2-DIMENSIONAL COLORED NOISE SYNTHESIS

By Laurence G. Hassebrook

2-14-2012, 2-21-2017

Please follow the tutorial and reproduce the figures with your own code.

1. Rectangular and Circular Binary Transfer Functions

Define the basic parameters of this visualization as

```
% 2-Dimensional Stationary Colored Noise Filters
clear all;
% dimensions of image
Nx=512;
My=Nx;
% Filter low and high cutoff frequency ranges
Bandwidthfactor=0.2;
aspect=1;
seathresh=0;
%
TcHighX=floor(1+2*Bandwidthfactor*(Nx/2))
TcHighY=floor(TcHighX*aspect)
sigmax=(TcHighX-1)/2
sigmay=(TcHighY-1)/2
% Fractional Power or Fractal filter
BetaFractalx=2
BetaFractaly=2
% Illumination direction
ThetaX=45;% illumination angle parallel to X dim
ThetaY=-45; % illumination angle parallel to Y dim
```

Generate a 0 mean, unity variance Gaussian noise image "*w*." Use a 2-D DFT to create the DFT transform of *w* to be *W*.

```
% Input noise
w=randn(My,Nx);
W=fft2(w);
stdw=mean(std(w))
```

Create a rectangular window (ie. irect) to be used to select a rectangular area of frequencies of the input, in the frequency domain.

```
% RECTANGULAR FILTER
% Form a Rectangular Donut
H1rect=irect(TcHighY,TcHighX,My,Nx);
% scale H such that the output variance
% is unity for each pixel given unity input
Rxx=ifft2(H1rect.^2);
denom=sqrt(abs(Rxx(1,1)));
H1rect=H1rect./denom;
%
```

```
figure(1)
imagesc(H1rect)
colormap gray;
axis image;
axis off;
title('Rectangular Binary Window')
print -djpeg fig1
% convolve input noise with filter
wcolor=real(ifft2(W.*H1rect));
wmax=max(max(wcolor))
wmin=min(min(wcolor))
stdRect=mean(std(wcolor))
%
wmetalic=metalic(wcolor,1,ThetaX,ThetaY);
```

Likewise, create a circular shaped filter using icirc.

```
% CIRC filter
% Form a Circular Window
H1circ=icirc(TcHighY,TcHighX,My,Nx);
% scale H such that the output variance
% is unity for each pixel given unity input
Rxx=ifft2(H1circ.^2);
denom=sqrt(abs(Rxx(1,1)));
H1circ=H1circ./denom;
% convolve input noise with filter
wcolor=real(ifft2(W.*H1circ));
%
wmetalic=metalic(wcolor,1,ThetaX,ThetaY);
```

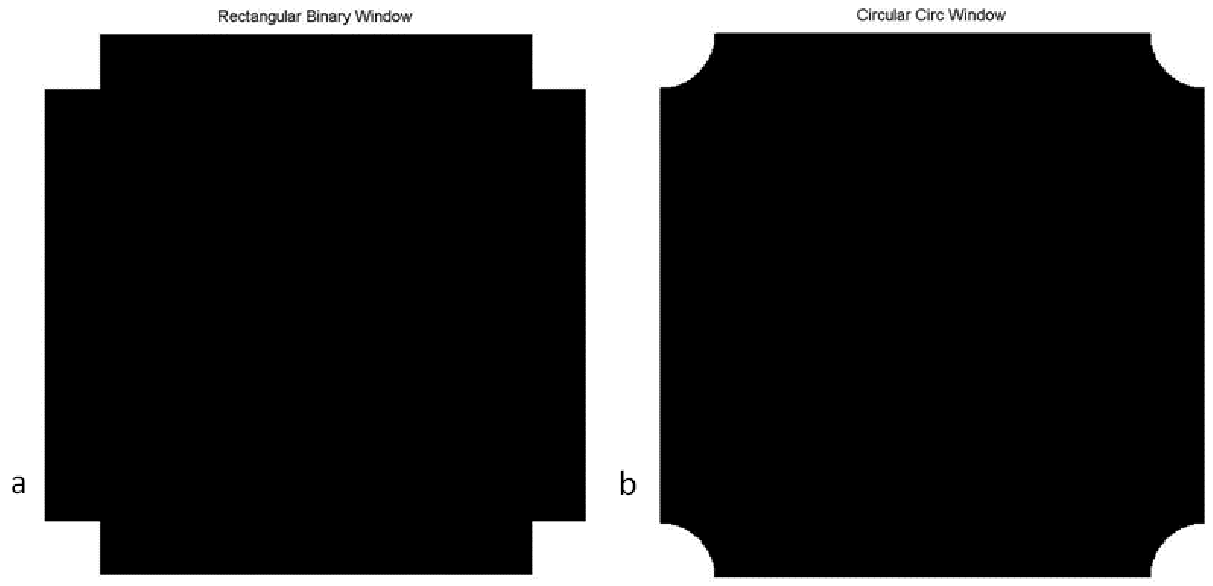


Figure 1: (a) Seperable rectangular filter transfer function. (b) Circular symmetric filter transfer function.

Multiply the filters in Fig. 1 to generate the noise patterns shown in Fig. 2

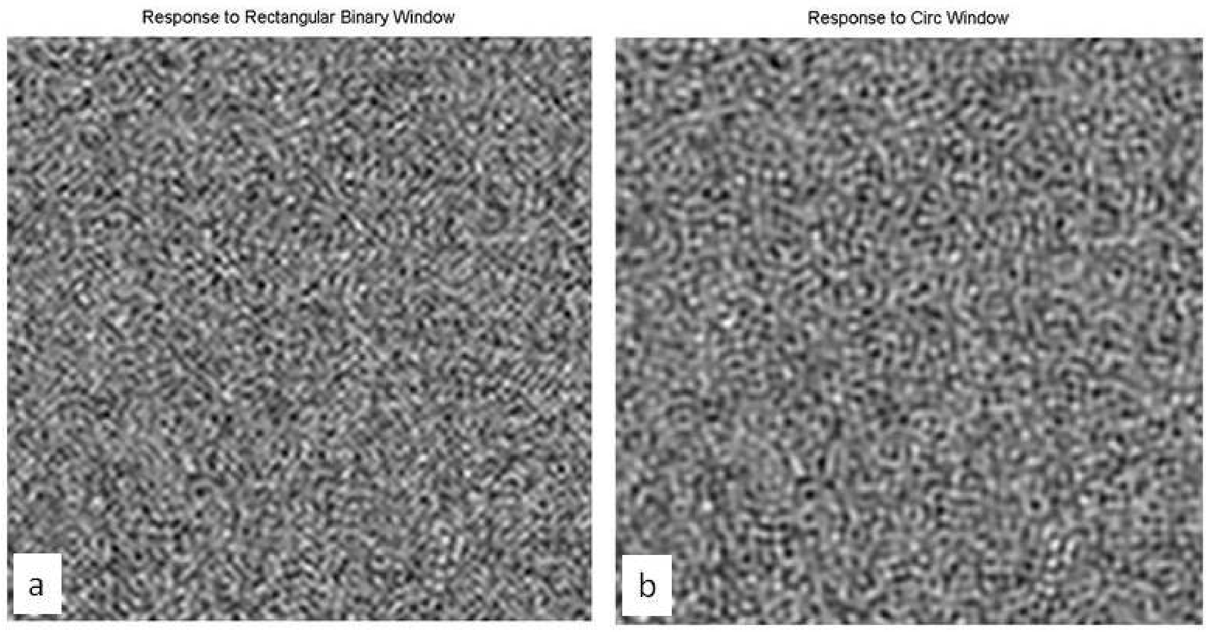


Figure 2: Outputs from the filters in Fig. 1, given the same Gaussian noise input.

Note that in Fig. 2 (a) the patterns have straighter alignment versus the more circular or curved shapes in Fig. 2 (b). This difference is due to the direction bias the rectangular filter shape has on the output noise distribution. To visualize the noise characteristics further we use a “surface norm” technique that treats the image as a 3-D surface where Z is the intensity. At each point a surface normal vector is formed. Assuming the viewing direction is along the Z direction, the illumination angle is controlled by an angleX and angleY parameter. The result is analogous to a metallic surface reflection of a collimated illumination where the reflection is controlled by the surface normals and the illumation angle. The results for Fig 2 are shown in Fig. 3

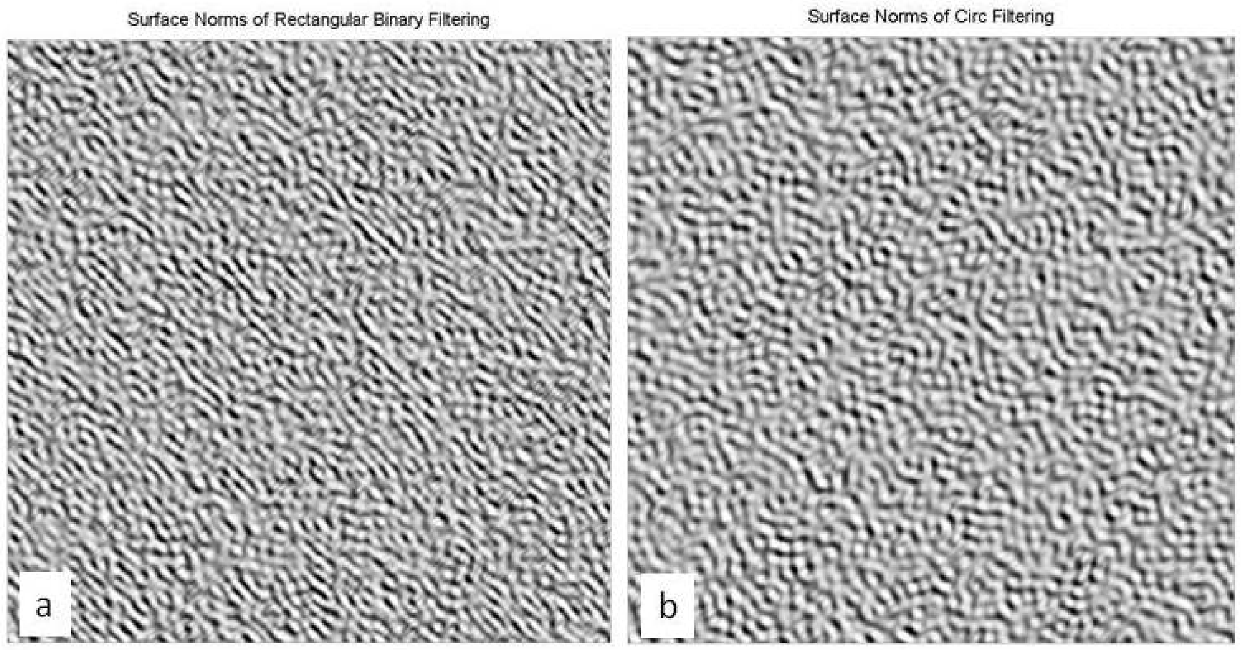

Surface Norms of Rectangular Binary Filtering
Surface Norms of Circ Filtering
a
b

Figure 3: Surface norms for the noise outputs in Fig. 2.

2. Gaussian Transfer Function

In many ways, we can see “worm” like structure in surface norms of Fig. 3. Now consider a Gaussian filter/transfer function that will include a weighted combination of low and high frequencies. The Guassian filter is both separable and circular symmetric. The filter is shown in Fig. 4.

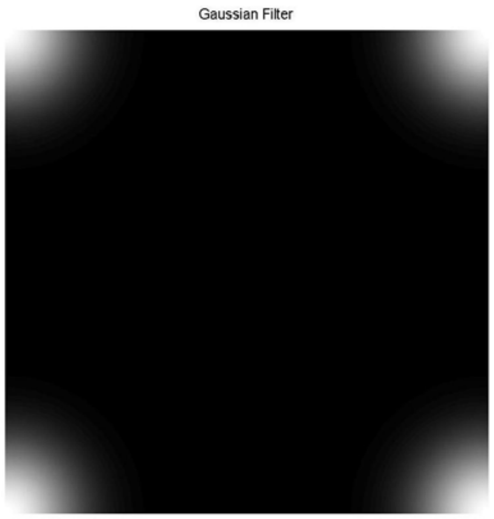


Figure 4: The Gaussian shaped transfer function.

The resulting noise output is shown in Fig 5 (a) and its surface norms in Fig. 5(b) where the surface has less deterministic looking structure.

```
% GAUSSIAN FILTER
Hgauss=igauss(sigmay,sigmax,My,Nx);
% scale H such that the output variance
% is unity for each pixel given unity input
Rxx=ifft2(Hgauss.^2);
denom=sqrt(abs(Rxx(1,1)));
Hgauss=Hgauss./denom;
% convolve input noise with filter
wcolor=real(ifft2(W.*Hgauss));
%
wmetalic=metalic(wcolor,1,ThetaX,ThetaY);
```

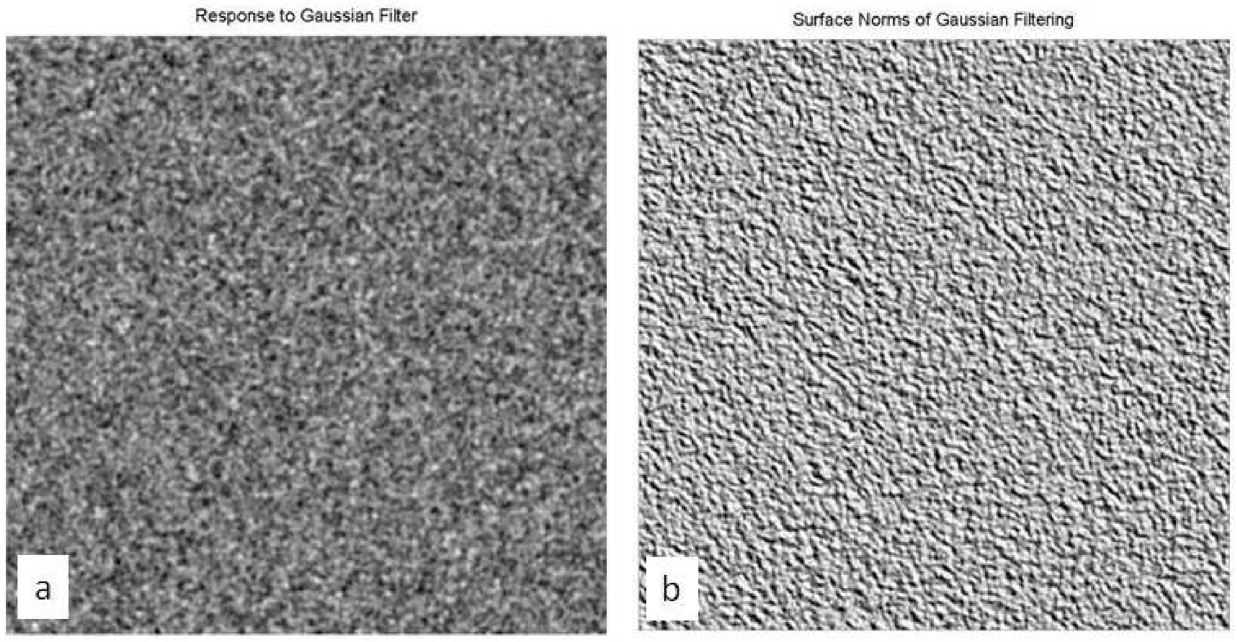

Response to Gaussian Filter
Surface Norms of Gaussian Filtering
a
b

Figure 5: (a) Outpu noise pattern of the Gaussian filter. (b) Surface model of noise pattern.

The noise in Fig. 5 resembles patterns seen in nature such as a microscopic view of a metallic surface.

3. Fractal Transfer Functions

The idea of modeling natural surfaces became very apparent with the development of fractal noise models.[1] The primary idea of fractal patterns is the scaling characteristic corresponding to the slope of the frequency response of $\log(H(f)) = \log(1/f^{\beta}) = -\beta \log(f)$. There is a big difference in the separable fractal filter versus the circularly symmetric one as shown in Fig. 6.

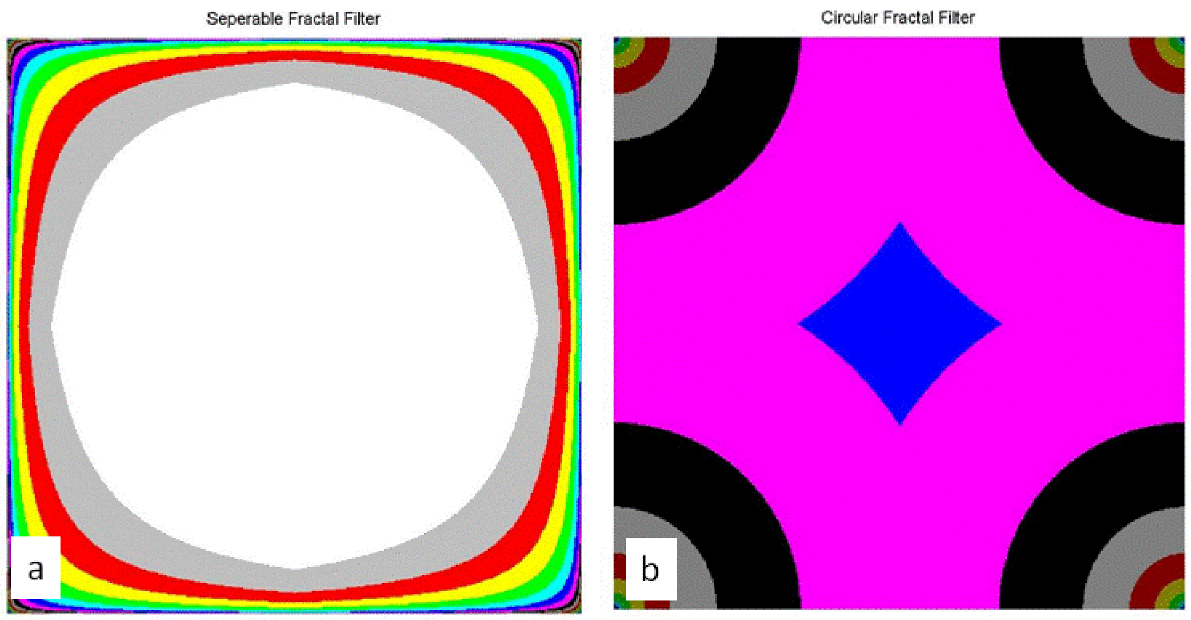


Figure 6: (a) Seperable fractal transfer function and (b) circularly symmetric fractal transfer function.

The fractal transfer functions in Fig. 6 are log() and color coded because their structures are difficult to view in gray level.

```
% SEPERABLE FRACTAL FILTER
Hfractal=iHfractal(BetaFractalx,BetaFractaly,My,Nx,1);
% scale H such that the output variance
% is unity for each pixel given unity input
Rxx=ifft2(Hfractal.^2);
denom=sqrt(abs(Rxx(1,1)));
Hfractal=Hfractal./denom;
%
figure(10)
imagesc(log(Hfractal+.000001))
colormap vga;%gray;
axis image;
axis off;
title('Seperable Fractal Filter')
print -djpeg fig10
% convolve input noise with filter
wcolor=real(ifft2(W.*Hfractal));
%
wmetalic=metalic(wcolor,1,ThetaX,ThetaY);
% CIRCULAR FRACTAL FILTER
Hfractal=iHfractal(BetaFractalx,BetaFractaly,My,Nx,0);
% scale H such that the output variance
% is unity for each pixel given unity input
Rxx=ifft2(Hfractal.^2);
denom=sqrt(abs(Rxx(1,1)));
Hfractal=Hfractal./denom;
%
figure(13)
imagesc(log(Hfractal+.000001))
% convolve input noise with filter
```

```
wcolor=real(ifft2(W.*Hfractal));
%
wmetalic=metalic(wcolor,1,ThetaX,ThetaY);
```

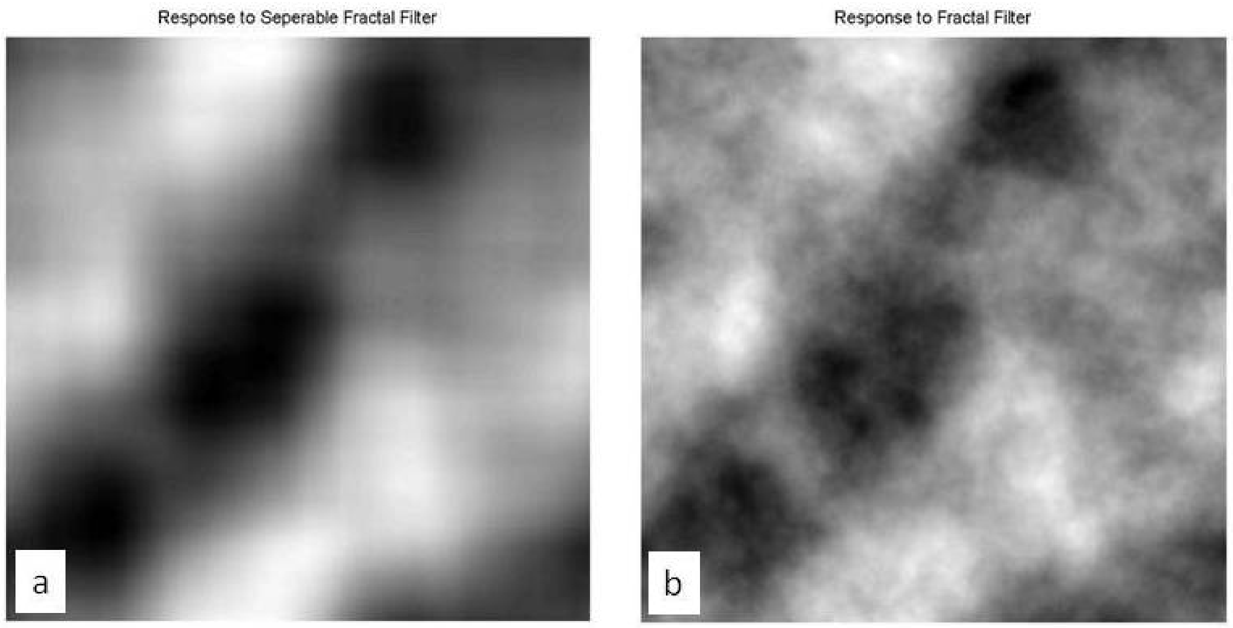

Response to Seperable Fractal Filter
Response to Fractal Filter
a
b

Figure 7: (a) Separable fractal response and (b) circularly symmetric fractal response.

The noise outputs of the two fractal filters are shown in Fig. 7. Notice that at first glance, they appear much lower frequency than the noise response in Fig. 2.

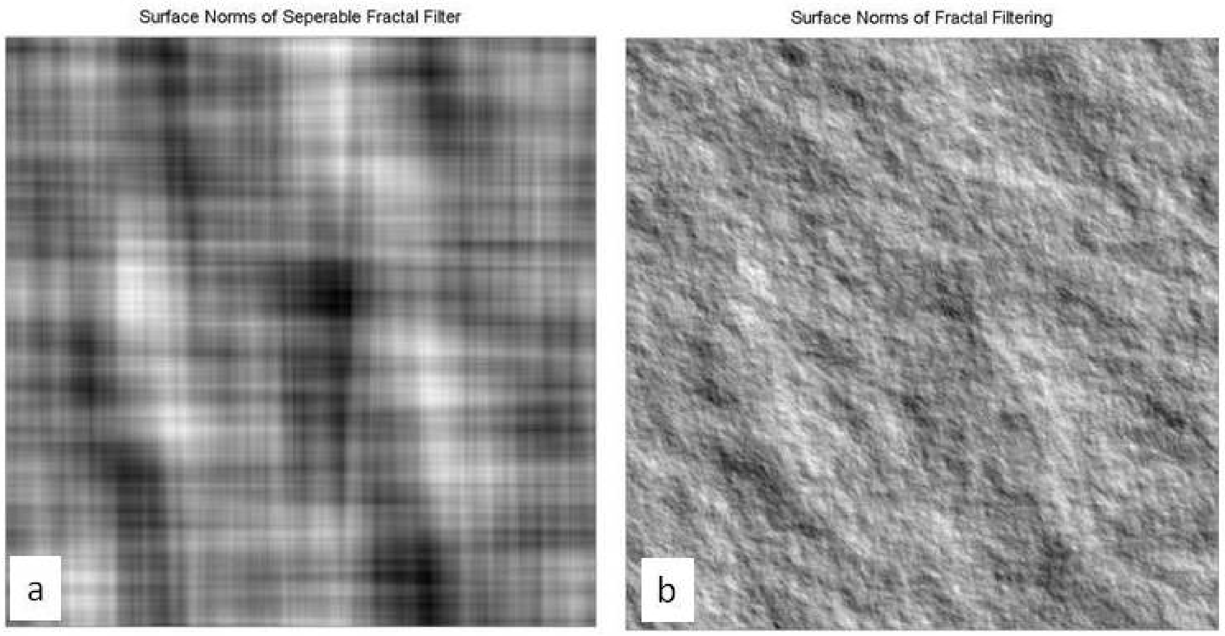

Surface Norms of Seperable Fractal Filter
Surface Norms of Fractal Filtering
a
b

Figure 8: (a) Separable fractal response surface model and (b) surface model of circular symmetric fractal response.

The surface models of the patterns in Fig. 7, shown in Fig. 8, reveal the higher frequency as well as the distinguishing characteristics of the separable versus circularly symmetric responses. In the 1970s, it became apparent that our perception of roughness requires a reference surface. So the fractal researchers demonstrated this by putting in “lakes” and “seas” into the noise mixture to give a more consistent perception of roughness. Taking the data in Fig. 7(b) and applying a threshold to segment the noise above and replacing the other with a constant “sea” level demonstrates this affect. In Fig. 9 (a) we see a pattern that resembles clouds and in Fig. 9 (b) the pattern resembles islands versus lakes and oceans.

```
% VISUALIZE ROUGHNESS BY "SEA" LEVEL THRESHOLD
Isea=find(wcolor<seathresh);
wcolor(Isea)=seathresh;
%
wmetalic=metalic(wcolor,1,ThetaX,ThetaY);
```

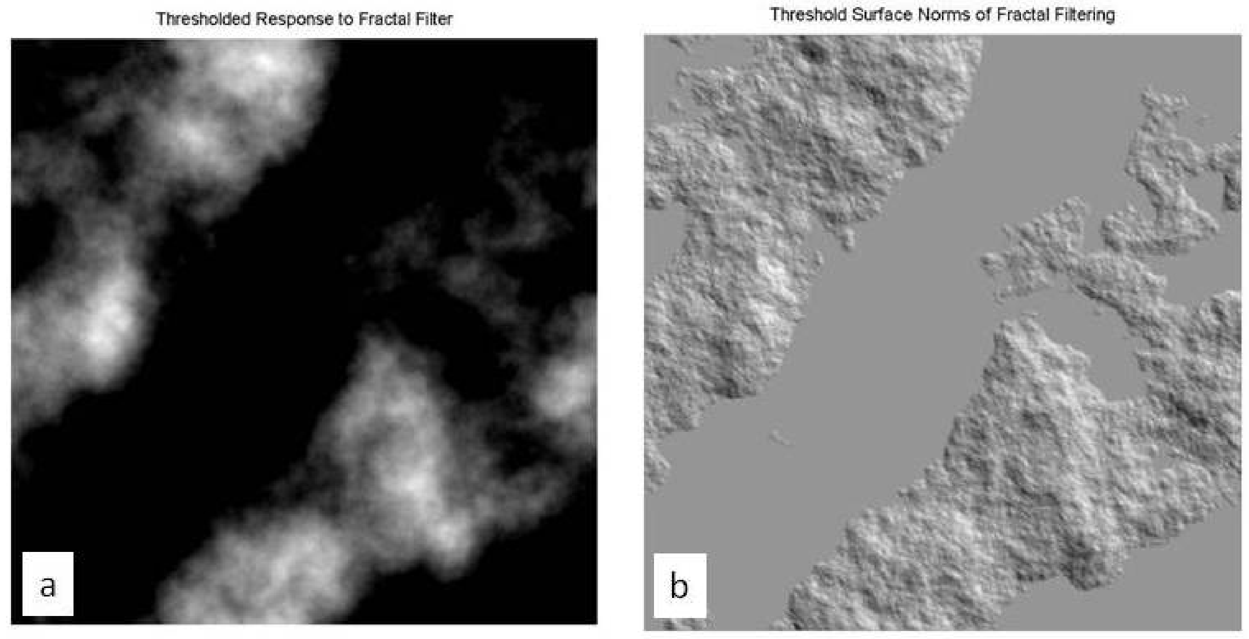
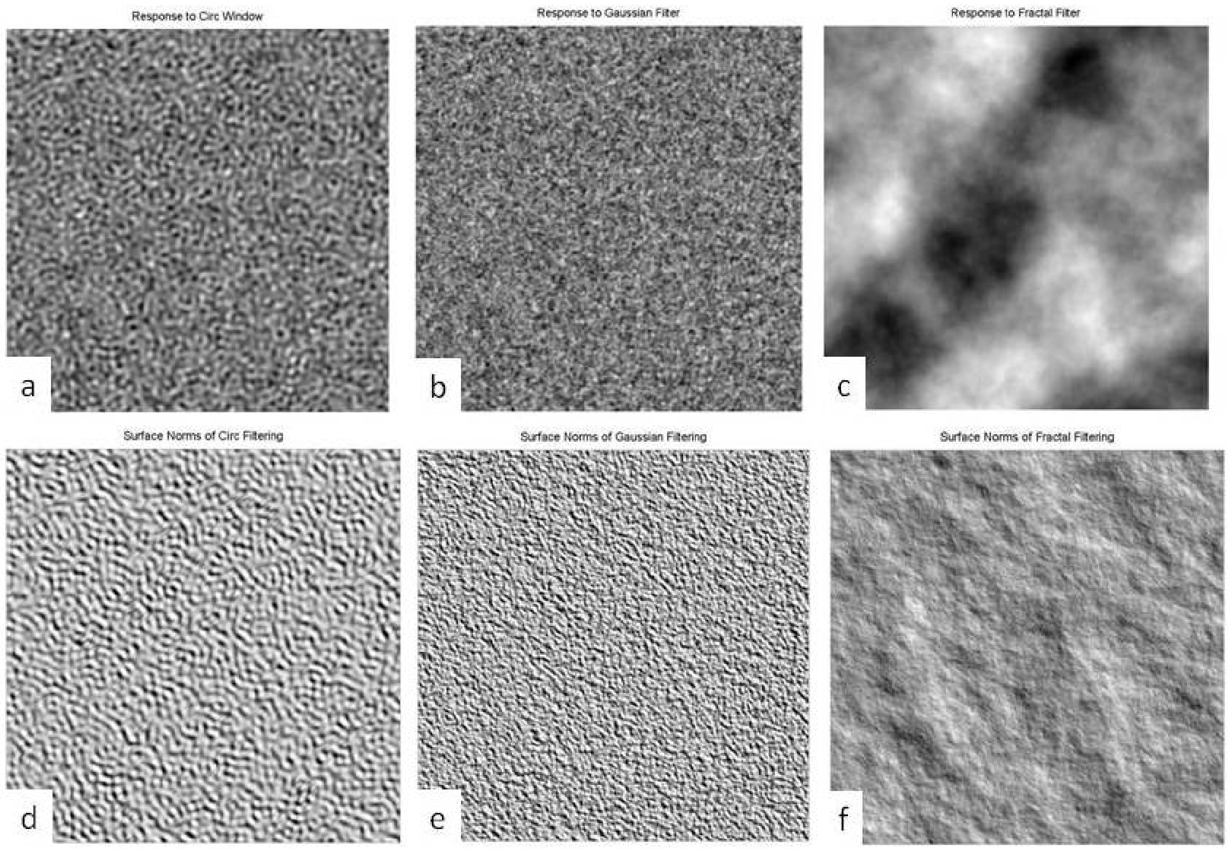

Thresholded Response to Fractal Filter
a
Threshold Surface Norms of Fractal Filtering
b

Figure 9: (a) Thresholded noise response of Fig. 7b and (b) the surface model of the thresholded noise response.

To summarize the visualization of the different noise types, all the circularly symmetric responses are shown in Fig. 10 where the top row is the gray level of the response and the bottom row is the surface model.


Response to Circ Window
Response to Gaussian Filter
Response to Fractal Filter
a
b
c
Surface Norms of Circ Filtering
Surface Norms of Gaussian Filtering
Surface Norms of Fractal Filtering
d
e
f

It can be seen in Fig. 10, that the fractal approach to filtering introduced a dramatic change in the noise characteristics. Researchers of the time realized the implications of this type of noise in modeling natural signals, imagery and surfaces.

4. REFERENCES

1. Benoit B. Mandelbrot, The Fractal Geometry of Nature, W.H Freeman and Company, New York. Copyright 1977, ISBN 13: -1186-5978-0-7167

APPENDIX

A.1 Filter Normalization for Unity Output Variance

We derive the normalization coefficient that will yield a variance of unity for each output pixel given the input pixel variance is unity.

Given a 1-D discrete-time input is a white Gaussian noise sequence $\boldsymbol{w}[n]$ where E{$\boldsymbol{w}[n]$}=0 and E{$\boldsymbol{w}^2[n]$}=1, consider output $\boldsymbol{y}[n]$ and filter $h[n]$. The result will apply to a 2-D discrete-space image. The noise system is given by:

$$\tilde{y}[t] = \sum_{\lambda} \tilde{w}[\lambda] h[t-\lambda] \qquad \text{(A.1)}$$

From Eq. (A.1) we form the autocorrelation as

$$\tilde{y}[t]\tilde{y}[t+\tau] = \sum_{\lambda}\tilde{w}[\lambda]h[t-\lambda]\sum_{\beta}\tilde{w}[\beta]h[t+\tau-\beta] = \sum_{\beta}\sum_{\lambda}h[t-\lambda]h[t+\tau-\beta]\tilde{w}[\lambda]\tilde{w}[\beta] \qquad \text{(A.2)}$$

Taking the expected value of Eq. (A.2) yields

$$E\{\tilde{y}[t]\tilde{y}[t+\tau]\} = \sum_{\beta}\sum_{\lambda}h[t-\lambda]h[t+\tau-\beta]\iint w_1[\lambda]w_2[\beta]\, f_{ww}(w_1,w_2)\,dw_1dw_2 \qquad \text{(A.3)}$$

The autocorrelation of the input noise is analytical kronecker delta function times the input variance. We will assume the input variance is unity and the kronecker is unity when $\lambda = \beta$ which removes one of the summations such that Eq. (A.2) becomes:

$$E\{\tilde{y}[t]\tilde{y}[t+\tau]\} = \sum_{\beta}\sum_{\lambda}h[t-\lambda]h[t+\tau-\beta]\delta[\lambda-\beta] = \sum_{\beta}h[t-\beta]h[t+\tau-\beta] \qquad \text{(A.4)}$$

The parameter t is just an offset in both factors of the summation so it can be removed and by using a change of variables the sign of the arguments can be reversed such that

$$E\{\tilde{y}[t]\tilde{y}[t+\tau]\} = R_{ww}(\tau) = \sum_{\beta}h[\beta]h[\beta-\tau] \; = h[\beta]*h[-\beta] \qquad \text{(A.5)}$$

The Discrete-Time version of the Wiener-Khinchin theorem yields the power spectral density as

$$S_{ww}(f) = DFT(R_{ww}(\tau)) = DFT(h[\beta]*h[-\beta]) = |H(f)|^2 \qquad \text{(A.6)}$$

Inverting Eq. A.6 yields:

$$R_{ww}(\tau) = DFT^{-1}(S_{ww}(f)) = DFT^{-1}\left(|H(f)|^2\right) \qquad \text{(A.7)}$$

Evaluating Eq. (A.7) at t=0 and gives the normalization coefficient such that

$$R_{ww}(0) = DFT^{-1}\left(|H(f)|^2\right)\Big|_{\tau=0} \qquad \text{(A.7)}$$

So the normalized filter in the frequency domain is the original divided by the square root of its autocorrelation evaluated at $\tau = 0$ such that:

$$H_{norm}(f) = \frac{H(f)}{\sqrt{R_{ww}(0)}} \qquad \text{(A.8)}$$