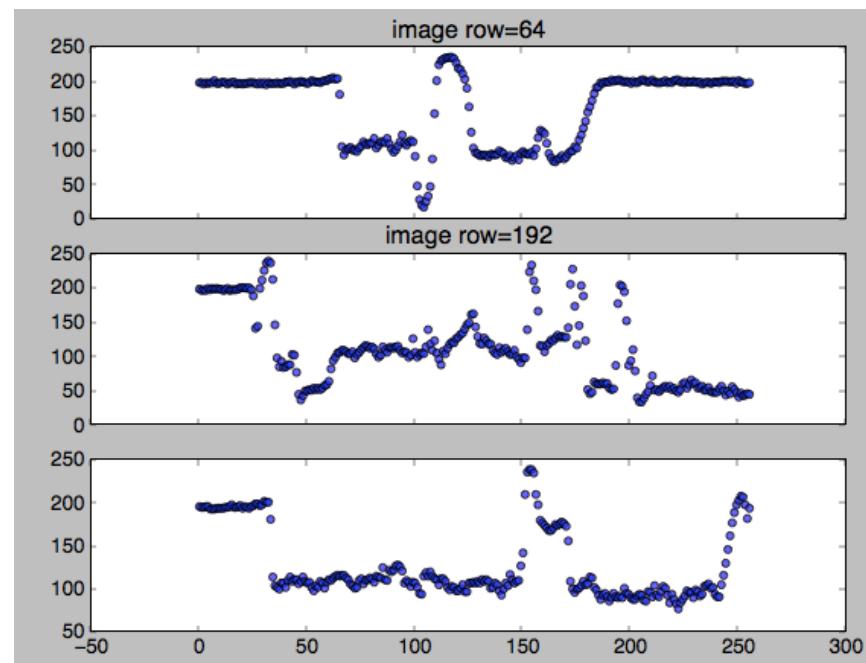
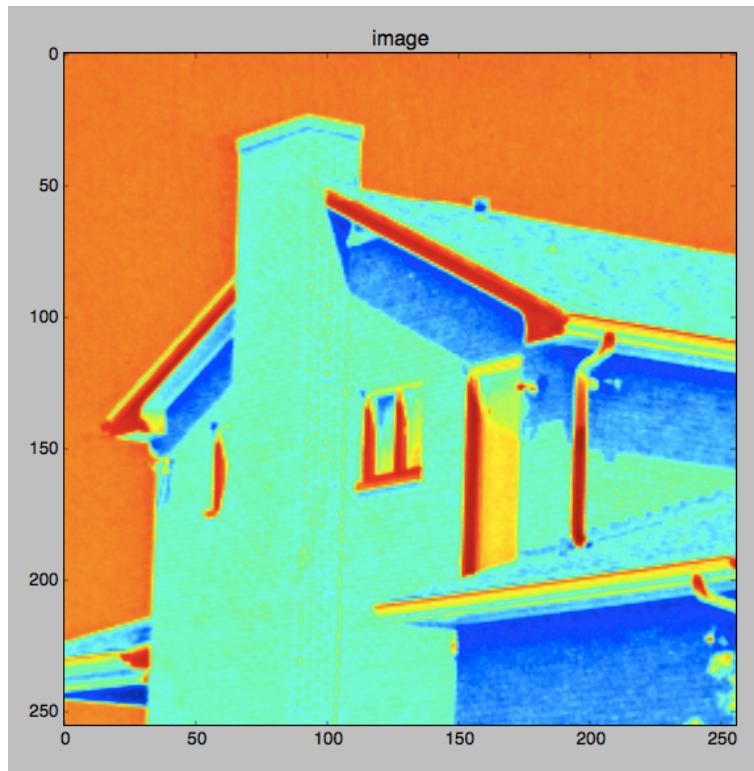


## **plots from the phase congruency algorithm**

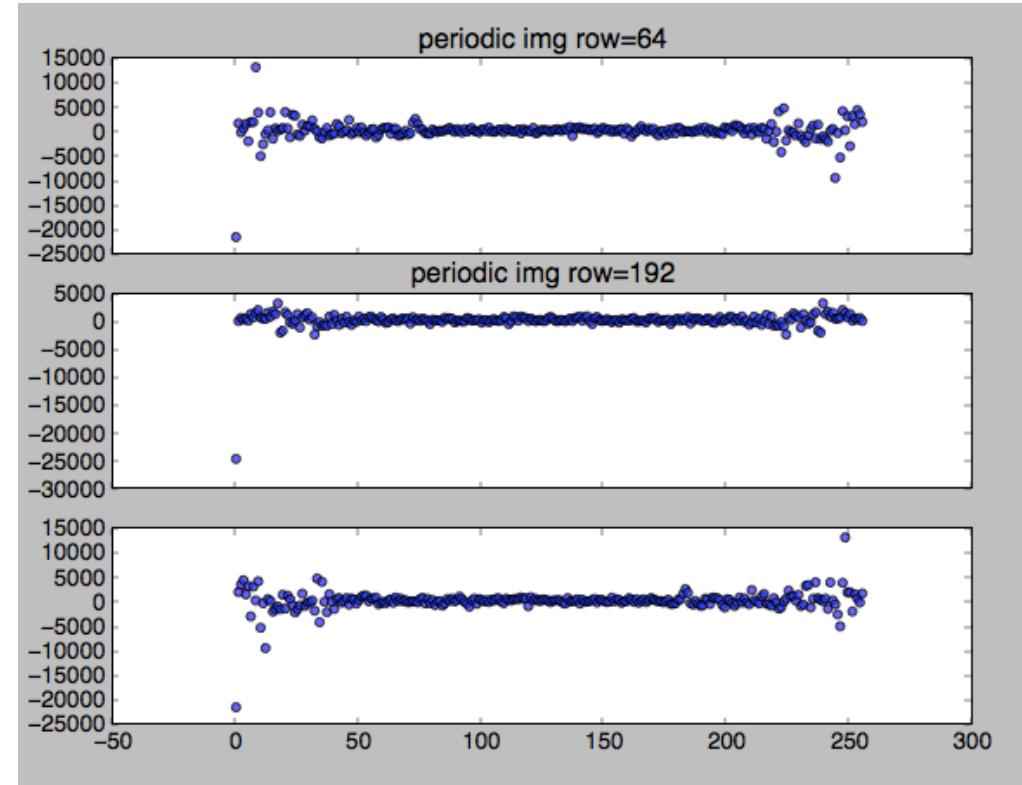
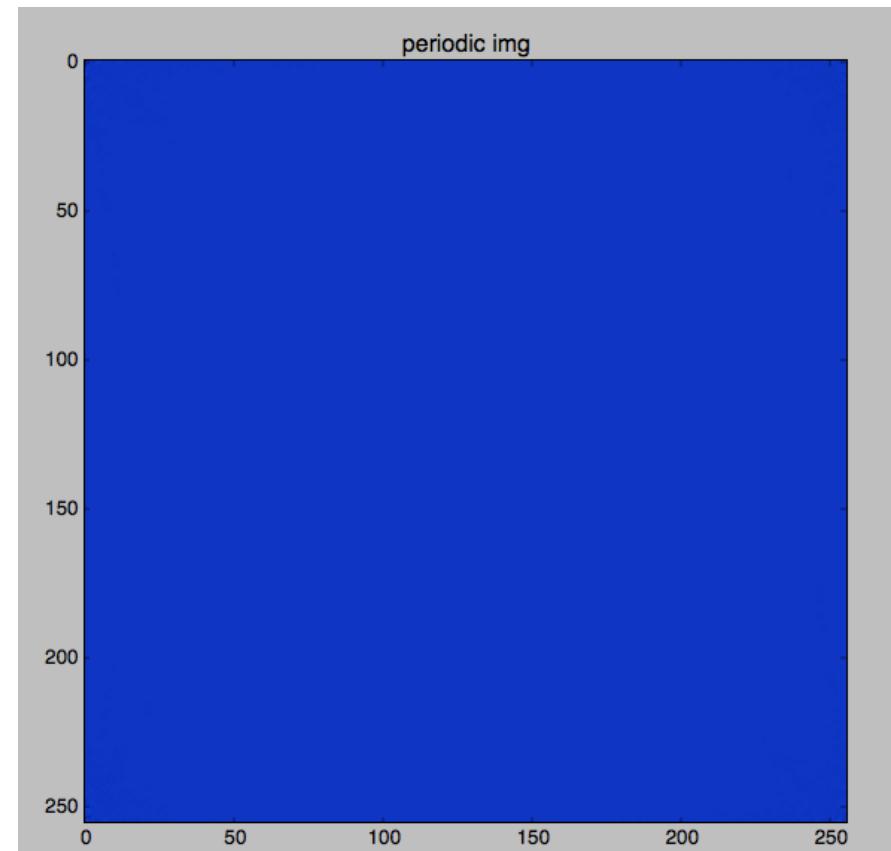
followed by a filter in progress, meant for use in calculating corners  
using curvature (needs 1st and 2nd derives)

house.gif

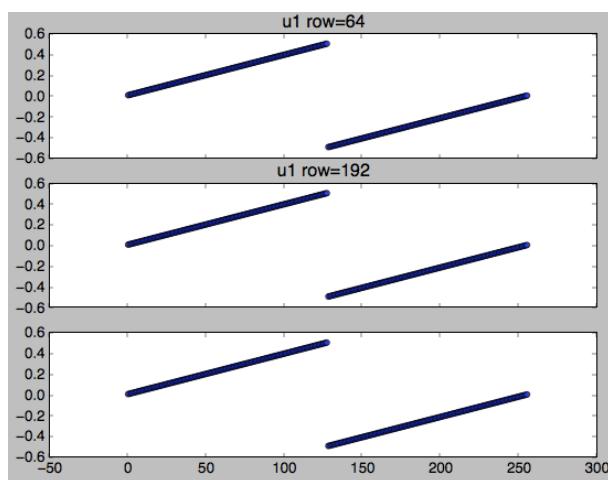


periodic FFT of img

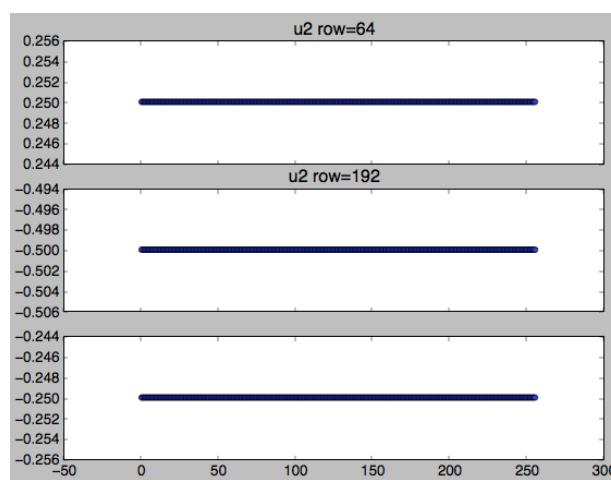
`_ , IM = perfft2(img)`



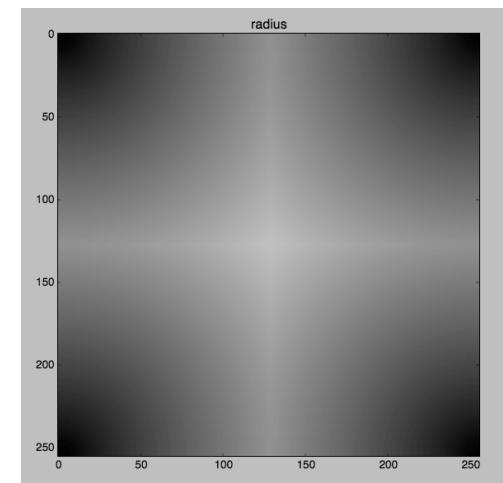
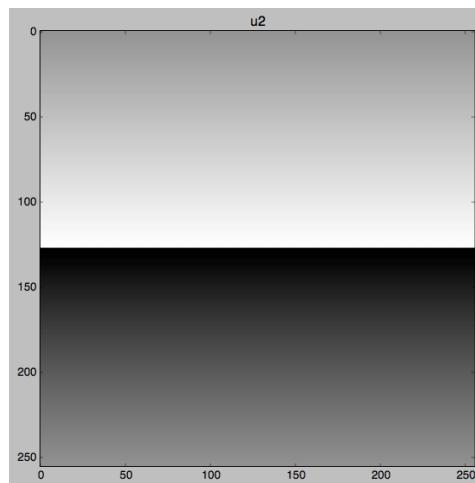
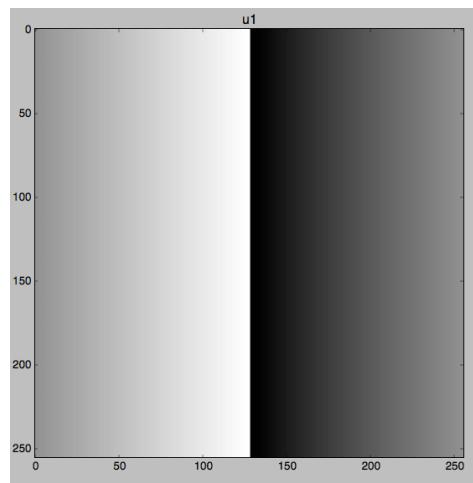
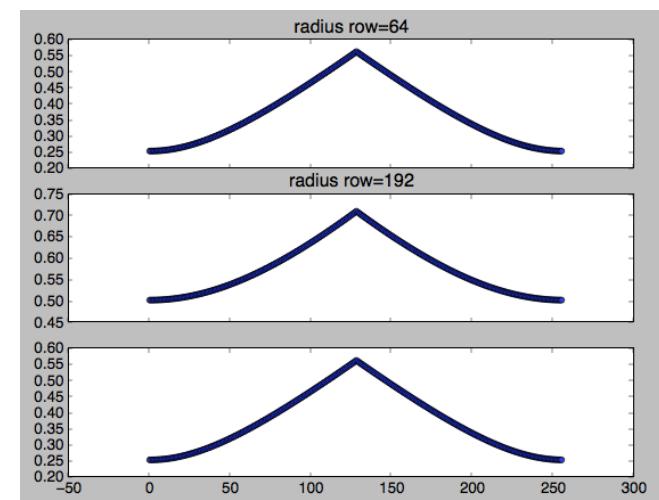
**u1**



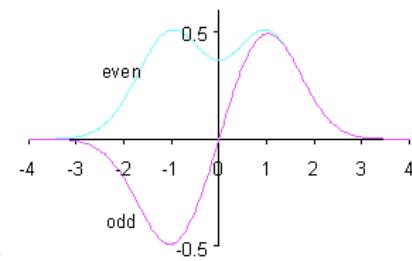
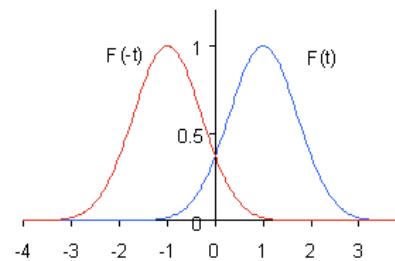
**u2**



**radius**

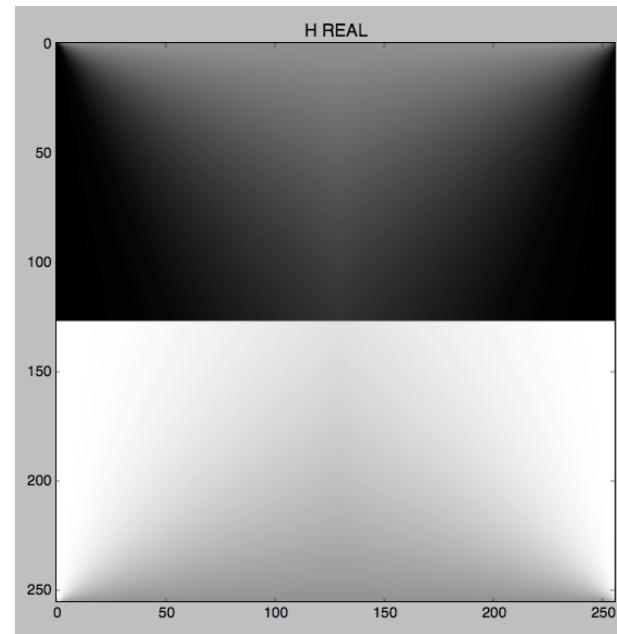
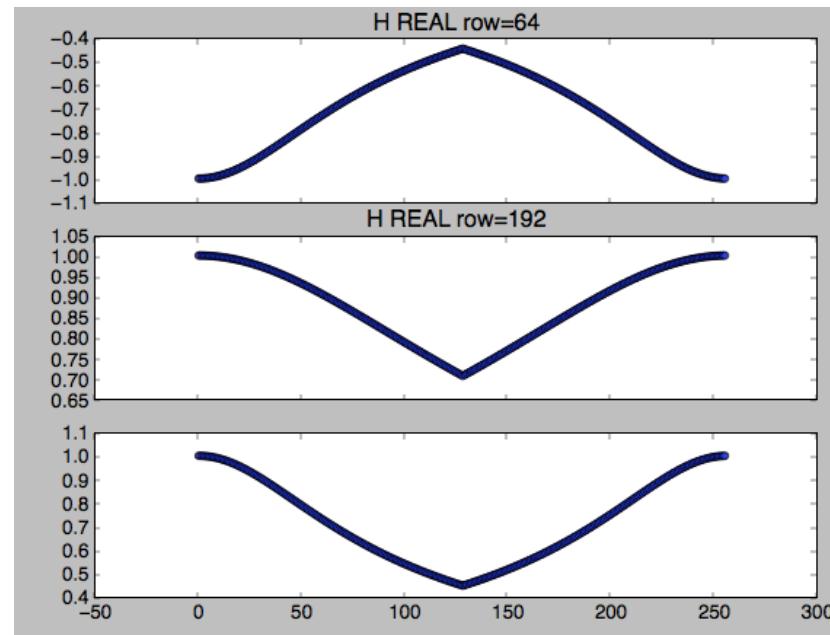


$$F(t) = \exp[-(t - 1)^2]$$

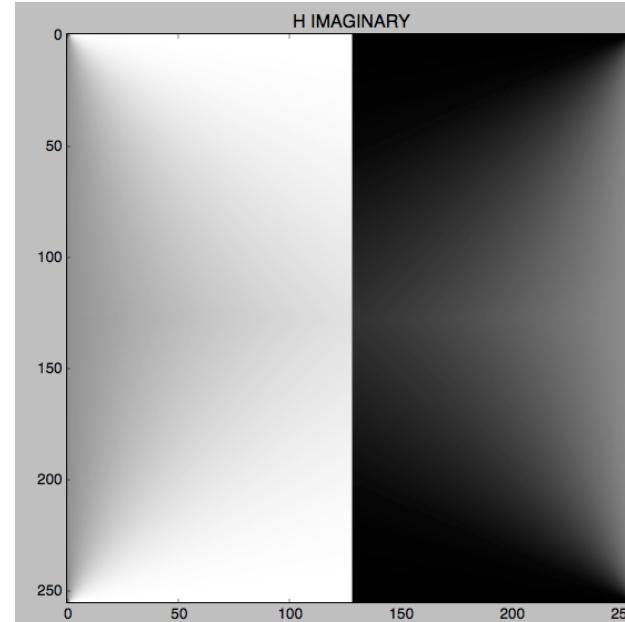
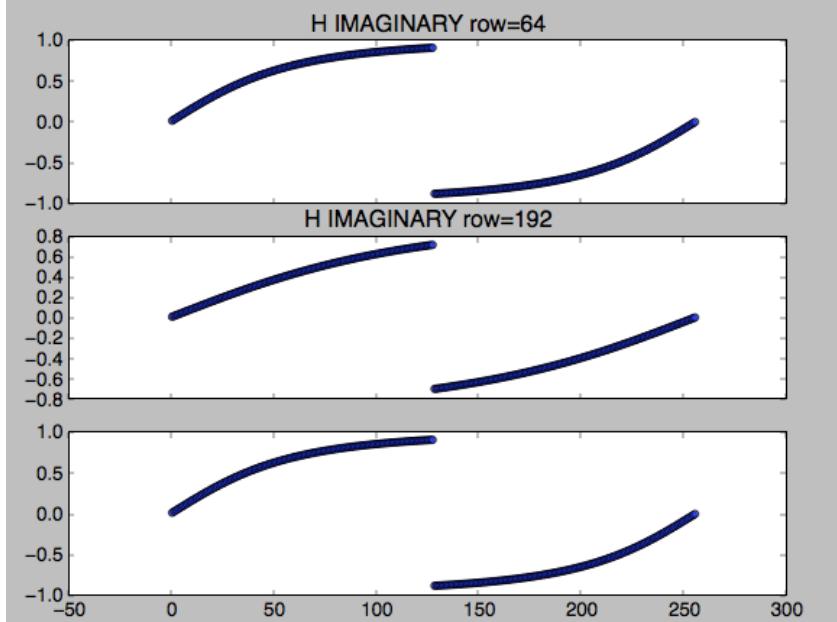


**h1 = real( $\mathbf{H}$ )**

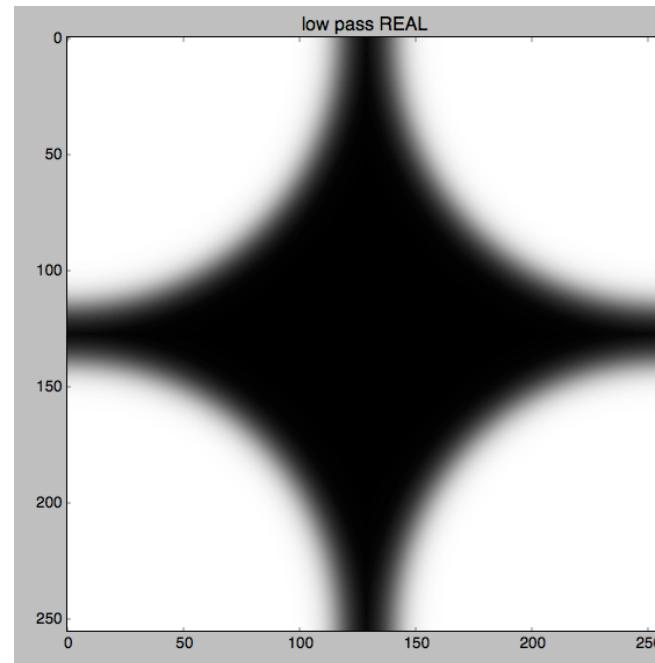
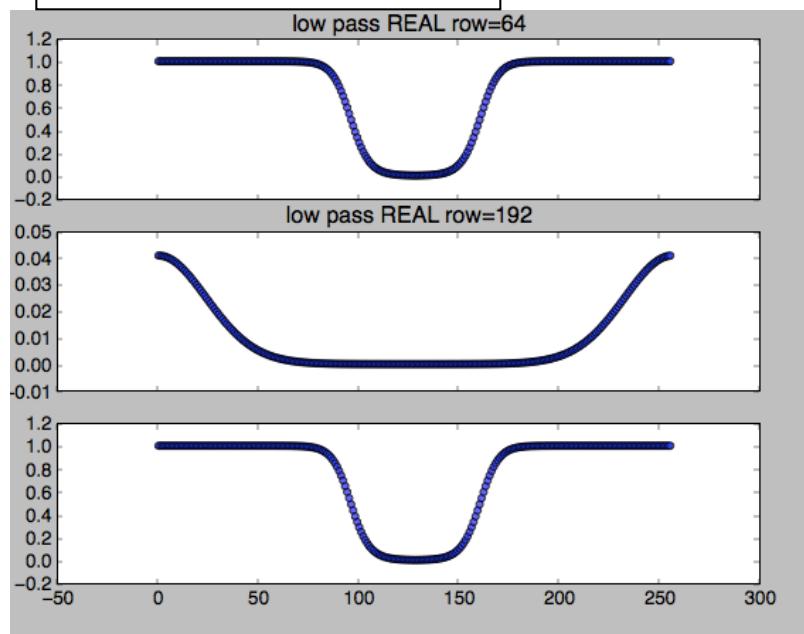
$$\mathbf{H} = (1j * \mathbf{u}_1 - \mathbf{u}_2) / \text{radius}$$



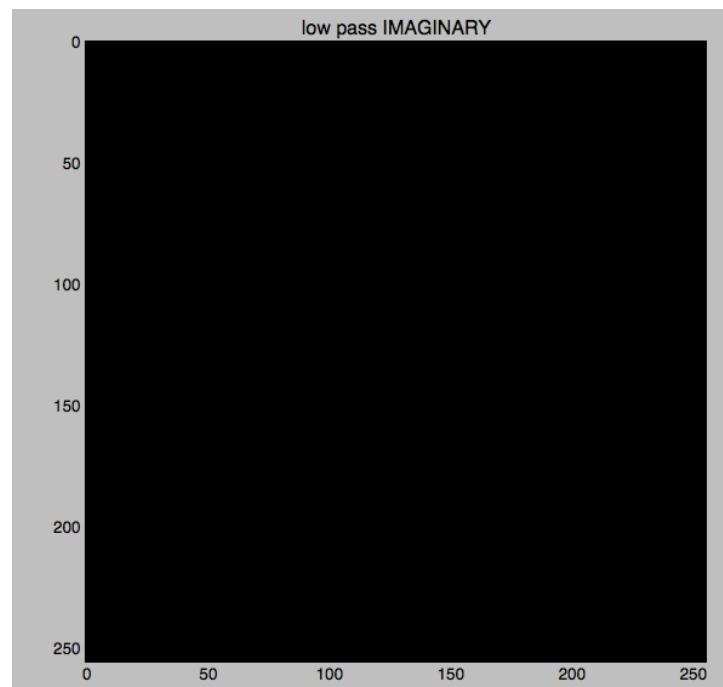
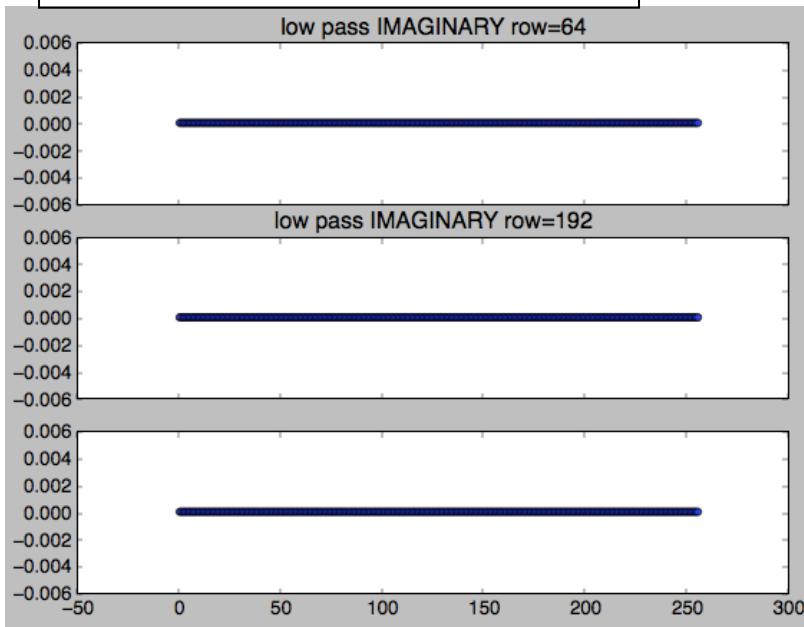
**h2 = imaginary( $\mathbf{H}$ )**



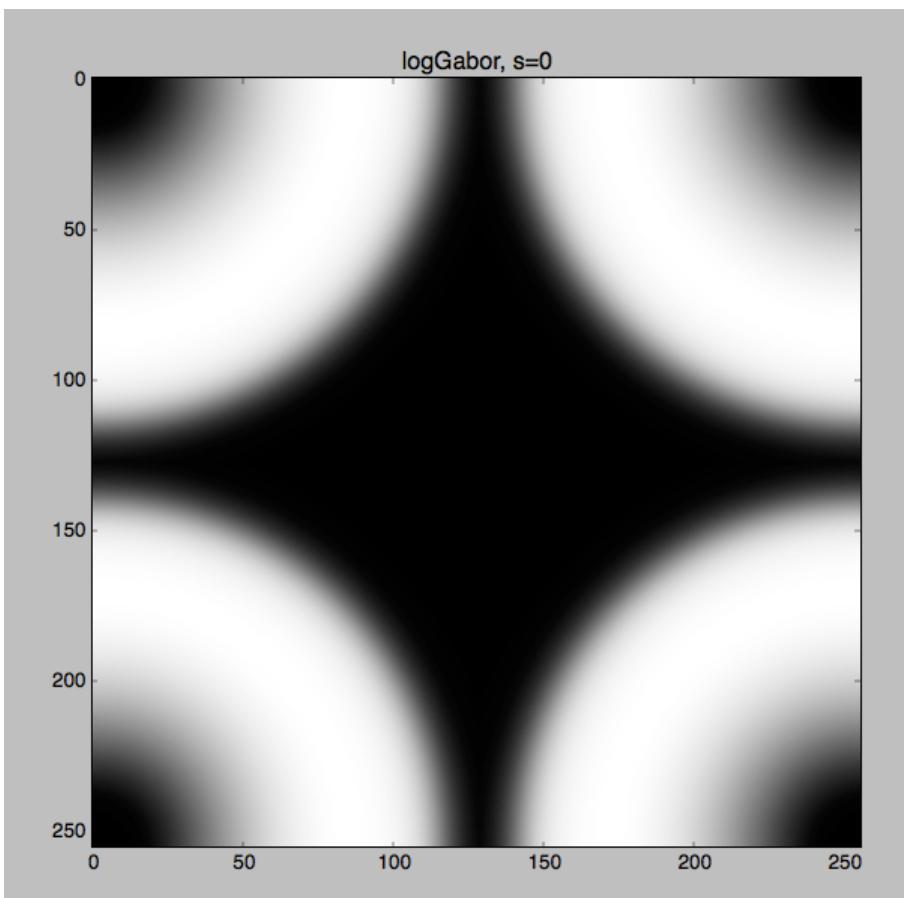
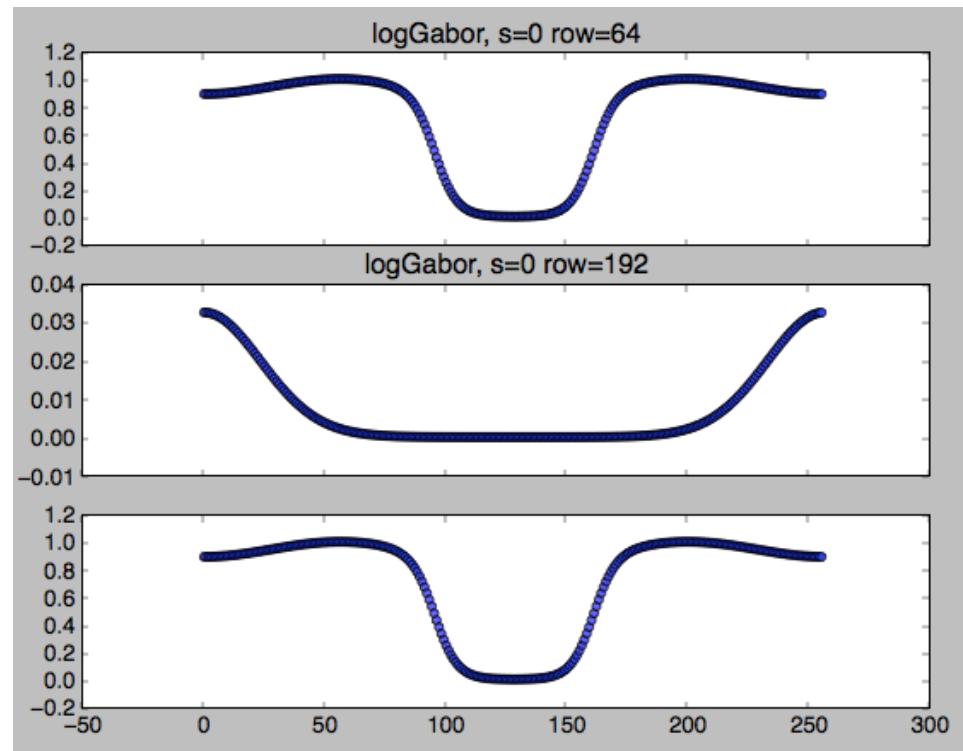
## real(**low pass filter**)



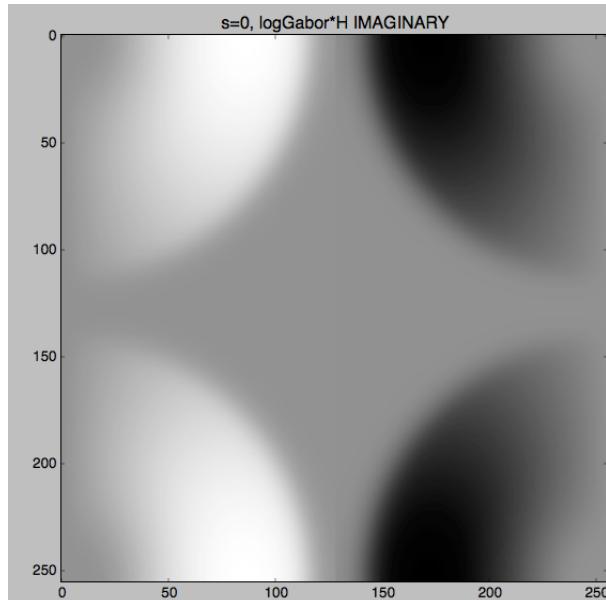
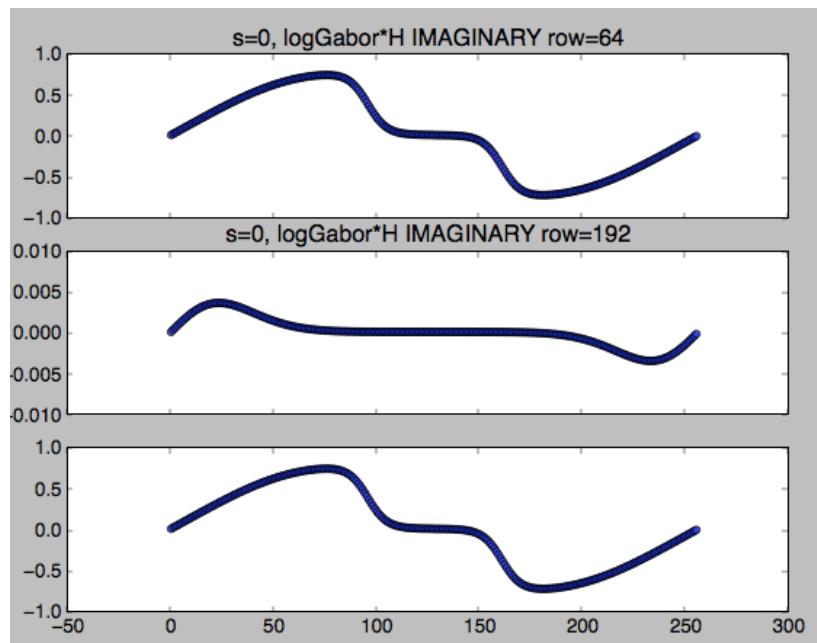
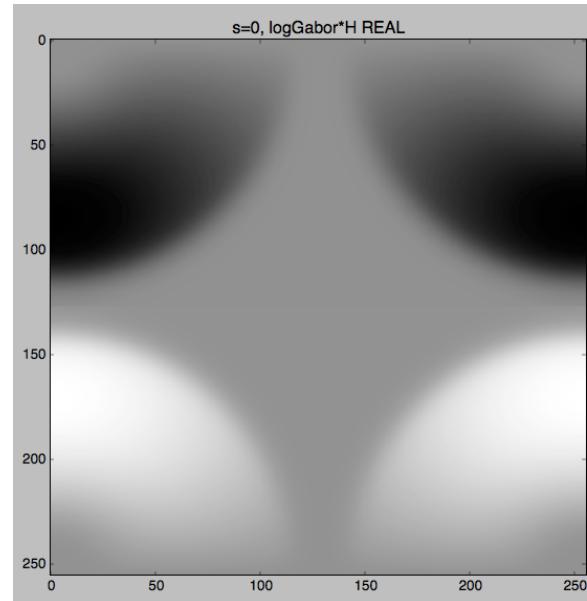
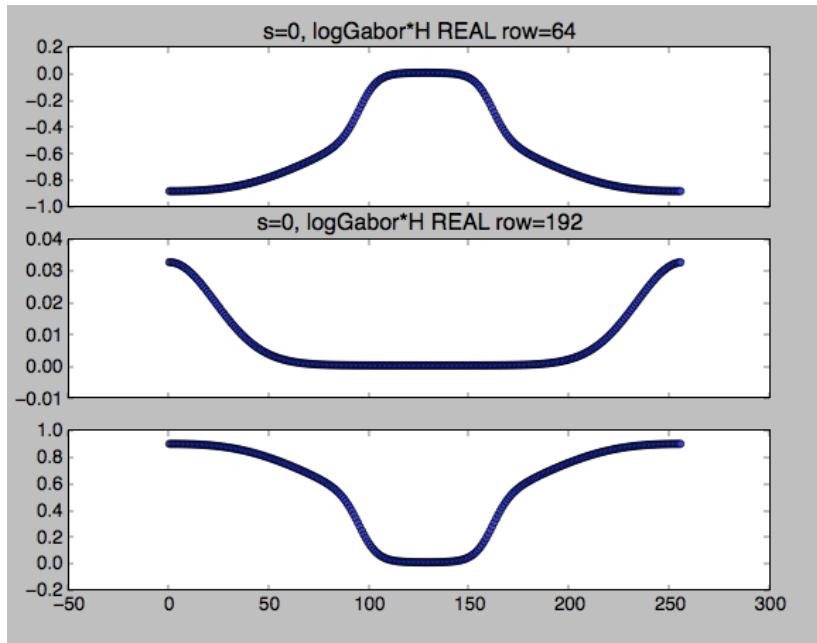
## imaginary(**low pass filter**)



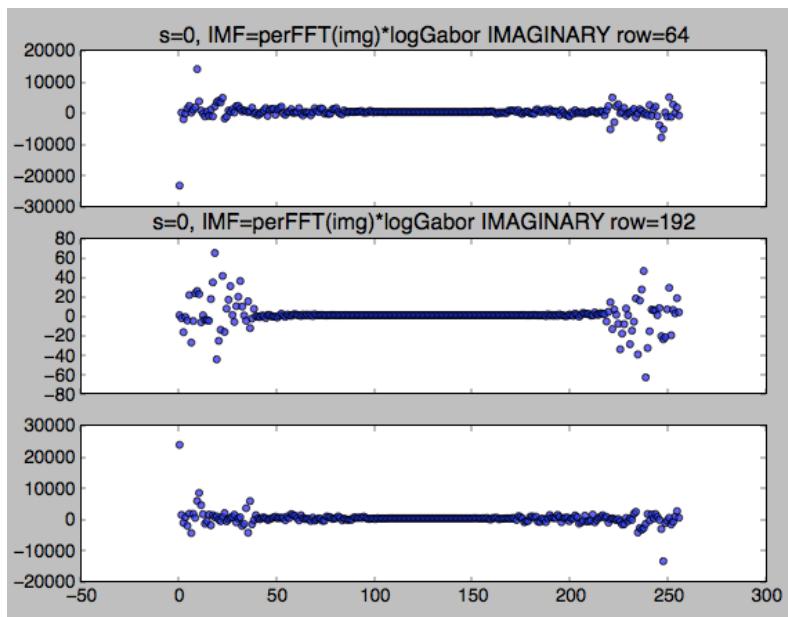
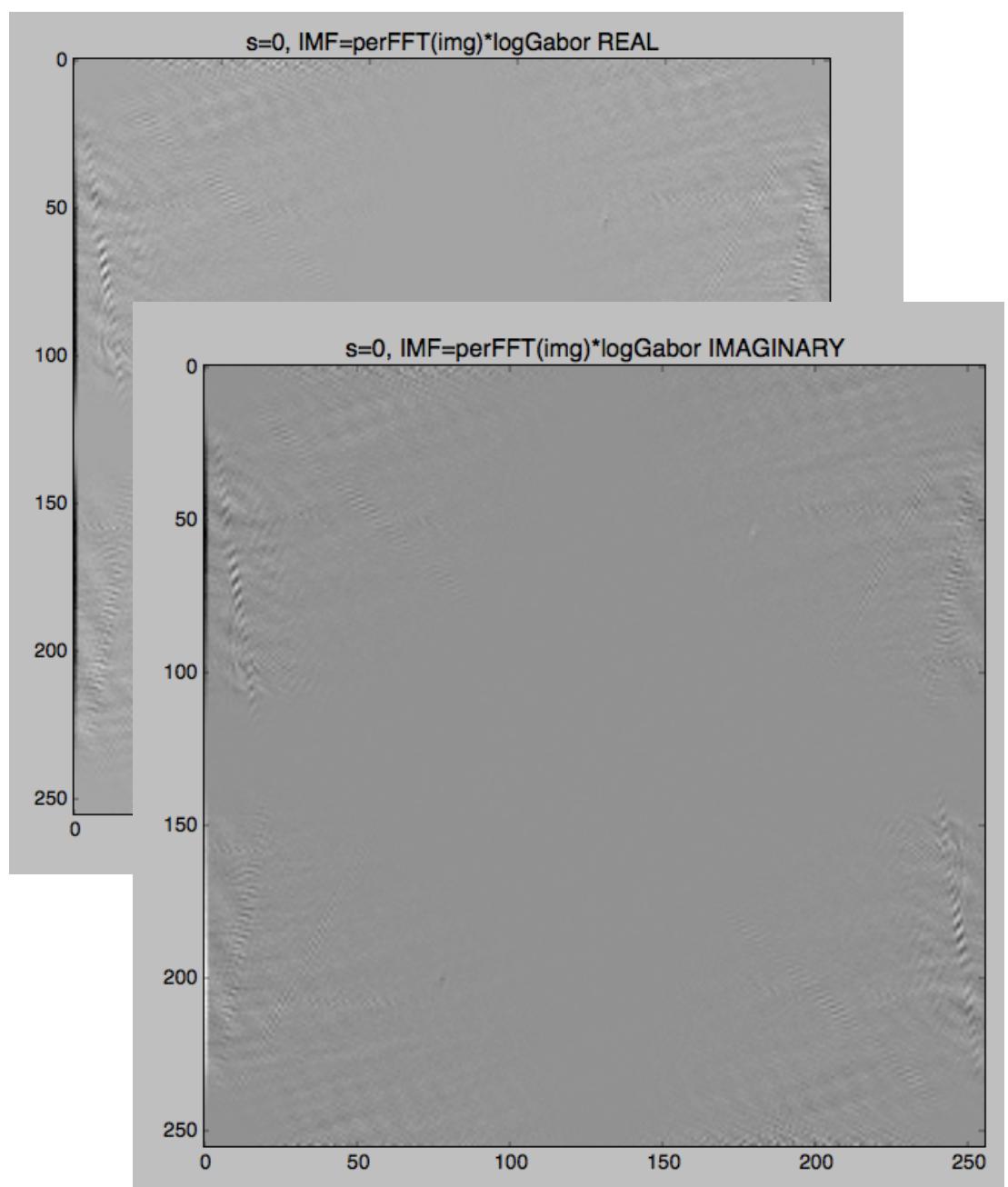
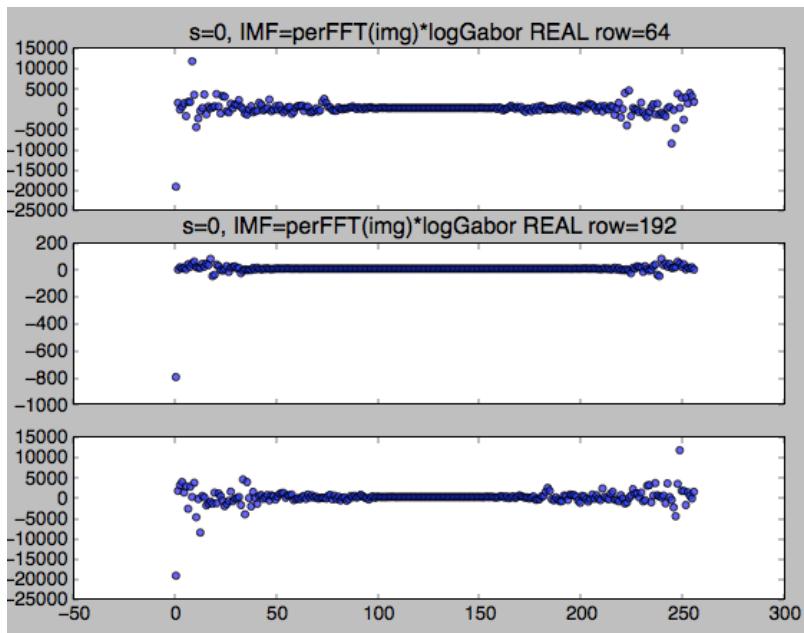
$s=0$ , logGabor \* low pass filter

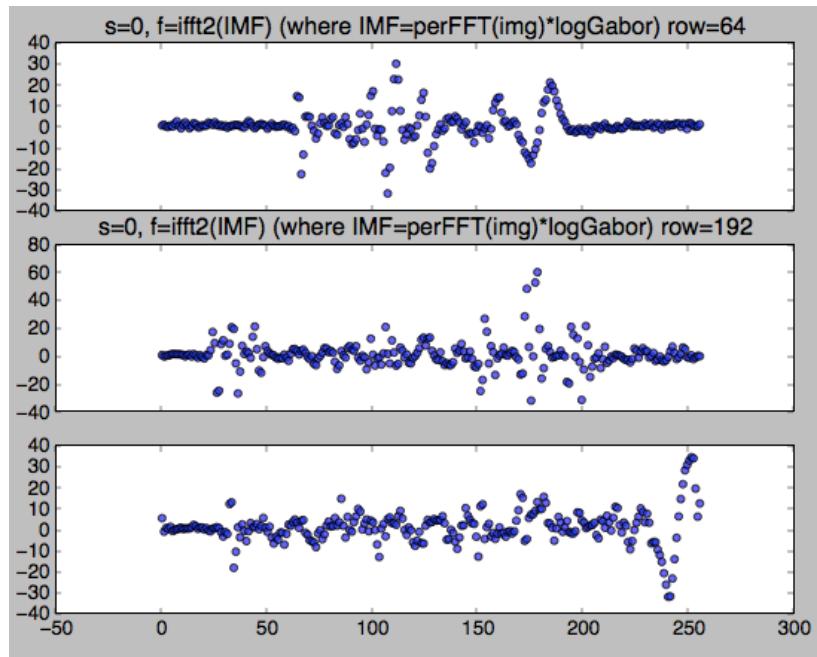


$s=0, \text{logGabor} * \text{low pass filter} * H$

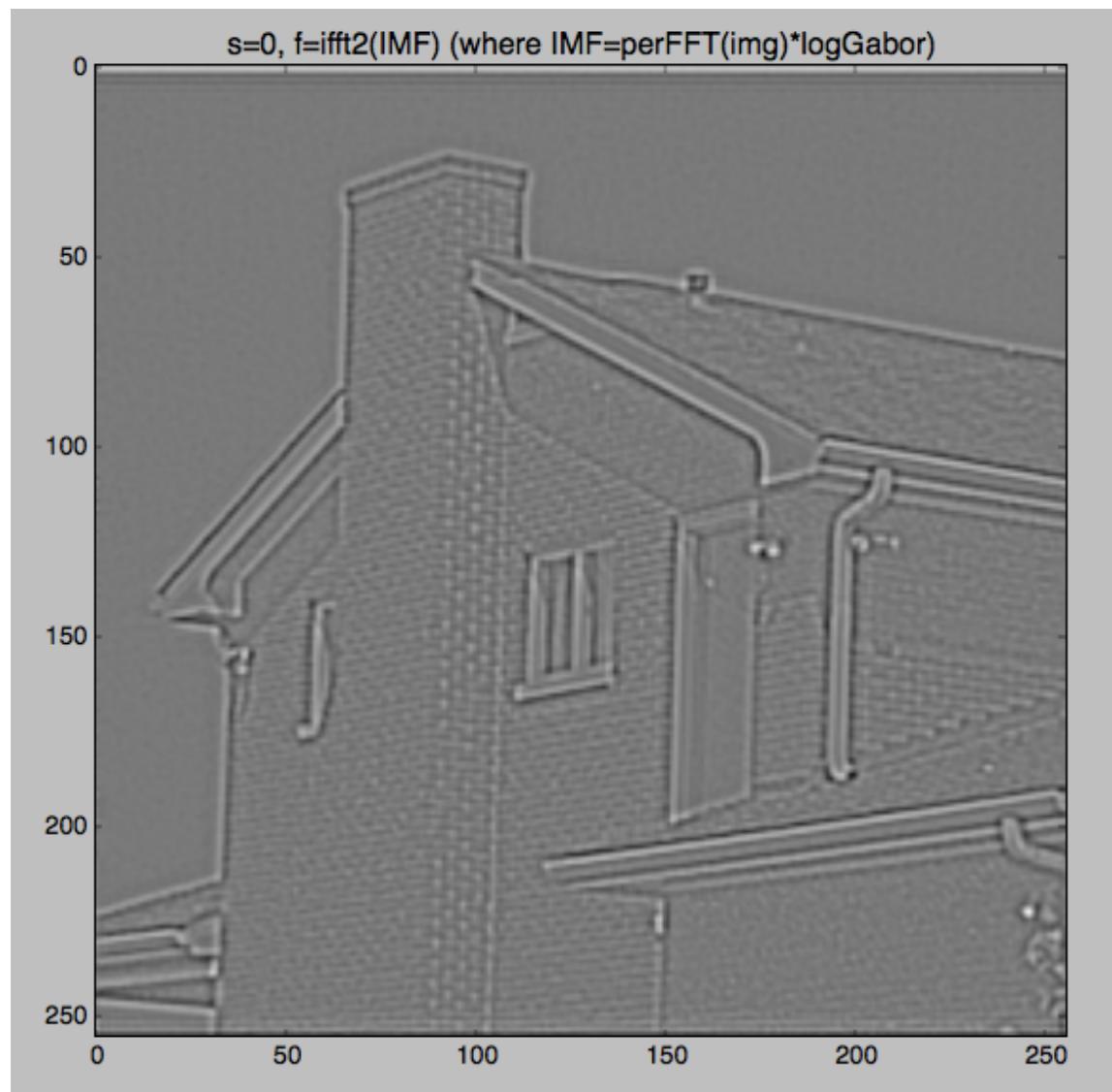


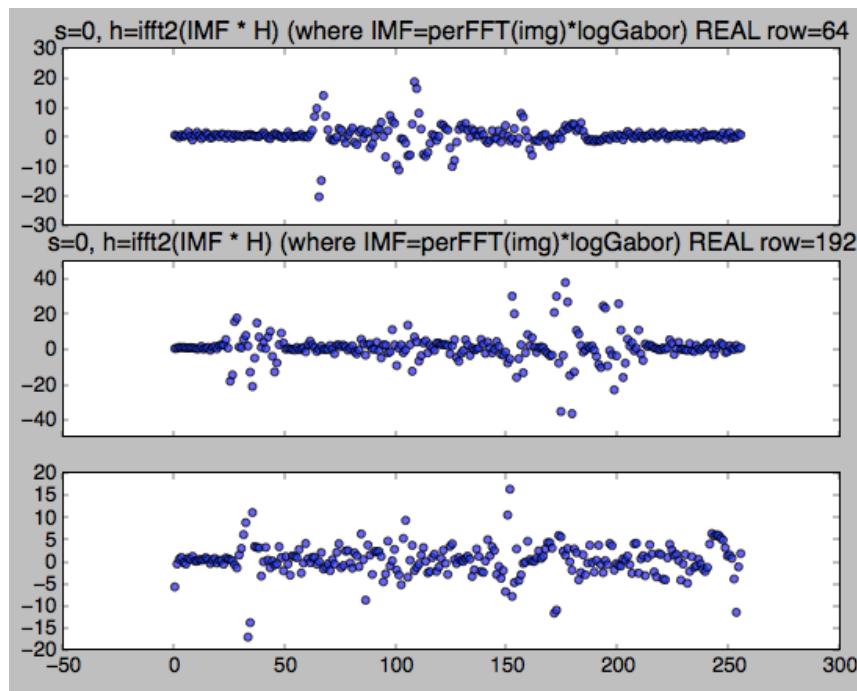
$s=0$ ,  $\text{IMF}=\text{perFFT(img)} * \text{logGabor}$



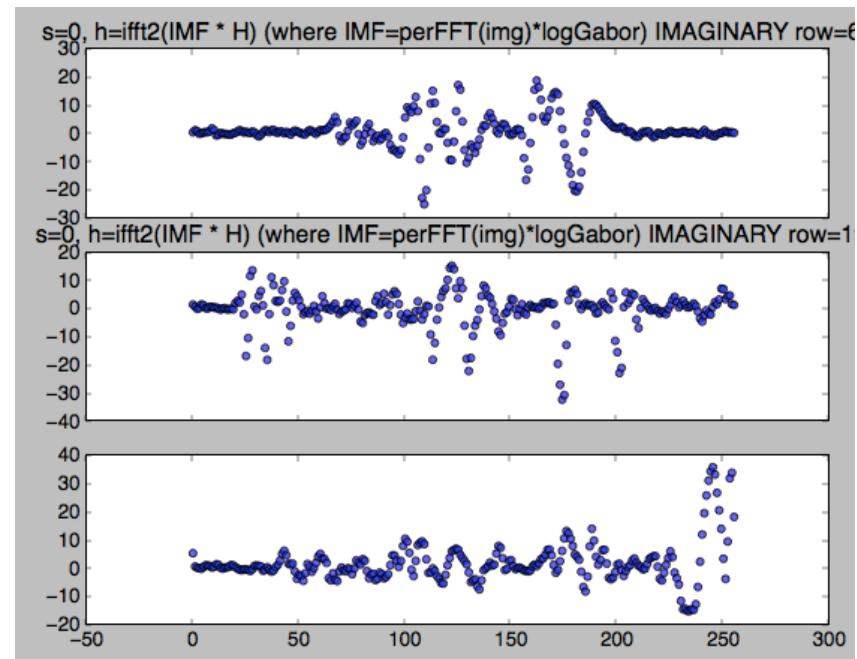
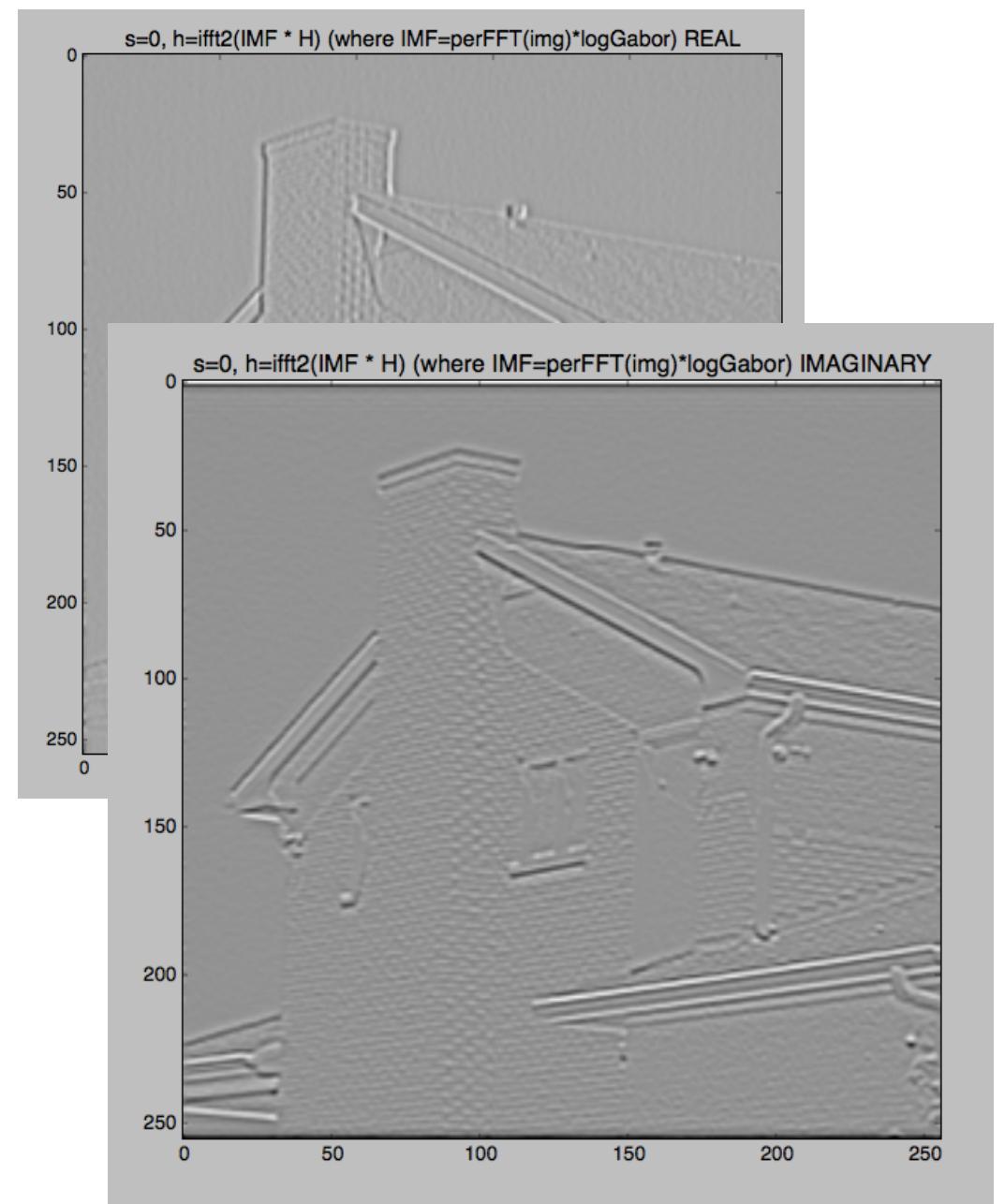


**s=0, f=ifft2(IMF)**  
**(where IMF=perFFT(img)\*logGabor)**

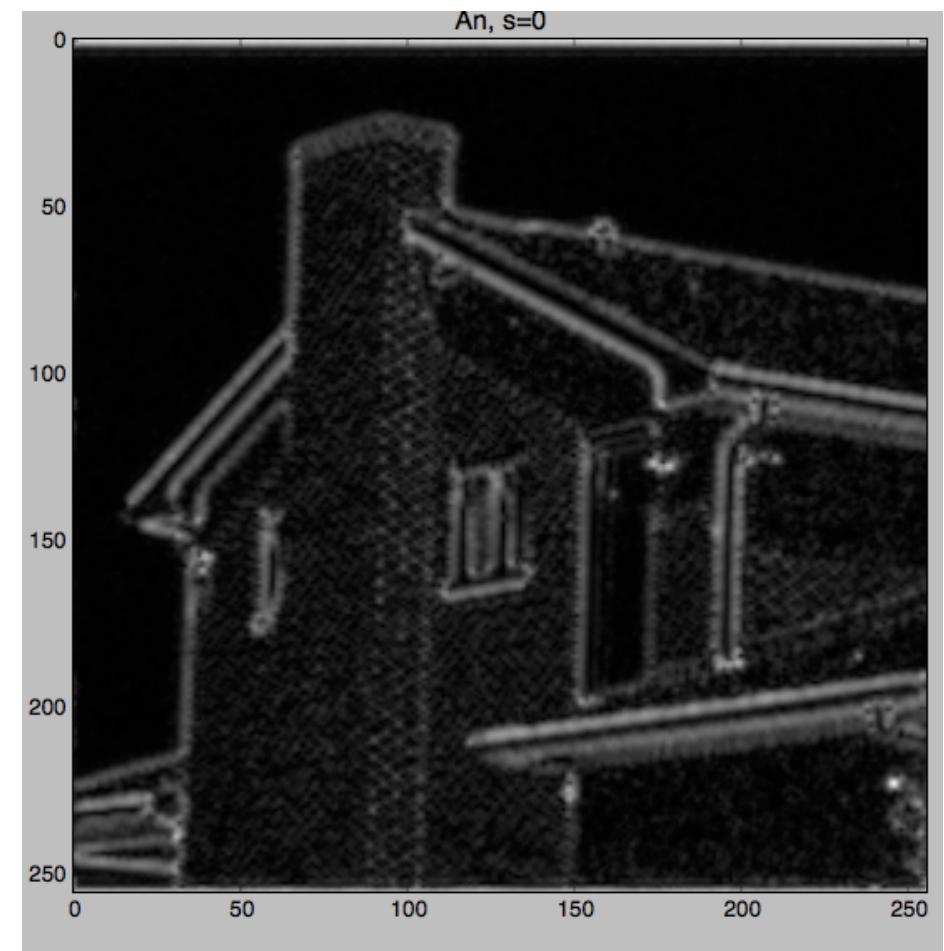
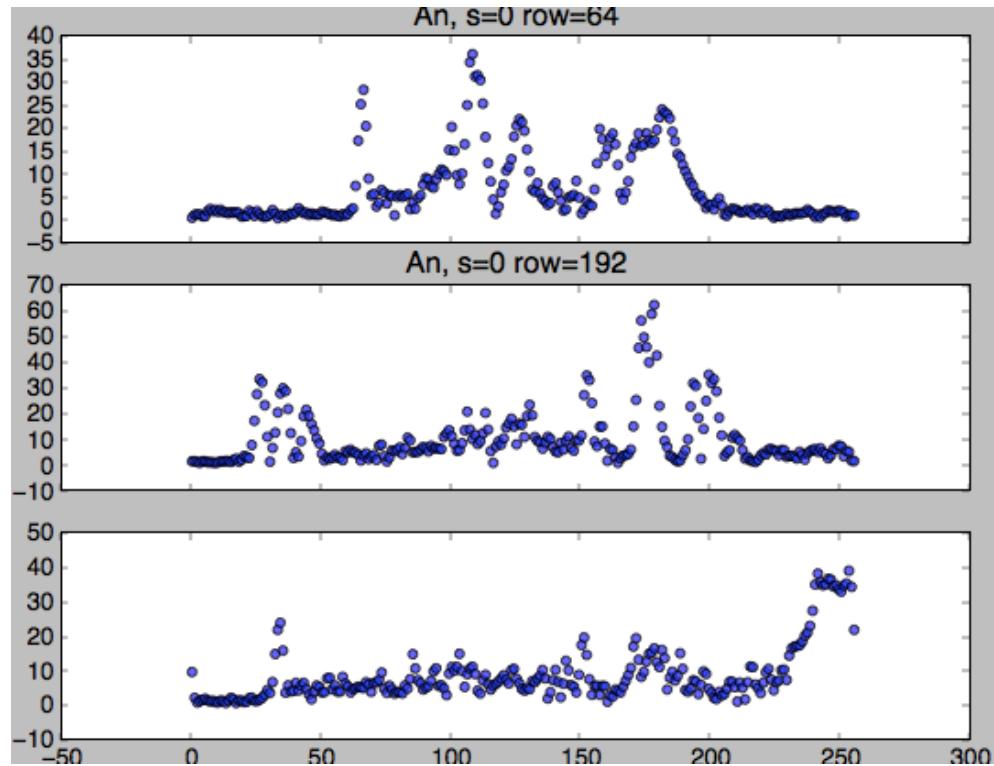




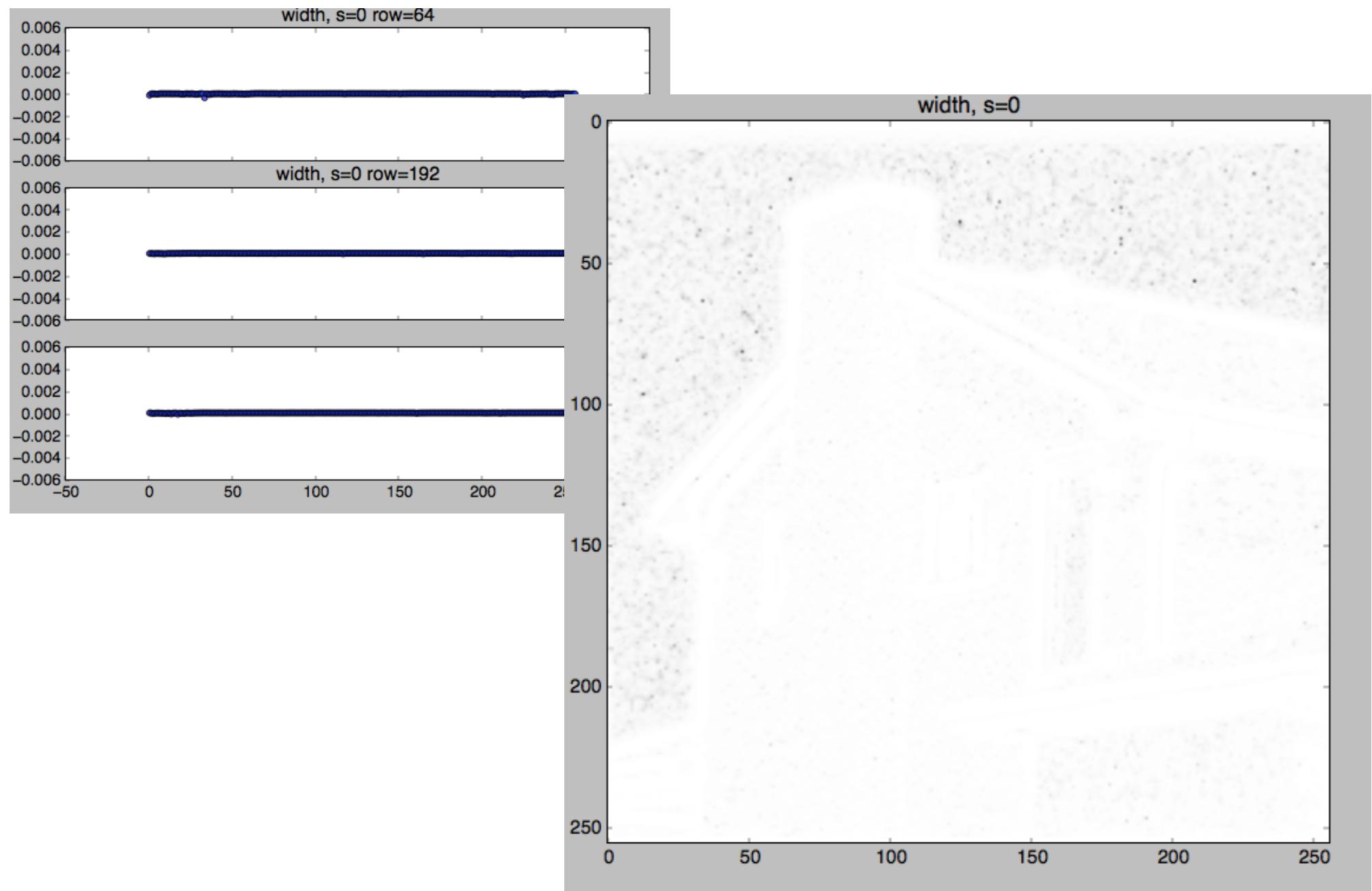
**s=0, h=ifft2(IMF \* H)  
(where IMF=perFFT(img)\*logGabor)**



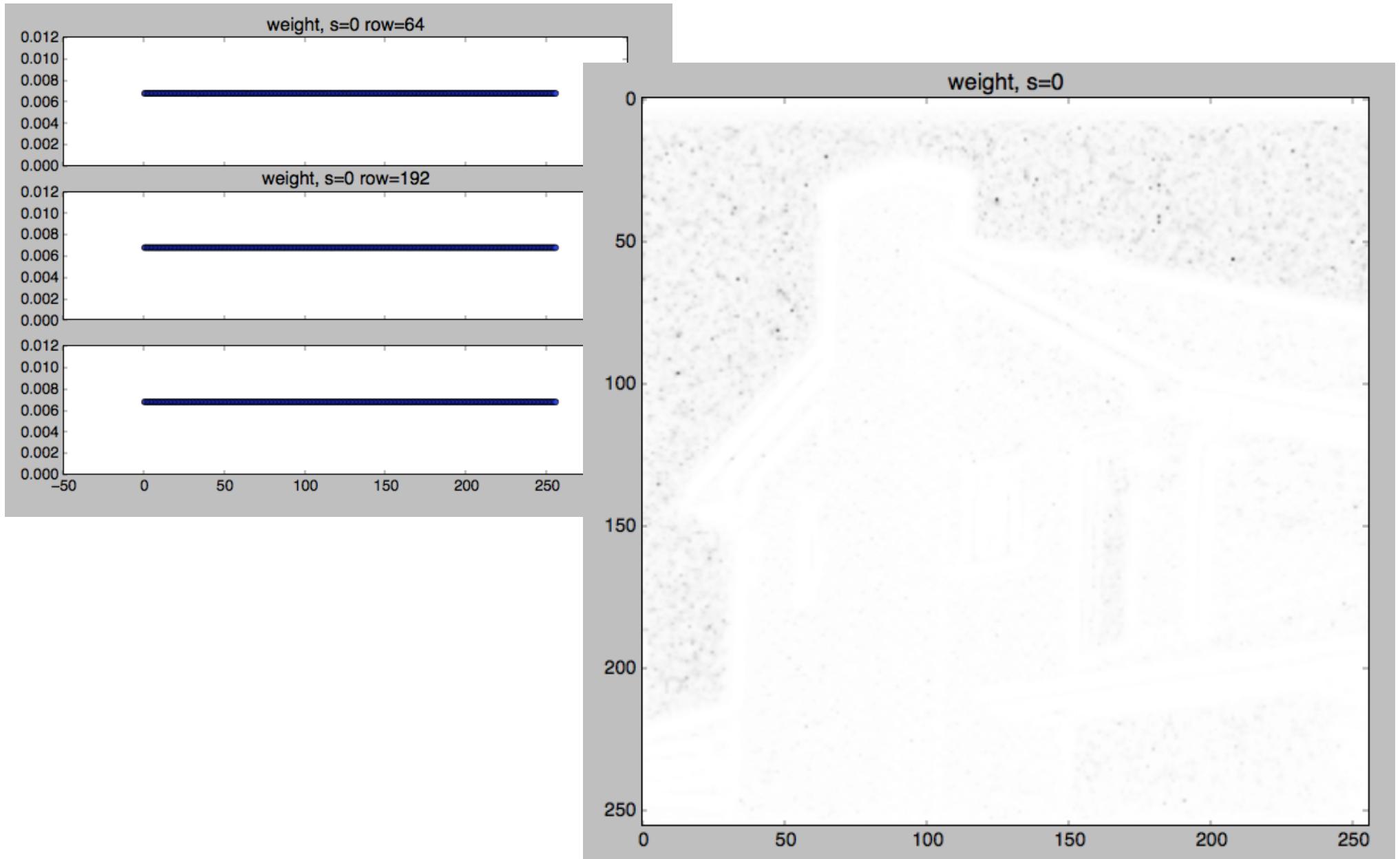
$s=0$ ,  $A_n = \sqrt{f * f + h1 * h1 + h2 * h2}$   
 (where  $f = \text{iFFT}(\text{perFFT(img)} * \text{logGabor})$   
 and  $h = \text{iFFT}(\text{perFFT(img)} * \text{logGabor} * H)$  )



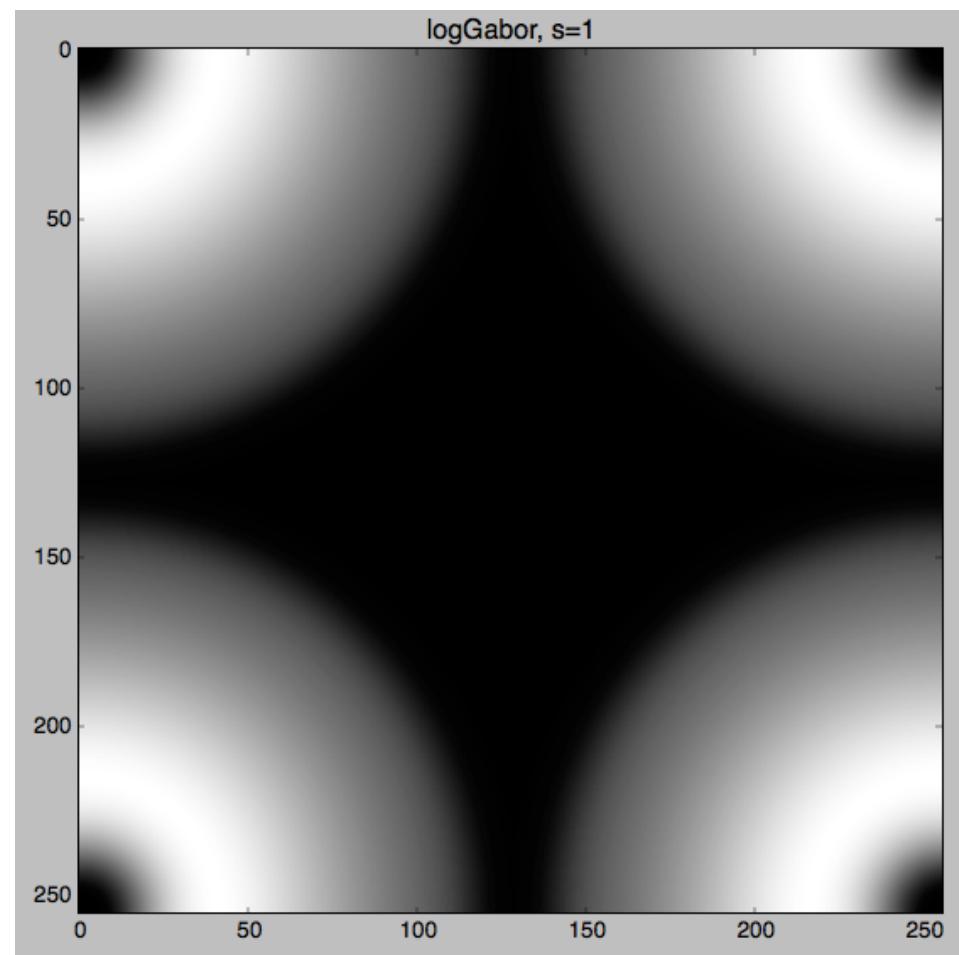
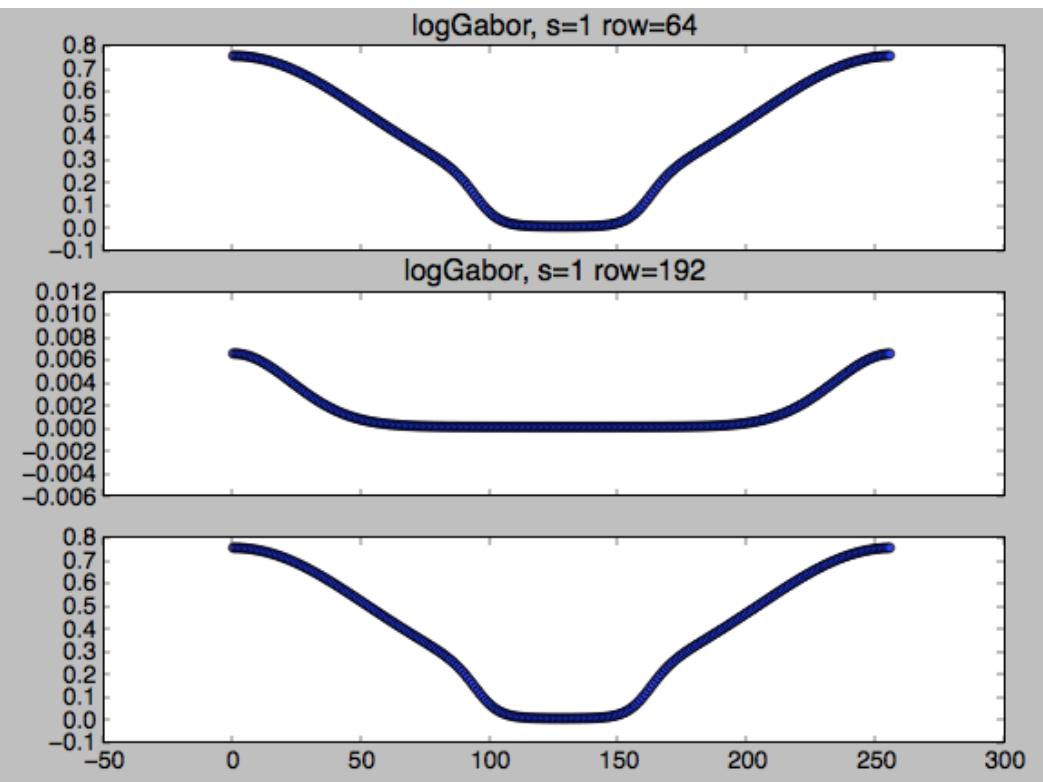
$s=0$ , **width** = (sumAn / (maxAn + epsilon) - 1.) / (nscale - 1)



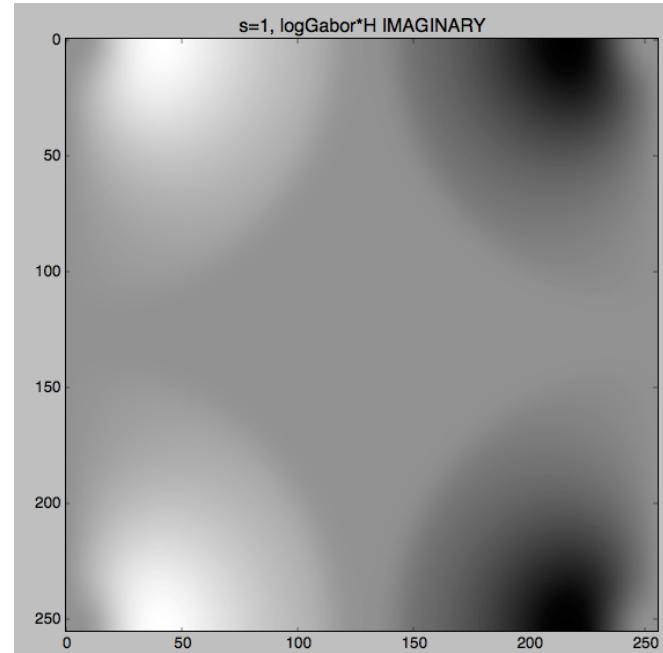
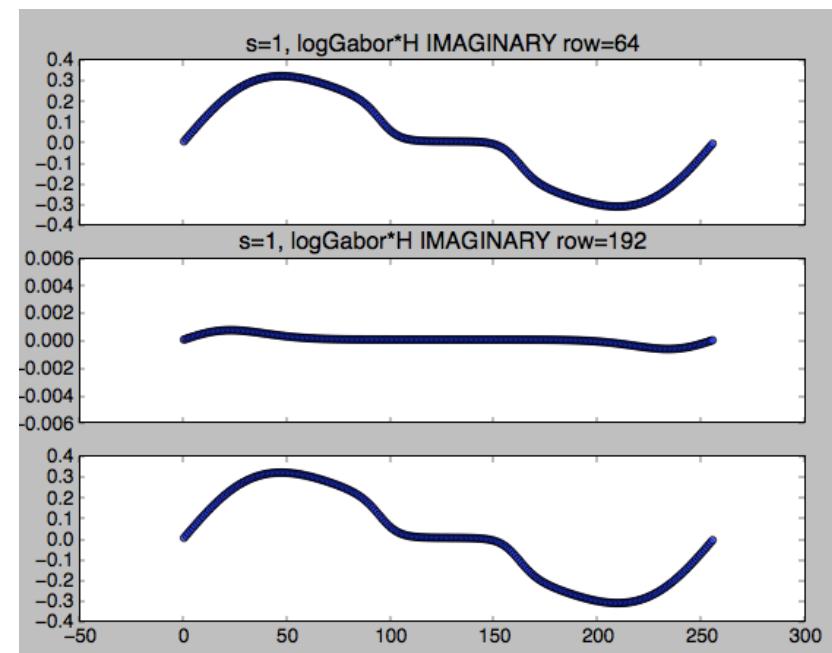
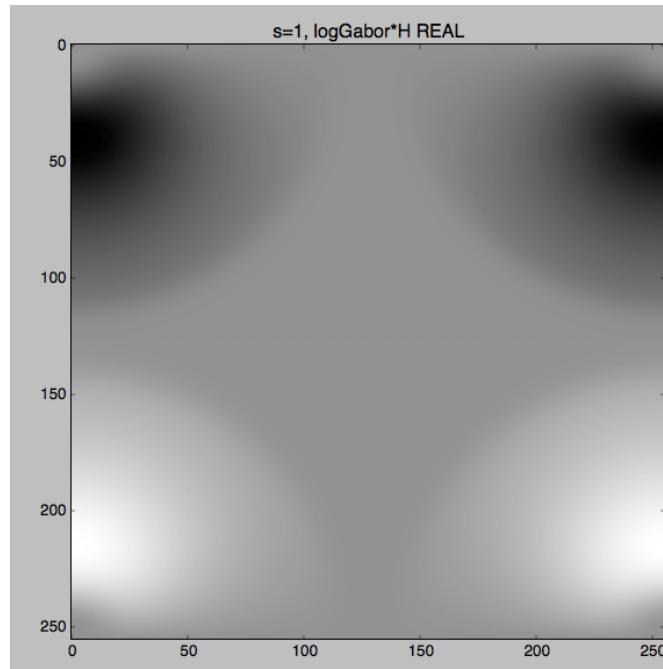
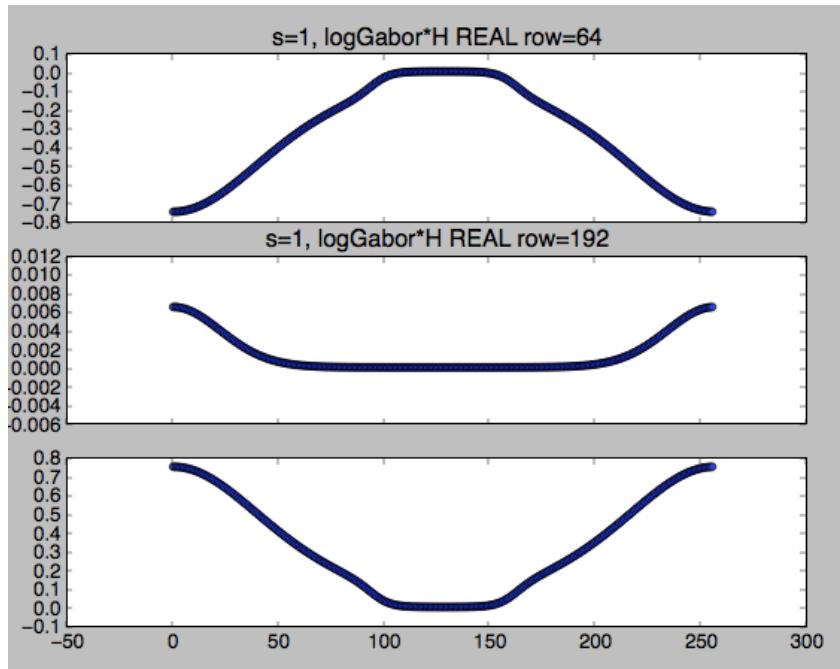
$s=0$ , **weight** =  $1. / (1. + \exp(g * (\text{cutoff} - \text{width})))$   
(where  $\text{cutoff}=0.5$ ,  $g=10.$ )

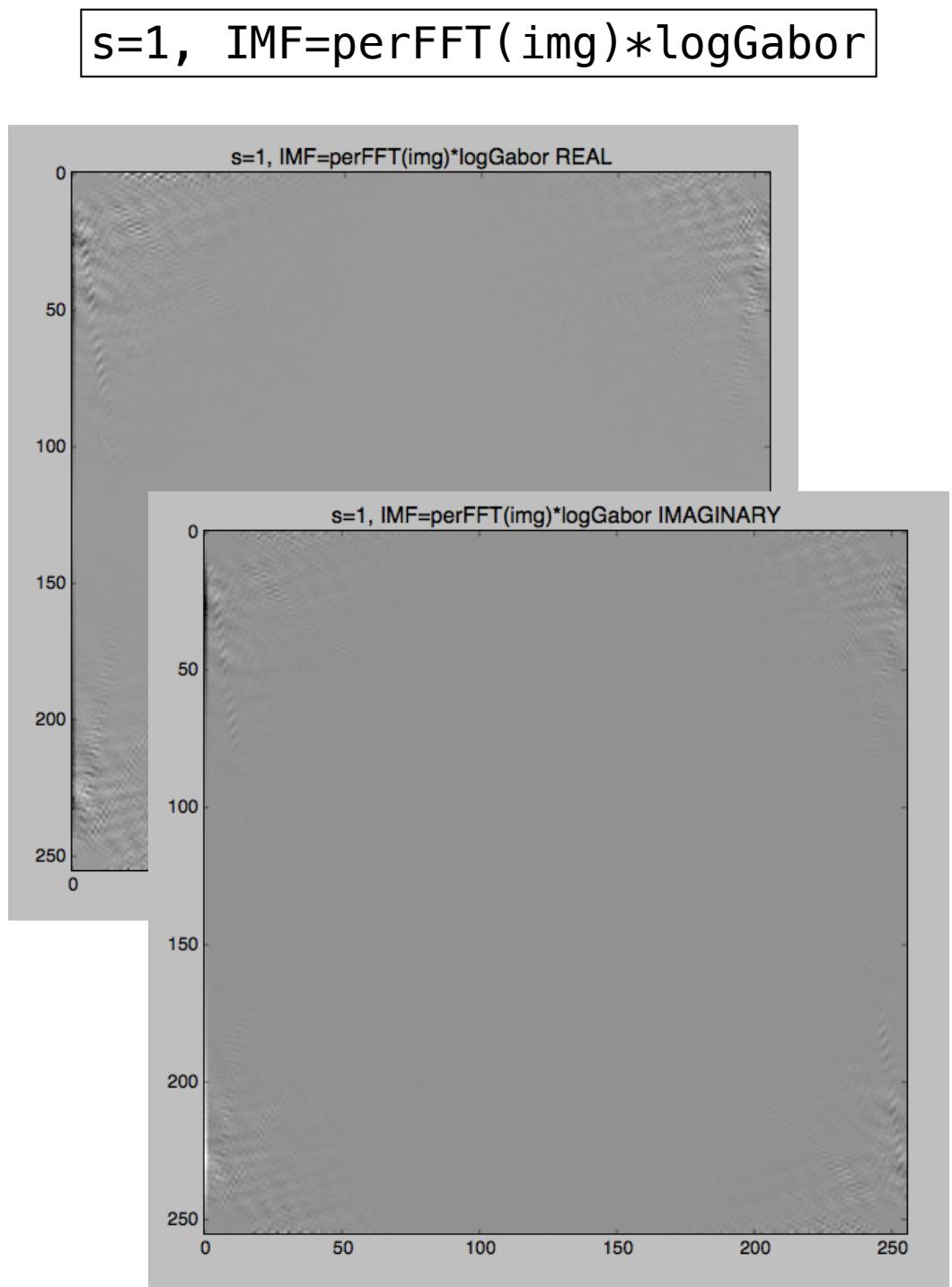
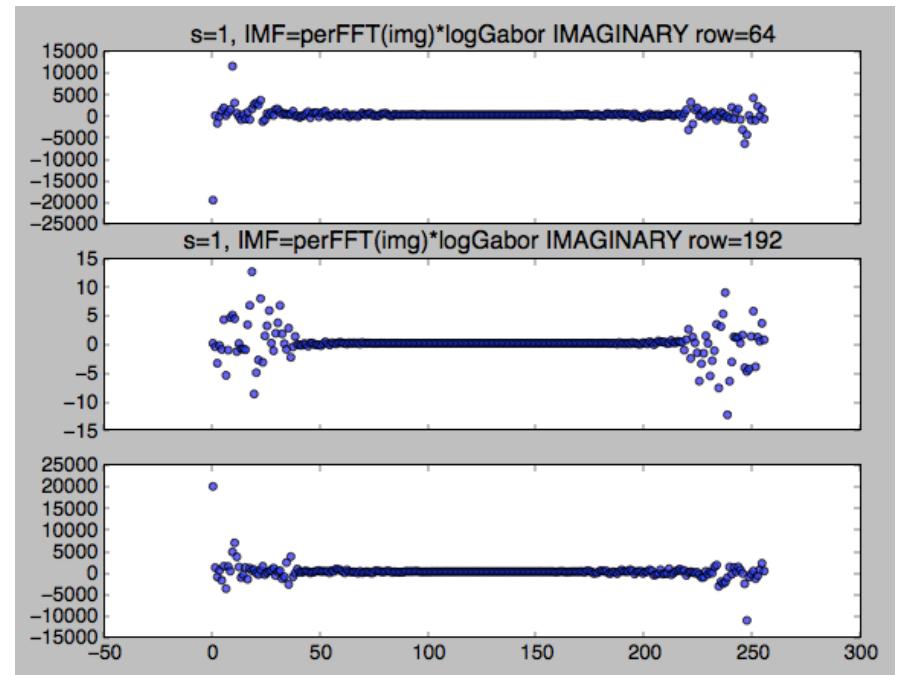
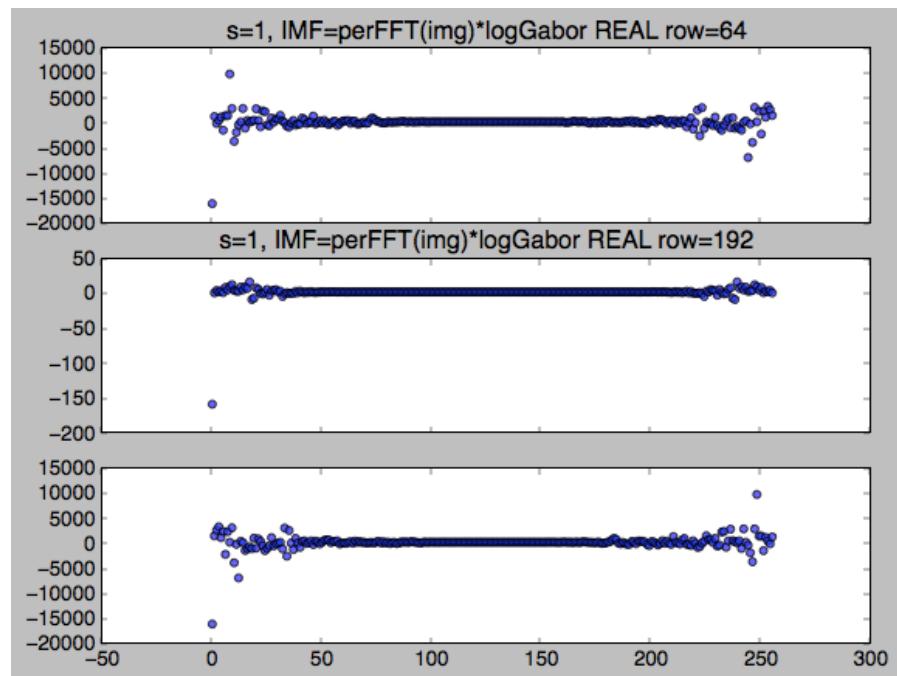


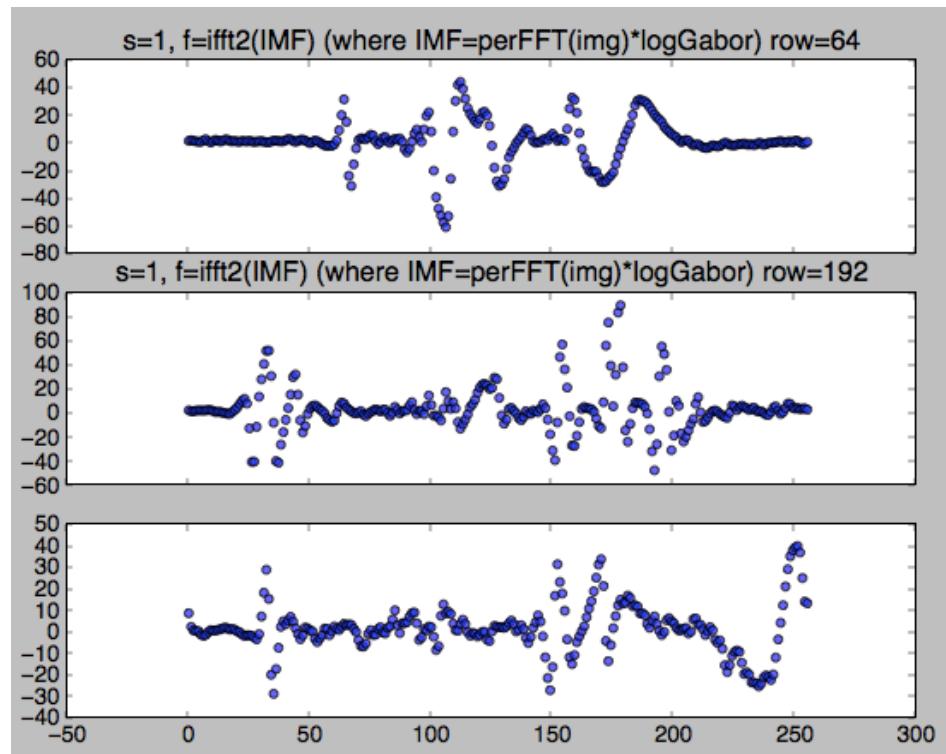
$s=1$ , logGabor \* low pass filter



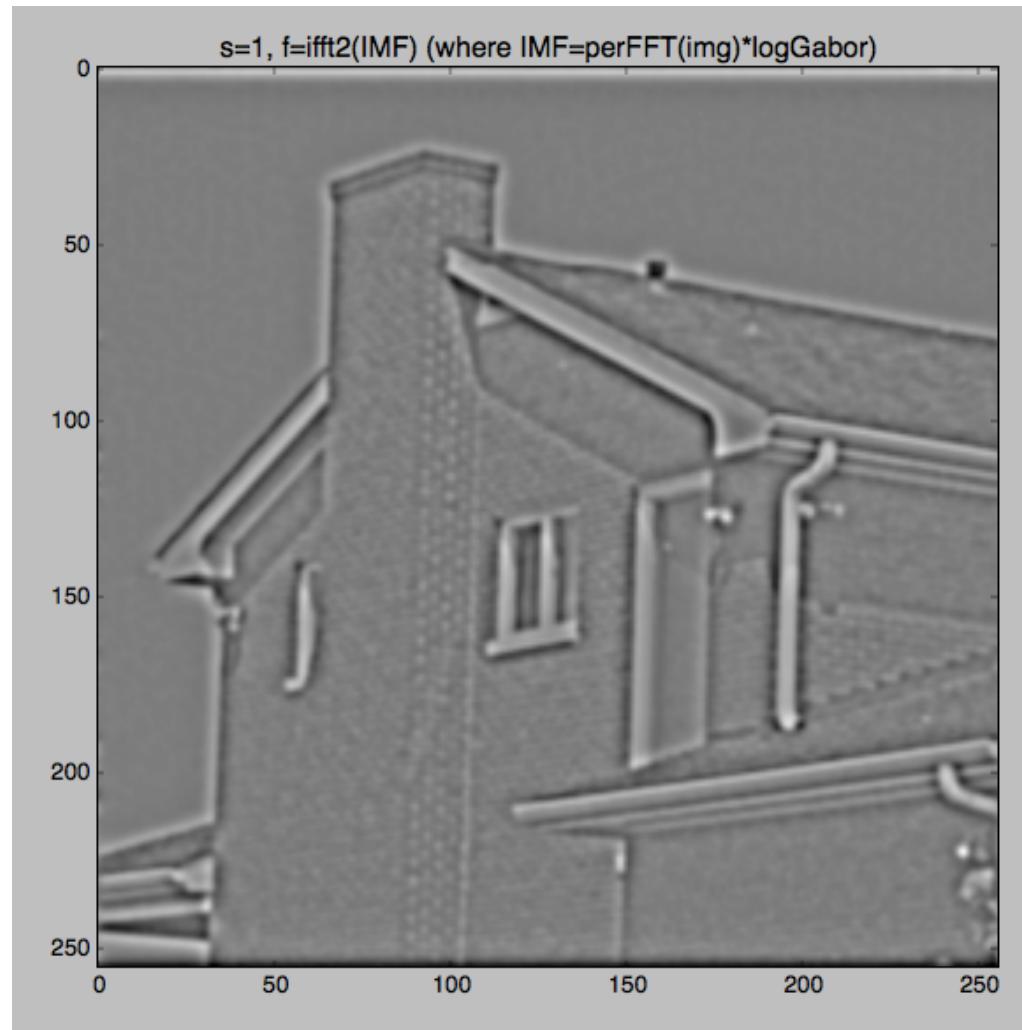
$s=1$ , logGabor \* low pass filter \* H



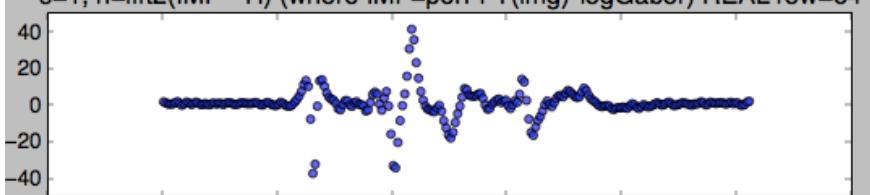




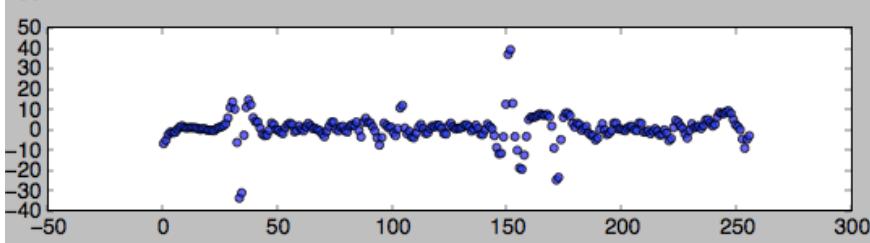
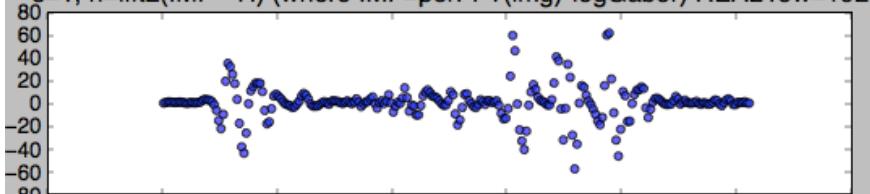
s=1, f=ifft2(IMF)  
(where IMF=perFFT(img)\*logGabor)



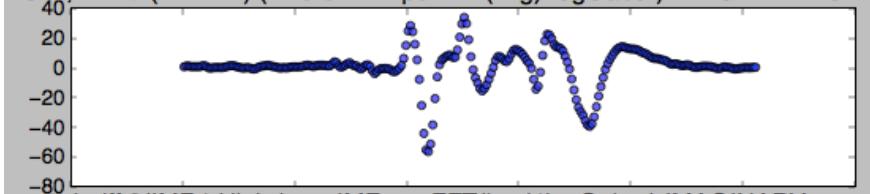
s=1, h=ifft2(IMF \* H) (where IMF=perFFT(img)\*logGabor) REAL row=64



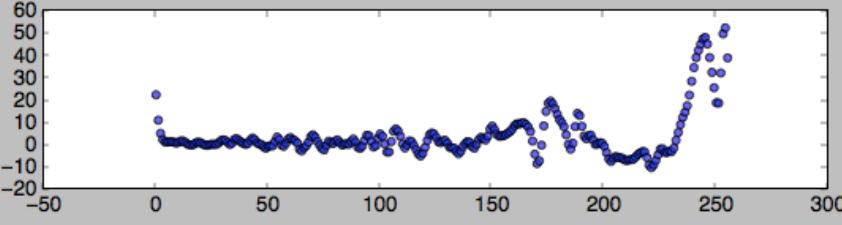
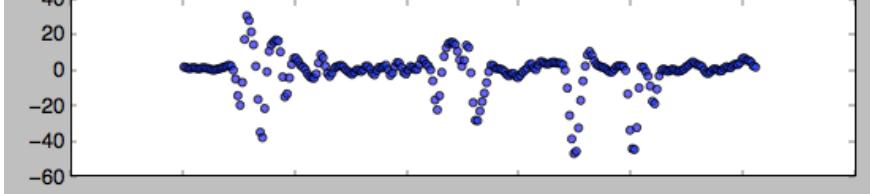
s=1, h=ifft2(IMF \* H) (where IMF=perFFT(img)\*logGabor) REAL row=192



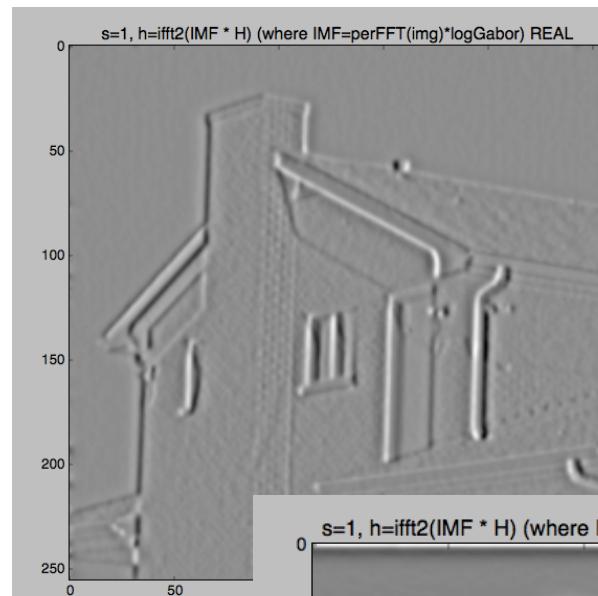
s=1, h=ifft2(IMF \* H) (where IMF=perFFT(img)\*logGabor) IMAGINARY row=



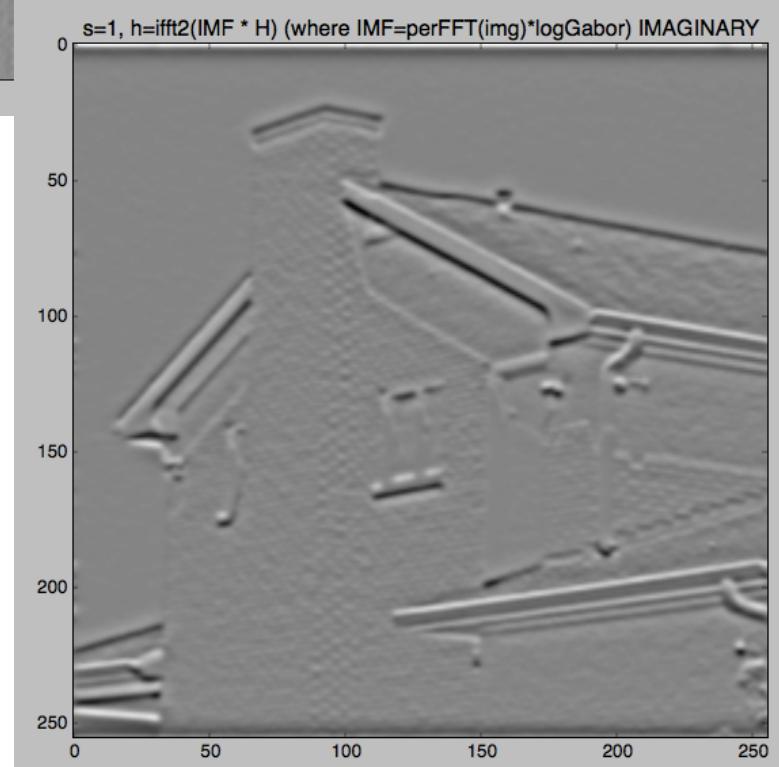
s=1, h=ifft2(IMF \* H) (where IMF=perFFT(img)\*logGabor) IMAGINARY row=1



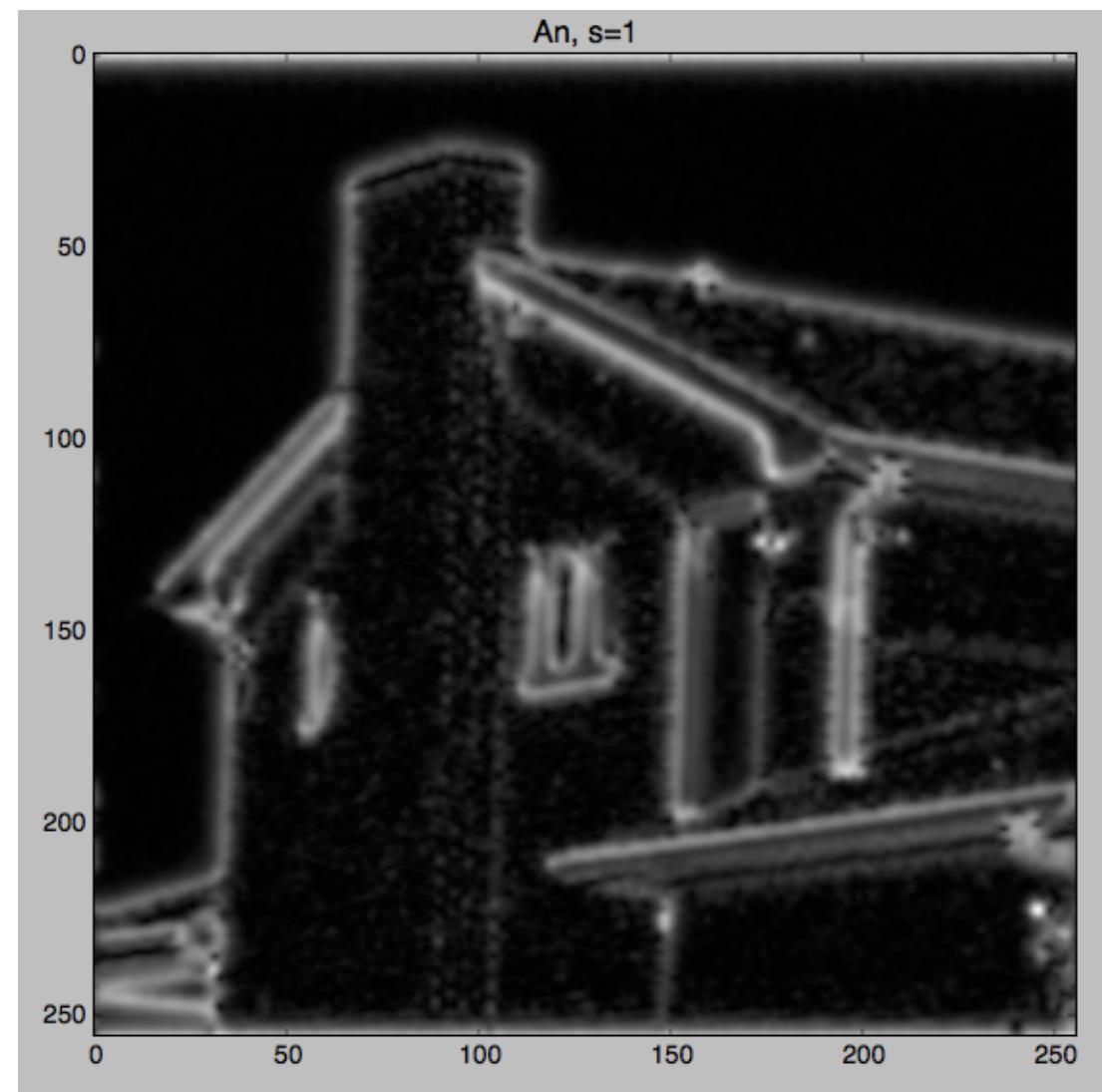
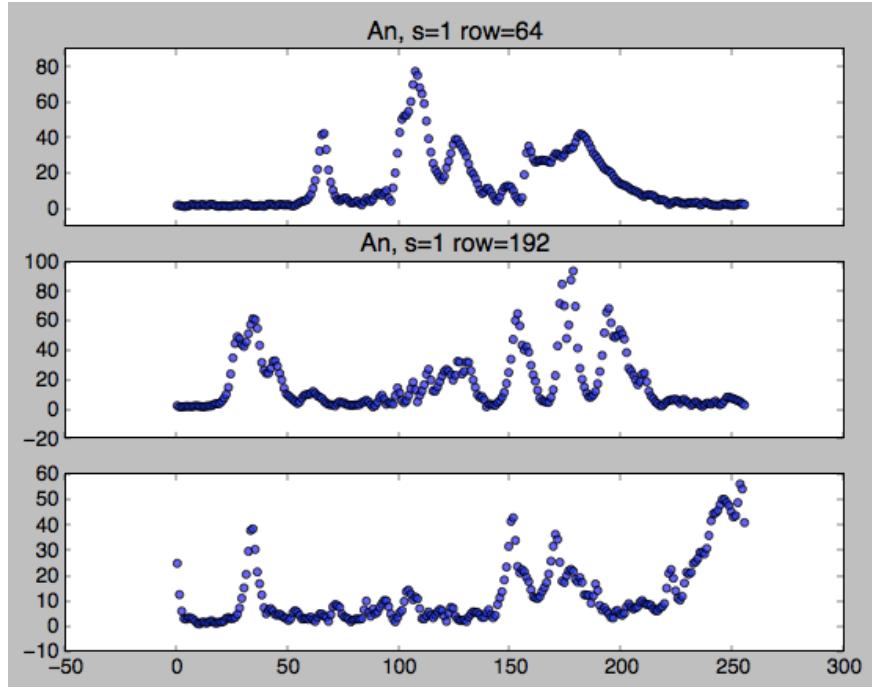
s=1, h=ifft2(IMF \* H)  
(where IMF=perFFT(img)\*logGabor)



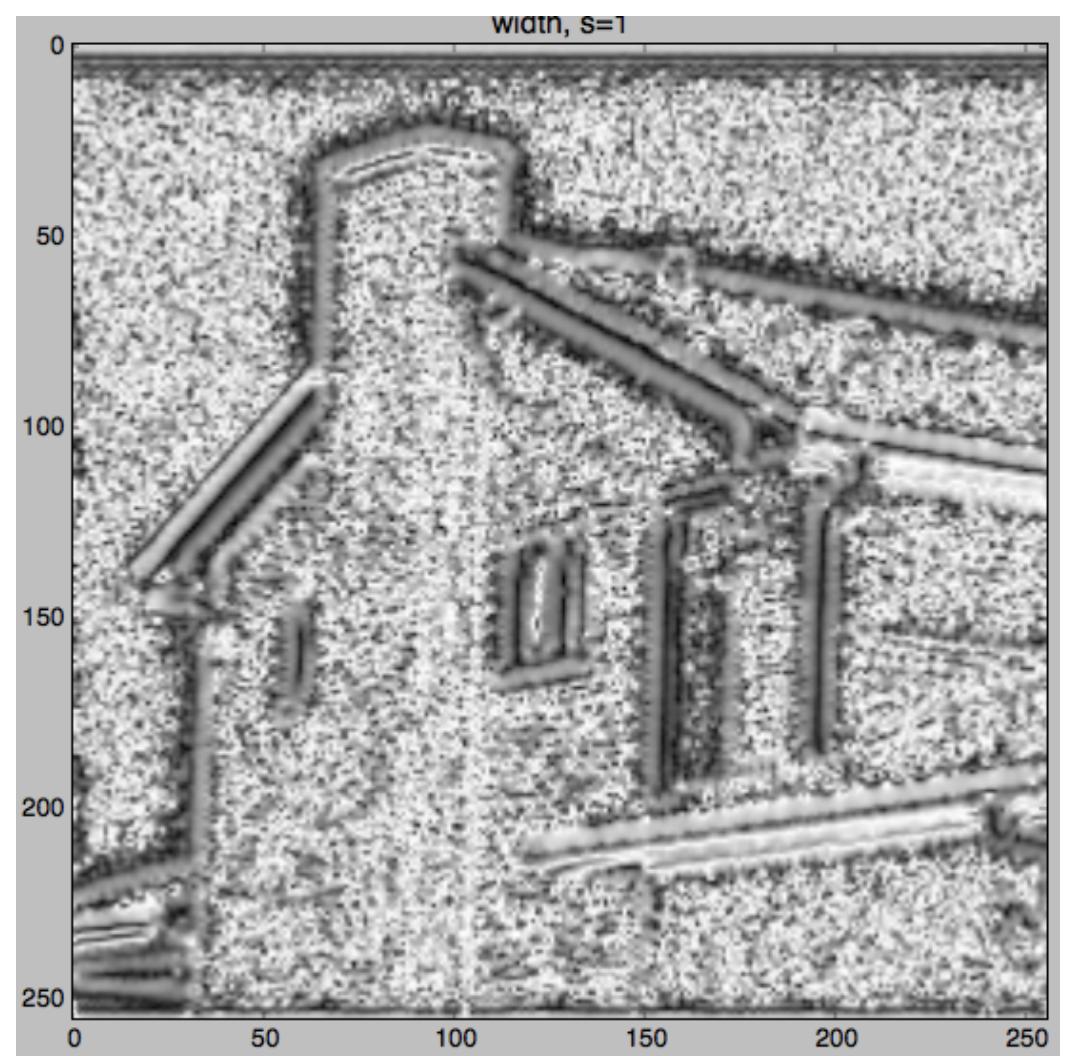
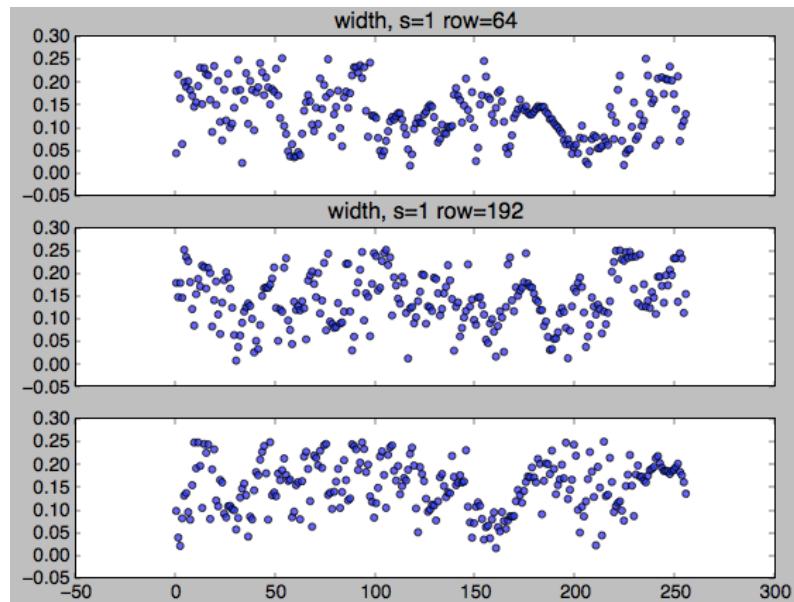
s=1, h=ifft2(IMF \* H) (where IMF=perFFT(img)\*logGabor) IMAGINARY



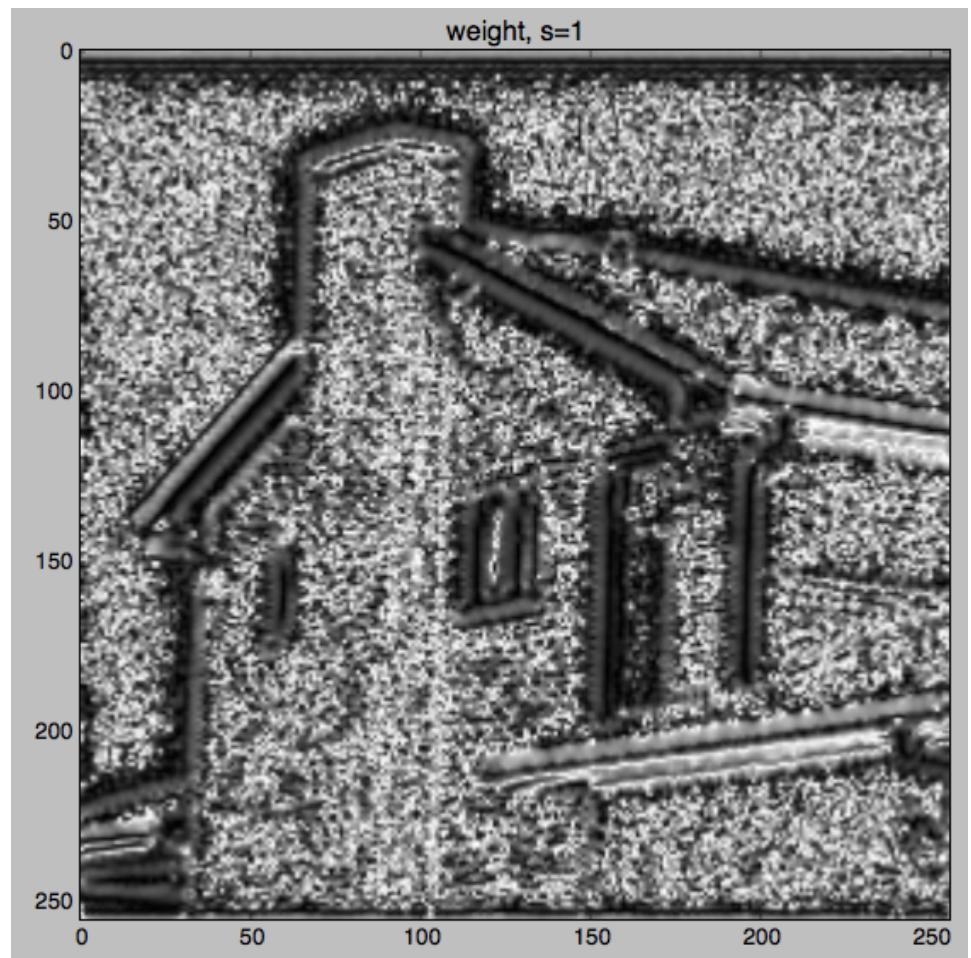
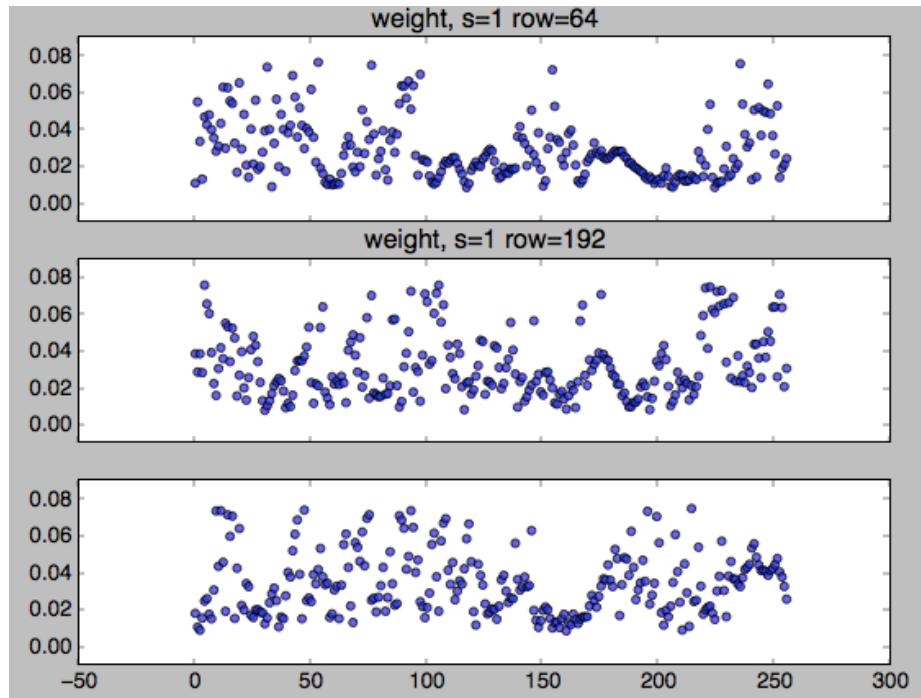
```
s=1, An = sqrt(f * f + h1 * h1 + h2 * h2)
(where f=ifft2(perFFT(img)*logGabor)
and h=ifft2(perFFT(img)*logGabor * H) )
```



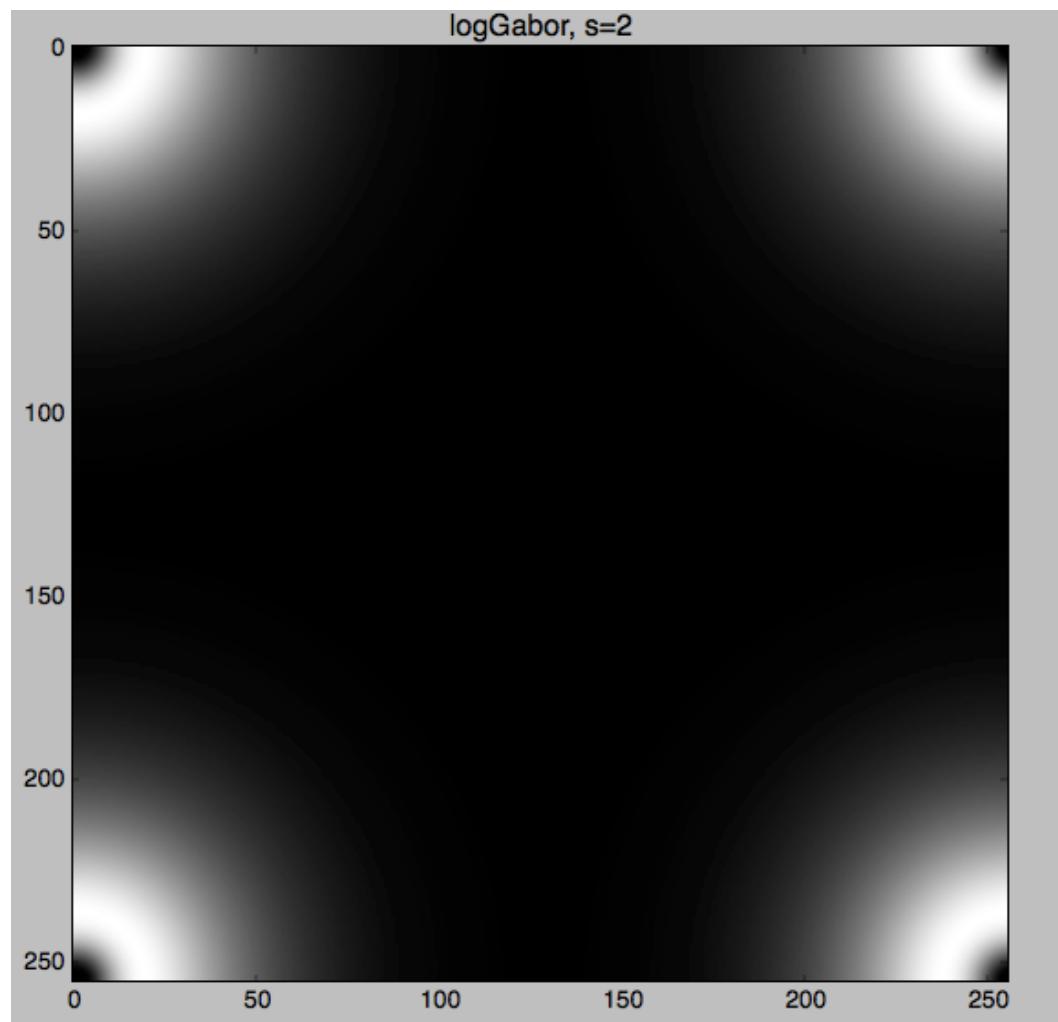
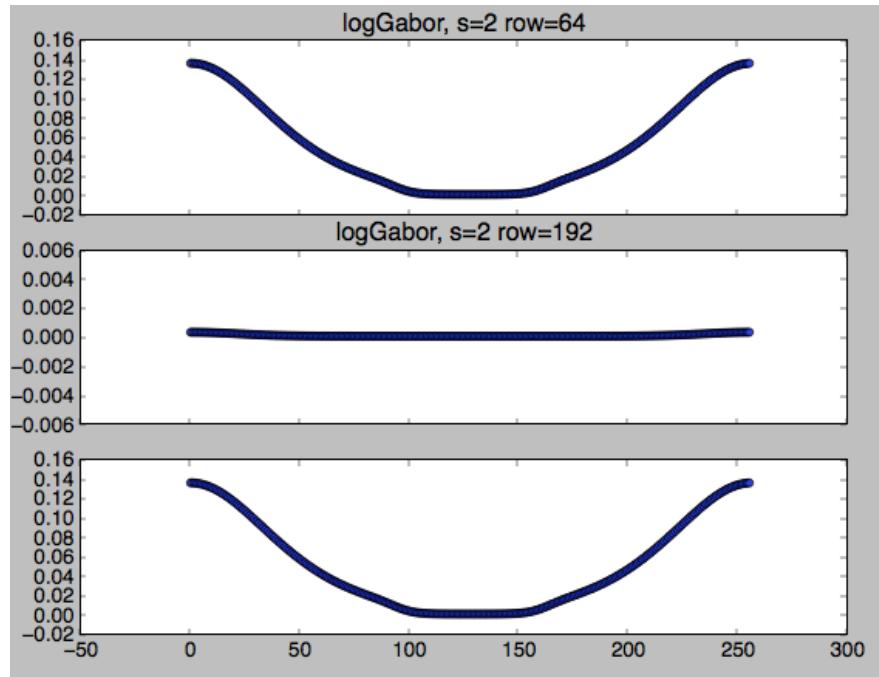
**s=1, width = (sumAn / (maxAn + epsilon) - 1.) / (nscale - 1)**



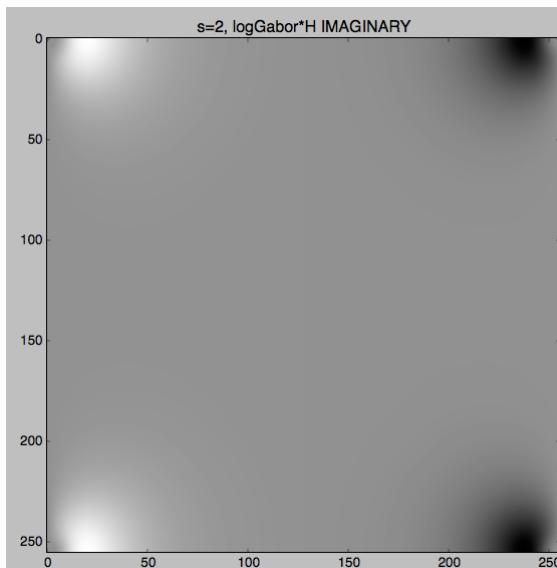
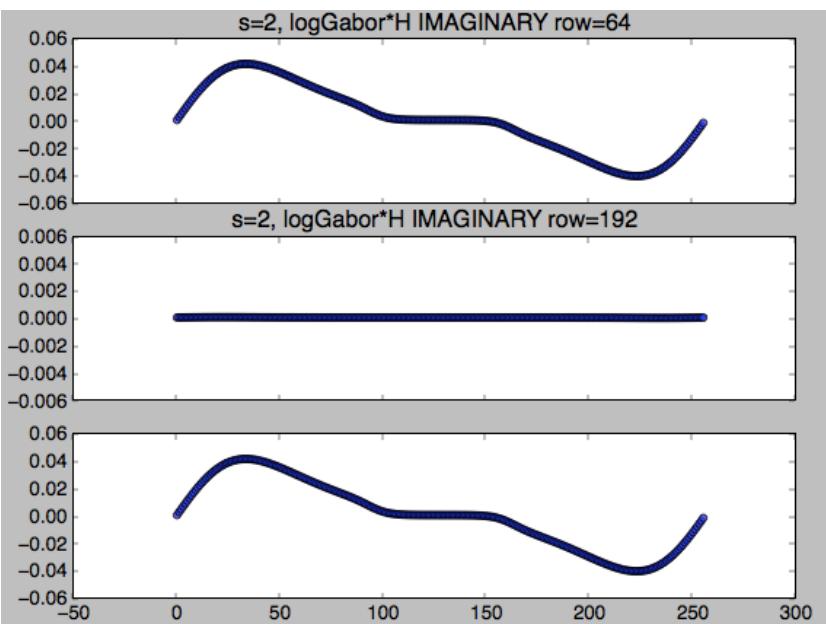
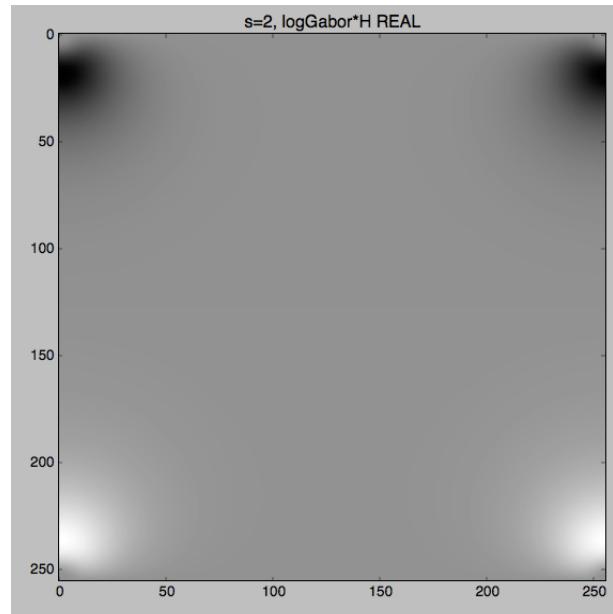
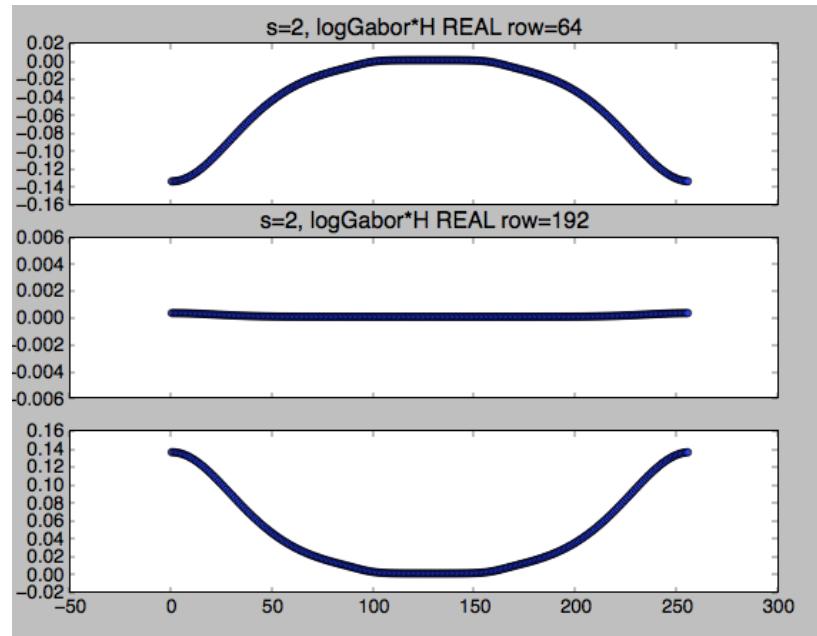
**s=1, weight =  $1. / (1. + \exp(g * (\text{cutoff} - \text{width})))$**   
(where cutoff=0.5, g=10.)



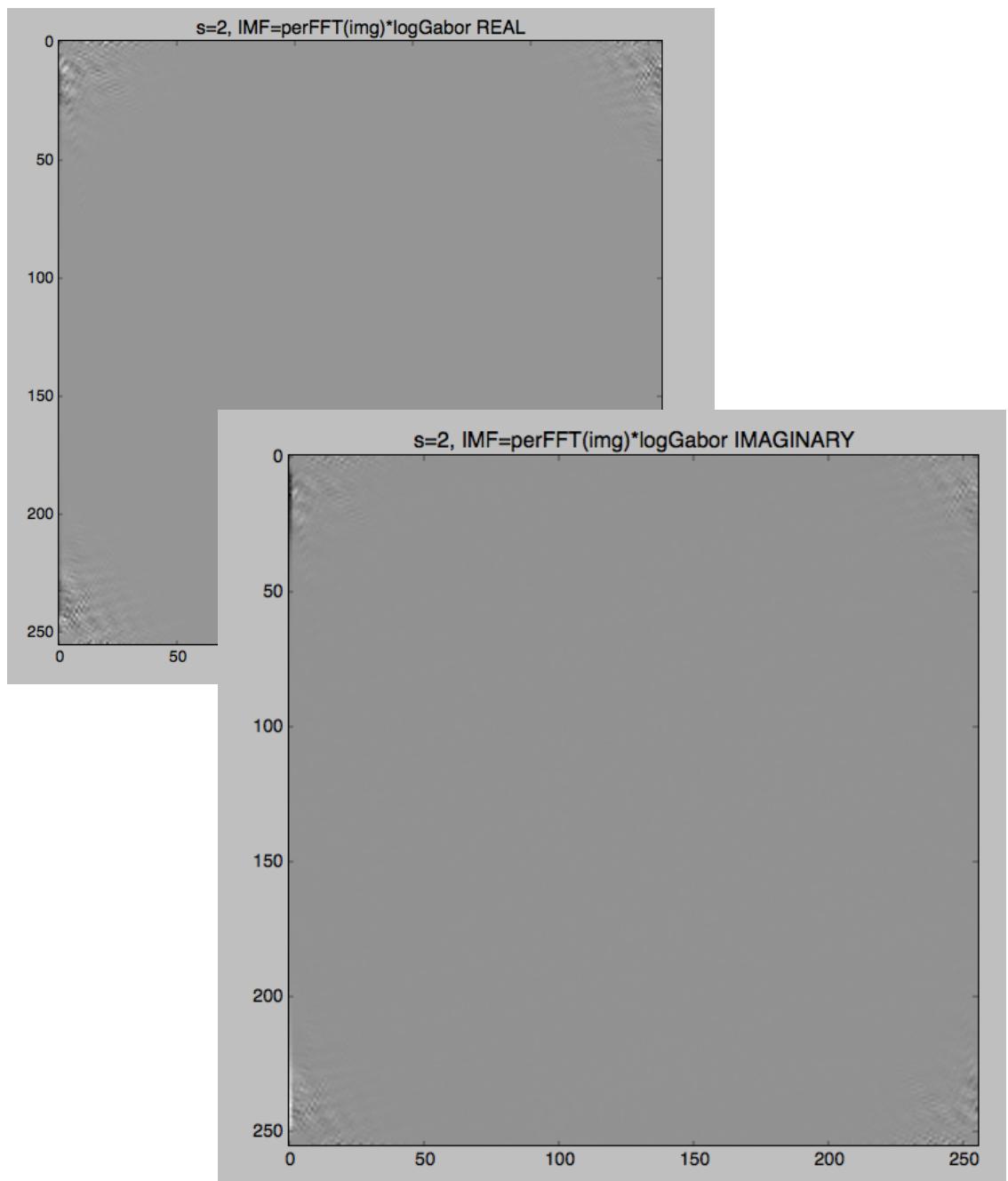
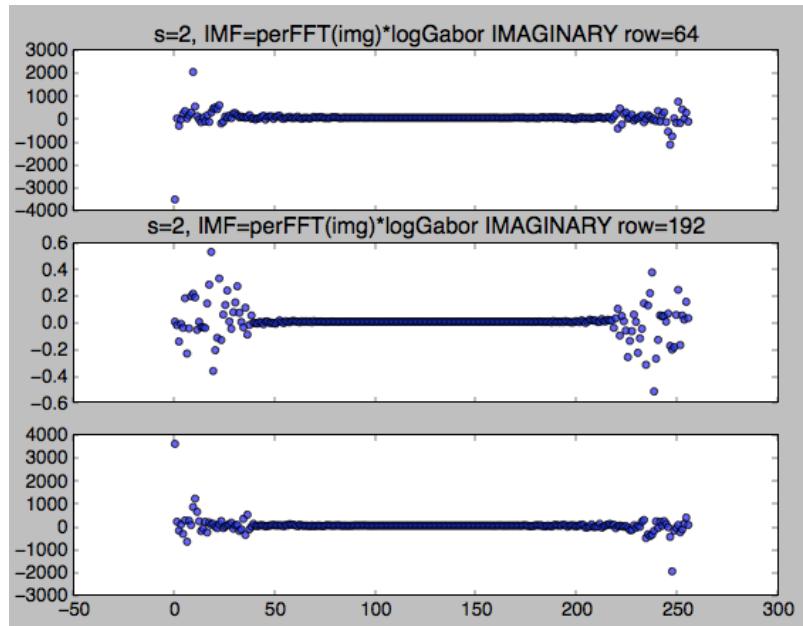
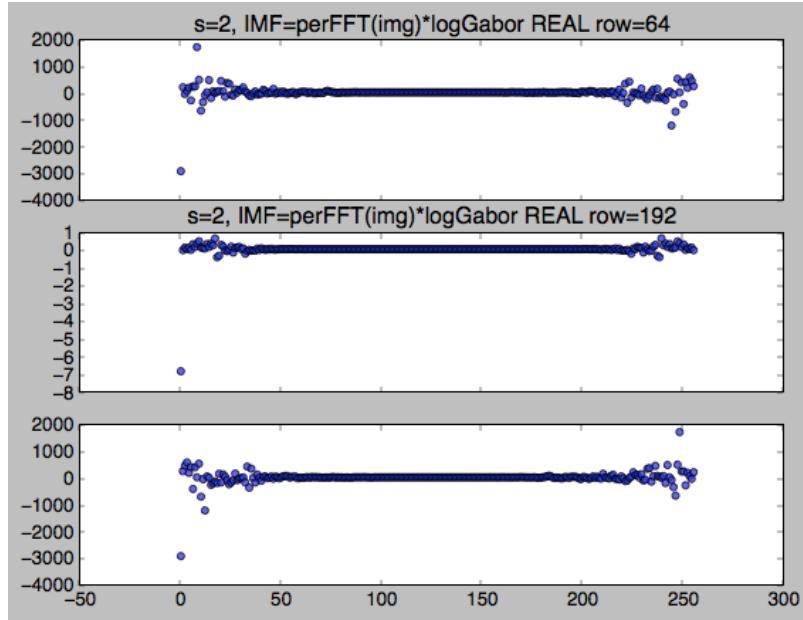
s=2, logGabor \* low pass filter

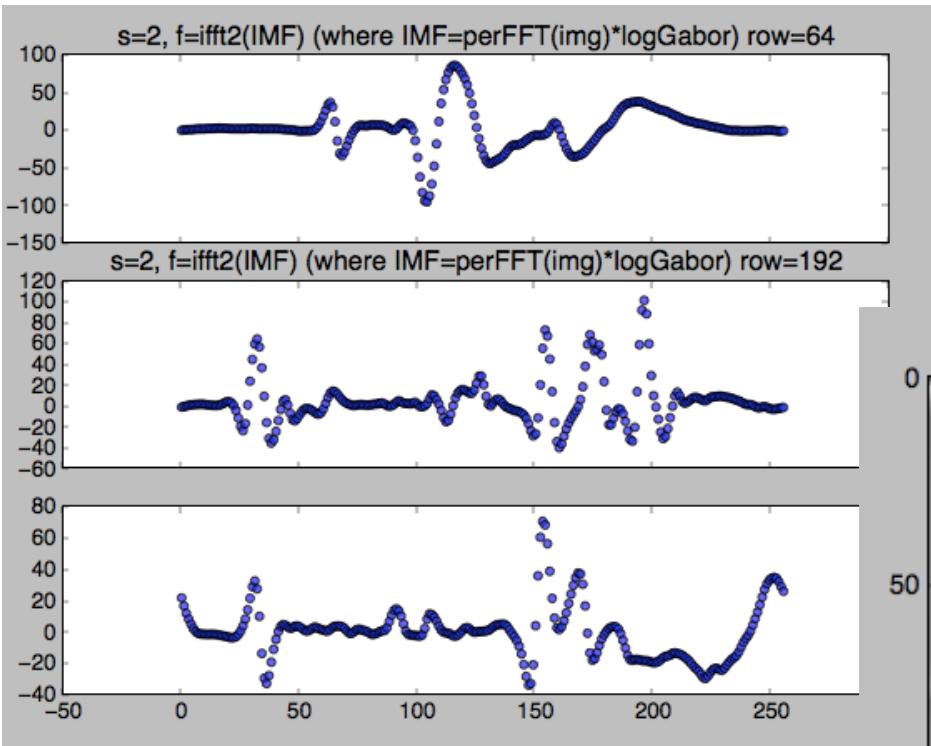


$s=2$ , logGabor \* low pass filter \* H

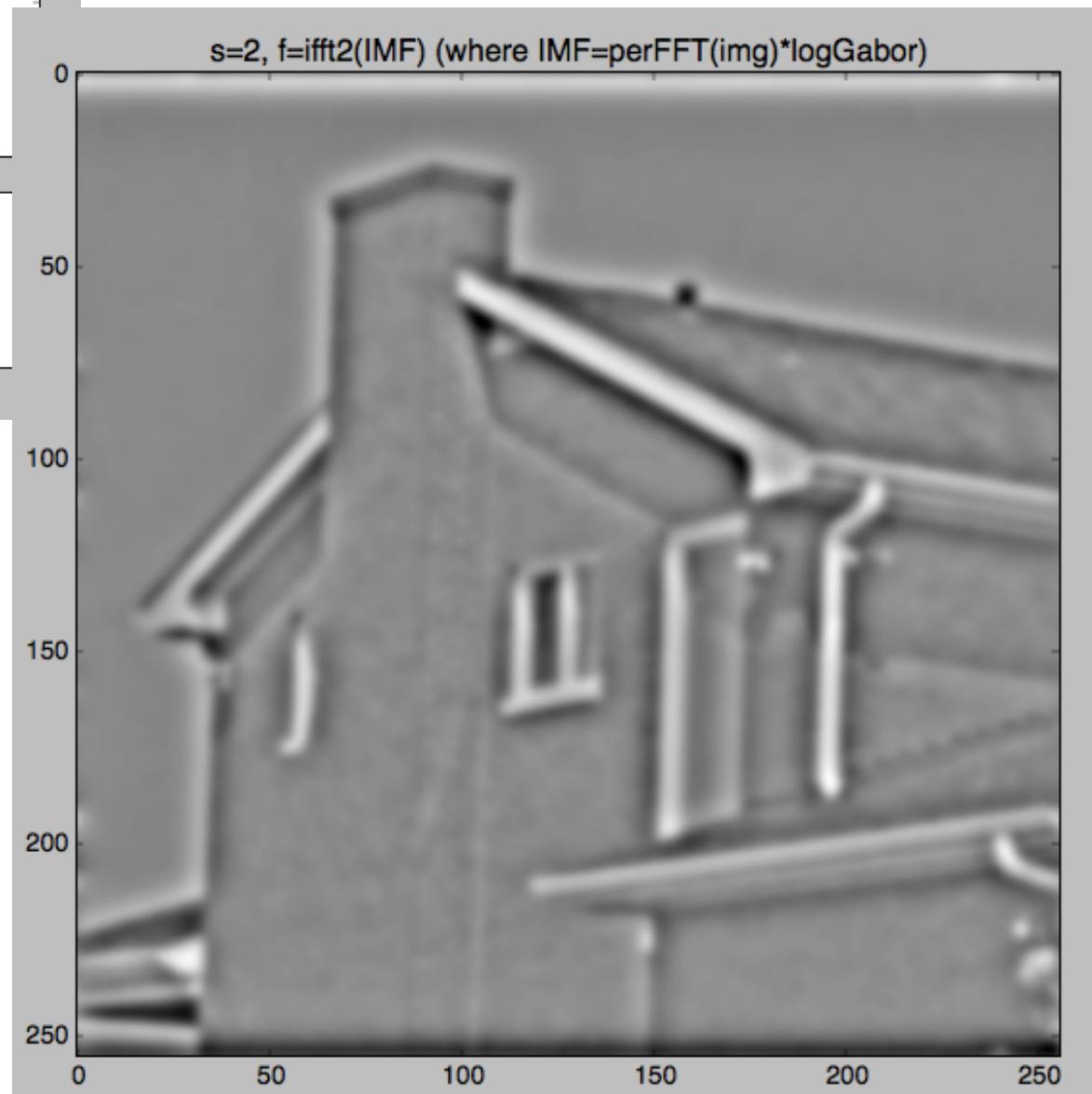


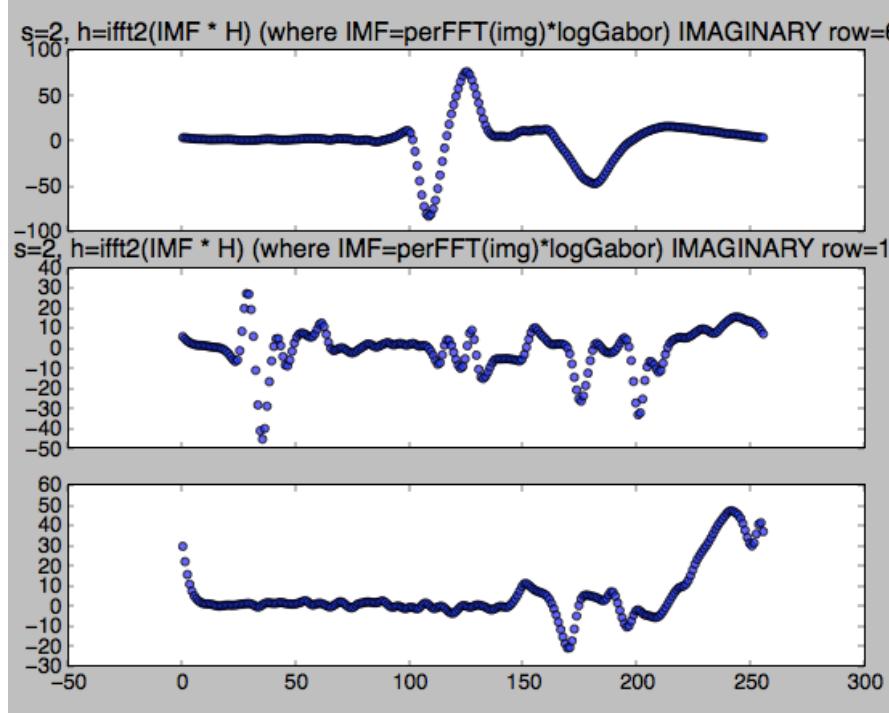
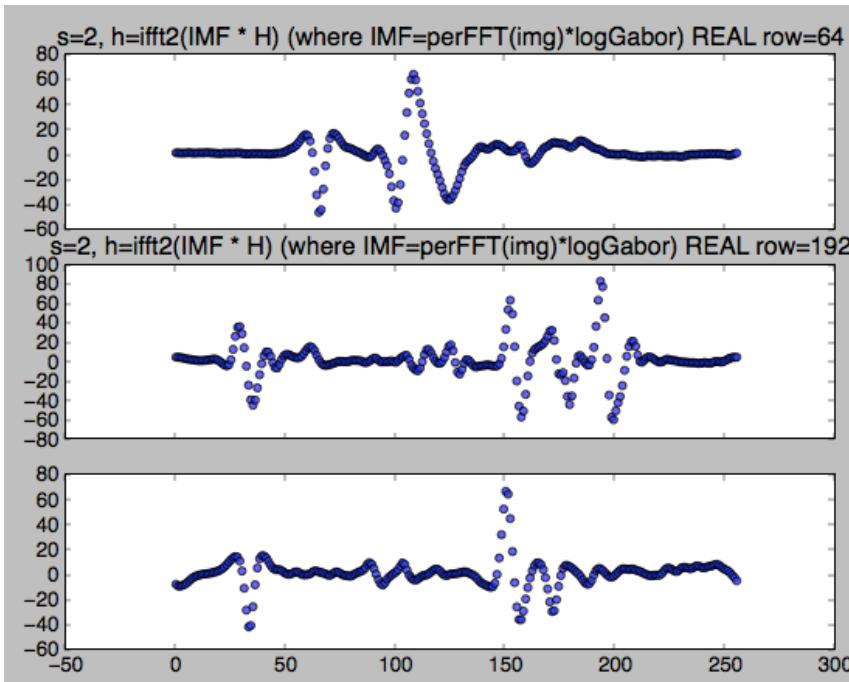
$s=2, \text{ IMF}=\text{perFFT(img)} * \text{logGabor}$



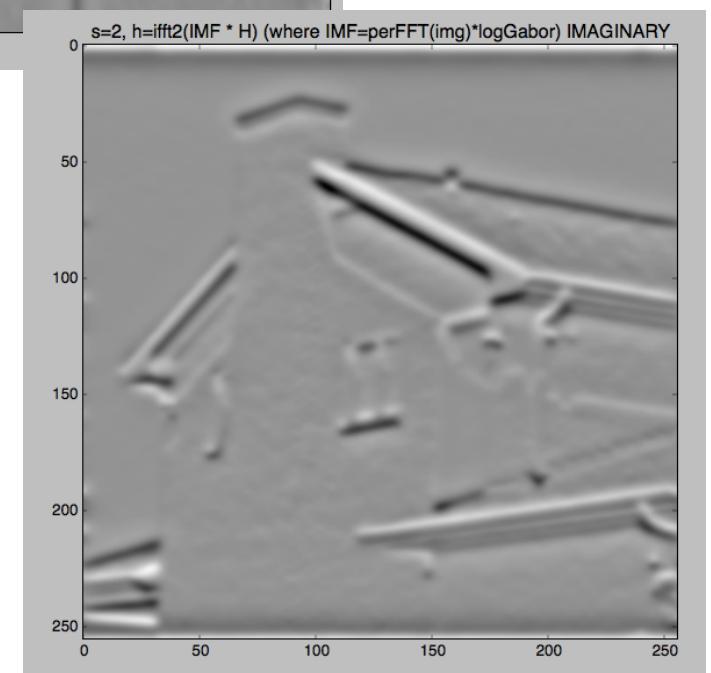
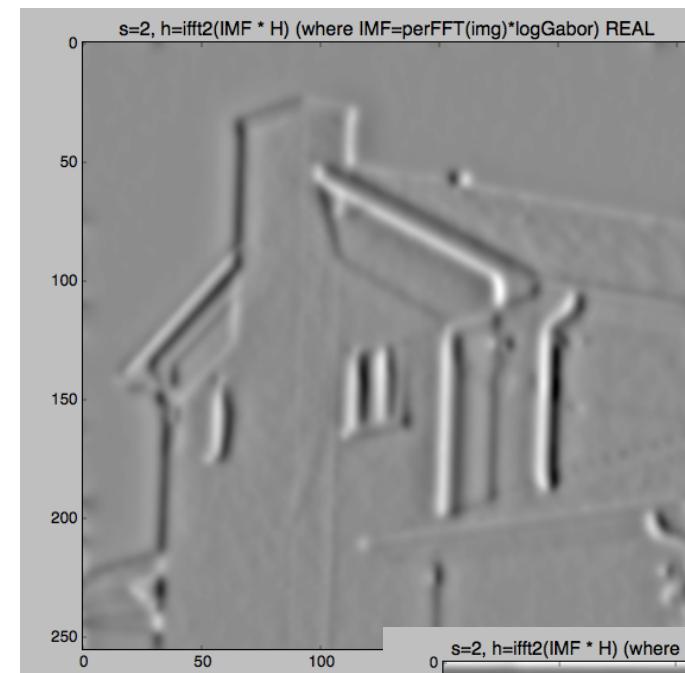


s=2, f=ifft2(IMF)  
(where IMF=perFFT(img)\*logGabor)

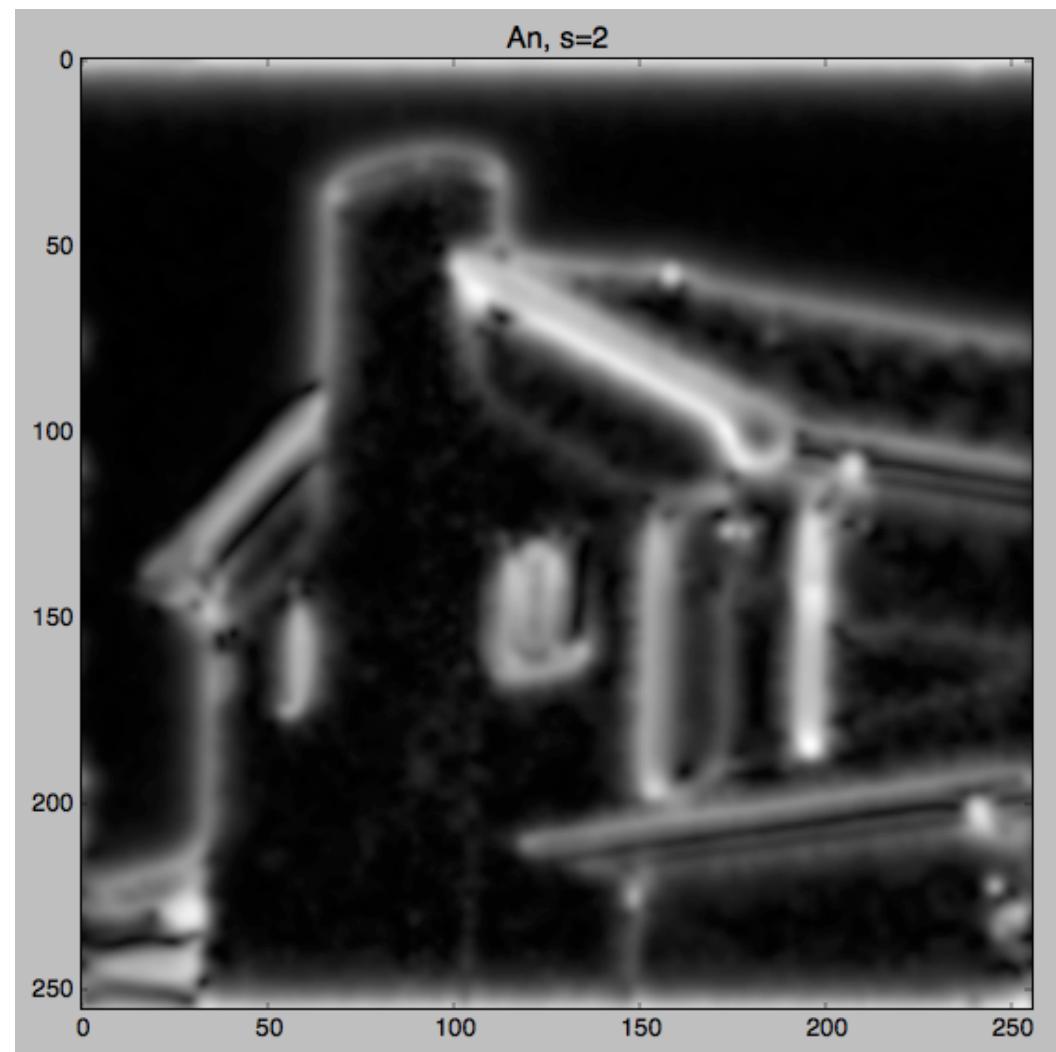
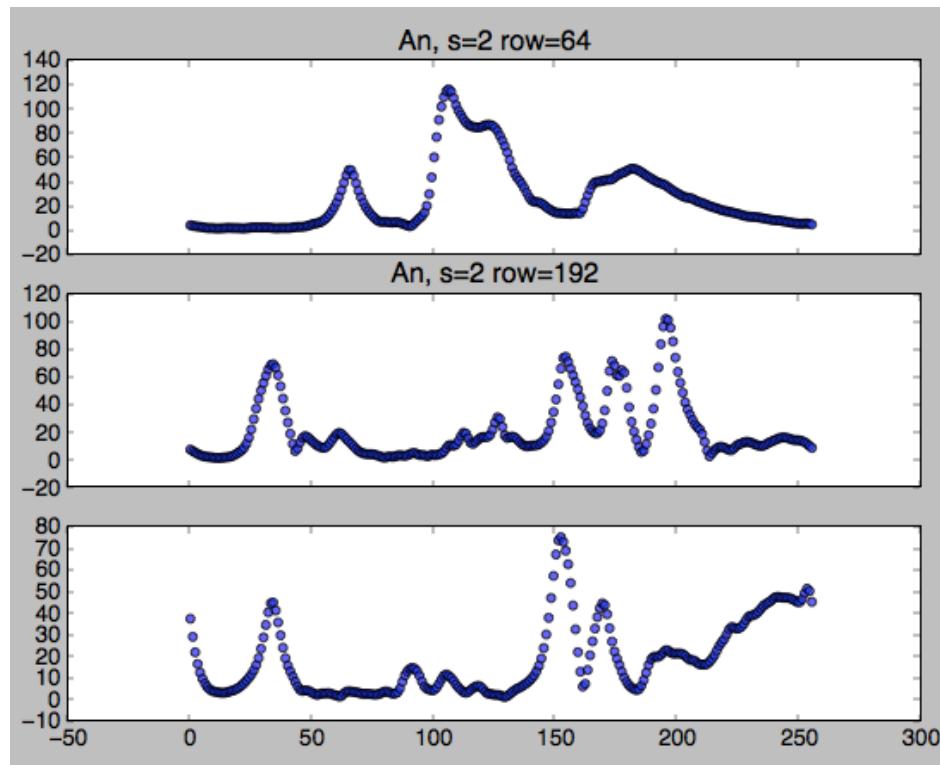




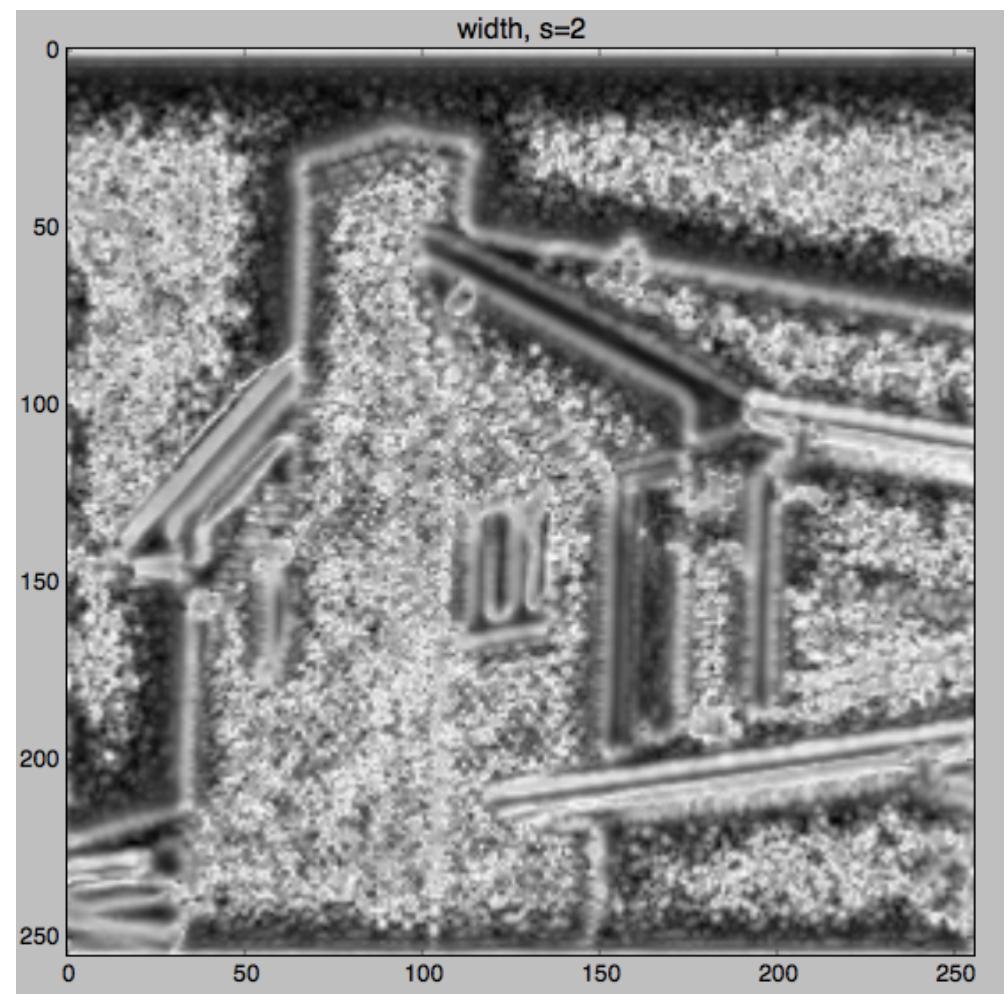
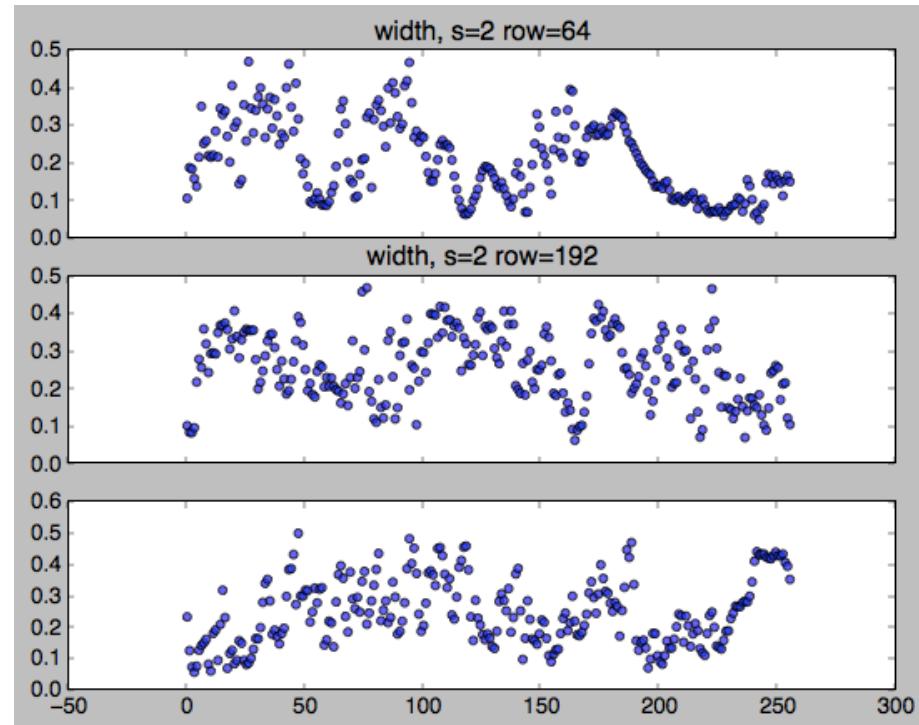
**s=2, h=ifft2(IMF \* H)  
(where IMF=perFFT(img)\*logGabor)**



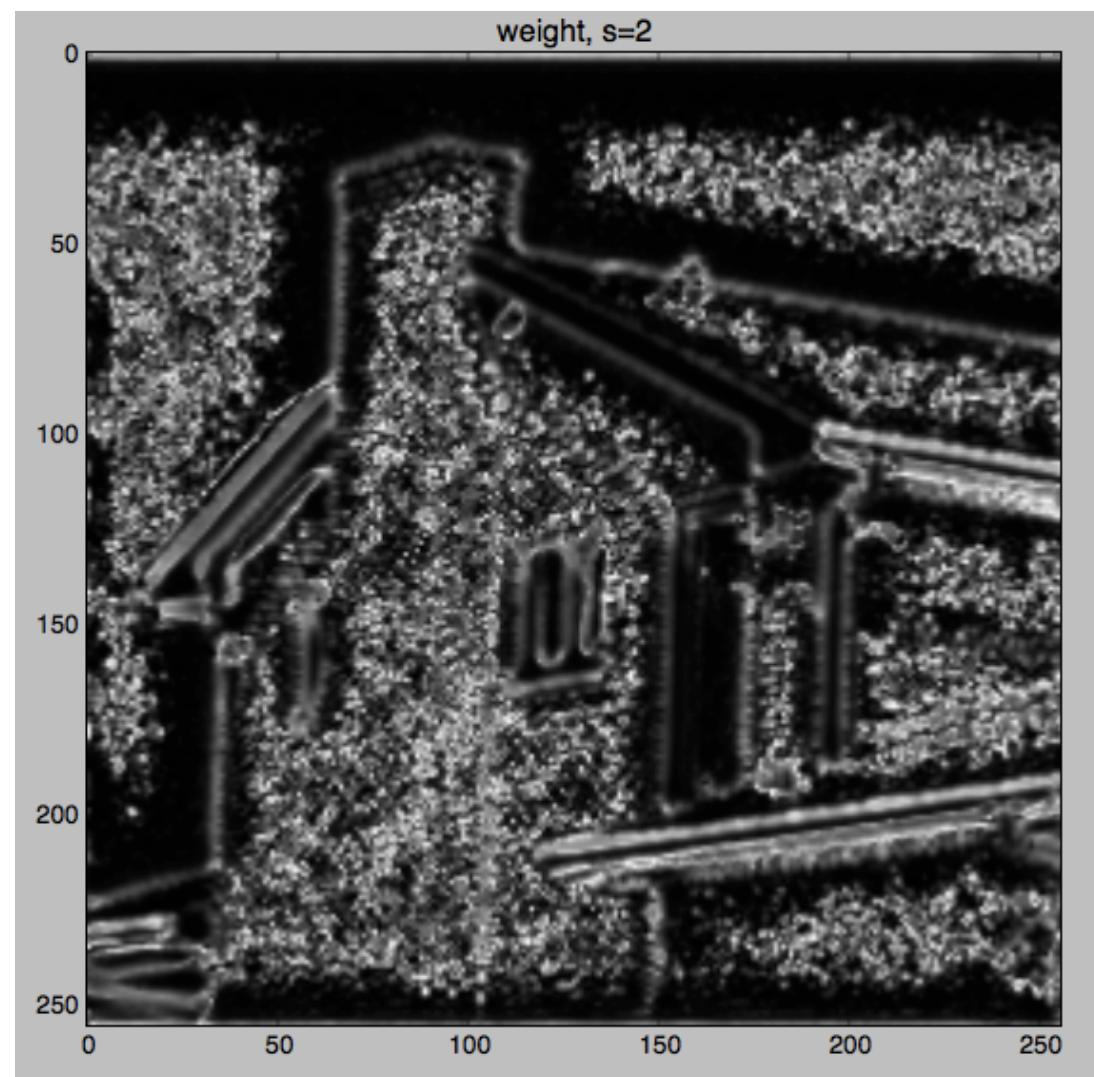
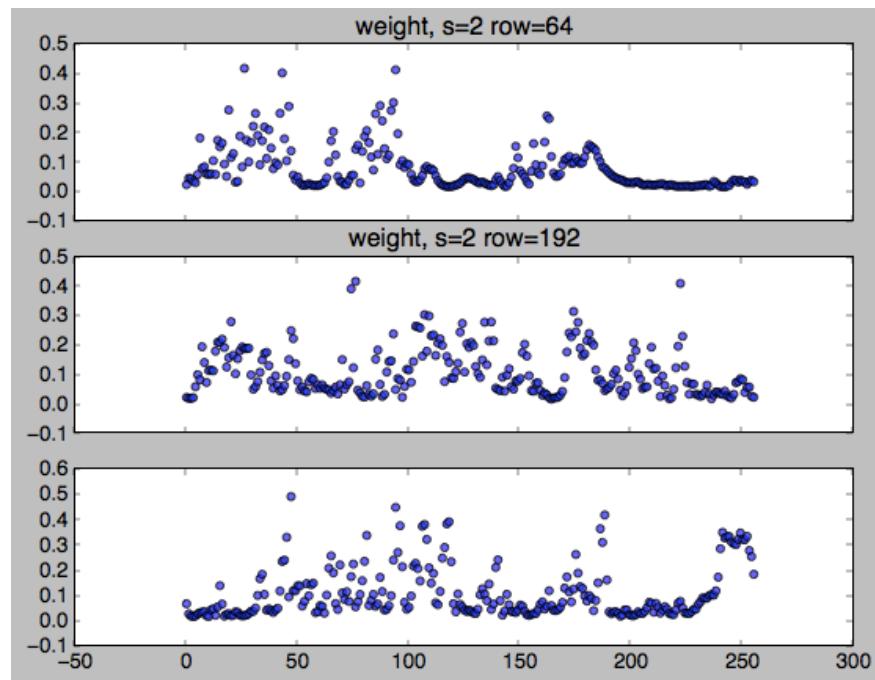
```
s=2, An = sqrt(f * f + h1 * h1 + h2 * h2)
(where f=ifft2(perFFT(img)*logGabor)
and h=ifft2(perFFT(img)*logGabor * H) )
```



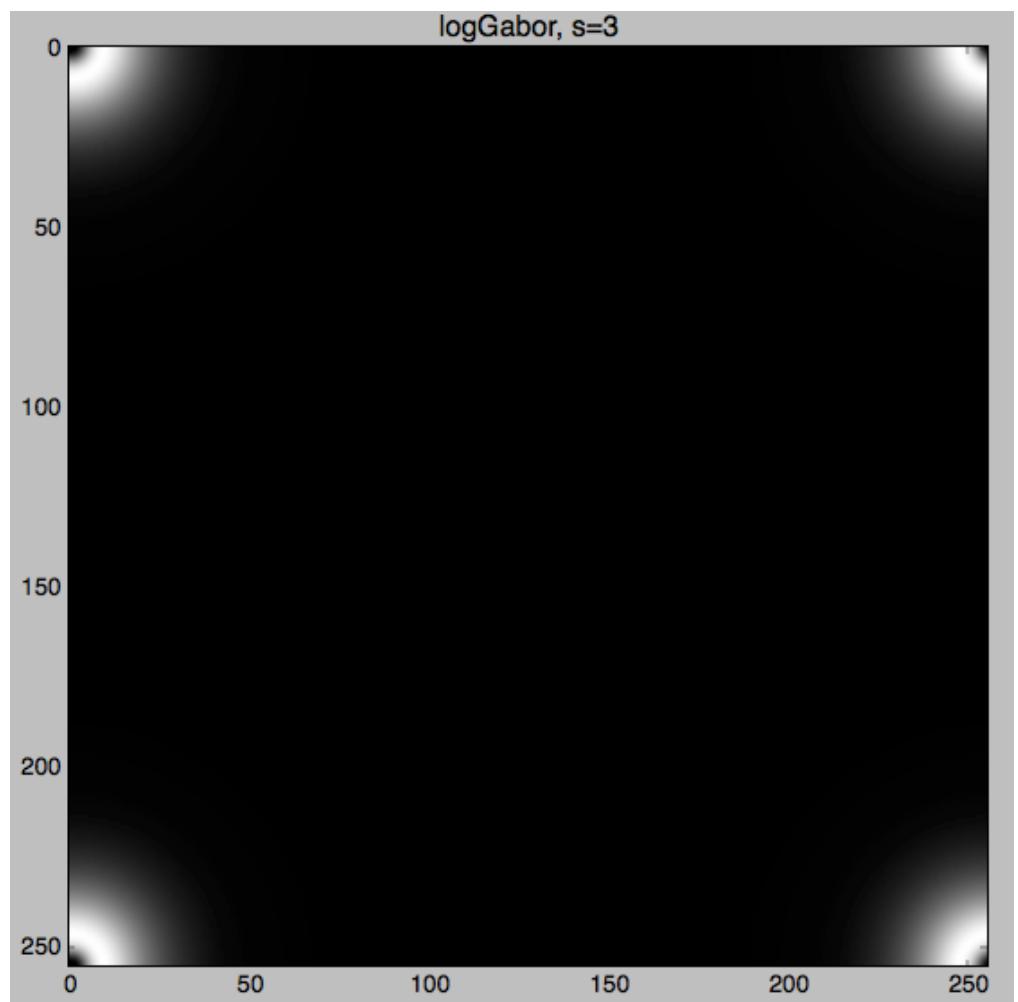
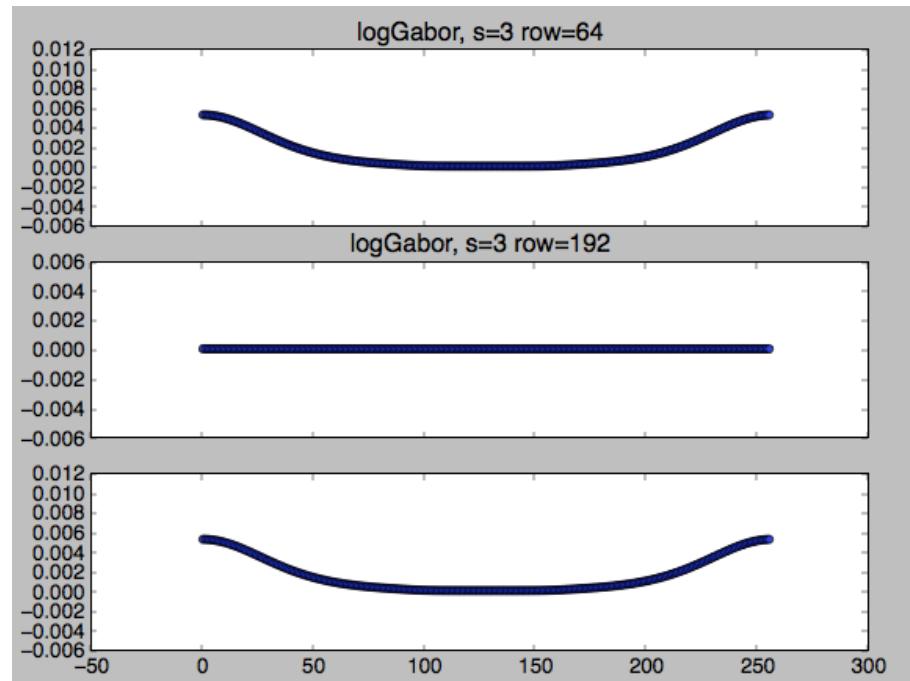
**s=2, width = (sumAn / (maxAn + epsilon) - 1.) / (nscale - 1)**



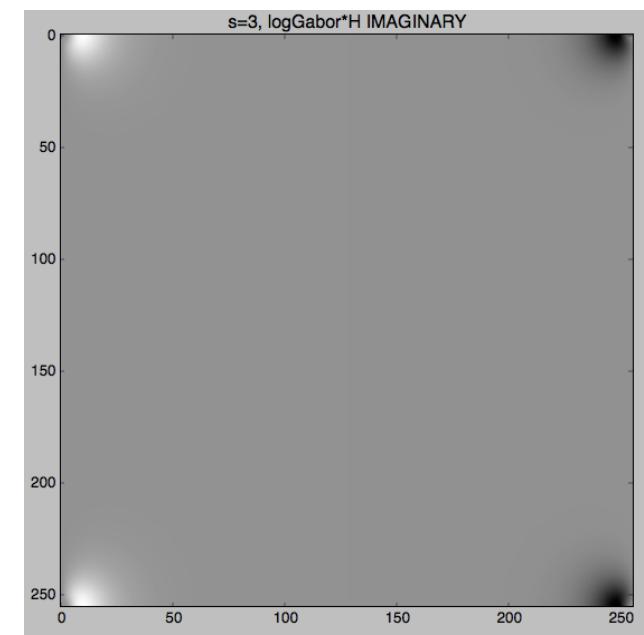
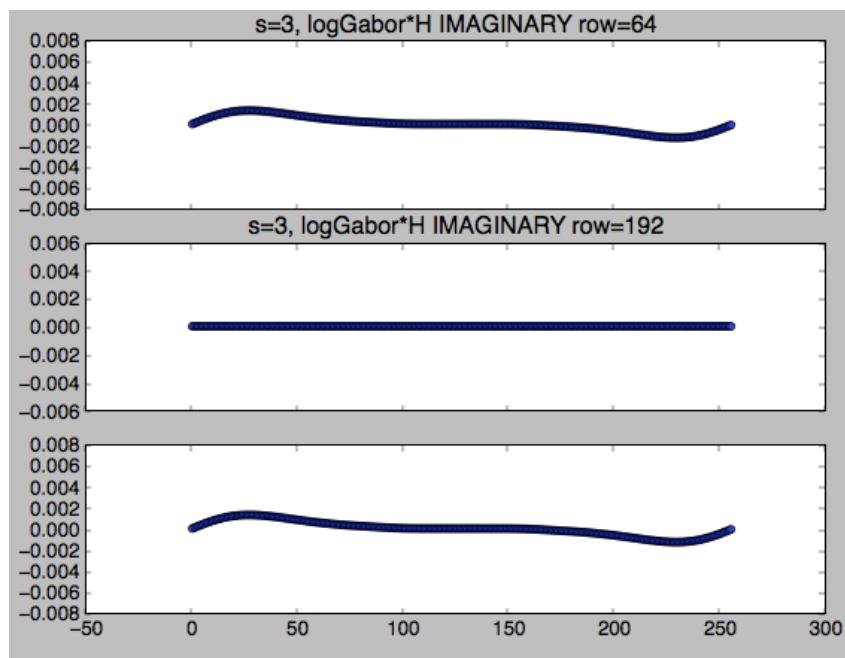
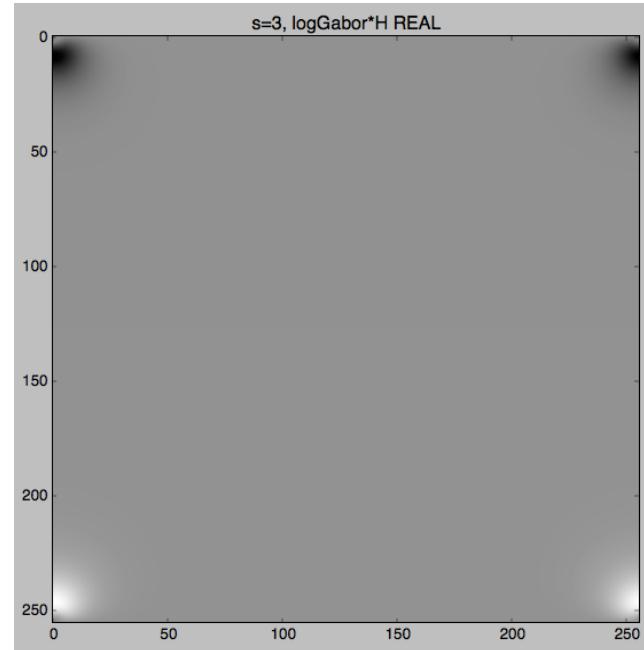
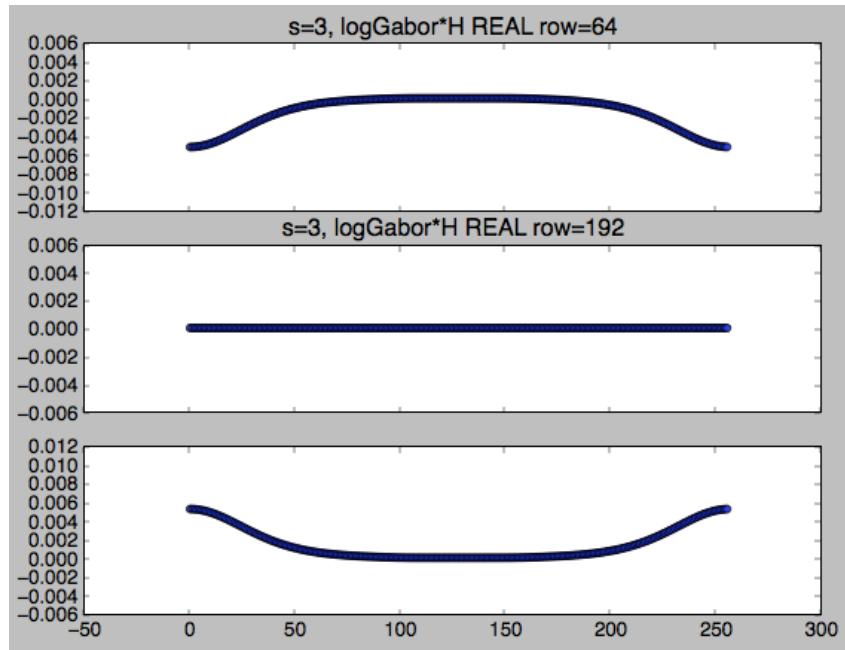
**s=2, weight =  $1. / (1. + \exp(g * (\text{cutoff} - \text{width})))$**   
(where cutoff=0.5, g=10.)



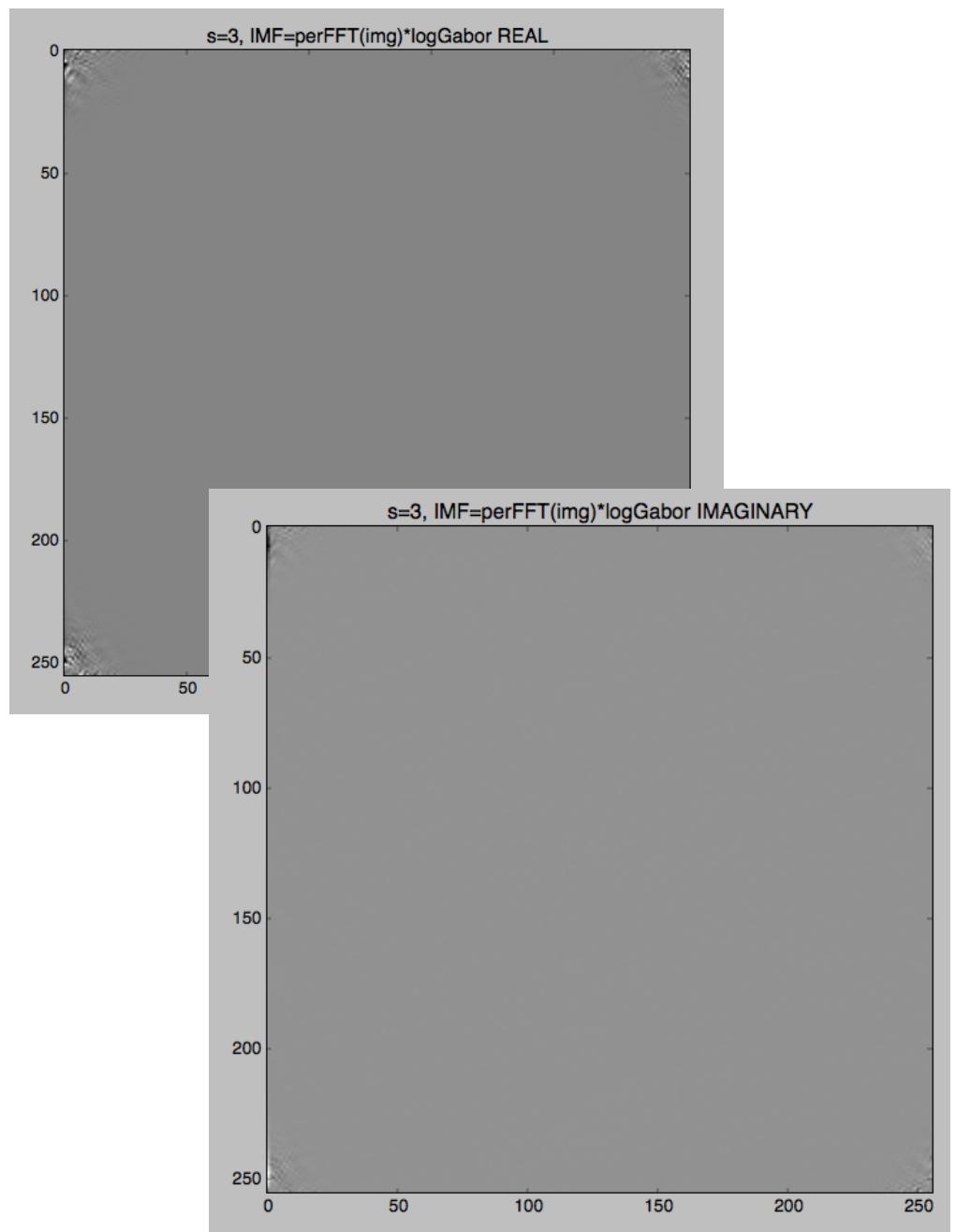
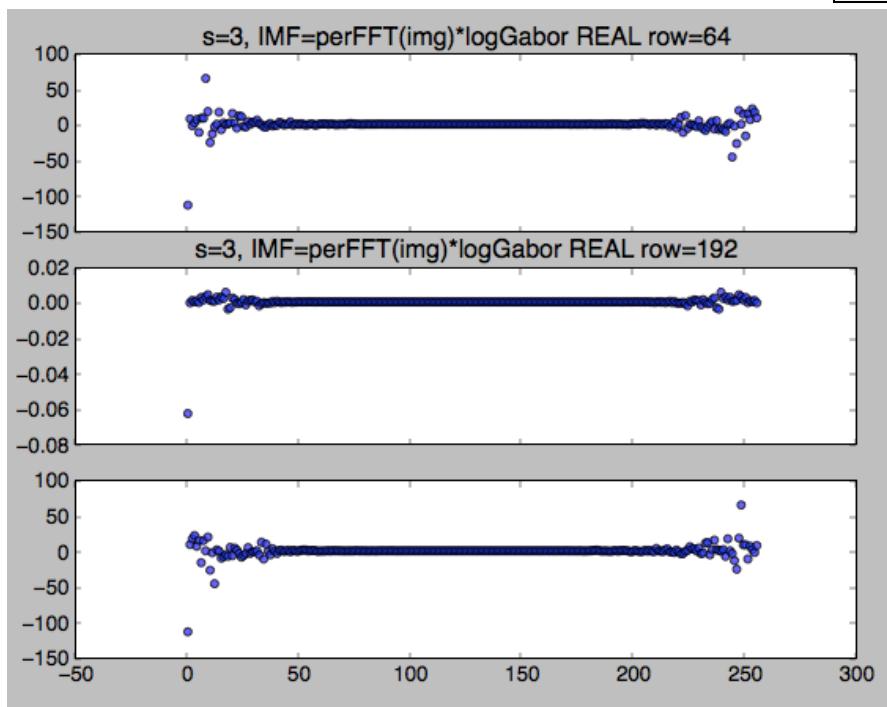
$s=3$ , logGabor \* low pass filter

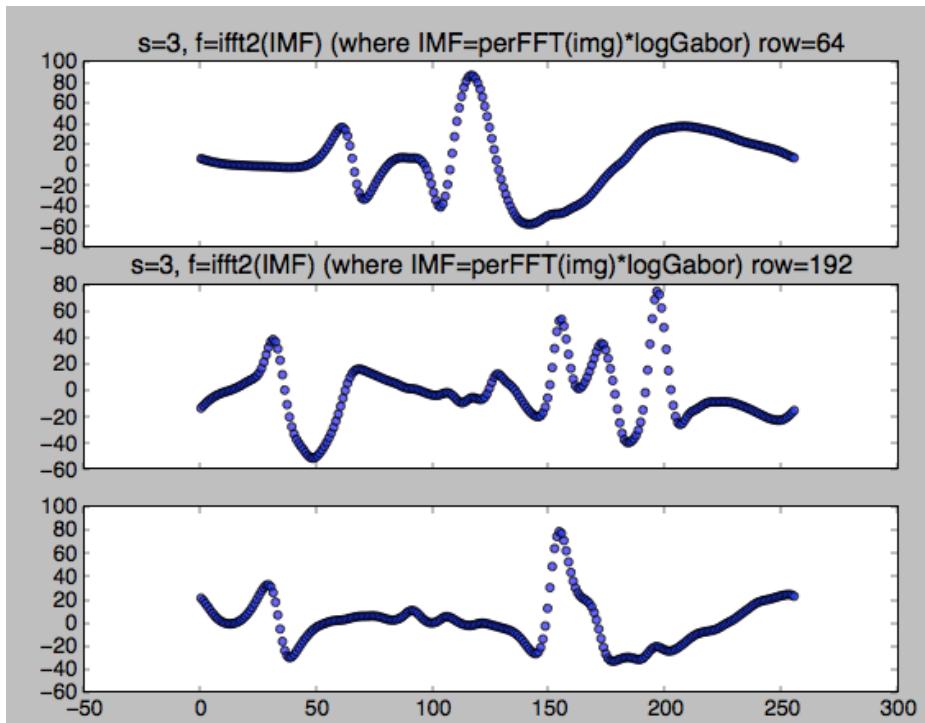


$s=3$ , logGabor \* low pass filter \* H

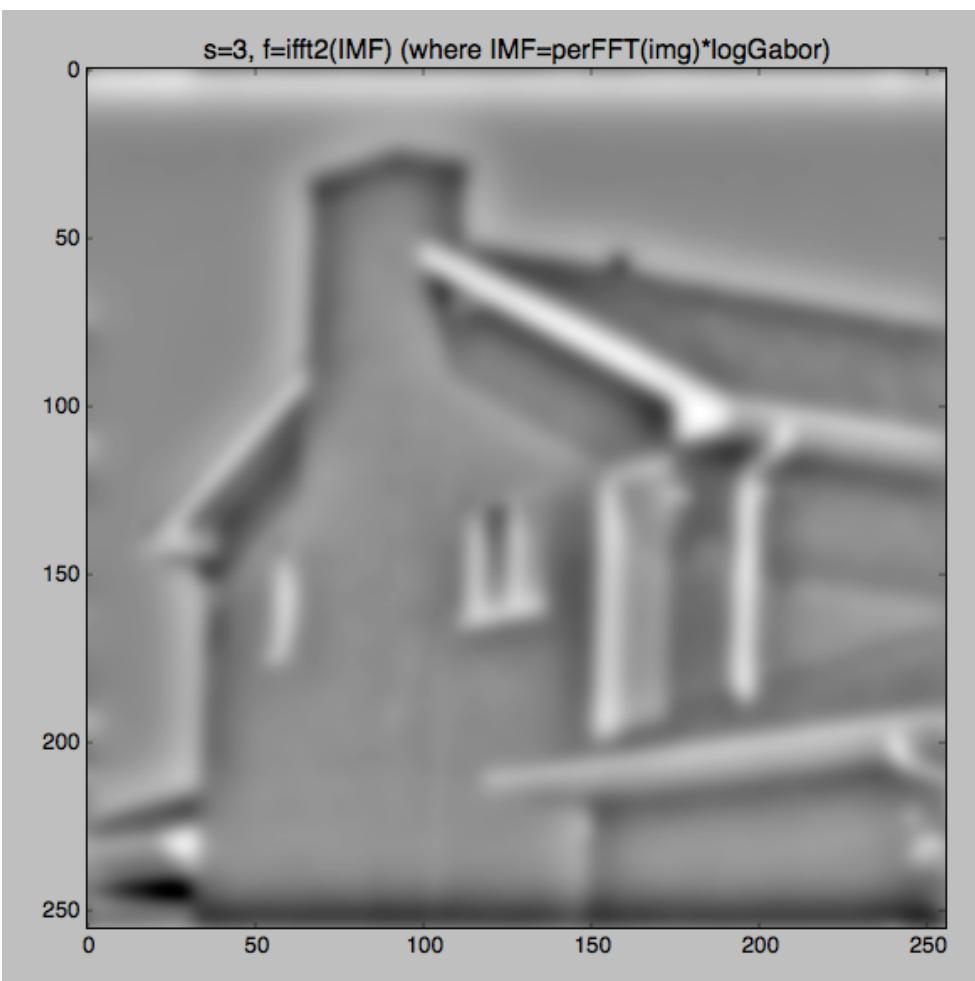


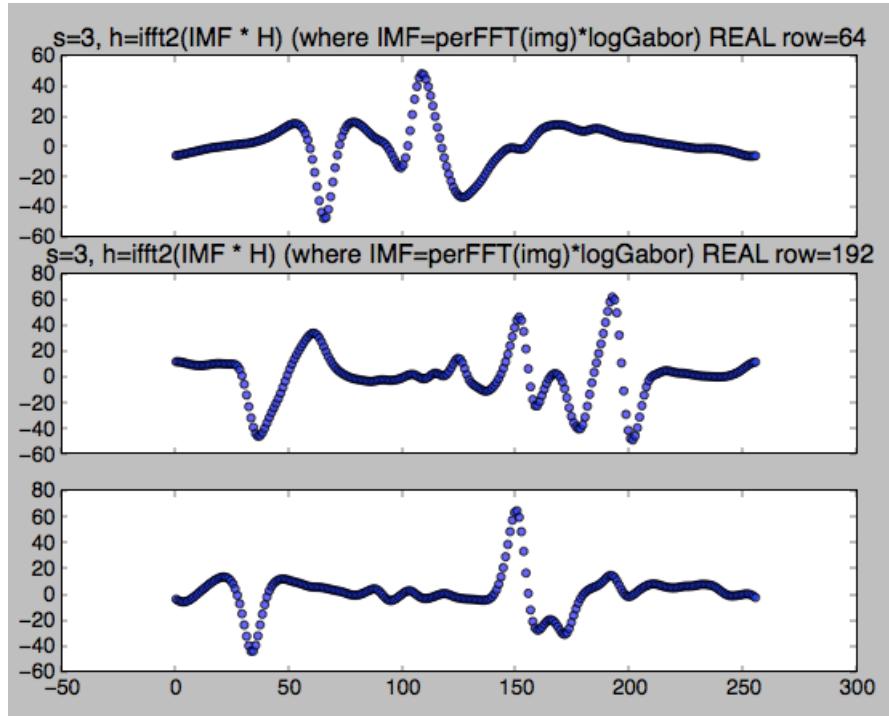
$s=3$ ,  $\text{IMF}=\text{perFFT}(\text{img}) * \text{logGabor}$



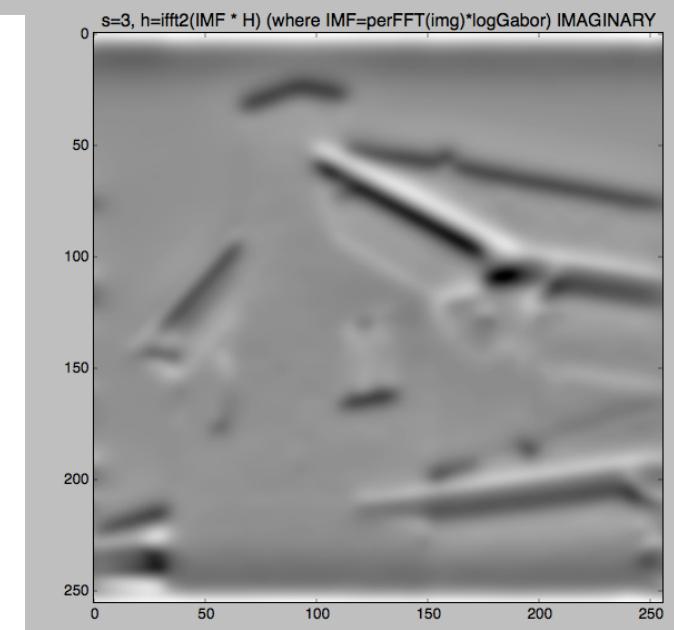
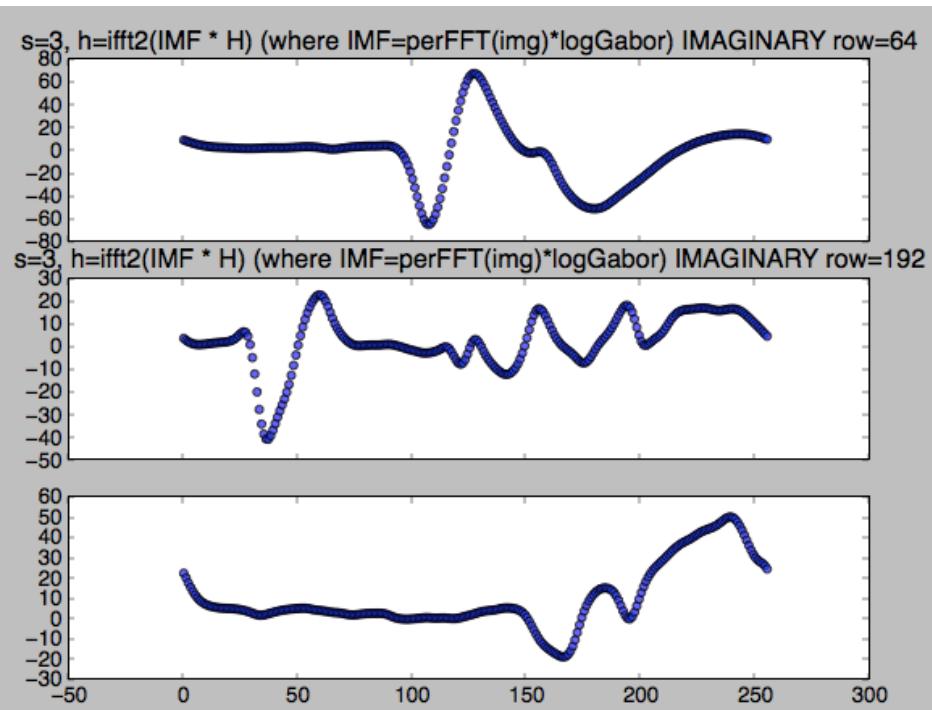
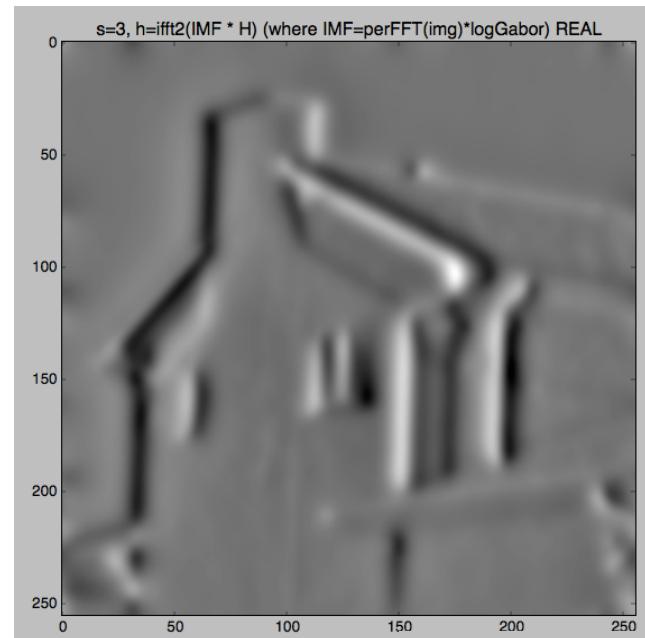


s=3, f=ifft2(IMF)  
(where IMF=perFFT(img)\*logGabor)





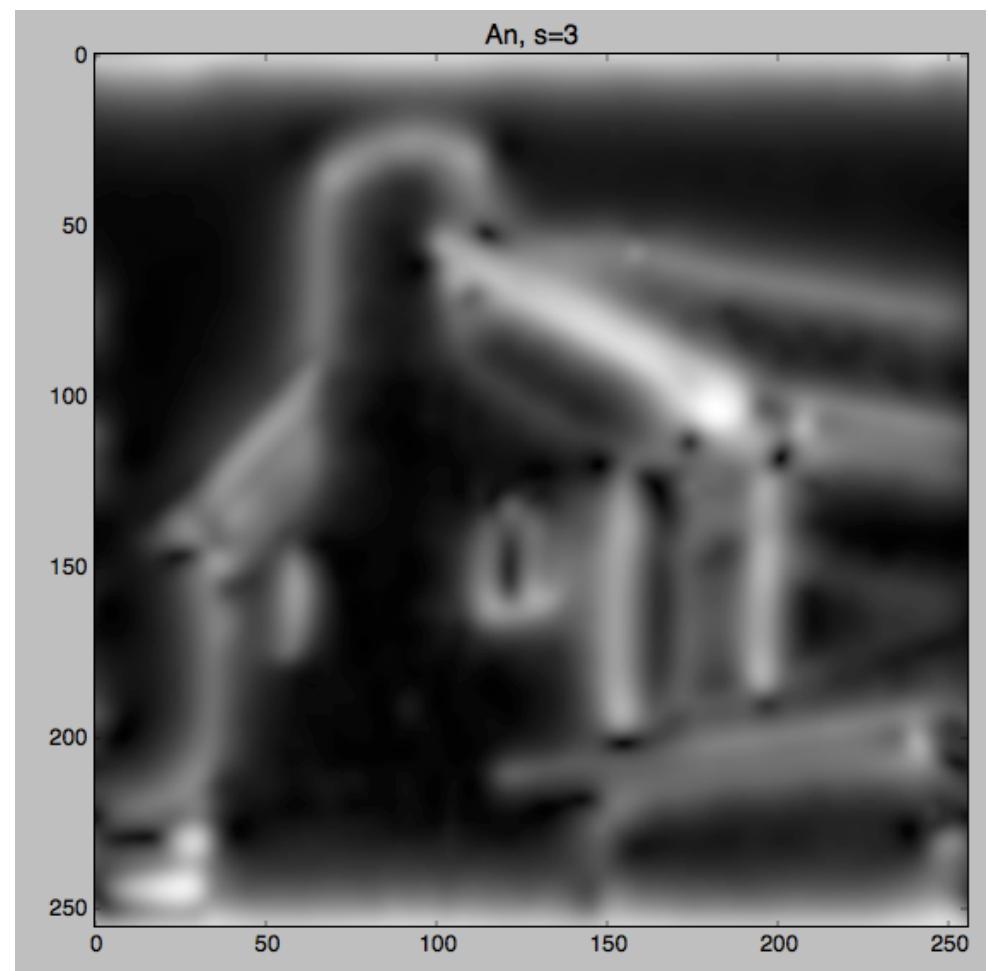
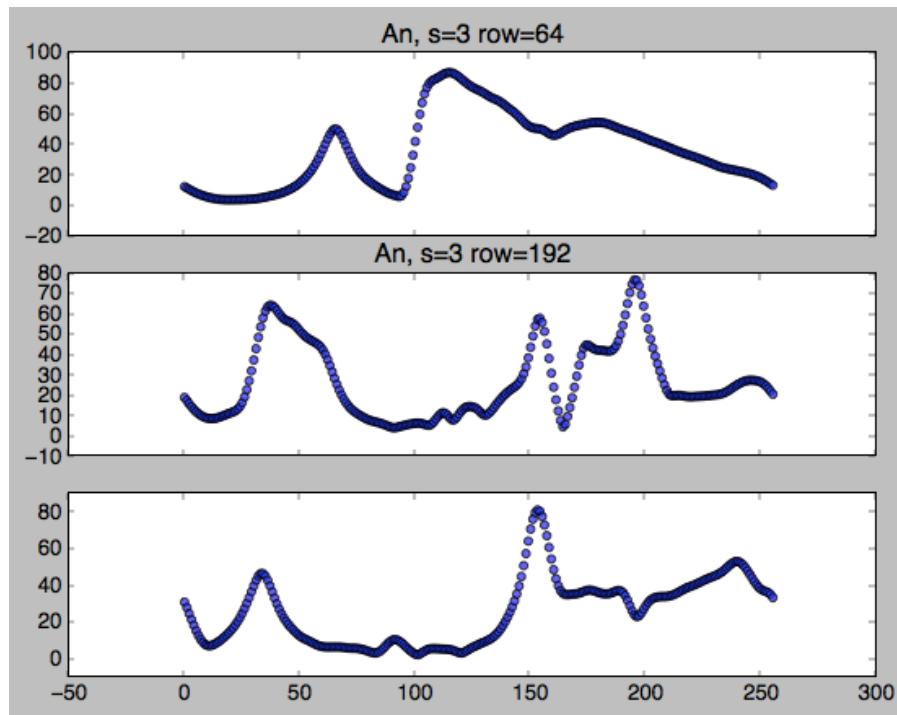
**s=3, h=ifft2(IMF \* H)  
(where IMF=perFFT(img)\*logGabor)**



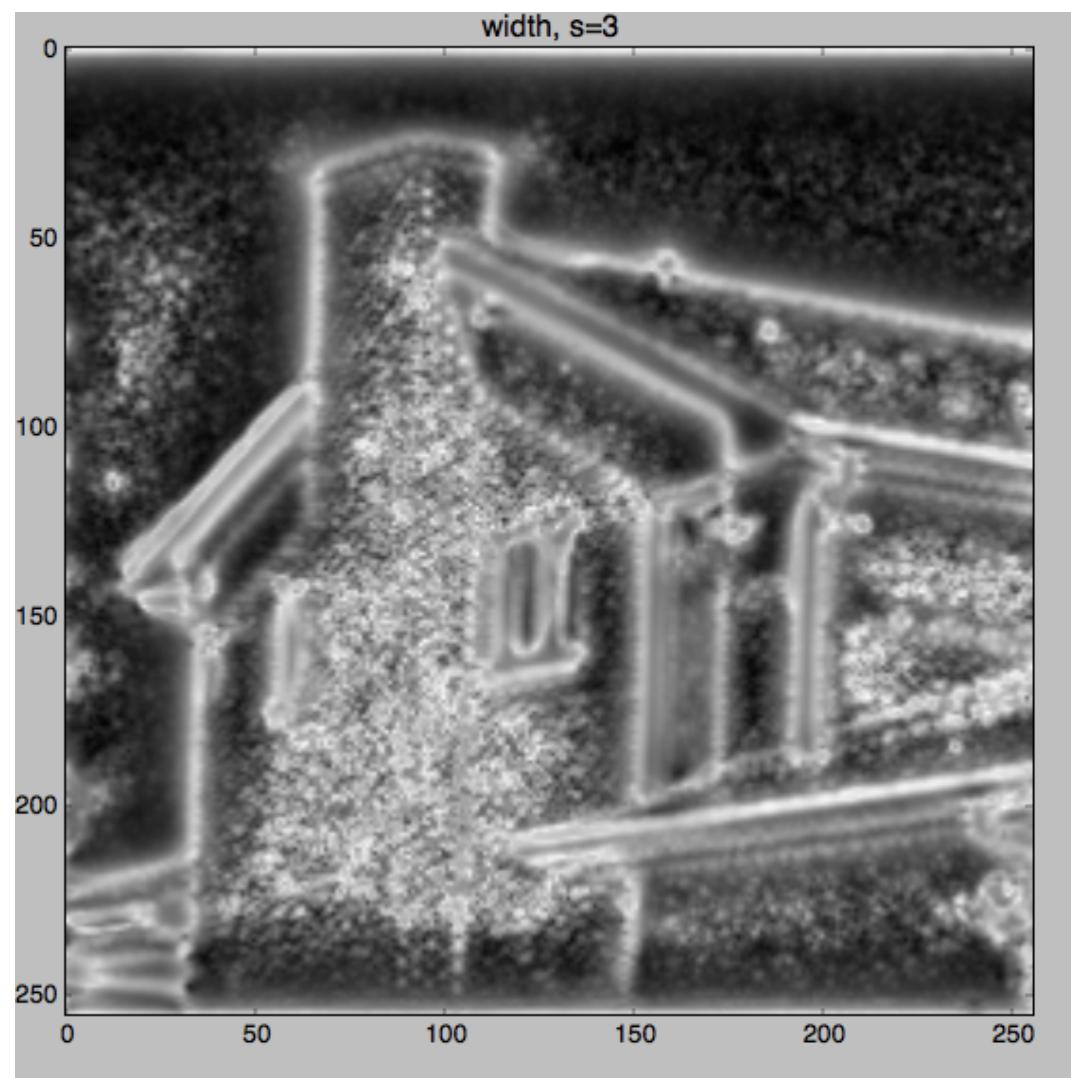
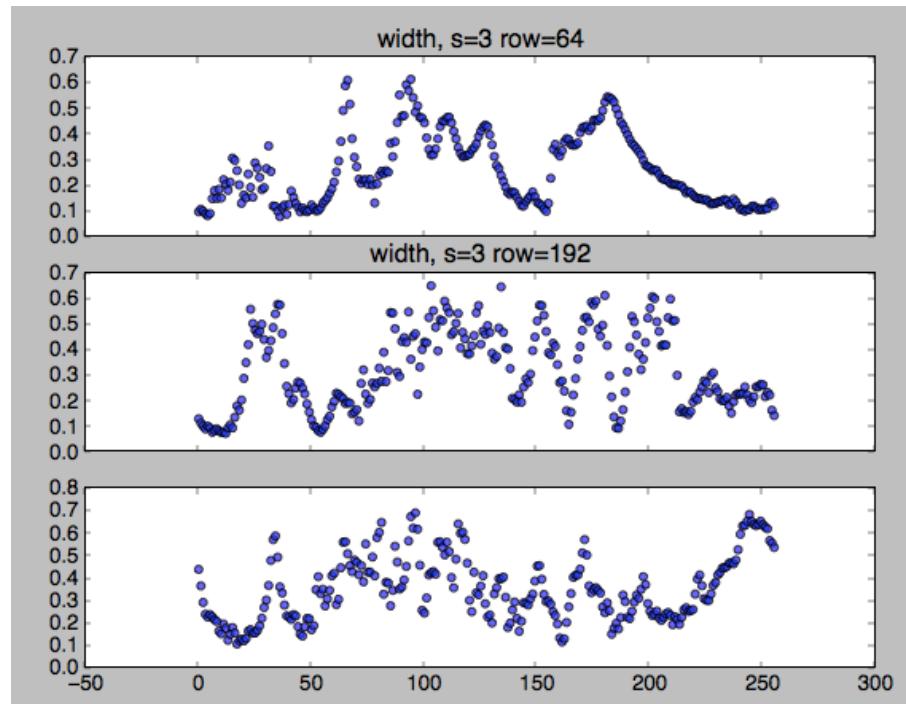
```

s=3, An = sqrt(f * f + h1 * h1 + h2 * h2)
(where f=ifft2(perFFT(img)*logGabor)
and h=ifft2(perFFT(img)*logGabor * H) )

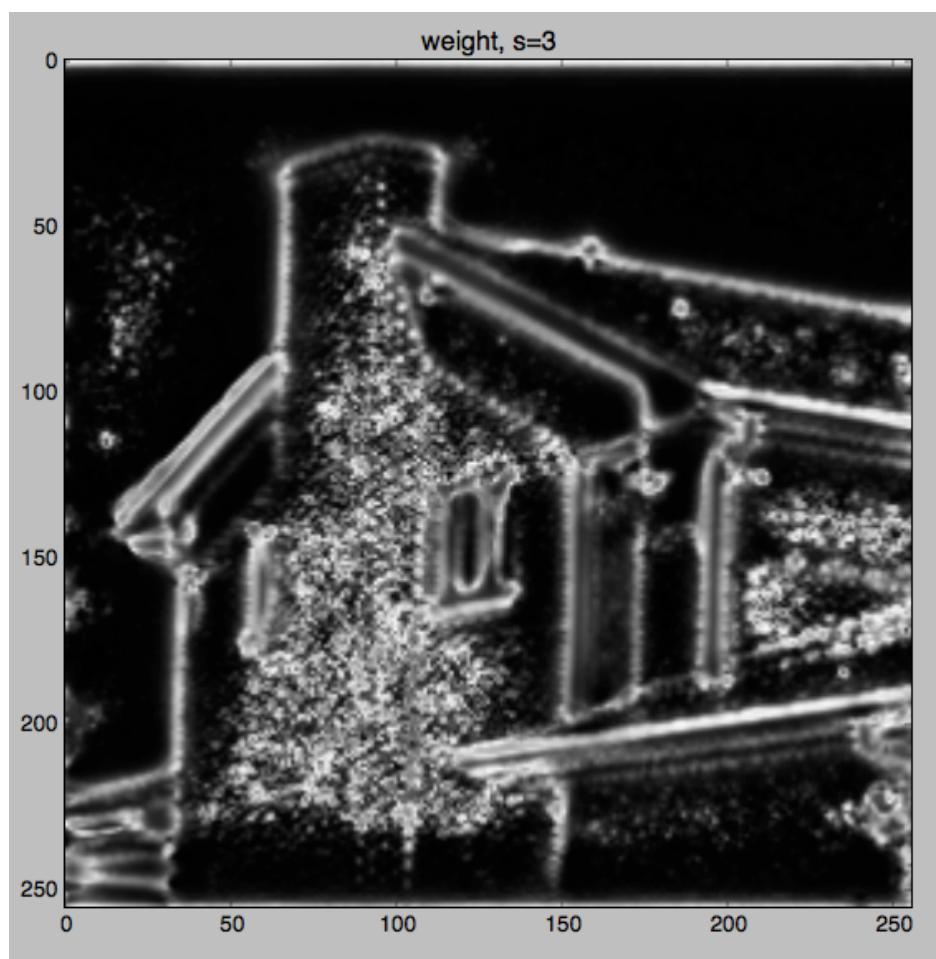
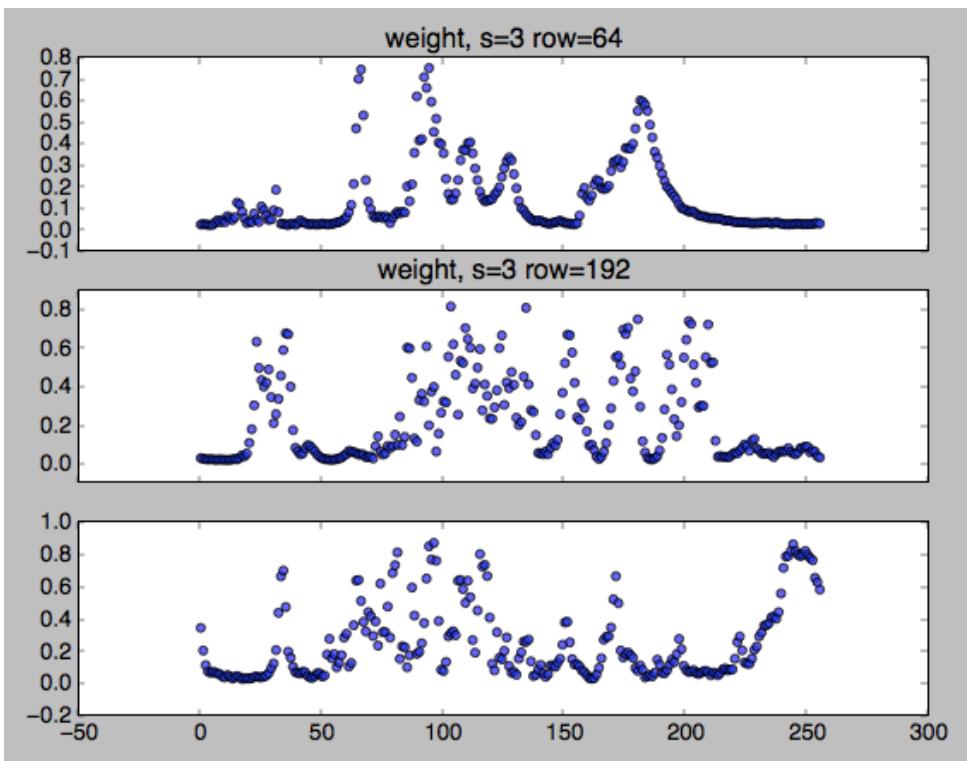
```



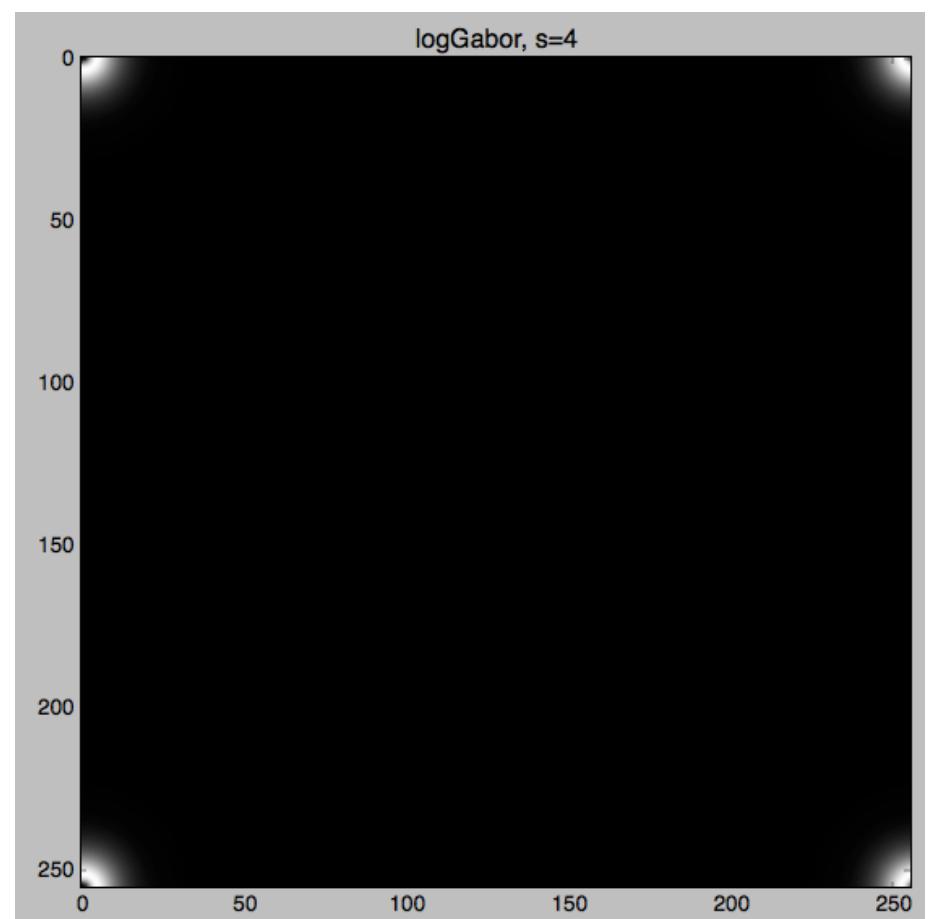
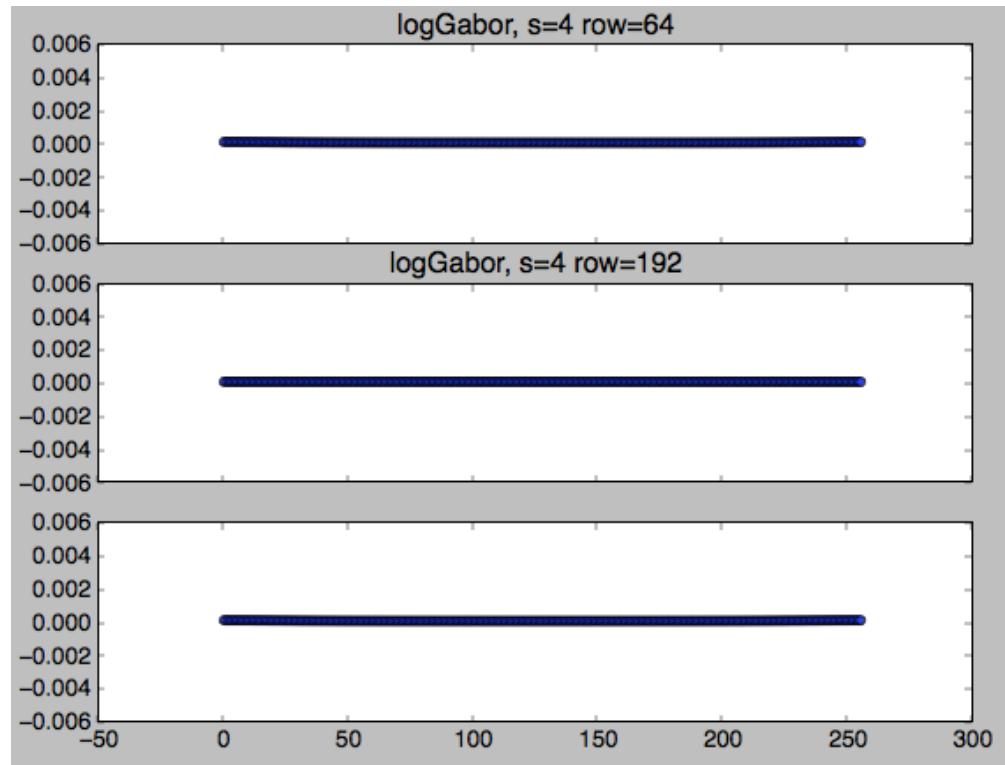
**s=3, width = (sumAn / (maxAn + epsilon) - 1.) / (nscale - 1)**



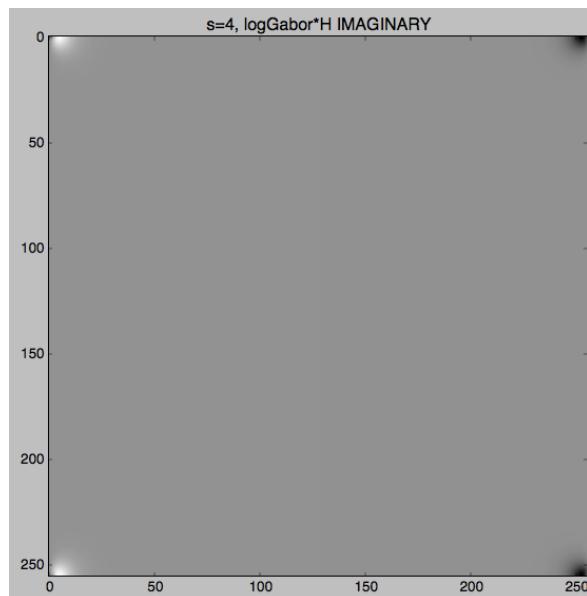
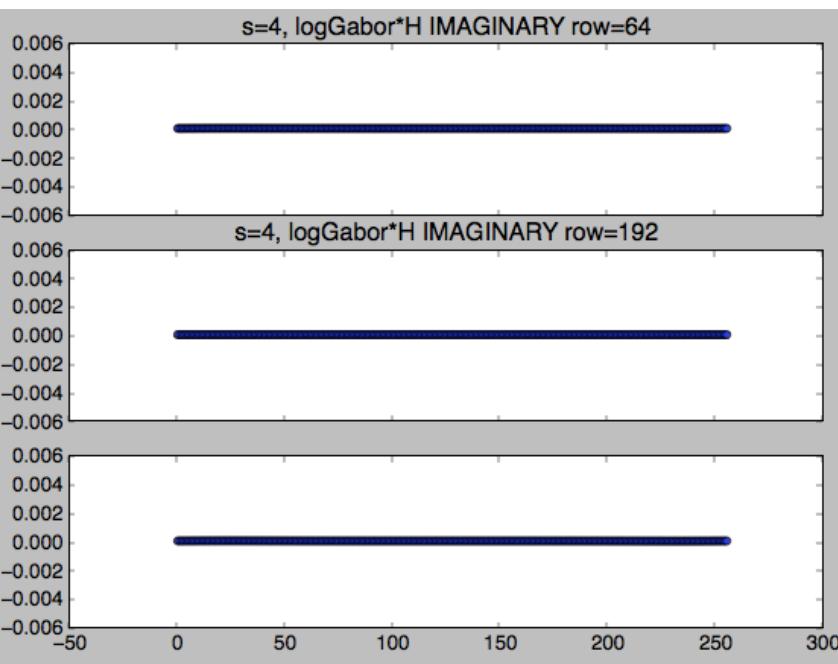
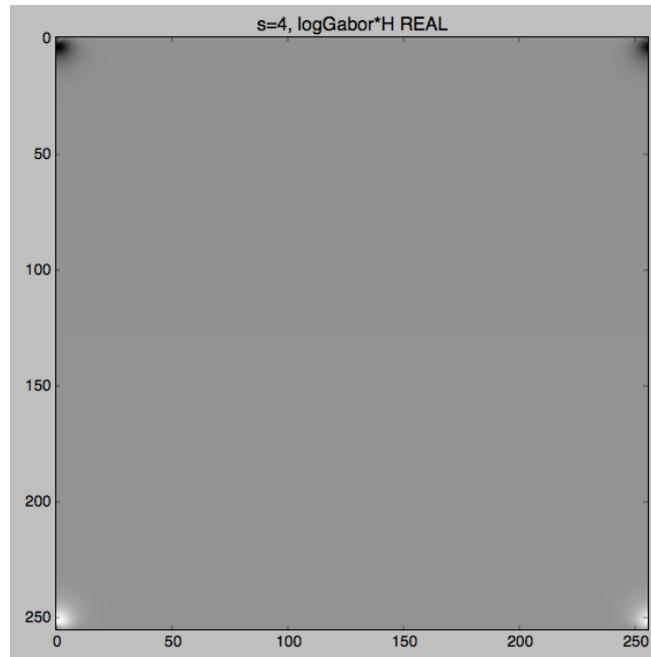
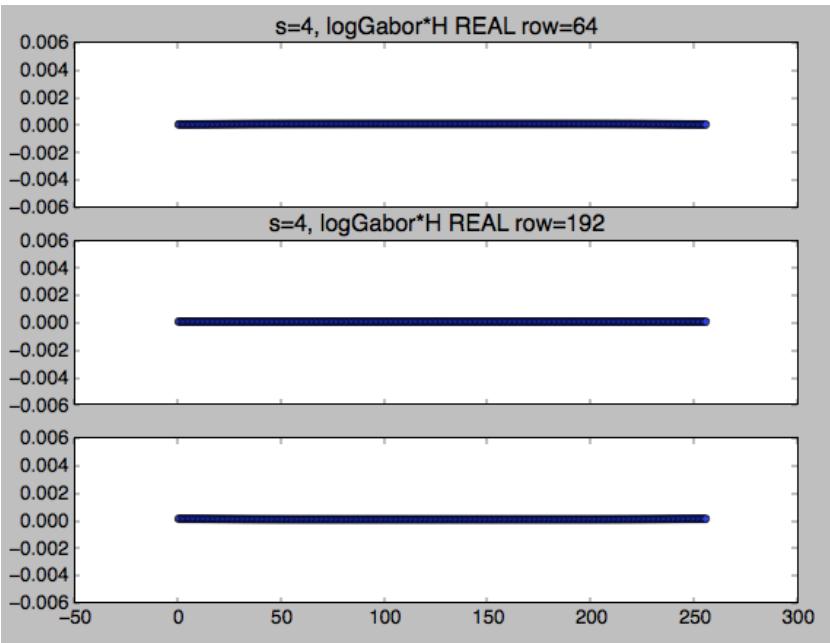
**s=3, weight =  $1. / (1. + \exp(g * (\text{cutoff} - \text{width})))$**   
(where cutoff=0.5, g=10.)



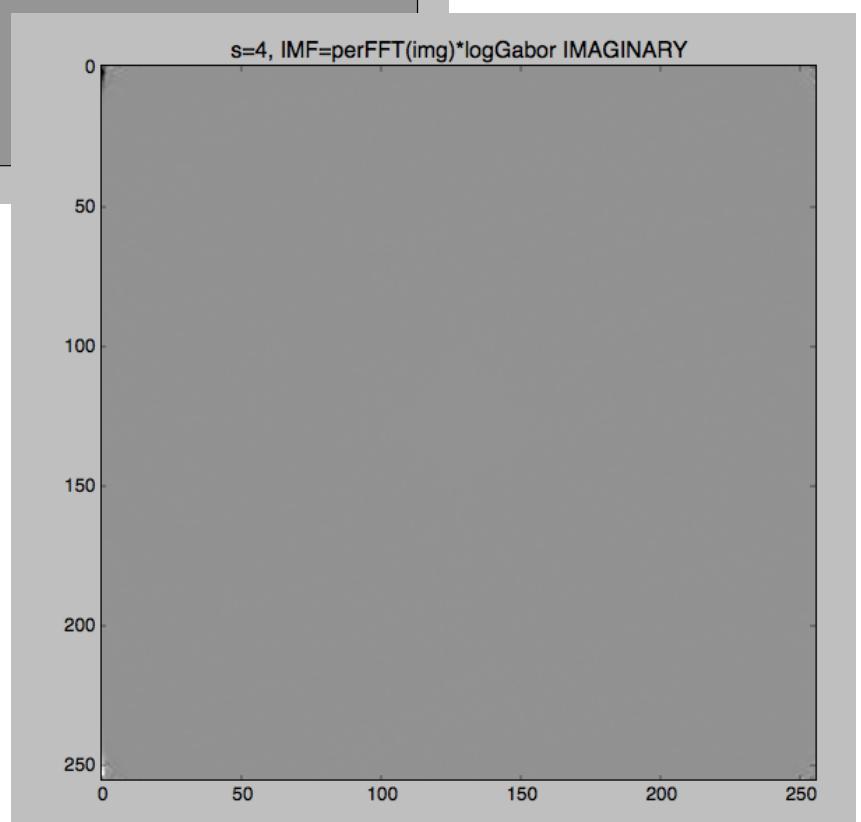
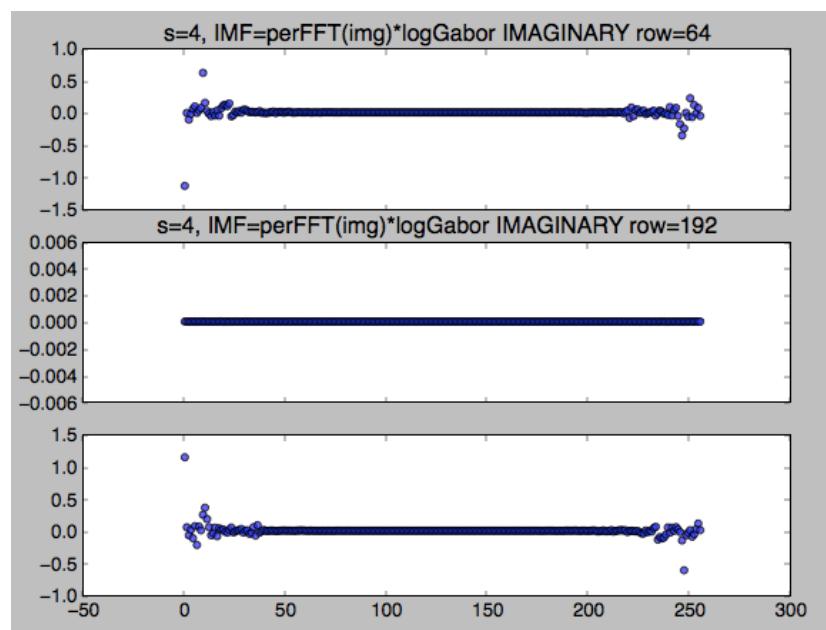
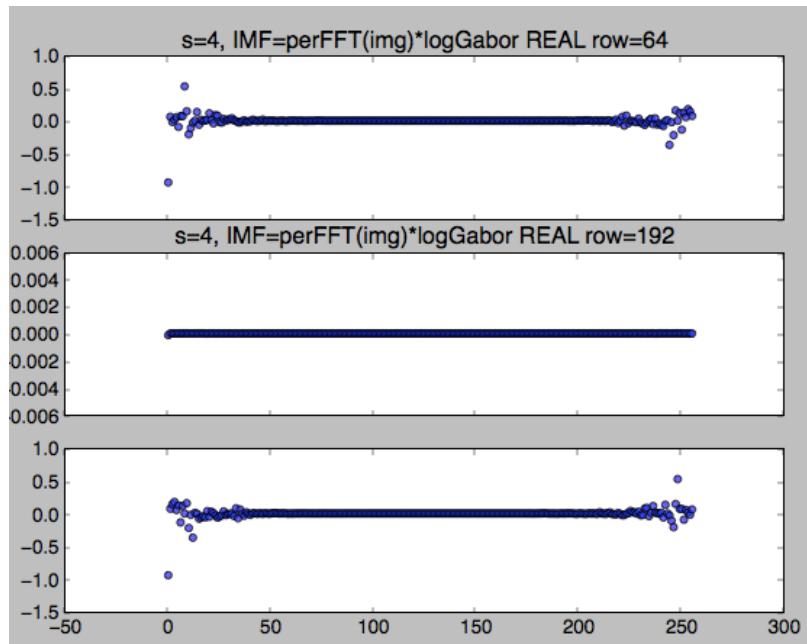
s=4, logGabor \* low pass filter

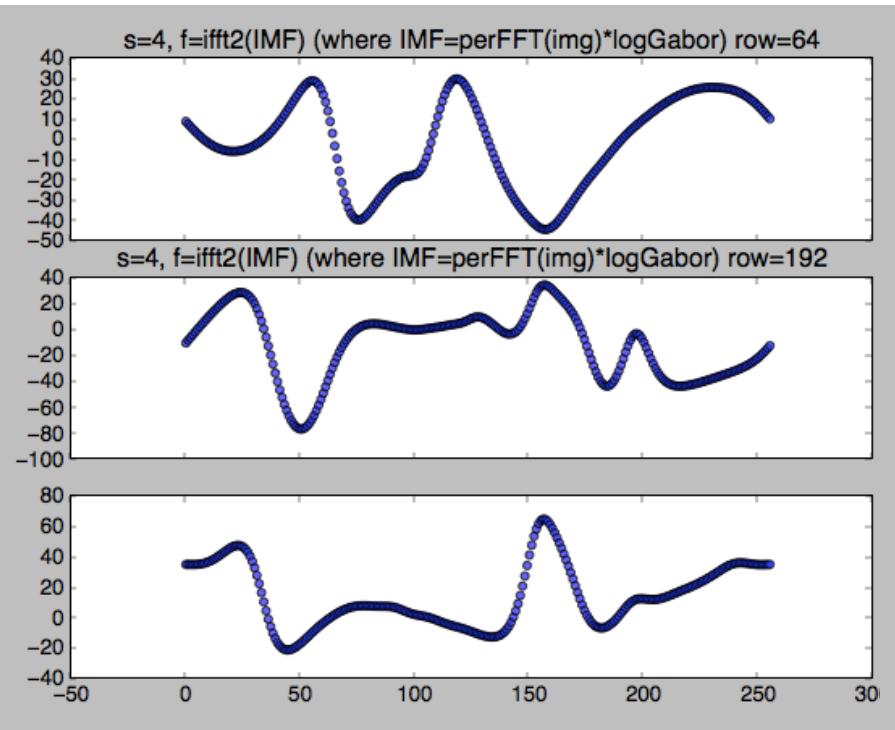


$s=4$ , logGabor \* low pass filter \* H

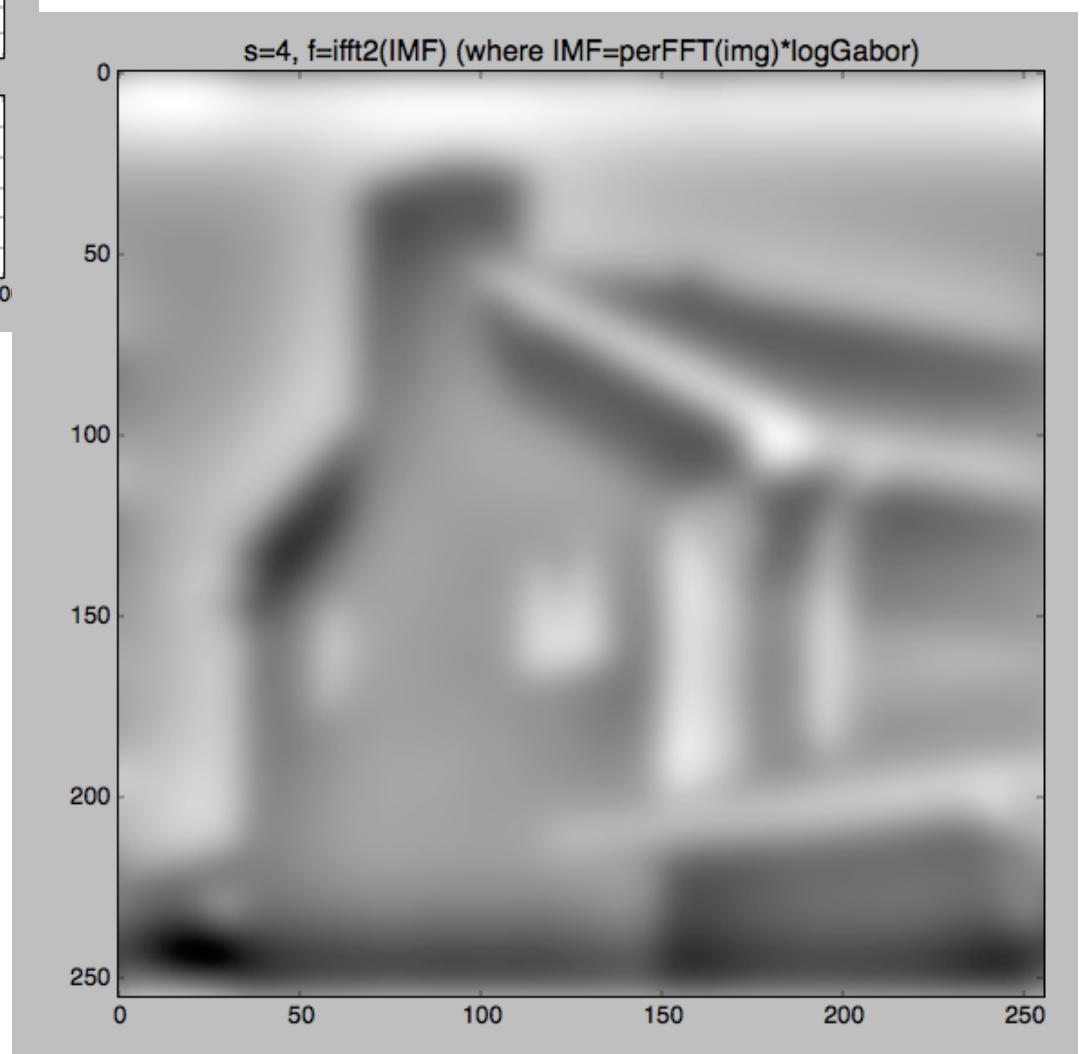


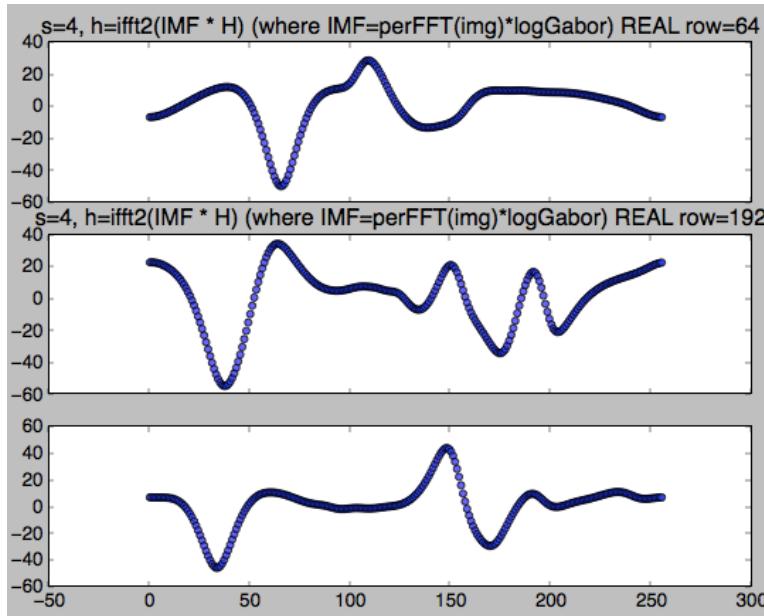
**s=4, IMF=perFFT(img)\*logGabor**



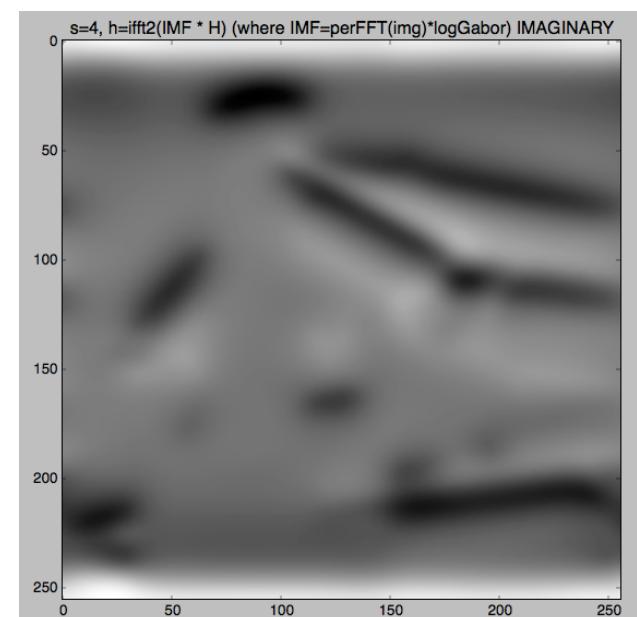
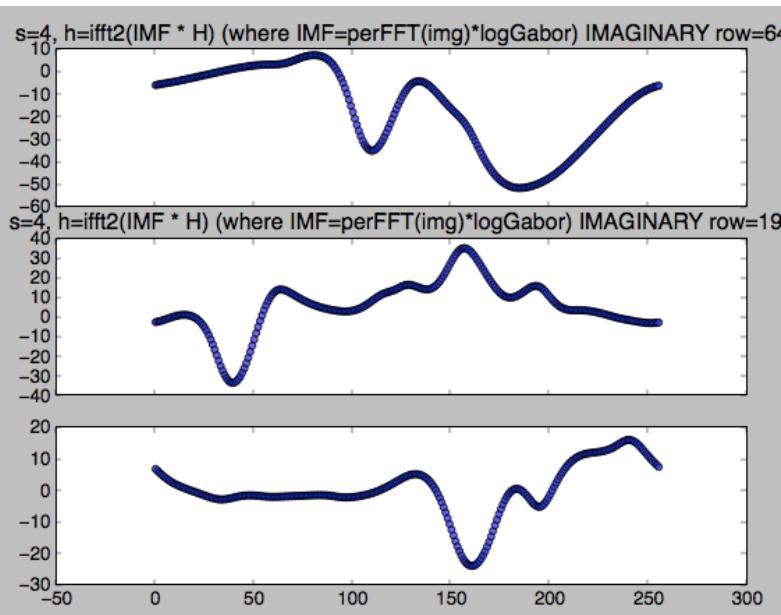
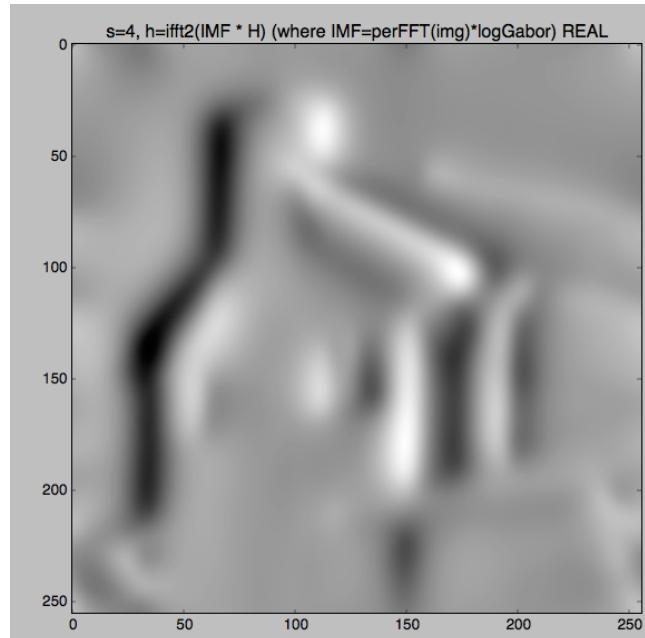


s=4, f=ifft2(IMF)  
(where IMF=perFFT(img)\*logGabor)

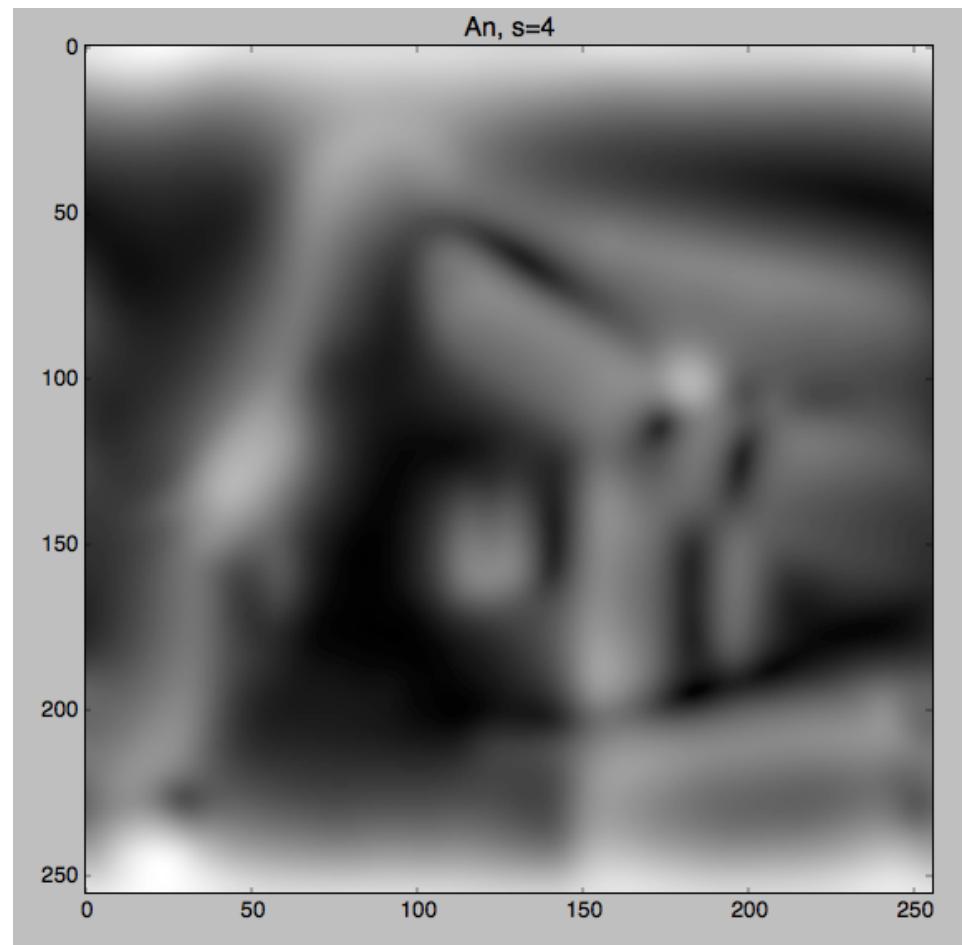
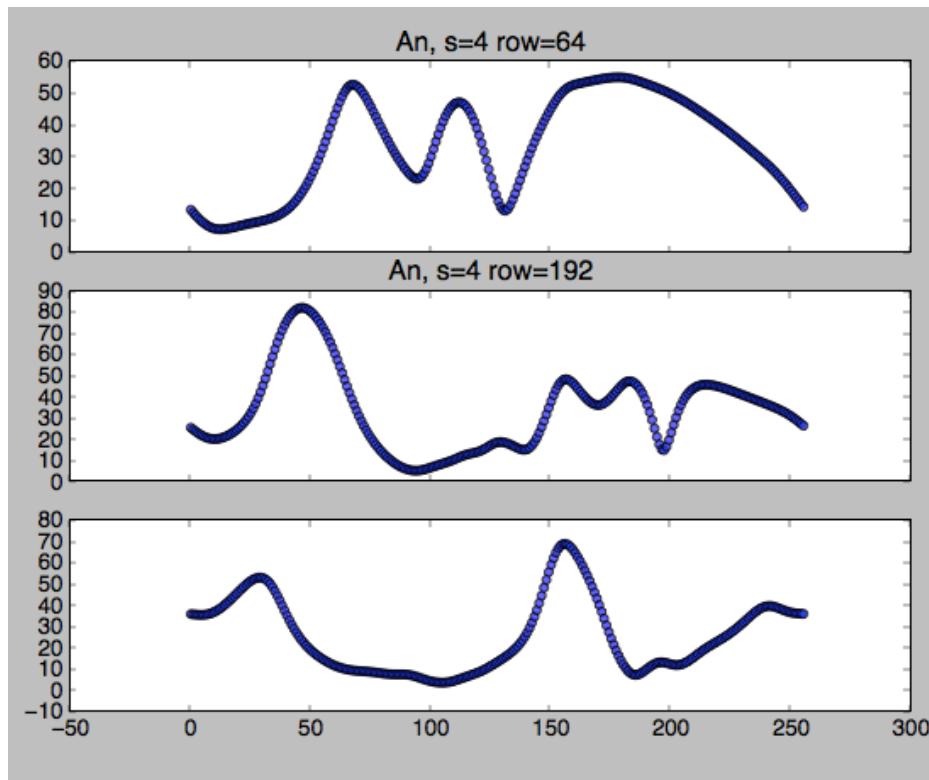




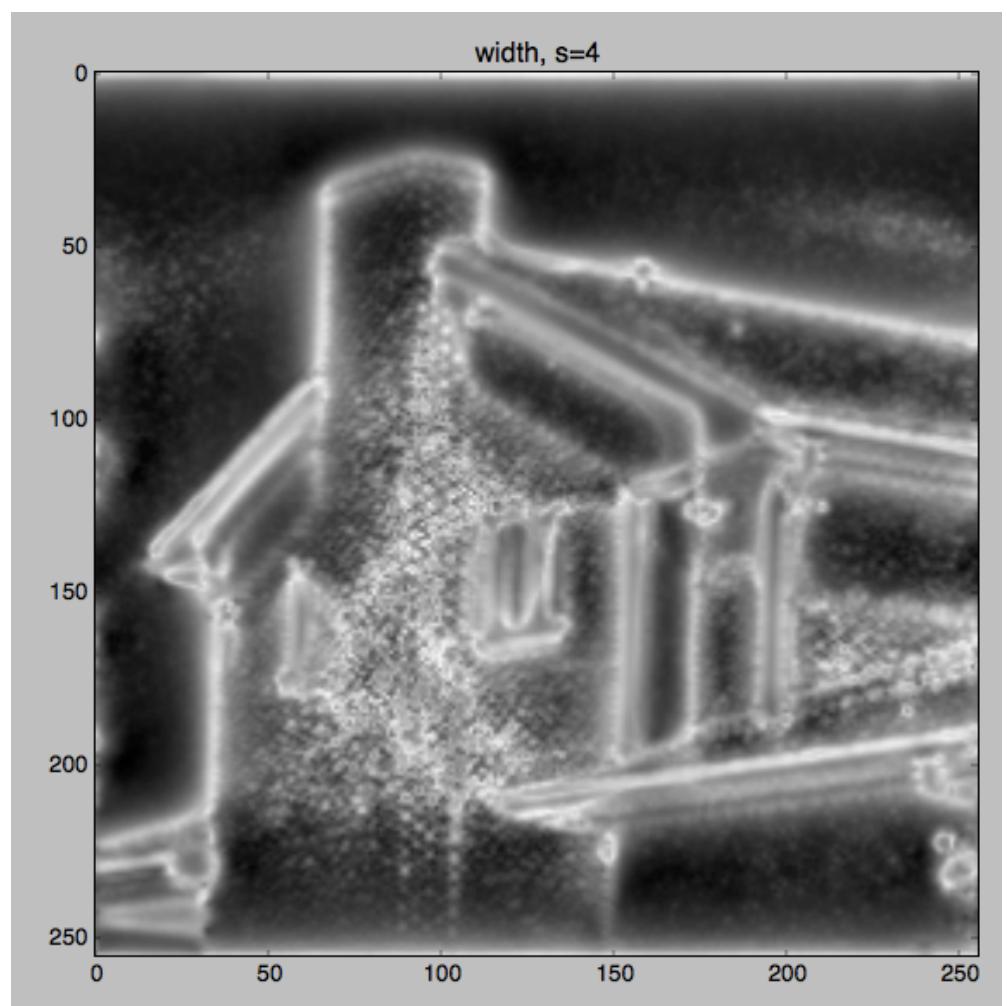
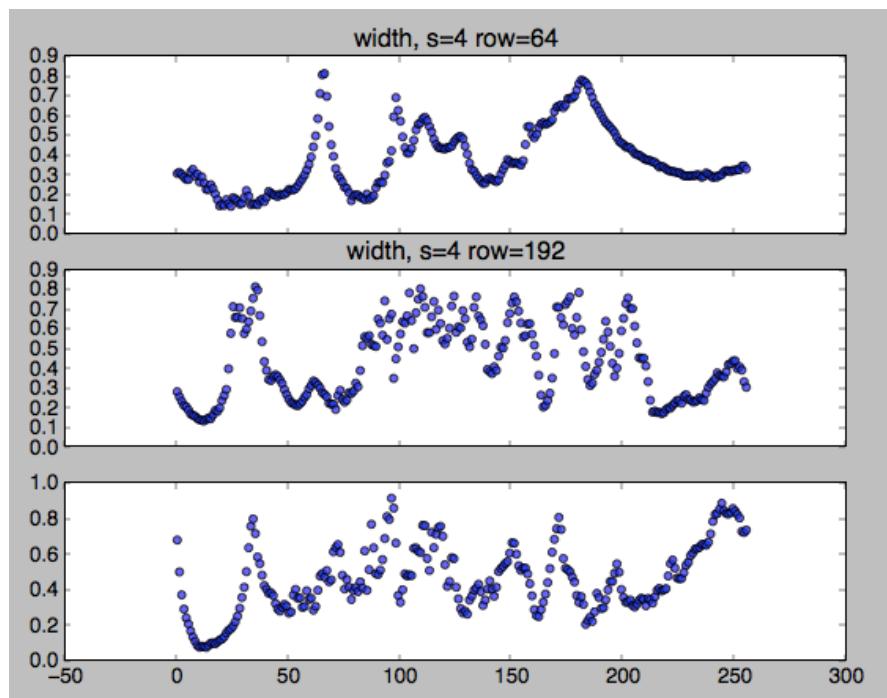
**s=4, h=ifft2(IMF \* H)  
(where IMF=perFFT(img)\*logGabor)**



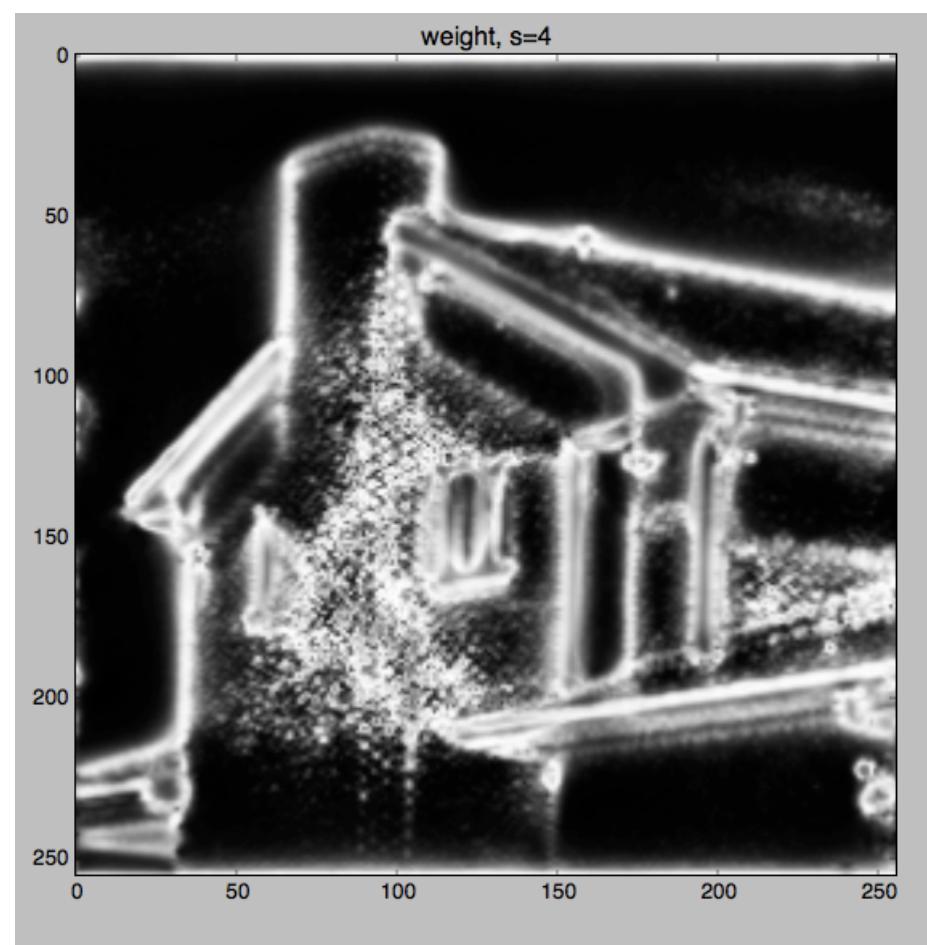
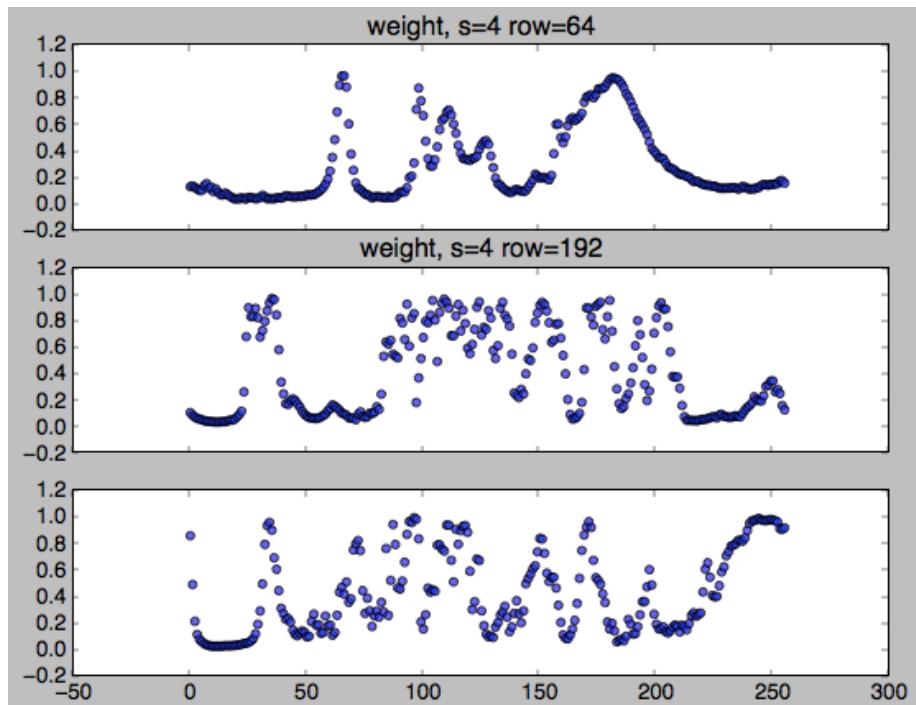
$s=4$ ,  $A_n = \sqrt{f * f + h1 * h1 + h2 * h2}$   
 (where  $f = \text{iFFT}(\text{perFFT(img)} * \text{logGabor})$   
 and  $h = \text{iFFT}(\text{perFFT(img)} * \text{logGabor} * H)$  )



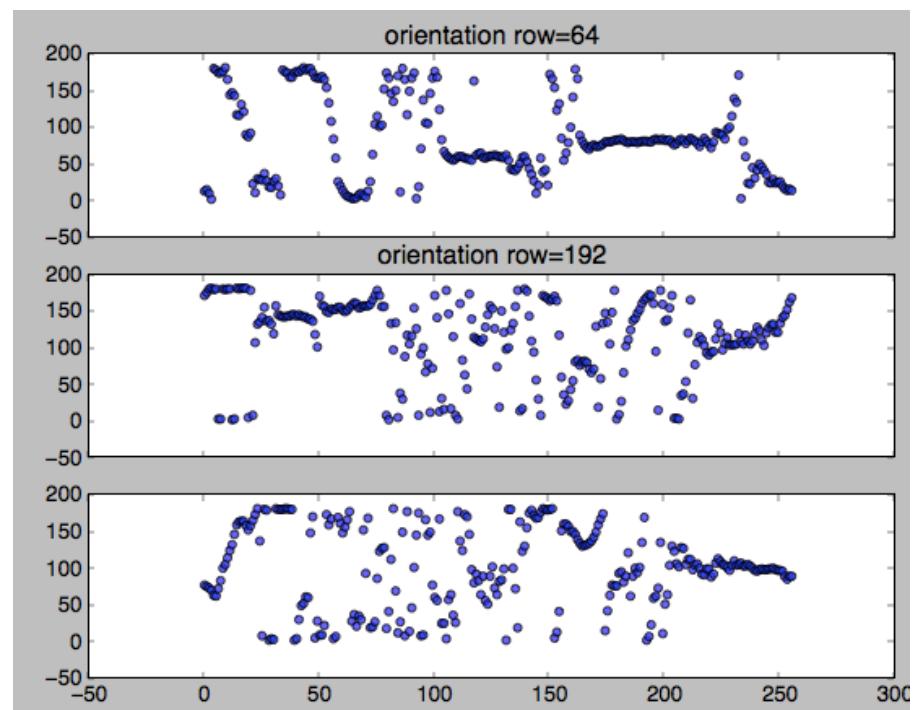
**s=4, width = (sumAn / (maxAn + epsilon) - 1.) / (nscale - 1)**



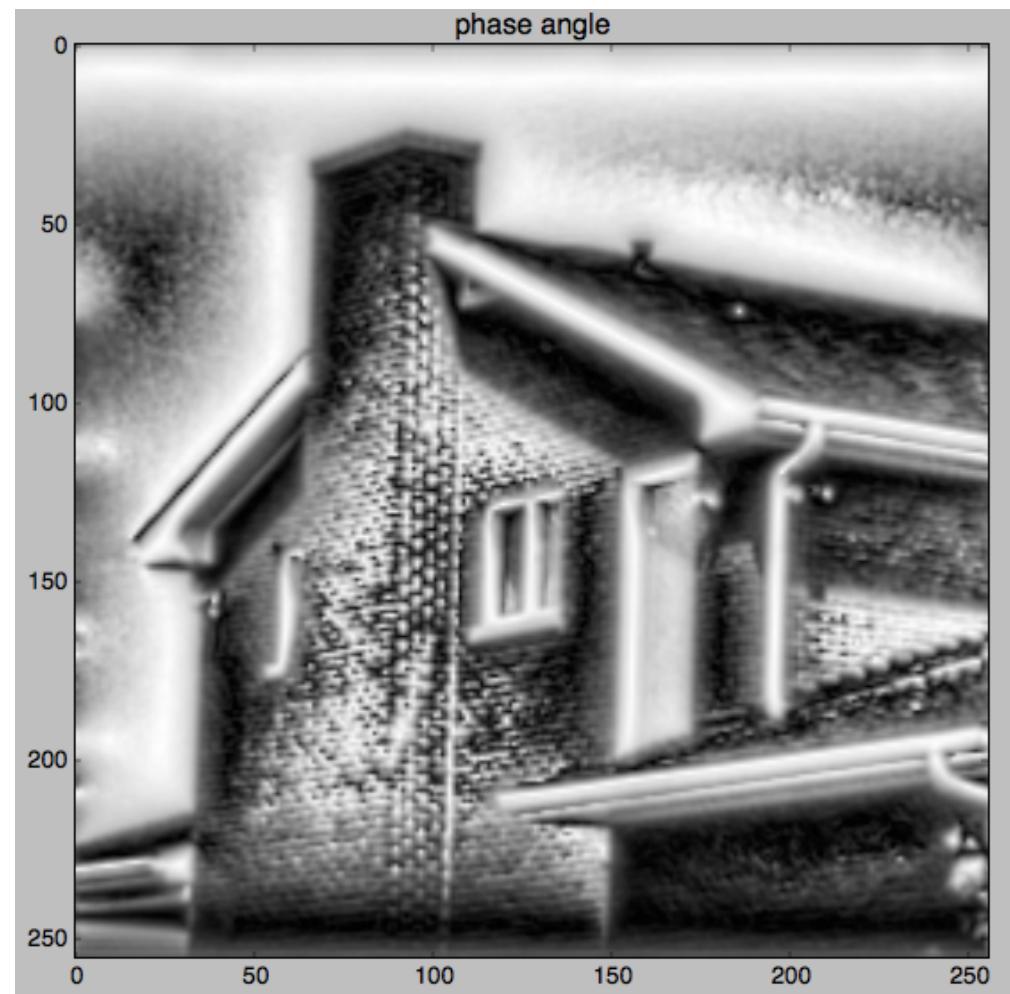
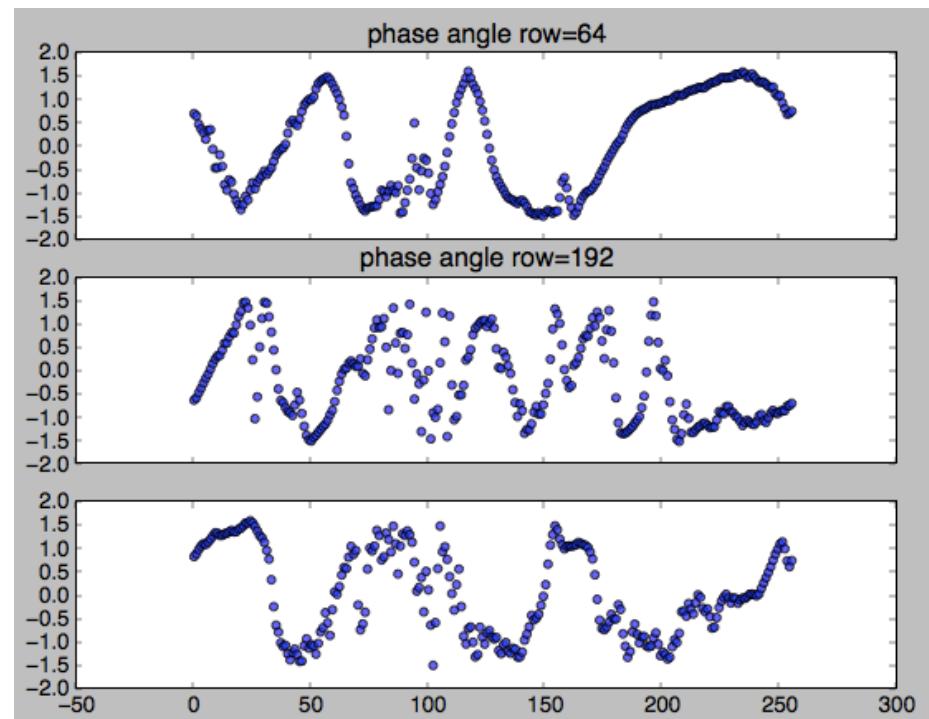
**s=4, weight =  $1. / (1. + \exp(g * (\text{cutoff} - \text{width})))$**   
(where cutoff=0.5, g=10.)



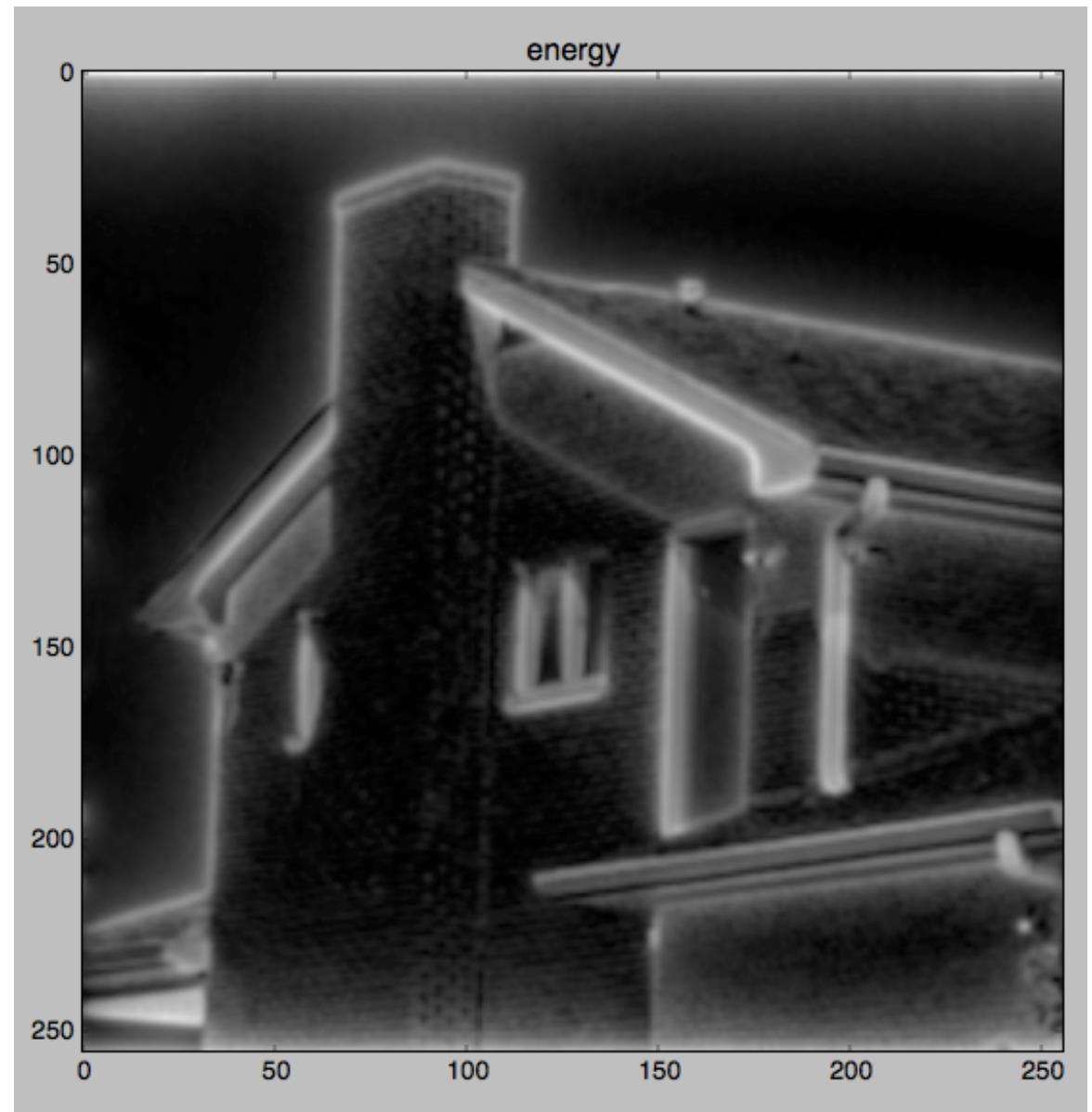
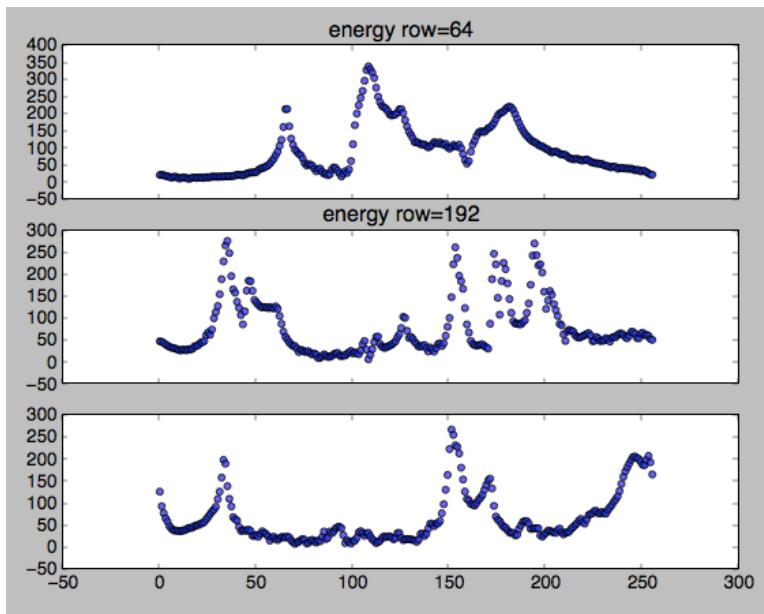
# orientation

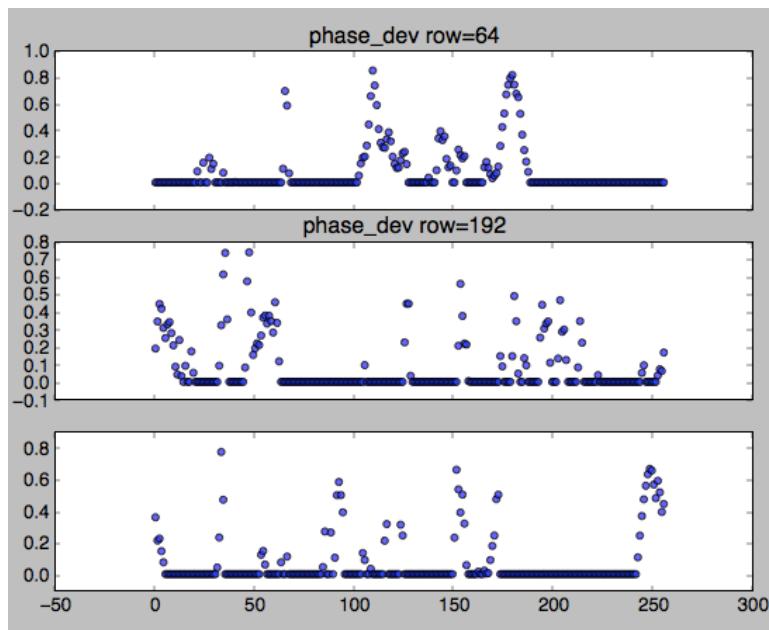


# phase angle

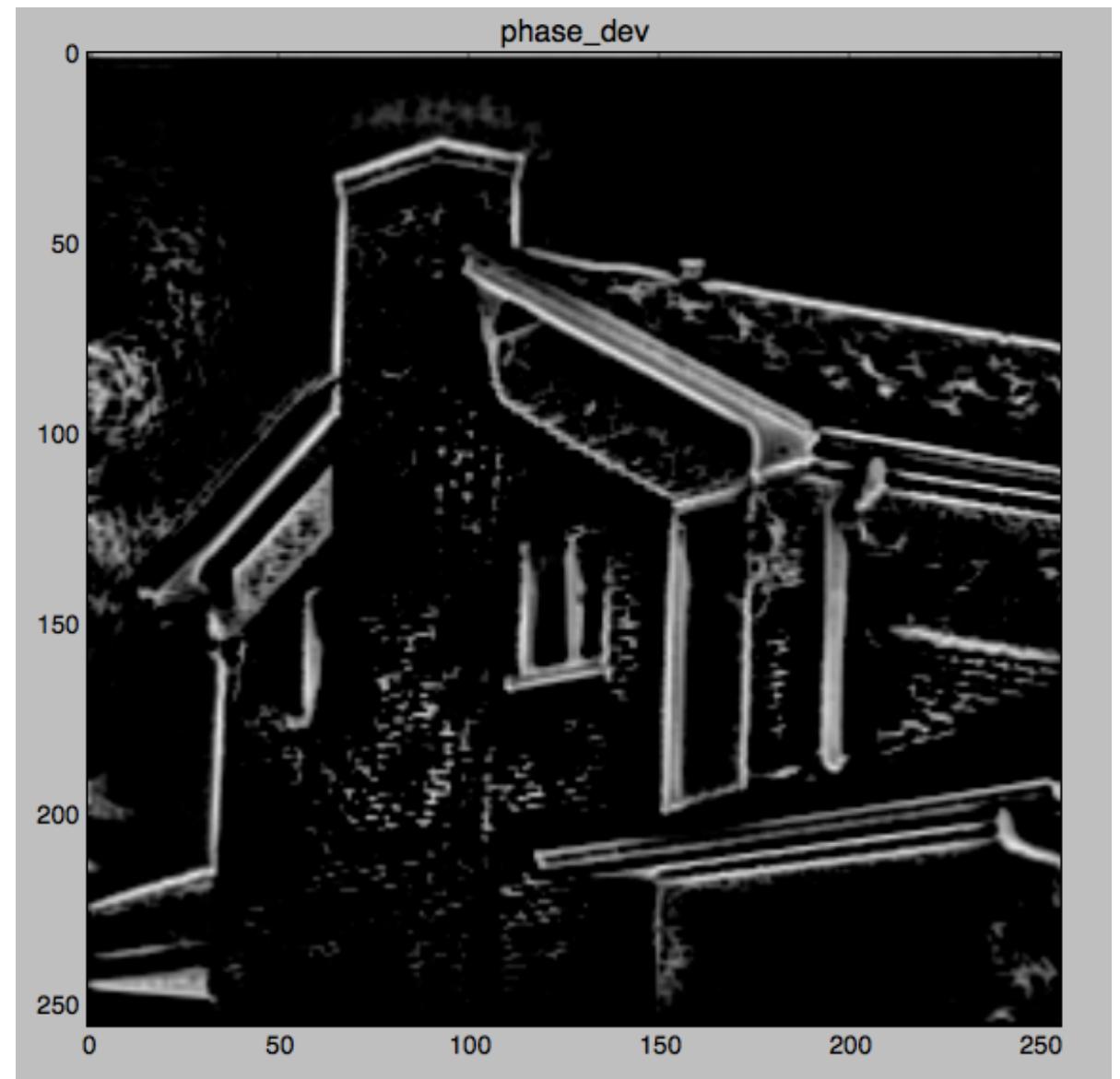


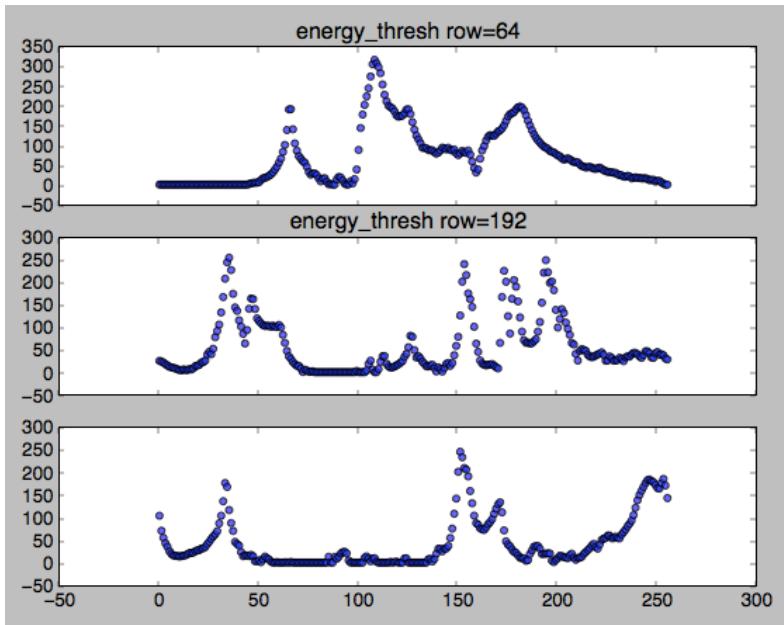
## energy



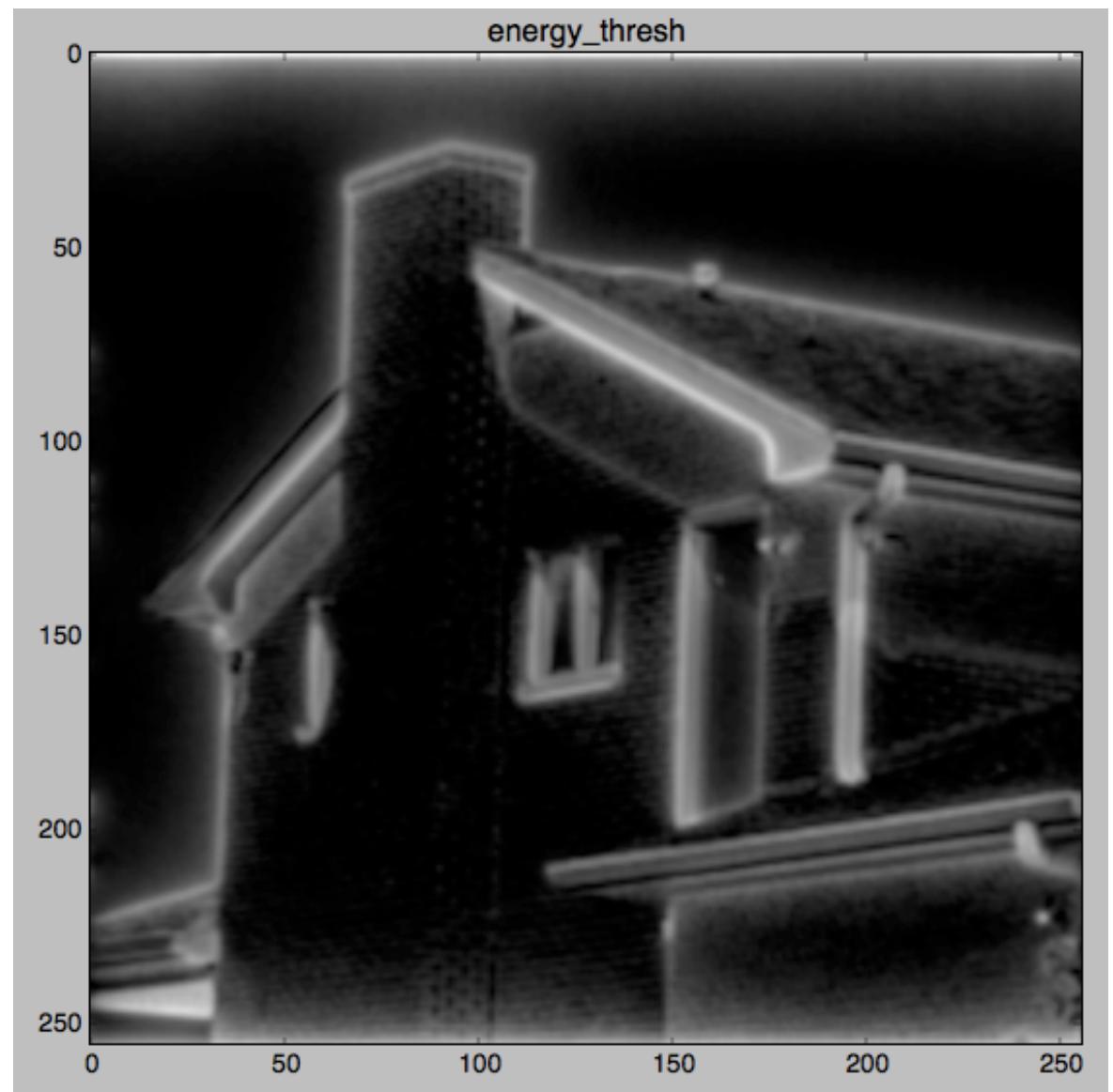


phase\_dev

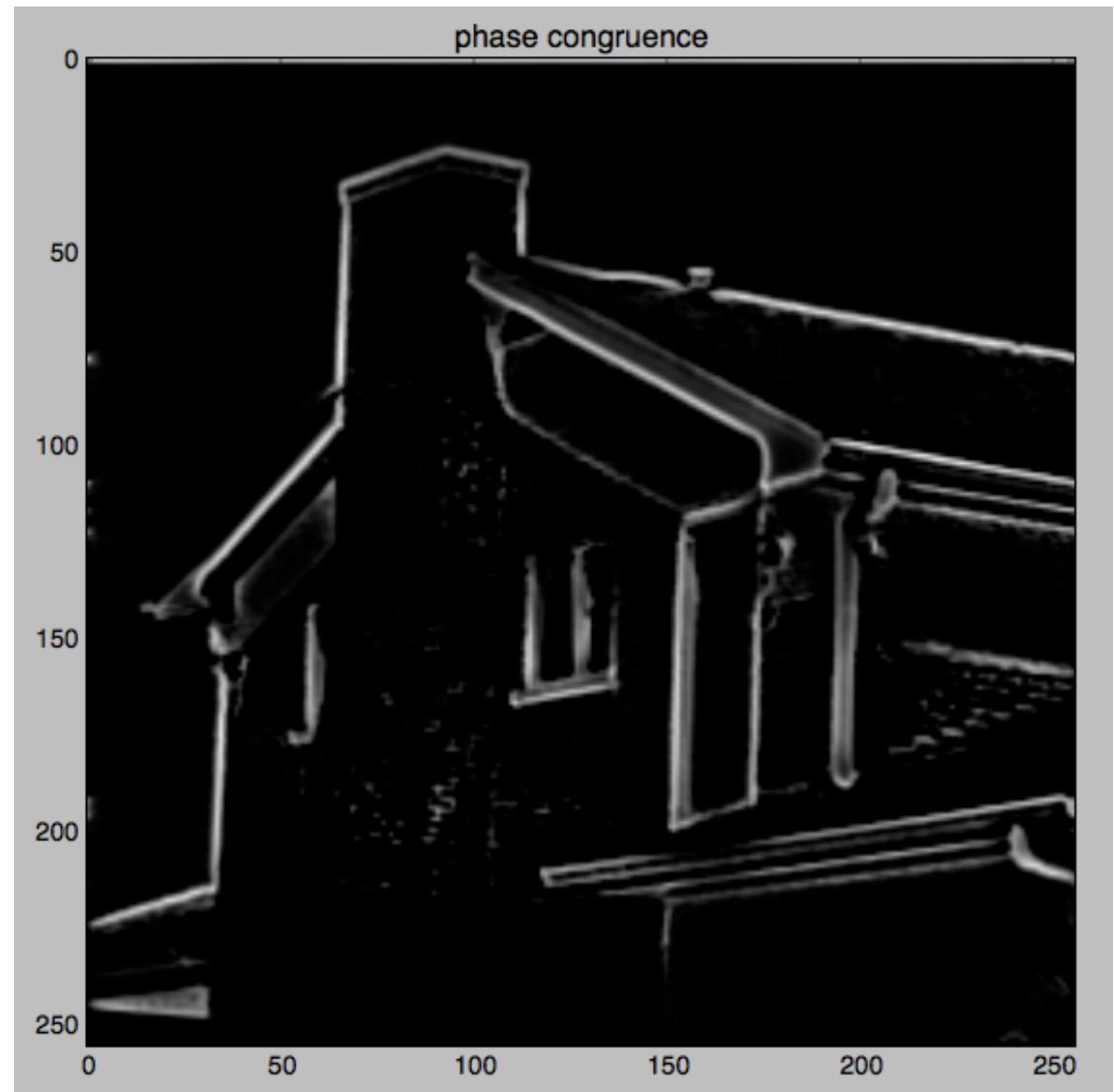
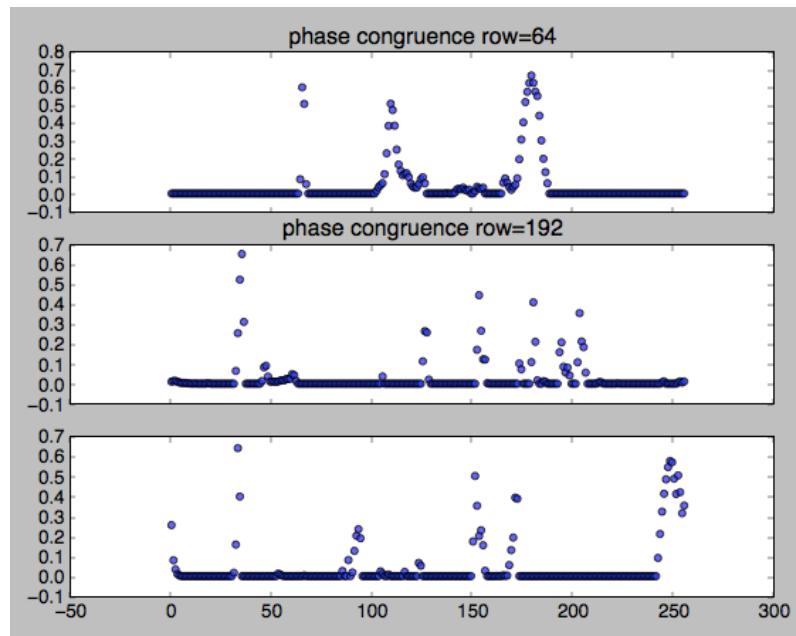




energy\_thresh



phase congruence =  
maximum(  
    1. - deviationGain \* arccos(energy / (sumAn + epsilon)),  
    0)

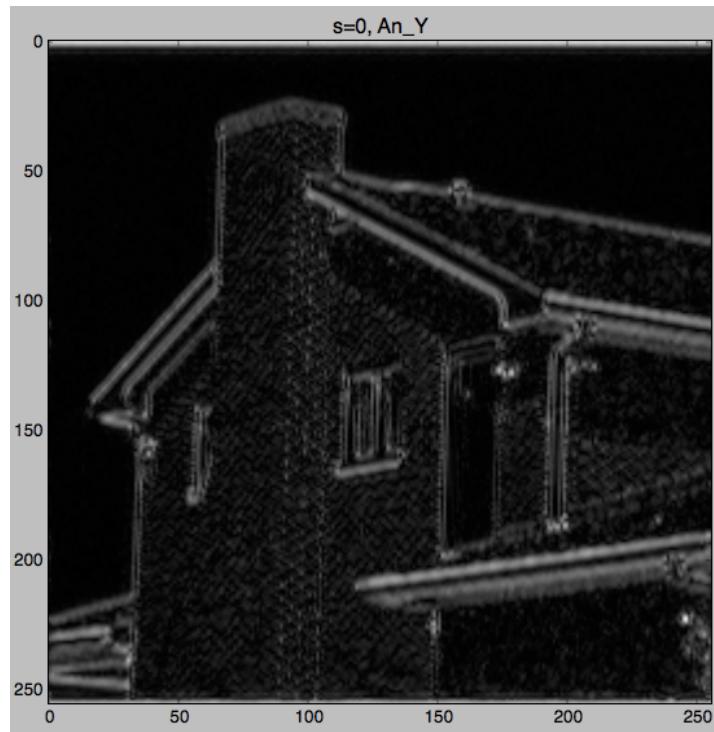
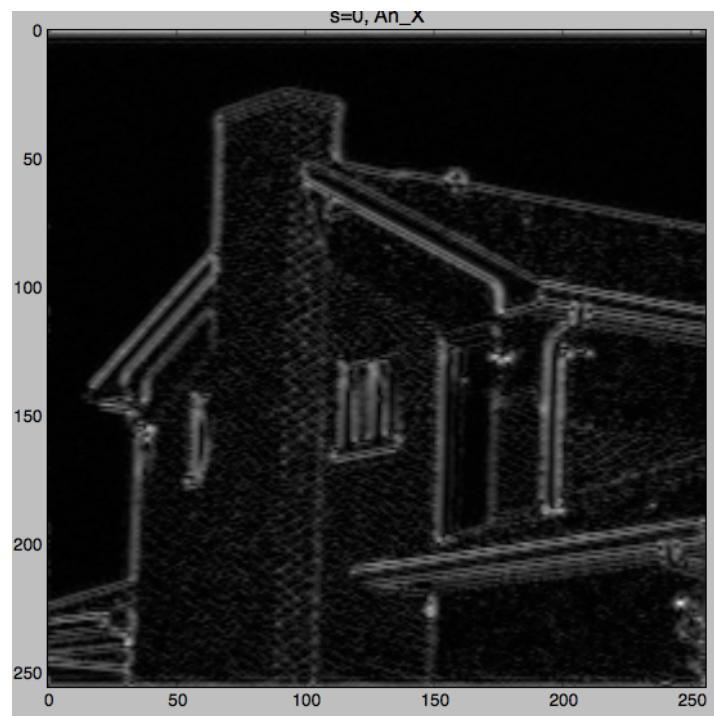
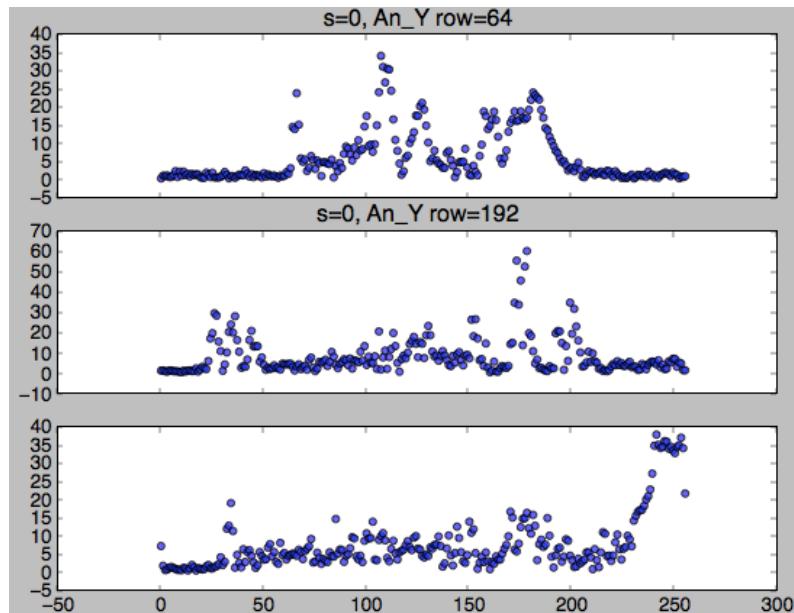
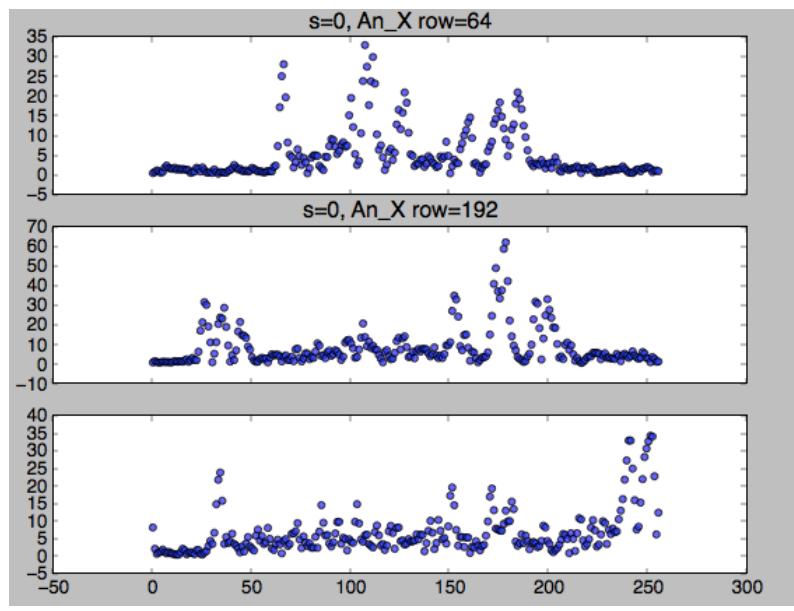


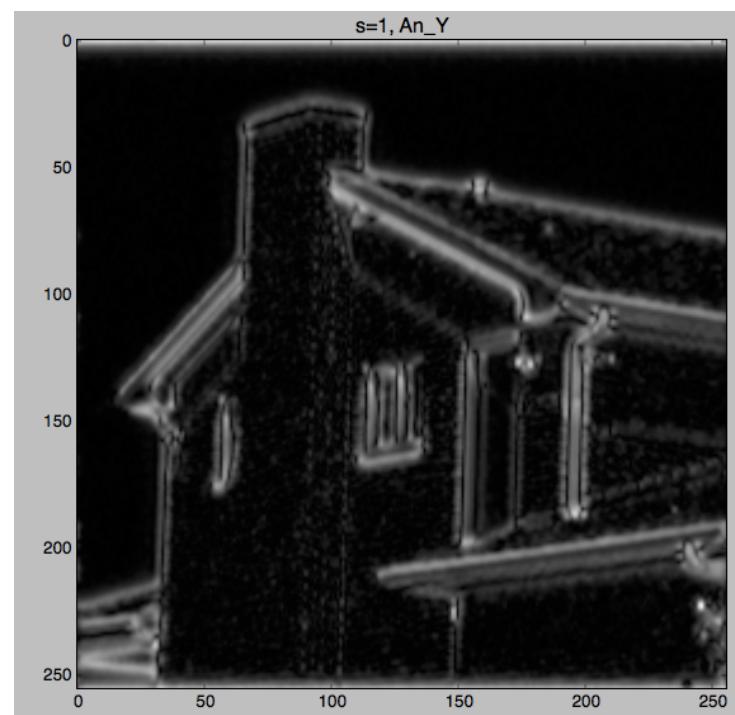
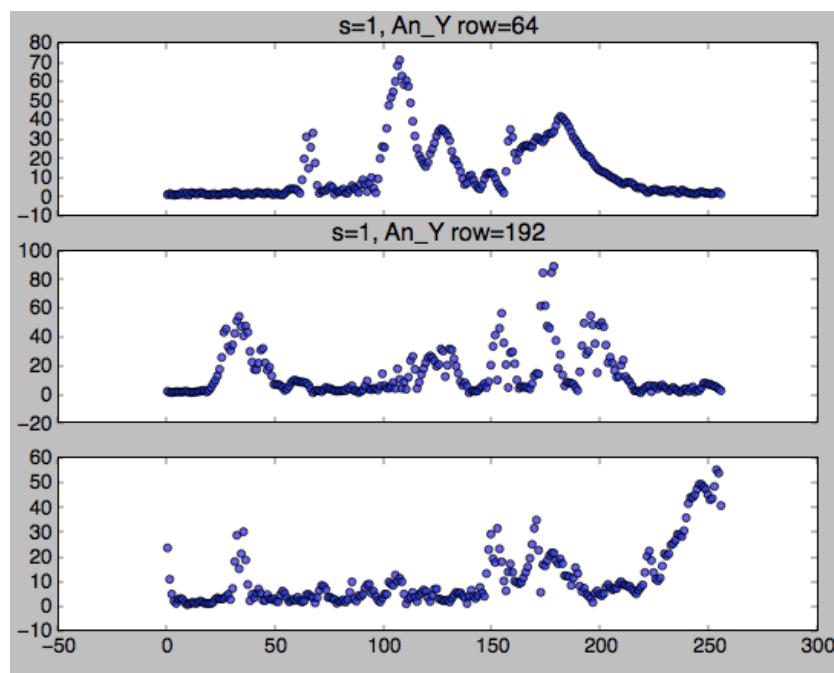
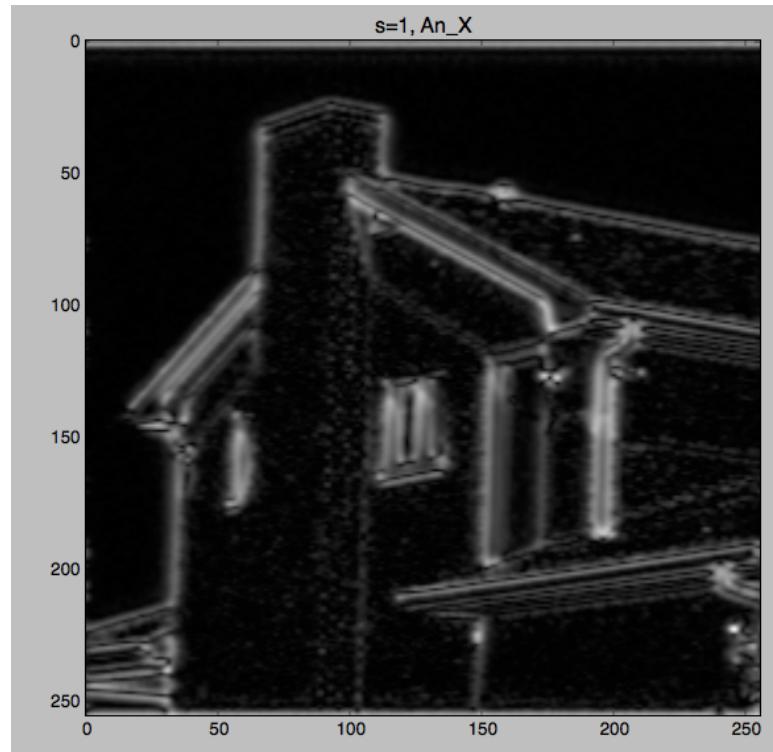
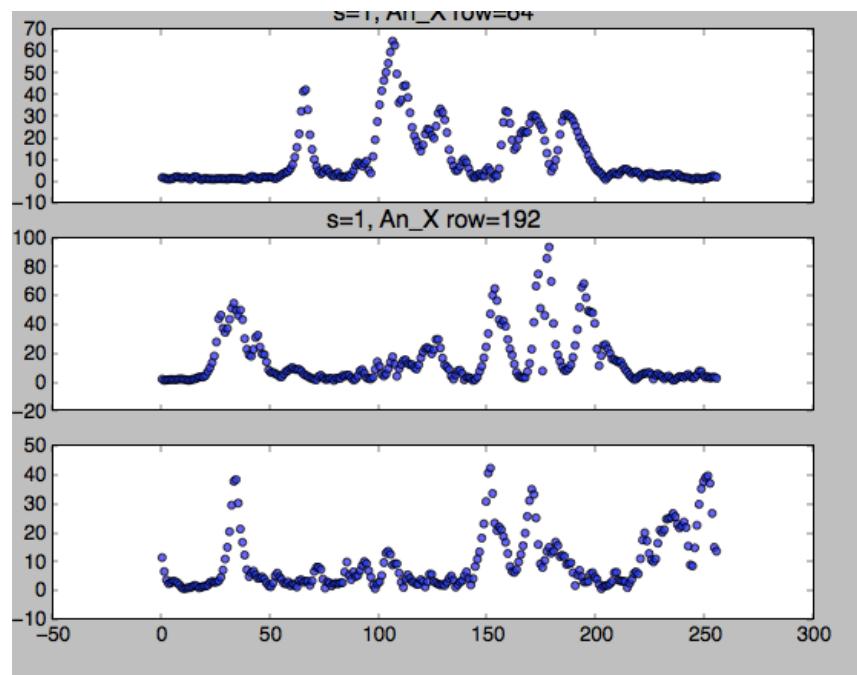
Looking at the x and y components of An

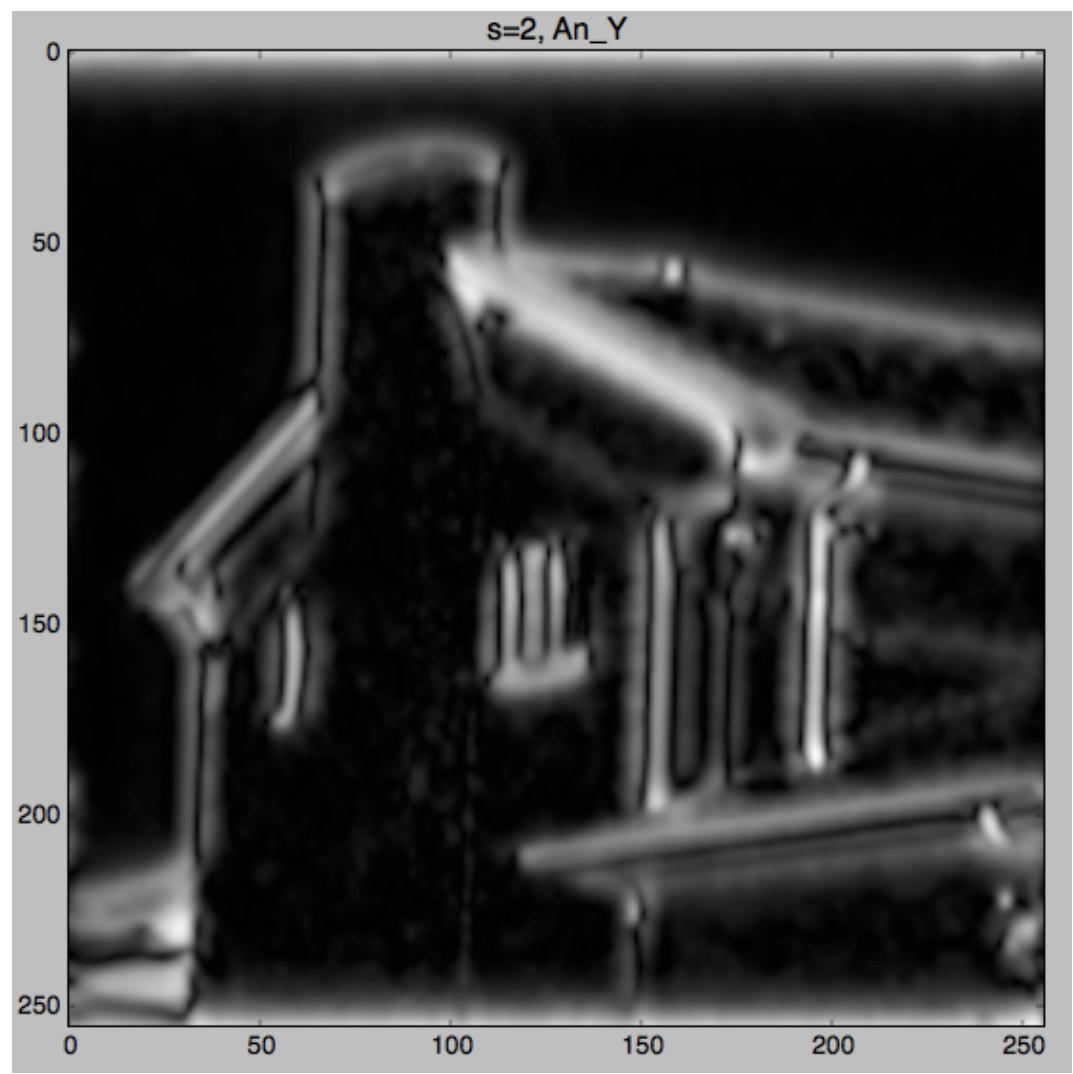
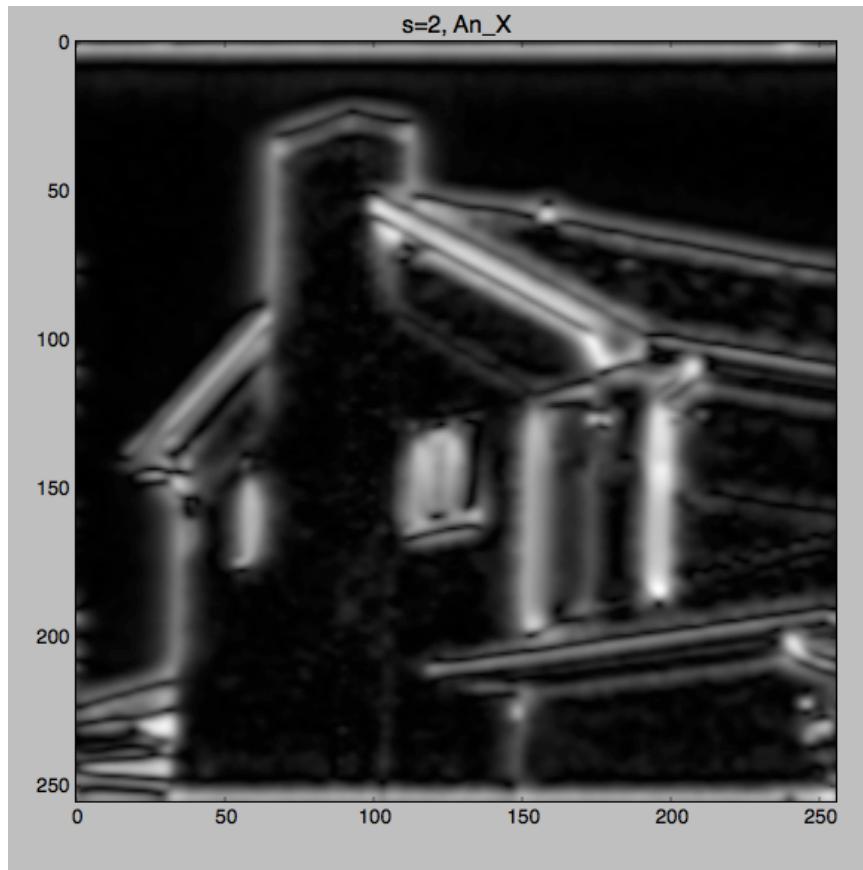
And afterwards, attempting to make a 2nd derive filter.

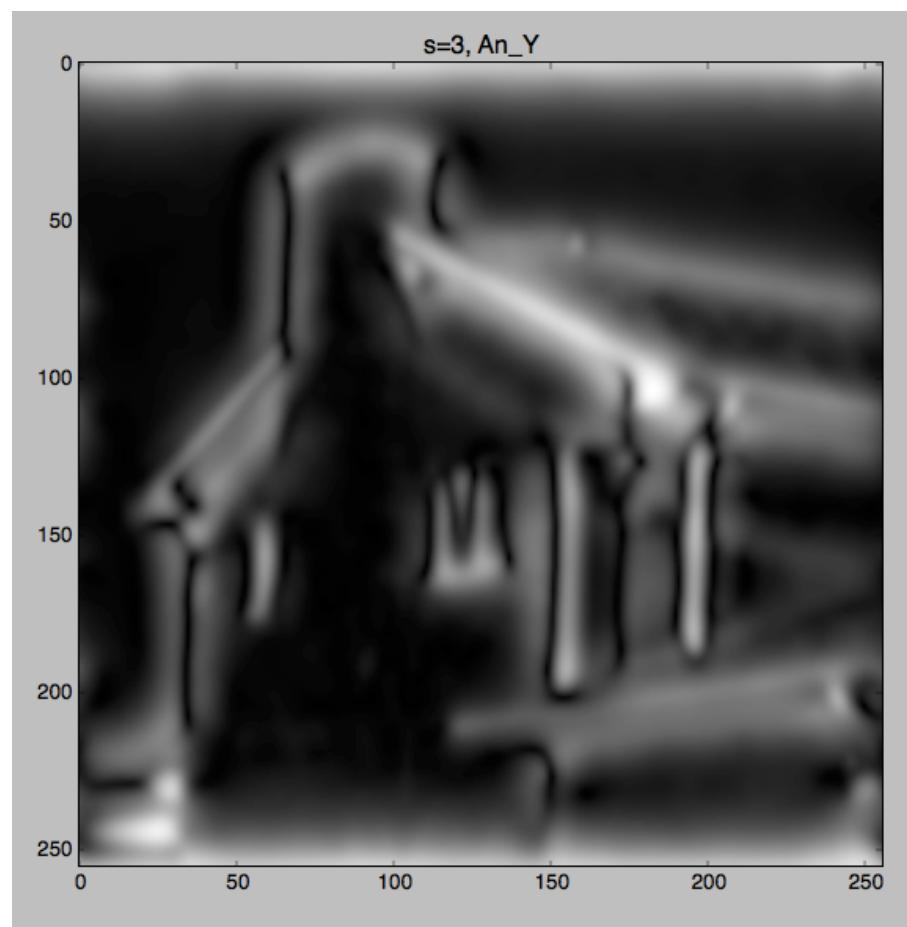
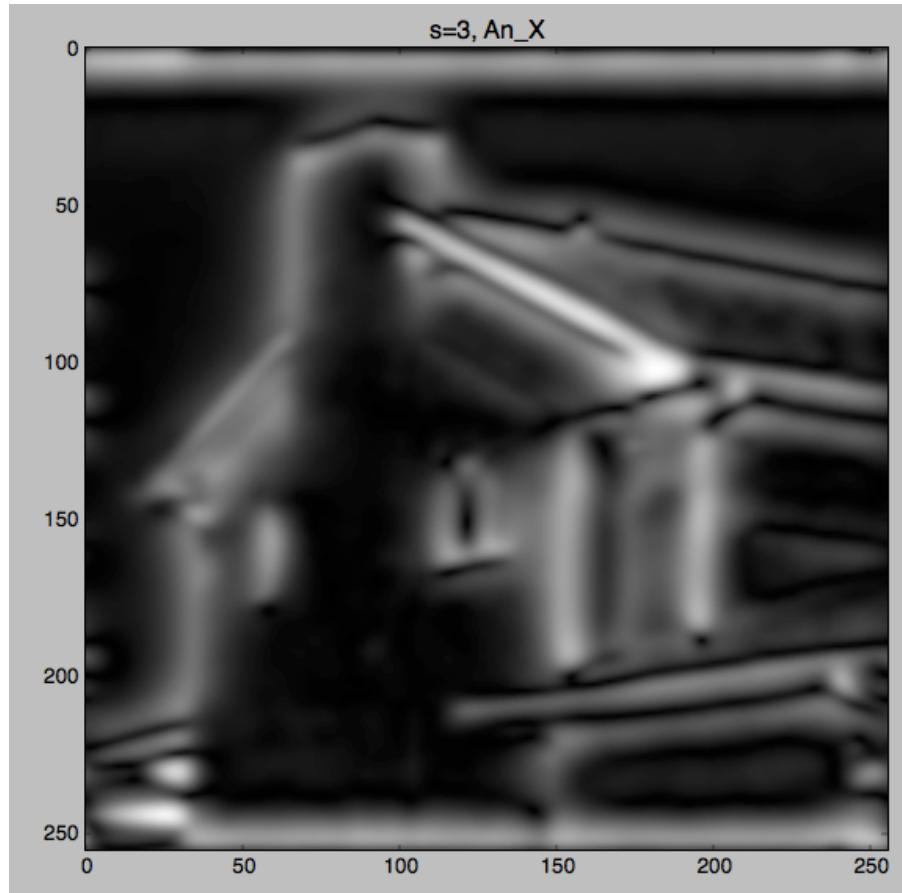
corner curvature uses the first and second derivatives in x and y

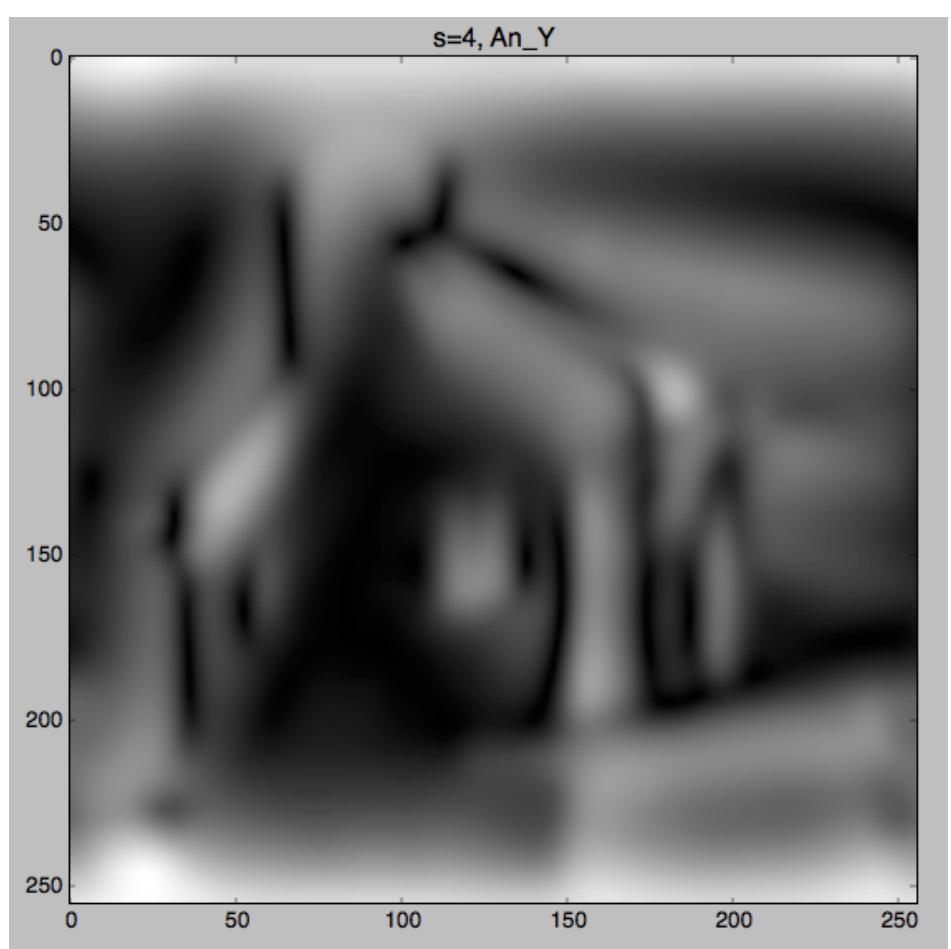
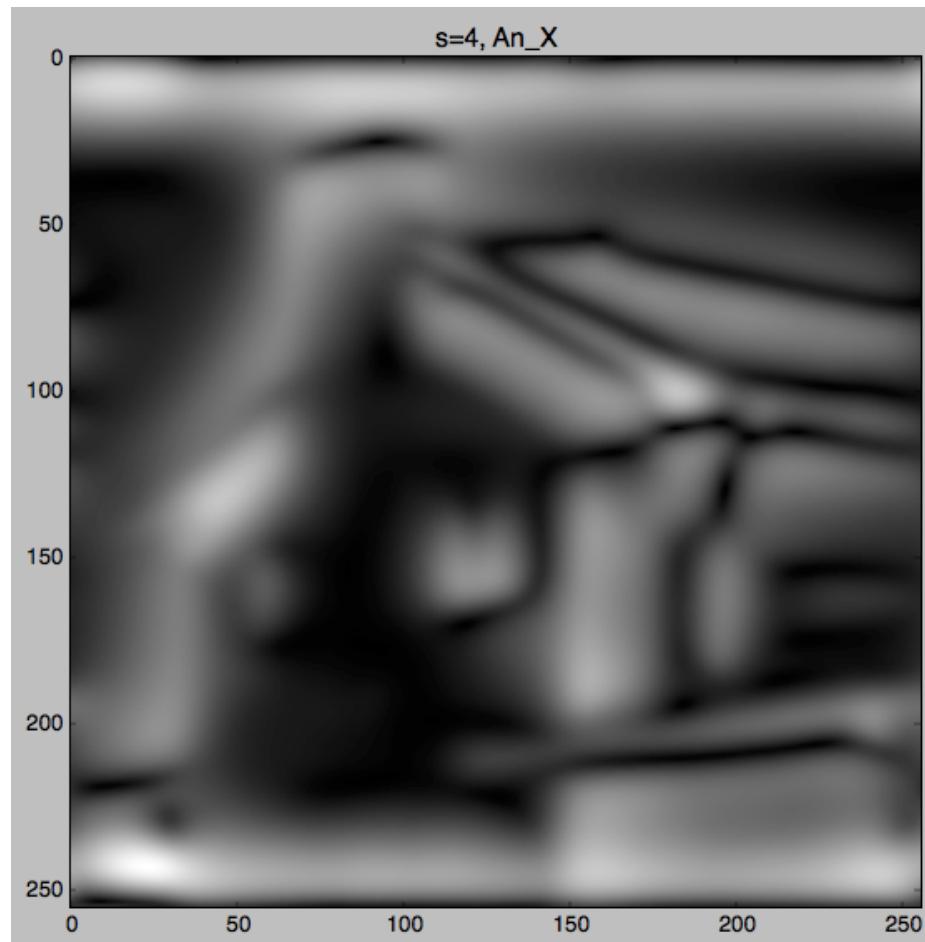
$$k(t,o\sim) = \frac{X_{dot}(t,o\sim) * Y_{dot\_dot}(t,o\sim) - Y_{dot}(t,o\sim) * X_{dot\_dot}(t,o\sim)}{(X_{dot}^2(t,o\sim) + Y_{dot}^2(t,o\sim))^{1.5}}$$



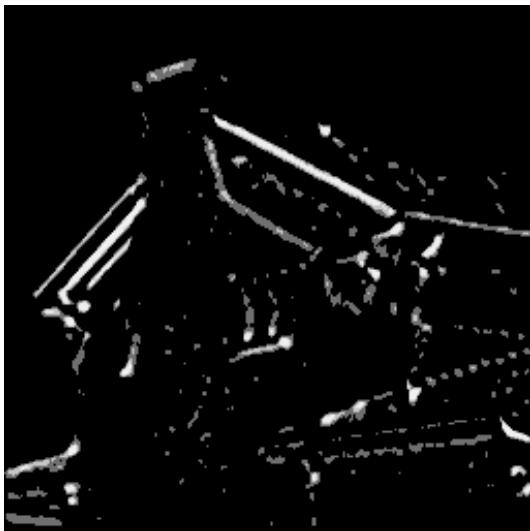




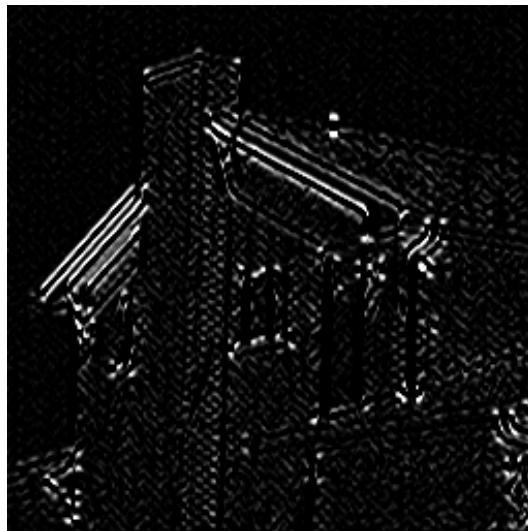




As contrast to the phase congruence result, can look at the test image blurred by  $\sigma = \sqrt{2}$  then a binomial kernel for the first derivative applied to one blurred image copy and the binomial kernel for the second derivative applied to another blurred image copy.



stretched 1st deriv



noisy 2nd deriv

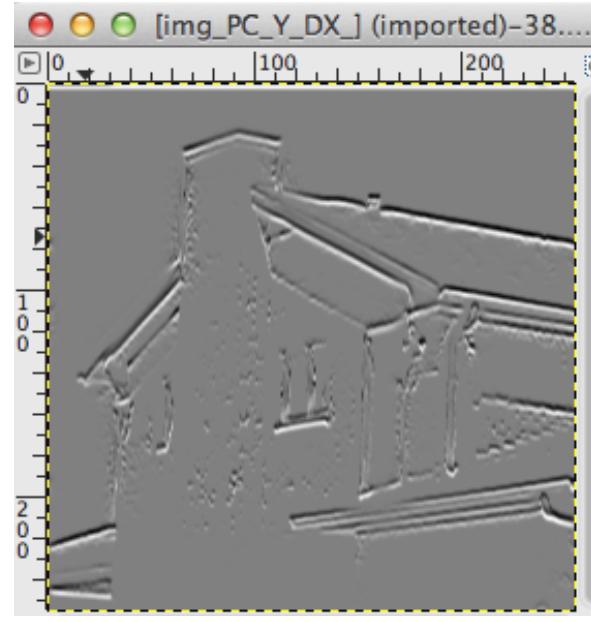
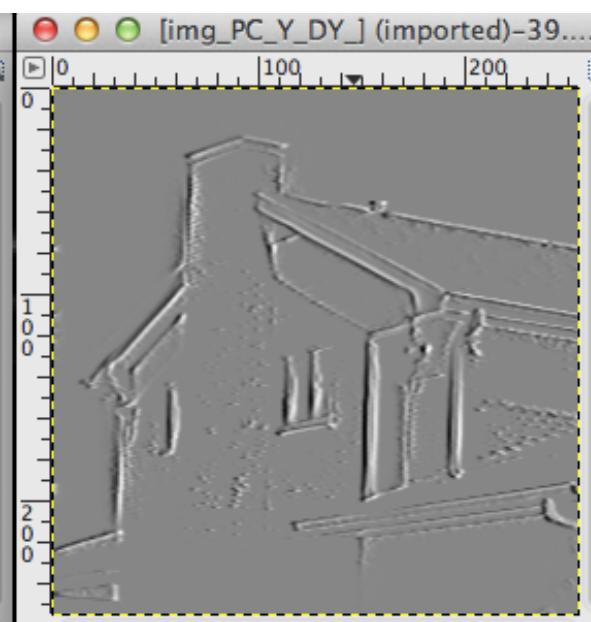
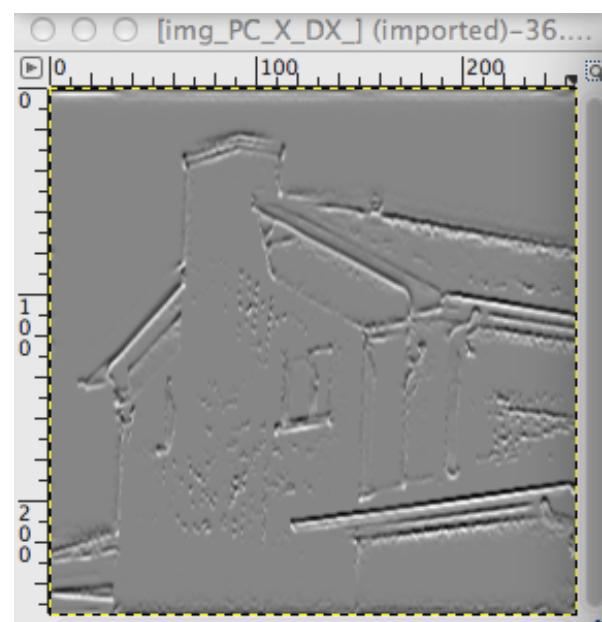
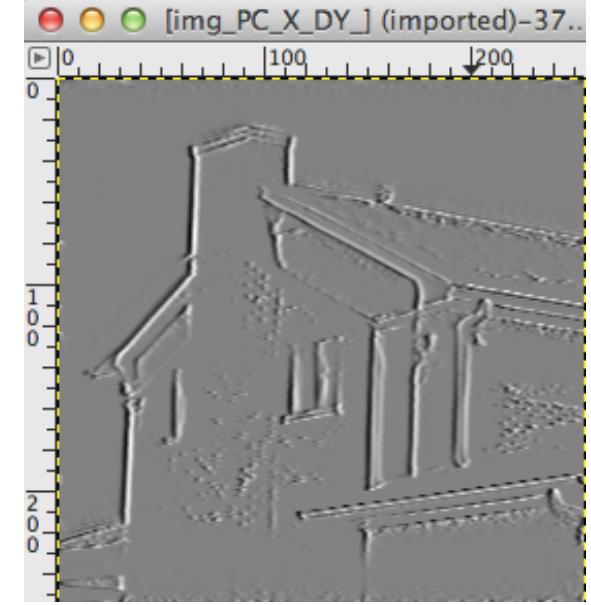
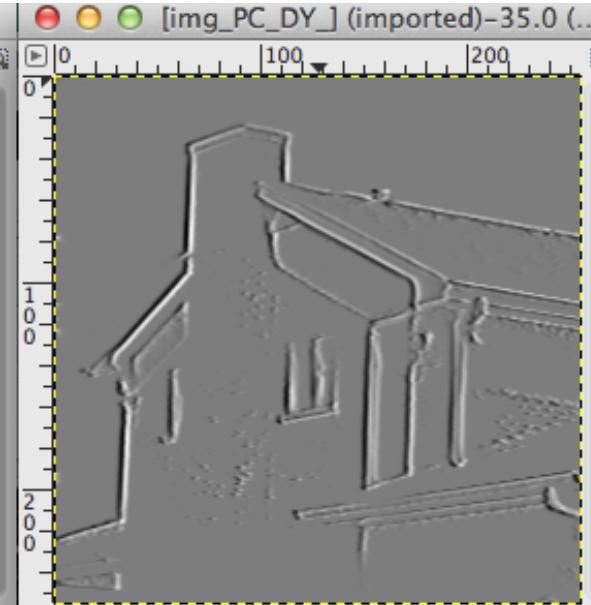
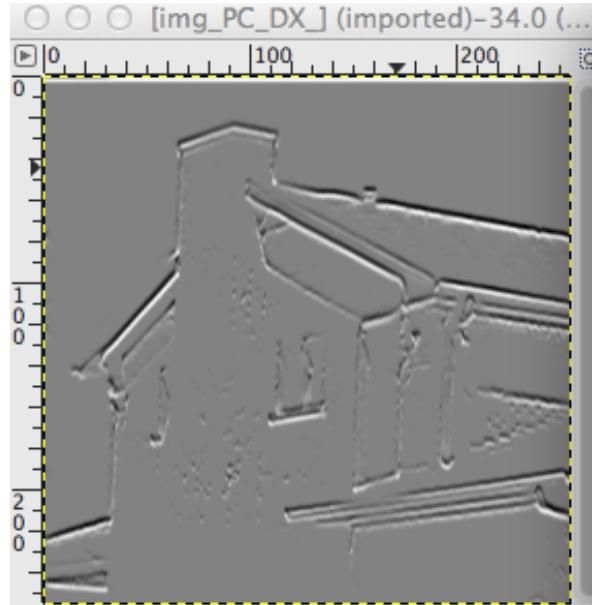


unfiltered curvature  
for one scale

Back to phase congruence: can use the thinned edges and calculate corners for those edges in many ways.

– one way is to calculate CSS corners using min and max of local curvature from the finest scale images tracked to the coarser to refine the location.

Phase congruence 2nd derivative images calculated by first derivative gaussian of sigma=1 to the PC (phase congruence) image, the PC\_x component image, and the PC\_y component image.



phase congruency: merton test, nScale=4, k=10, tLow=0.05, tHigh=0.3



pc img, all set to 255



pc img, thinned by NMS



phase angle steps

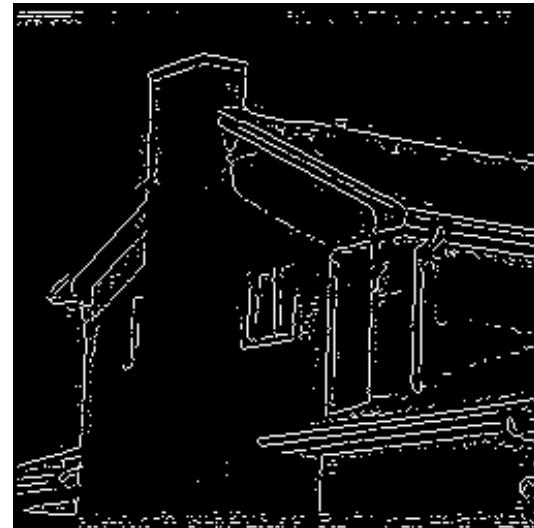


phase angle steps + lines

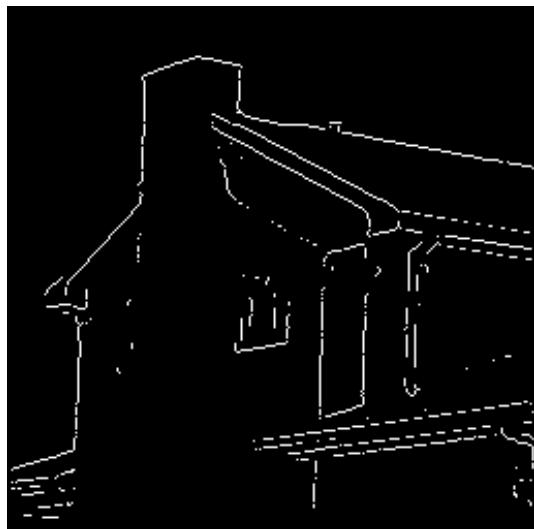
phase congruency: house test, nScale=4, k=10, tLow=0.05, tHigh=0.3



pc img, all set to 255



pc img, thinned by NMS

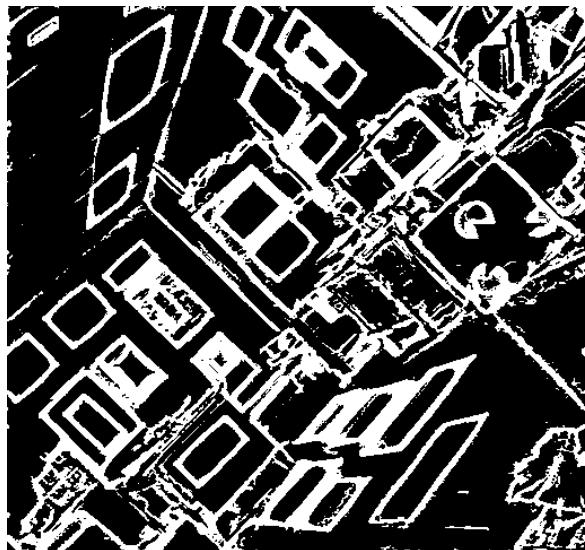


phase angle steps

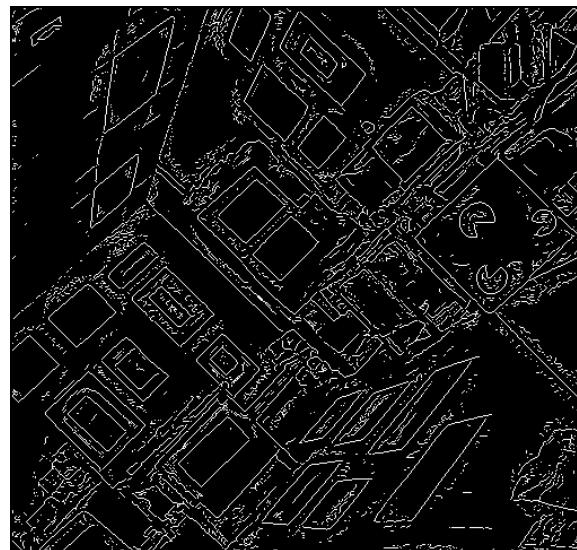


phase angle steps + lines

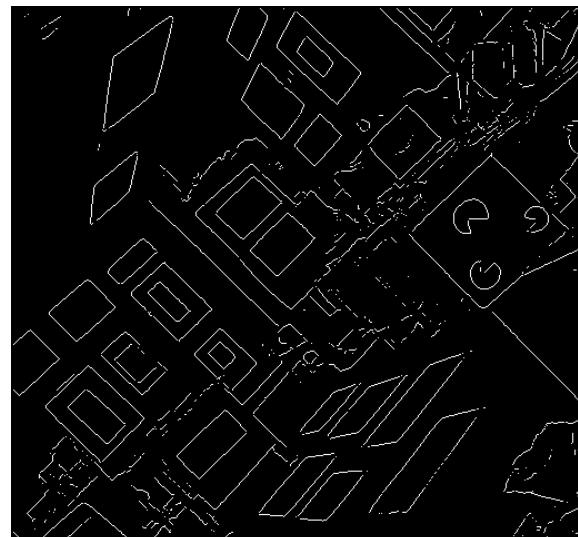
phase congruency: lab test, nScale=4, k=10, tLow=0.05, tHigh=0.3



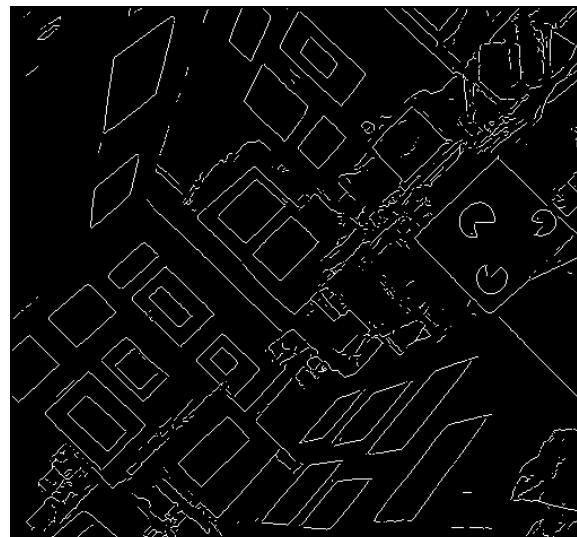
pc img, all set to 255



pc img, thinned by NMS



phase angle steps



phase angle steps + lines