In [2]:
```
#importing necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from skimage.io import imread, imshow
```

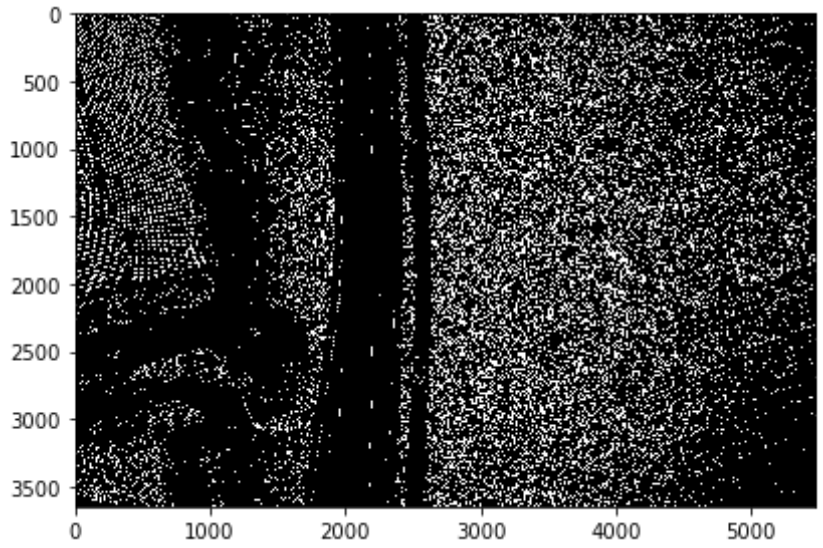
In [3]:
```
img_1 = imread('EP-00-00012_0119_0001.JPG')
imshow(img_1);

#Shape of image
print(img_1.shape)

#Pixel of image
piximg1 = np.reshape(img_1, (3648*5472*3))
print(piximg1)
```

```
(3648, 5472, 3)
[127 138 142 ...  56  51  47]
```

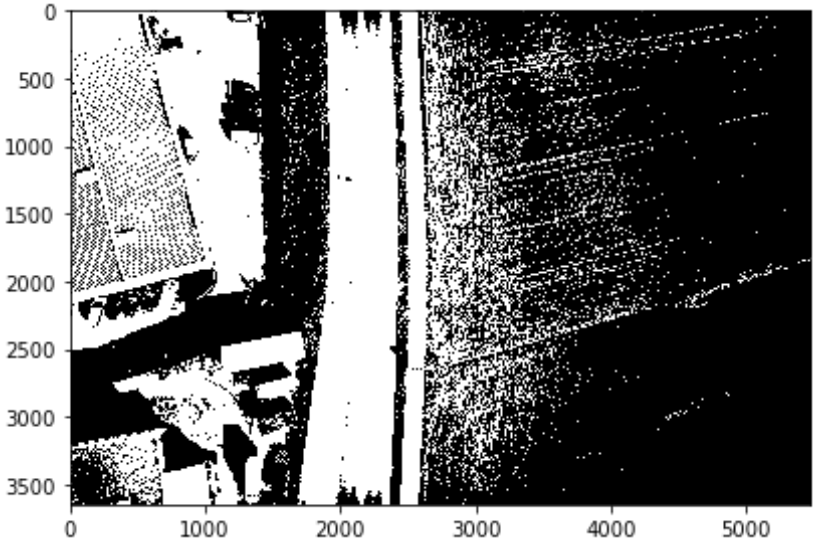
In [5]:
```
greyimg1 = imread('EP-00-00012_0119_0001.JPG', as_gray=True)
imshow(greyimg1);

#Shape of image
print(greyimg1.shape)

#Pixel of image
greypiximg_1 = np.reshape(greyimg1, (3648*5472))
print(greypiximg_1)
```

```
(3648, 5472)
[0.53314078 0.54490549 0.5684349  ... 0.19127098 0.19911412 0.20303569]
```

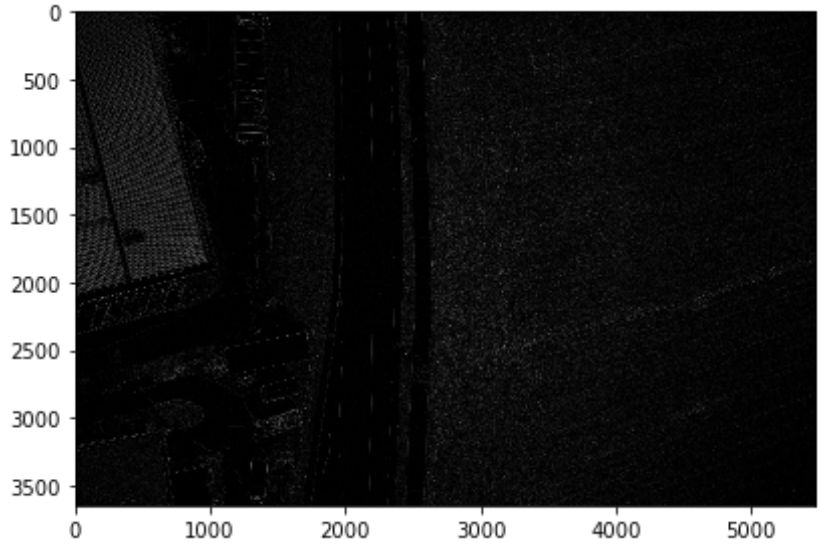
In [9]:
```
#Applyting Edge feature detection, canny algorithm as it helps to find
 the edges of the image in a wide range.

from skimage import feature
from skimage.feature import canny
can = feature.canny(greyimg1)
imshow(can, cmap='gray');
```



In [22]:
```
#Here we are going to find the threshold value for the image so that we
can understand the features of the image more clearly.
#Threshold is the image segmentation method.
#Selecting a wrong threshold values can distort the images.

from skimage.filters import threshold_otsu

thresh_val = threshold_otsu(greyimg1)

img_threhold = greyimg1 > thresh_val

# Show the original image
imshow(greyimg1)
plt.show()

# Show the thresholded image
imshow(img_threhold)
plt.show()
```

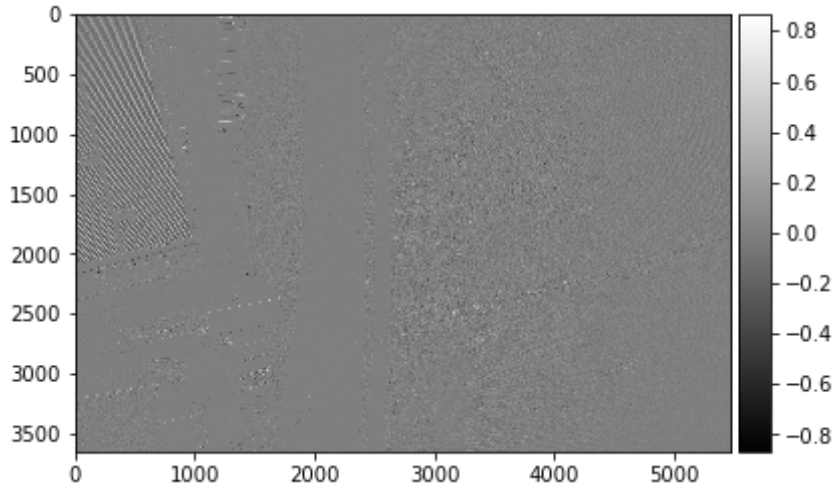


In [23]:
```
#Applyting Edge feature detection, Sobel Kernel. It helps to create ima
ges which have more underlining on edges.
from skimage import filters
sobel = filters.sobel(greyimg1)
imshow(sobel, cmap='gray');
```



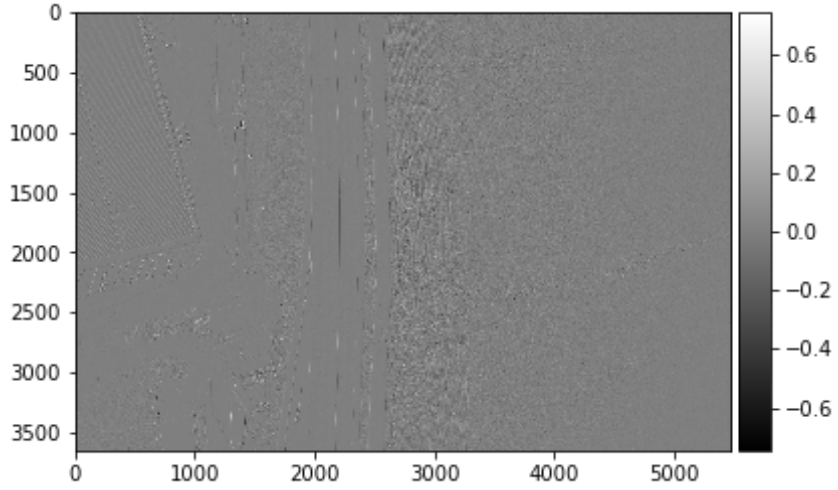
In [24]:
```
#Applyling Edge feature detection, prewitt kernel. The advantage of thi
s algorithm is that it detects edges both horizontally and vertically
from skimage.filters import prewitt_h,prewitt_v

#calculating horizontal edges using prewitt kernel
pkh = prewitt_h(greyimg1)

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

Out[24]:
```
<matplotlib.image.AxesImage at 0x1e20e4c3438>
```



In [52]:
```
#calculating vertical edges using prewitt kernel
pkv = prewitt_v(greyimg1)
imshow(pkv, cmap='gray')
```

Out[52]: <matplotlib.image.AxesImage at 0x1e200091b70>

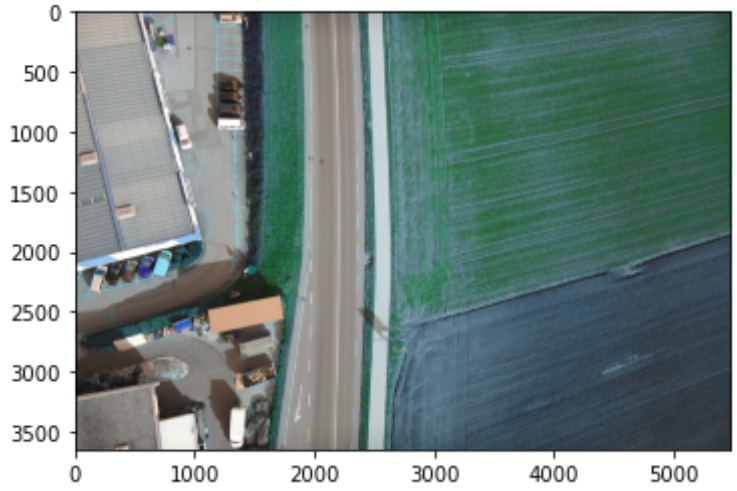


PCA Dimensionality Reduction method

In [37]:
```
# Importing necessary libraries
import cv2
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
```

In [38]:
```
# Reading the image
img_1 = cv2.imread('EP-00-00012_0119_0001.JPG')
plt.imshow(img_1)
```

Out[38]:
```
<matplotlib.image.AxesImage at 0x1e201a85978>
```



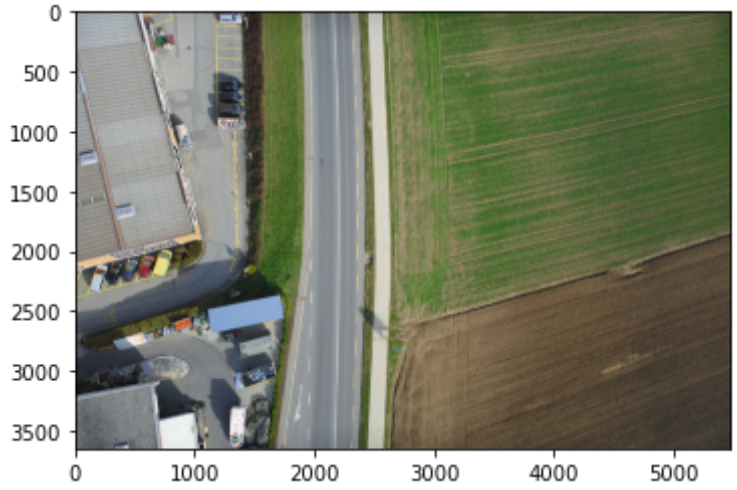
In [39]:
```
blue,green,red = cv2.split(img_1) # Splits the image in R,G,B arrays.
```

In [49]:
```
#Applying PCA to each array which we got after splitting the image.
#We are also applying inverse transform to transformed array. As we are
just keeping some components, inverse transform is important to recreat
e the original dimensions of the base image.

pca = PCA(200) # we are initializing PCA with first 30 principal compon
ents
```

```
red_transformed = pca.fit_transform(red)
red_inverted = pca.inverse_transform(red_transformed)


green_transformed = pca.fit_transform(green)
green_inverted = pca.inverse_transform(green_transformed)


blue_transformed = pca.fit_transform(blue)
blue_inverted = pca.inverse_transform(blue_transformed)
```

In [50]:
```
# compressing the image
img_compress= (np.dstack((red_inverted, green_inverted, blue_inverted
))).astype(np.uint8)
plt.imshow(img_compress)
```

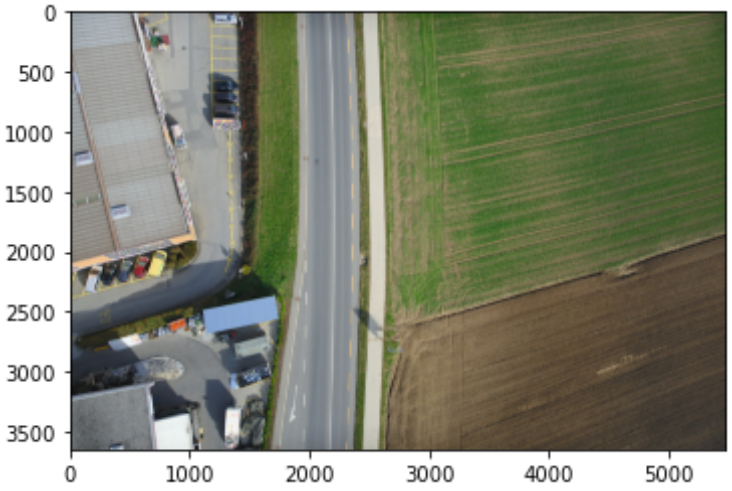
Out[50]: <matplotlib.image.AxesImage at 0x1e20e898f60>



In [51]:
```
pca = PCA(400) # we are increasing the number of components

red_transformed = pca.fit_transform(red)
red_inverted = pca.inverse_transform(red_transformed)
```

```
green_transformed = pca.fit_transform(green)
green_inverted = pca.inverse_transform(green_transformed)


blue_transformed = pca.fit_transform(blue)
blue_inverted = pca.inverse_transform(blue_transformed)

img_compress= (np.dstack((red_inverted, green_inverted, blue_inverted
))).astype(np.uint8)
plt.imshow(img_compress)
```

Out[51]: <matplotlib.image.AxesImage at 0x1e2009542e8>



In [53]:
```
import cv2
from sklearn.decomposition import FastICA
from pylab import *
import matplotlib.pyplot as plt
```

Refrences

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/