Image feature extraction

In [1]:

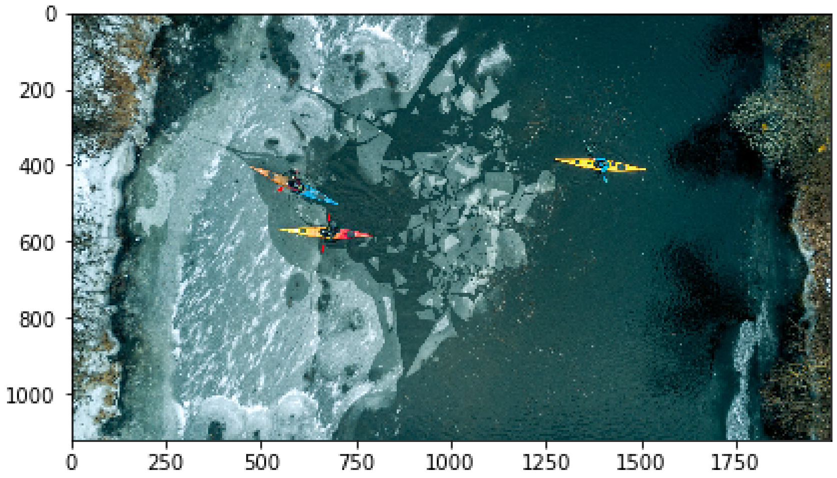
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from skimage.io import imread, imshow
```

In [2]:

```
image1 = imread('image1.jpg')
imshow(image1)
```

Out[2]:

```
<matplotlib.image.AxesImage at 0x1ea89cb0520>
```

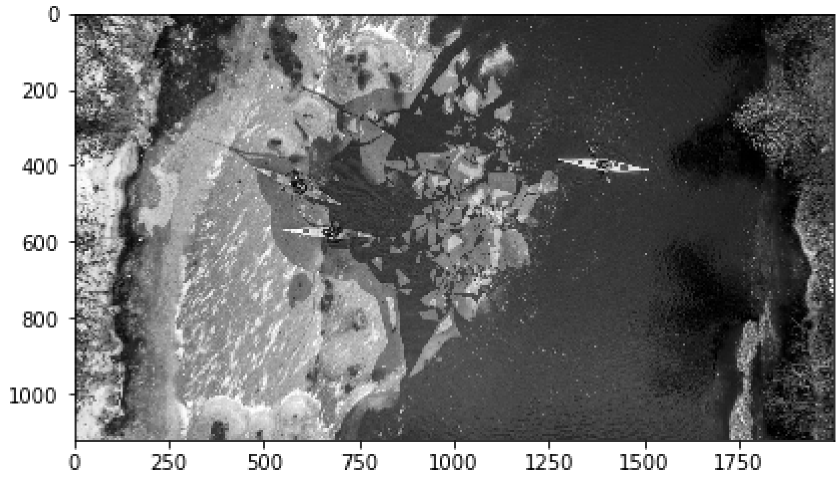


In [3]:

```
image2 = imread('image1.jpg', as_gray=True)
imshow(image2)
```

Out[3]:

```
<matplotlib.image.AxesImage at 0x1ea8a3d2d00>
```



In [4]:
```
print(image1.shape)
print(image2.shape)
```

```
(1121, 1996, 3)
(1121, 1996)
```

In [9]:
```
print(image1.size)
print(image2.size)
```

```
6712548
2237516
```

In [14]:
```
pixel_feat1 = np.reshape(image2, (1121 * 1996))
pixel_feat1
```

Out[14]:
```
array([0.24580353, 0.19090157, 0.17521529, ..., 0.2694651 , 0.21456314,
       0.09748157])
```

In [15]:
```
pixel_feat2 = np.reshape(image1, (1121 * 1996 * 3))
pixel_feat2
```

Out[15]:
```
array([45, 68, 62, ..., 23, 26, 19], dtype=uint8)
```

Prewitt

In [20]:
```
from skimage.filters import prewitt_h
from skimage.filters import prewitt_v
```

In [21]:
```
pre_hor = prewitt_h(image2)
pre_ver = prewitt_v(image2)
```

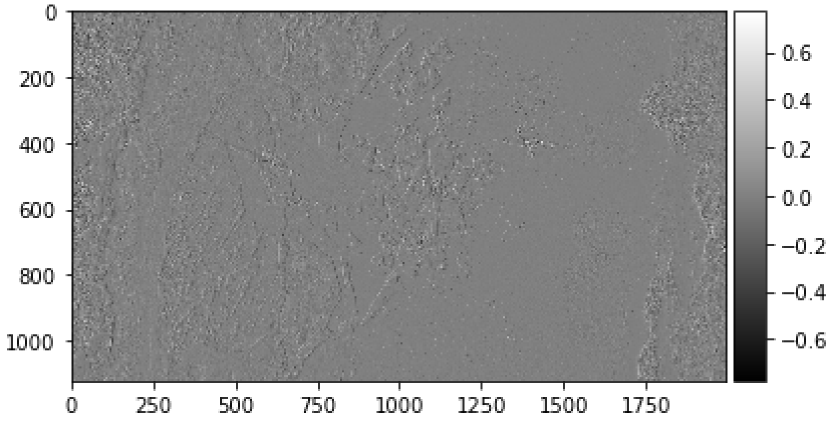
In [22]:
```
ed_sobel = filters.sobel(image2)
```

In [25]:
```
from skimage import feature
```

In [26]:
```
can = feature.canny(image2)
```

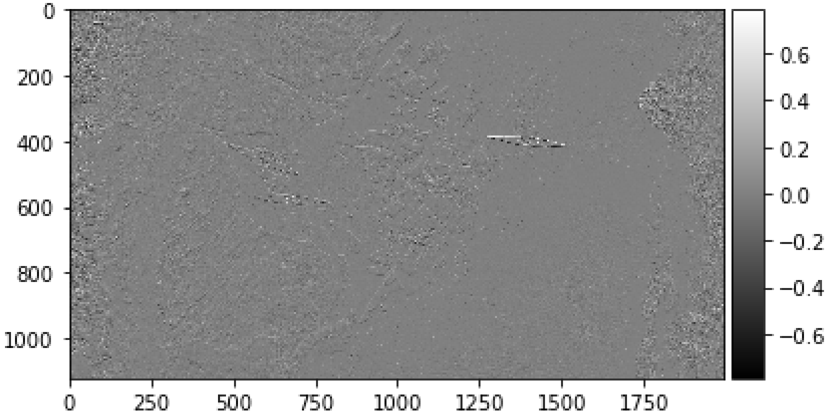
In [28]:
```
imshow(pre_ver, cmap='gray')
```

Out[28]:
```
<matplotlib.image.AxesImage at 0x1ea8f87af40>
```



In [29]:
```
imshow(pre_hor, cmap='gray')
```

Out[29]:
```
<matplotlib.image.AxesImage at 0x1ea8ffe6eb0>
```

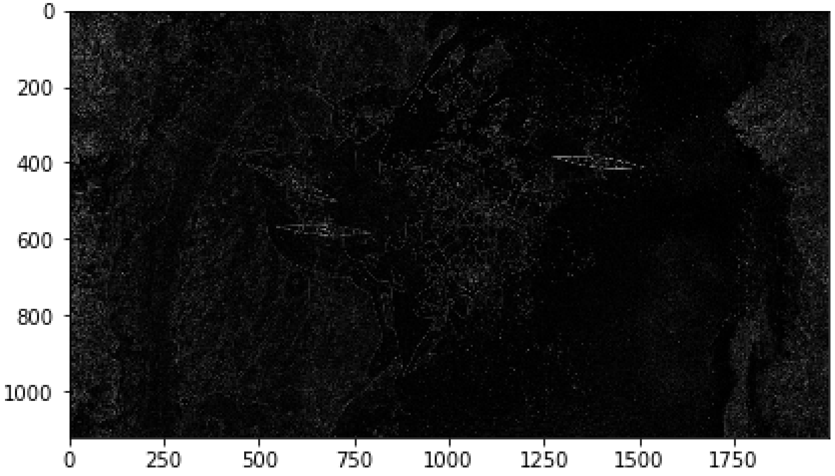


In [30]:

```
imshow(ed_sobel, cmap='gray')
```

Out[30]:

```
<matplotlib.image.AxesImage at 0x1ea90079220>
```

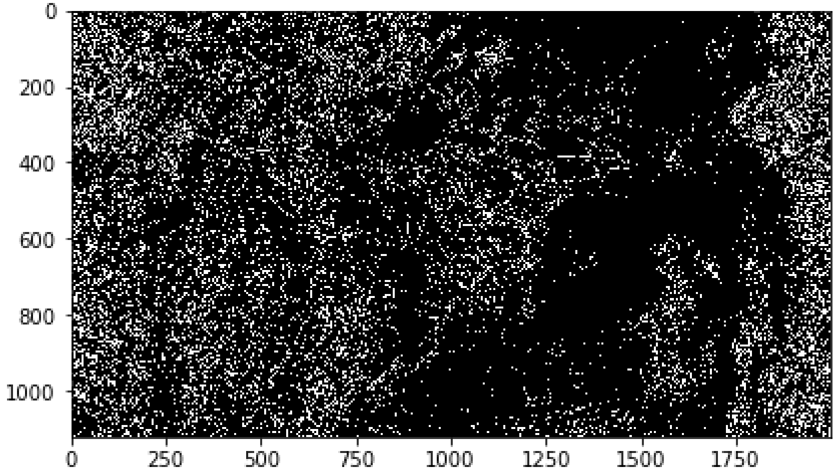


In [32]:

```
imshow(can, cmap='gray')
```

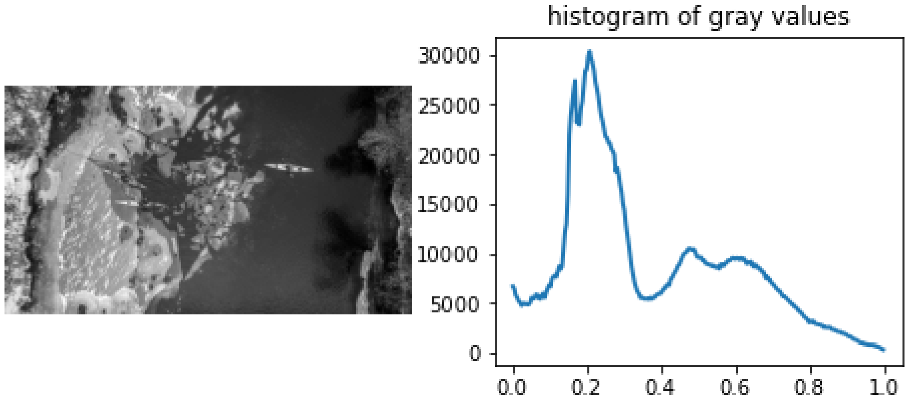
Out[32]:

```
<matplotlib.image.AxesImage at 0x1ea900c54f0>
```



In [33]:
```
from skimage.exposure import histogram
hist, hist_centers = histogram(image2)

#Plotting the Image and the Histogram of gray values
fig, axes = plt.subplots(1, 2, figsize=(8, 3))
axes[0].imshow(image2, cmap=plt.cm.gray)
axes[0].axis('off')
axes[1].plot(hist_centers, hist, lw=2)
axes[1].set_title('histogram of gray values')
```

Out[33]:
```
Text(0.5, 1.0, 'histogram of gray values')
```



PCA

In [5]:

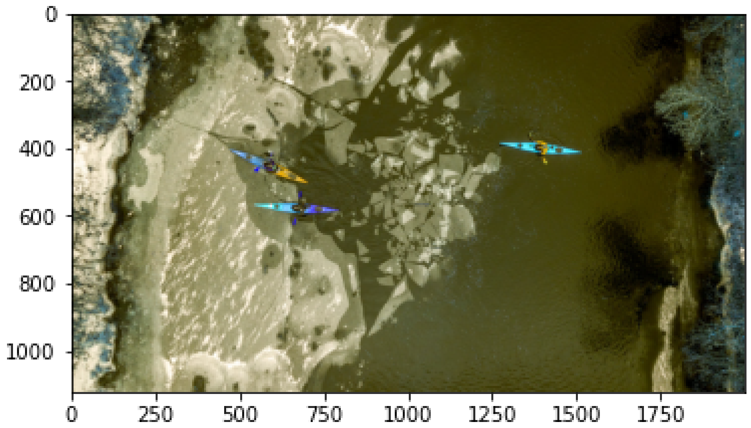
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
```

In [6]:

```
img = cv2.imread('image1.jpg') #you can use any image you want.
plt.imshow(img)
```

Out[6]:

```
<matplotlib.image.AxesImage at 0x29e19484190>
```



In [7]:

```
# Splitting the image in R,G,B arrays.

blue,green,red = cv2.split(img)
#it will split the original image into Blue, Green and Red arrays.
```

In [8]:

```
pca = PCA(20)

#Applying to red channel and then applying inverse transform to transformed array.
red_transformed = pca.fit_transform(red)
red_inverted = pca.inverse_transform(red_transformed)

#Applying to Green channel and then applying inverse transform to transformed array.
green_transformed = pca.fit_transform(green)
```

```
green_inverted = pca.inverse_transform(green_transformed)

#Applying to Blue channel and then applying inverse transform to transformed array.
blue_transformed = pca.fit_transform(blue)
blue_inverted = pca.inverse_transform(blue_transformed)
```

In [9]:

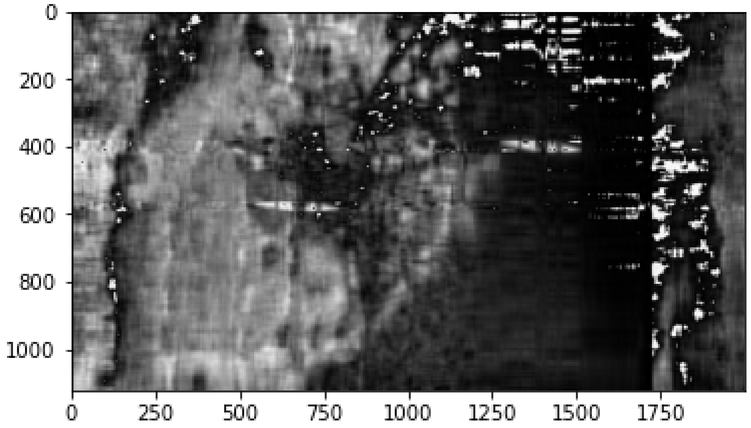
```
img_compressed = (np.dstack((red_inverted, red_inverted, red_inverted))).astype(np.uint8)
```

In [10]:

```
#viewing the compressed image
plt.imshow(img_compressed)
```

Out[10]:

```
<matplotlib.image.AxesImage at 0x29e1a000730>
```



In [11]:

```
pca = PCA(200)
```

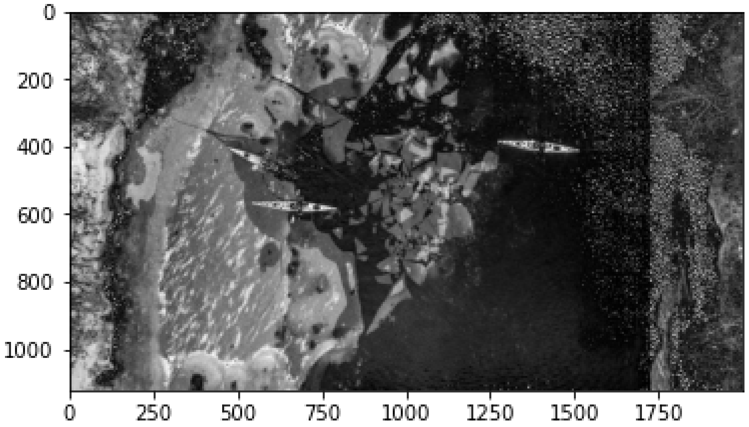
In [12]:

```
#Applying to red channel and then applying inverse transform to transformed array.
red_transformed = pca.fit_transform(red)
red_inverted = pca.inverse_transform(red_transformed)

#Applying to Green channel and then applying inverse transform to transformed array.
green_transformed = pca.fit_transform(green)
green_inverted = pca.inverse_transform(green_transformed)

#Applying to Blue channel and then applying inverse transform to transformed array.
blue_transformed = pca.fit_transform(blue)
blue_inverted = pca.inverse_transform(blue_transformed)
```

```
In [13]: img_compressed = (np.dstack((red_inverted, red_inverted, red_inverted))).astype(np.uint8)
```

```
In [14]: plt.imshow(img_compressed)
```

```
Out[14]: <matplotlib.image.AxesImage at 0x29e1a069220>
```



ICA

```
In [15]: from sklearn.decomposition import FastICA
         from pylab import *
         from skimage import data, io, color
```

```
In [18]: image = io.imread("image1.jpg", as_gray = True)
```

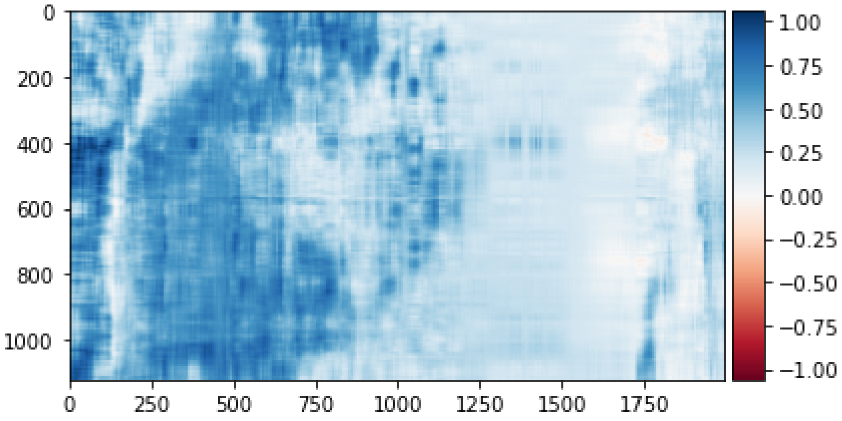
```
In [19]: ica = FastICA(n_components = 10)
```

```
In [20]: ica.fit(image)
```

```
Out[20]: FastICA(n_components=10)
```

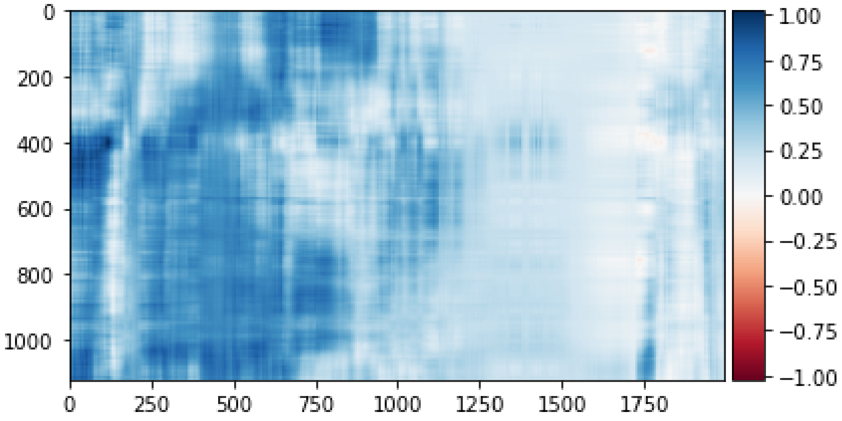
```
In [21]: image_ica = ica.fit_transform(image)
         restored = ica.inverse_transform(image_ica)
```

```
# show image to screen
io.imshow(restored)
show()
```



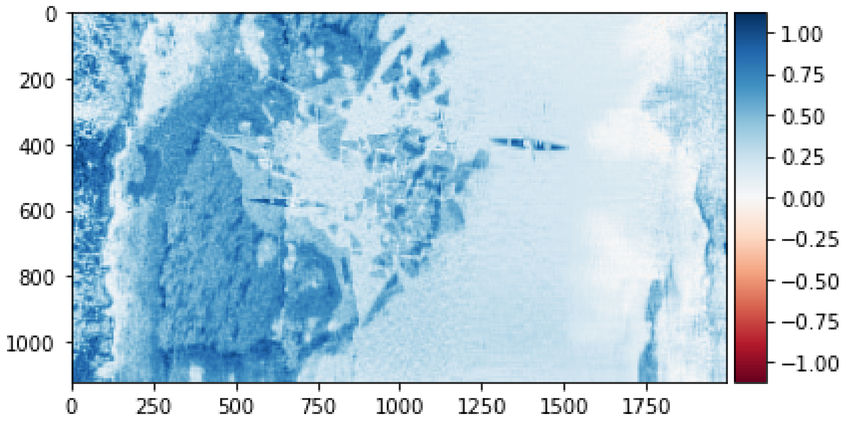
In [23]:

```
ica = FastICA(n_components = 5)
ica.fit(image)
image_ica = ica.fit_transform(image)
restored = ica.inverse_transform(image_ica)

# show image to screen
io.imshow(restored)
show()
```



In [24]:

```
ica = FastICA(n_components = 50)
ica.fit(image)
image_ica = ica.fit_transform(image)
restored = ica.inverse_transform(image_ica)

# show image to screen
io.imshow(restored)
show()
```



References:
http://theautomatic.net/2018/06/23/ica-on-images-with-python/
https://www.askpython.com/python/examples/principal-component-analysis-for-image-data
https://towardsdatascience.com/feature-extraction-techniques-d619b56e31be
https://www.pluralsight.com/guides/building-features-from-image-data-in-python
https://analyticsindiamag.com/image-feature-extraction-using-scikit-image-a-hands-on-guide/