width and height (imgW, imgH) respectively were returned to us from the output image matrix pointer. These properties are defined by the end user during the construction of the Retinal

Filter. Once these arguments are passed to (2), the function converts the pixel sizes given as input (imgW,imgH) to an angular value. Since the fall off of photoreceptor density is a function of eccentricity this conversion was needed. In addition, the conversion factor is dependent on the angle of view of the specific camera used thus it is left to the end user to define such values. The 'calibrated values' listed in the table below are given for the wide angle camera used in the lab.

Calibrated Values	**End User representation**
theta_h = 58 ° theta_w= 44.82 °	theta_h = imgH/pixelsPerDegree theta_w = imgW/pixelsPerDegree

Note that in both cases the values are given in degrees; theta_h and theta_w represent the angular height and width respectively.

If the user wishes to get the Cone Photo Density the following equation is used (Weber & Triesch, 2009)

$$Y=c1\ \exp(\lambda1\ E)+c2\ \exp(\lambda2\ E)+c3\ \exp(\lambda3\ E) \qquad \textbf{(3)}$$

Where Y represents the cone photoreceptor density in 1000 cells/mm^2 as a function of visual field eccentricity E in degrees. c1,c2,c3 and λ1, λ2, λ3 are constants determined experimentally.

Alternatively the Rod photodensity equation was determined using a matlab fourier8 approximation by manually giving raw data points (Purves D, et al., Neuroscience. 2nd edition; 2001. Anatomical Distribution of Rods and Cones.)

```
Y = (a0 + a1*cos(x*w) + b1*sin(x*w) + a2*cos(2*x*w) + b2*sin(2*x*w) +
a3*cos(3*x*w) + b3*sin(3*x*w) + a4*cos(4*x*w) + b4*sin(4*x*w) +          (4)
a5*cos(5*x*w) + b5*sin(5*x*w) + a6*cos(6*x*w) + b6*sin(6*x*w) +
a7*cos(7*x*w) + b7*sin(7*x*w) + a8*cos(8*x*w) + b8*sin(8*x*w));
```

Both (3) and (4) are consistent with experimental data. See Figure 1.

<u>Perform the Foveation</u>

The number of pixels within a region to blur over is dependent on the density function of the rods and cones given by equations (3) and (4). The function was applied and Figure 2 shows the result with the cone receptor density applied.

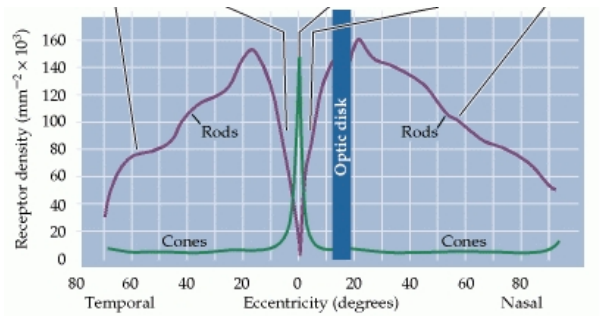
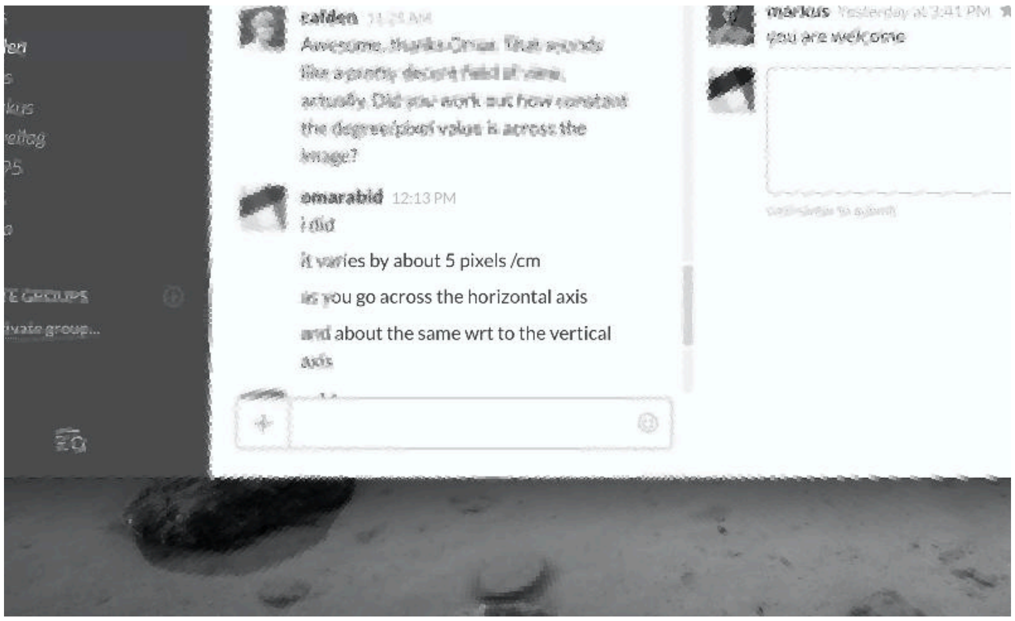


Figure 1: *The receptor density of rods and cones (Anatomical Distribution of Rods and Cones) modeled in Tarzann.*


caiden [illegible]
Awesome, thanks Omar. That sounds like a pretty decent field of view, actually. Did you work out how constant the degree/pixel value is across the image?

omarabid 12:13 PM
i did
it varies by about 5 pixels /cm
as you go across the horizontal axis
and about the same wrt to the vertical axis

markus Yesterday at 3:41 PM
you are welcome

Figure 2: *Sample foveation applied to an image.*