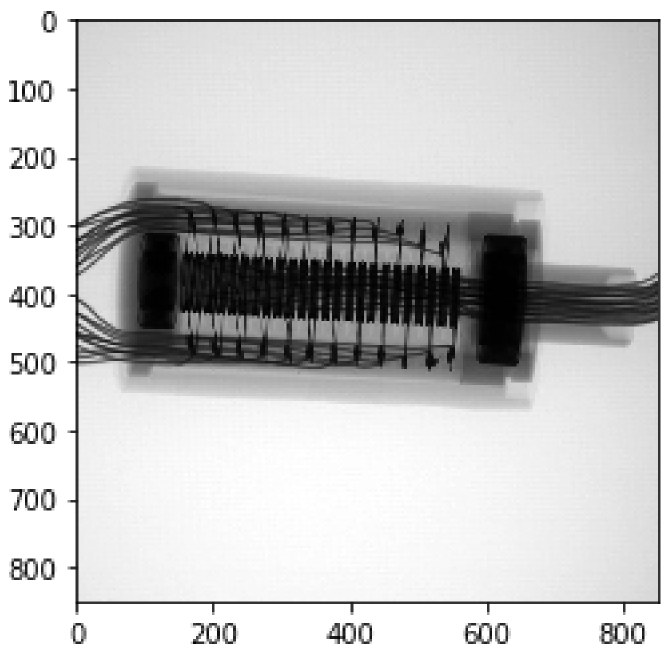
filteres

January 19, 2018

```
In [1]: %matplotlib inline
        import matplotlib.pyplot as plt

        from skimage import data, img_as_ubyte
        from skimage.io import imread
        from skimage.filters import rank, gaussian, median, roberts, sobel, prewitt, laplace
        from skimage.morphology import square
        from skimage.util import random_noise
        from skimage.feature import blob_log

        from ipywidgets import interact, interactive, fixed, interact_manual
        import ipywidgets as widgets
        from IPython.display import display

        import numpy as np

        plt.rcParams['image.cmap'] = 'gray'
        plt.rcParams['image.interpolation'] = 'none'

        image = img_as_ubyte(imread('https://www.eledus.cz/wp-content/uploads/2017/05/RTG-Krou
        plt.imshow(image)
```

```
C:\Users\Petr\Anaconda3\lib\site-packages\skimage\util\dtype.py:122: UserWarning: Possible pre
  .format(dtypeobj_in, dtypeobj_out))
```

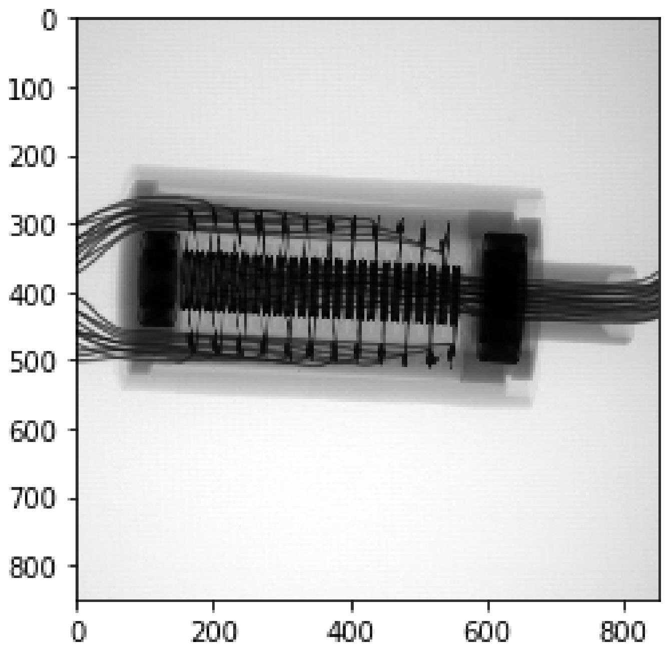
```
Out[1]: <matplotlib.image.AxesImage at 0x2ca37970d30>
```



0.1 Mean filter

Move slider to change size of the pixel neighbour

```
In [2]: plt.imshow(image)

        mean_slider = widgets.IntSlider(min=1,max=30,step=1,value=1, continuous_update=False)

        @interact(neigh=mean_slider)
        def mean_display(neigh):
            plt.imshow(rank.mean(image, selem=square(neigh)))
```

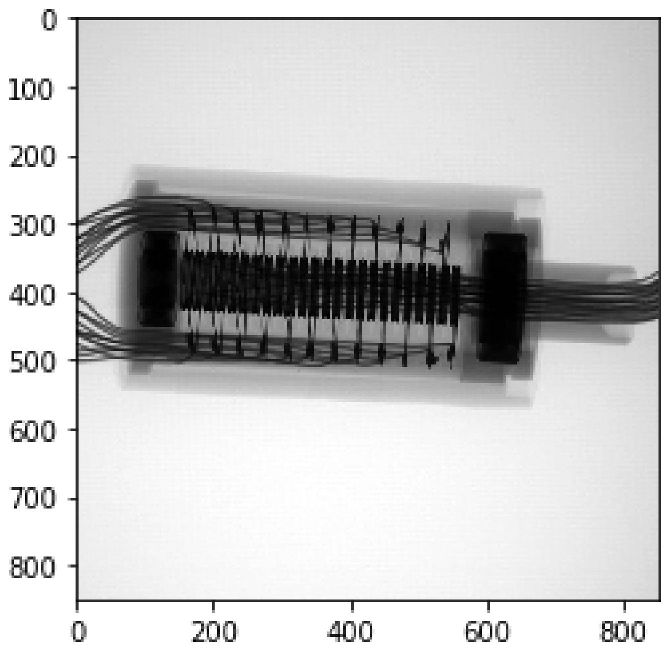


```
A Jupyter Widget
```

0.2 Gaussian filter

Move slider to change sigma

```
In [3]: plt.imshow(image)

        gaussian_slider = widgets.FloatSlider(min=0.1,max=10,step=0.1,value=0.1,continuous_upda

        @interact(sigma=gaussian_slider)
        def gaussian_display(sigma):
            plt.imshow(gaussian(image, sigma))
```



A Jupyter Widget

0.3 Median filter

Move sliders to apply median filter and noise on original image

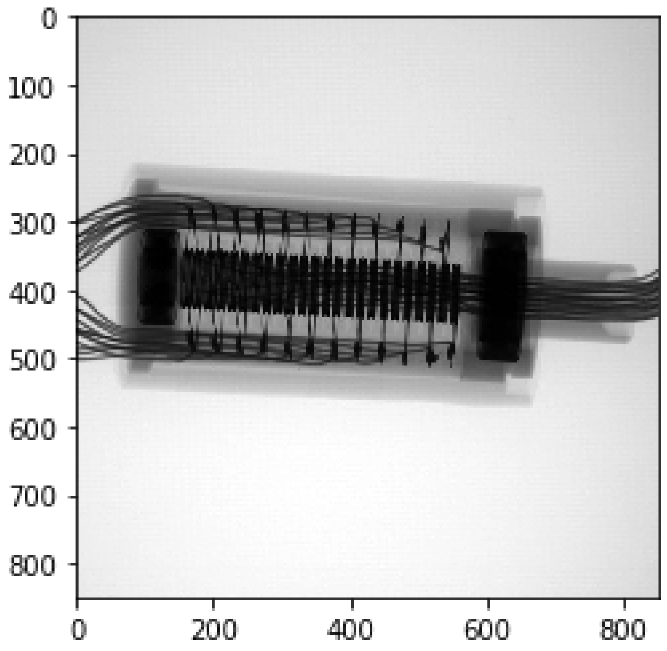
```
In [4]: noise_slider = widgets.FloatSlider(min=0.0,max=1,step=0.05,value=0.1,continuous_update
        median_slider = widgets.IntSlider(min=1,max=10,step=1,value=1,continuous_update=False)

        @interact(noise_level=noise_slider)
        def noise_display(noise_level):
            noised_image = random_noise(image, amount=noise_level, mode='s&p')
            plt.imshow(noised_image)

            @interact(neigh=median_slider)
            def median_display(neigh):
                plt.imshow(median(noised_image, selem=square(neigh)))
```

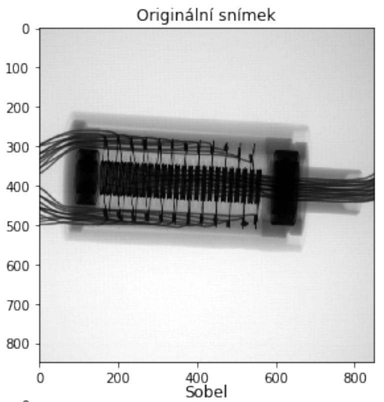
A Jupyter Widget

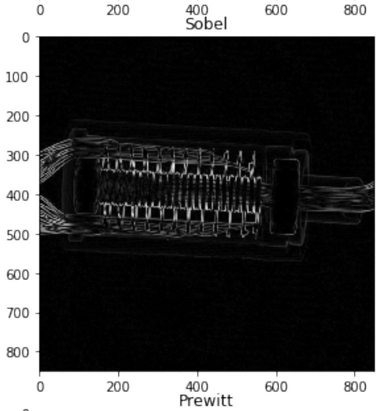
1 Edge detection methods

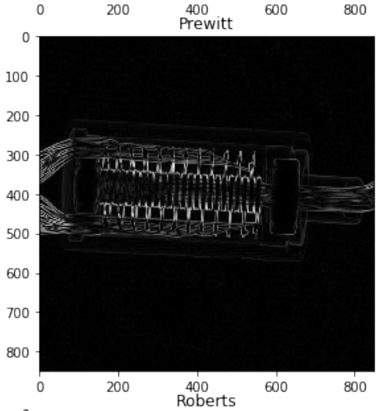
```
In [5]: plt.imshow(image)
        f, ax = plt.subplots(5, 1, figsize=(20,20))
        f.tight_layout()
        ax[0].imshow(image)
        ax[0].set_title('Originální snímek')

        ax[1].imshow(sobel(image))
        ax[1].set_title('Sobel')

        ax[2].imshow(prewitt(image))
        ax[2].set_title('Prewitt')

        ax[3].imshow(roberts(image))
        ax[3].set_title('Roberts')

        ax[4].imshow(laplace(image))
        ax[4].set_title('Laplace')
```

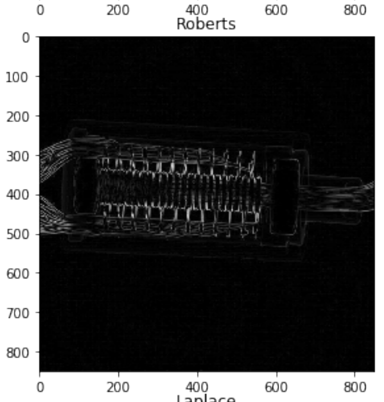
```
Out[5]: <matplotlib.text.Text at 0x2ca38c69e48>
```

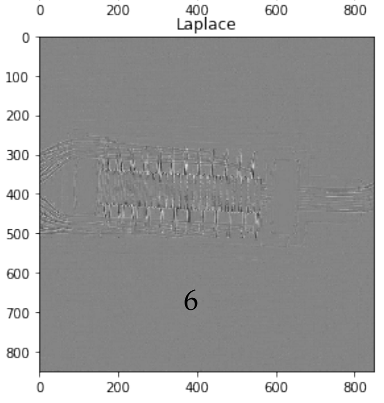



Originální snímek
0
100
200
300
400
500
600
700
800


Sobel


Prewitt


Roberts


Laplace