The Canny edge detector with scikit-image

The Canny edge detector is a popular edge detection algorithm, developed by John F. Canny. This algorithm has the following multiple steps:

1. **Smoothing/noise reduction**: The edge detection operation is sensitive to noise. Hence, at the very outset, a 5 x 5 Gaussian filter is used to remove noise from the image.
2. **Computing magnitude and orientation of the gradient**: The Sobel horizontal and vertical filters are then applied to the image to compute the edge gradient **magnitude** and **direction** for each pixel, as discussed previously. The gradient angle (direction) computed is then rounded to one of four angles representing horizontal, vertical, and two diagonal directions for each pixel.
3. **Non-maximum suppression**: In this step, the edges are thinned – any unwanted pixels which may not constitute an edge are removed. To do this, every pixel is checked if it is a local maximum in its neighborhood in the direction of gradient. As a result, a binary image is obtained with thin edges.

4. **Linking and hysteresis thresholding**: In this step, whether all the edges detected are strong edges or not is decided. For this, a couple of (hysteresis) threshold values, `min_val` and `max_val`, are used. Sure edges are the ones that have an intensity gradient value higher than `max_val`. Sure non-edges are the ones that have an intensity gradient value below `min_val`, and they are discarded. The edges that lie between these two thresholds are classified as edges or non-edges, based on their connectivity. If they are connected to sure-edge pixels, they are considered to be part of edges. Otherwise, they are also discarded. This step also removes small pixel noise (assuming that the edges are long lines).

So finally, the algorithm outputs the strong edges of the image. The following code block shows how the Canny edge detector can be implemented with `scikit-image`:

```
im = rgb2gray(imread('../images/tiger3.jpg'))
im = ndimage.gaussian_filter(im, 4)
im += 0.05 * np.random.random(im.shape)
edges1 = feature.canny(im)
edges2 = feature.canny(im, sigma=3)
fig, (axes1, axes2, axes3) = pylab.subplots(nrows=1, ncols=3, figsize=(30, 12), sharex=Tru
axes1.imshow(im, cmap=pylab.cm.gray), axes1.axis('off'), axes1.set_title('noisy image', fo
axes2.imshow(edges1, cmap=pylab.cm.gray), axes2.axis('off')
axes2.set_title('Canny filter, $\sigma=1$', fontsize=50)
axes3.imshow(edges2, cmap=pylab.cm.gray), axes3.axis('off')
axes3.set_title('Canny filter, $\sigma=3$', fontsize=50)
fig.tight_layout()
pylab.show()
```

The following screenshot shows the output of the previous code; the edges are detected with the Canny filter with different sigma values for the initial Gaussian LPF. As can be seen, with a lower value of sigma, the original image gets less blurred to start with and hence more edges (finer details) can be found: