

# Sobel Edgeness Signature

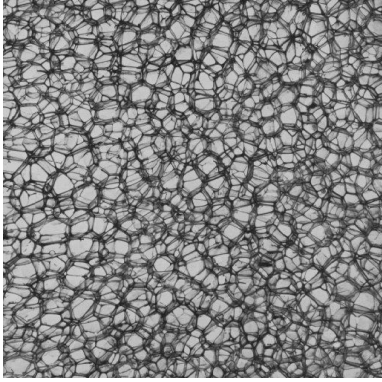
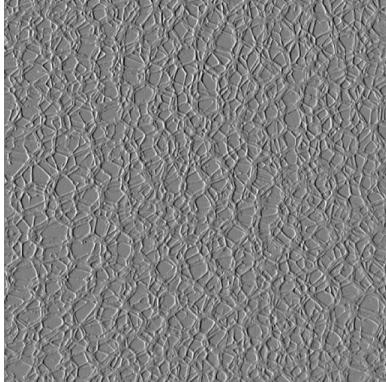
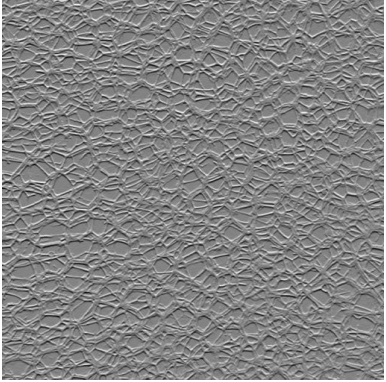
Heather Bradfield & Dylan Hart

## 1. Calculating a Sobel Filter

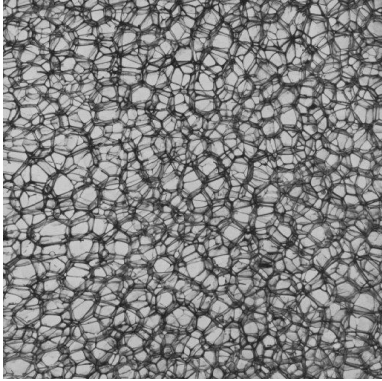
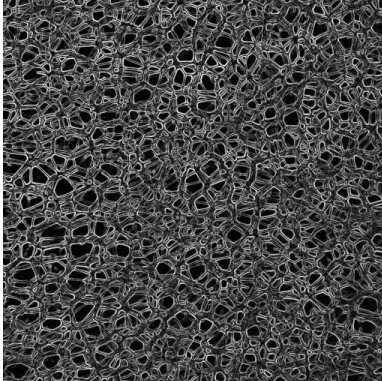
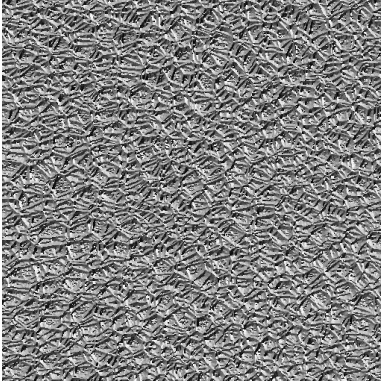
A Sobel filter is calculated by turning the image into a field of vectors. These vectors represent the gradient of the image at each pixel calculated from a 3x3 neighborhood. This is done by combining the results of two convolutions to calculate the approximate slopes of the X and Y directions. The kernels for this convolution are listed below as  $K_x$  and  $K_y$ .

$$K_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad K_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

The results of these convolutions are shown below:

		
Original (BUBBLES)	X Convolution	Y Convolution

Both of the convolutions look very similar, but have a clear difference in direction. Each pixel in these images make up the X or Y component of that pixel's gradient vector. The magnitude of the vector can be calculated using the distance formula,  $\sqrt{X^2 + Y^2}$ , and the phase of the vector can be calculated using the inverse tangent,  $\text{atan2}(Y, X)$ . These computed images are shown below:

		
Original (BUBBLES)	Magnitude	Phase

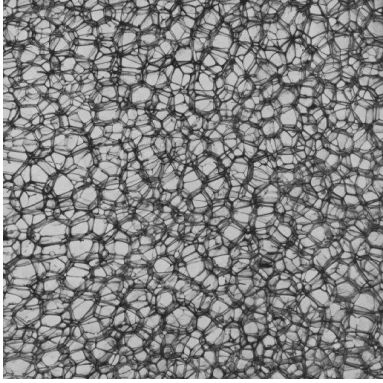
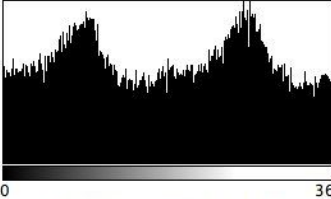
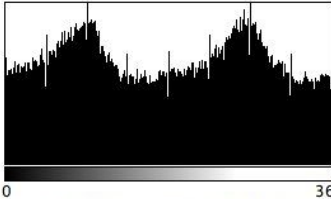
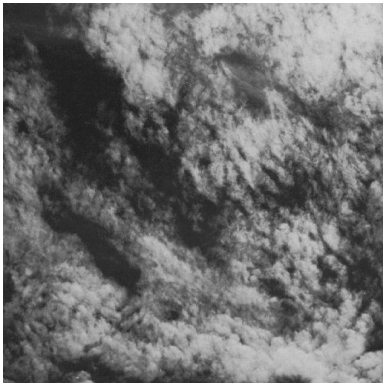
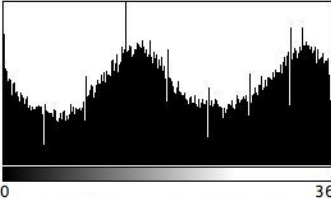
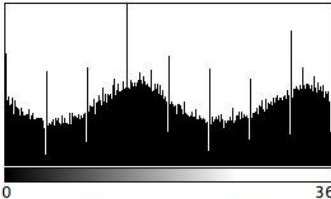
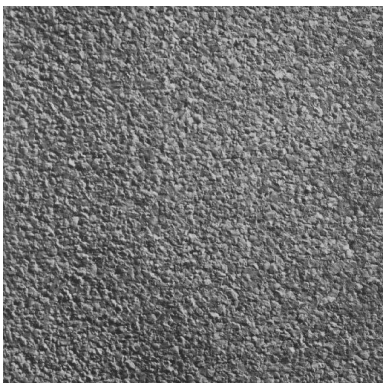
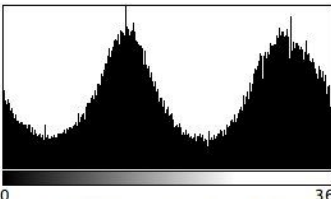
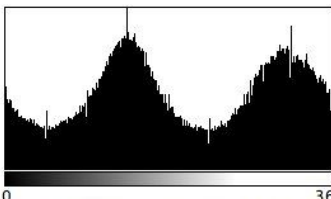
The Magnitude image clearly outlines the edges in the original image. The phase image is a bit harder to read, but it shows the angle of the slope of each point of the original image.

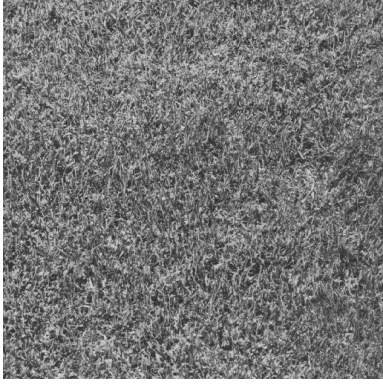
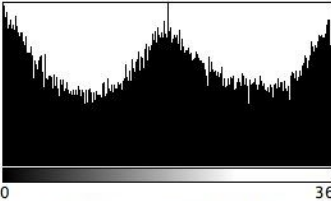
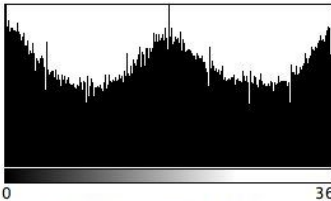
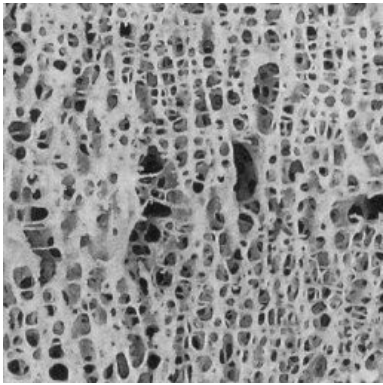
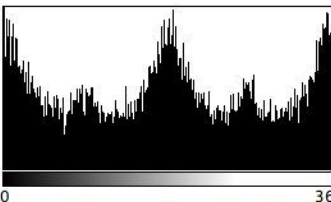
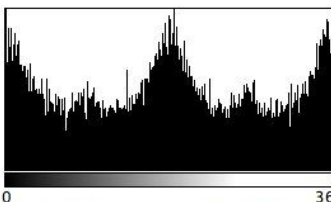
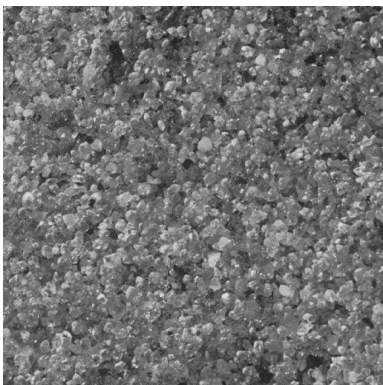
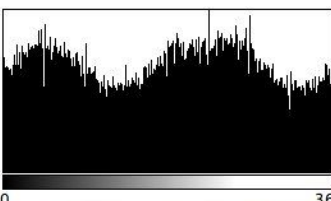
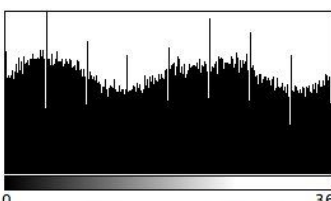
## 2. Sobel Edgeness

This vector field can be used to determine the “edgeness” of a particular point in the image; a point with a larger vector will have steeper slope and therefore be a harder edge. But what constitutes an edge? It is easy to point out edges by eye, but how can we determine what which points are edges in software? A simple way to accomplish this is via thresholding. When thresholded, any point that has an above-average gradient magnitude will be counted as an edge. Optionally, this thresholding can be skipped in order to use all data points as edges.

## 3. Calculating the Signature

Once we have determined which points in the image are edges, we can compute a signature from those edges. This is done by partitioning the edges into  $N$  buckets based on their phase values. As such, each bucket will represent a set of edges that flow in mostly the same direction. When  $N = 6$ , edges with angles  $[0^\circ, 30^\circ)$  will be in the first bucket,  $[30^\circ, 60^\circ)$  in the next and so on. The magnitudes of all the edges in each bucket are then summed. These sums are the buckets’ values. The buckets are then displayed in histogram format for the user.

	 <p>Count: 262144    Min: 62226.847  Mean: 94791.742    Max: 151841.02  StdDev: 20077.364    Mode: 0 (0)  Bins: 256    Bin Width: 1.406</p>	 <p>Count: 262144    Min: 80350.817  Mean: 126446.49    Max: 196069.06  StdDev: 23655.532    Mode: 0 (0)  Bins: 256    Bin Width: 1.406</p>
Original (BUBBLES)	Signature	Sig. No Thresh.
	 <p>Count: 262144    Min: 7129.747  Mean: 28820.271    Max: 59061.082  StdDev: 8543.914    Mode: 0 (0)  Bins: 256    Bin Width: 1.406</p>	 <p>Count: 262144    Min: 7129.747  Mean: 40962.269    Max: 111129.60  StdDev: 11590.950    Mode: 0 (0)  Bins: 256    Bin Width: 1.406</p>
Original (CLOUD)	Signature	Sig. No Thresh.
	 <p>Count: 262144    Min: 26520.757  Mean: 87545.827    Max: 193543.58  StdDev: 43663.777    Mode: 0 (0)  Bins: 256    Bin Width: 1.406</p>	 <p>Count: 262144    Min: 44673.693  Mean: 129205.06    Max: 274456.45  StdDev: 50574.858    Mode: 0 (0)  Bins: 256    Bin Width: 1.406</p>
Original (CORK)	Signature	Sig. No Thresh.

	 <p>Count: 262144    Min: 57766.289  Mean: 91091.891    Max: 152585.83  StdDev: 21108.661    Mode: 0 (0)  Bins: 256    Bin Width: 1.406</p>	 <p>Count: 262144    Min: 83030.667  Mean: 134356.91    Max: 216257.19  StdDev: 25788.808    Mode: 0 (0)  Bins: 256    Bin Width: 1.406</p>
Original (GRASS)	Signature	Sig. No Thresh.
	 <p>Count: 65536    Min: 8914.445  Mean: 21738.163    Max: 45718.076  StdDev: 7878.396    Mode: 0 (0)  Bins: 256    Bin Width: 1.406</p>	 <p>Count: 65536    Min: 14801.878  Mean: 29980.490    Max: 59496.841  StdDev: 9419.004    Mode: 0 (0)  Bins: 256    Bin Width: 1.406</p>
Original (NONOS160)	Signature	Sig. No Thresh.
	 <p>Count: 262144    Min: 30061.170  Mean: 51178.726    Max: 77212.194  StdDev: 8417.843    Mode: 0 (0)  Bins: 256    Bin Width: 1.406</p>	 <p>Count: 262144    Min: 37110.477  Mean: 75550.416    Max: 124883.47  StdDev: 10745.491    Mode: 0 (0)  Bins: 256    Bin Width: 1.406</p>
Original (SAND)	Signature	Sig. No Thresh.

## 4. Our Implementation

Our implementation consists of two plugins. The first is *Sobel.ijm* which is an ImageJ macro that computes the magnitude and phase images. It was our first attempt at implementing the problem, but we figured it would be easier to switch to java so that we could use the built in *HistogramWindow* class. It was left in as it generates nice examples of the Sobel Filter. The second is *SobelEdgeness\_.java* which computes the sobel edgeness signature and displays the result using the built in histogram.

## 5. Future Improvements

For future implementation, we would like to improve on ImageJ's Histogram class or write our own Histogram class. It would be useful to have a live histogram so that whenever the image changes, the signatures updates as well. We would also like to add a feature where you could compare the Sobel Edgeness signatures of multiple images.