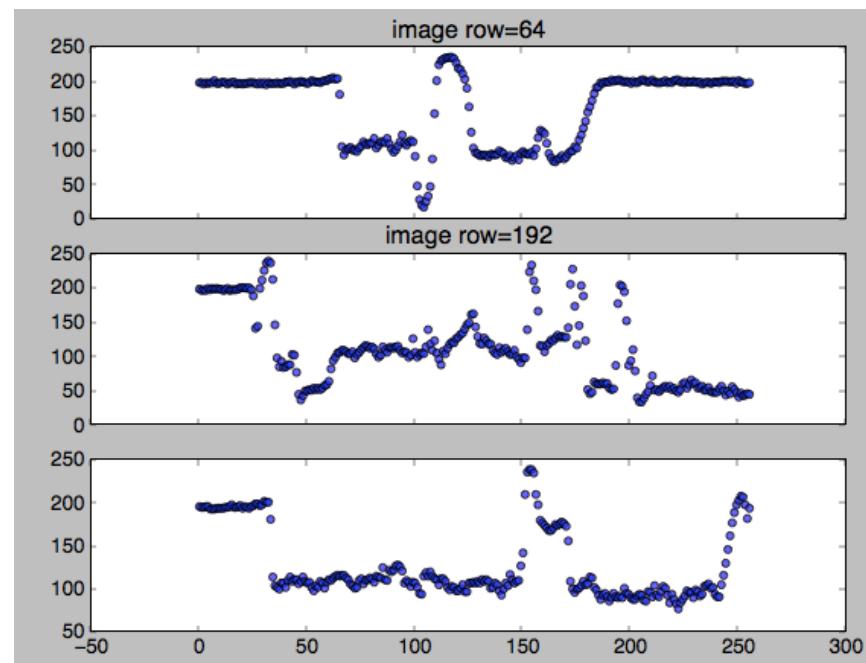
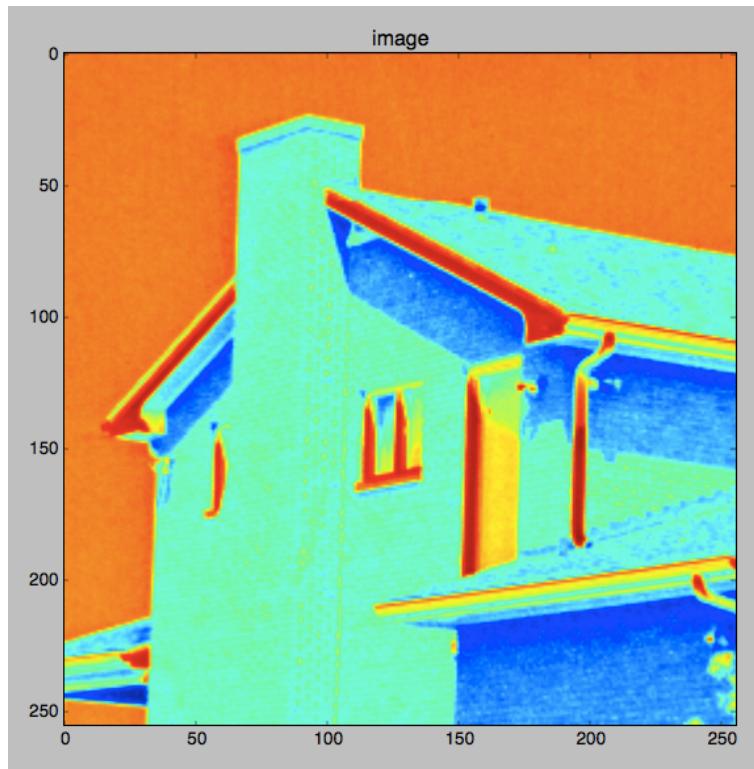


plots from the phase congruency algorithm

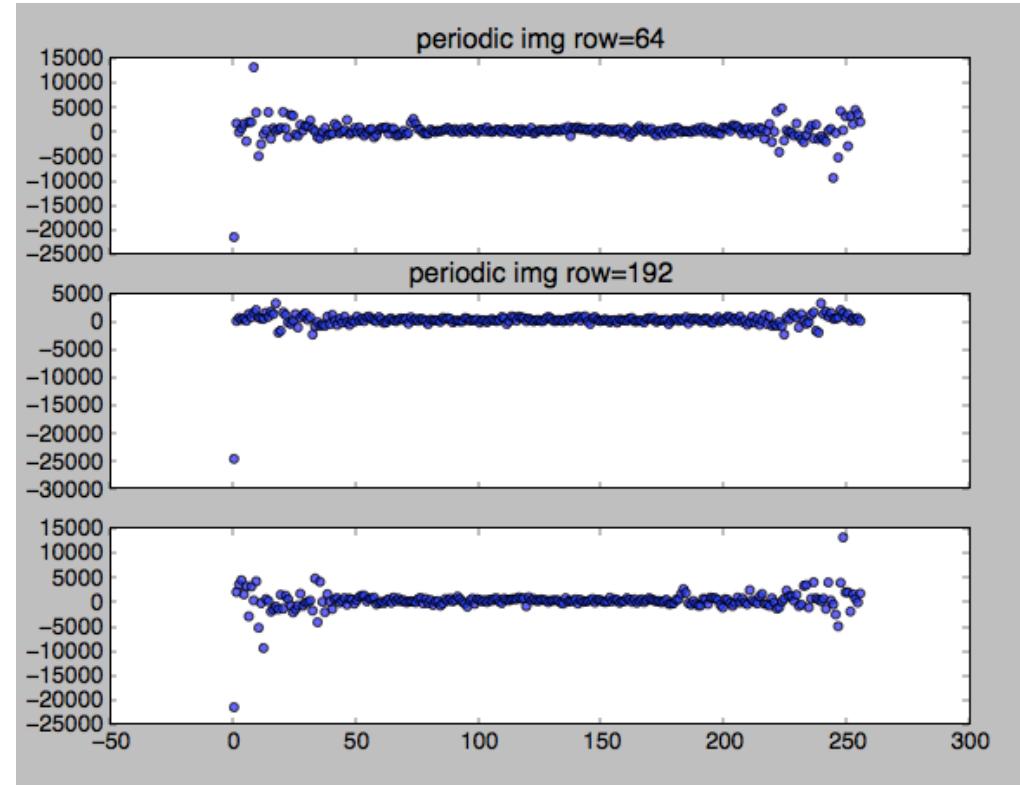
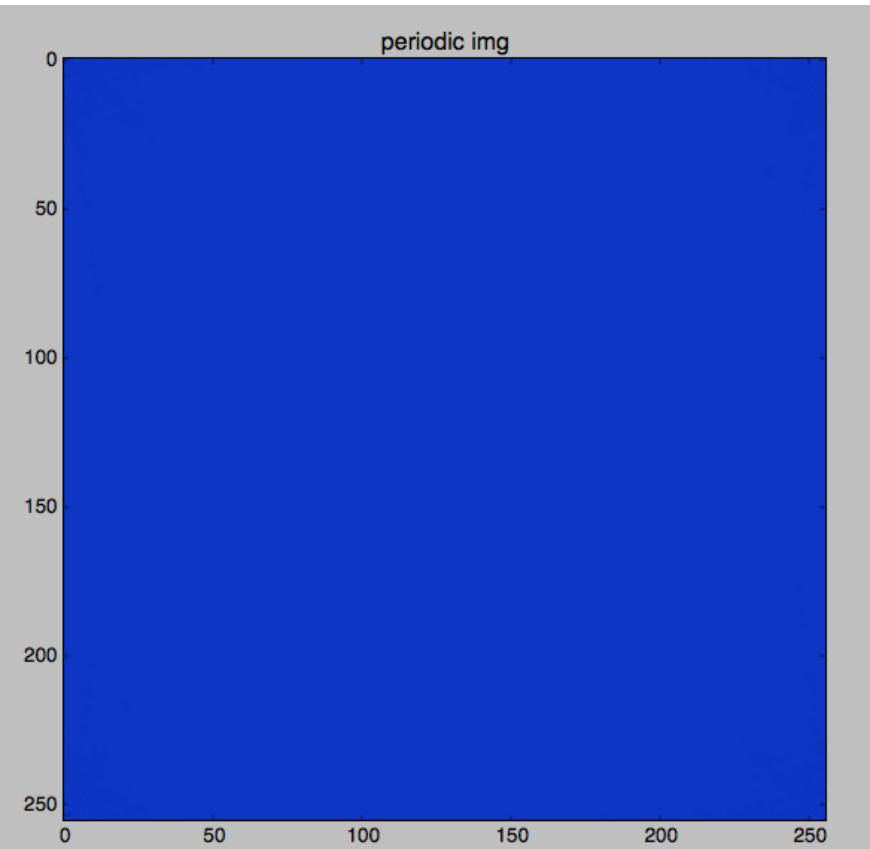
followed by a filter in progress, meant for use in calculating corners
using curvature (needs 1st and 2nd derives)

house.gif

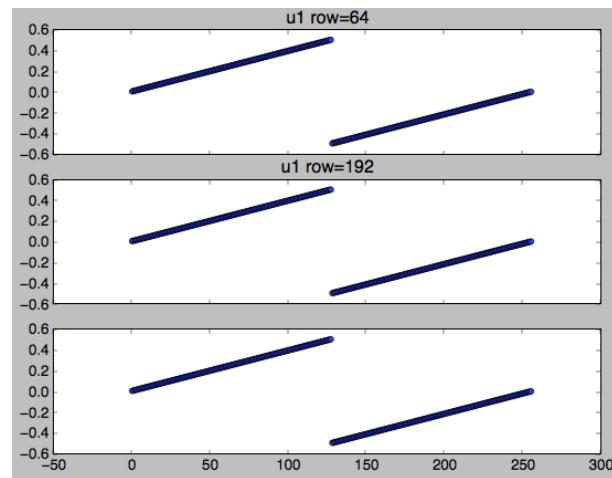


periodic FFT of img

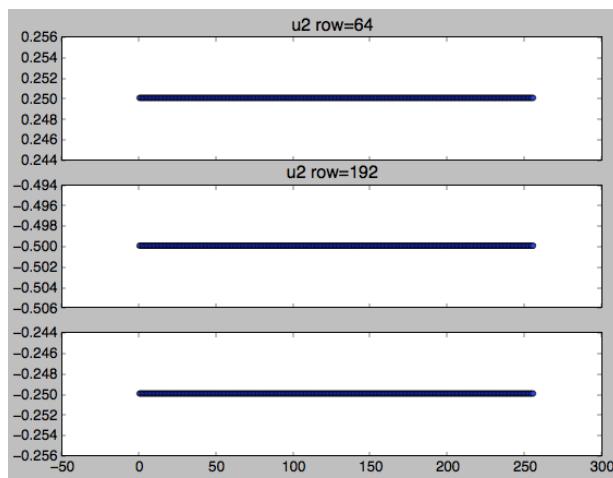
_ , **IM** = **perfft2(img)**



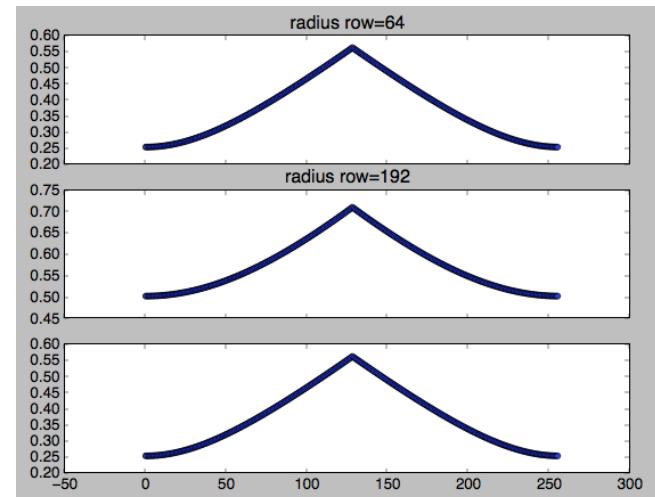
u1



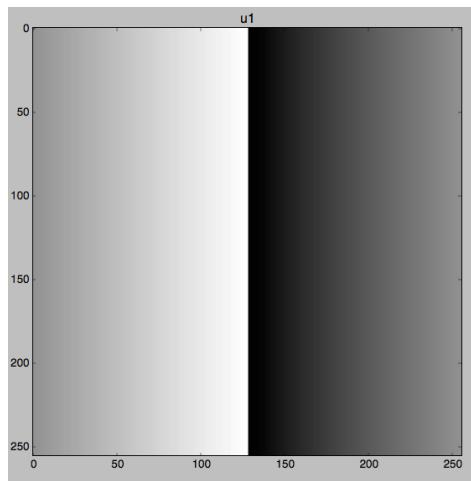
u2



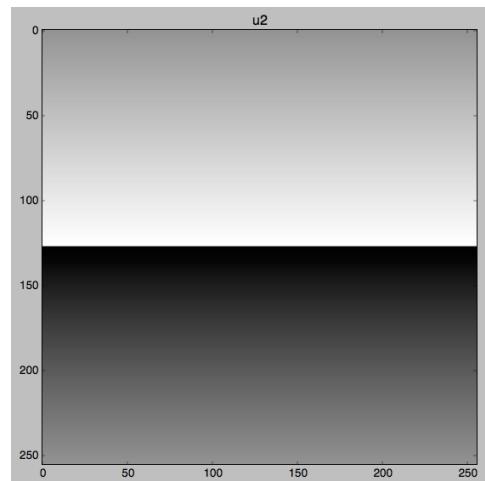
radius



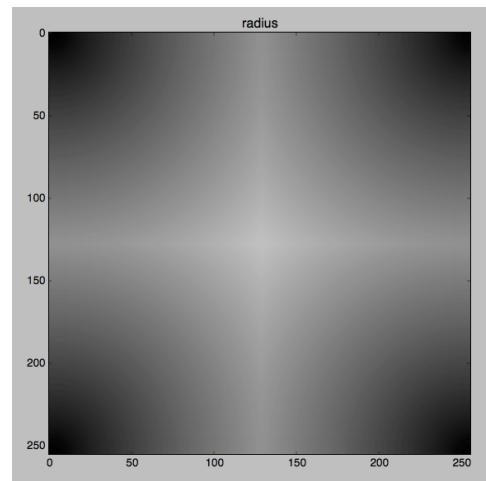
u1



u2

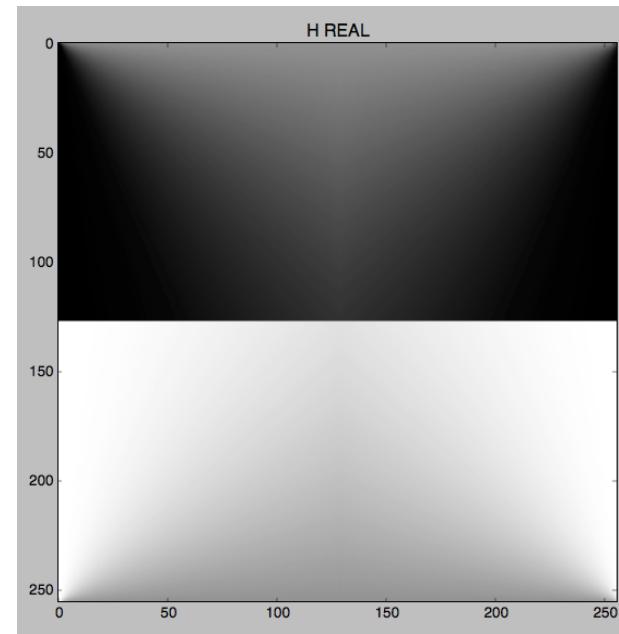
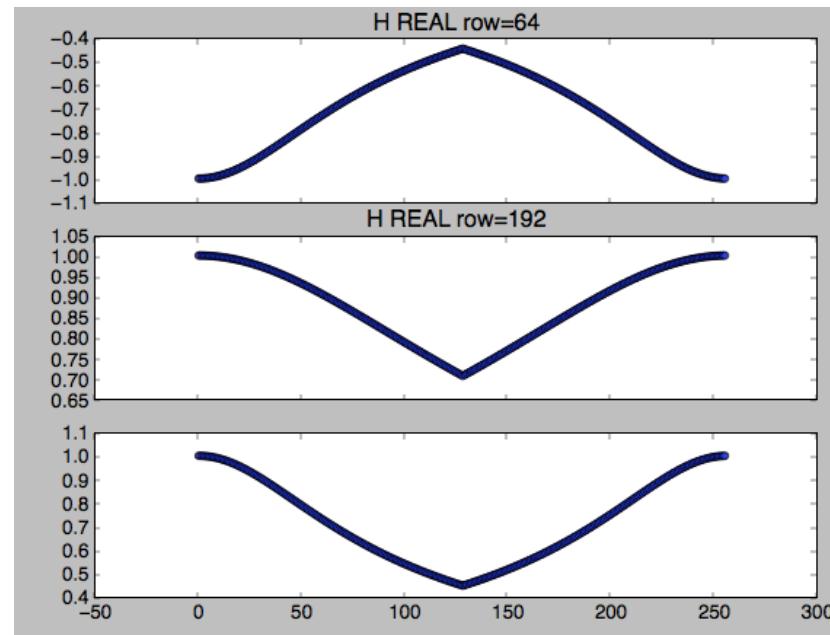


radius

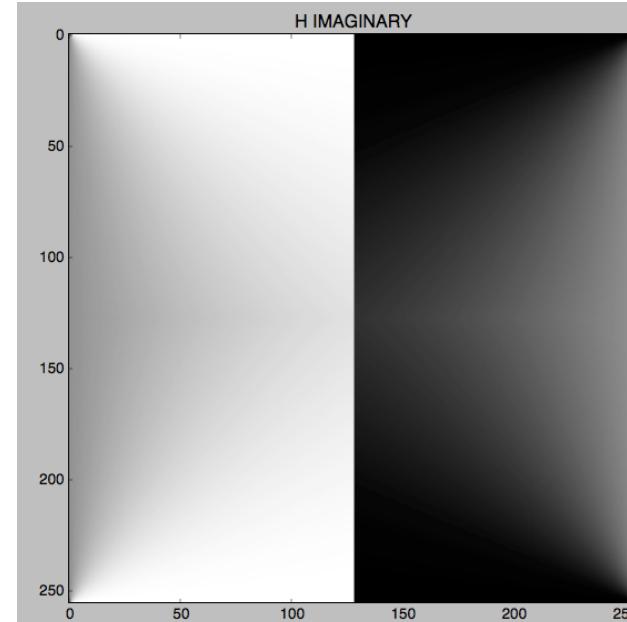
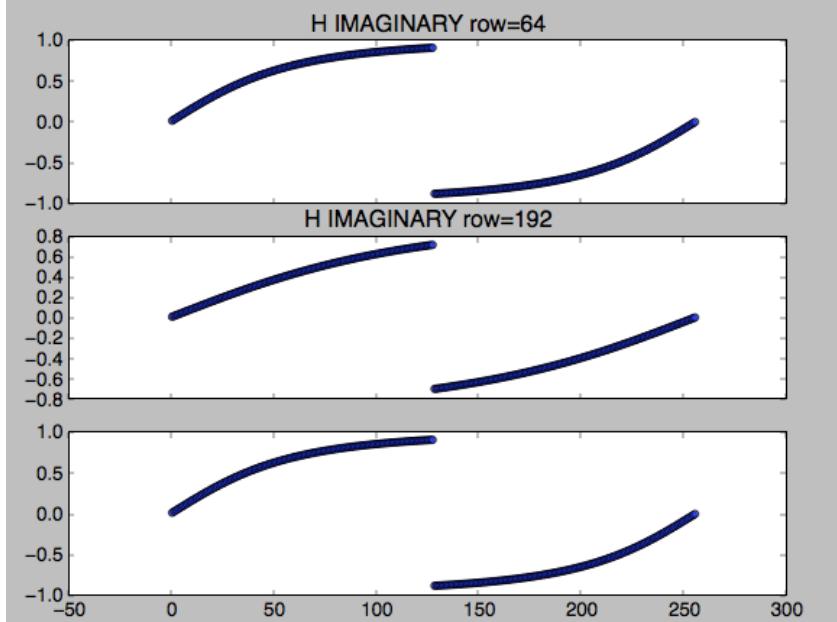


h1 = real(\mathbf{H})

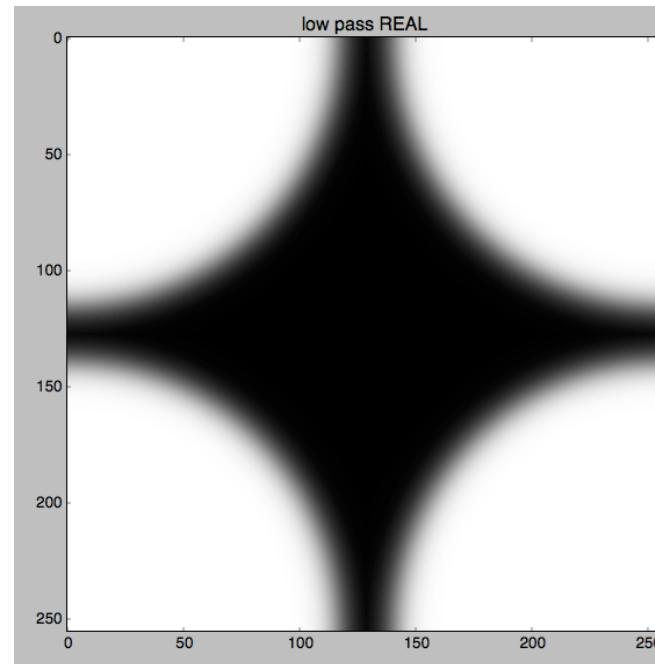
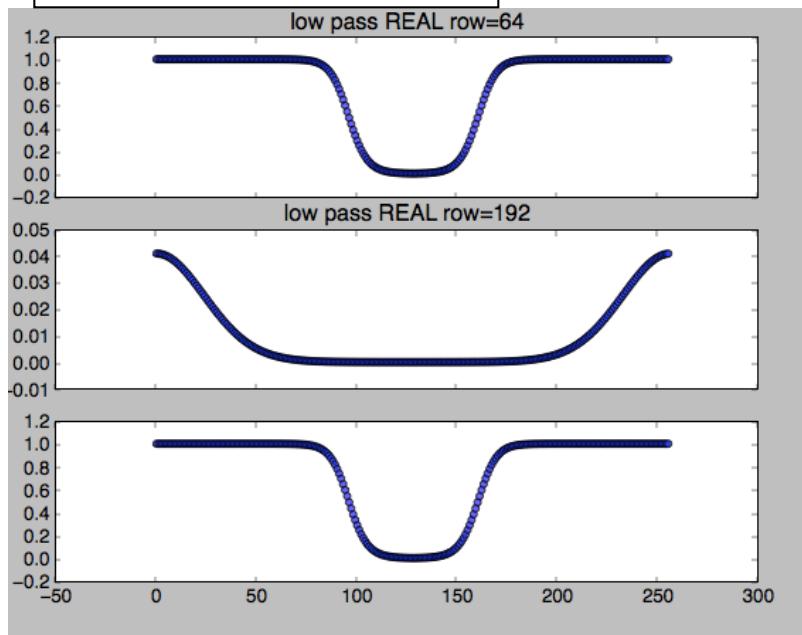
$$\mathbf{H} = (1j * \mathbf{u}_1 - \mathbf{u}_2) / \text{radius}$$



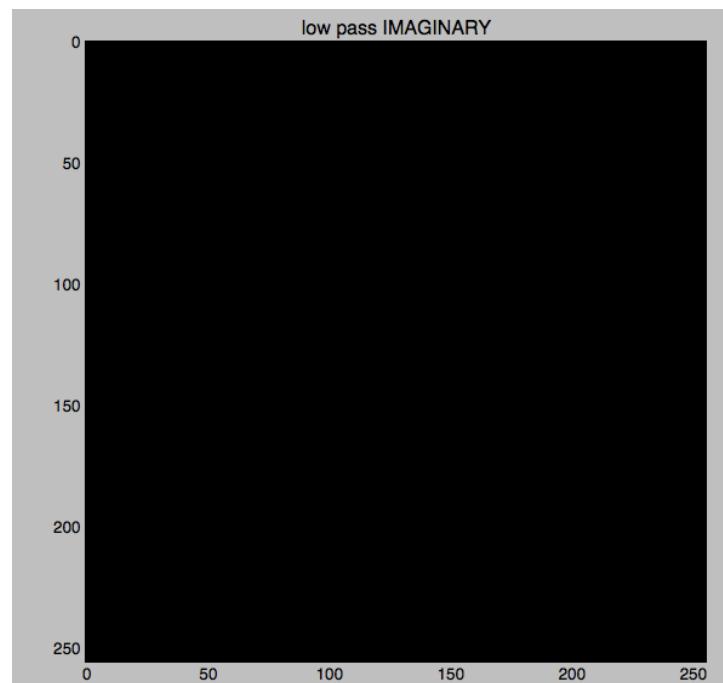
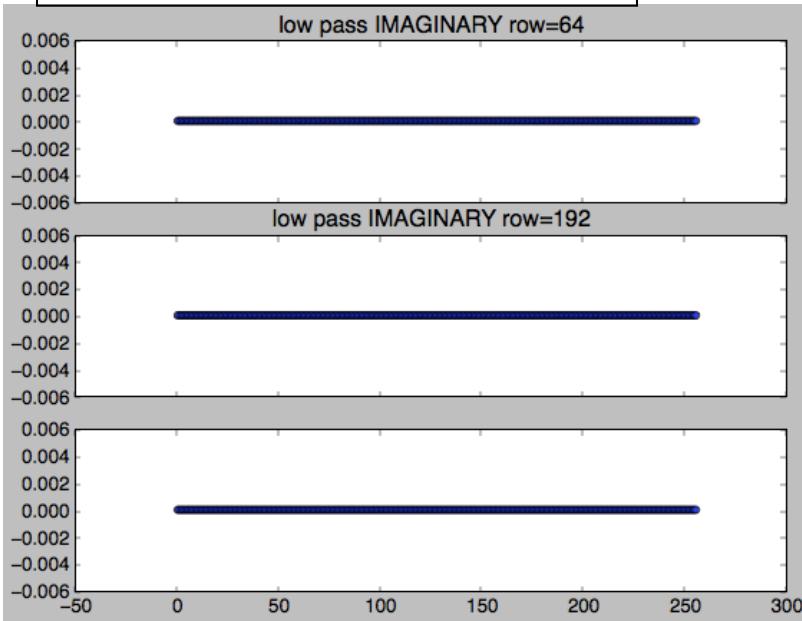
h2 = imaginary(\mathbf{H})



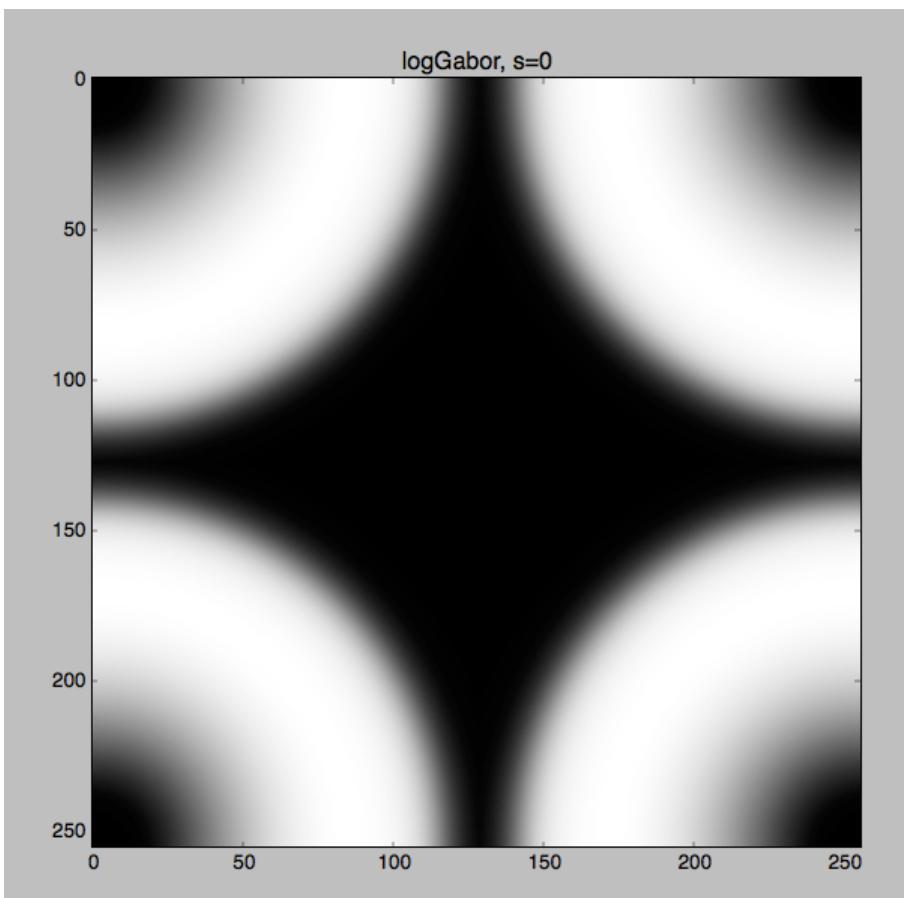
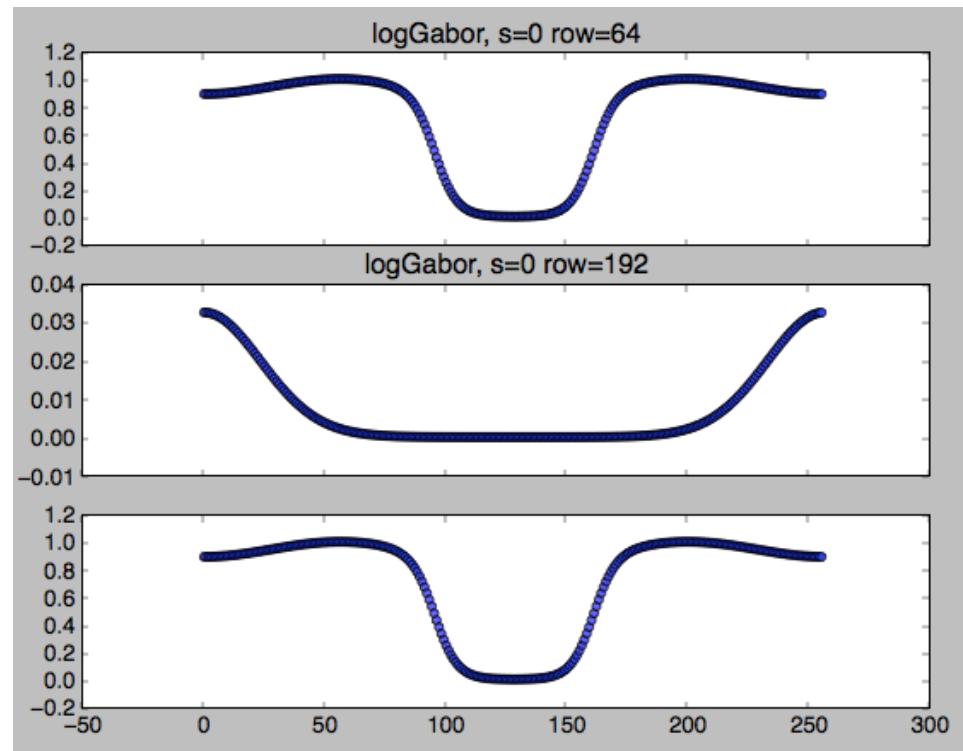
real(**low pass filter**)



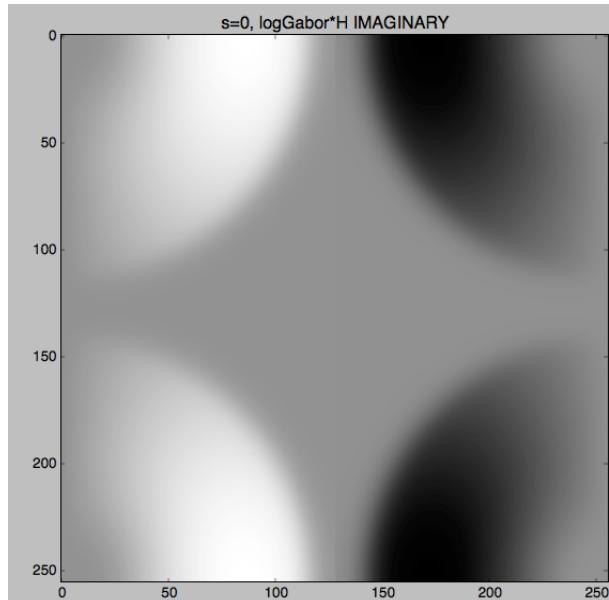
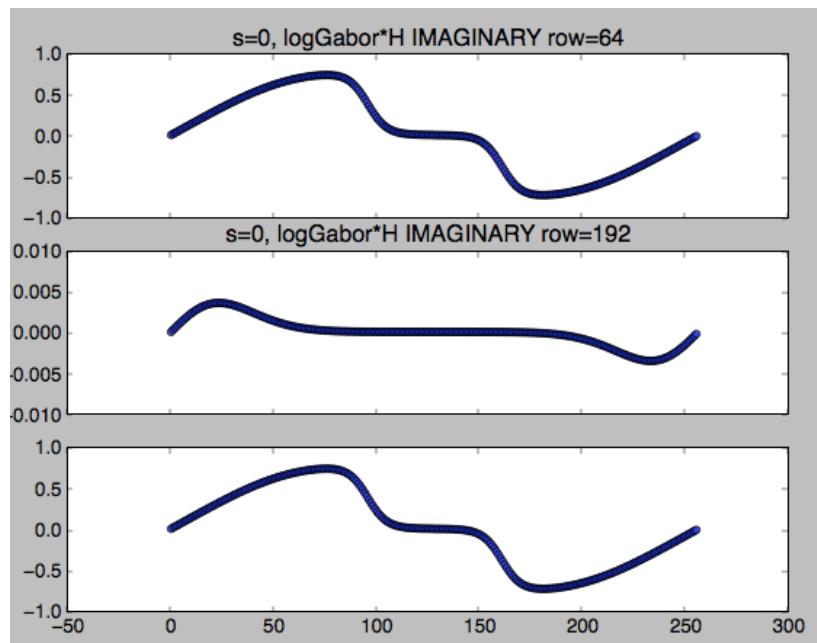
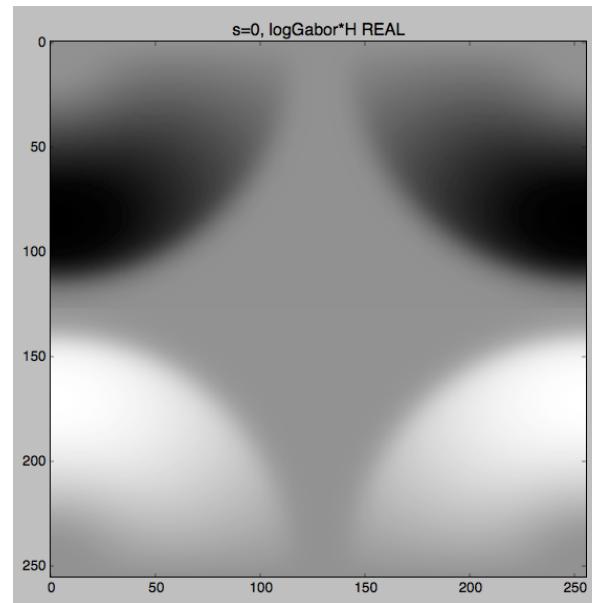
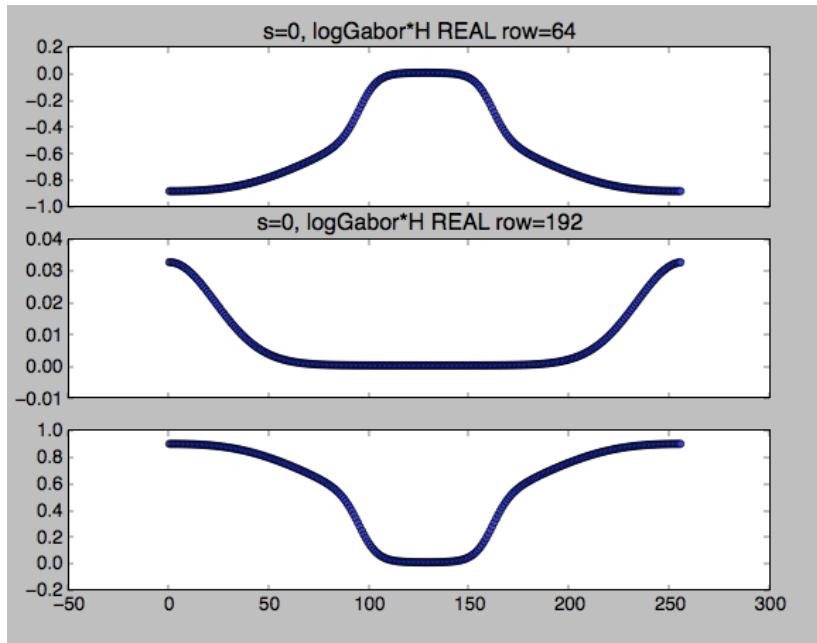
imaginary(**low pass filter**)



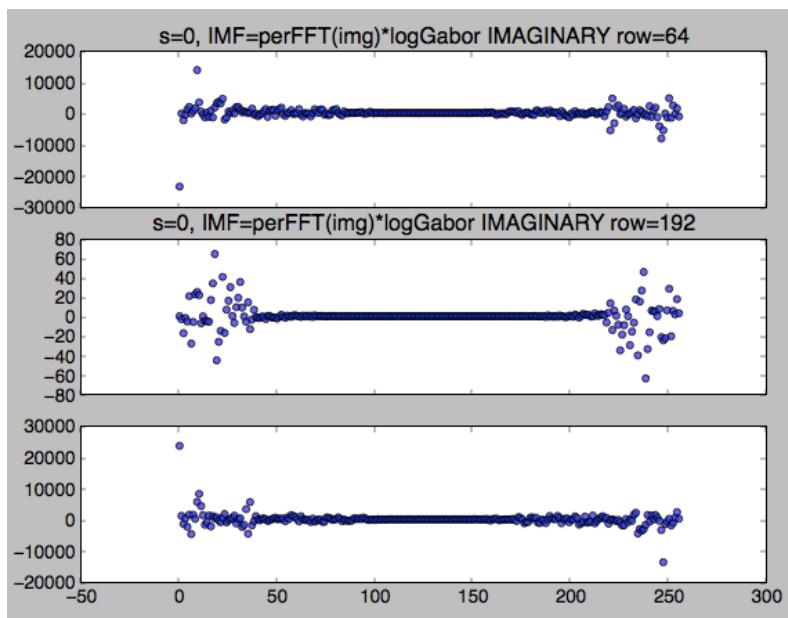
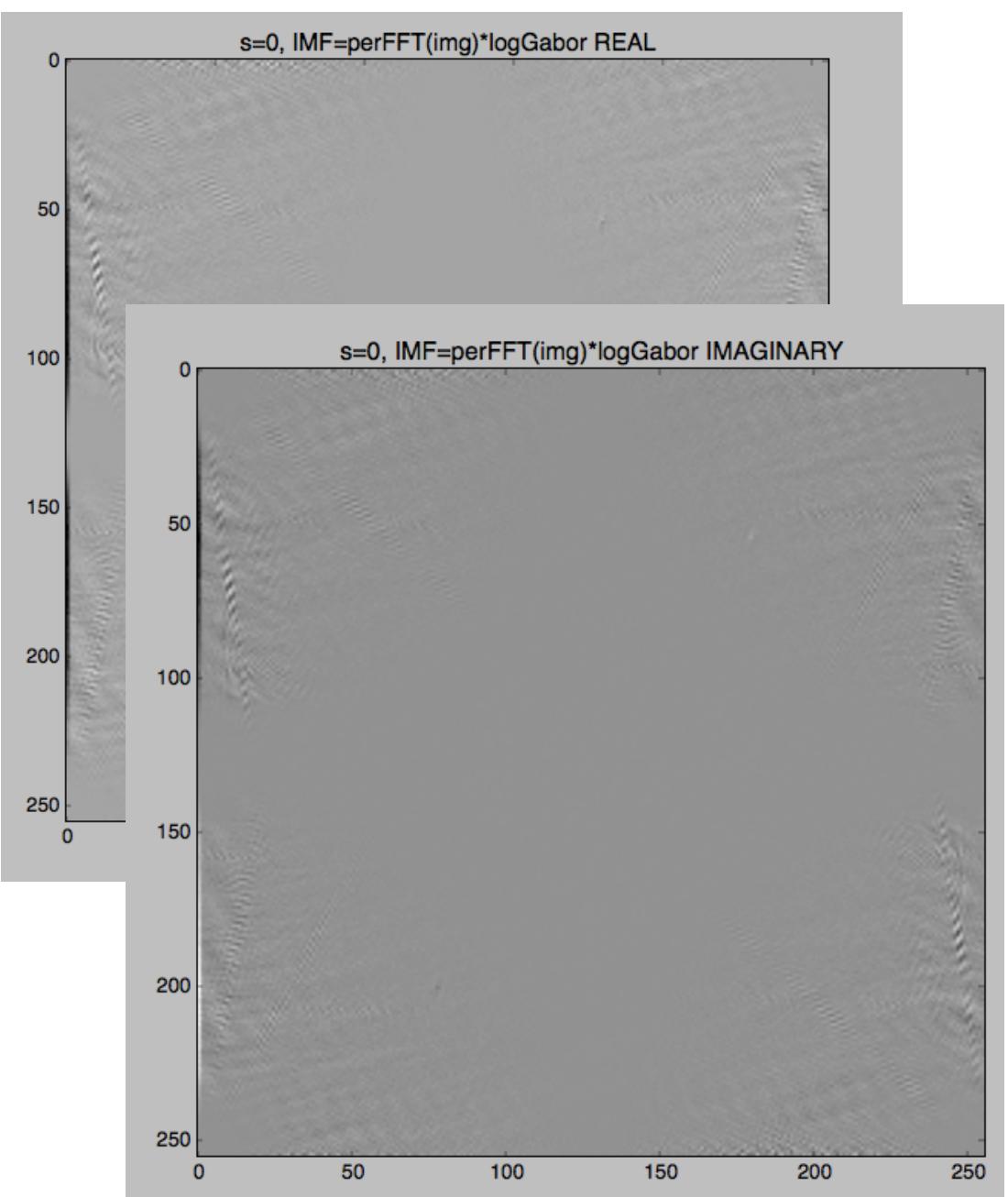
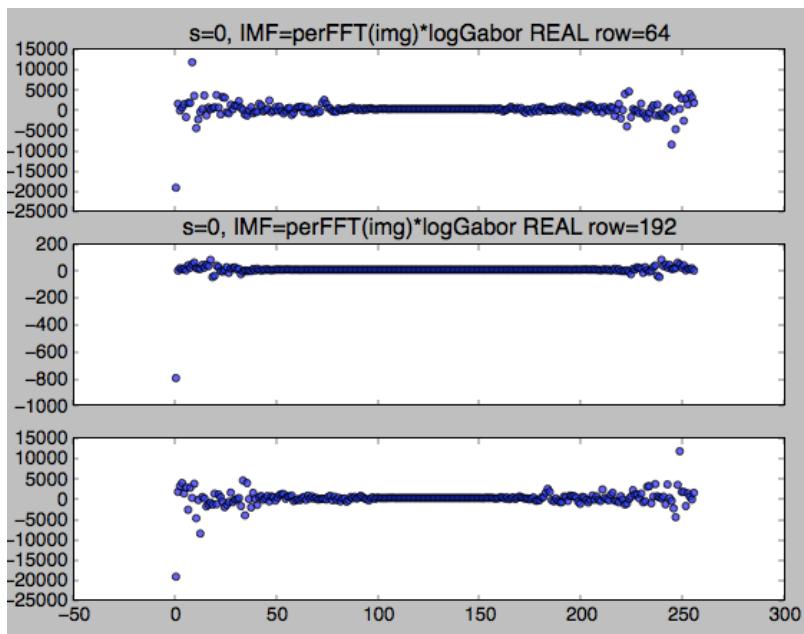
$s=0$, logGabor * low pass filter

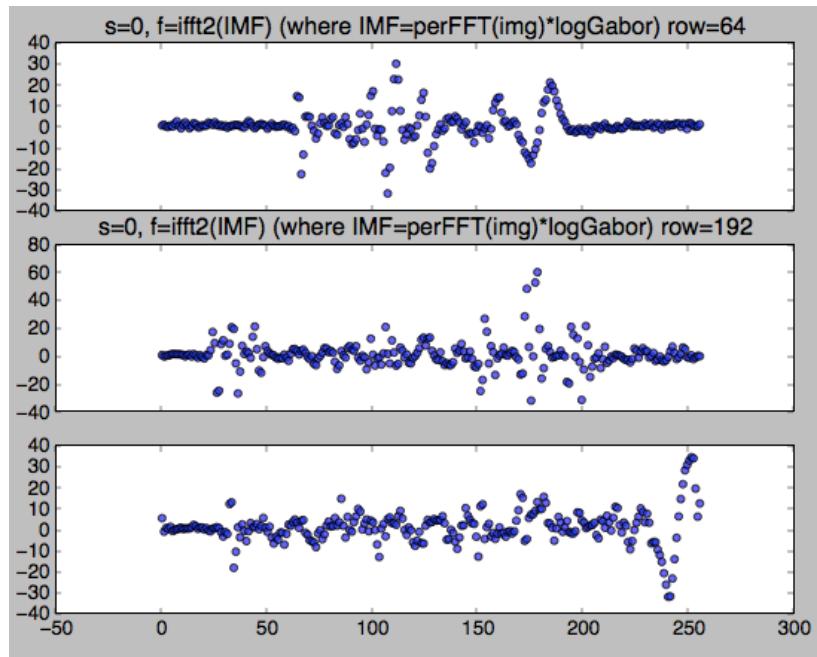


$s=0, \text{logGabor} * \text{low pass filter} * H$

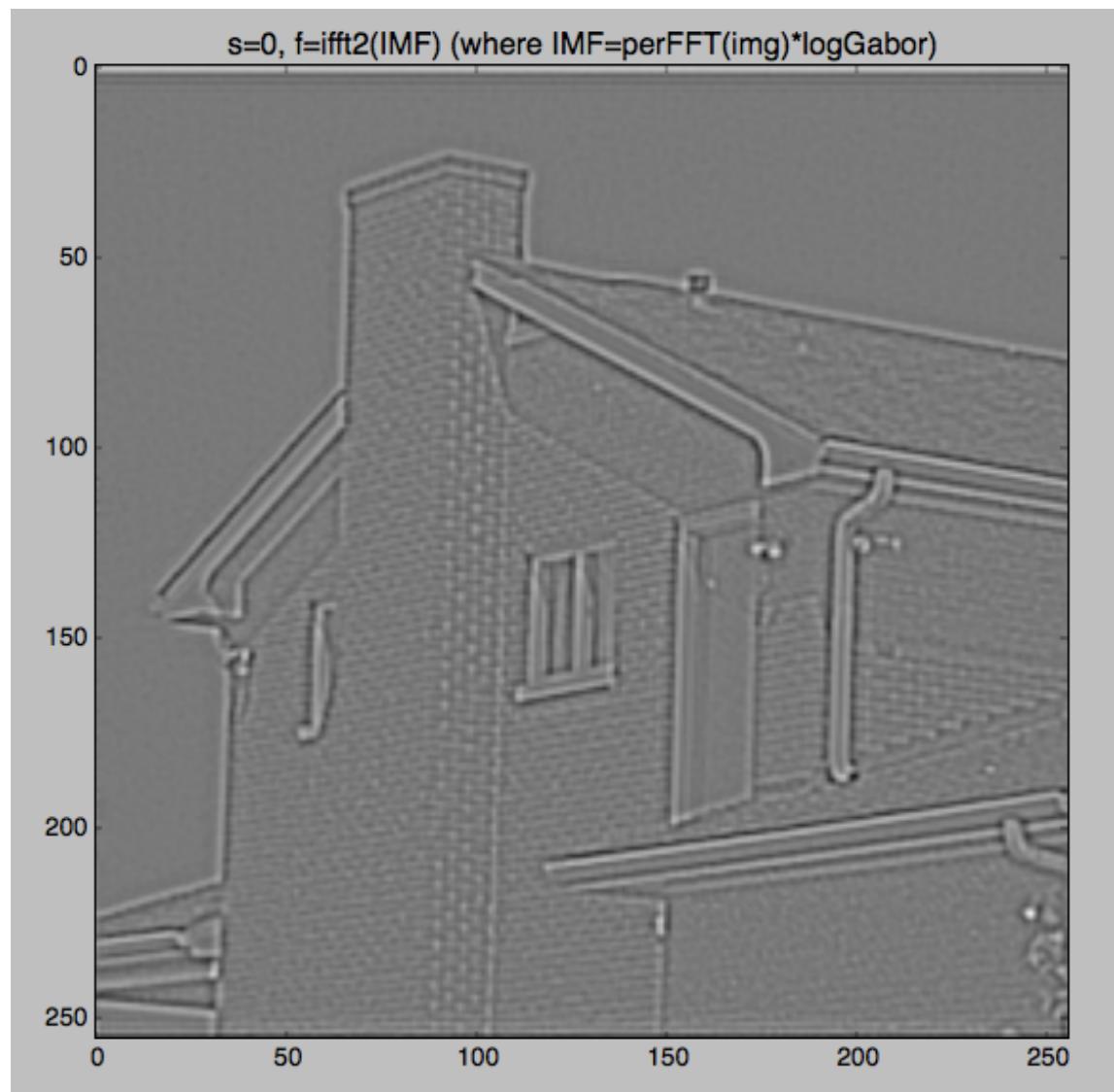


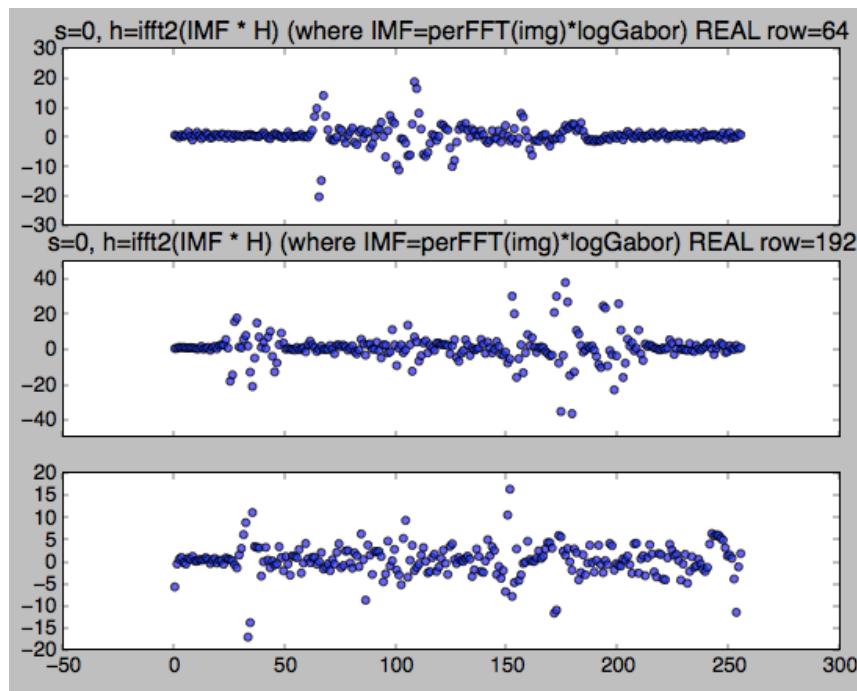
$s=0$, $\text{IMF}=\text{perFFT(img)} * \text{logGabor}$



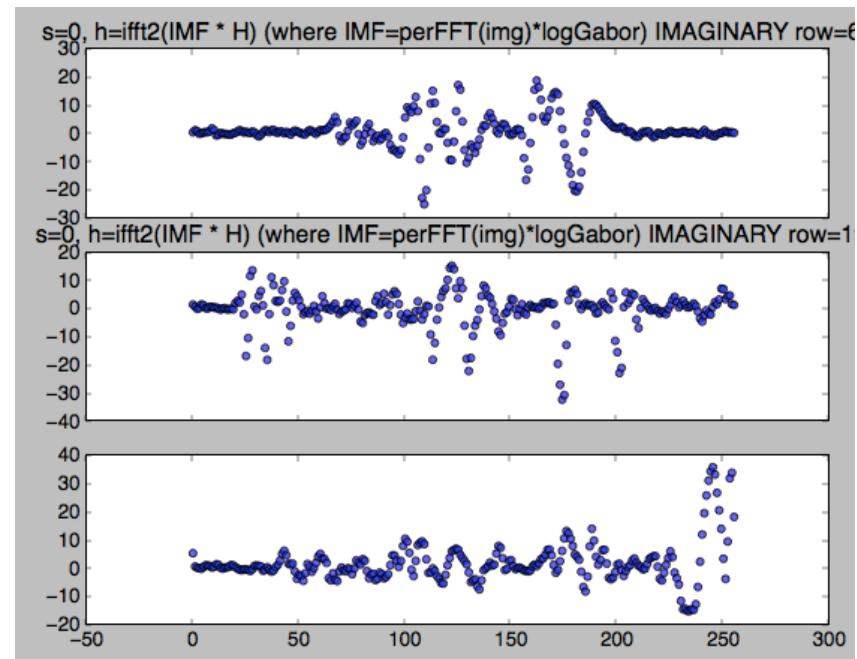
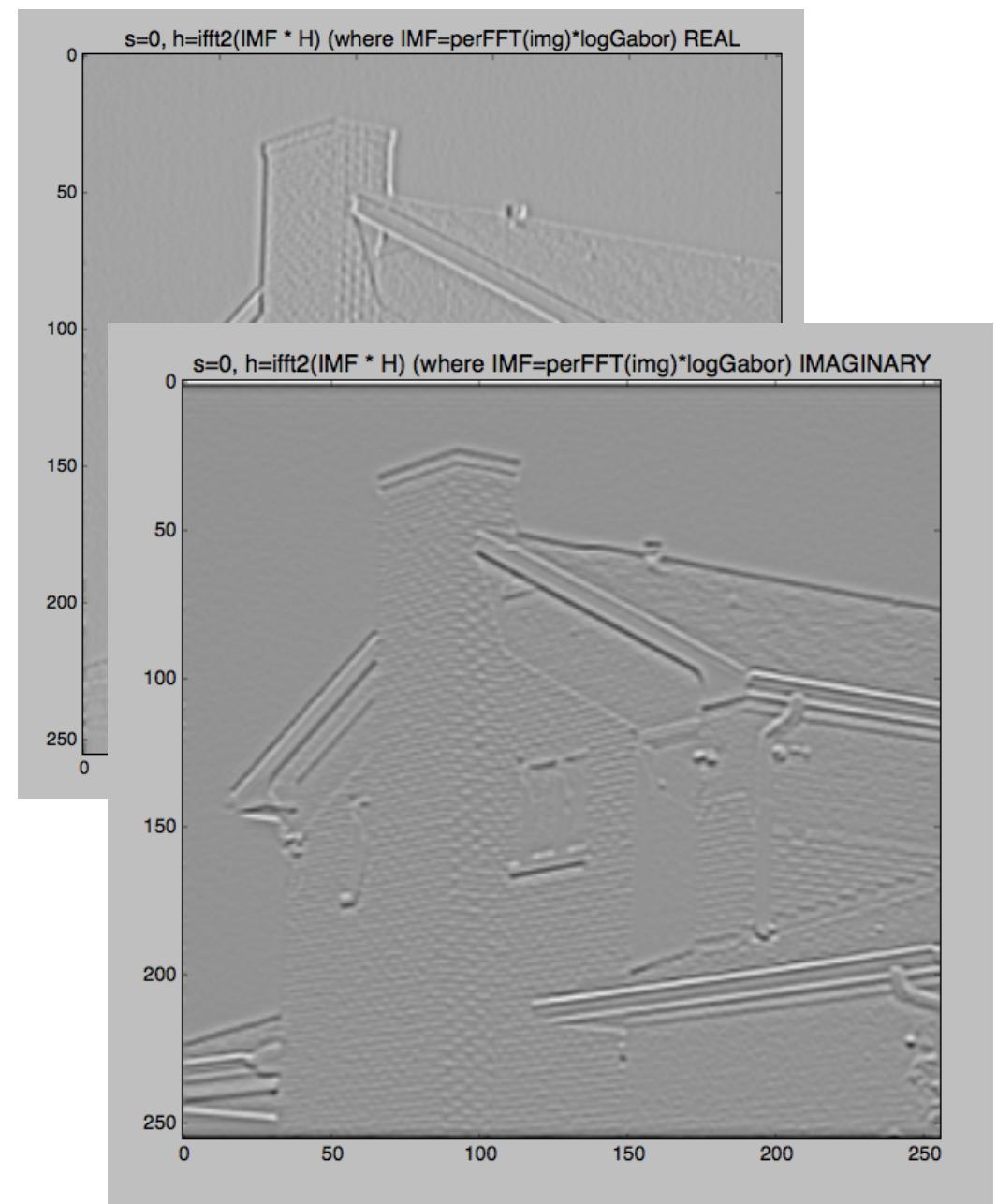


s=0, f=ifft2(IMF)
(where IMF=perFFT(img)*logGabor)

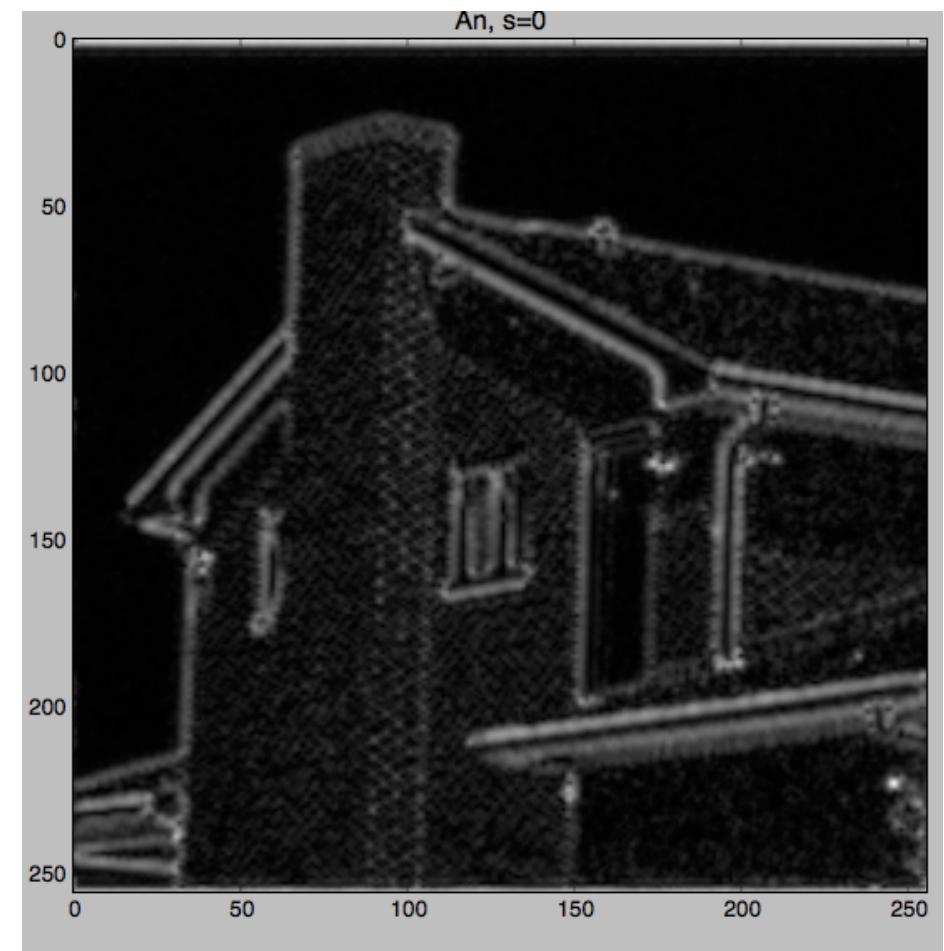
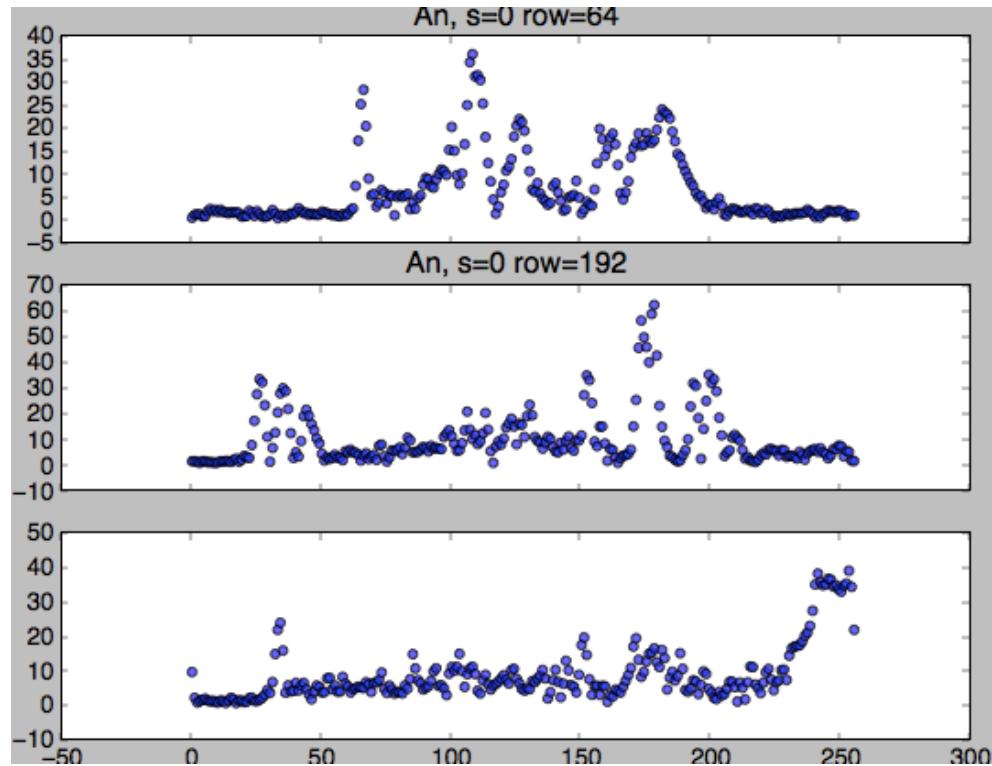




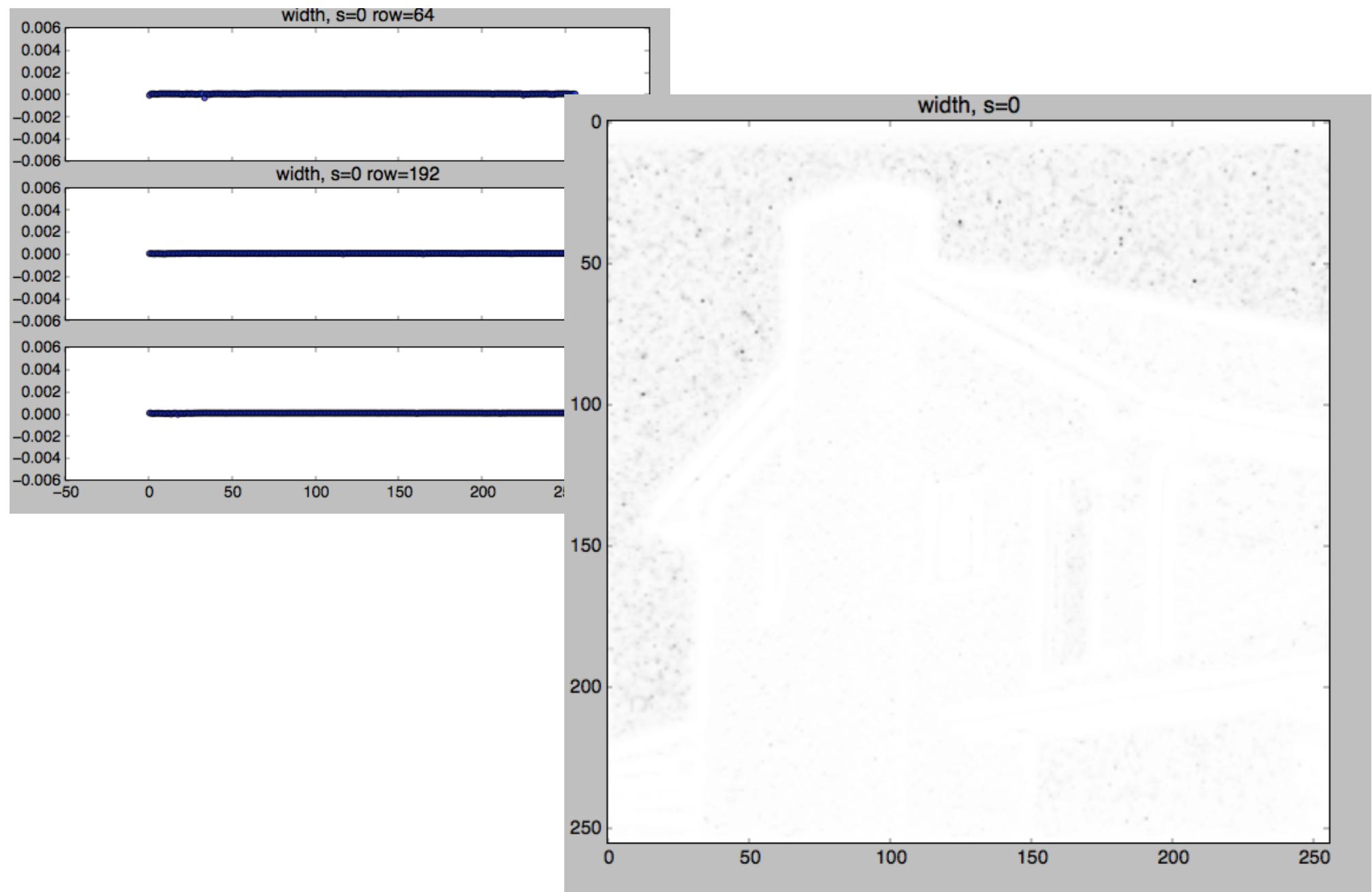
**s=0, h=ifft2(IMF * H)
(where IMF=perFFT(img)*logGabor)**



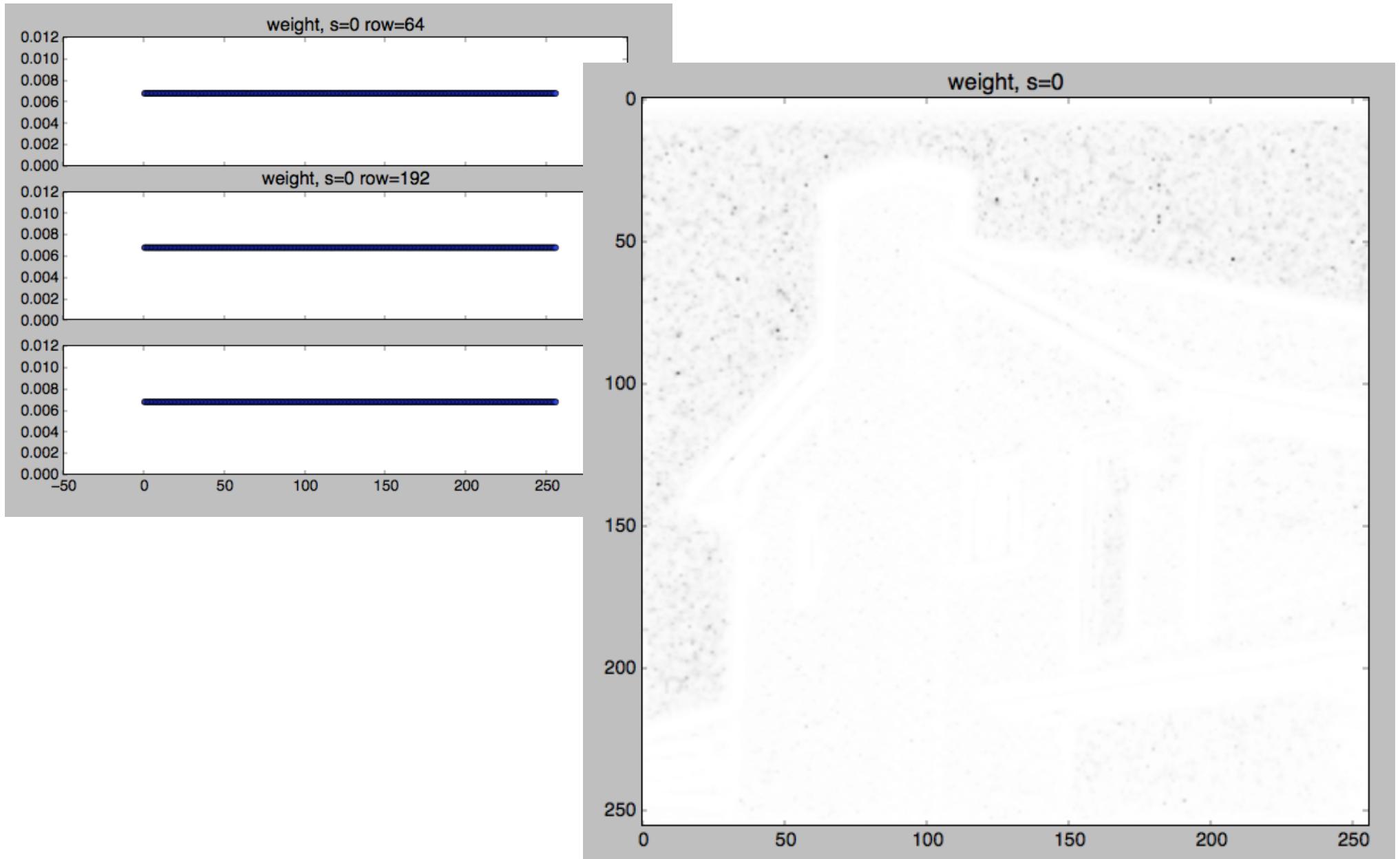
$s=0$, $A_n = \sqrt{f * f + h1 * h1 + h2 * h2}$
 (where $f = \text{iFFT}(\text{perFFT(img)} * \text{logGabor})$
 and $h = \text{iFFT}(\text{perFFT(img)} * \text{logGabor} * H)$)



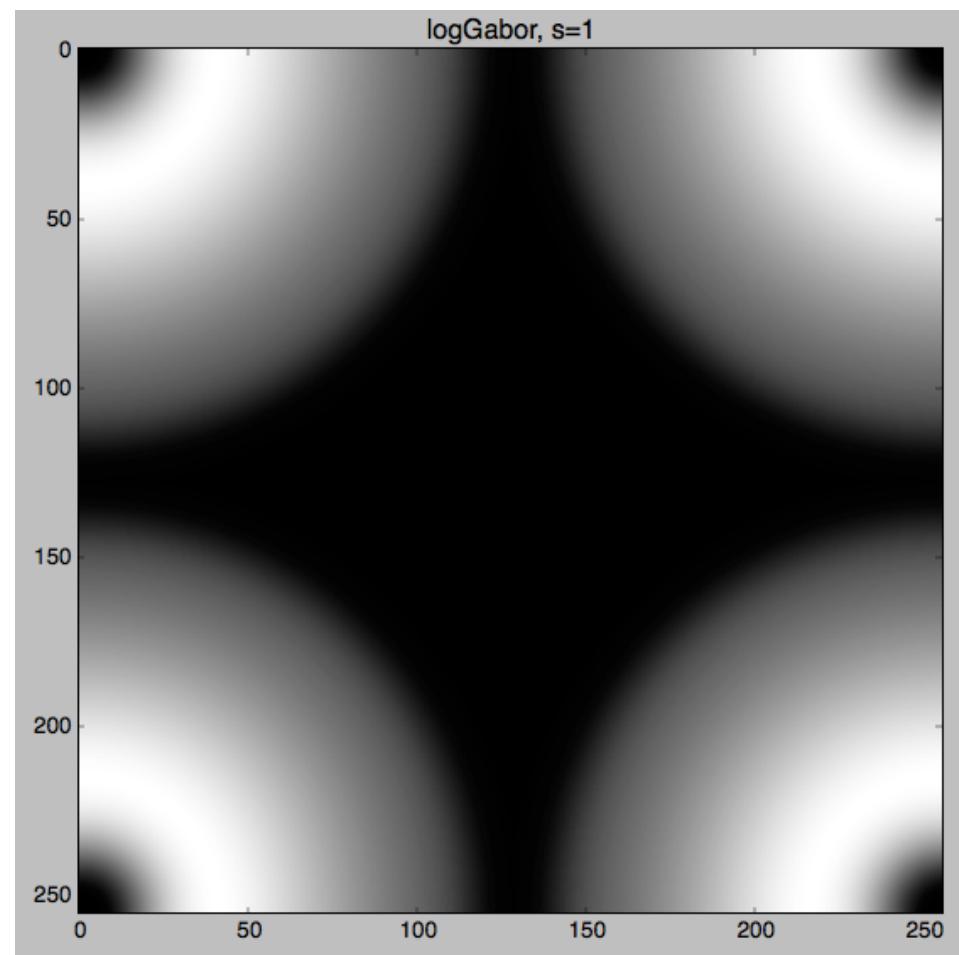
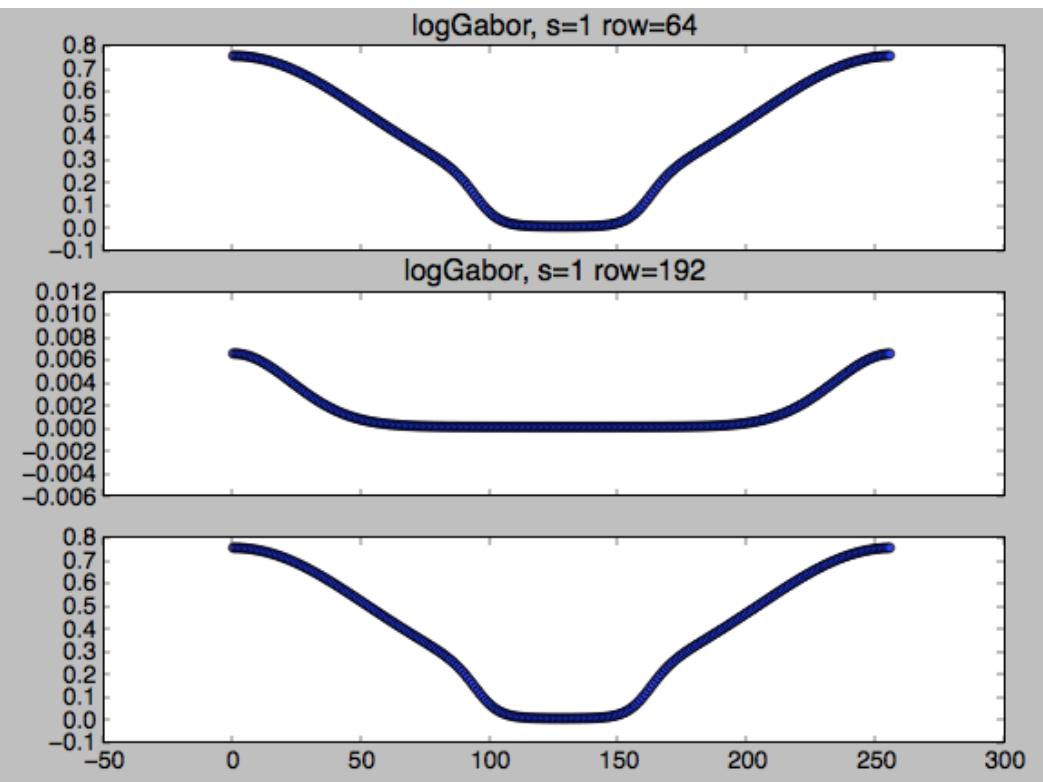
$s=0$, **width** = (sumAn / (maxAn + epsilon) - 1.) / (nscale - 1)



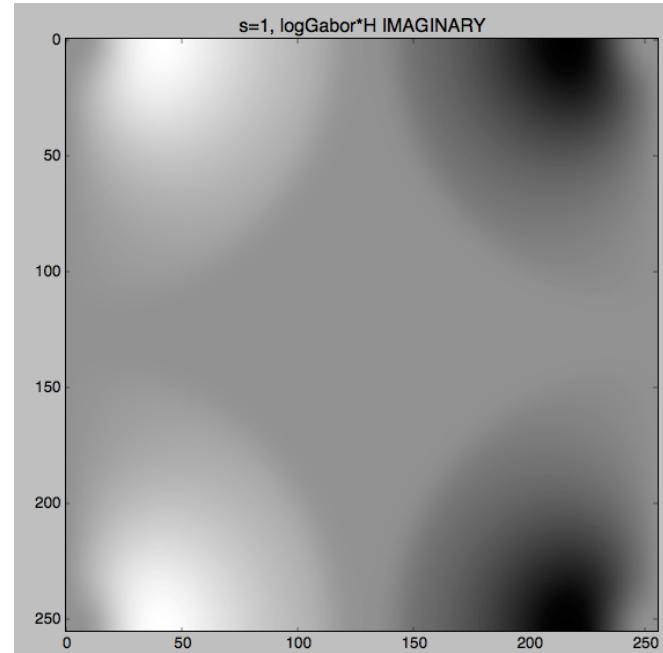
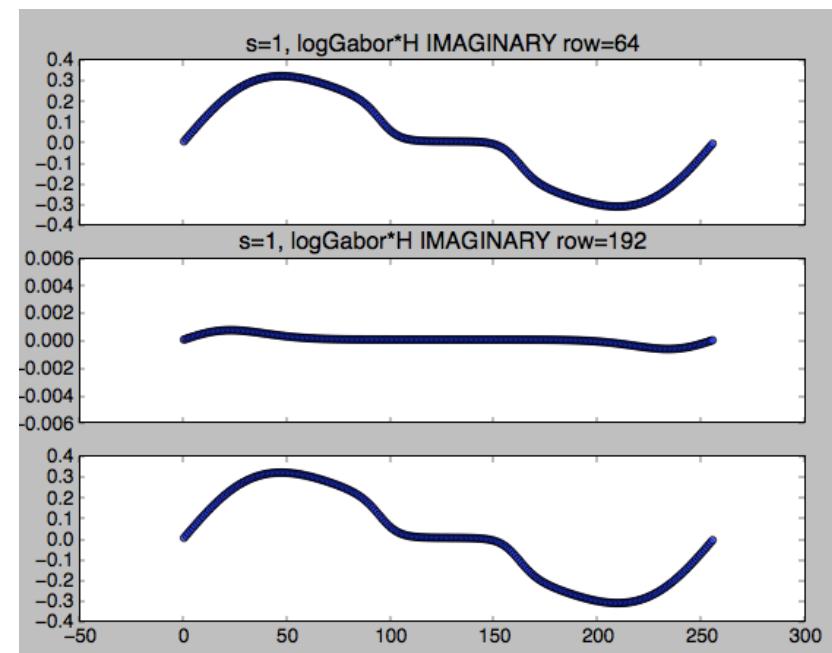
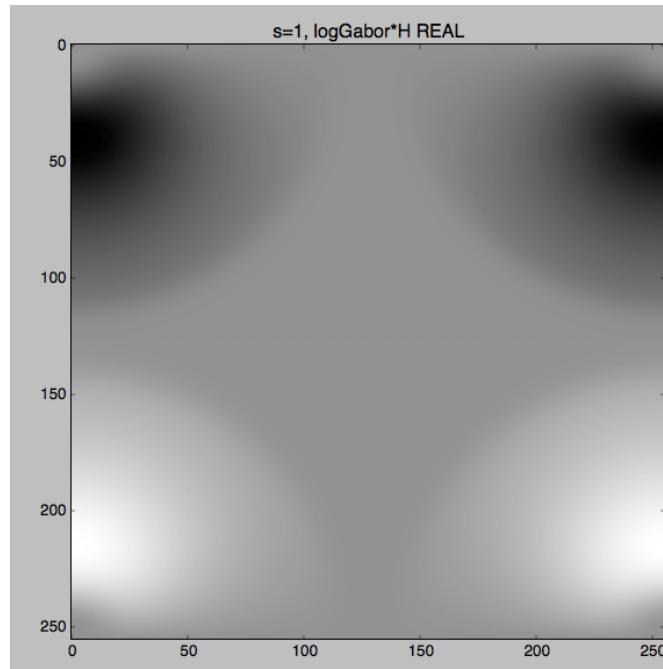
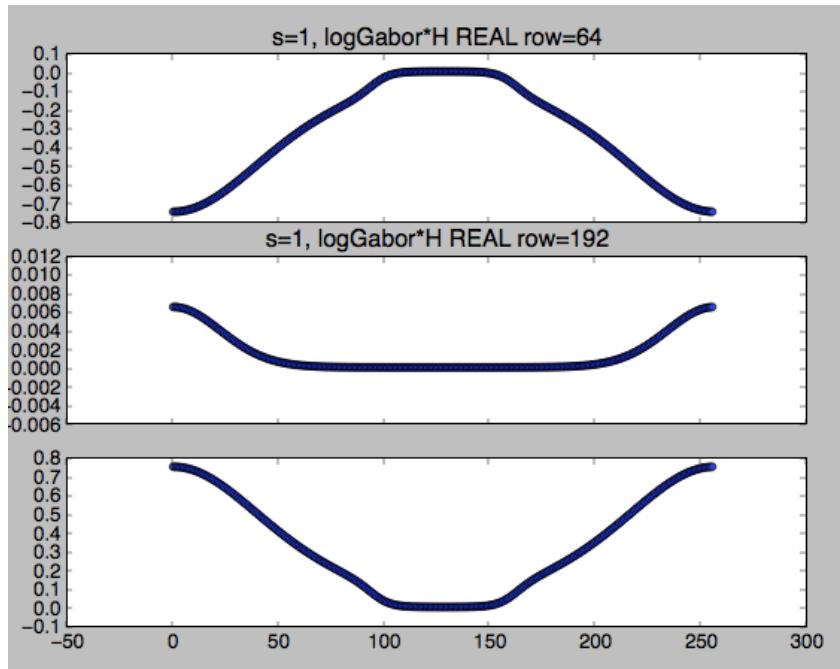
$s=0$, **weight** = $1. / (1. + \exp(g * (\text{cutoff} - \text{width})))$
(where $\text{cutoff}=0.5$, $g=10.$)

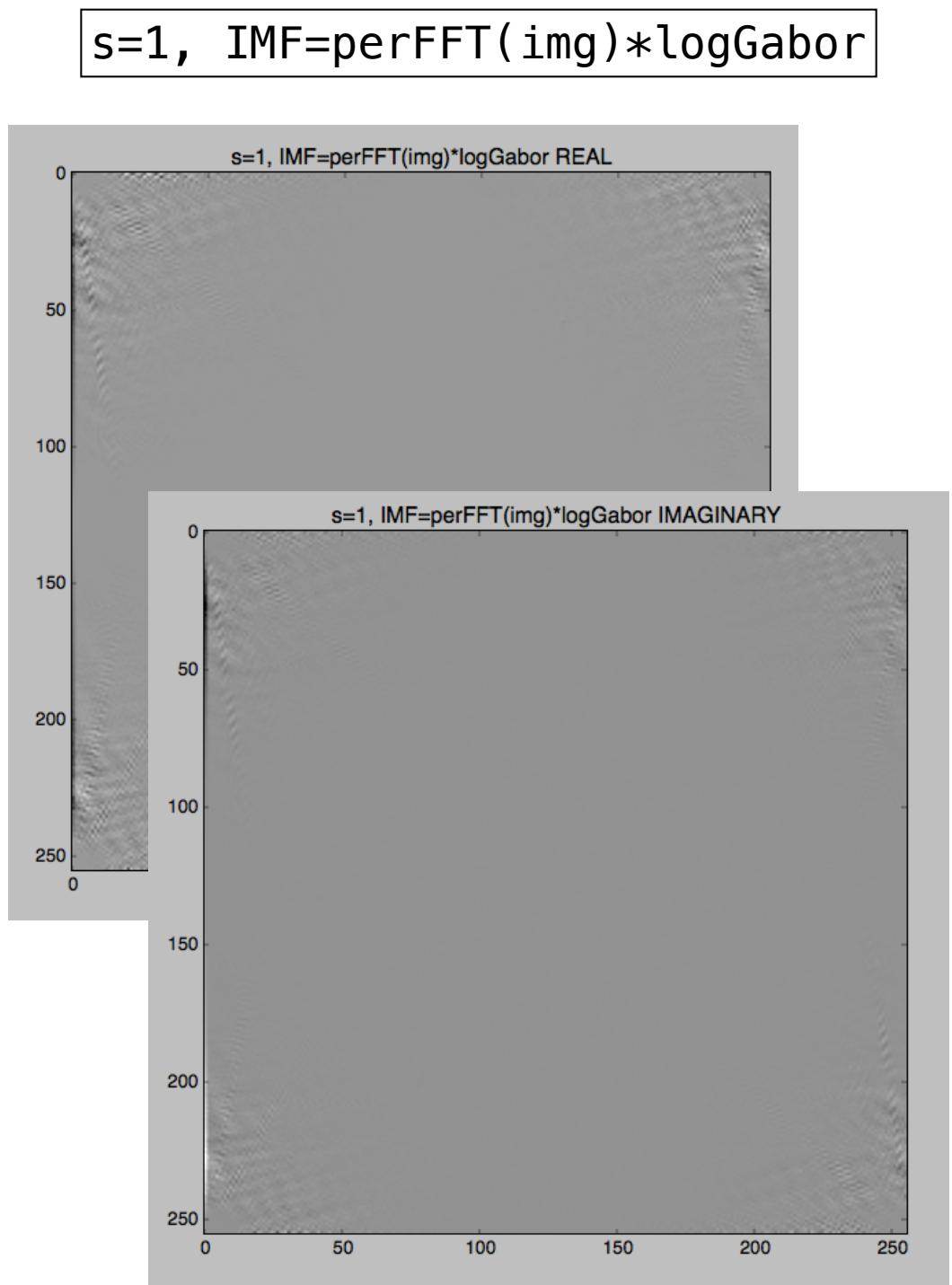
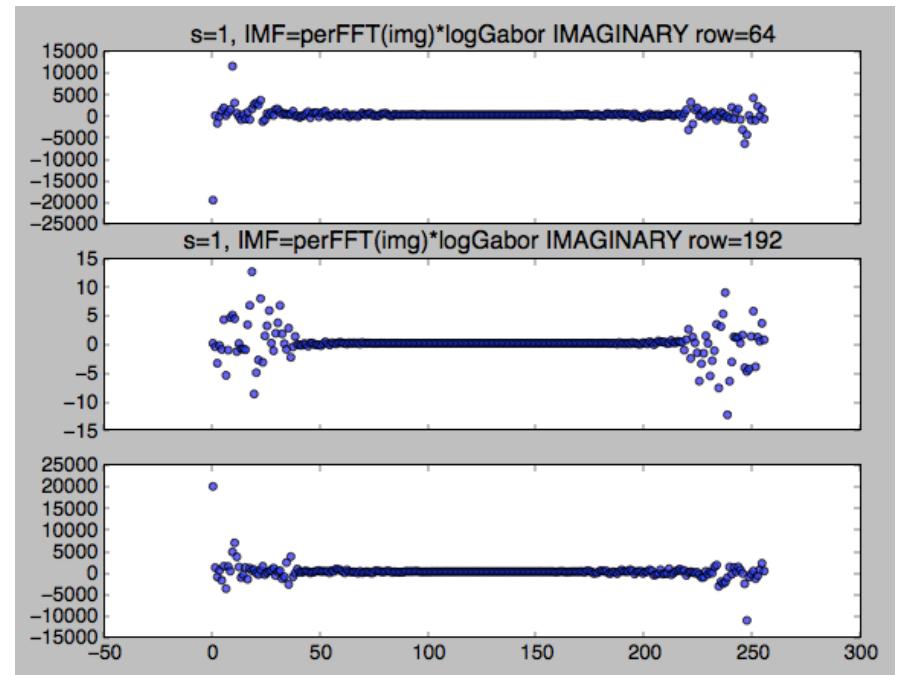
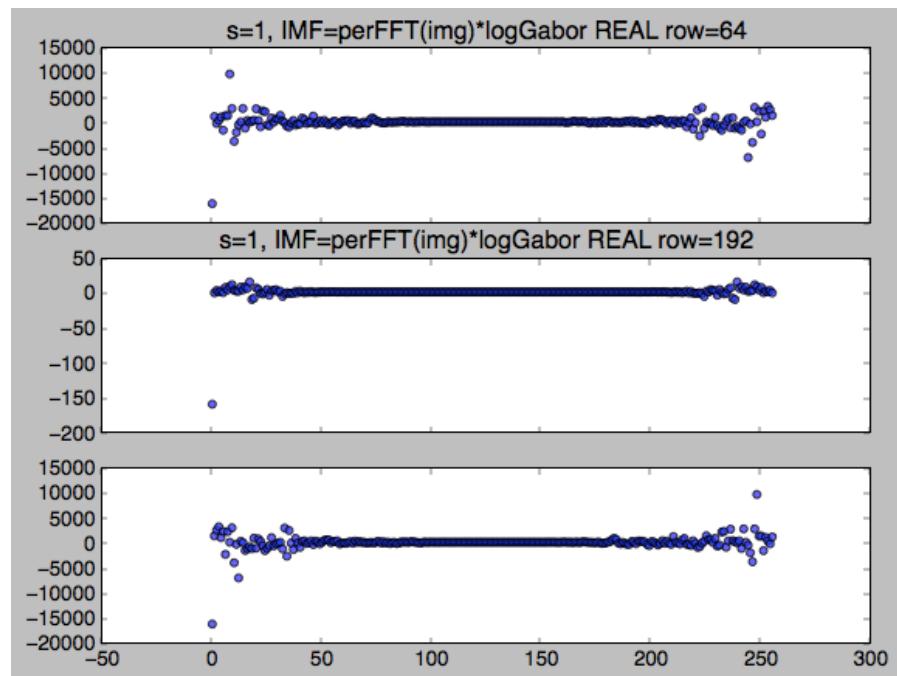


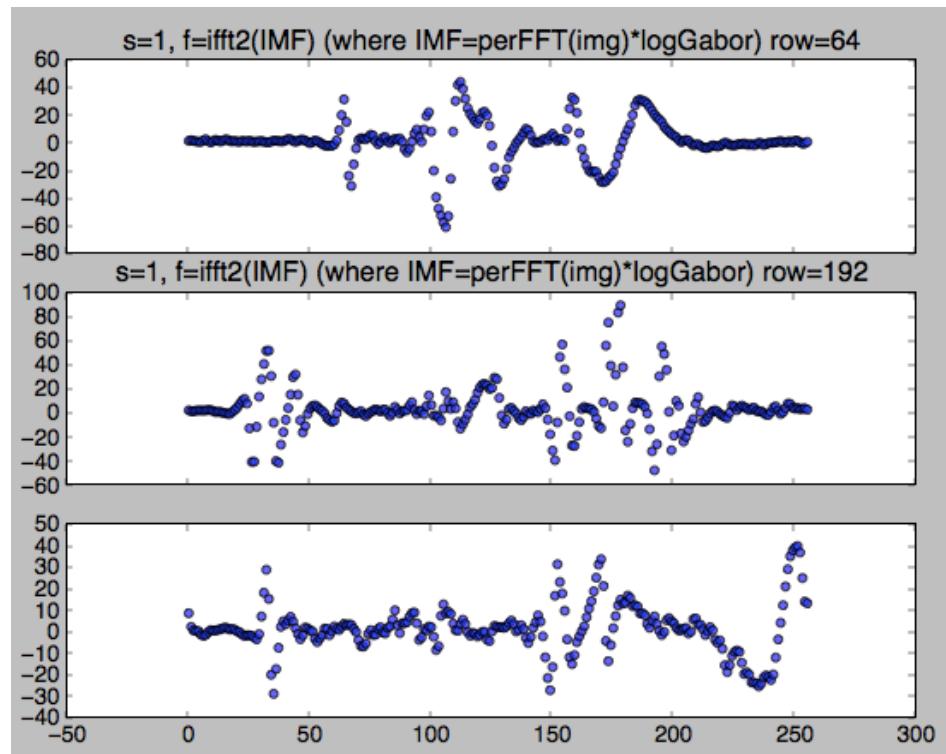
$s=1$, logGabor * low pass filter



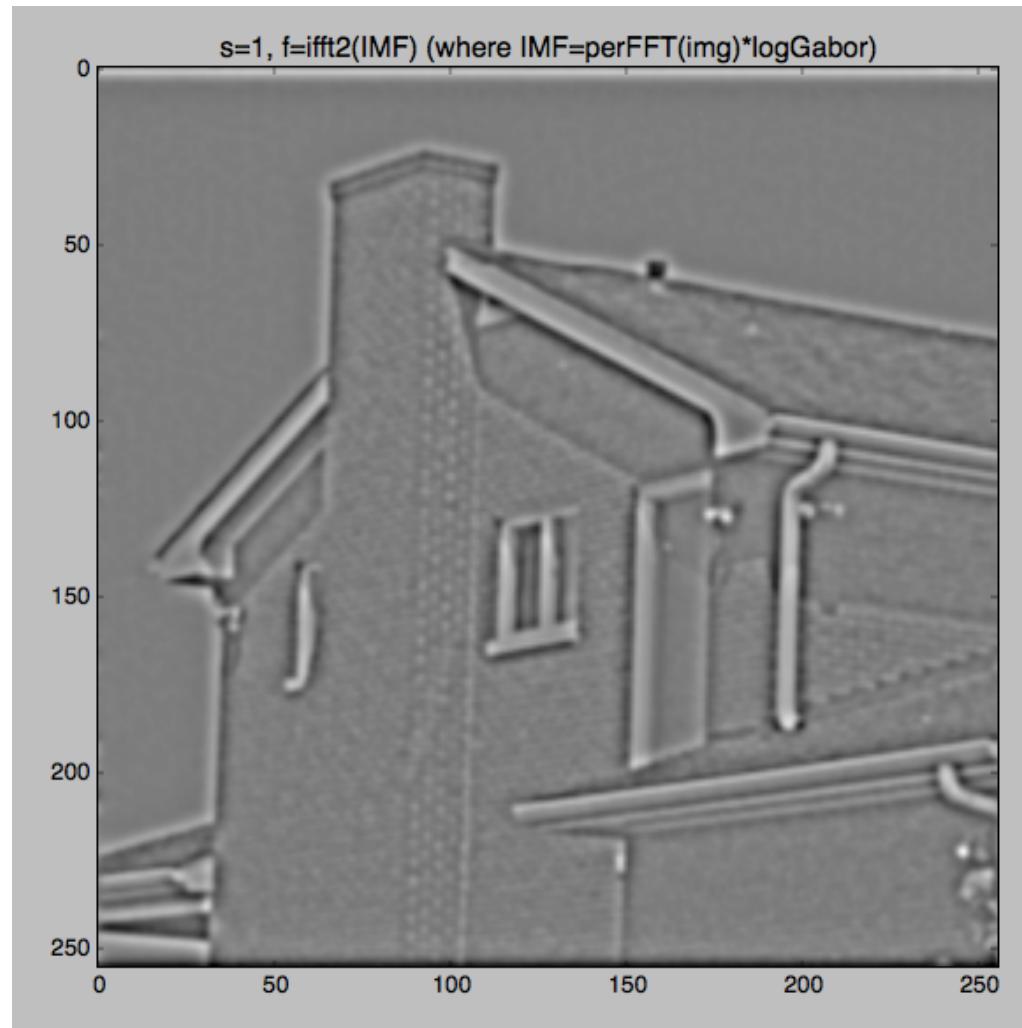
$s=1$, logGabor * low pass filter * H



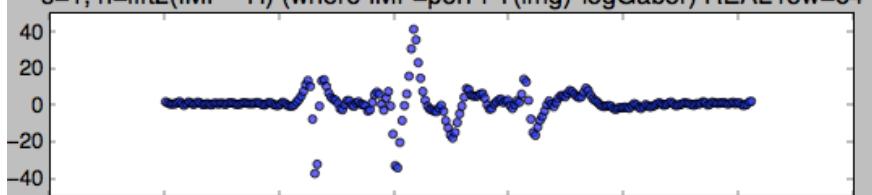




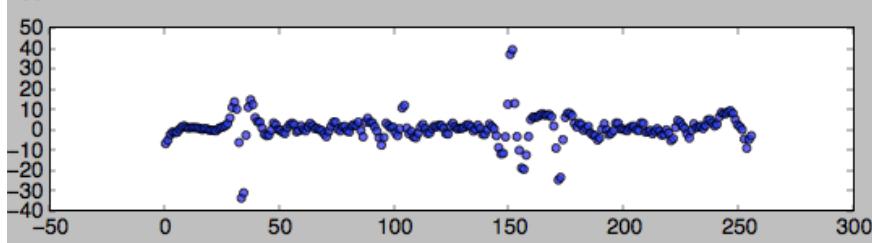
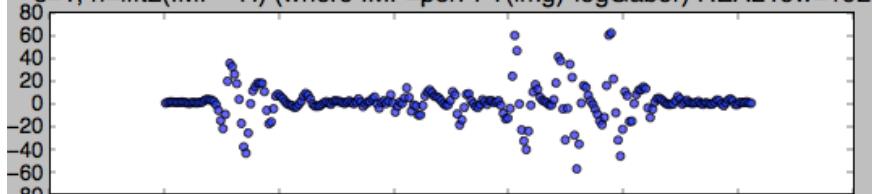
s=1, f=ifft2(IMF)
(where IMF=perFFT(img)*logGabor)



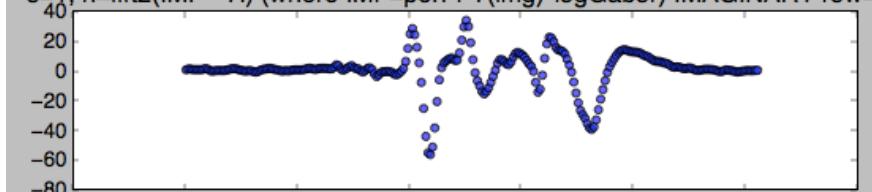
s=1, h=ifft2(IMF * H) (where IMF=perFFT(img)*logGabor) REAL row=64



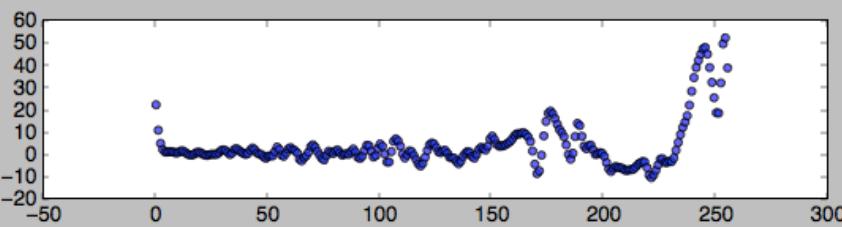
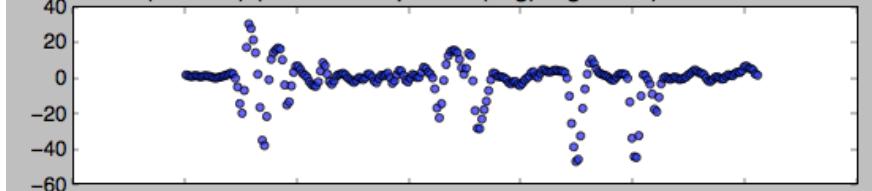
s=1, h=ifft2(IMF * H) (where IMF=perFFT(img)*logGabor) REAL row=192



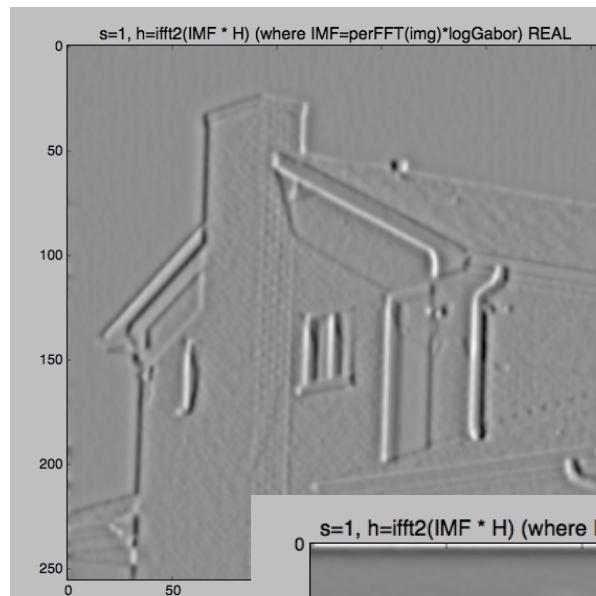
s=1, h=ifft2(IMF * H) (where IMF=perFFT(img)*logGabor) IMAGINARY row=



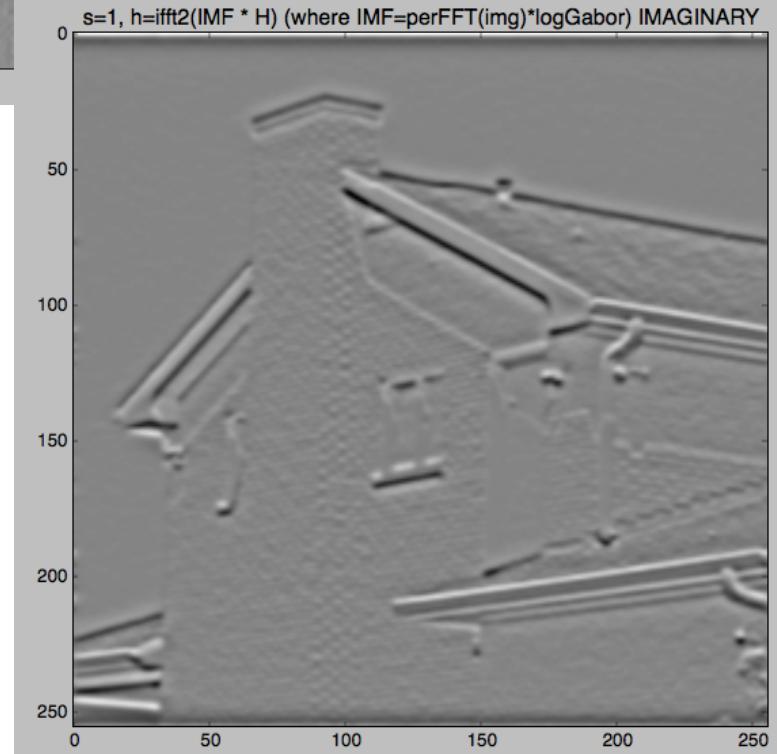
s=1, h=ifft2(IMF * H) (where IMF=perFFT(img)*logGabor) IMAGINARY row=1



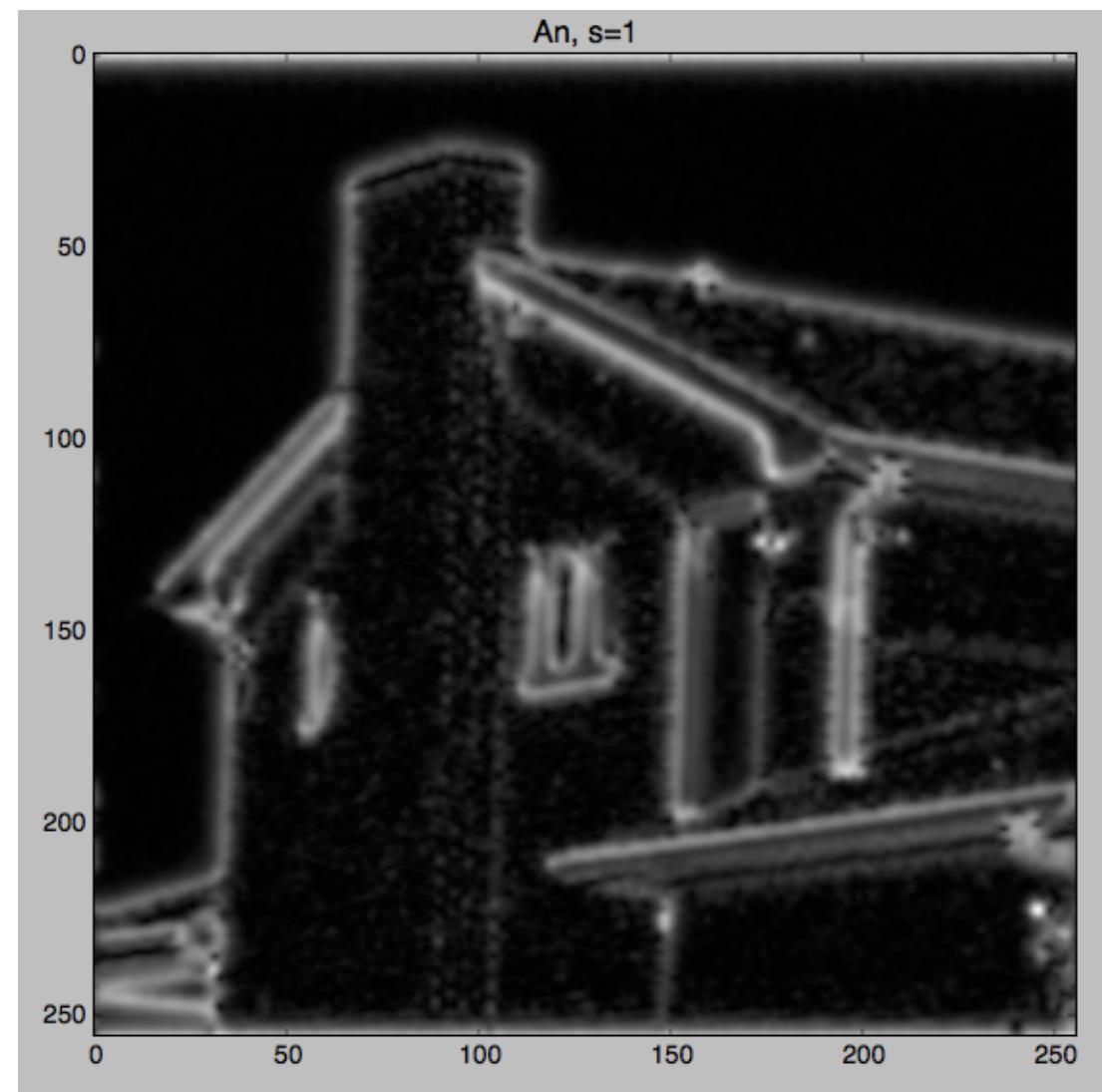
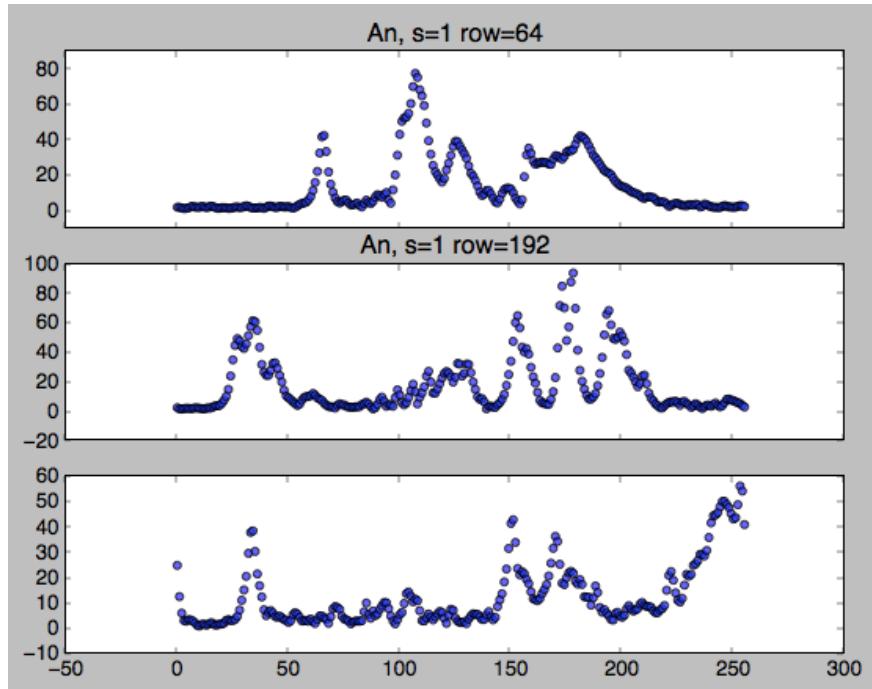
s=1, h=ifft2(IMF * H)
(where IMF=perFFT(img)*logGabor)



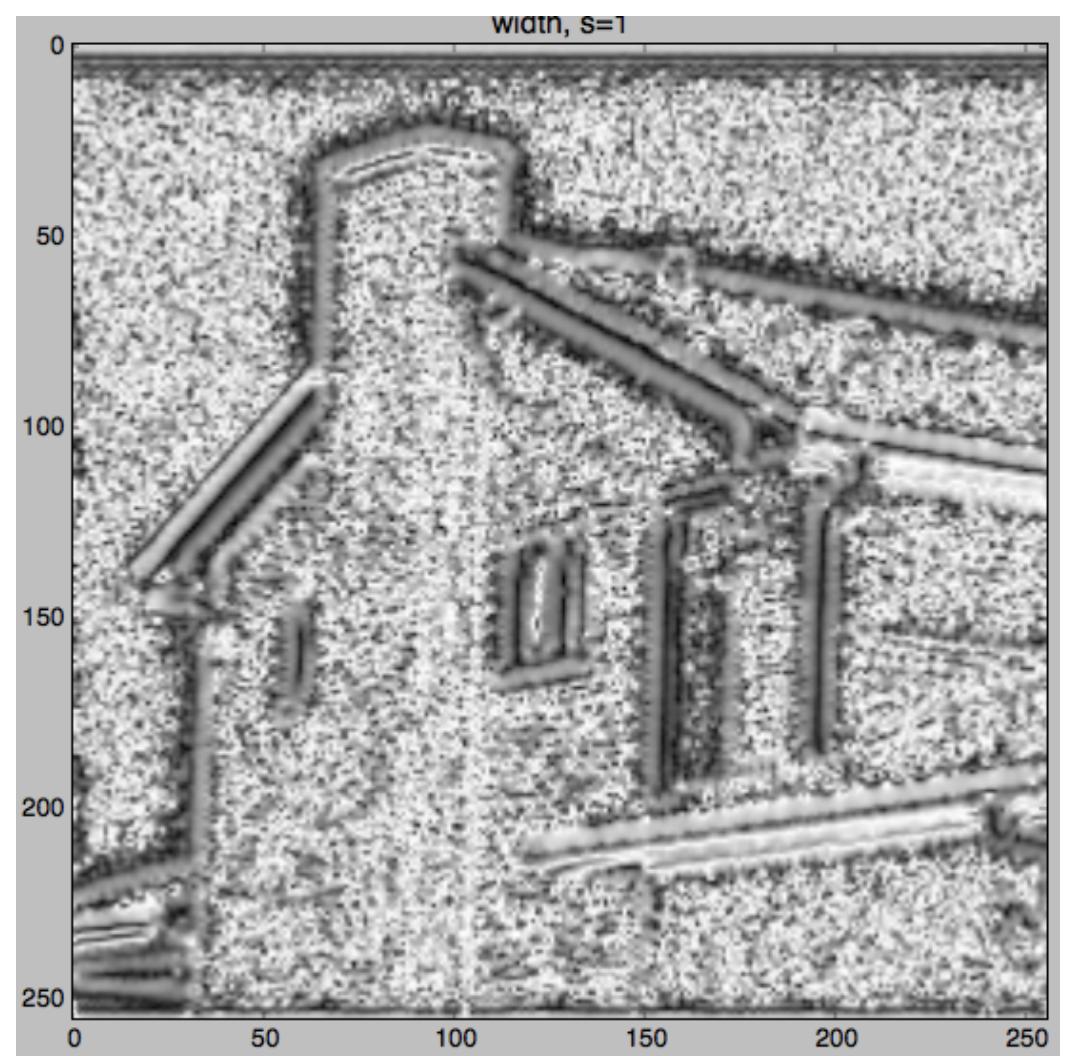
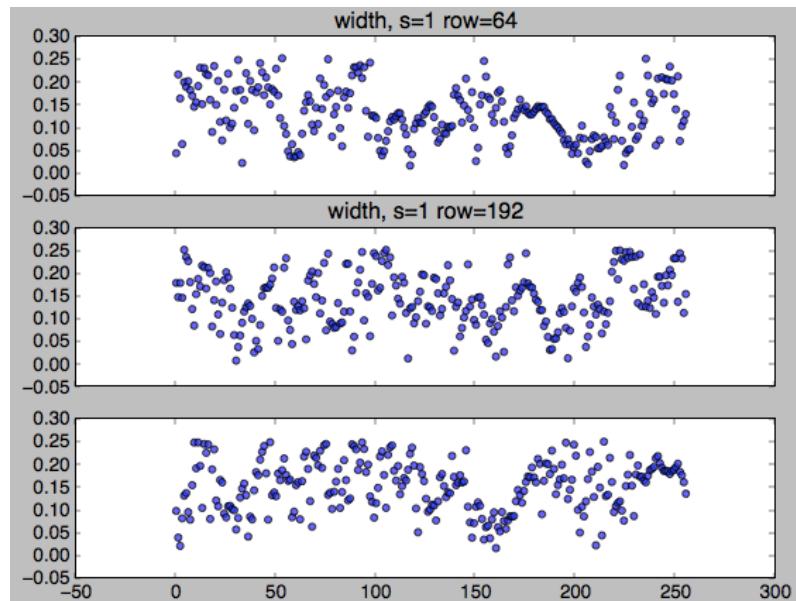
s=1, h=ifft2(IMF * H) (where IMF=perFFT(img)*logGabor) IMAGINARY



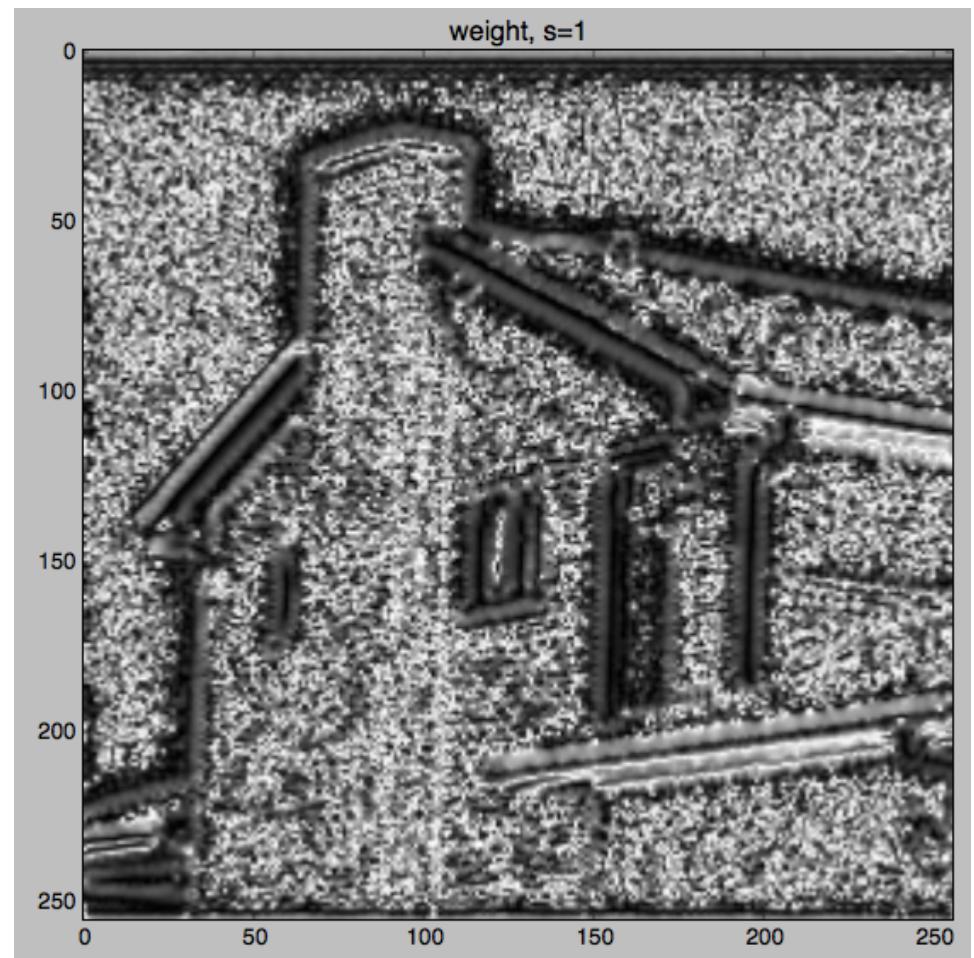
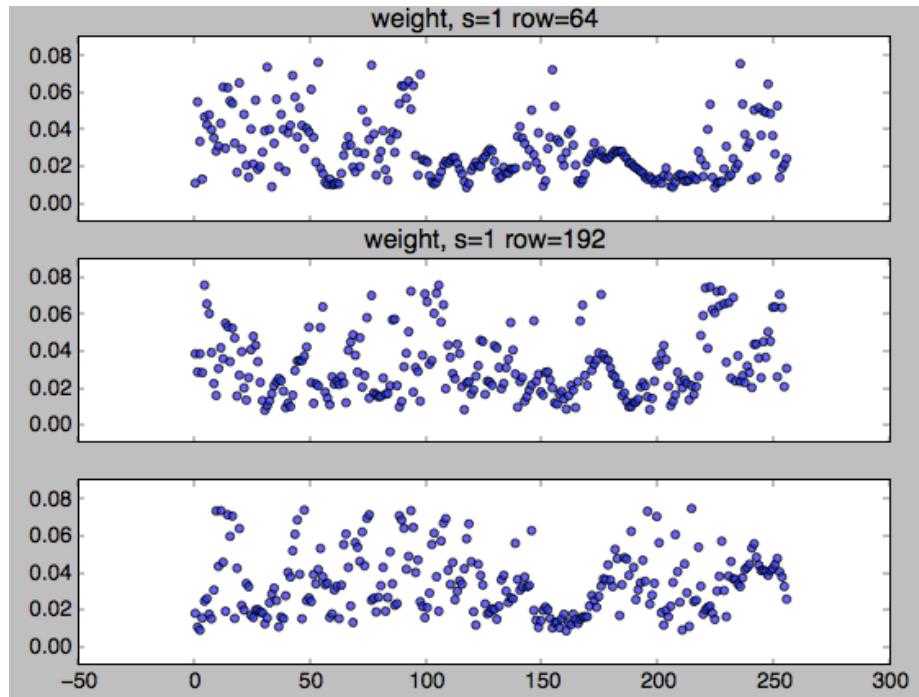
```
s=1, An = sqrt(f * f + h1 * h1 + h2 * h2)
(where f=ifft2(perFFT(img)*logGabor)
and h=ifft2(perFFT(img)*logGabor * H) )
```



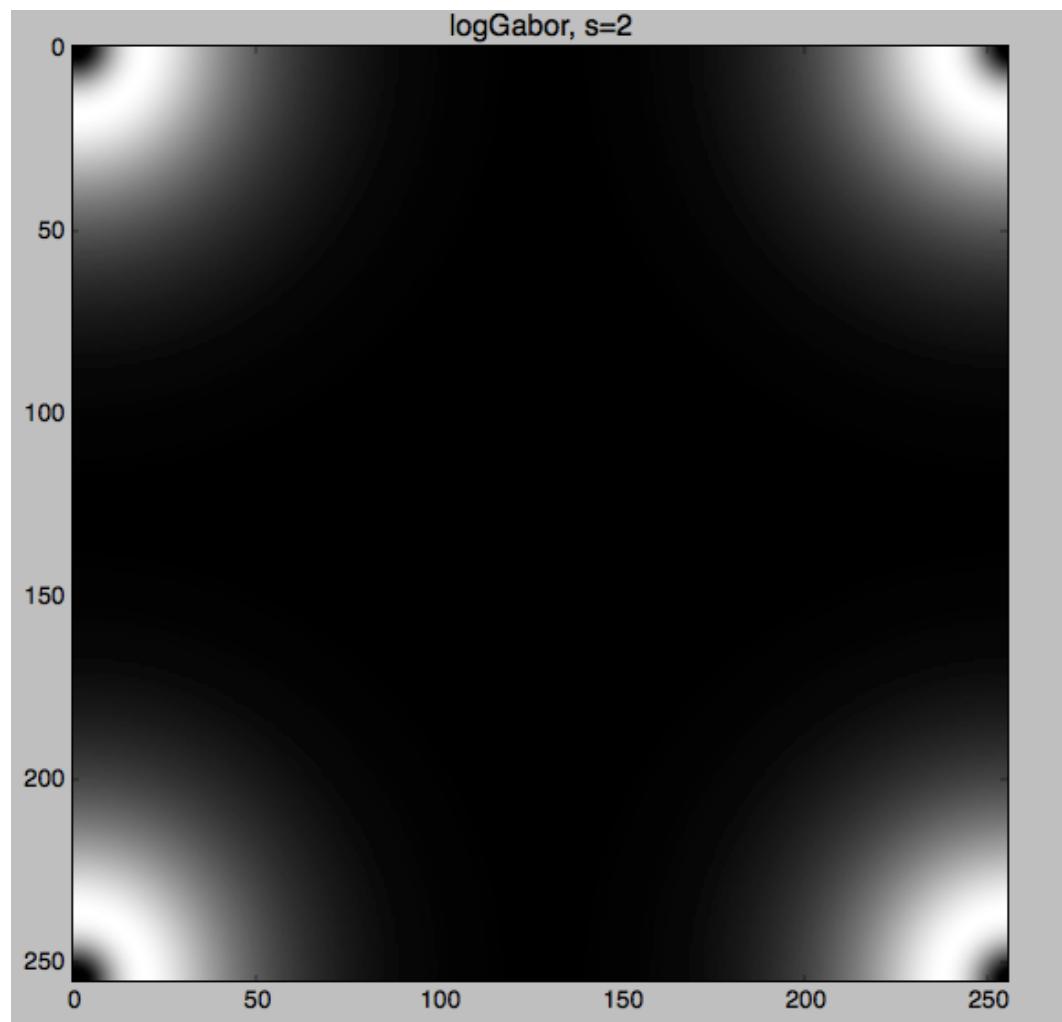
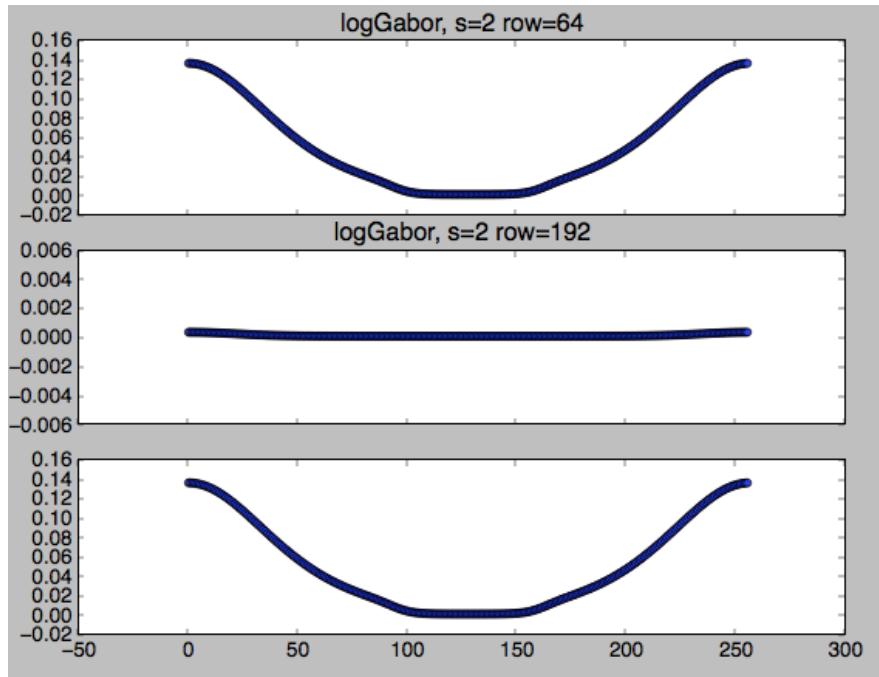
s=1, width = (sumAn / (maxAn + epsilon) - 1.) / (nscale - 1)



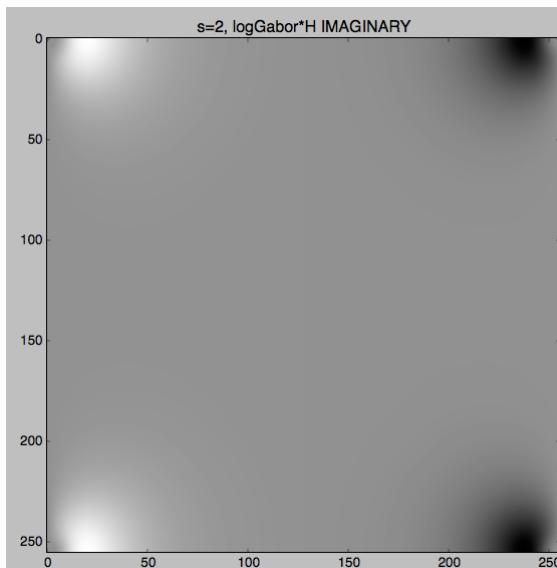
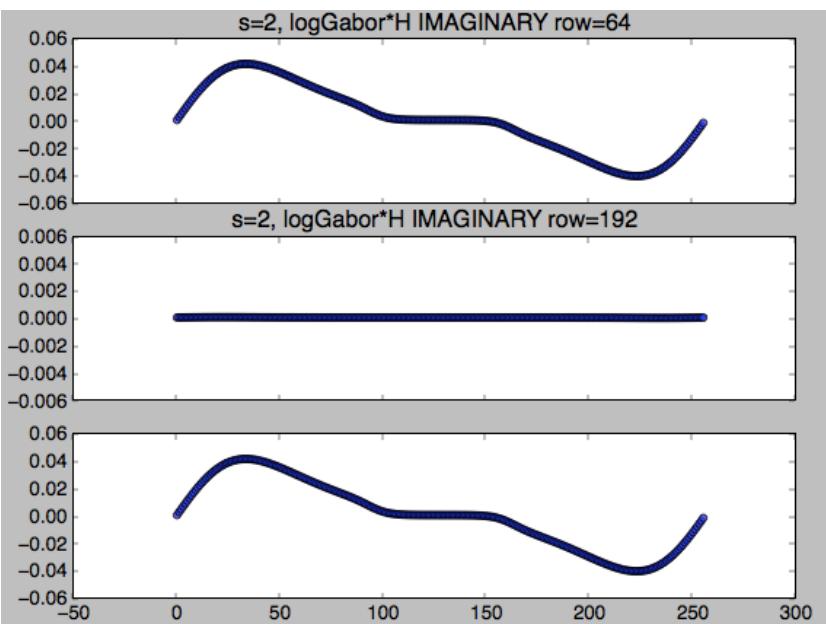
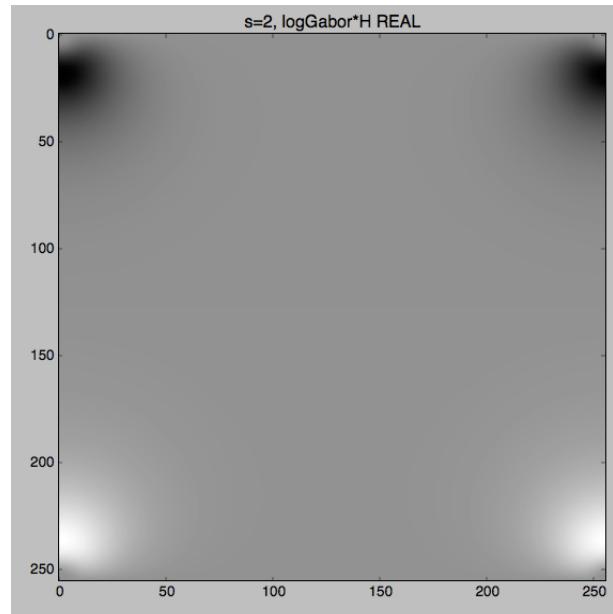
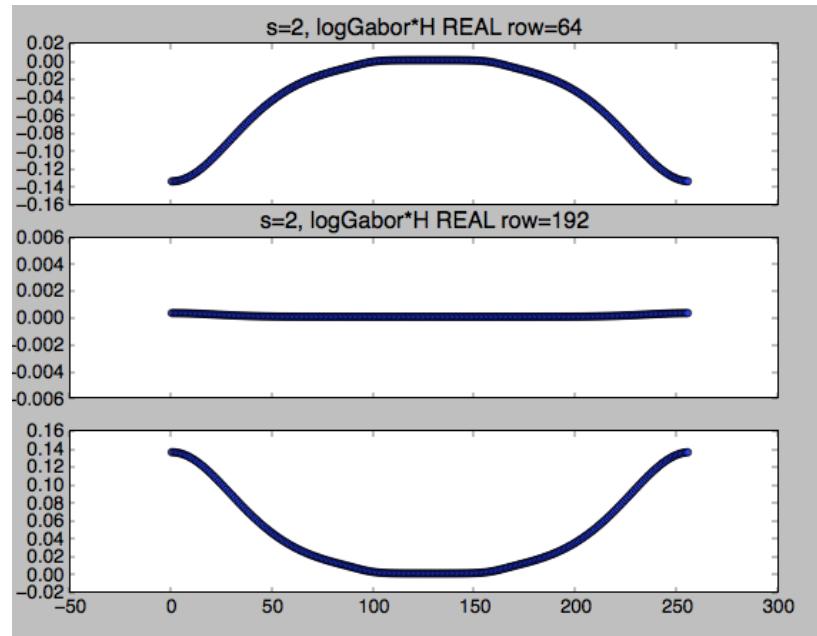
s=1, weight = $1. / (1. + \exp(g * (\text{cutoff} - \text{width})))$
(where cutoff=0.5, g=10.)



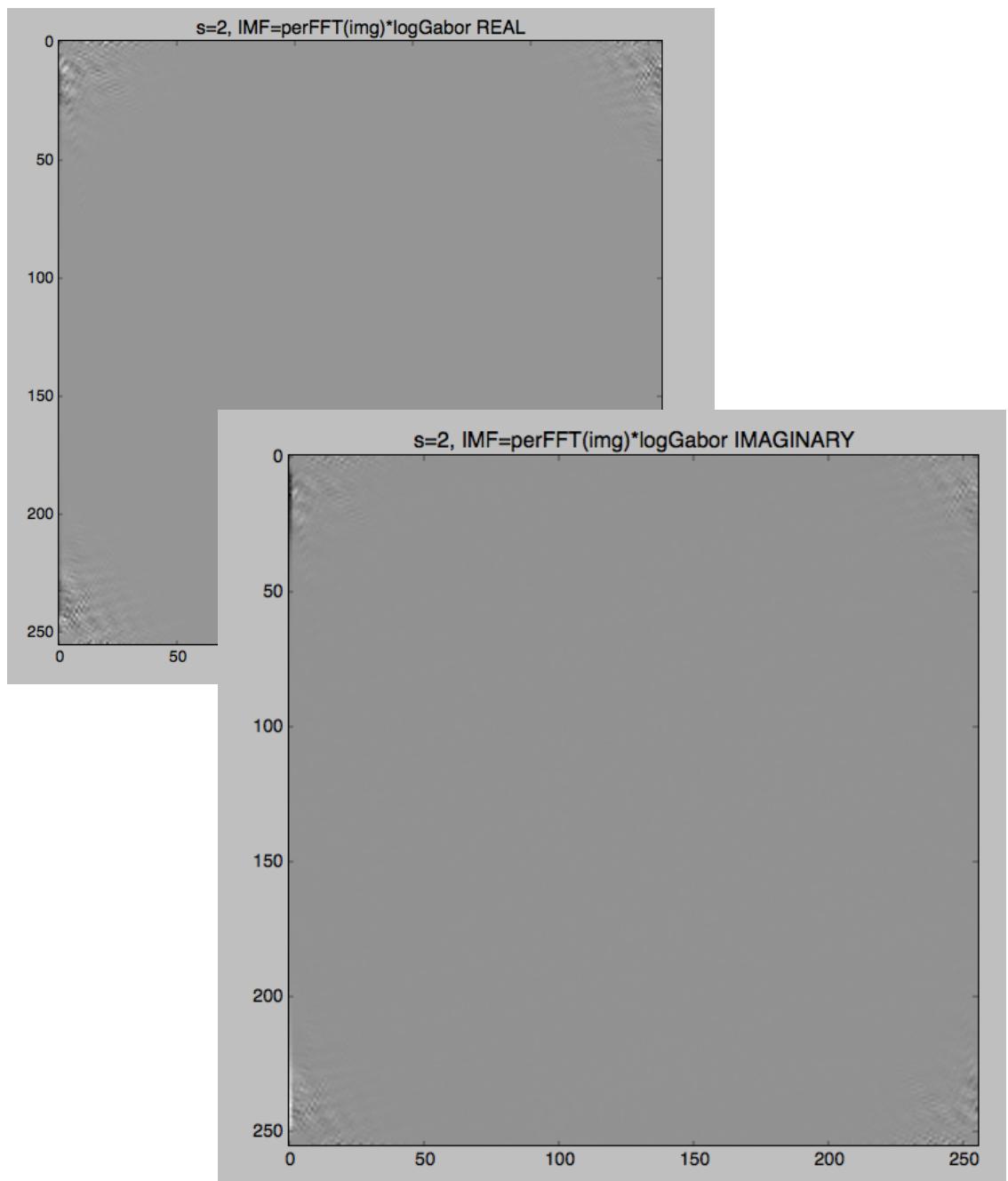
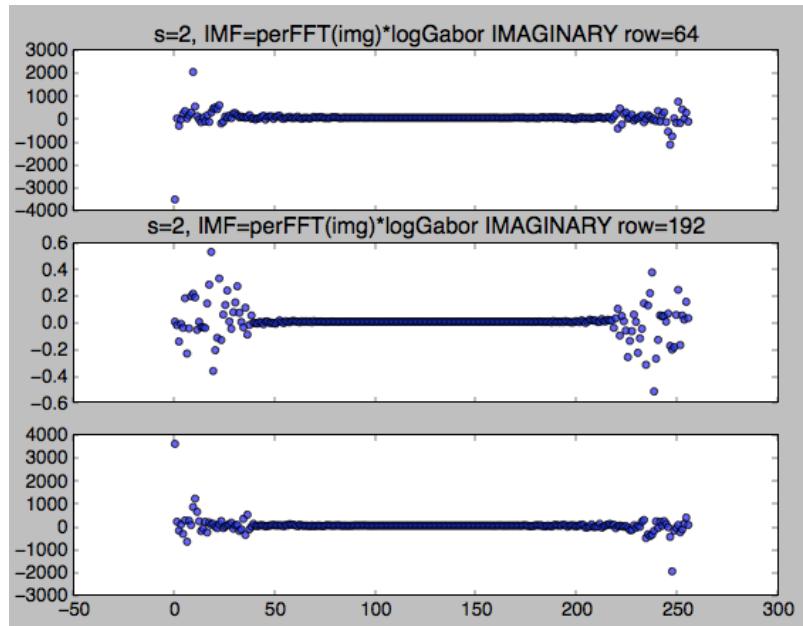
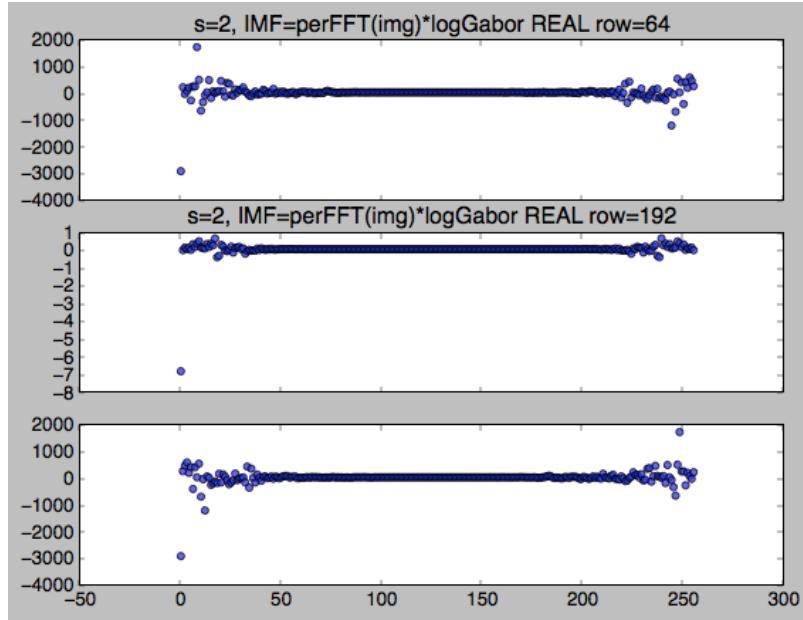
s=2, logGabor * low pass filter

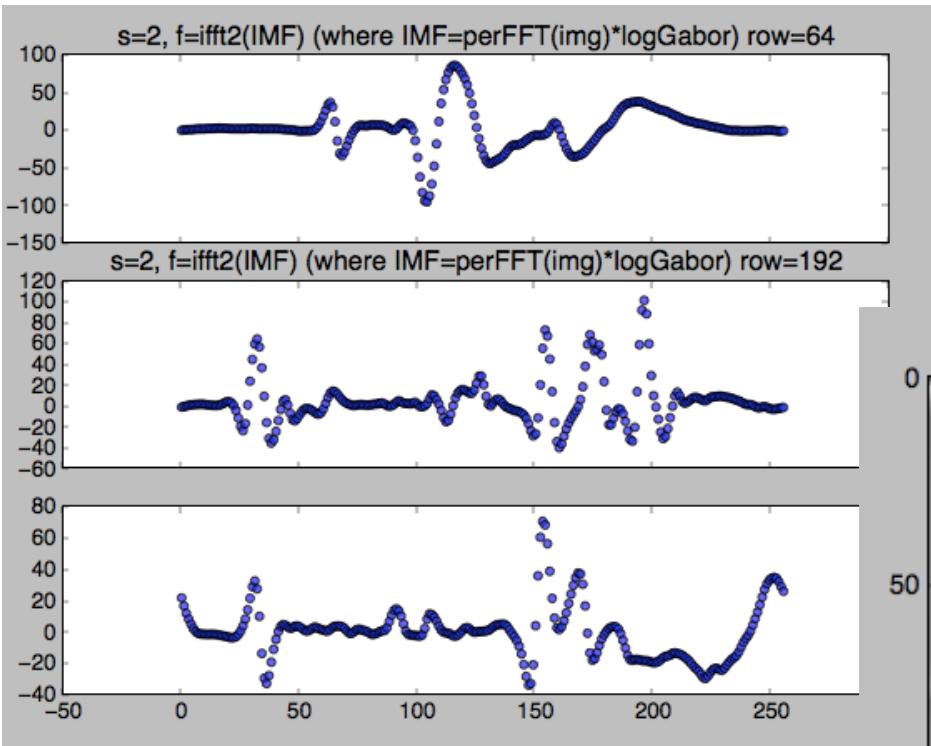


$s=2$, logGabor * low pass filter * H

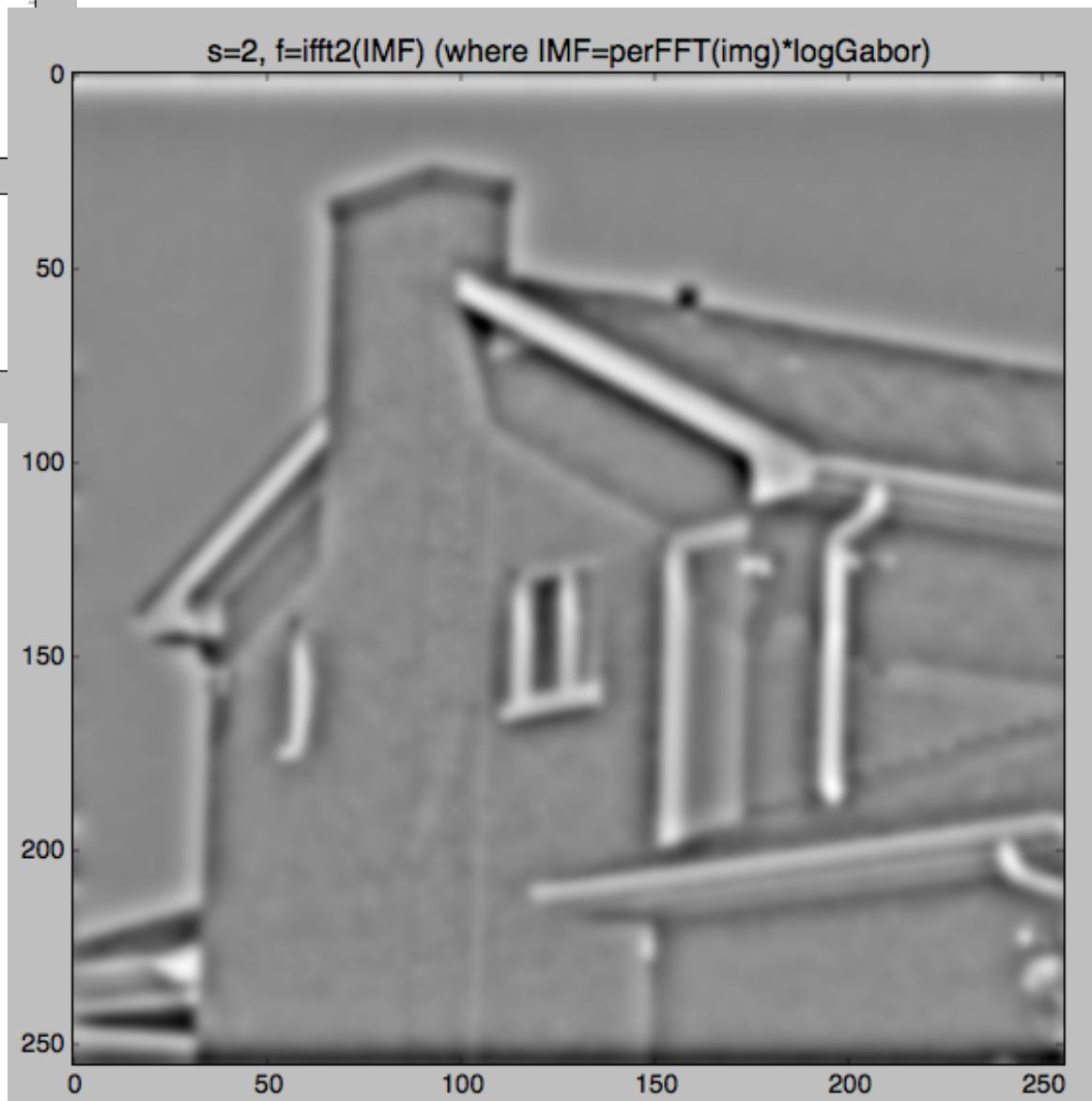


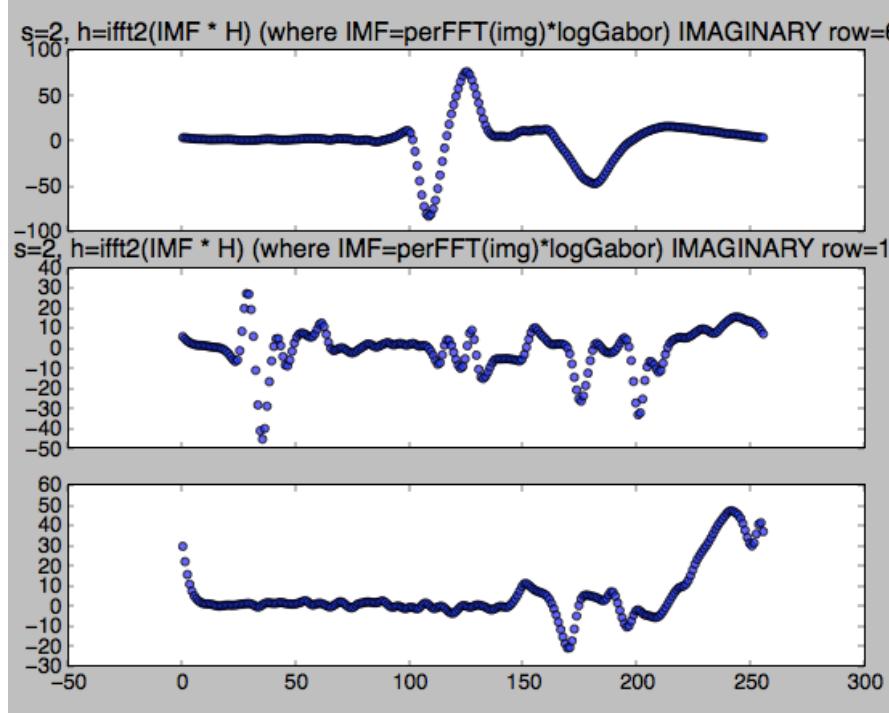
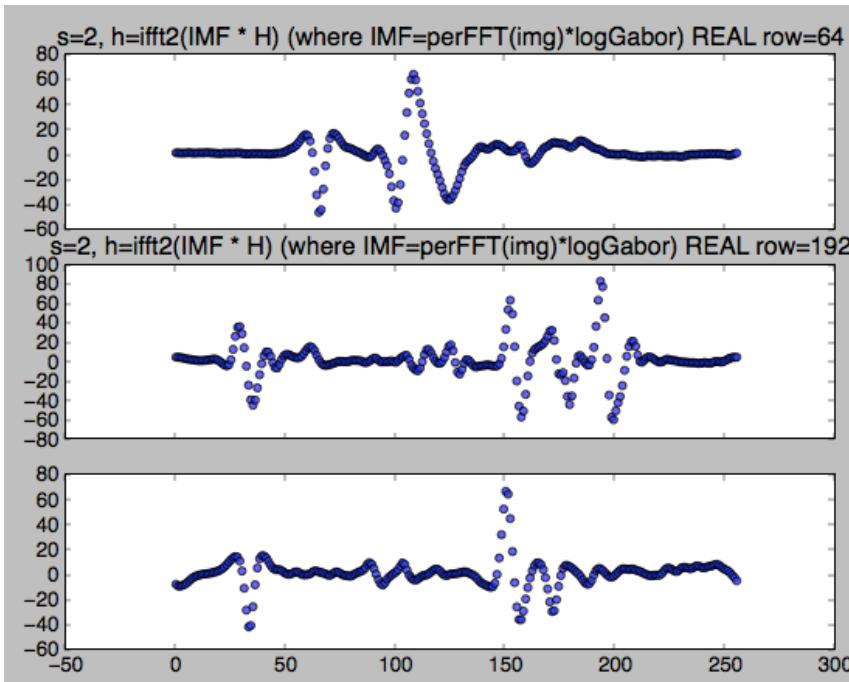
$s=2, \text{ IMF}=\text{perFFT(img)} * \text{logGabor}$



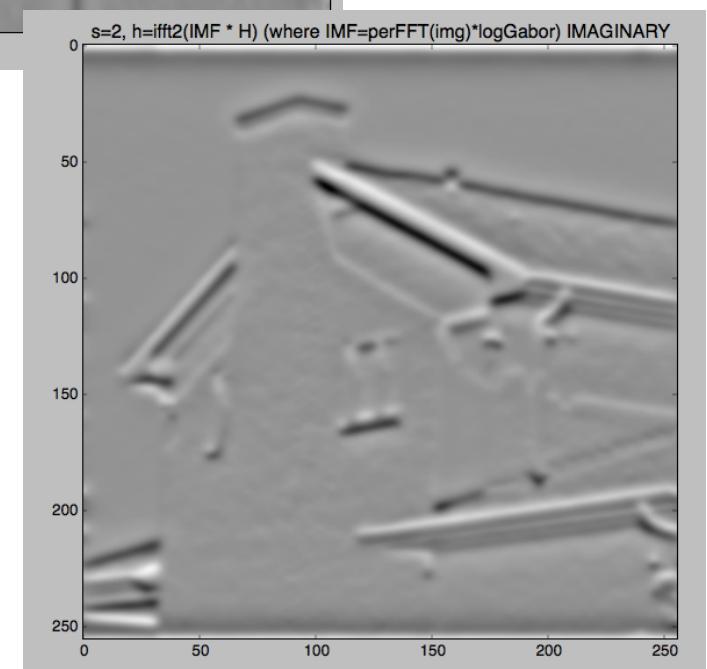
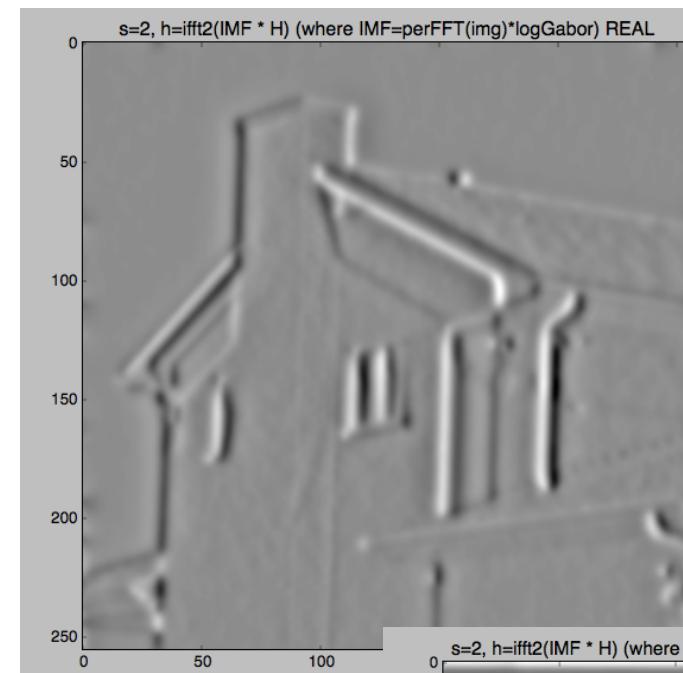


s=2, f=ifft2(IMF)
(where IMF=perFFT(img)*logGabor)

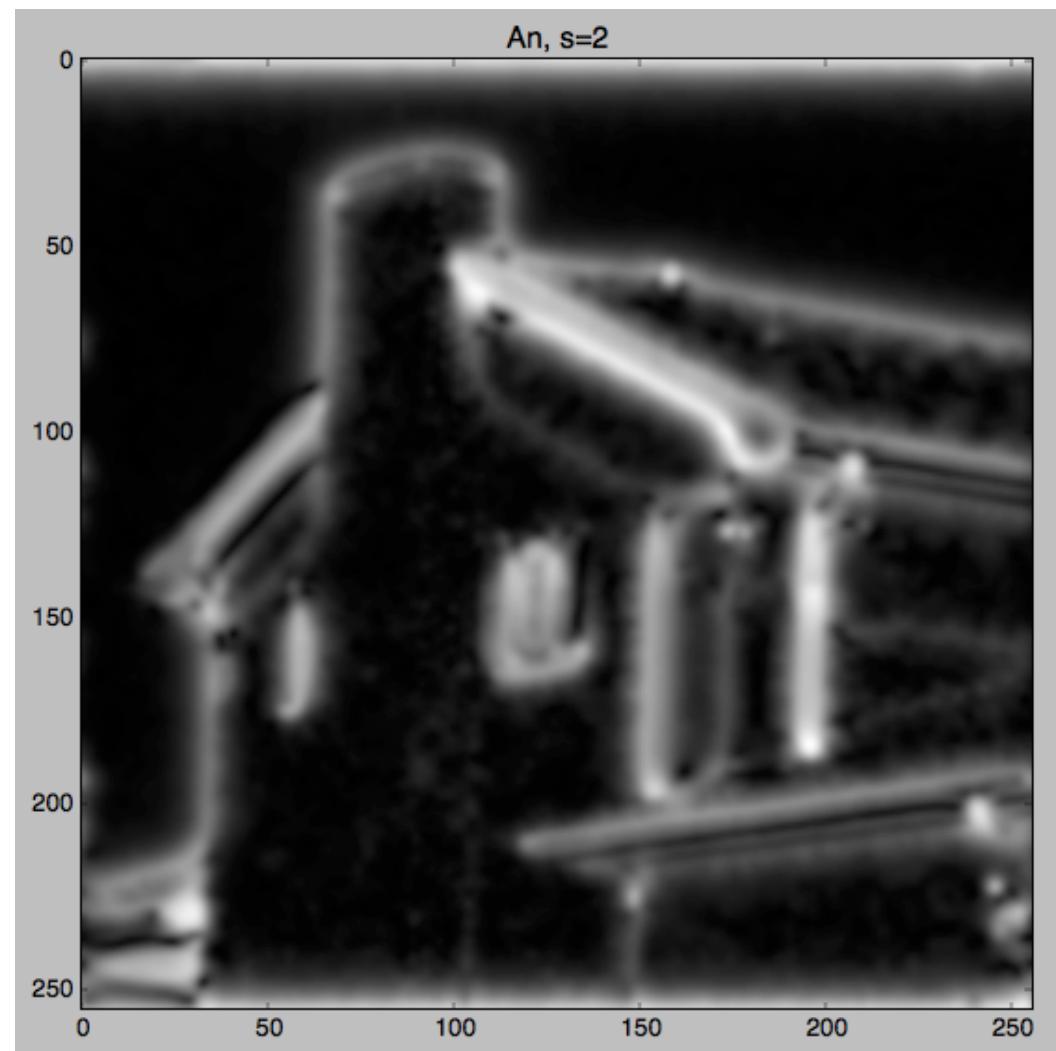
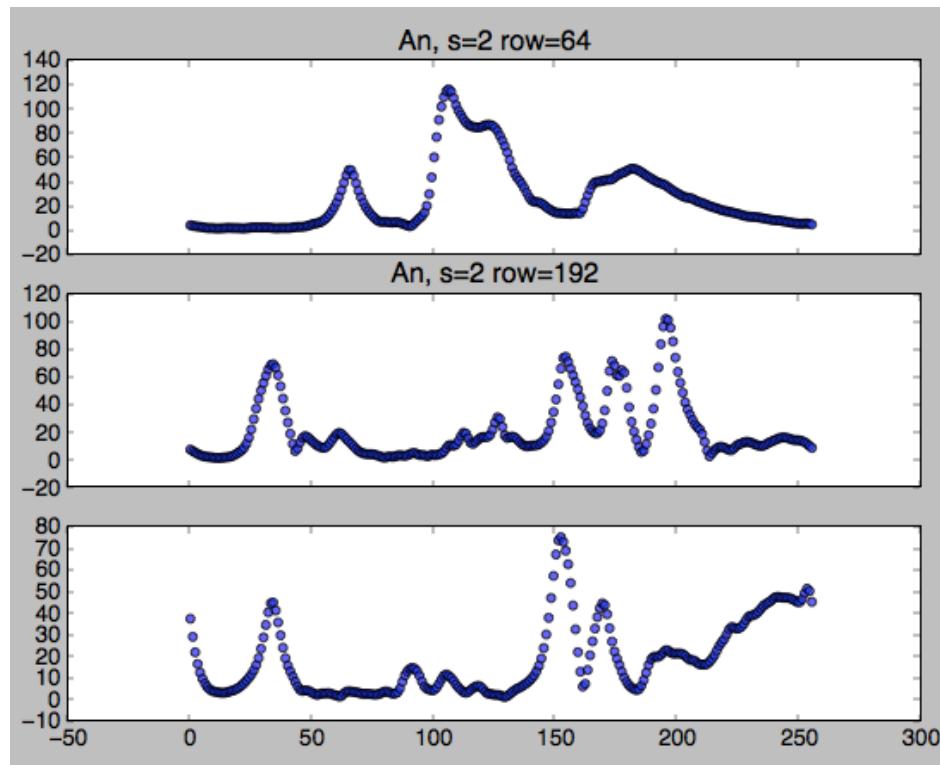




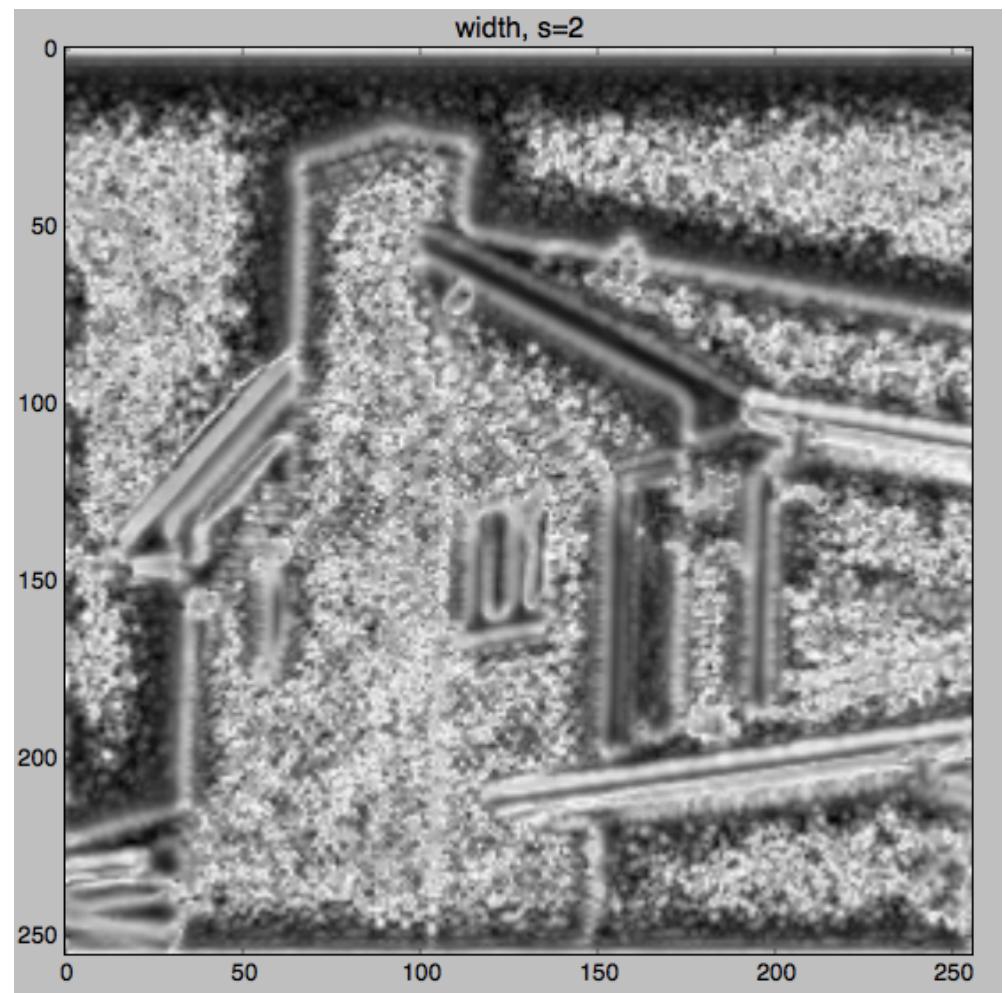
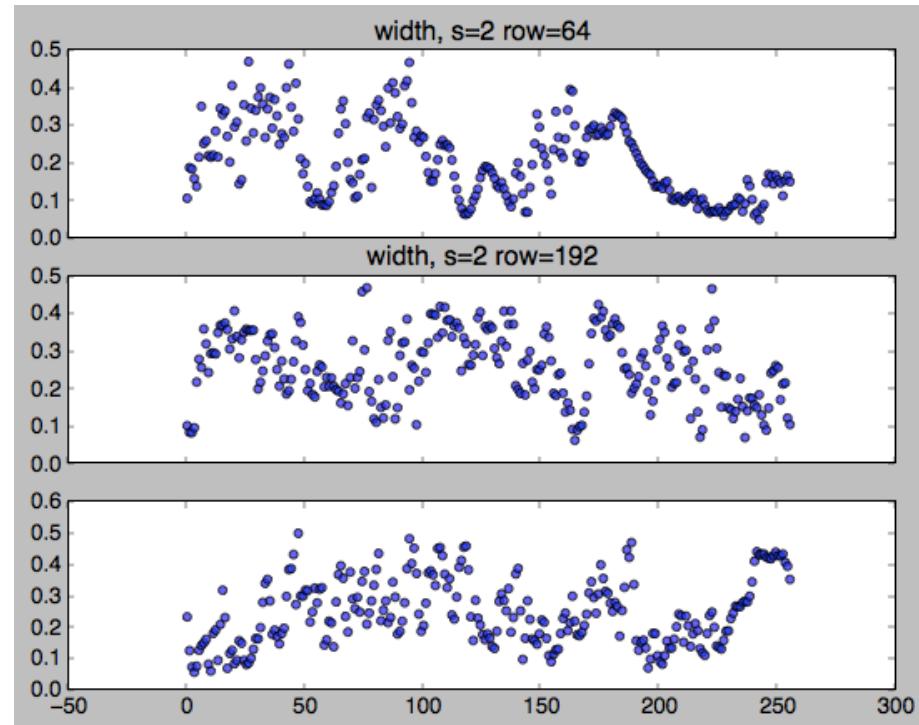
**s=2, h=ifft2(IMF * H)
(where IMF=perFFT(img)*logGabor)**



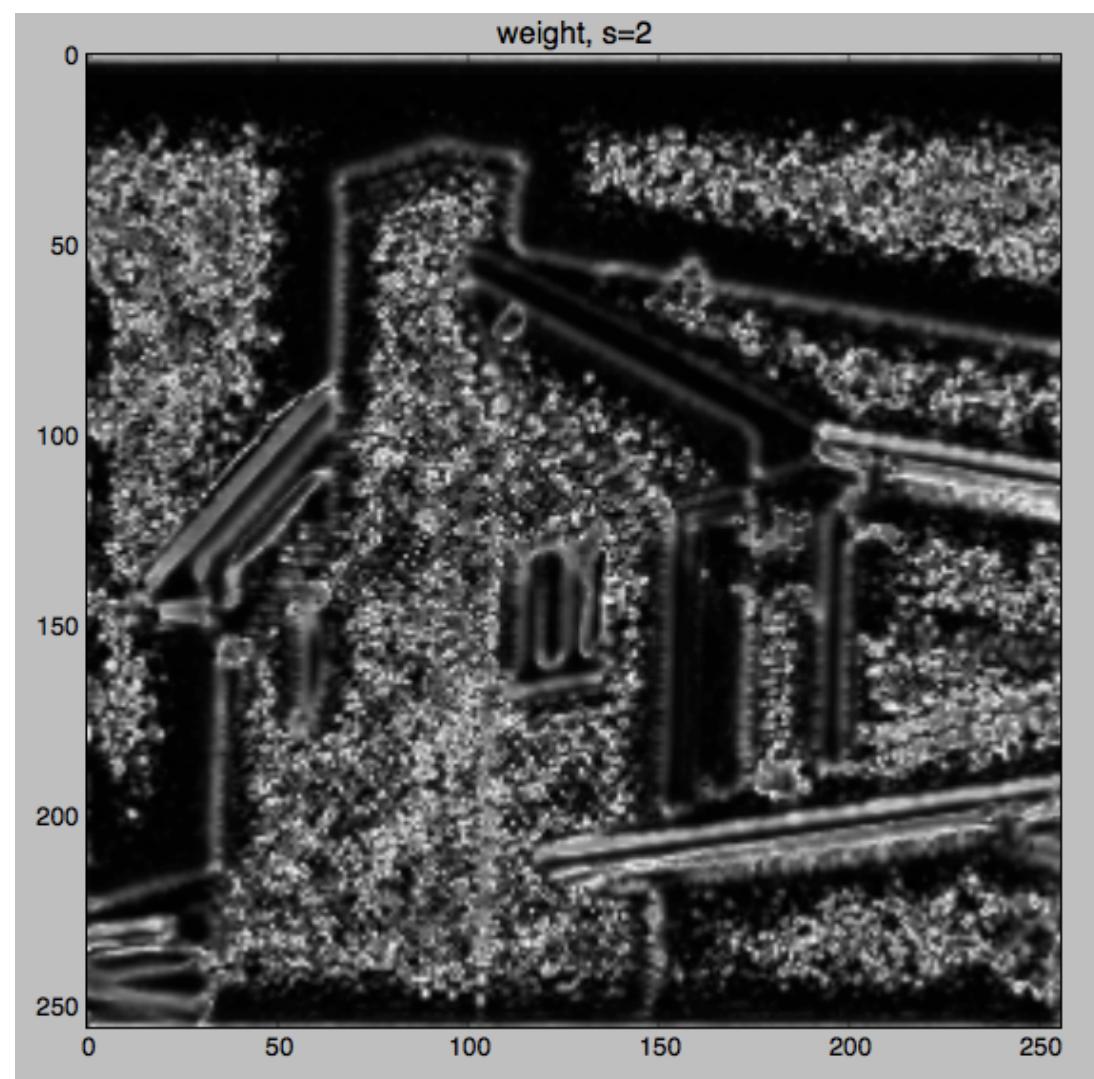
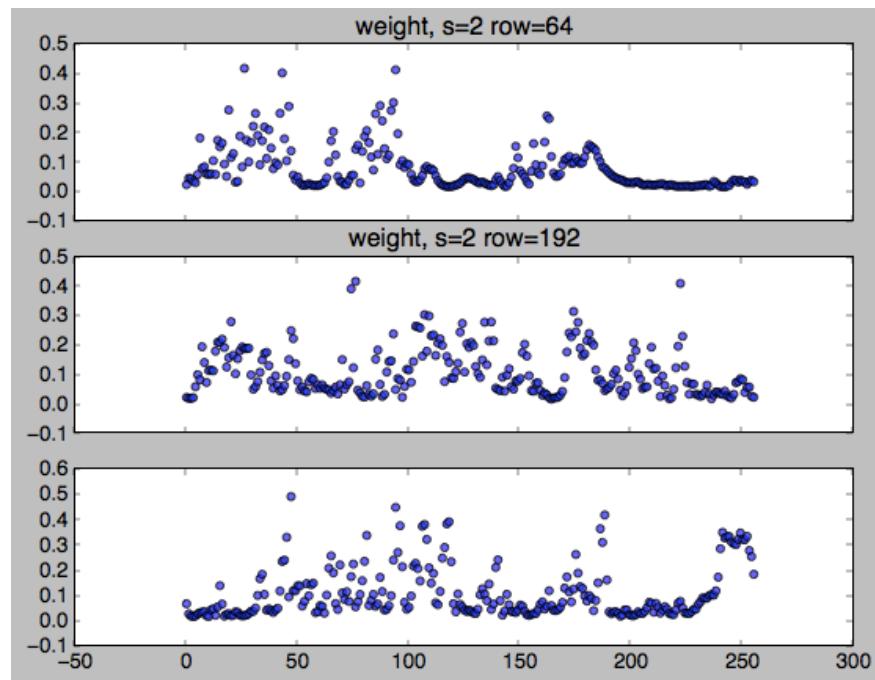
```
s=2, An = sqrt(f * f + h1 * h1 + h2 * h2)
(where f=ifft2(perFFT(img)*logGabor)
and h=ifft2(perFFT(img)*logGabor * H) )
```



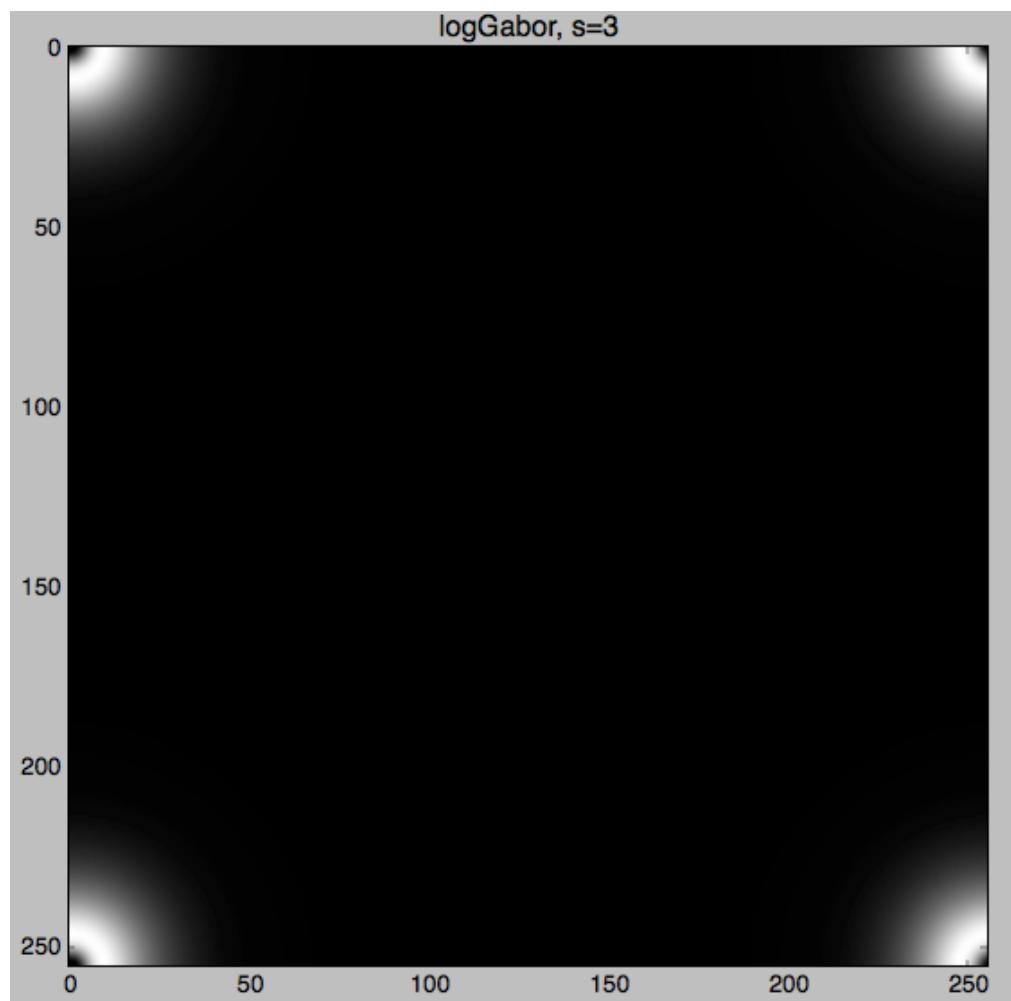
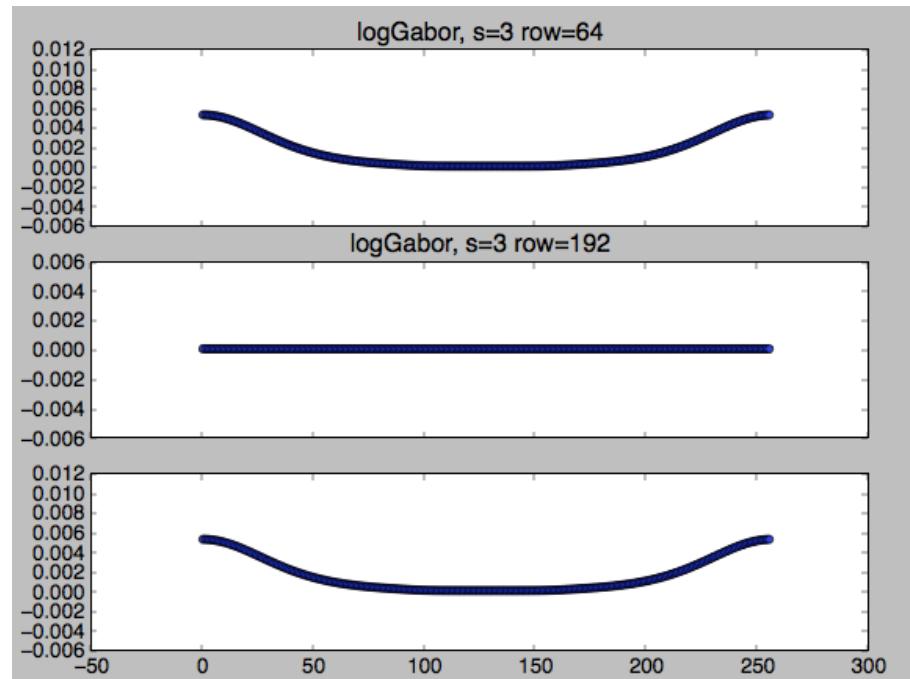
s=2, width = (sumAn / (maxAn + epsilon) - 1.) / (nscale - 1)



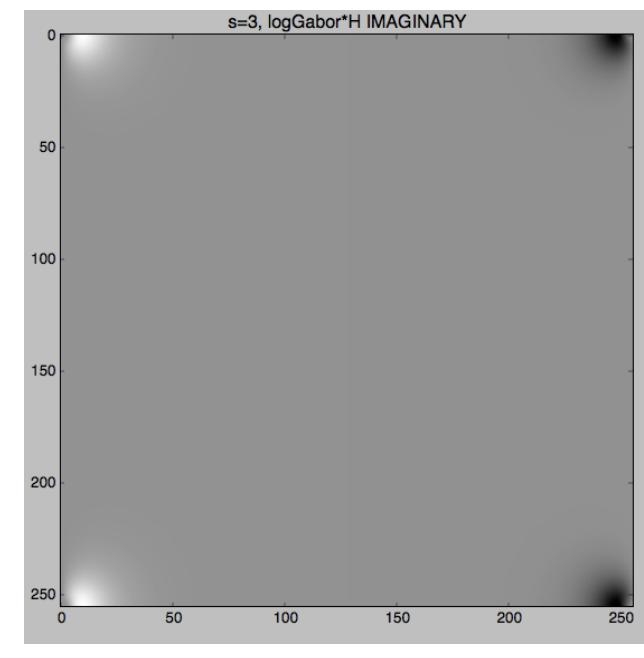
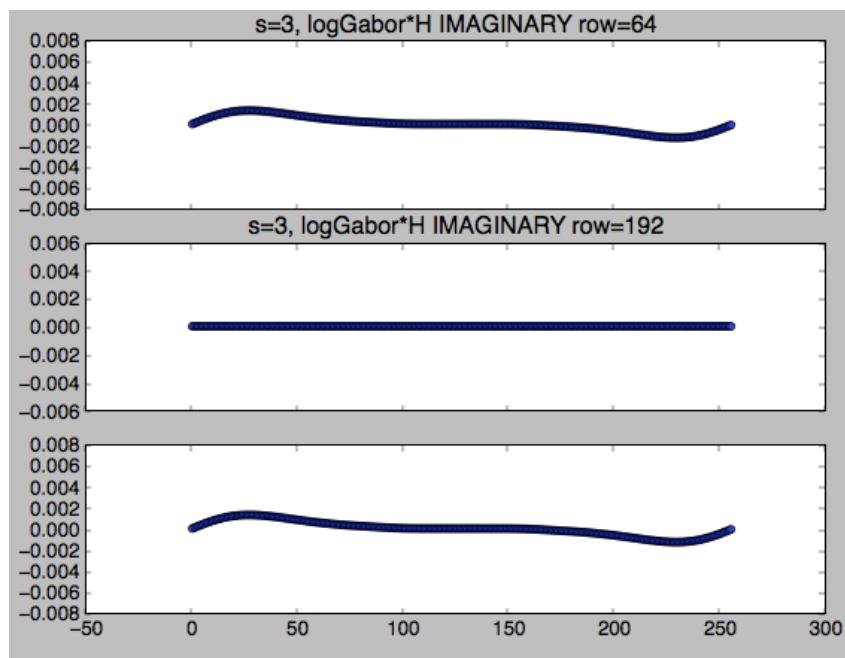
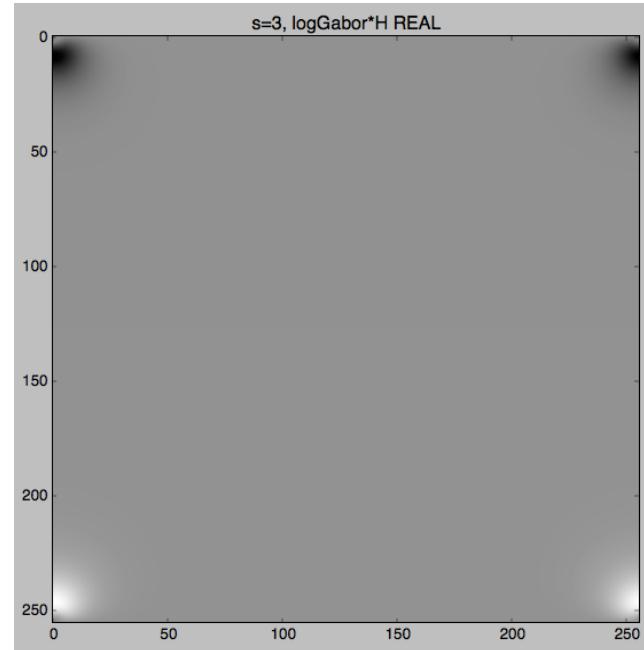
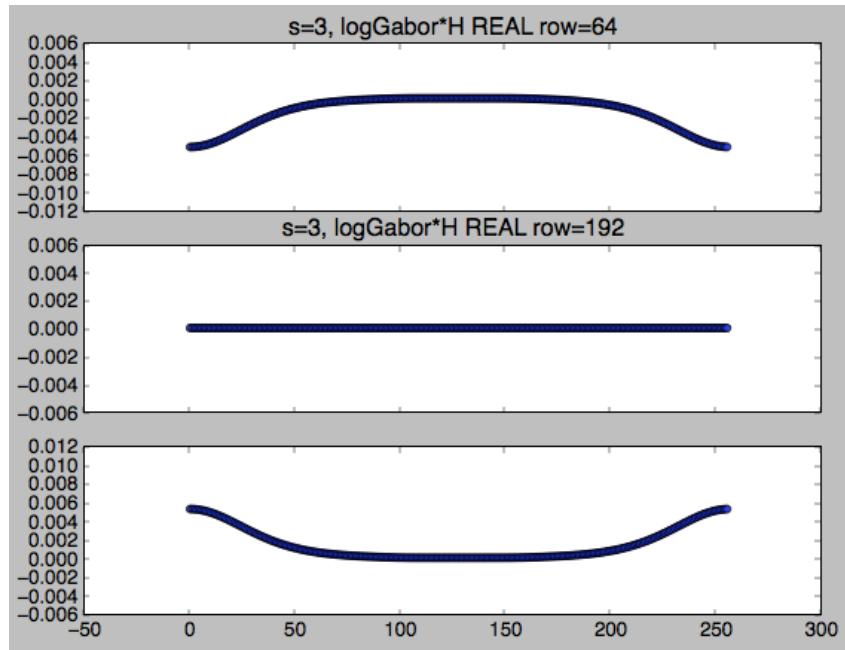
s=2, weight = $1. / (1. + \exp(g * (\text{cutoff} - \text{width})))$
(where cutoff=0.5, g=10.)



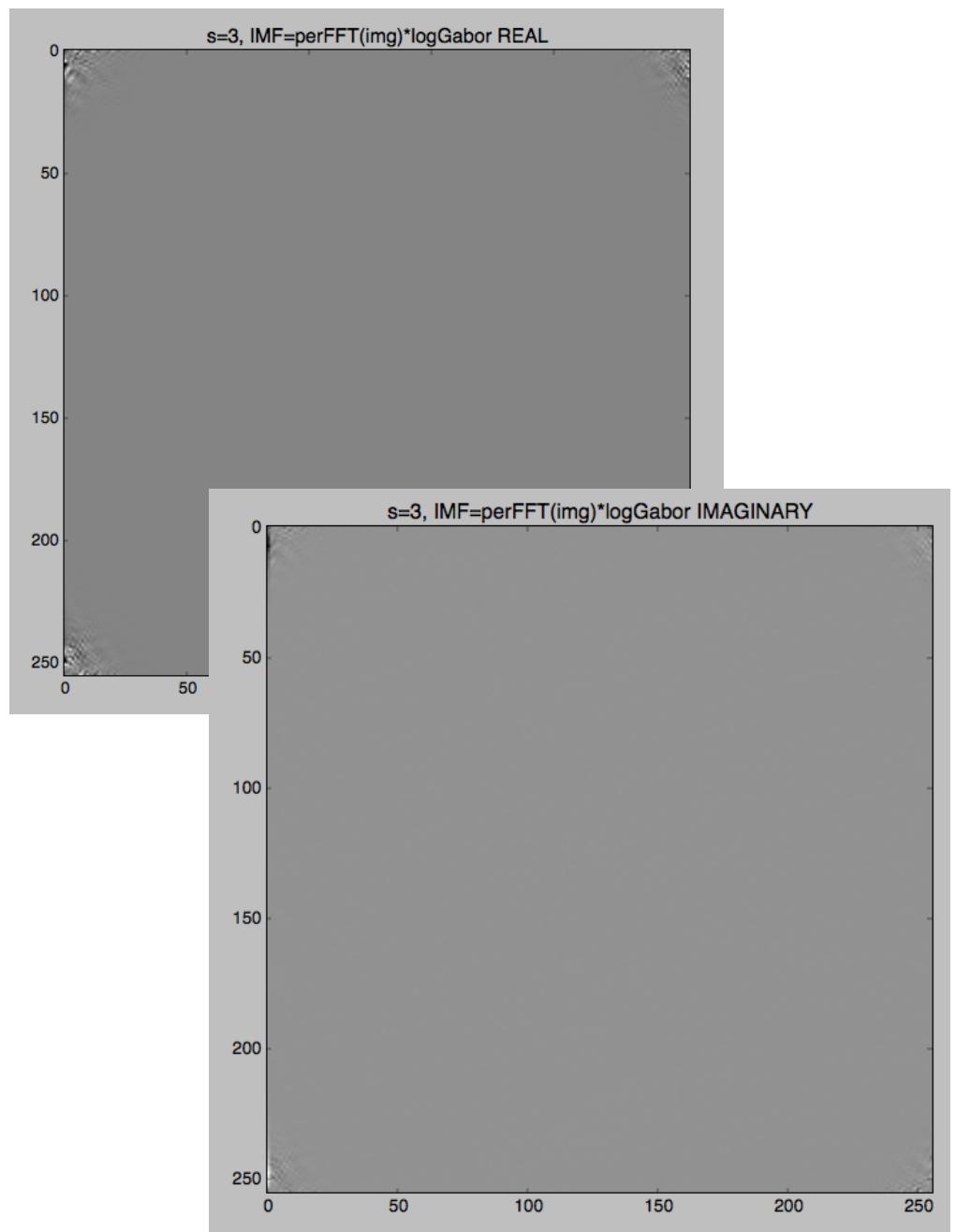
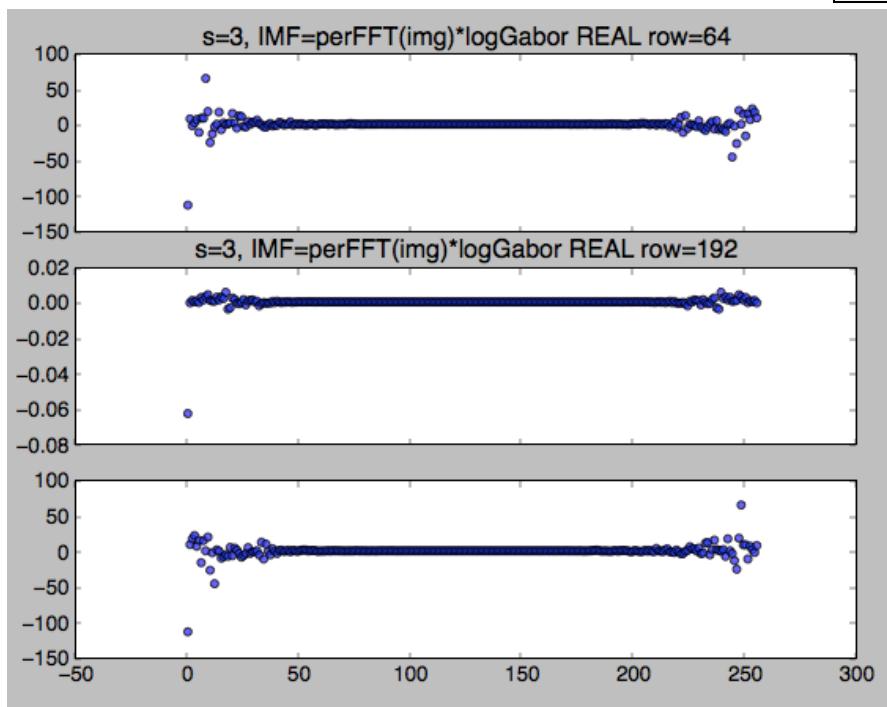
$s=3$, logGabor * low pass filter

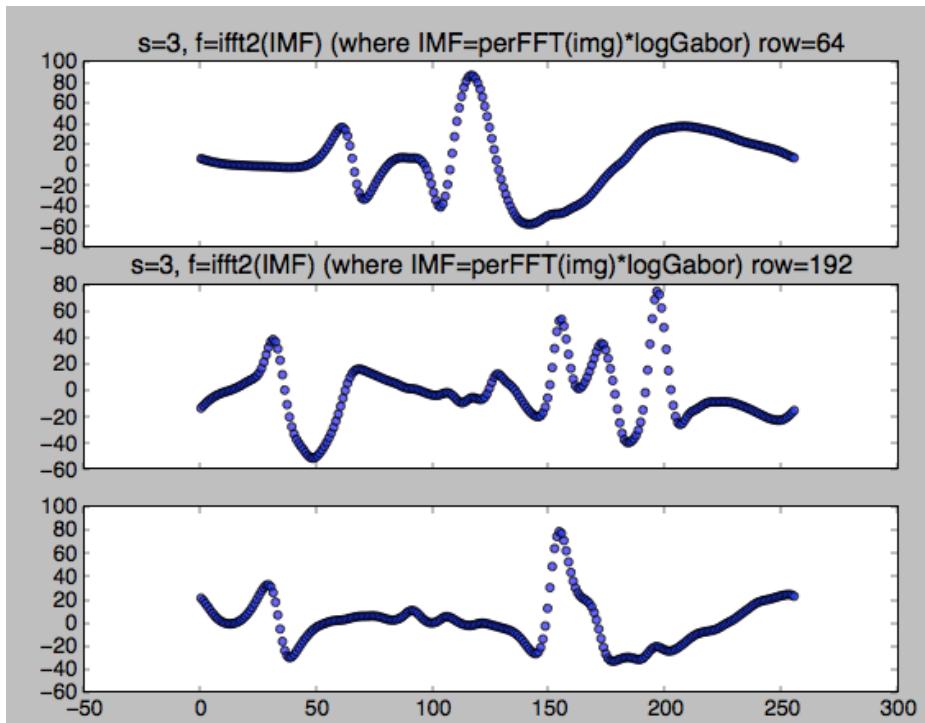


$s=3$, logGabor * low pass filter * H

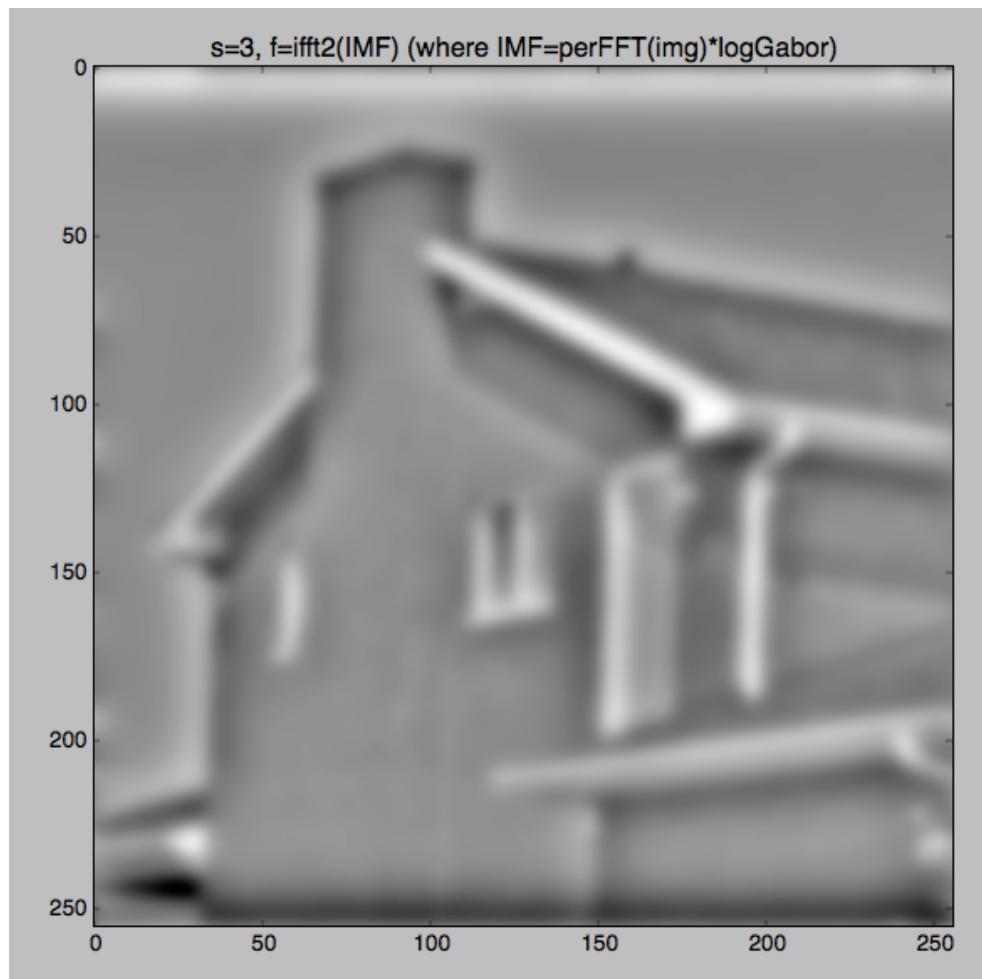


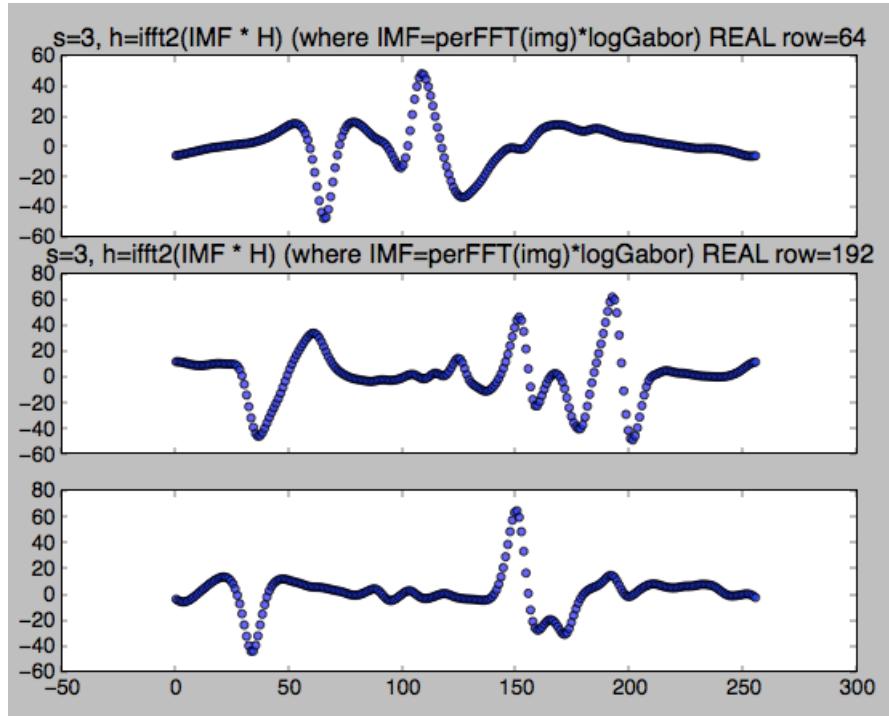
$s=3$, $\text{IMF}=\text{perFFT}(\text{img}) * \text{logGabor}$



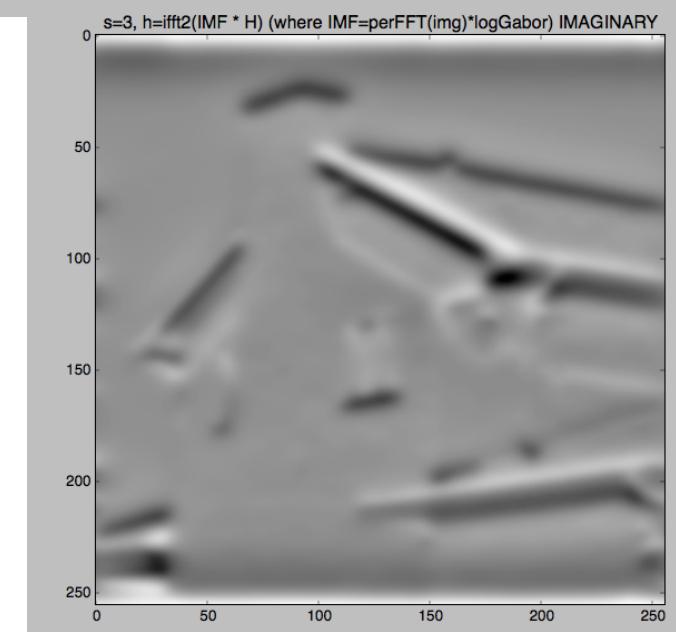
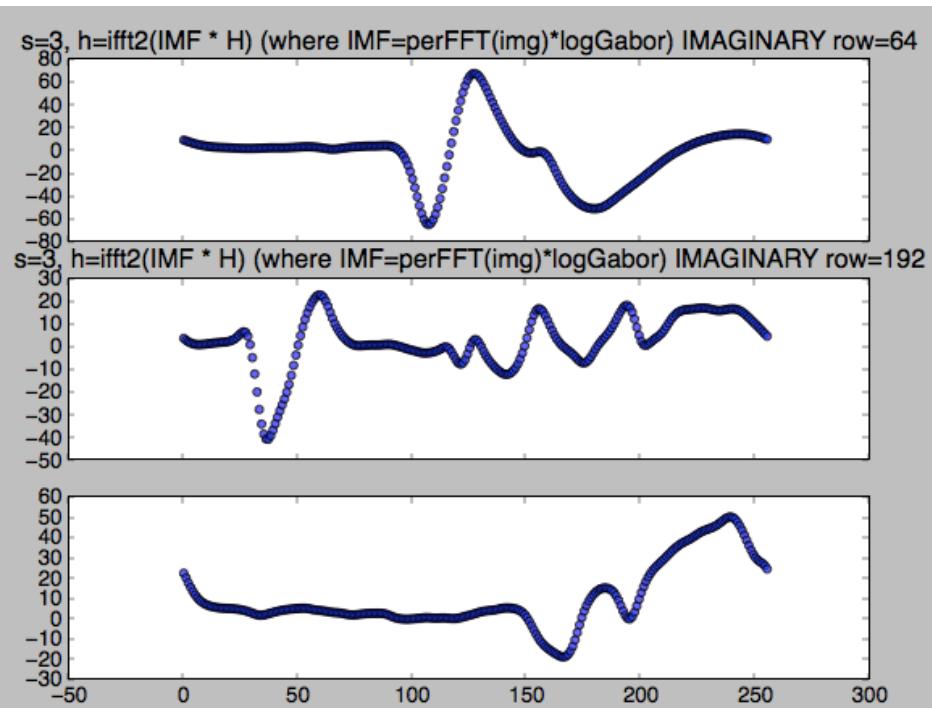
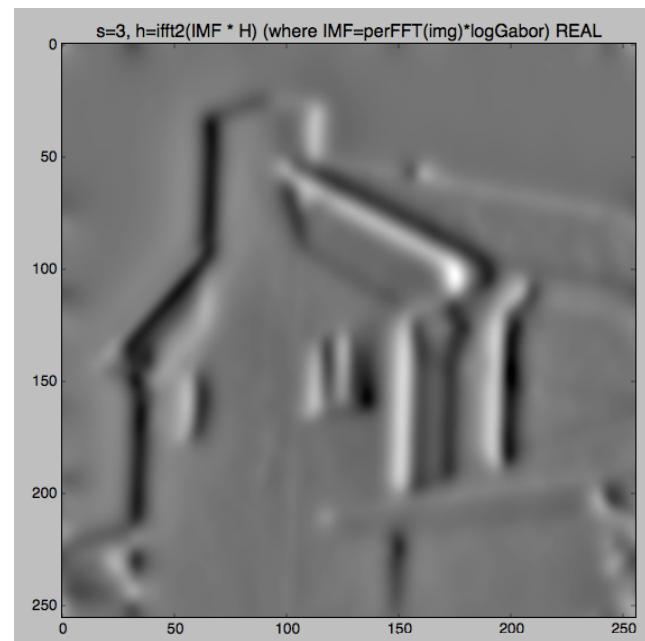


s=3, f=ifft2(IMF)
(where IMF=perFFT(img)*logGabor)





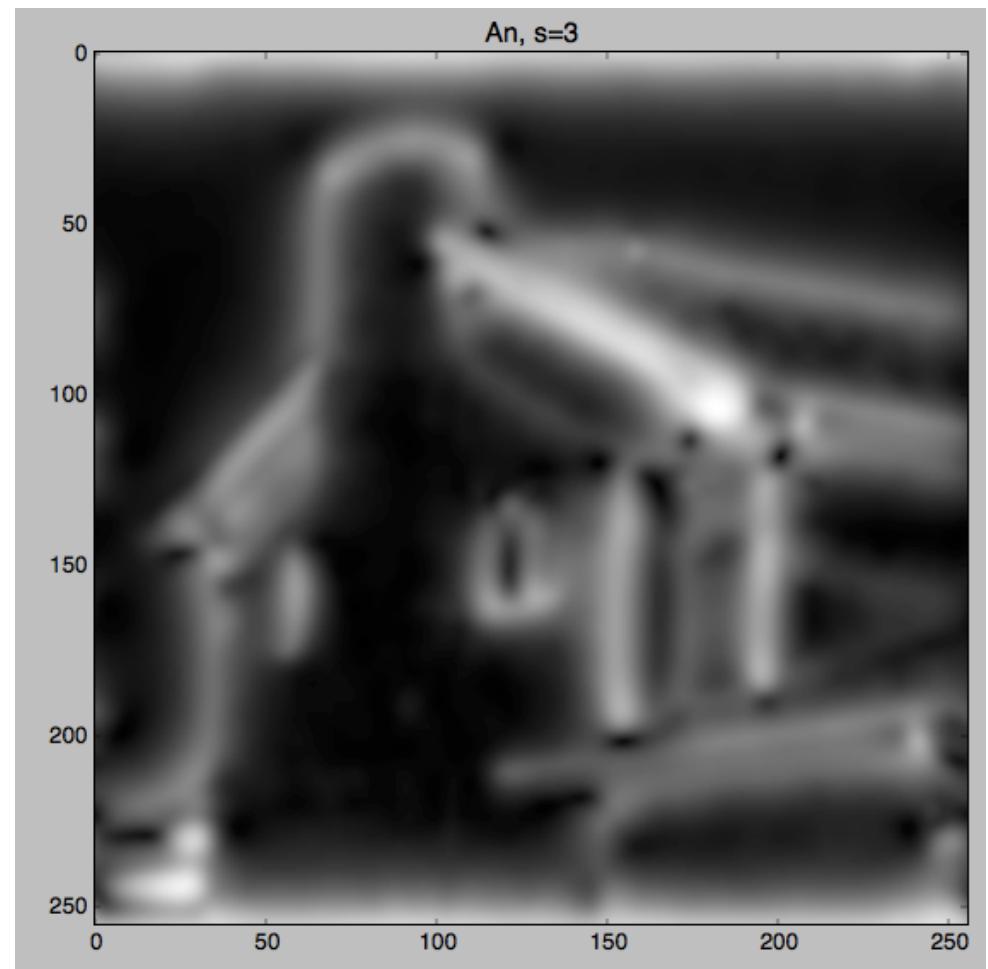
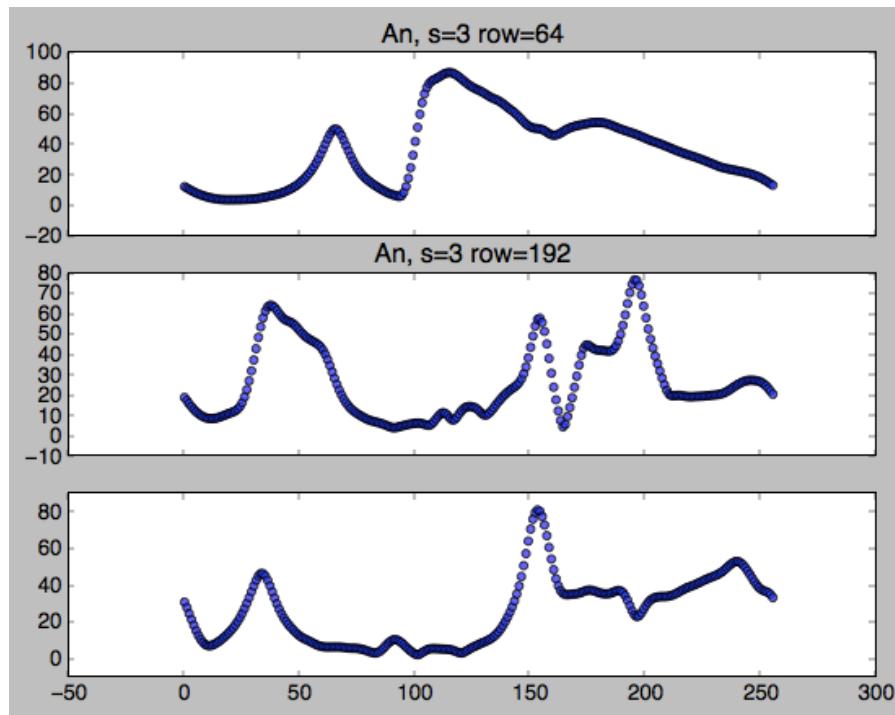
**s=3, h=ifft2(IMF * H)
(where IMF=perFFT(img)*logGabor)**



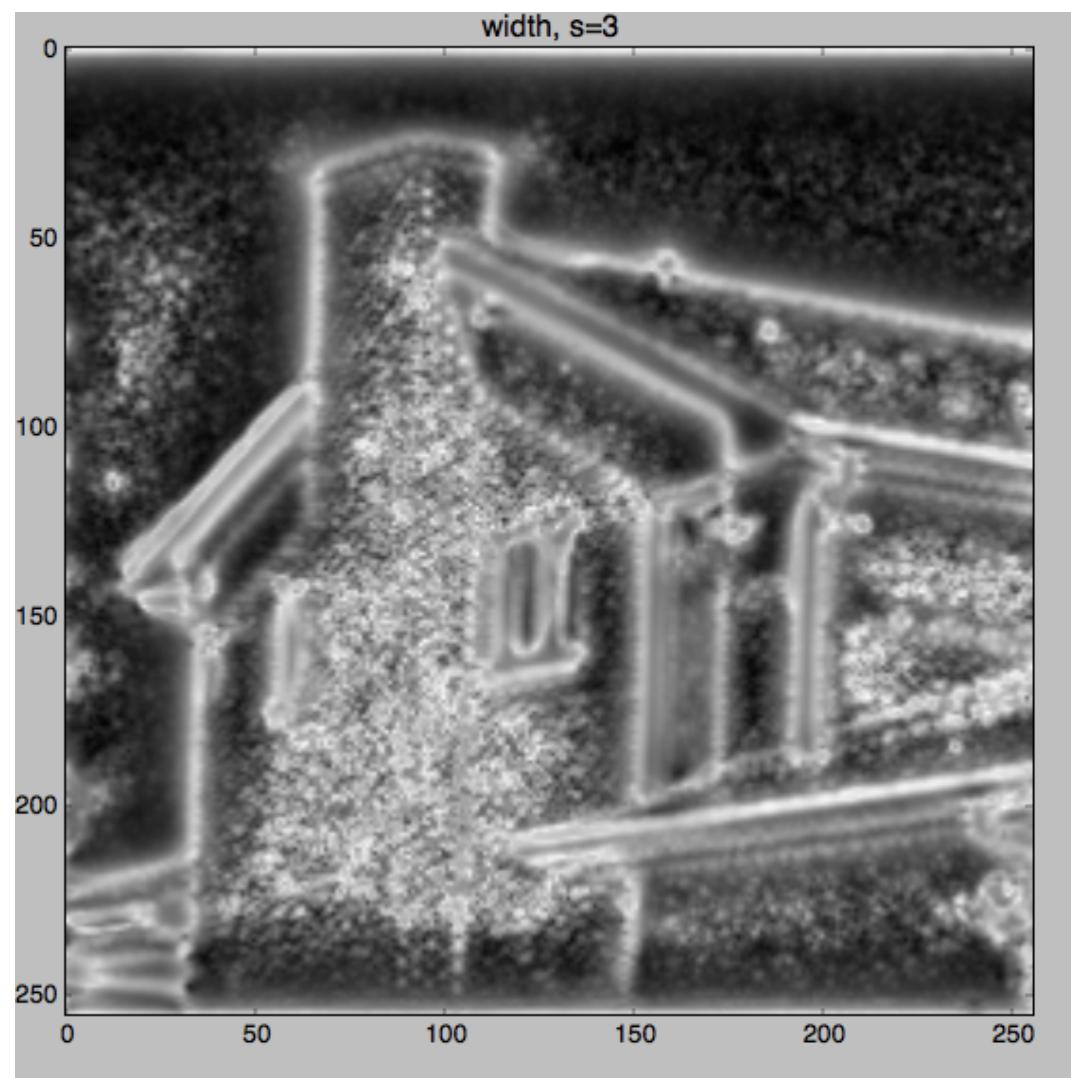
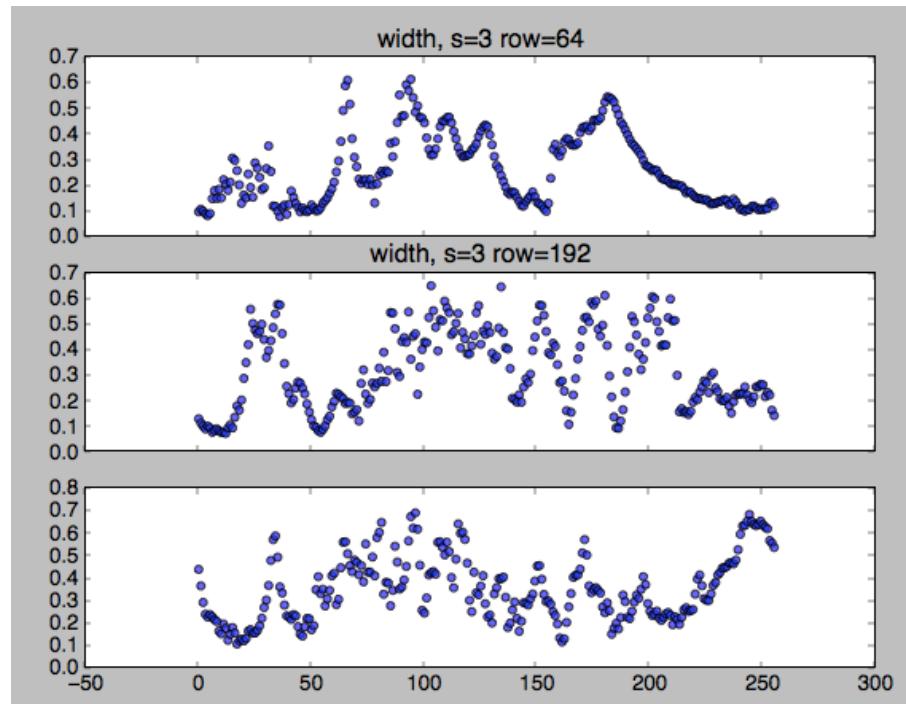
```

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

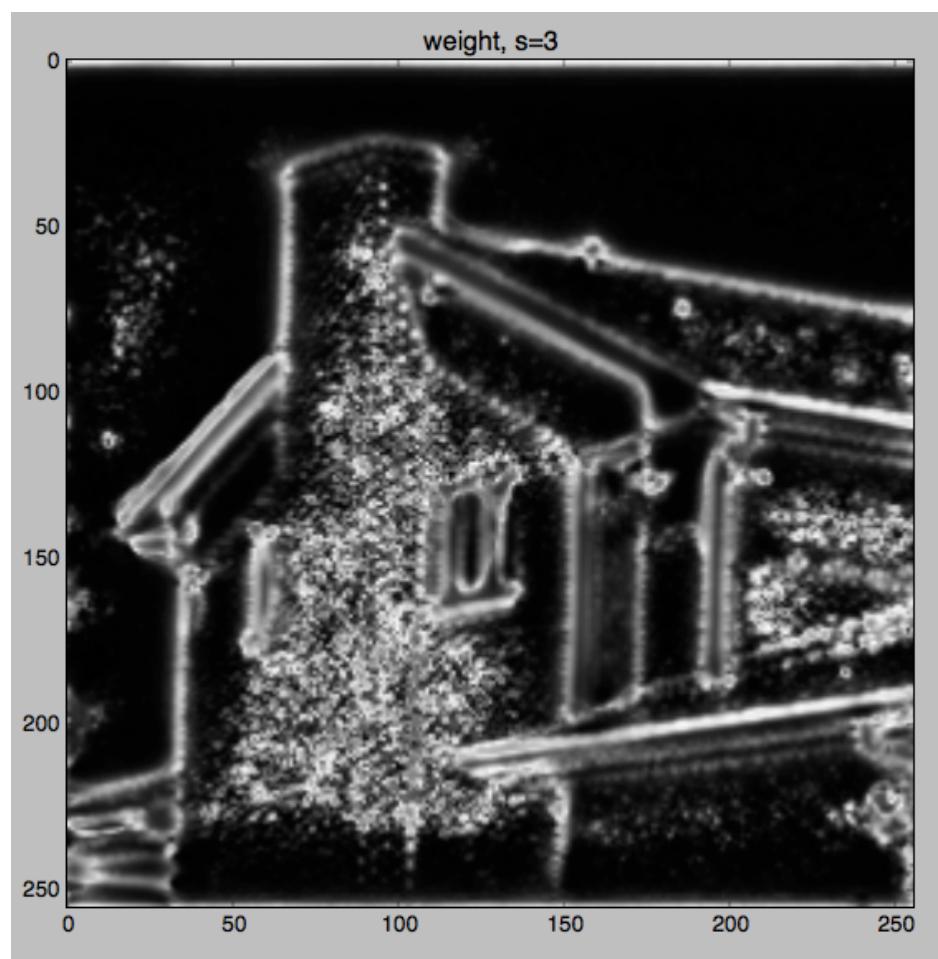
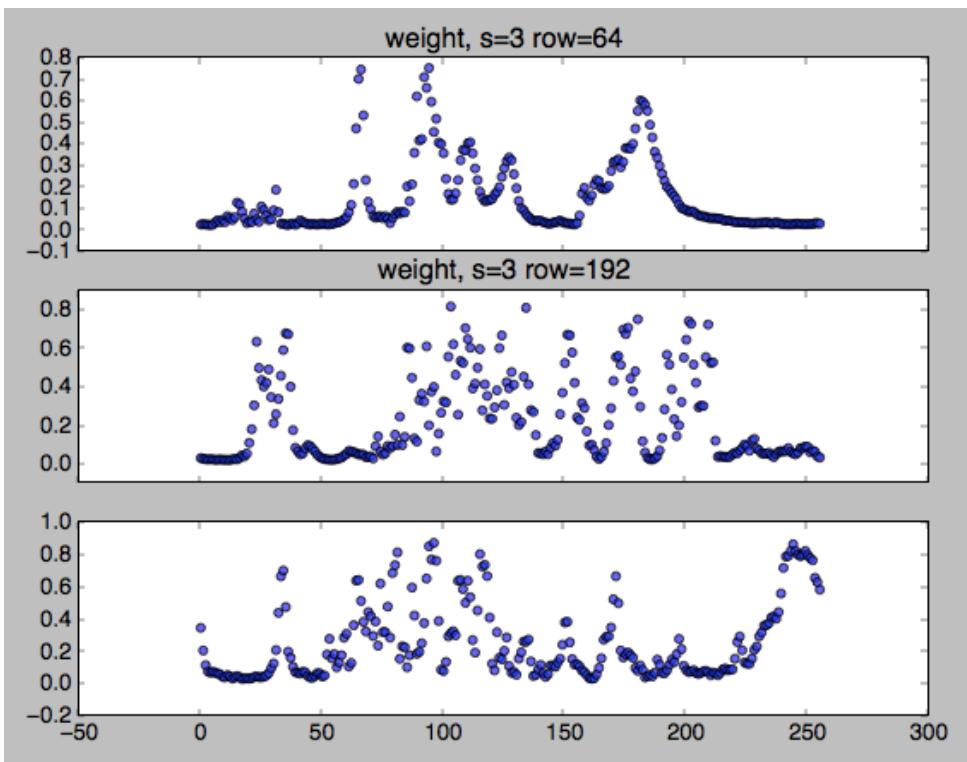
```



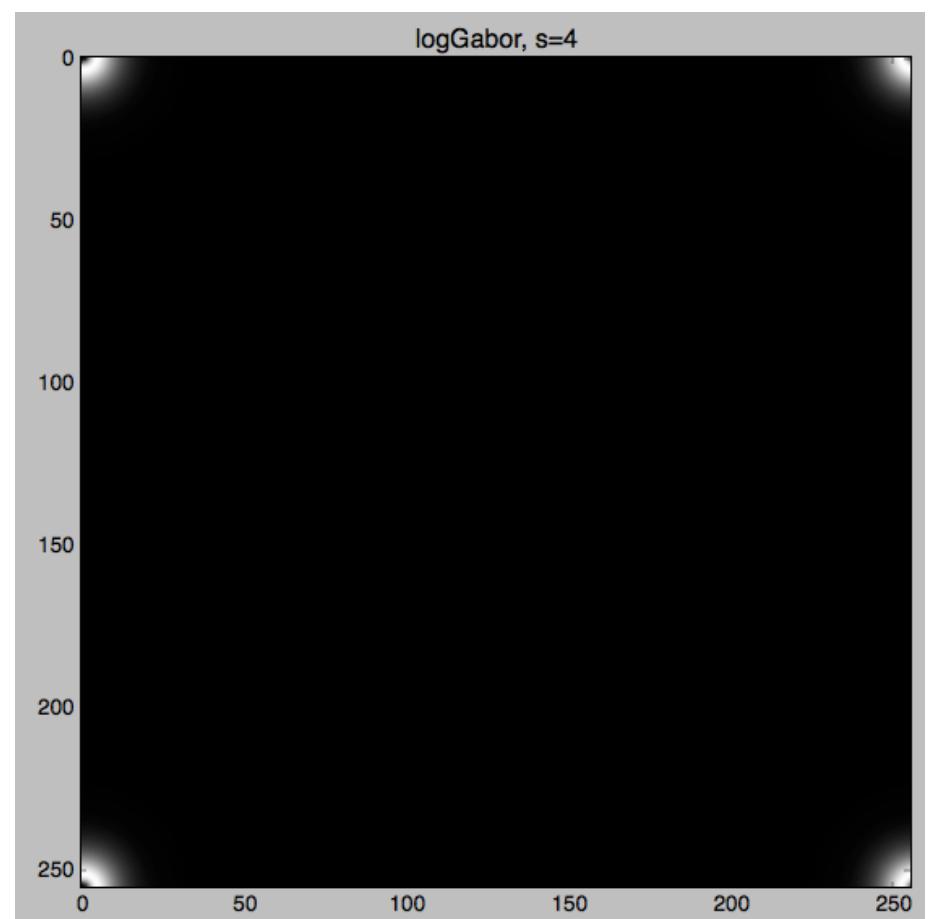
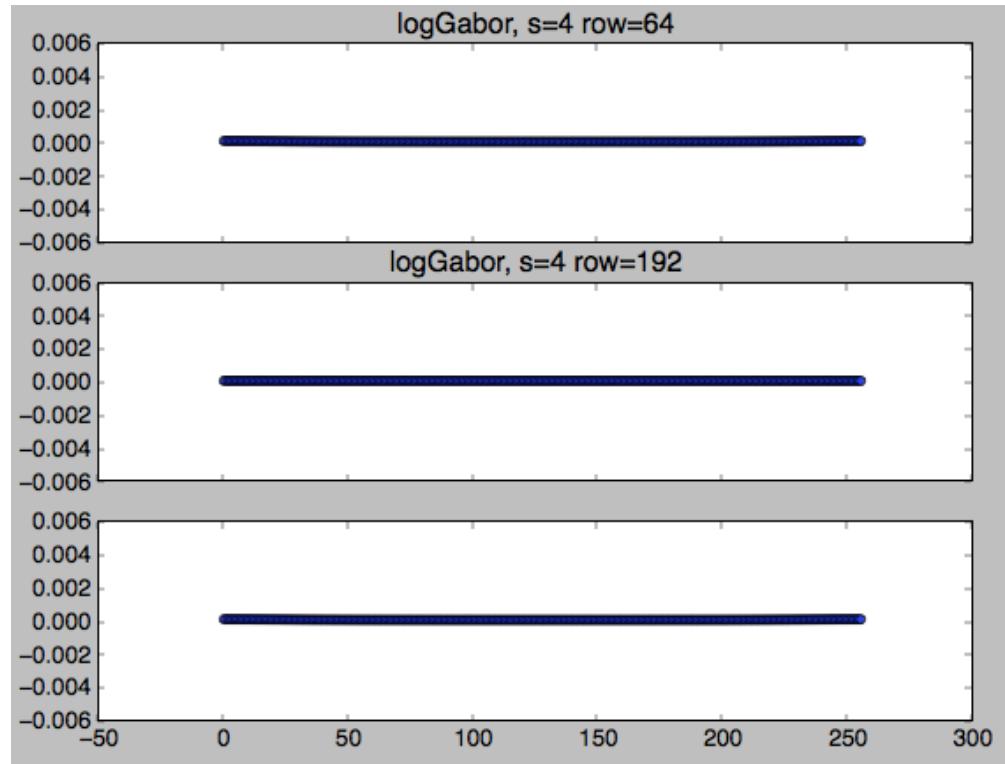
s=3, width = (sumAn / (maxAn + epsilon) - 1.) / (nscale - 1)



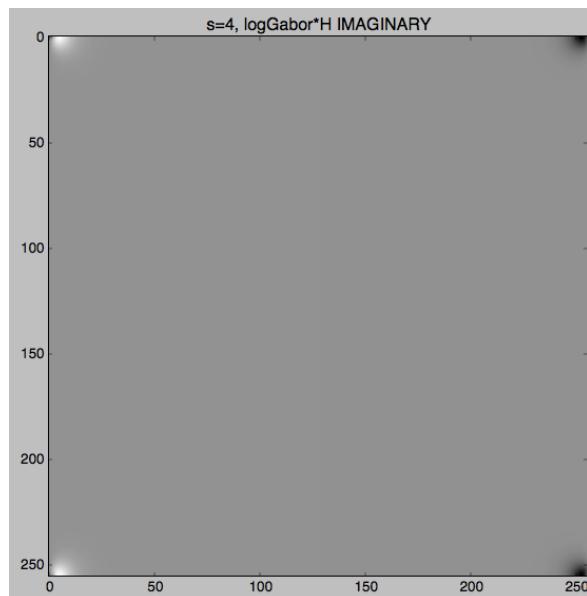
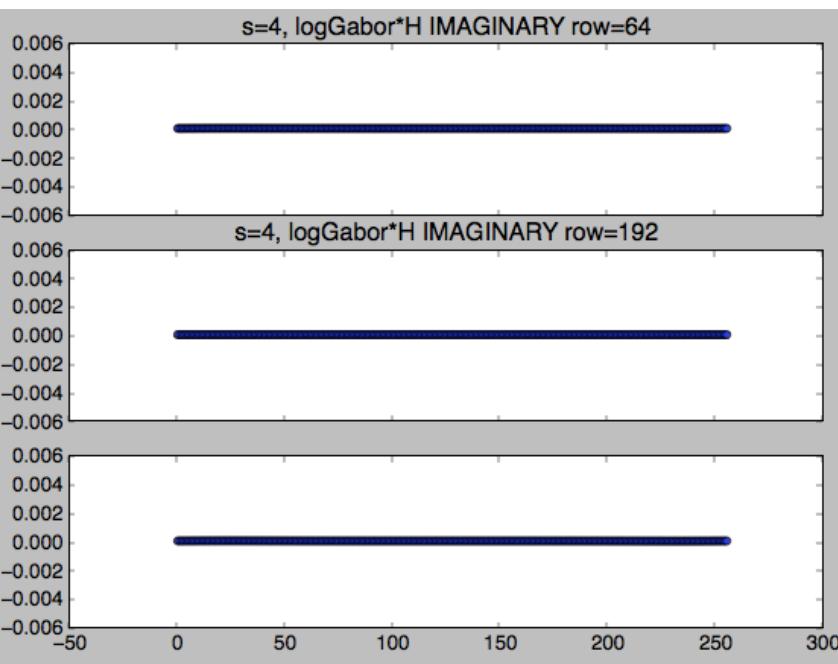
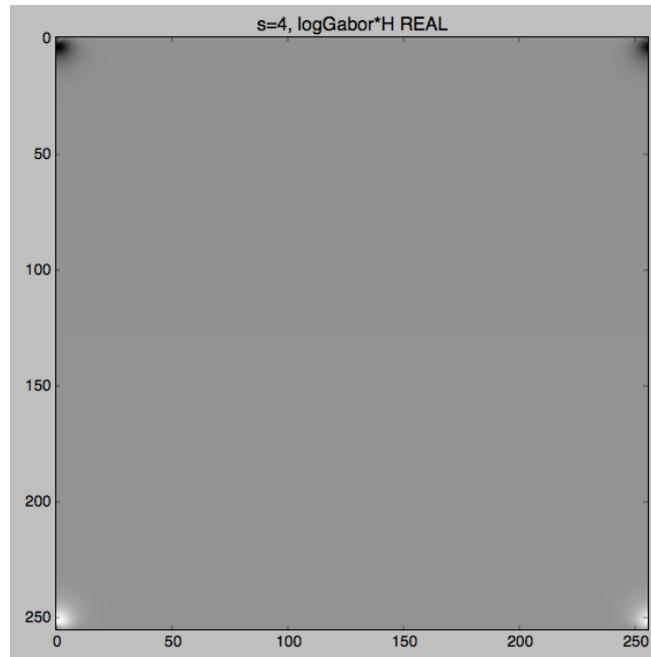
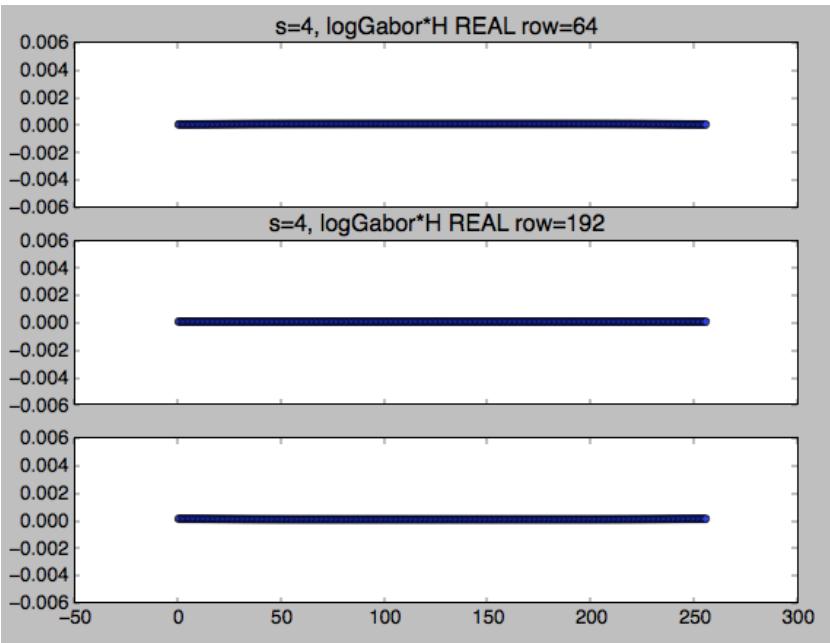
s=3, weight = $1. / (1. + \exp(g * (\text{cutoff} - \text{width})))$
(where cutoff=0.5, g=10.)



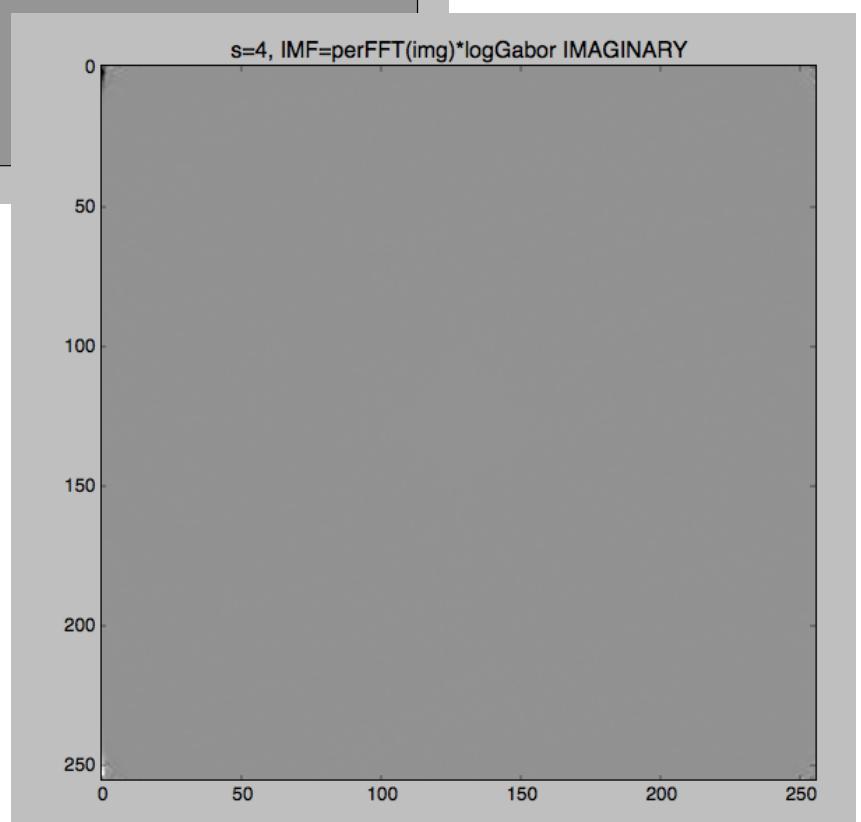
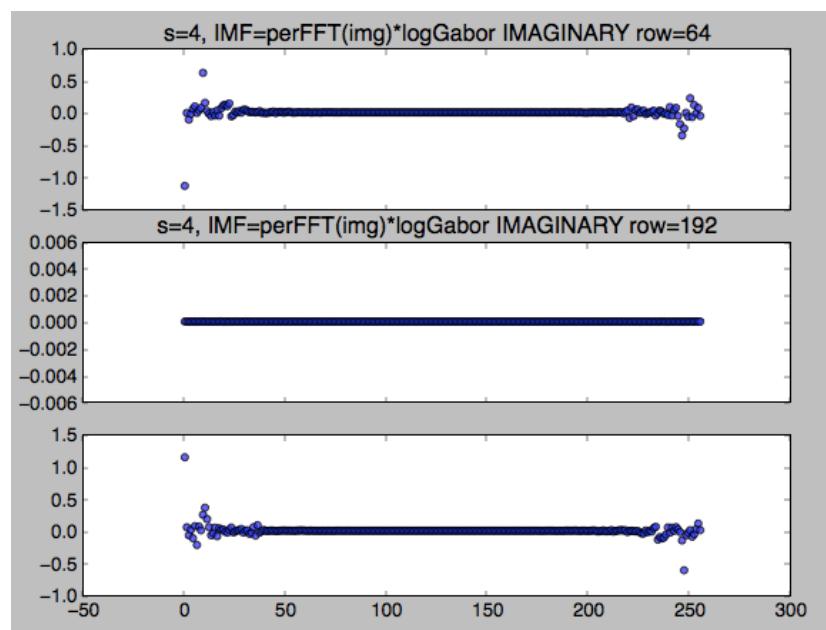
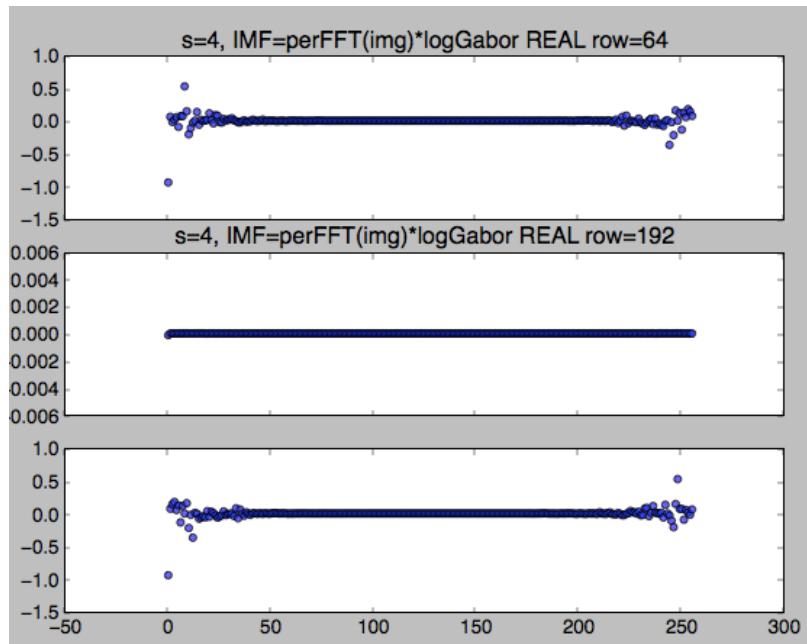
s=4, logGabor * low pass filter

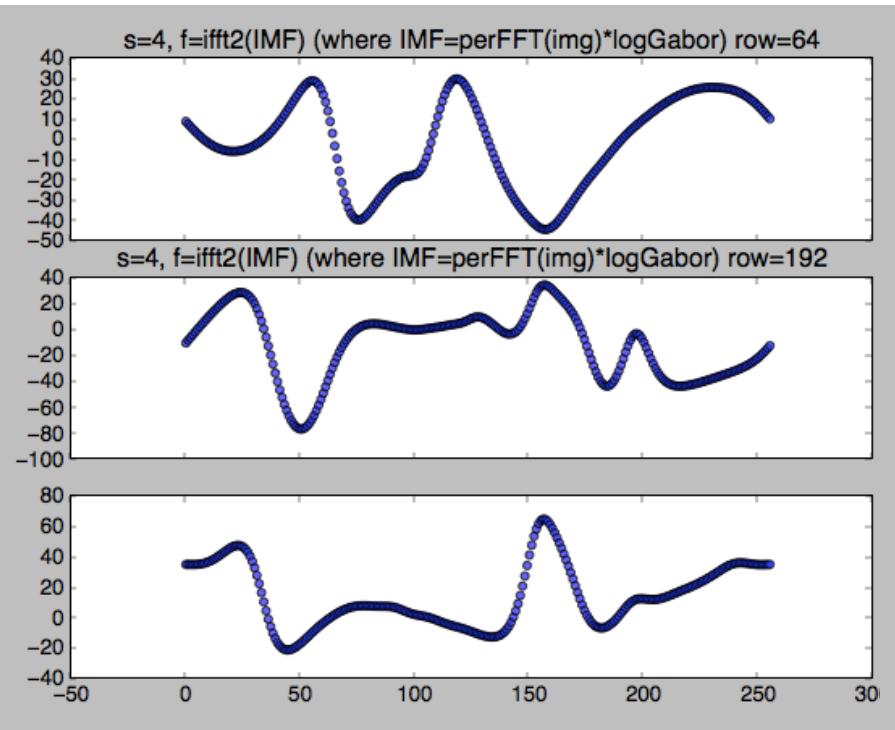


$s=4$, logGabor * low pass filter * H

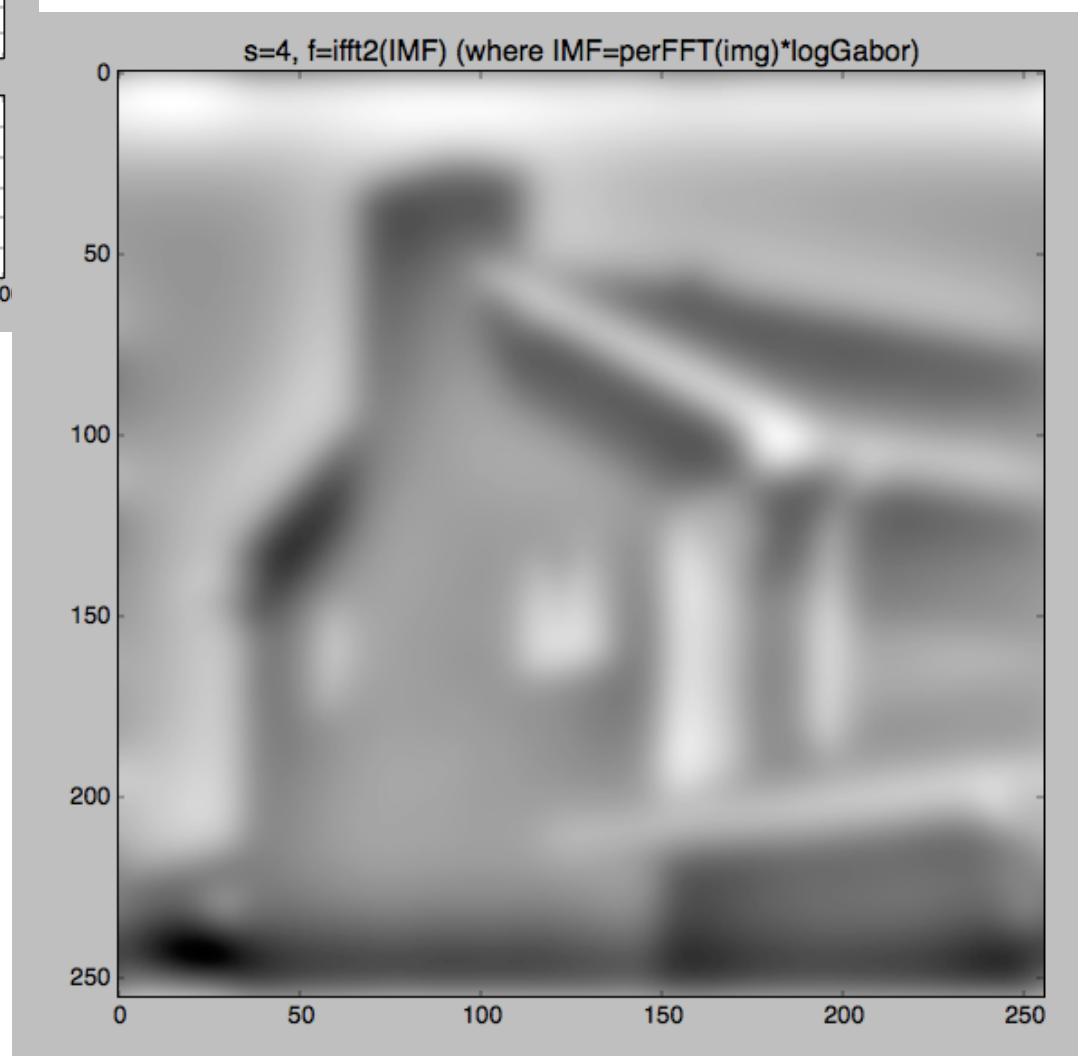


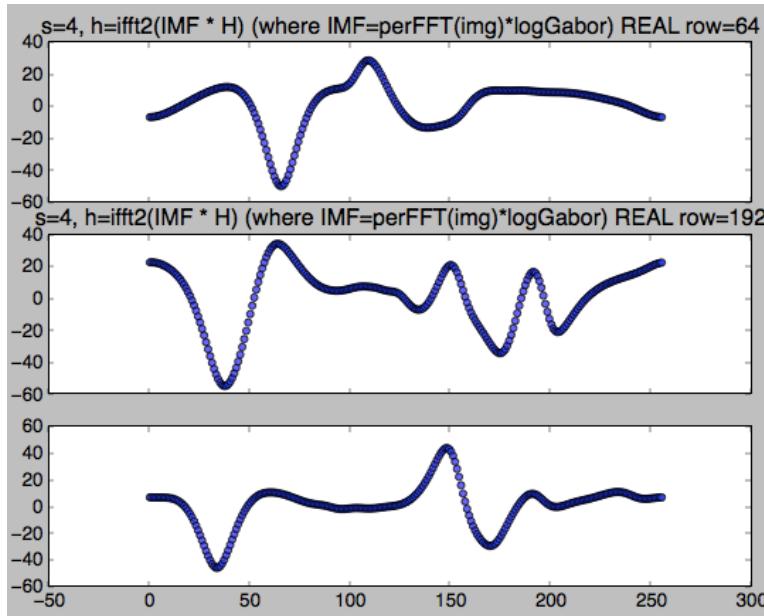
s=4, IMF=perFFT(img)*logGabor



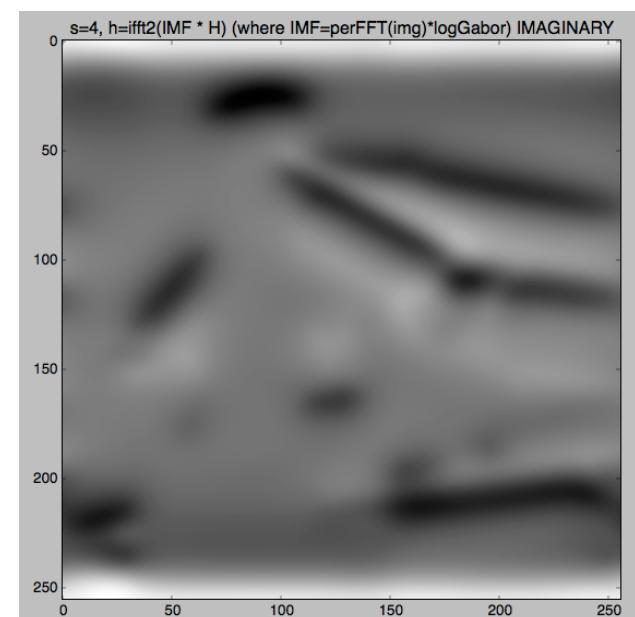
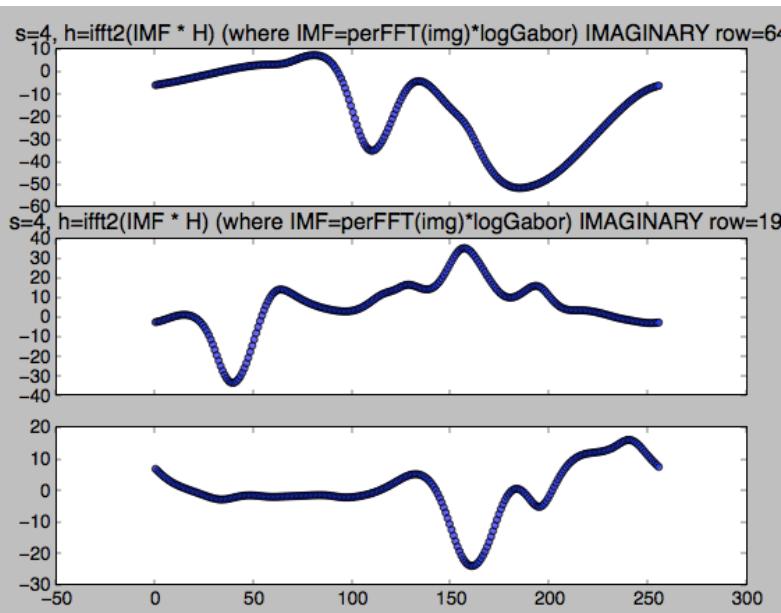
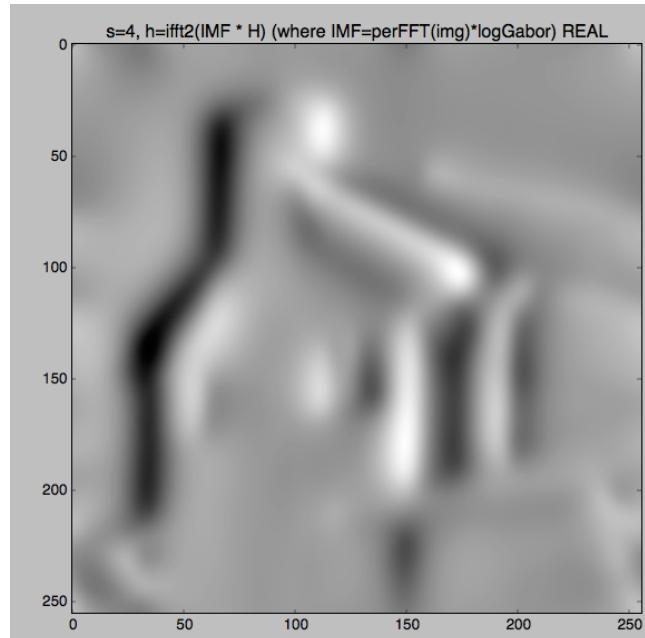


s=4, f=ifft2(IMF)
(where IMF=perFFT(img)*logGabor)

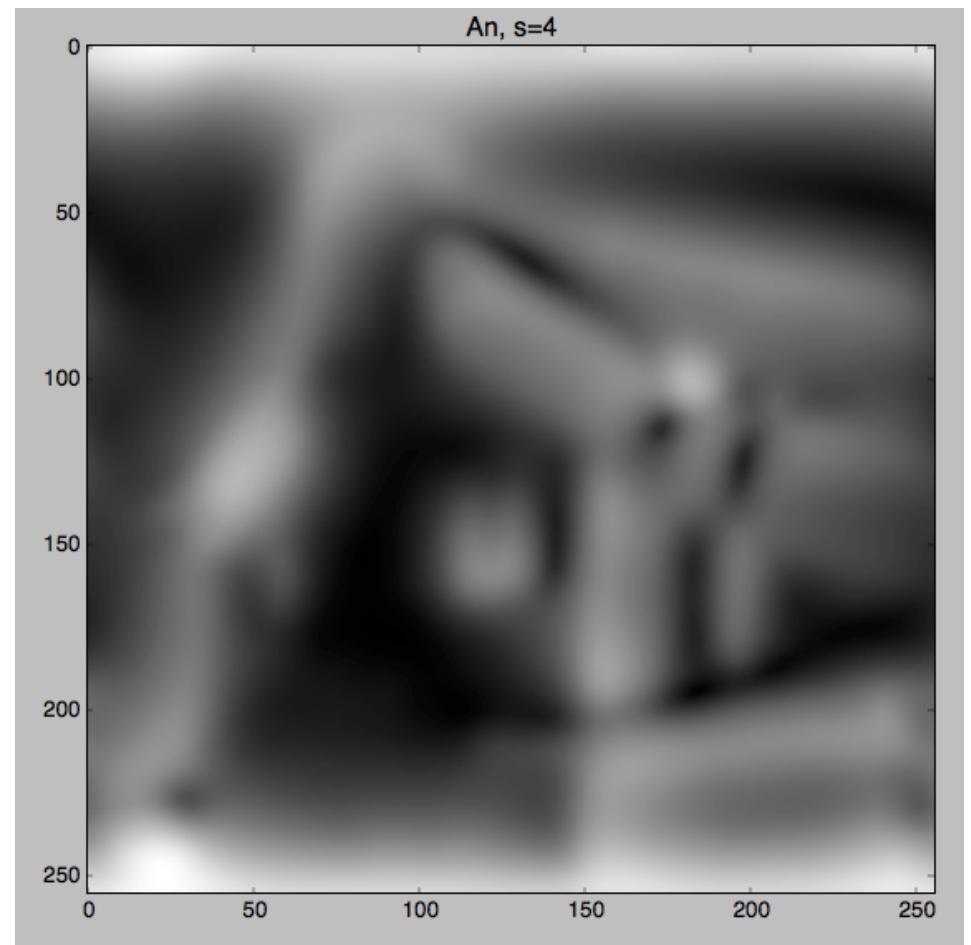
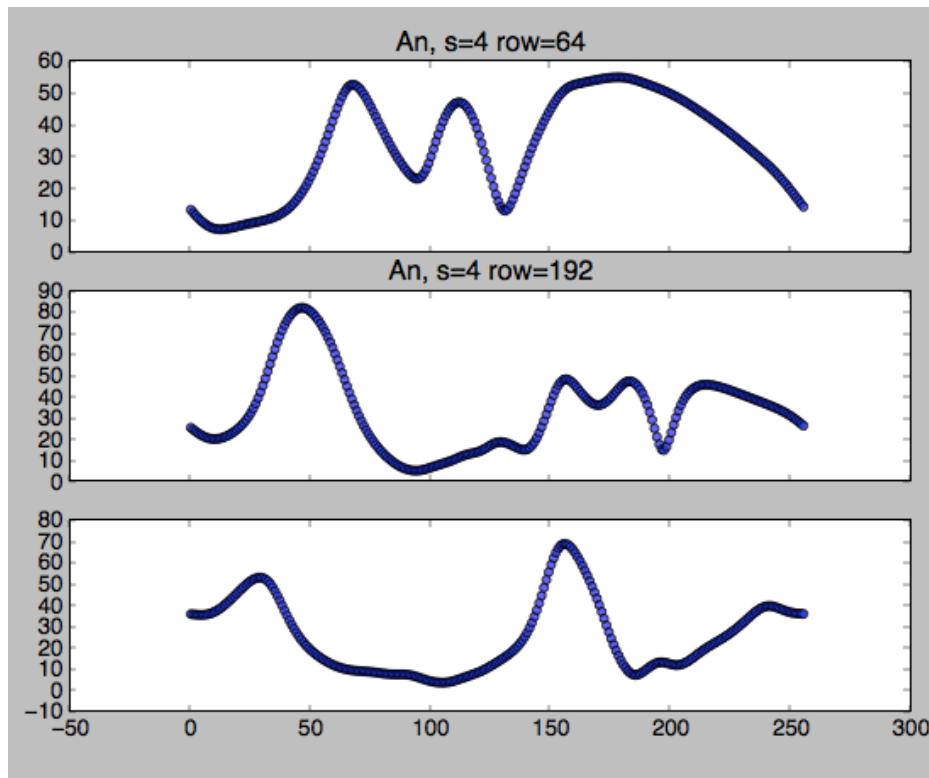




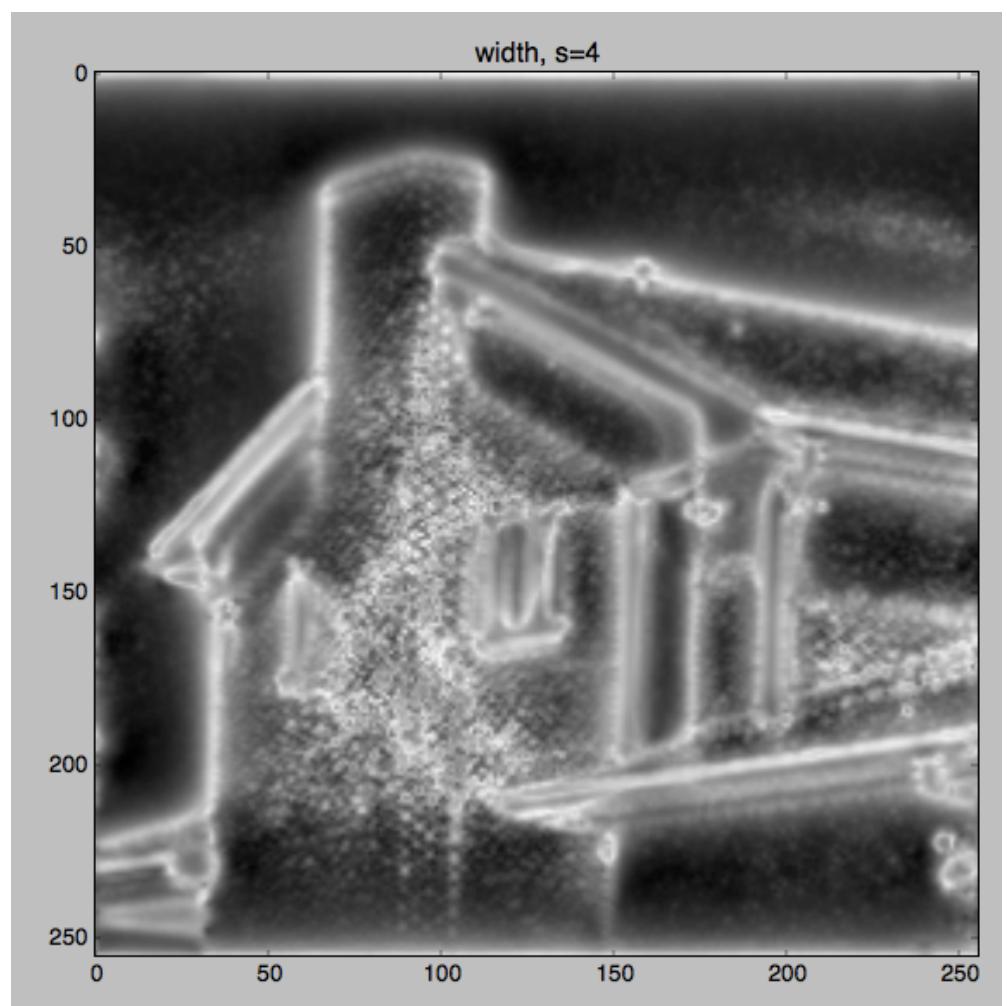
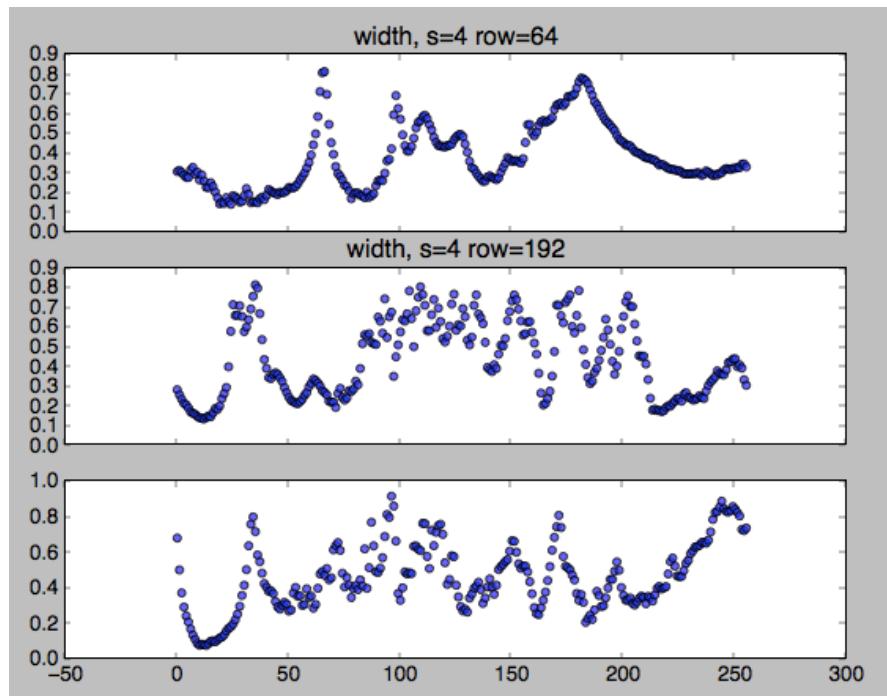
**s=4, h=ifft2(IMF * H)
(where IMF=perFFT(img)*logGabor)**



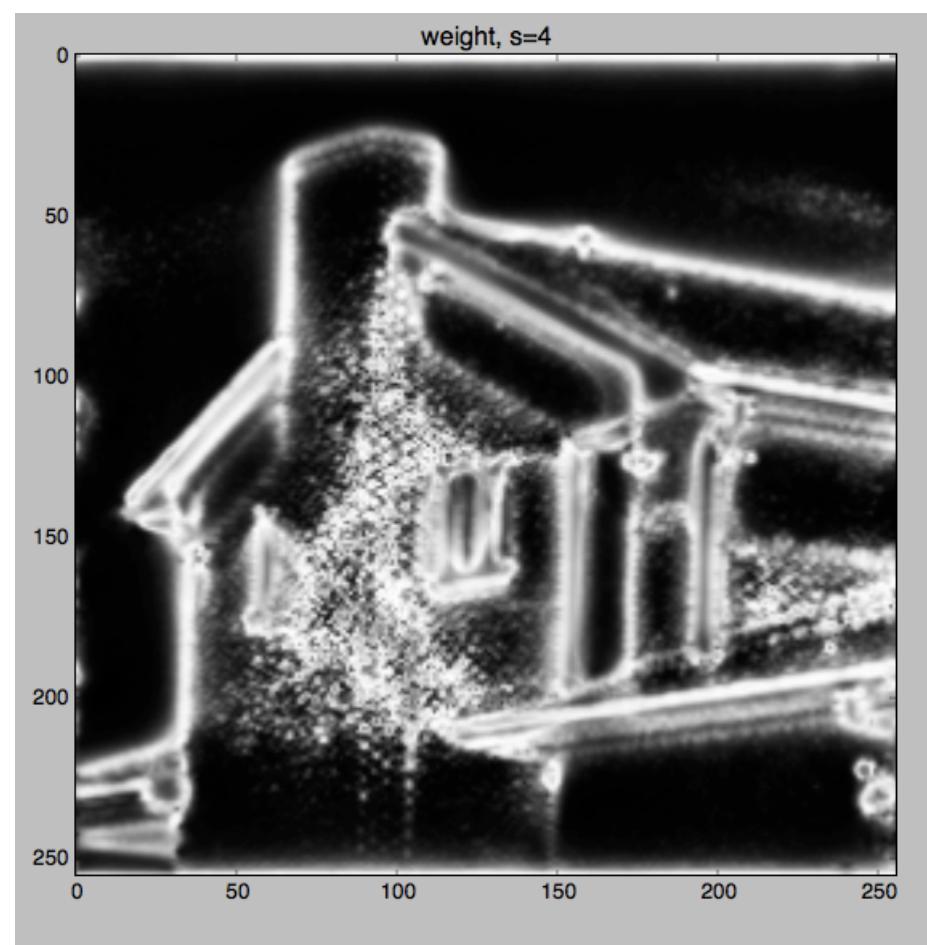
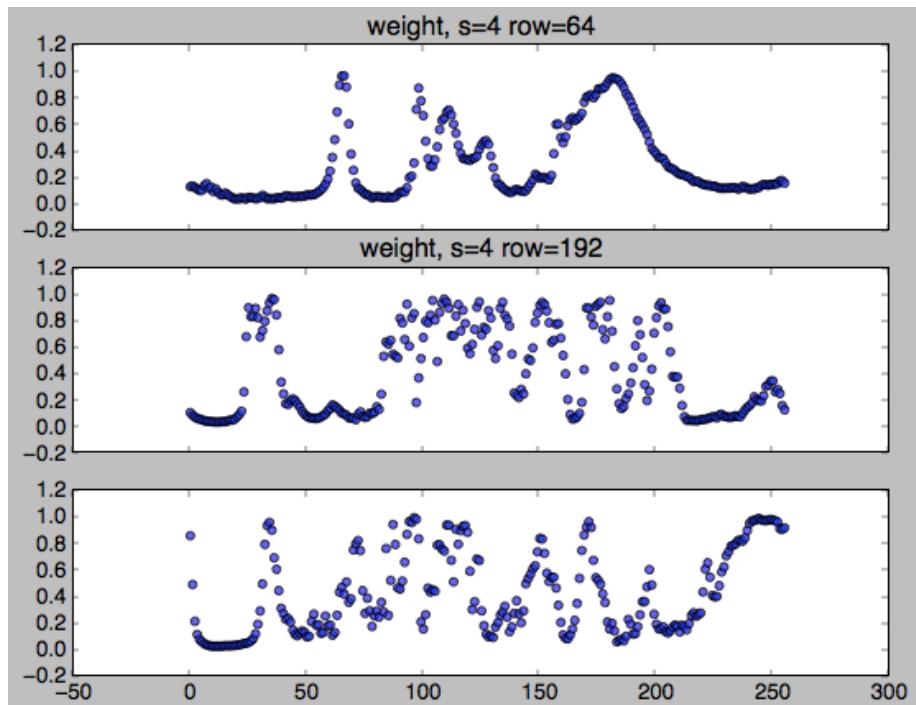
$s=4$, $A_n = \sqrt{f * f + h1 * h1 + h2 * h2}$
(where $f=\text{iFFT}(\text{perFFT(img)} * \text{logGabor})$
and $h=\text{iFFT}(\text{perFFT(img)} * \text{logGabor} * H)$)



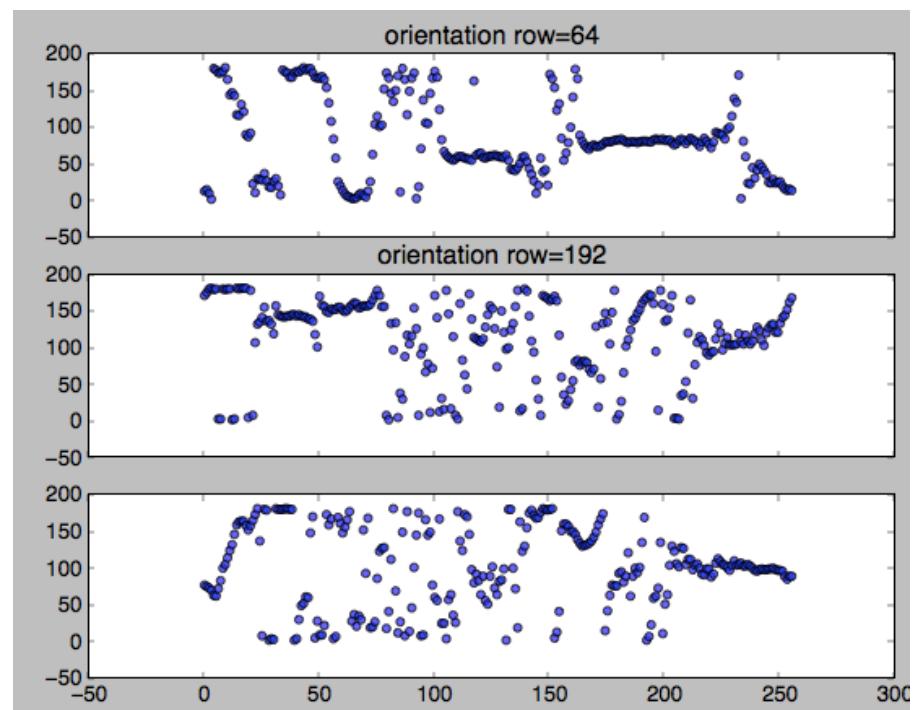
s=4, width = (sumAn / (maxAn + epsilon) - 1.) / (nscale - 1)



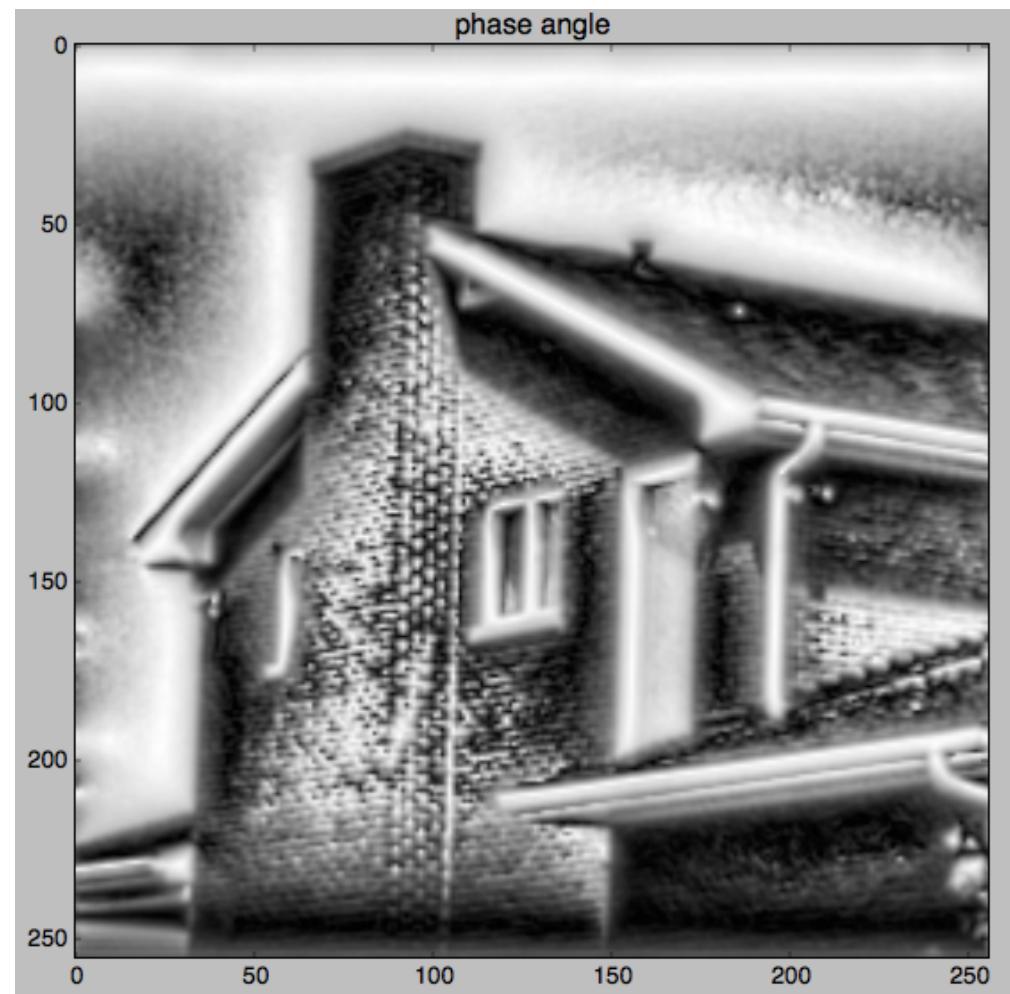
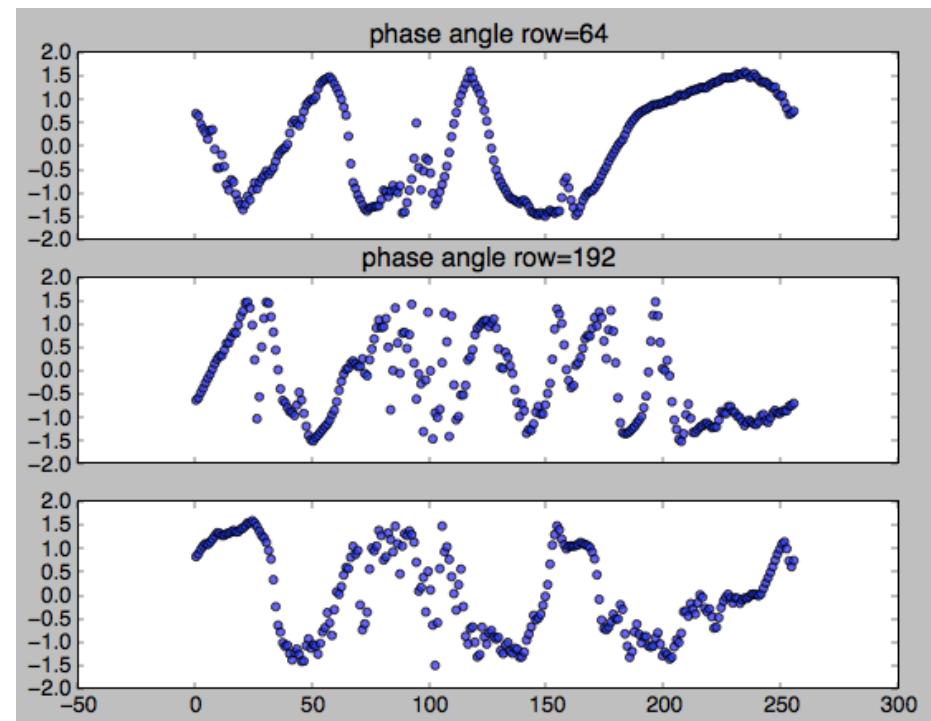
s=4, weight = $1. / (1. + \exp(g * (\text{cutoff} - \text{width})))$
(where cutoff=0.5, g=10.)



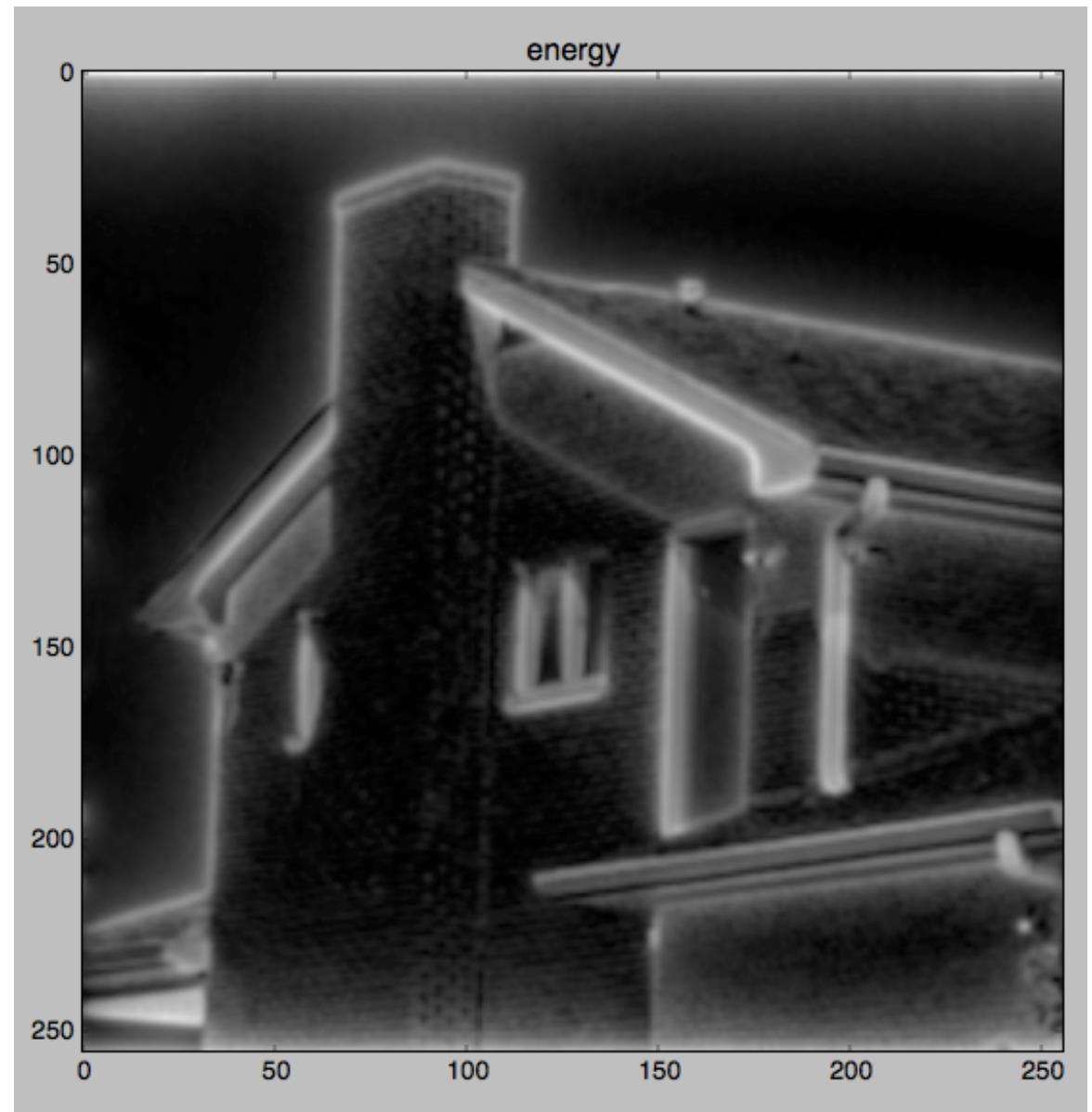
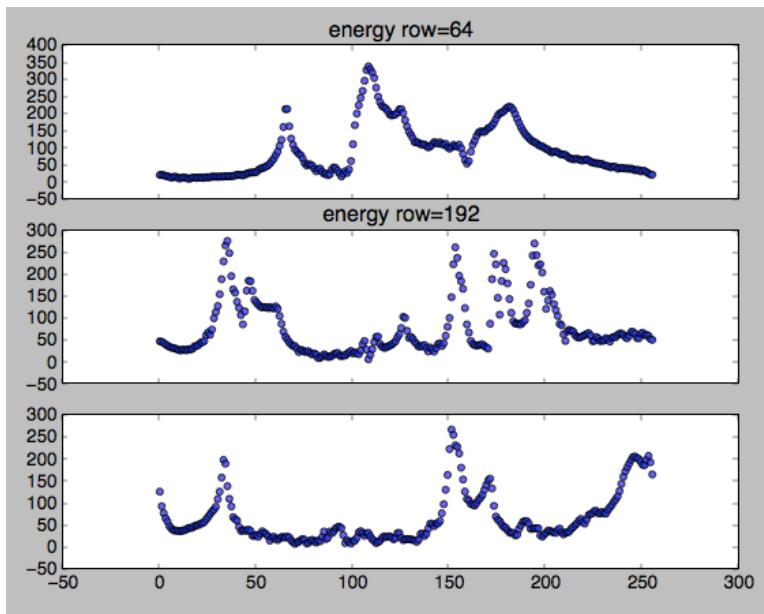
orientation

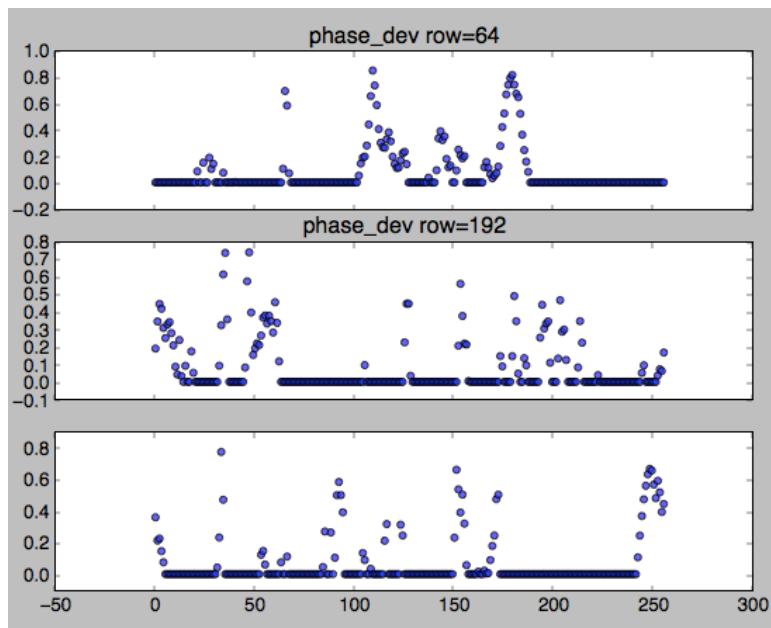


phase angle

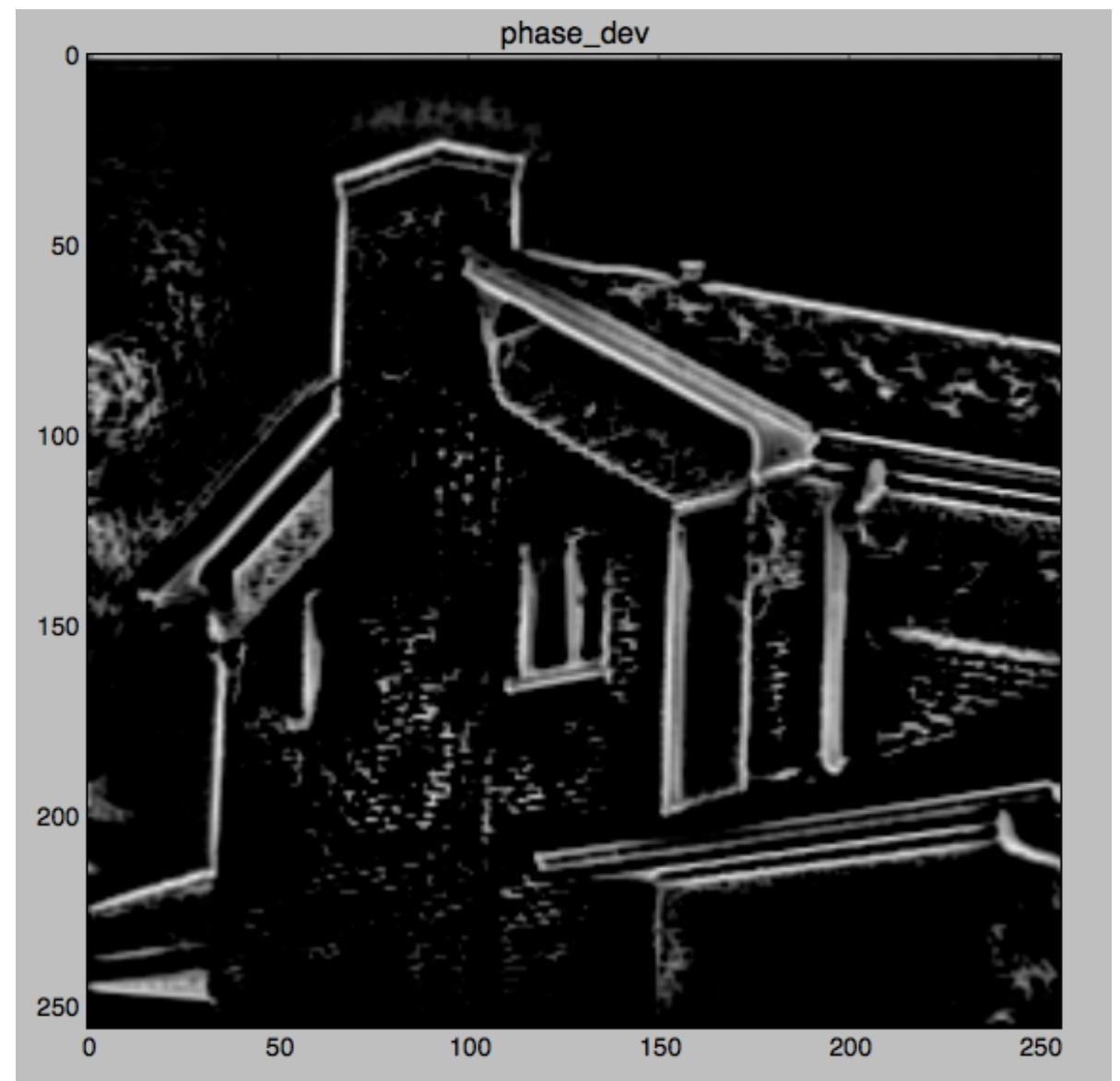


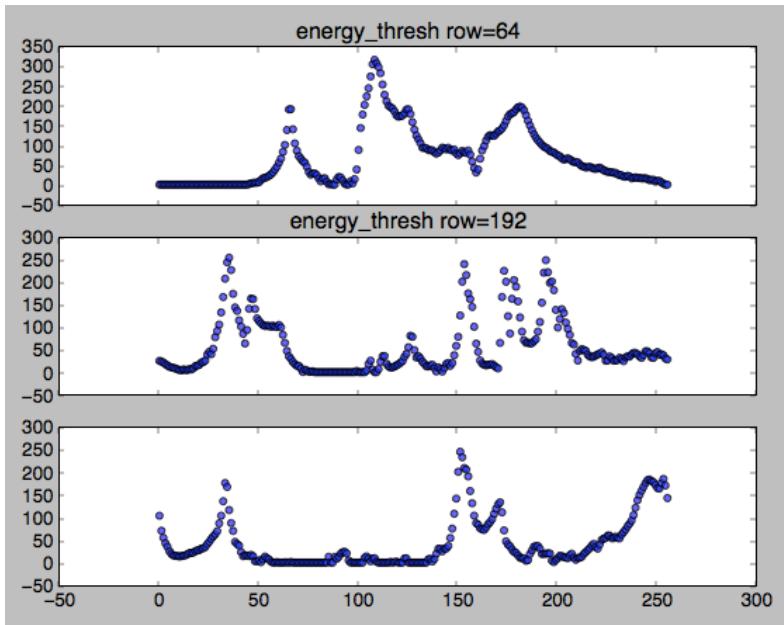
energy



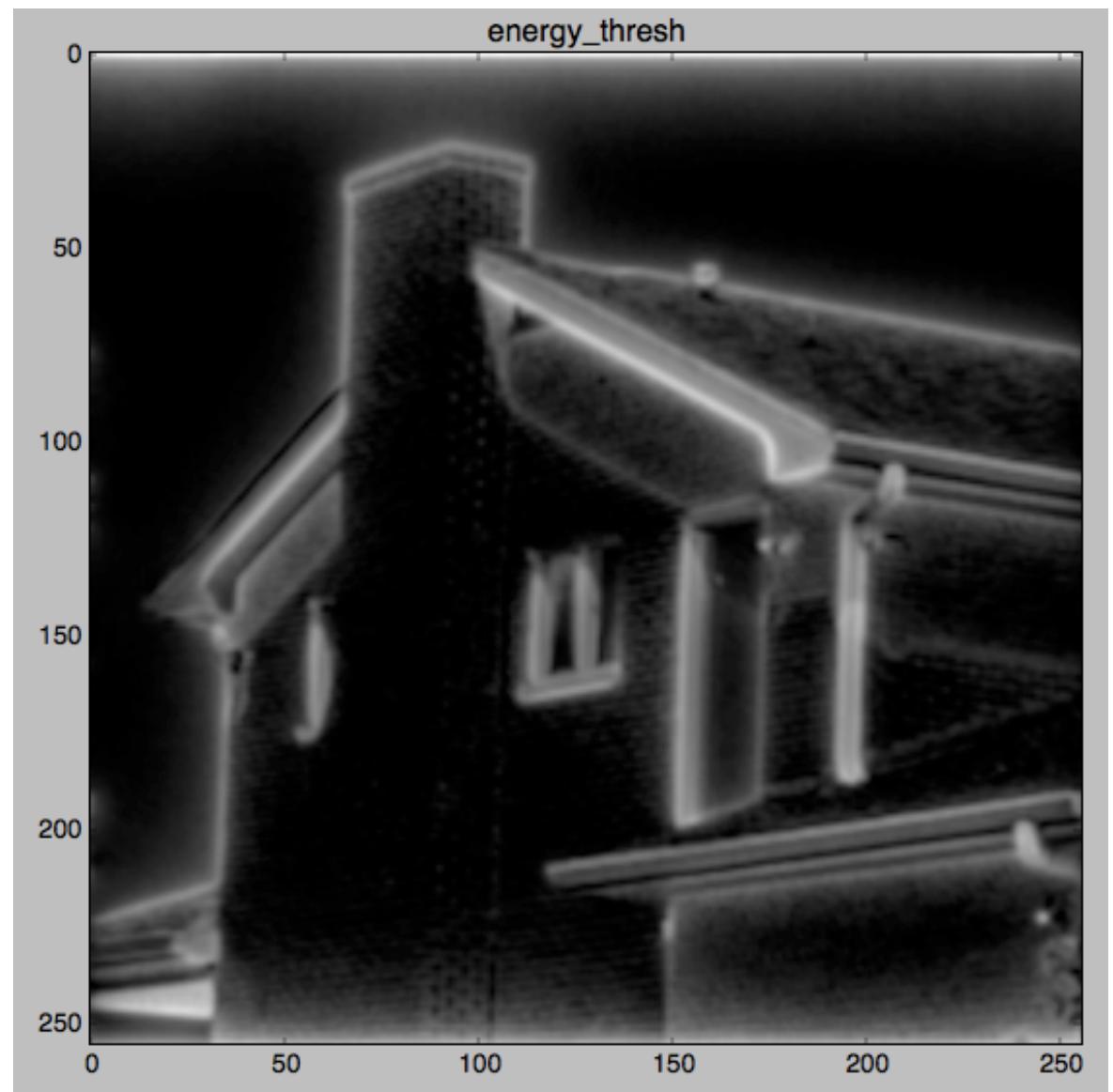


phase_dev

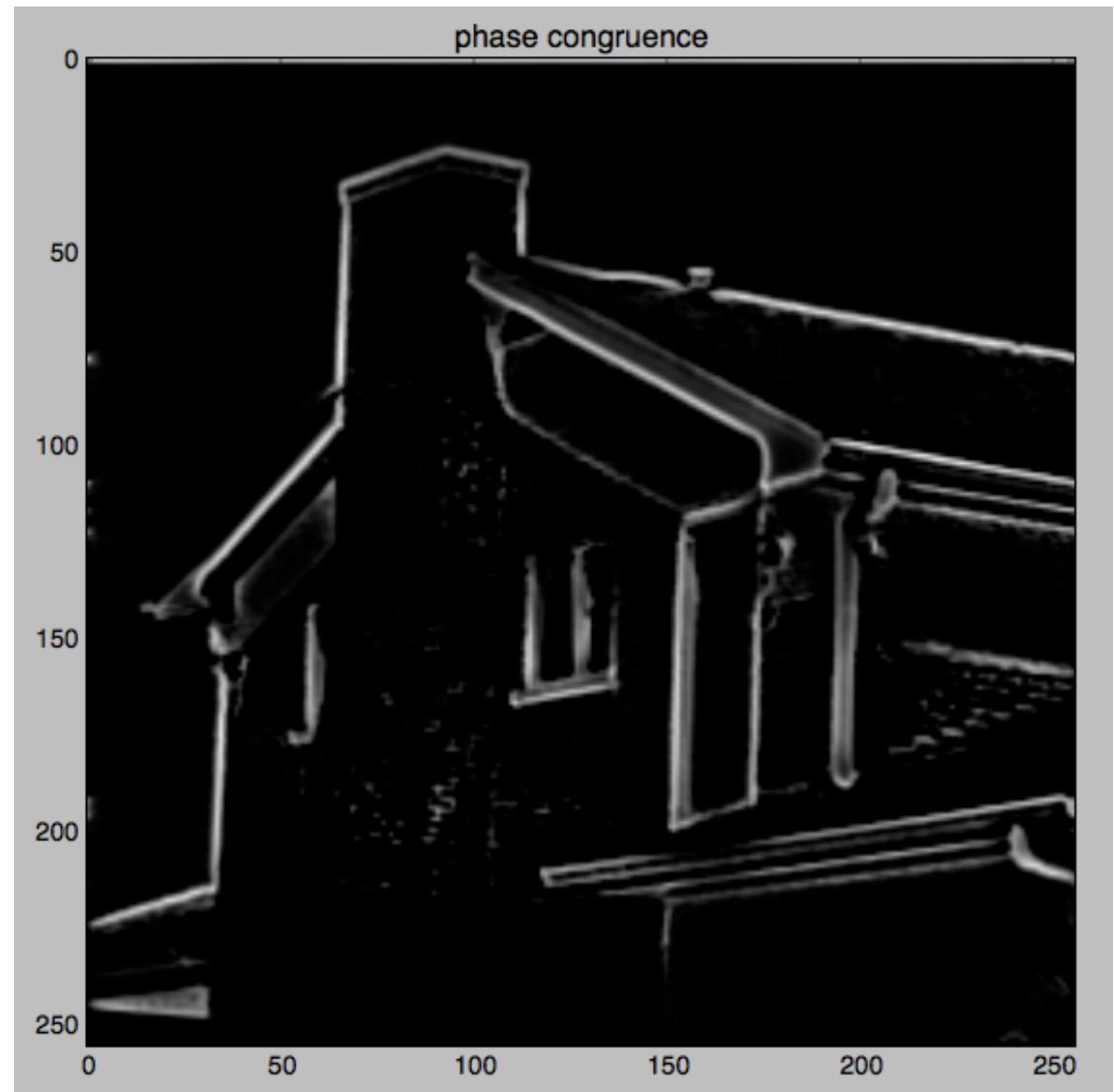
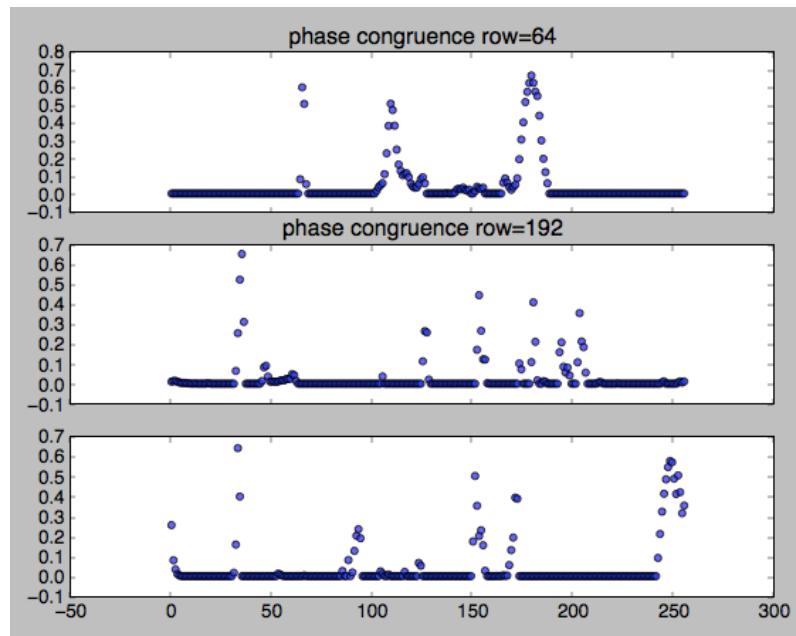




energy_thresh



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

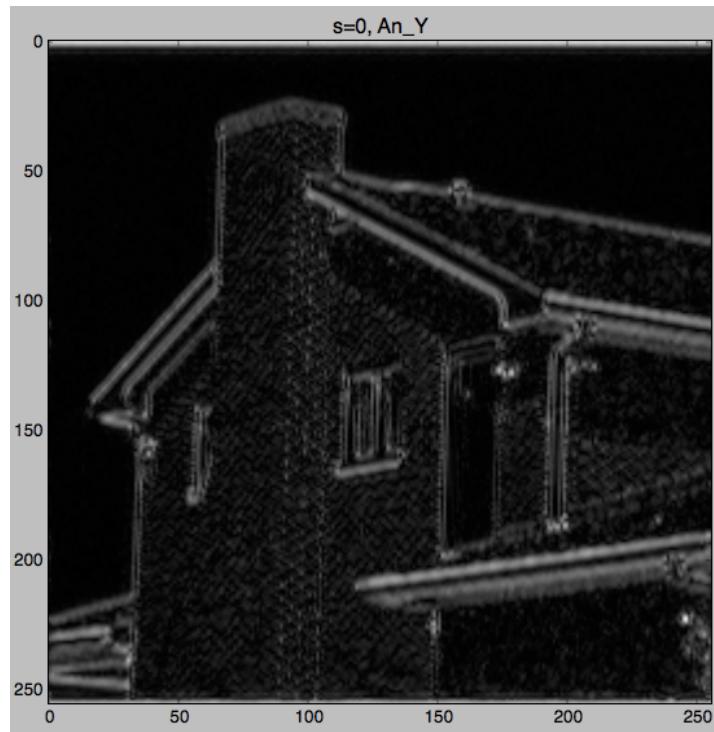
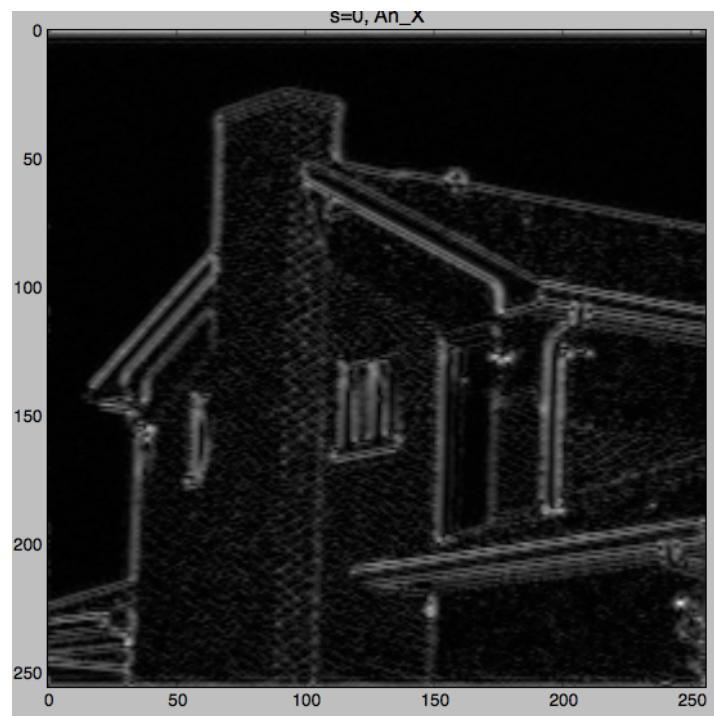
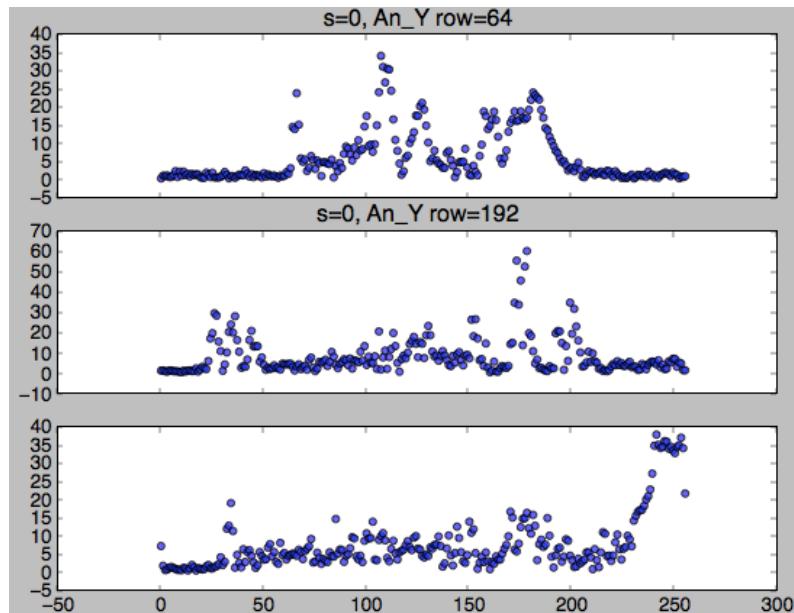
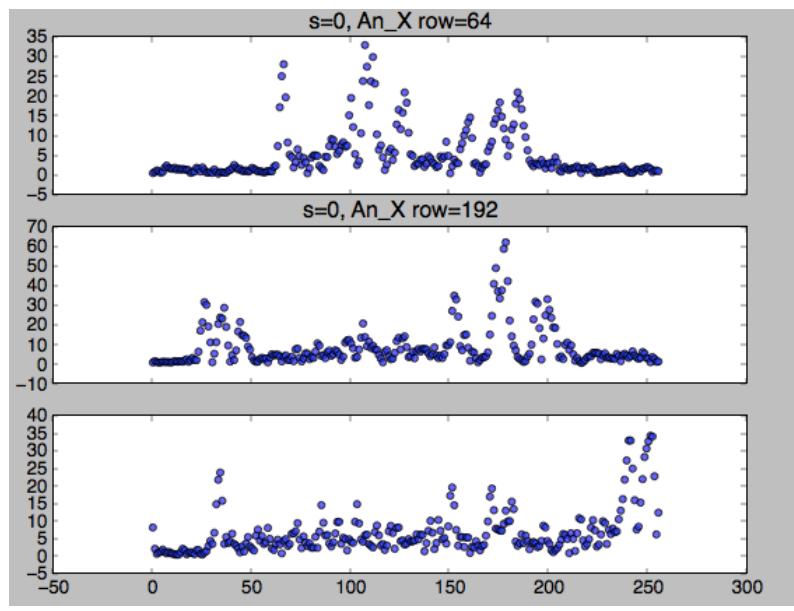


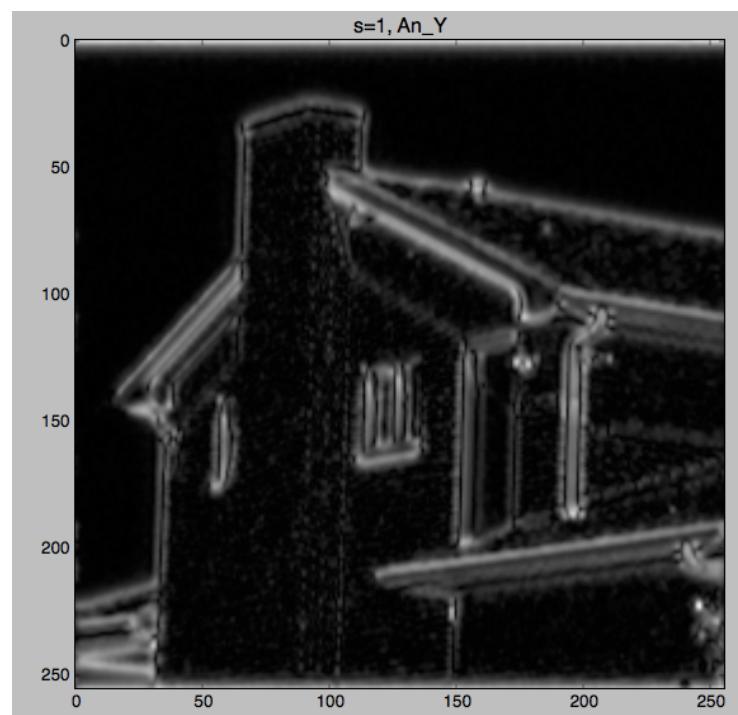
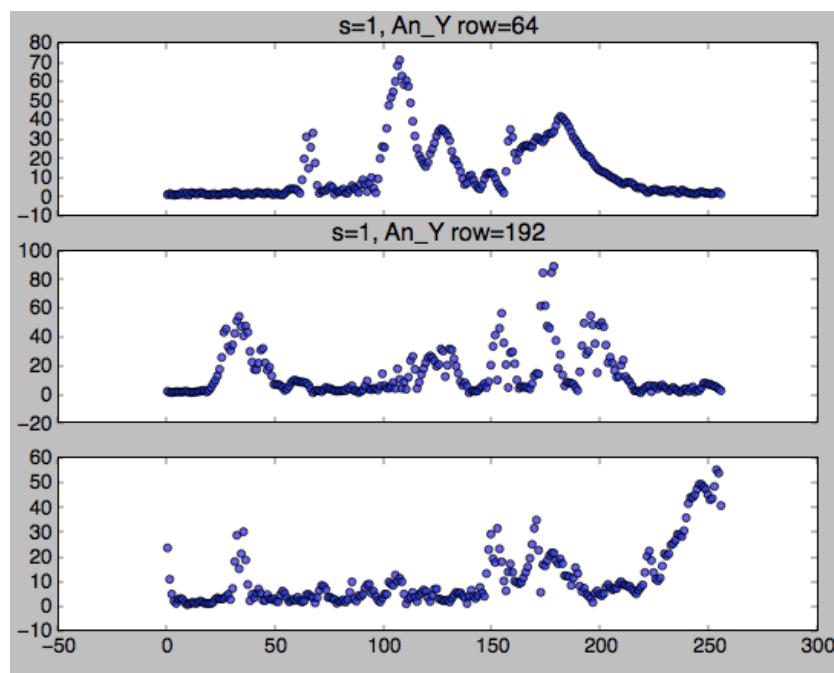
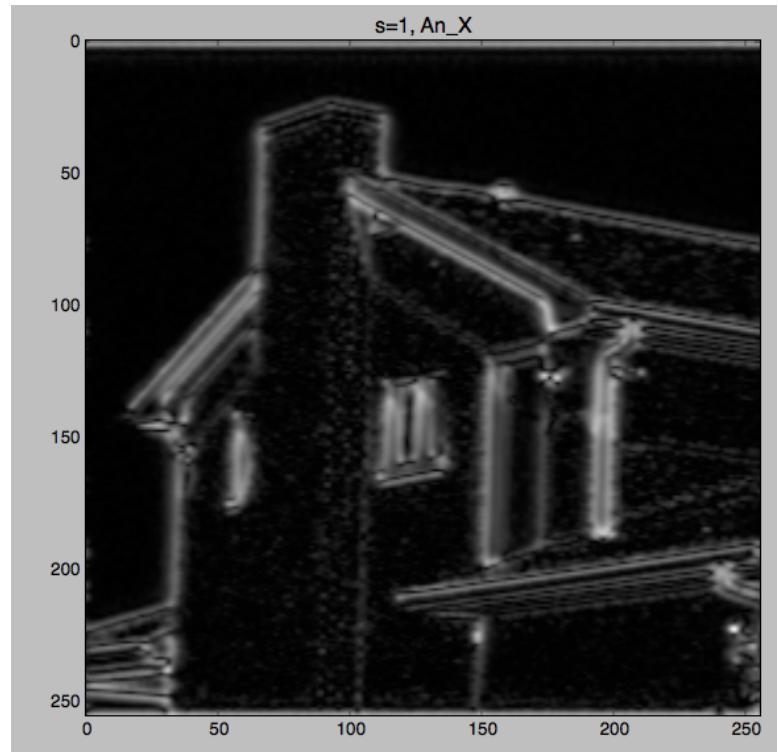
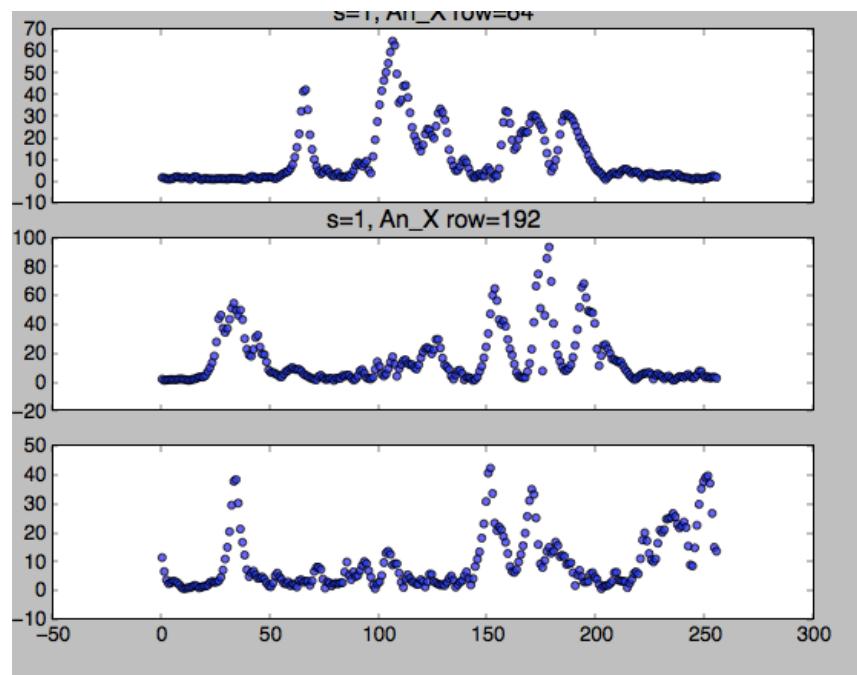
Looking at the x and y components of An

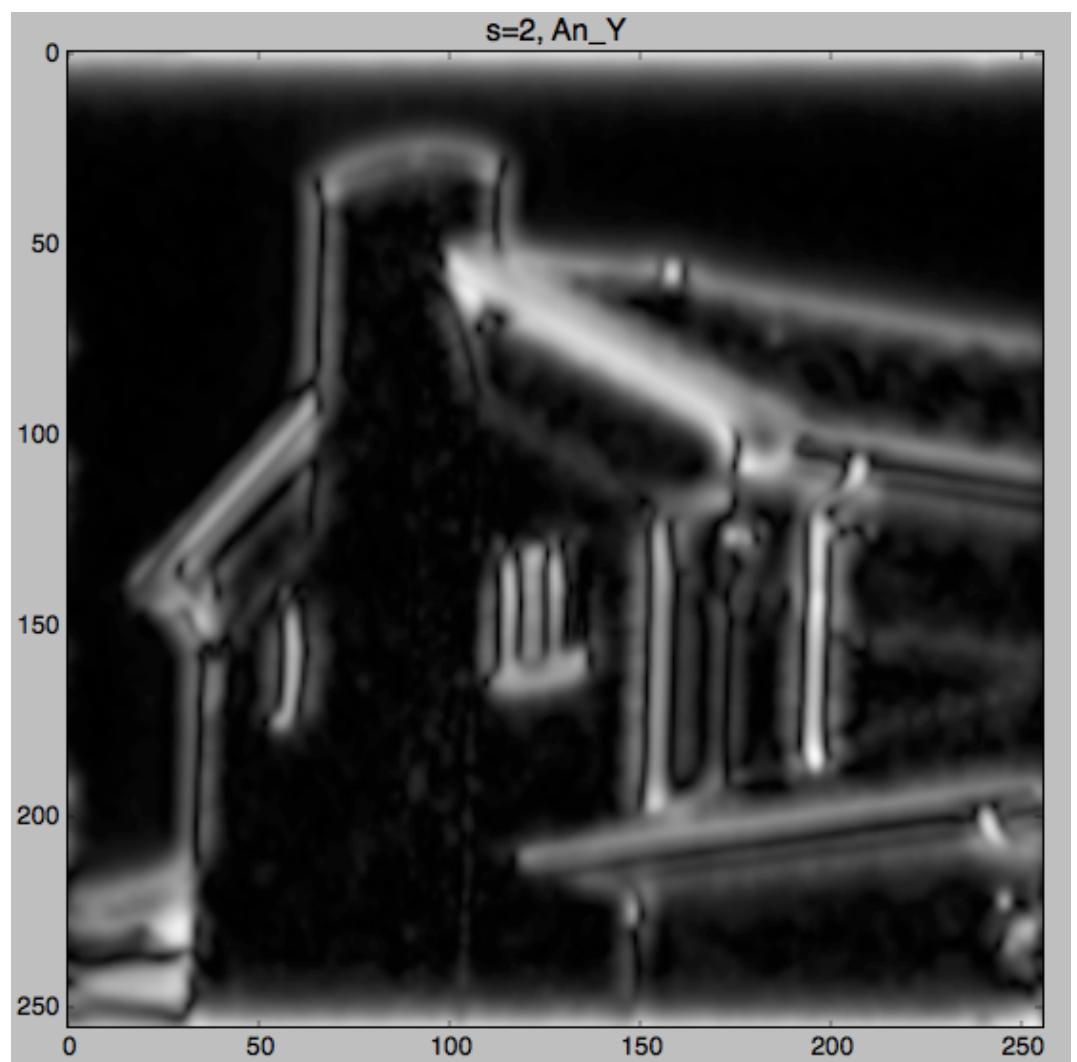
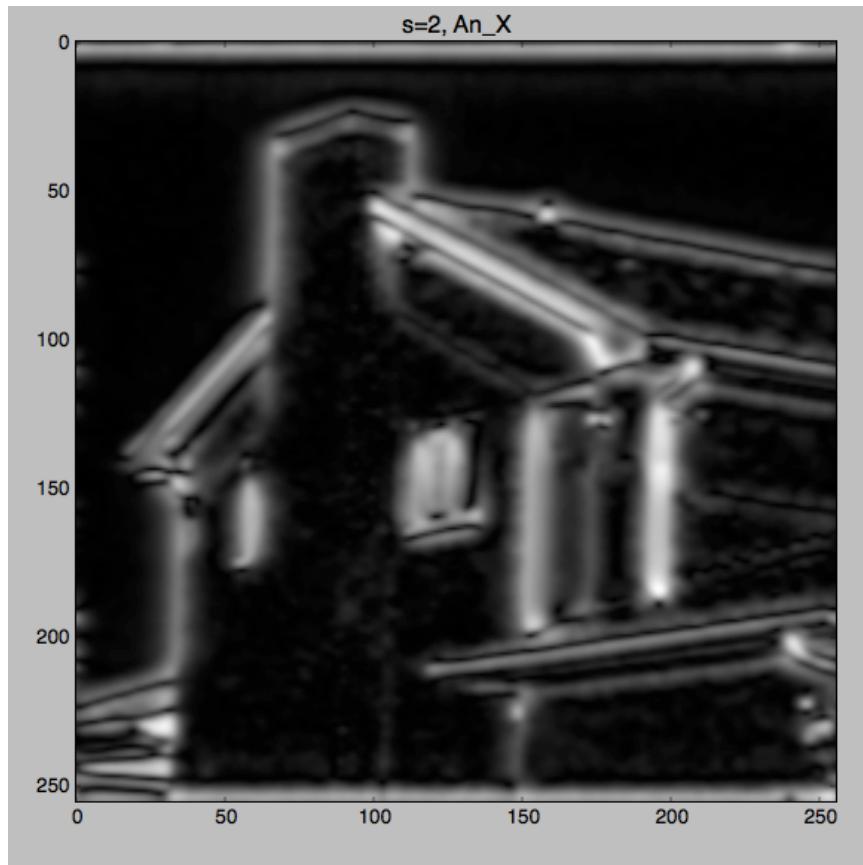
And afterwards, attempting to make a 2nd derive filter.

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

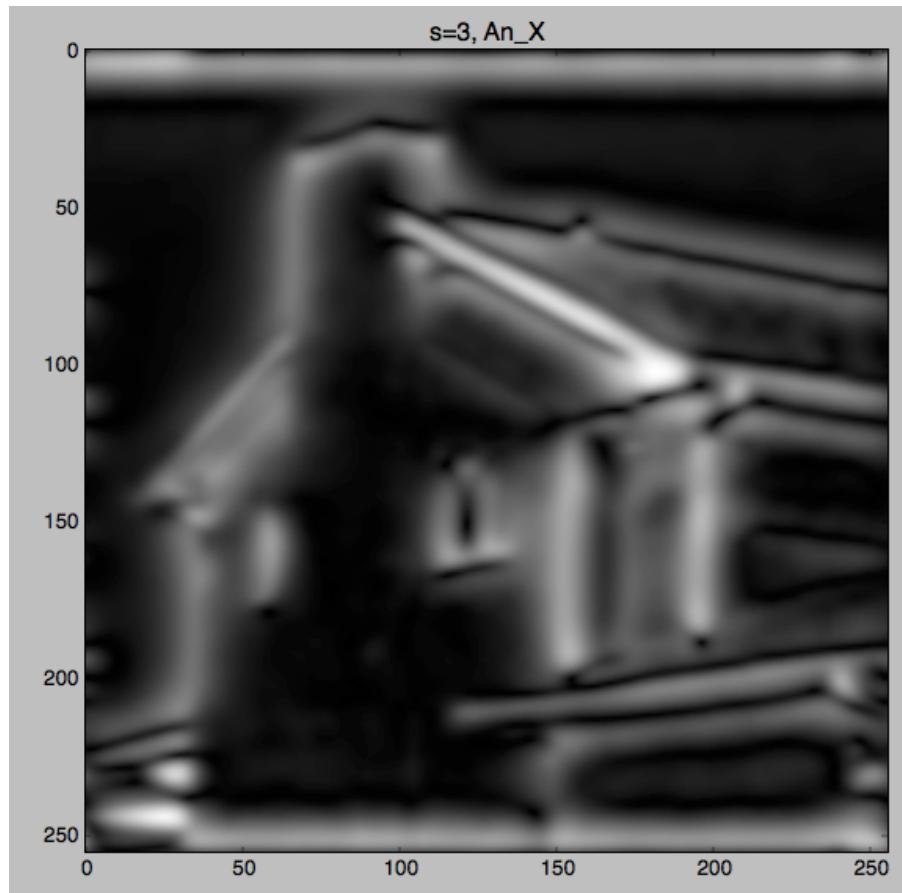
$$k(t,o\sim) = \frac{X_{dot}(t,o\sim) * Y_{dot_dot}(t,o\sim) - Y_{dot}(t,o\sim) * X_{dot_dot}(t,o\sim)}{(X_{dot}^2(t,o\sim) + Y_{dot}^2(t,o\sim))^{1.5}}$$



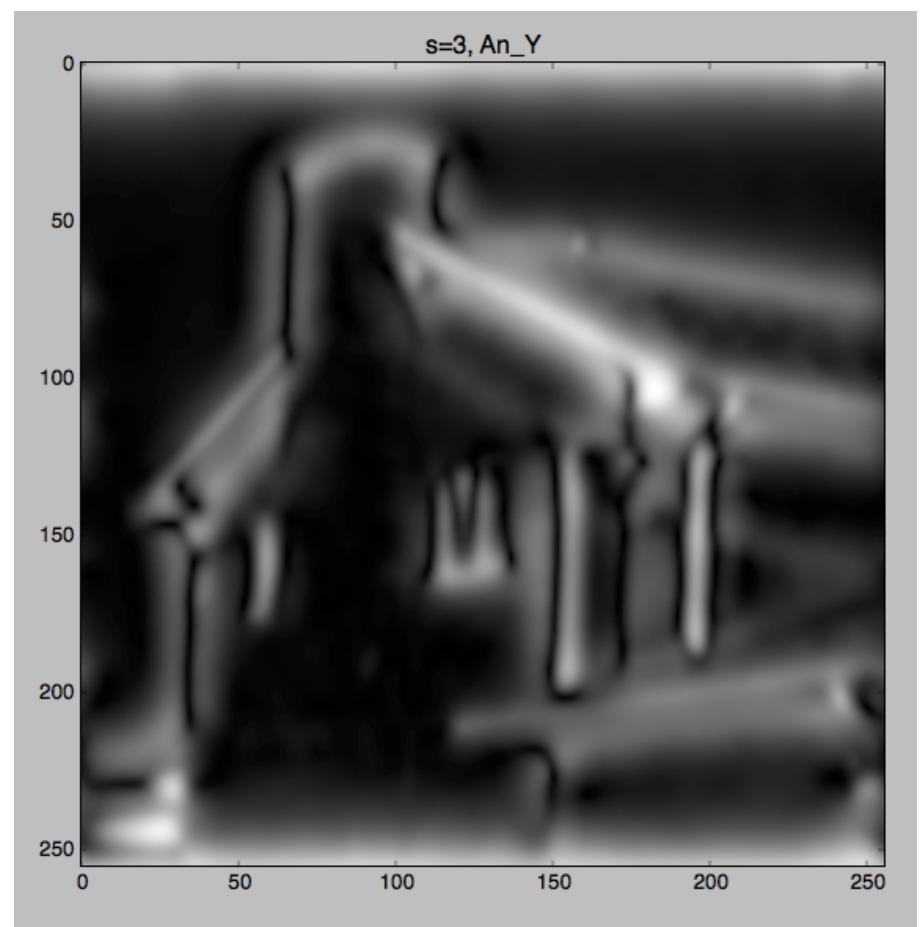


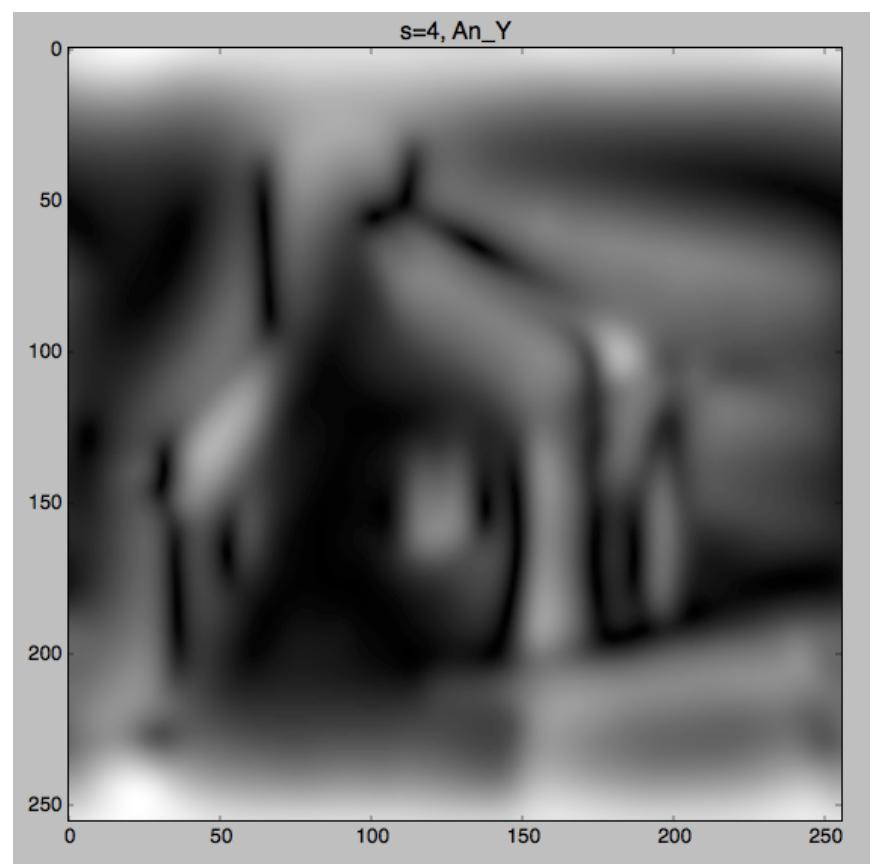
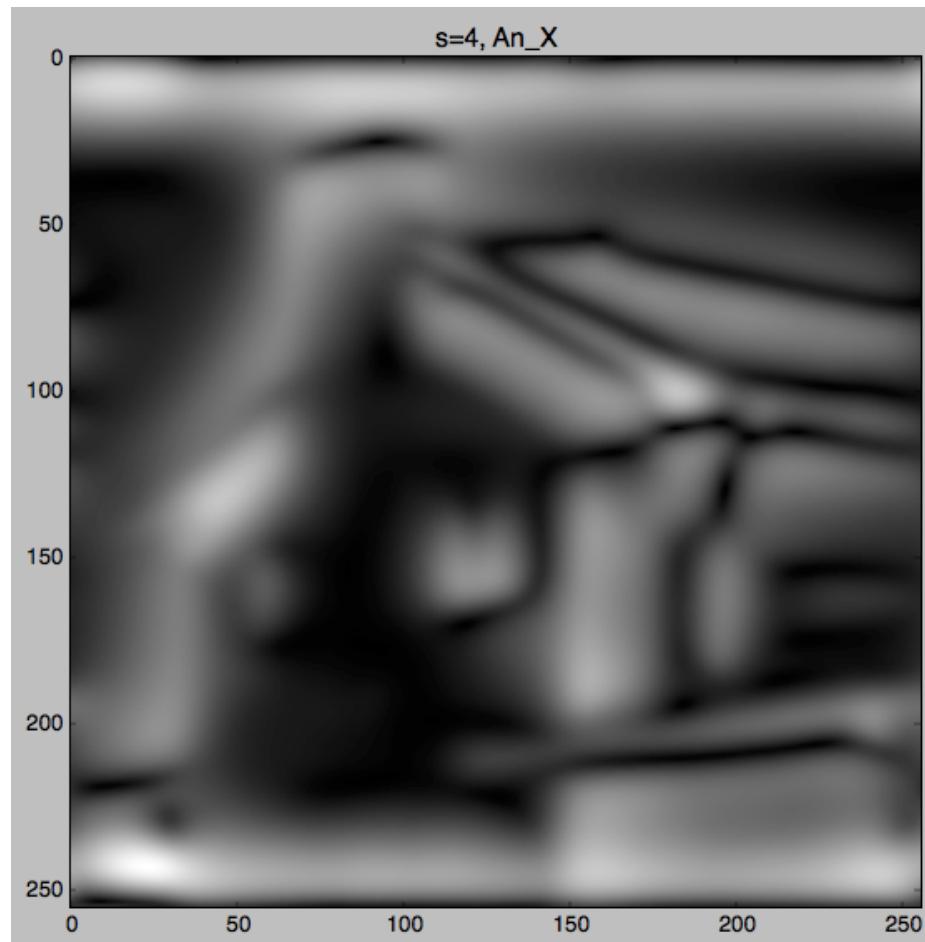


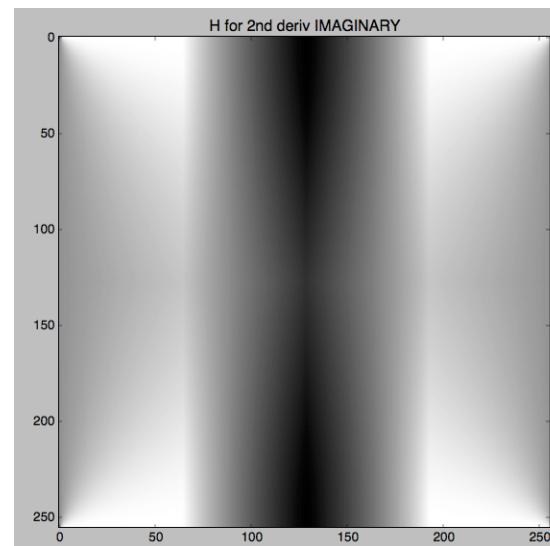
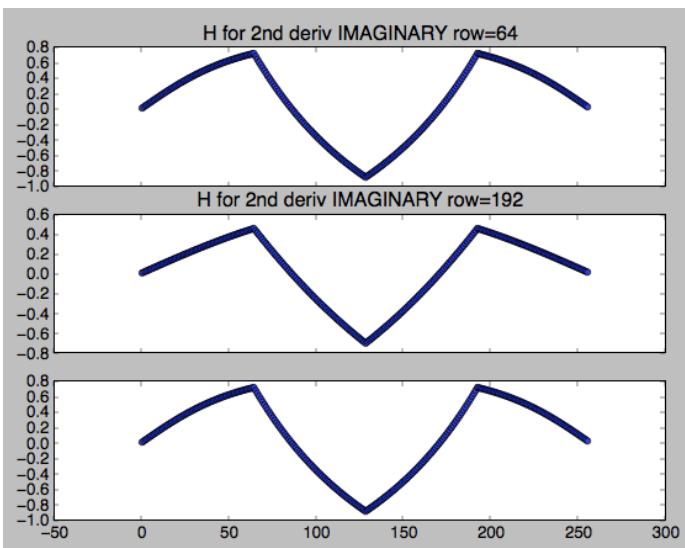
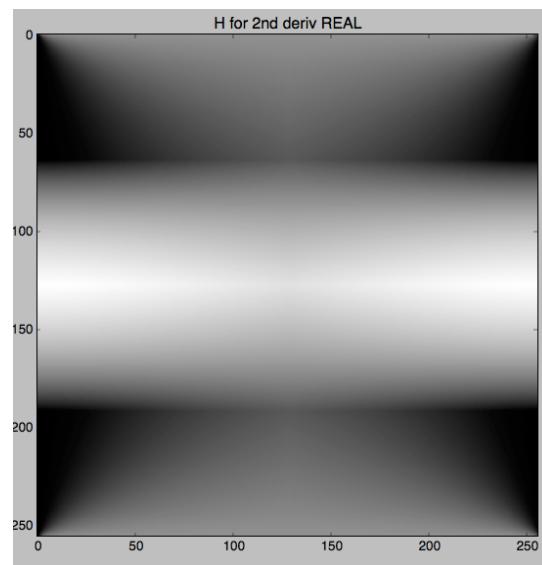
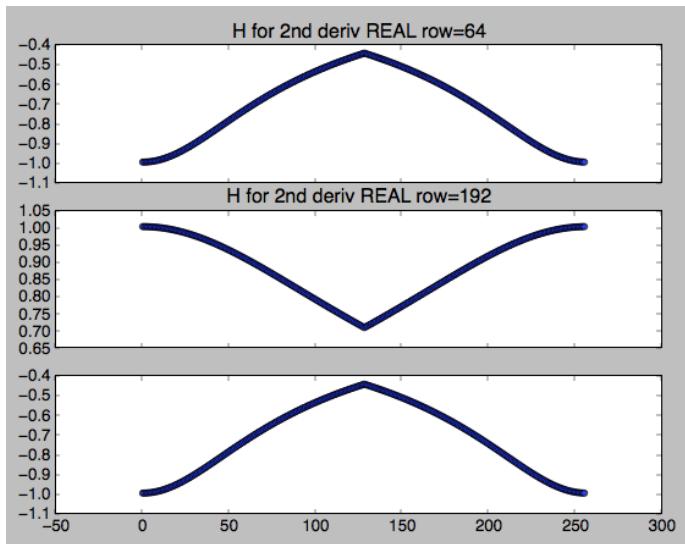
s=3, An_X

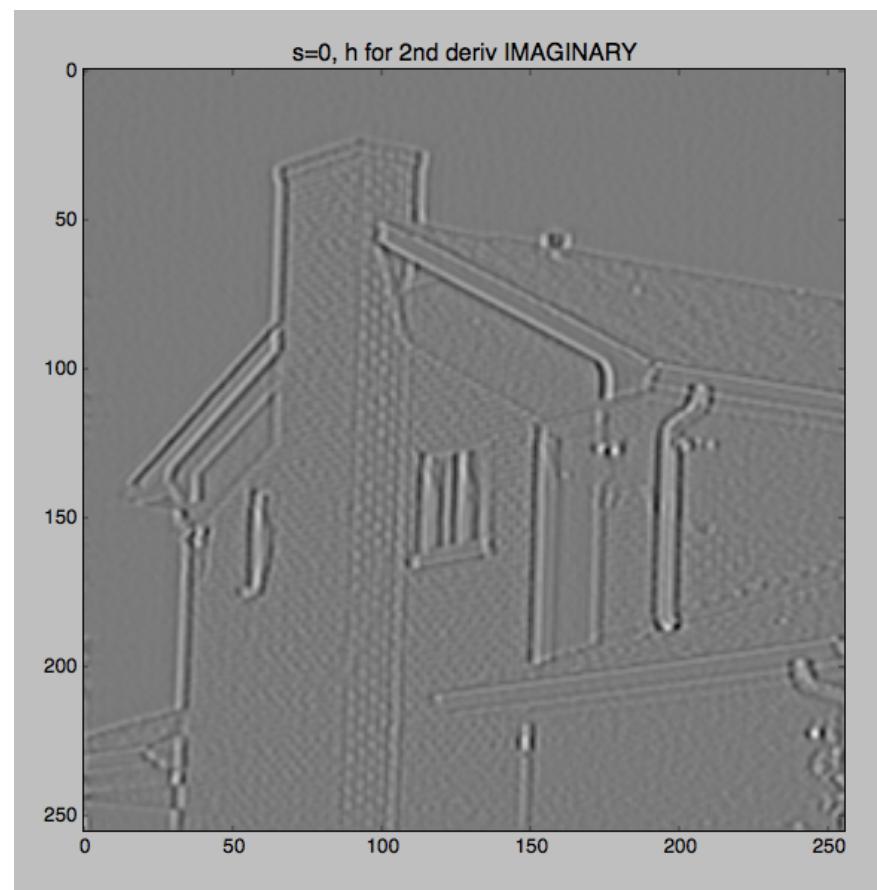
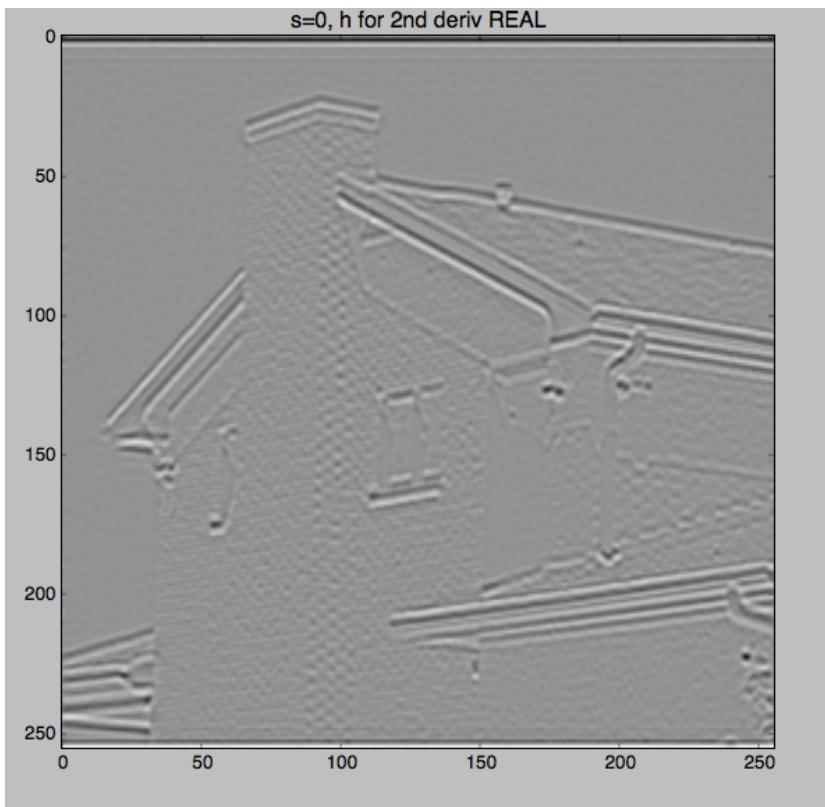


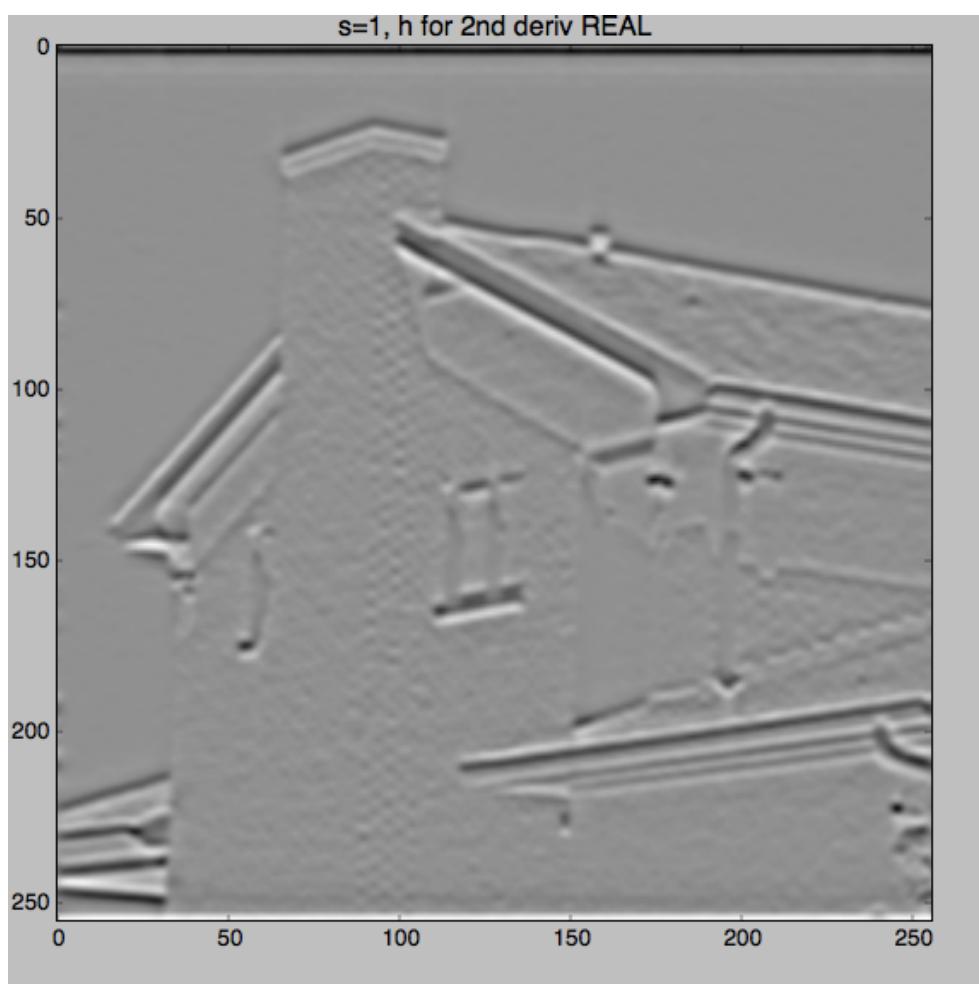
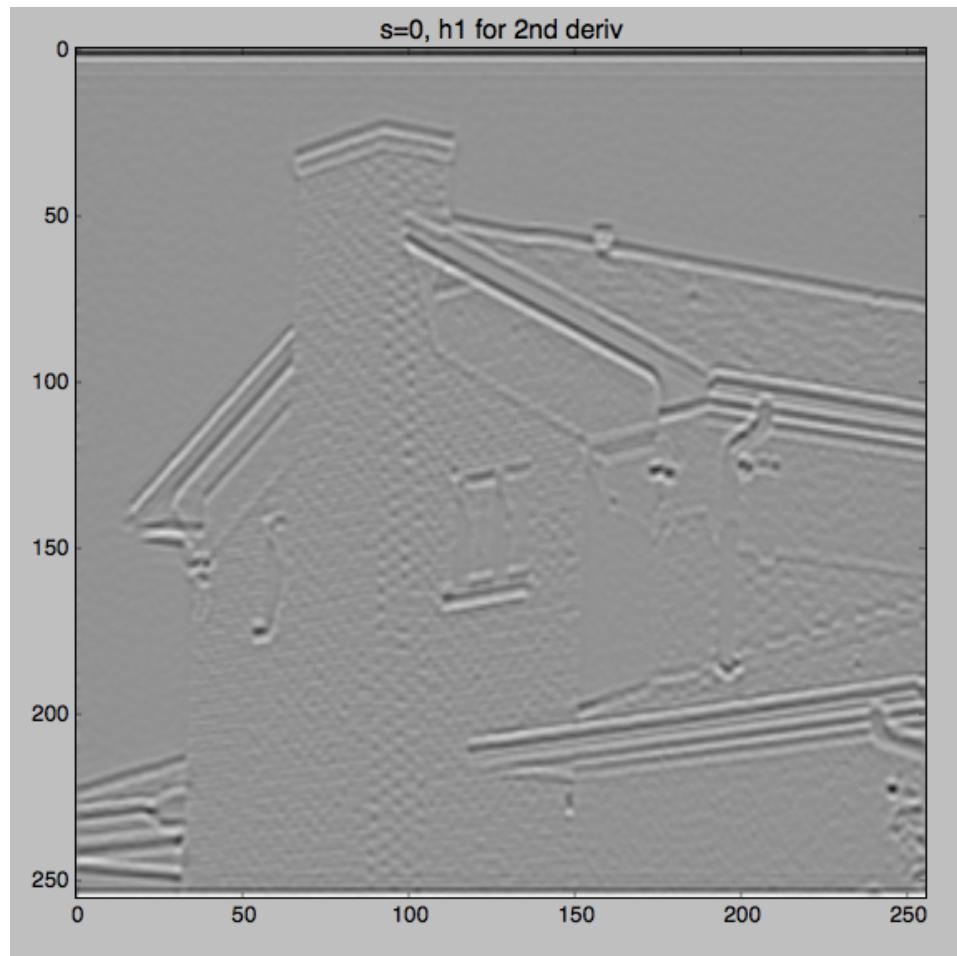
s=3, An_Y



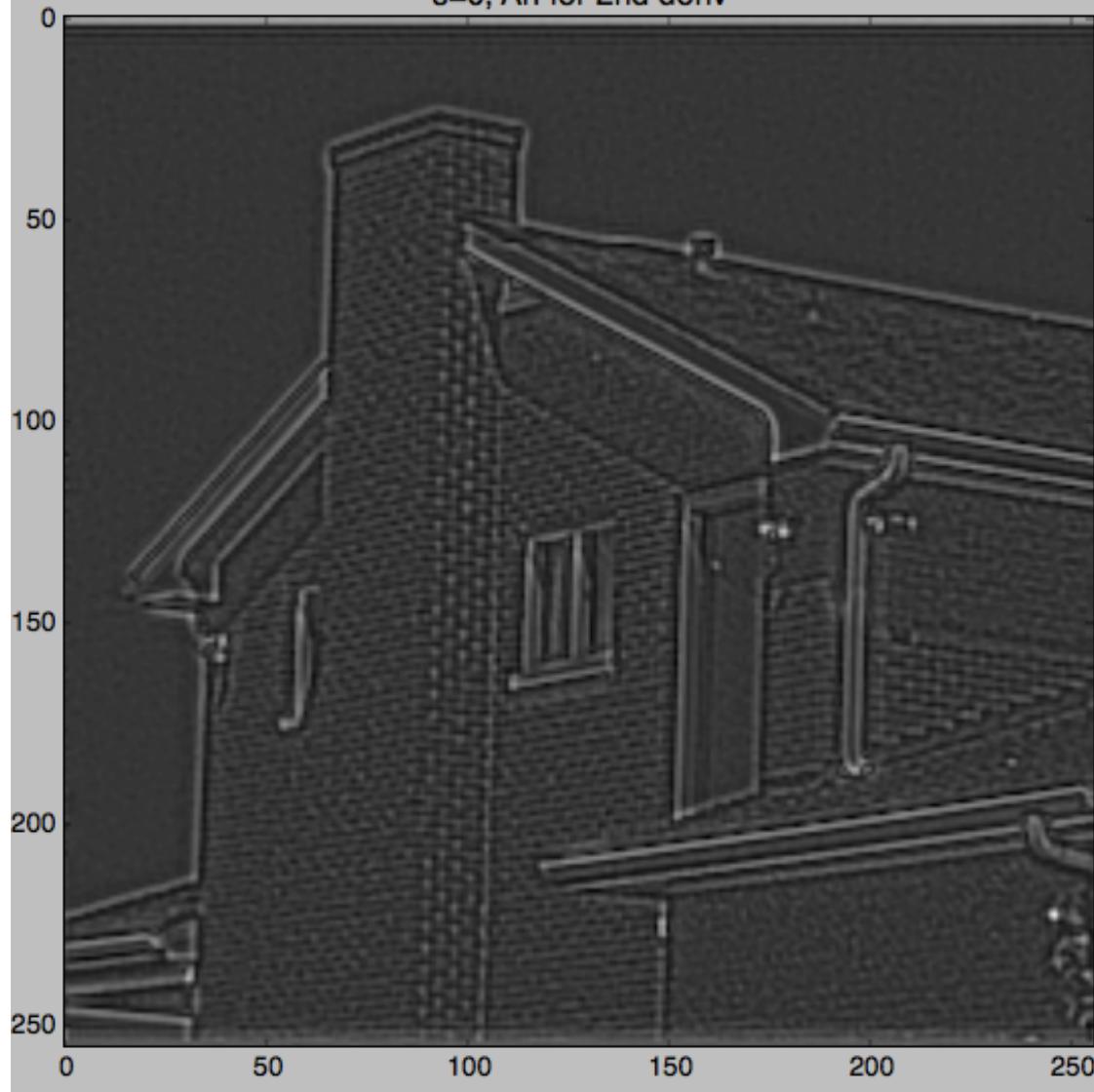




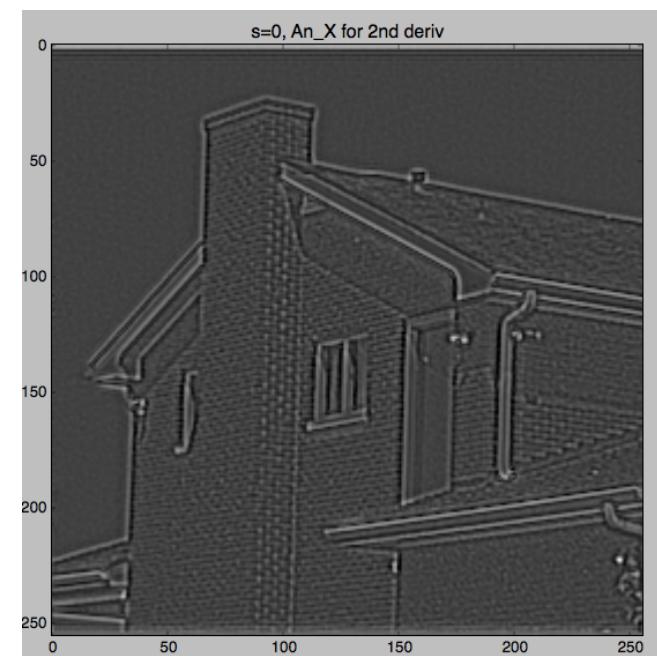




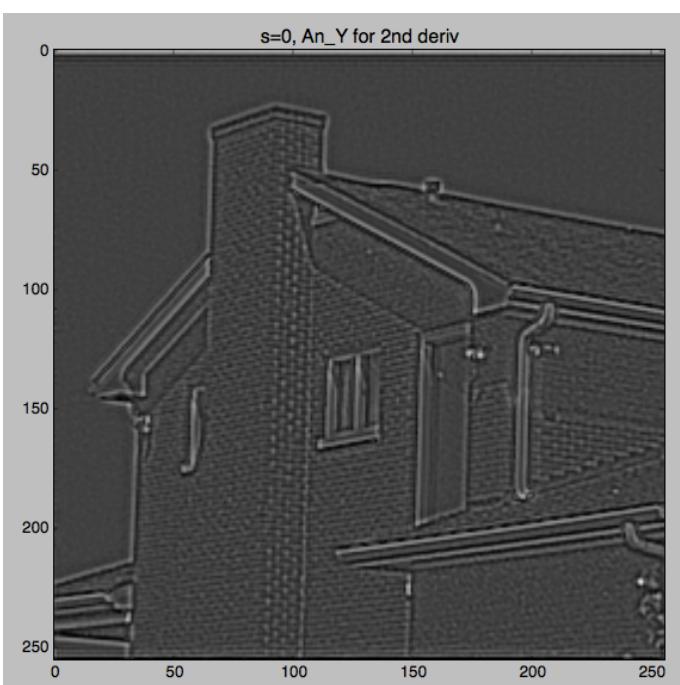
s=0, An for 2nd deriv



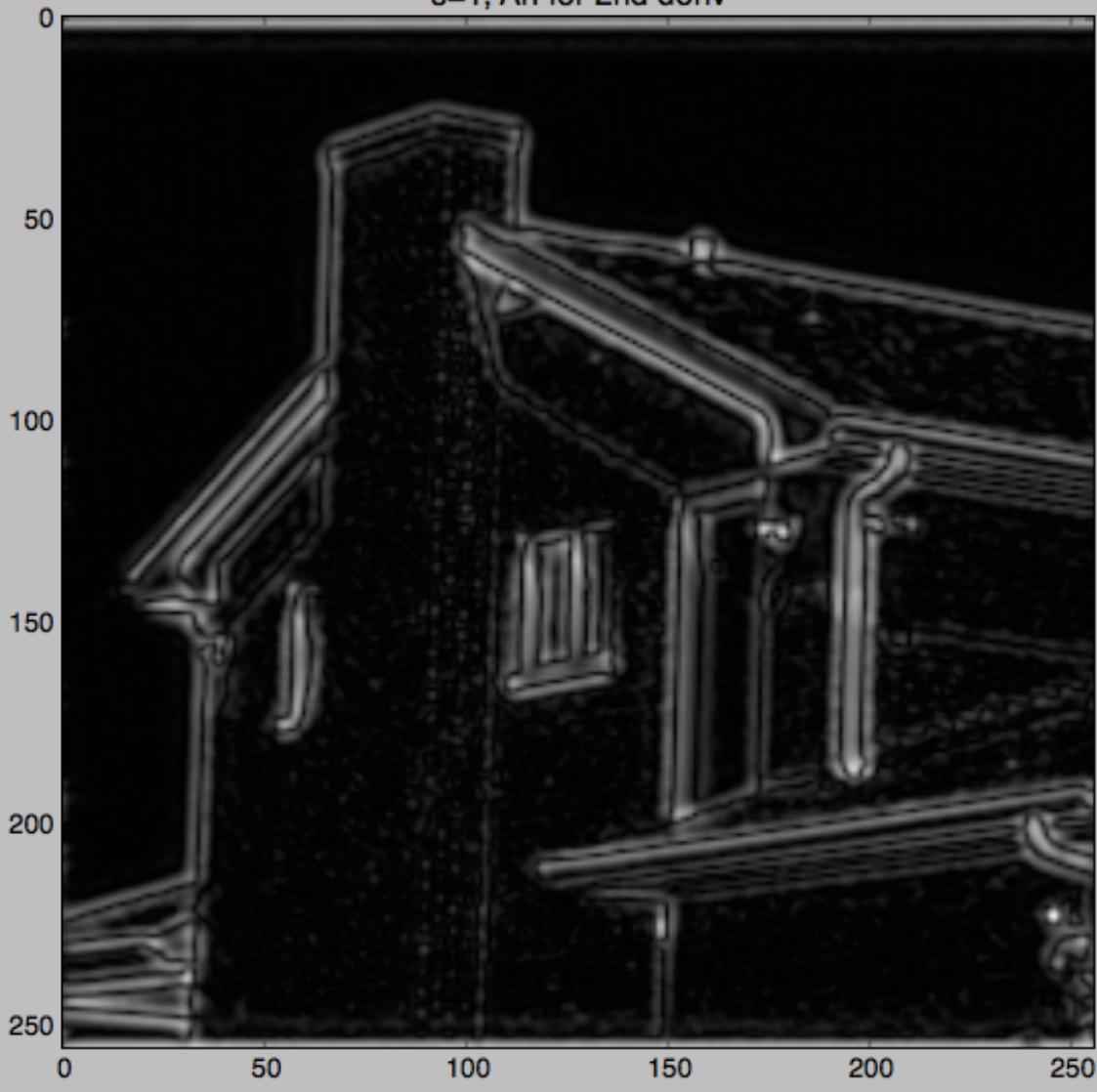
s=0, An_X for 2nd deriv



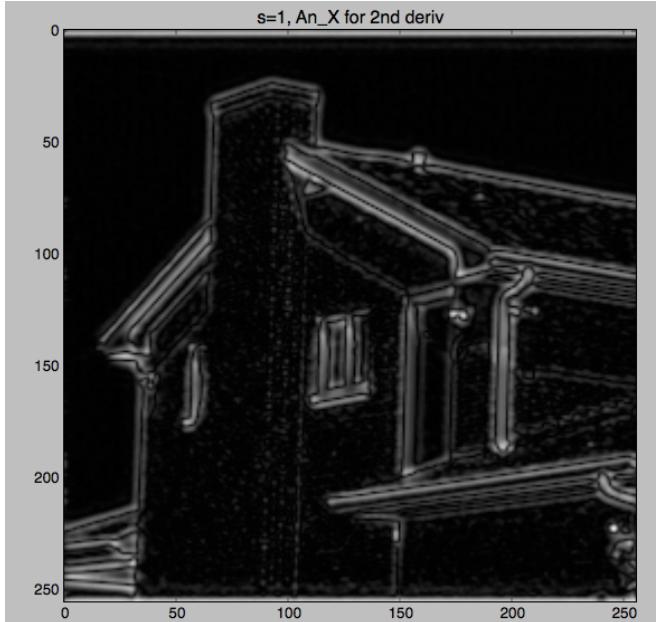
s=0, An_Y for 2nd deriv



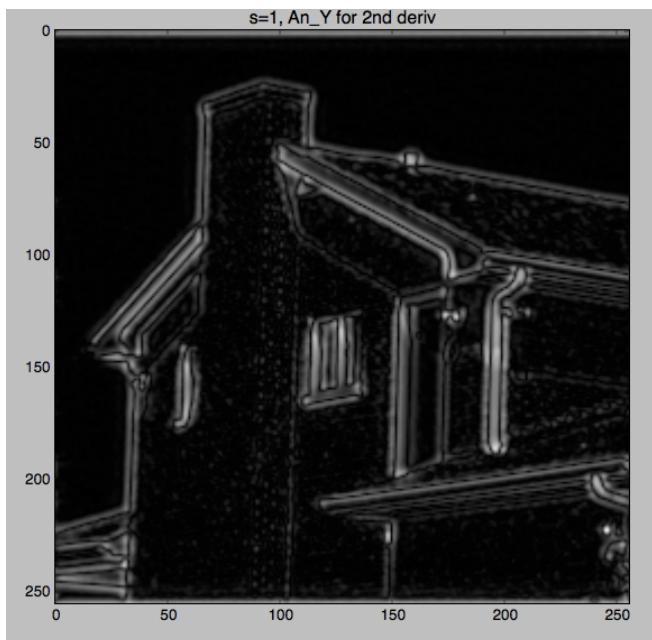
s=1, An for 2nd deriv



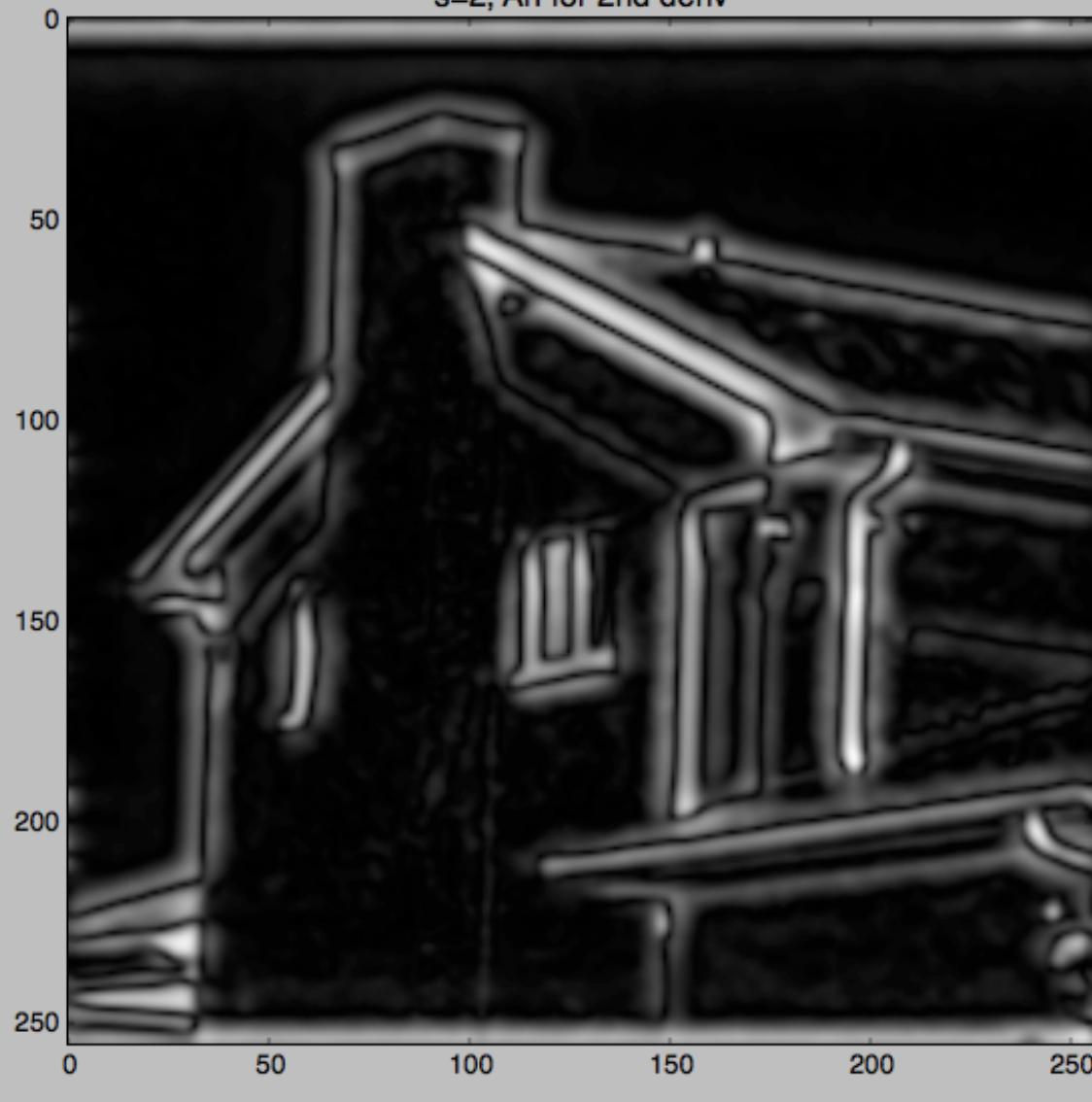
s=1, An_X for 2nd deriv



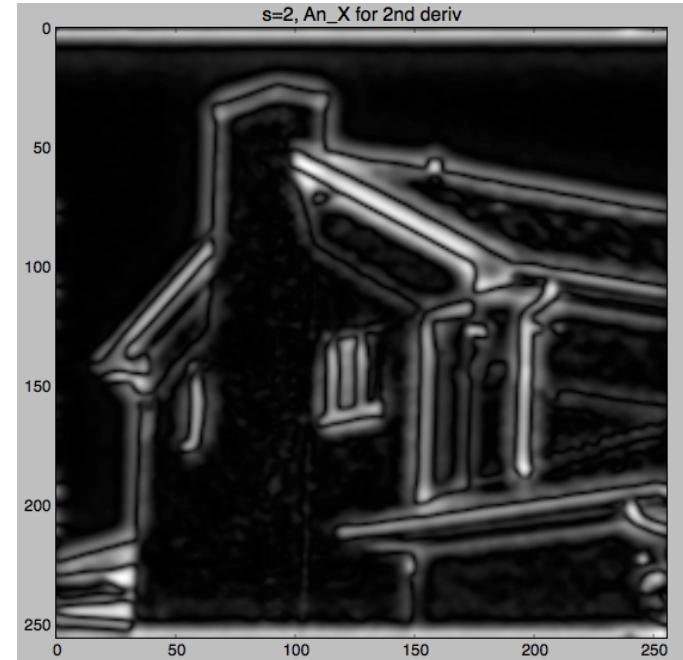
s=1, An_Y for 2nd deriv



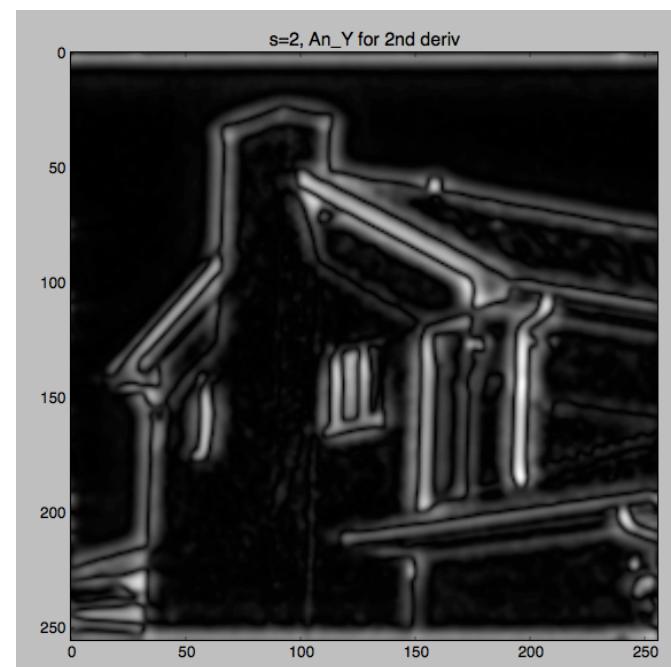
s=2, An for 2nd deriv

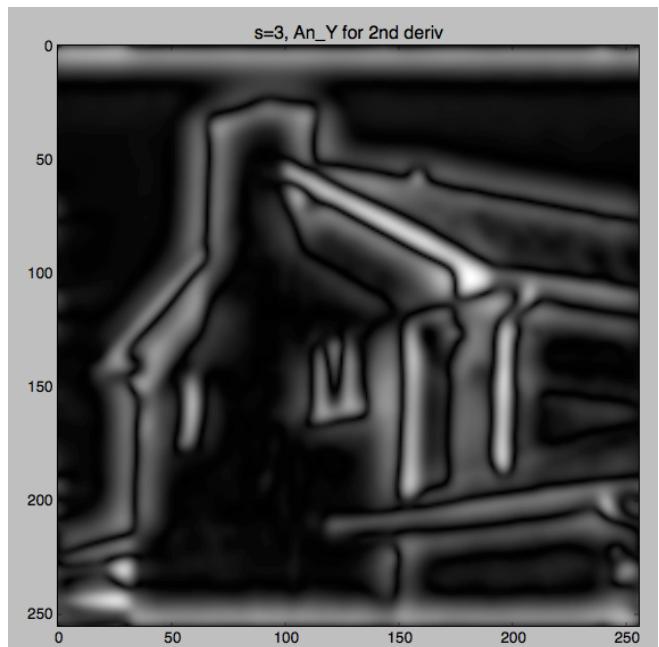
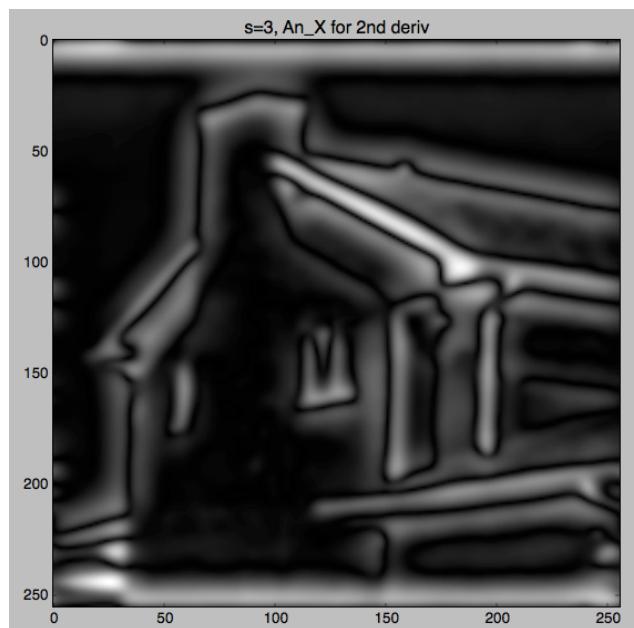
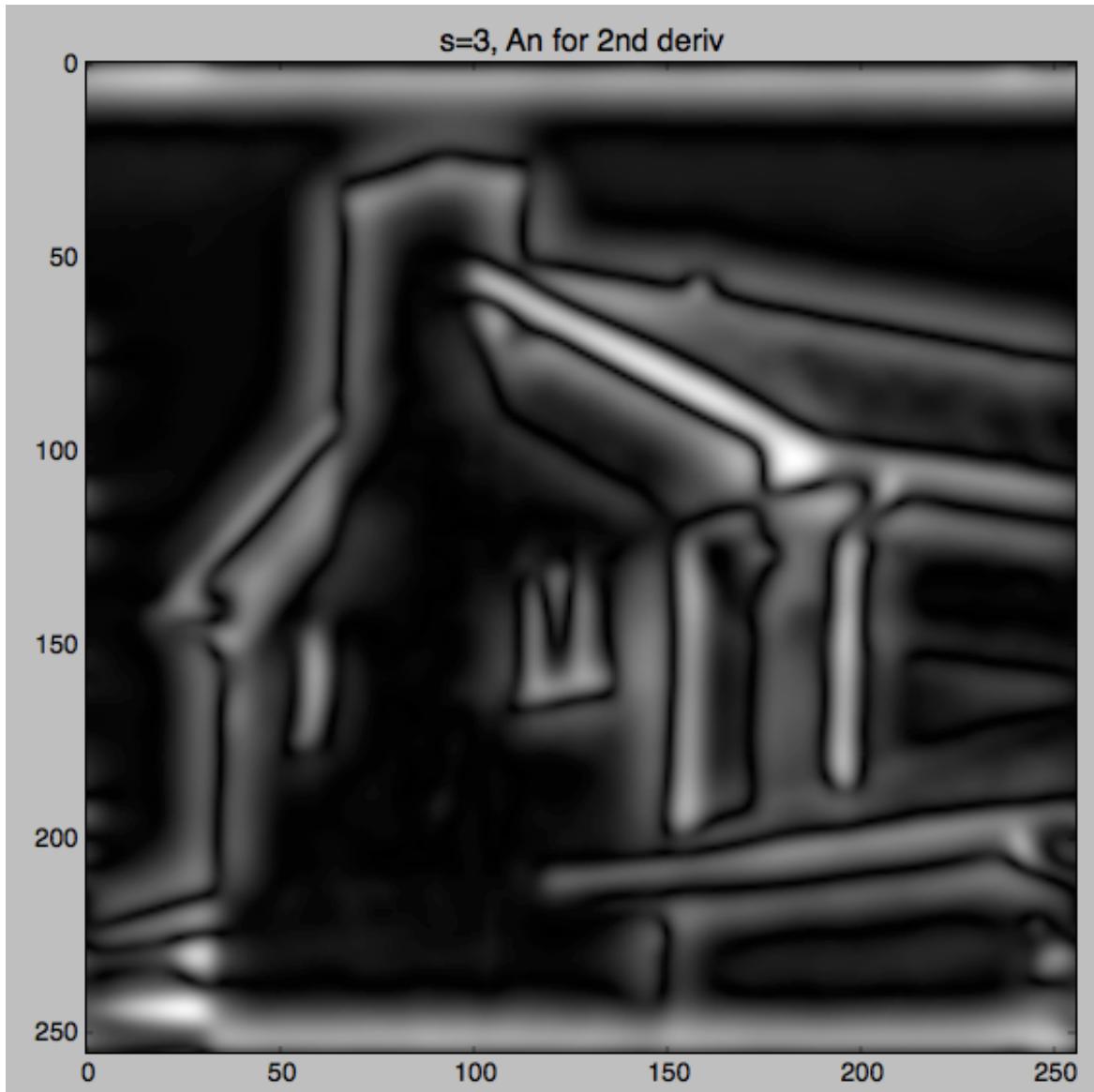


s=2, An_X for 2nd deriv

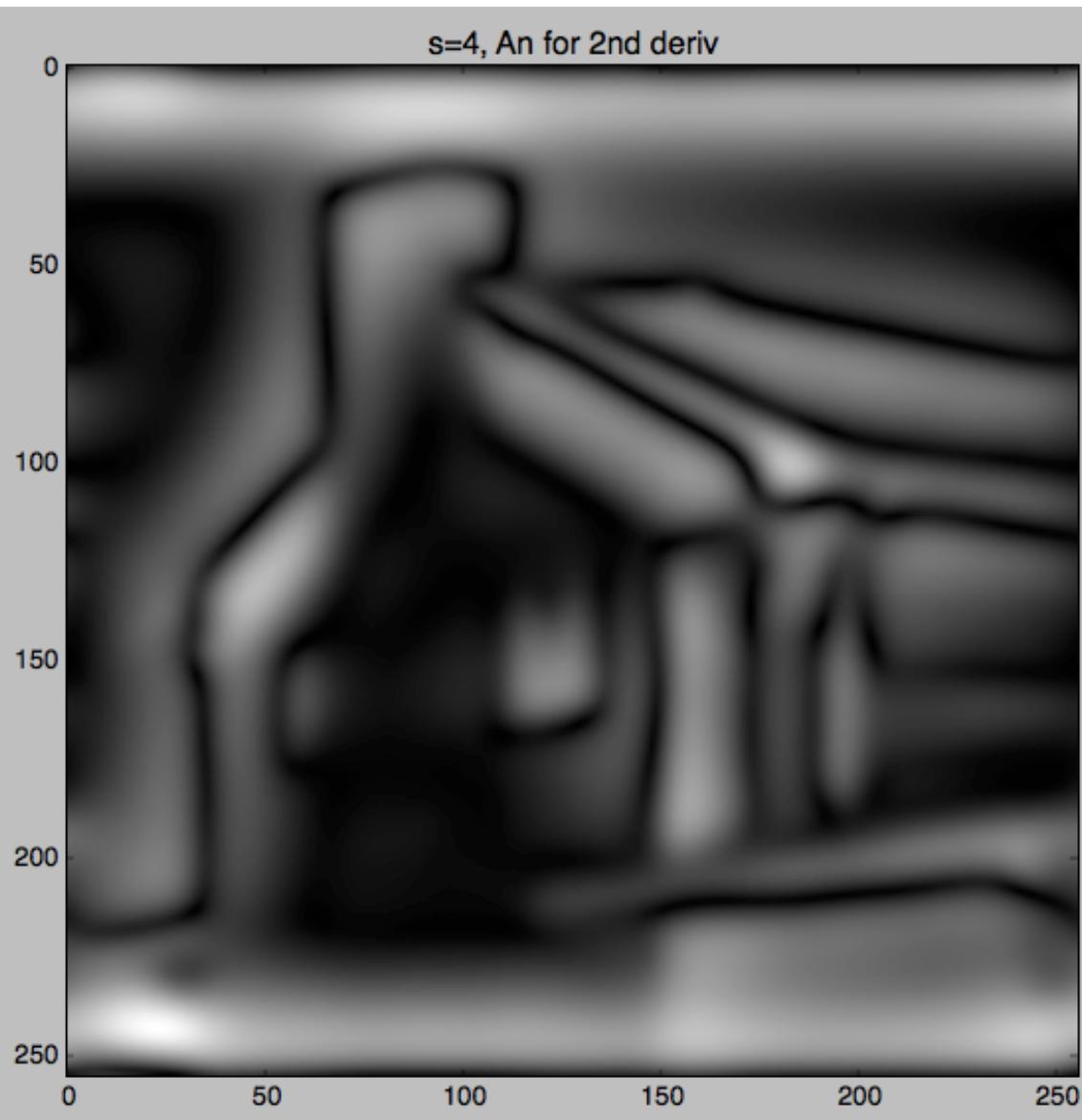


s=2, An_Y for 2nd deriv

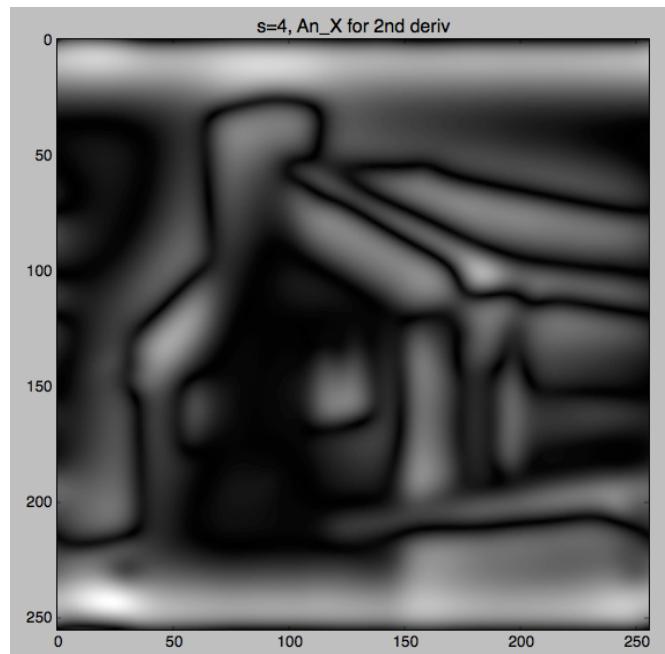




s=4, An for 2nd deriv



s=4, An_X for 2nd deriv



s=4, An_Y for 2nd deriv

