Cleaning up MPRAGEs collected with prescan normalize

Import image loading functions

In [1]:
```
from nibabel import load, Nifti1Image
```

Load in the uncorrected image

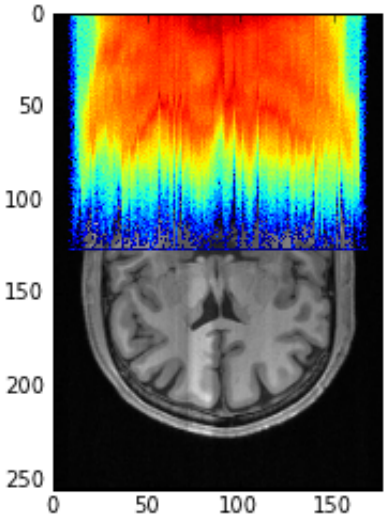
In [2]:
```
img = load('006-MEMPRAGE_4e_p2_1mm_iso.nii.gz')
data = img.get_data()
```

Compute and overlay histograms on the image data

In [3]:
```
hgrams = np.zeros((data.shape[0], 128))
for i in range(data.shape[0]):
    foo = data[i, :, :]
    x, bins = histogram(foo[foo>100], 128)
    hgrams[i,:] = log(x)
clf()
imshow(data[:,128,:].T, cmap=cm.gray)
imshow(hgrams.T, interpolation='nearest')
imshow(data[:,128,:].T, alpha=0)
```

```
-c:5: RuntimeWarning: divide by zero encountered in log
```

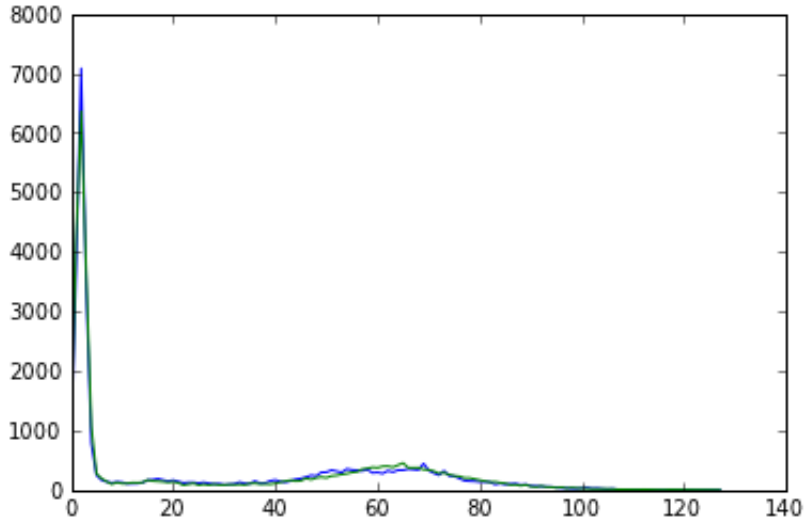
Out[3]:
```
<matplotlib.image.AxesImage at 0x10b2670d0>
```



Compute histograms to show that there is not really much intensity distribution difference between the off slices.

In [4]:
```
h1,_ = histogram(data[80,:,128:].ravel(),128)
h2,_ = histogram(data[105,:,128:].ravel(),128)
plot(vstack((h1,h2)).T)
```

Out[4]:
```
[<matplotlib.lines.Line2D at 0x10b42e450>,
 <matplotlib.lines.Line2D at 0x10b42e5d0>]
```

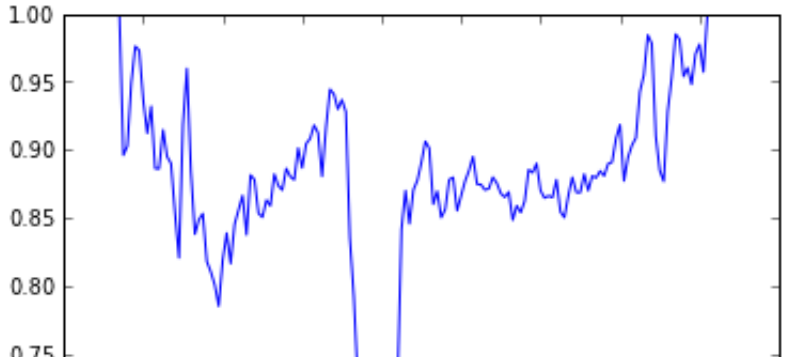


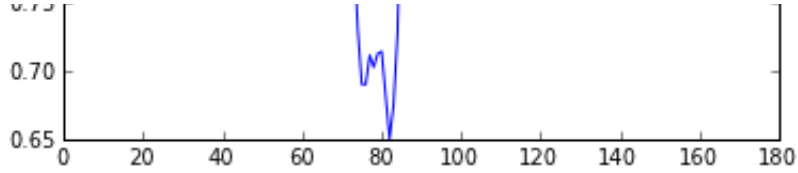
The correction algorithm

Compute an intensity profile cutting across sagittal slices and determine a scaling factor for each slice. You can vary the 0.85 to determine a best correction.

In [5]:
```
wmout = np.zeros((10, 176))
correction_factor = 0.85
axial_slices = range(115, 125)
axial_profile_slices = range(150, 190)
for idx, sl in enumerate(axial_slices):
    wm = np.max(data[:, sl, axial_profile_slices], axis=1).astype('float')
    wm = correction_factor*(wm/np.max(wm))-0.5
    wm = 1-wm*(wm>0)
    wmout[idx,:] = wm
wm = np.median(wmout, axis=0)
plot(wm)
```

Out[5]:
```
[<matplotlib.lines.Line2D at 0x10b522510>]
```

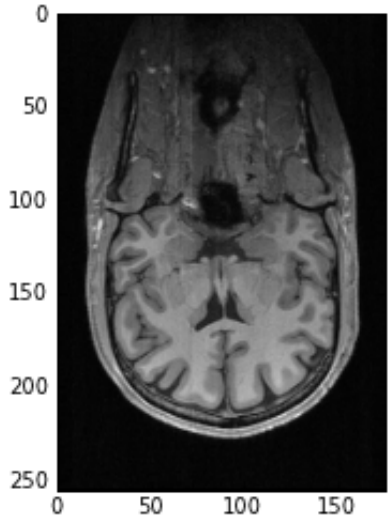




Apply scaling factor to each slice.

In [6]:
```
data2 = np.zeros(data.shape)
for i in range(data.shape[0]):
    data2[i,:,:] = wm[i]*data[i, :, :]
imshow(data2[:, 128, :].T, cmap=cm.gray)
```

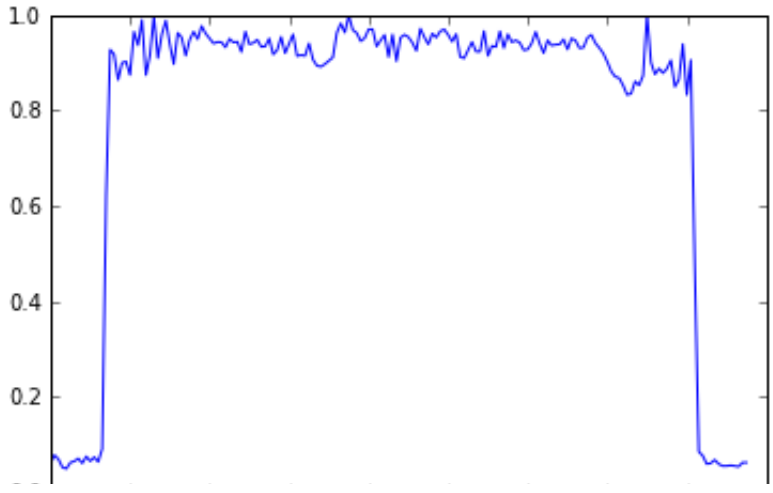
Out[6]: <matplotlib.image.AxesImage at 0x10b558f90>

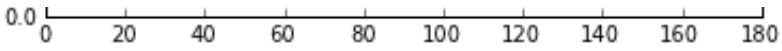


Display intensity profile after correction

In [7]:
```
wm = np.max(data2[:, 119, 150:190], axis=1).astype('float')
wm = (wm/np.max(wm))
#wm = 1-wm*(wm>0)
plot(wm)
```

Out[7]: [<matplotlib.lines.Line2D at 0x10b356650>]





Write out corrected image.

In [8]:
```
outimg = Nifti1Image(data2, img.get_affine(), img.get_header())
outimg.to_filename('corrected.nii.gz')
```