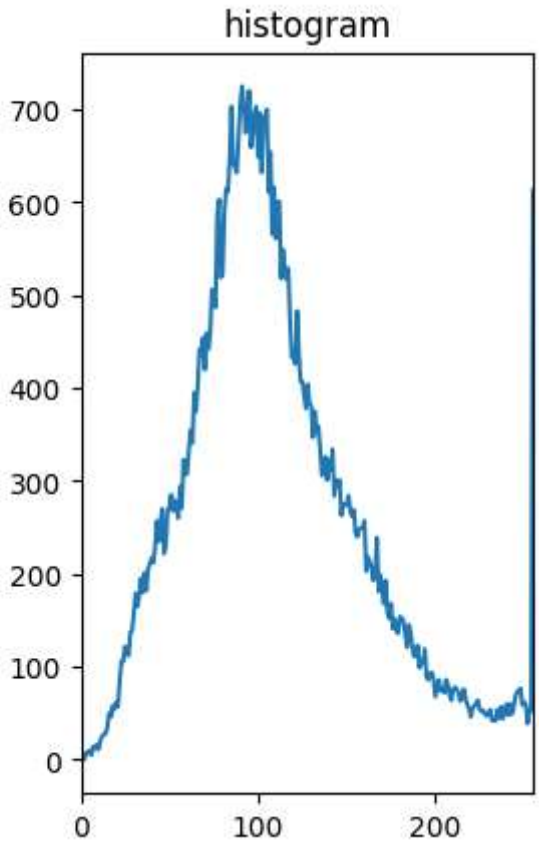
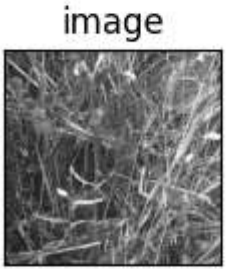
In [1]:
```
import cv2
import glob
import matplotlib.pyplot as plt
import numpy as np
import os
import pandas as pd
from sklearn.decomposition import PCA
```

In [2]:
```
# Siam weed

# 20171121-090047-1.jpg
# 20171121-090145-3.jpg
# 20171121-090211-3.jpg
# 20171121-090224-3.jpg
# 20171121-090235-3.jpg

# Negative

# 20180112-074718-2.jpg
# 20180112-074738-2.jpg
# 20180112-074748-2.jpg
# 20180112-074757-2.jpg
# 20180112-074759-1.jpg

path=r'C:\Users\vitta\OneDrive\Documents\Data_mining\Weed-4class-45'
os.chdir(path)

siam_list = [
'20171121-090047-1.jpg',
'20171121-090145-3.jpg',
'20171121-090211-3.jpg',
'20171121-090224-3.jpg',
'20171121-090235-3.jpg']

negatives=[
'20180112-074718-2.jpg',
'20180112-074738-2.jpg',
'20180112-074748-2.jpg',
'20180112-074757-2.jpg',
'20180112-074759-1.jpg']

siam_img=[cv2.imread(image,0) for image in siam_list]
negative_img=[cv2.imread(image,0) for image in negatives]
all_images=siam_img+negative_img
```

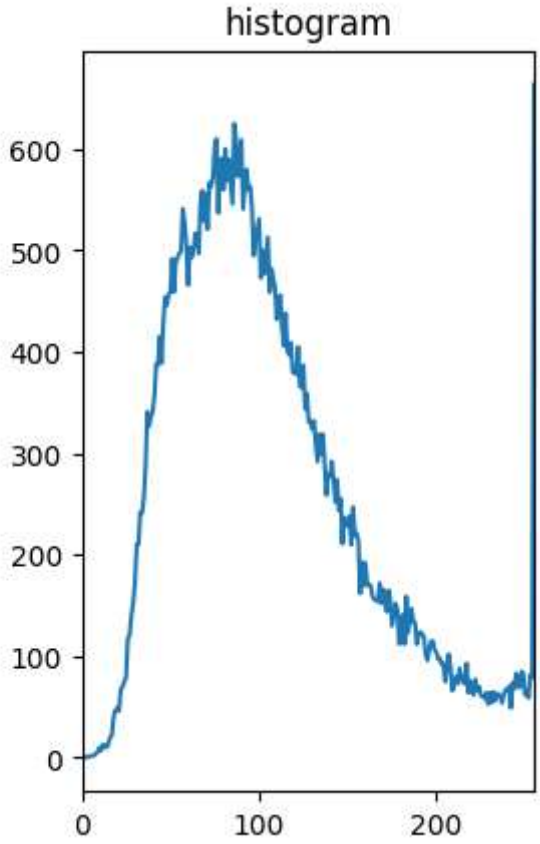
In [3]:
```
for j in range (0,10):
    plt.subplot(1,4,1)
    plt.imshow(all_images[j],cmap='gray')
    plt.title('image')
    plt.xticks([])
    plt.yticks([])
    plt.subplot(1,2,2)
    hist,bin = np.histogram(all_images[j].ravel(),256,[0,255])
    plt.xlim([0,255])
```

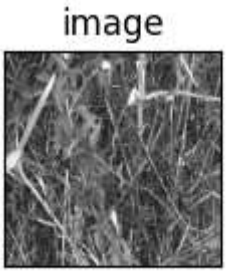
```
    plt.plot(hist)
    plt.title('histogram')
    plt.show()
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```



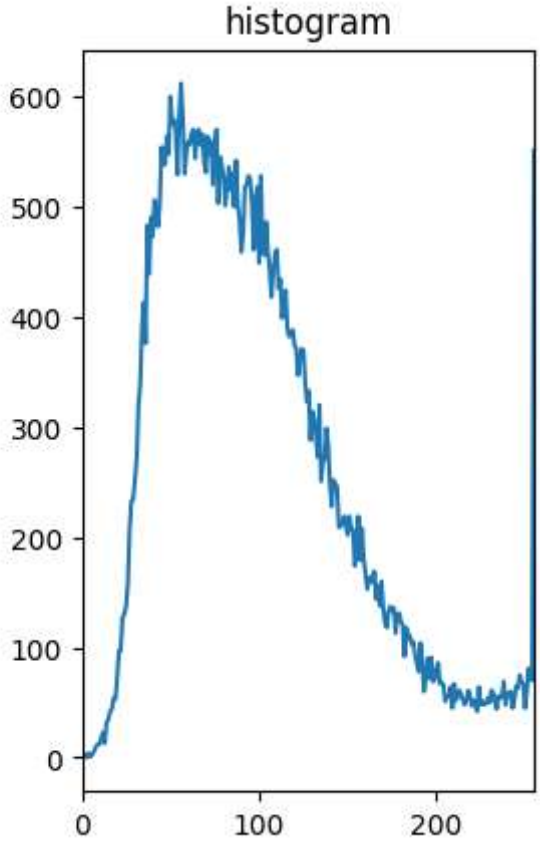


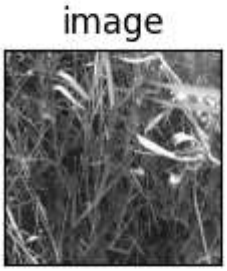
image


histogram
600
500
400
300
200
100
0
0
100
200

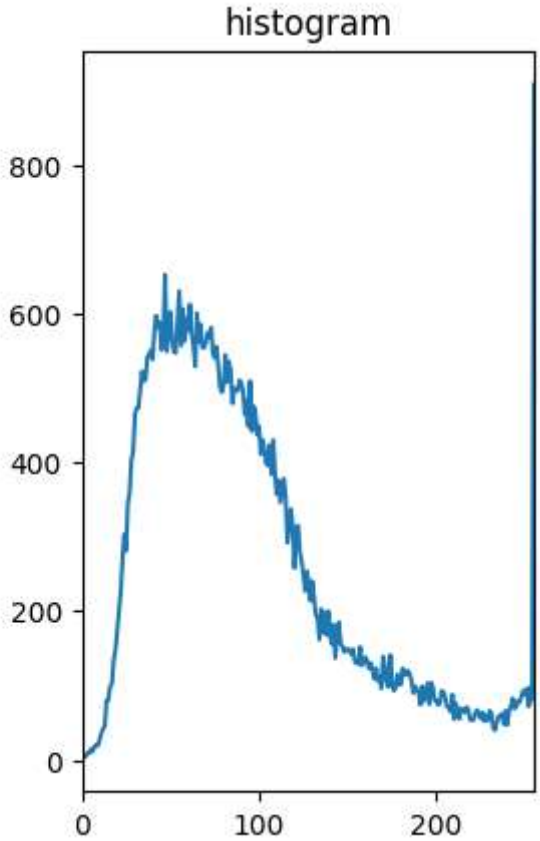


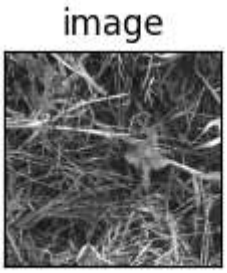
image


histogram
600
500
400
300
200
100
0
0
100
200

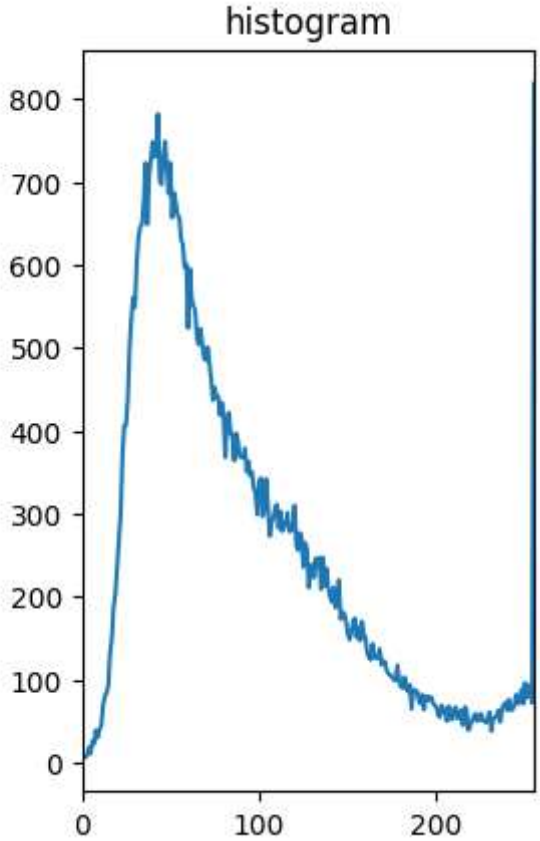


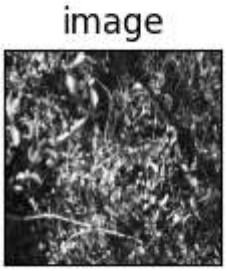
image


histogram
800
600
400
200
0
0
100
200

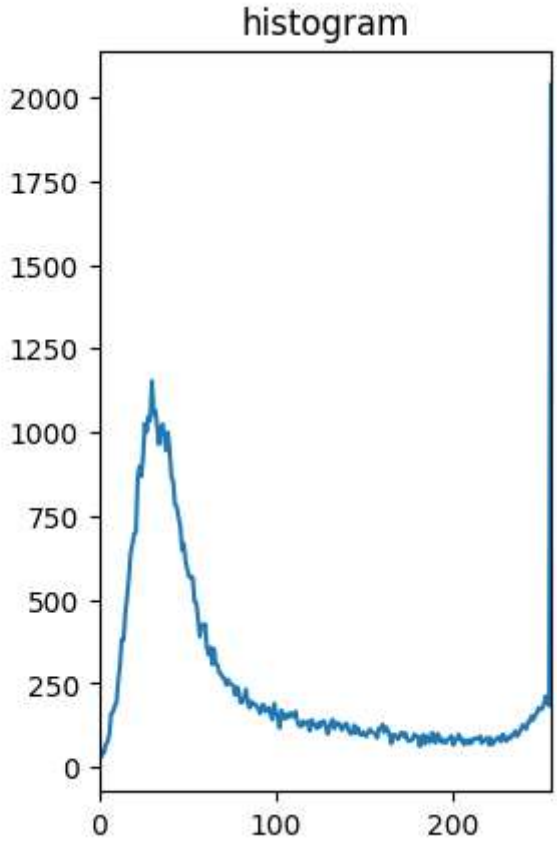


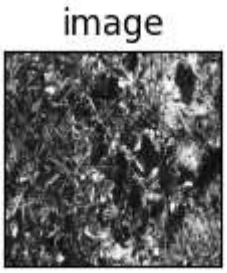
image


histogram
800
700
600
500
400
300
200
100
0
0
100
200

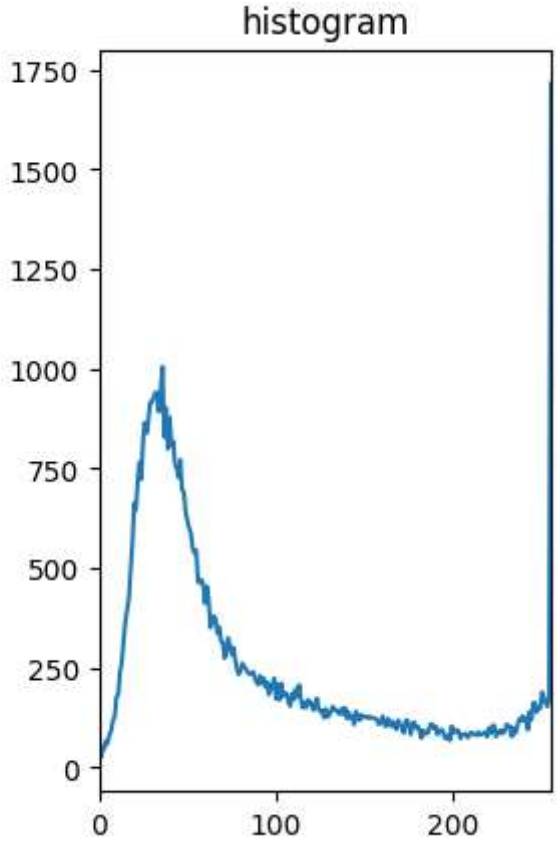


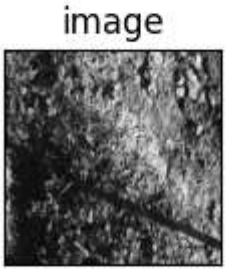
image


histogram
2000
1750
1500
1250
1000
750
500
250
0
0
100
200

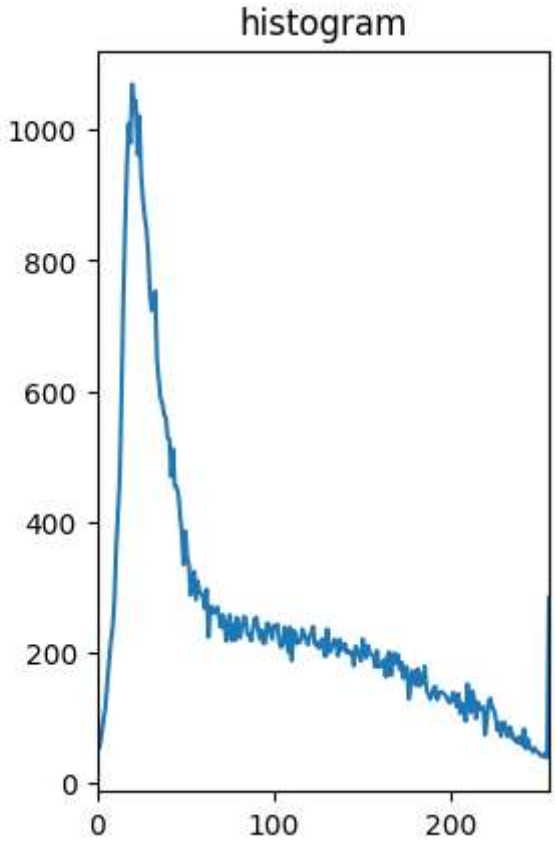


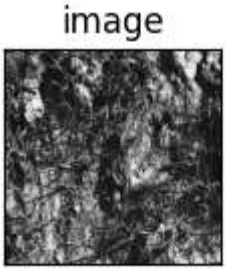
image


histogram
1750
1500
1250
1000
750
500
250
0
0
100
200

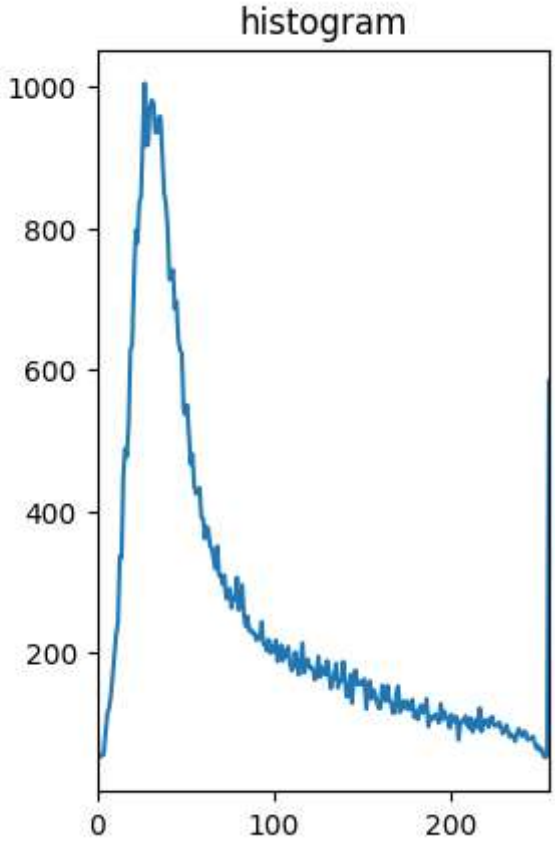


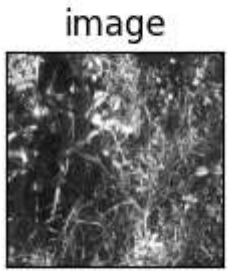
image

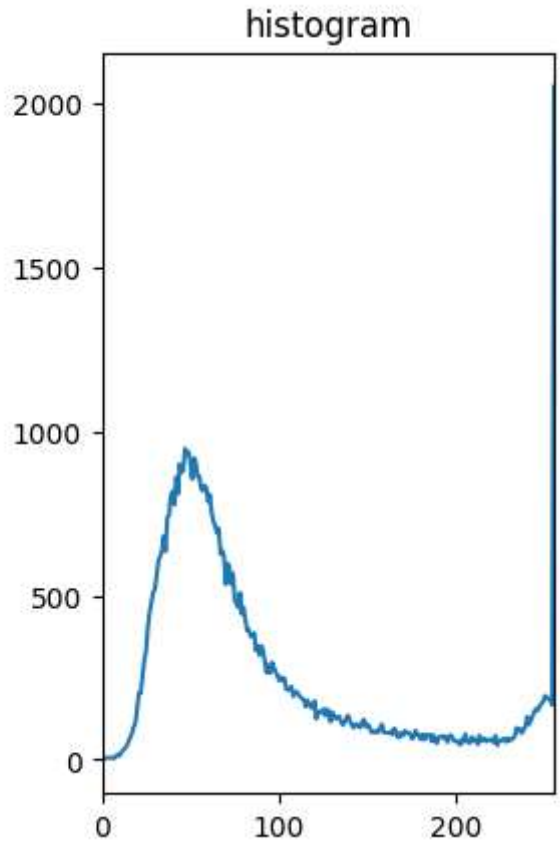

histogram
1000
800
600
400
200
0
0
100
200



image

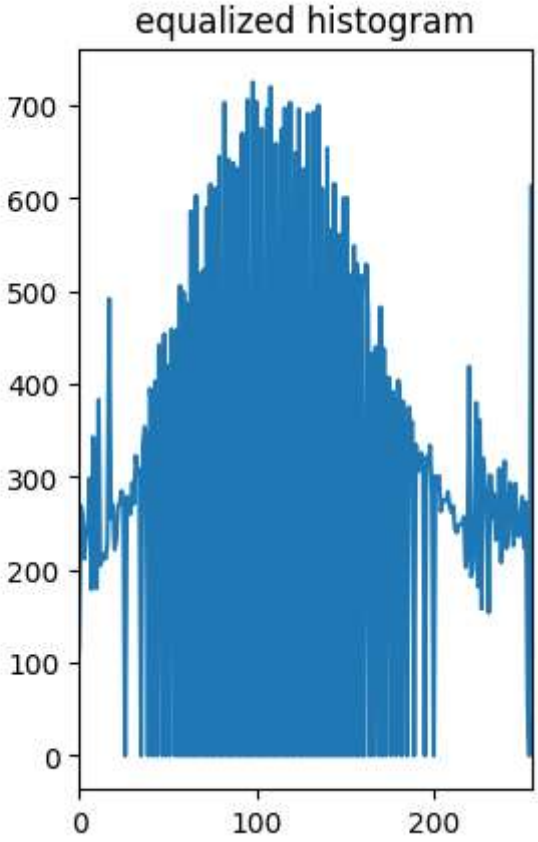

histogram
1000
800
600
400
200
0
100
200



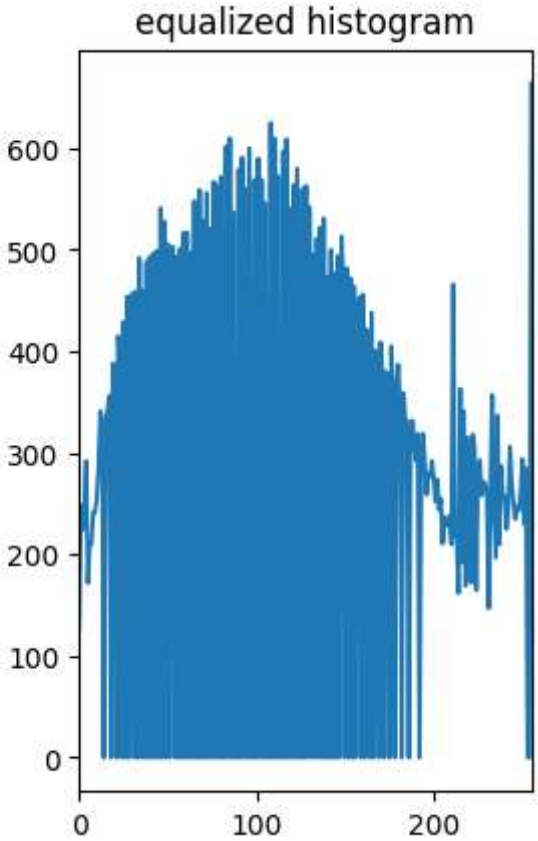


In [4]:
```
eq_images=[cv2.equalizeHist(img) for img in all_images]
for j in range (0,10):
    plt.subplot(1,4,1)
    plt.imshow(eq_images[j],cmap='gray')
    plt.title('equalized image')
    plt.xticks([])
    plt.yticks([])
    plt.subplot(1,2,2)
    hist,bin = np.histogram(eq_images[j].ravel(),256,[0,255])
    plt.xlim([0,255])
    plt.plot(hist)
    plt.title('equalized histogram')
    plt.show()
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

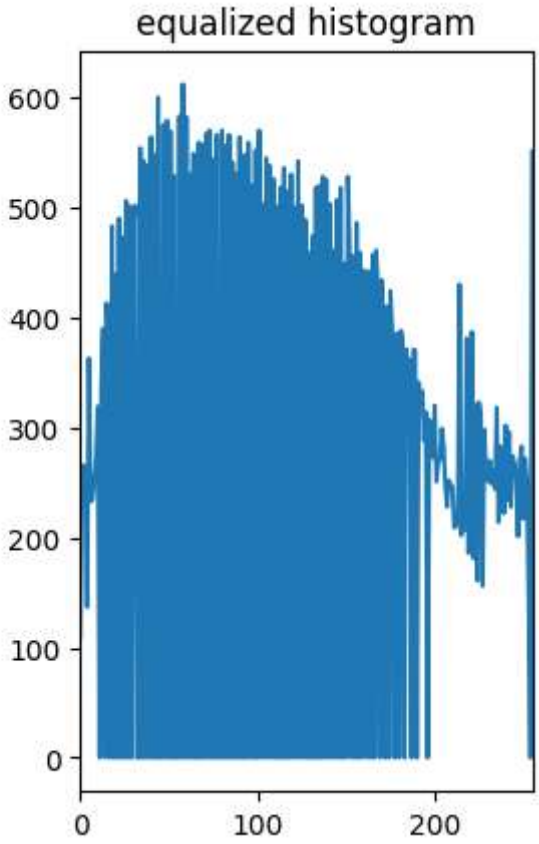
equalized image



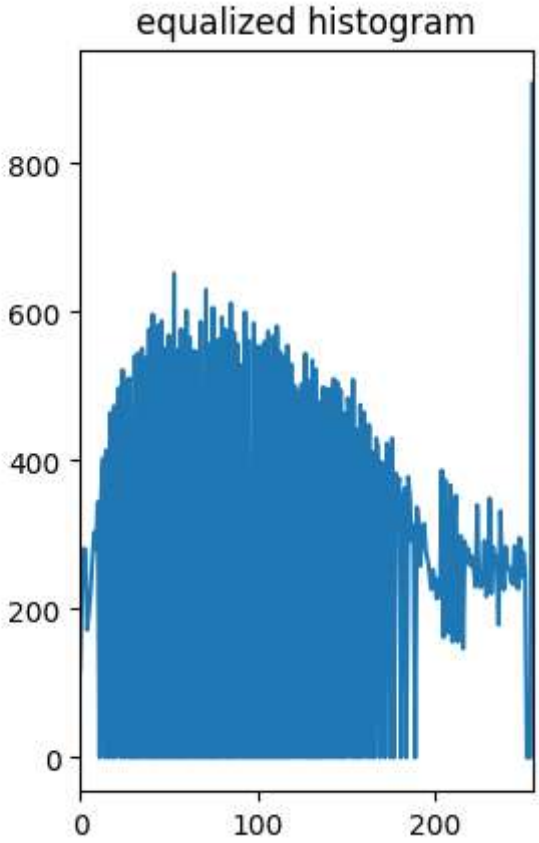
equalized image



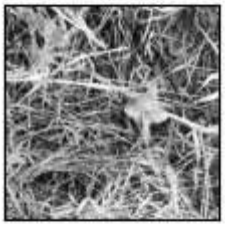
equalized image

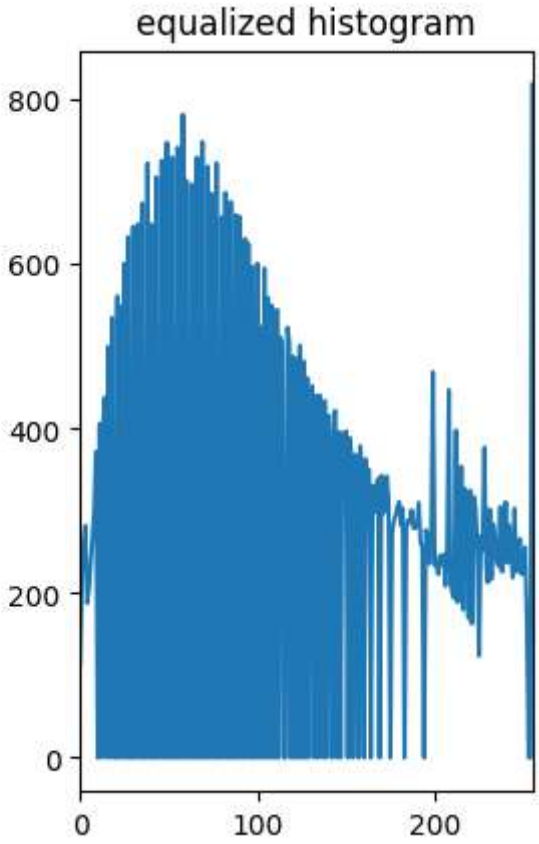


equalized image

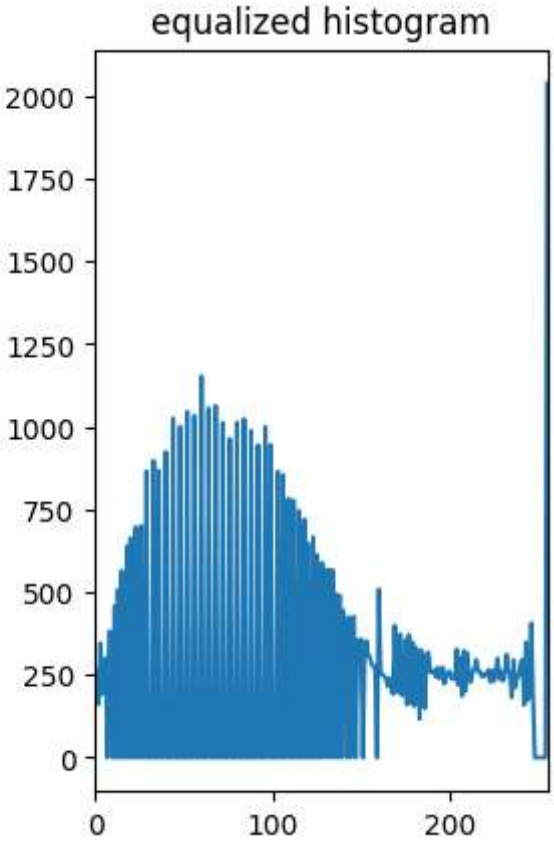


equalized image

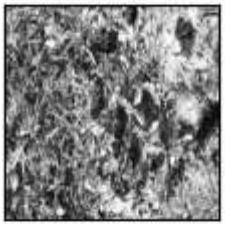


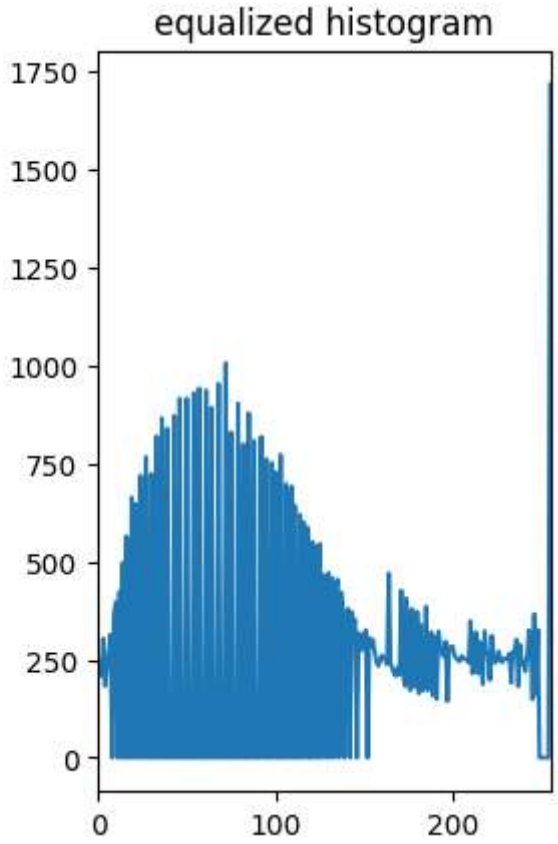


equalized image

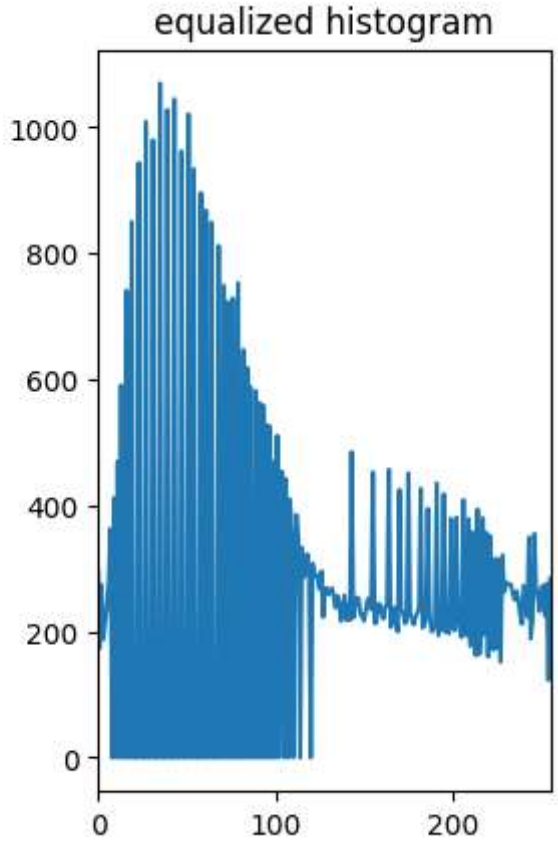


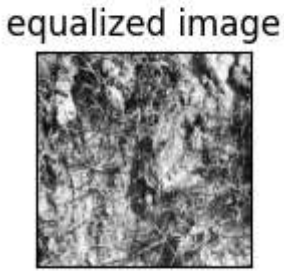
equalized image

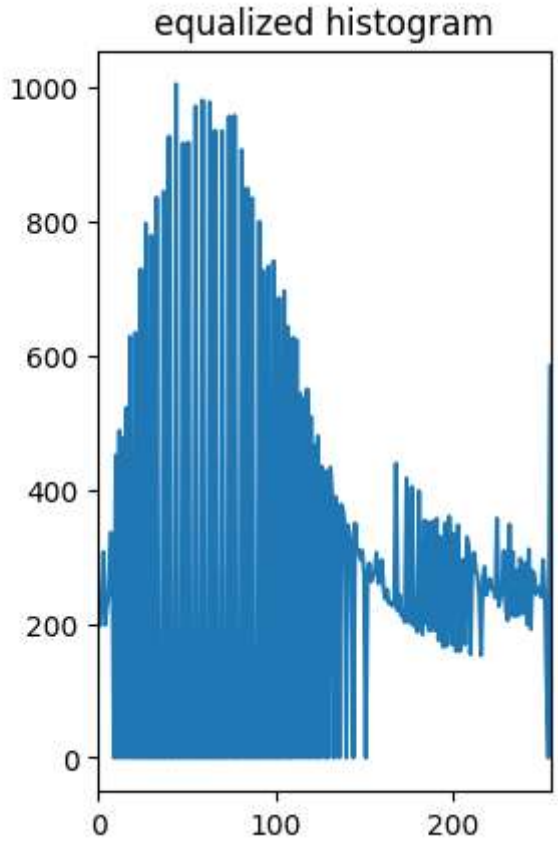


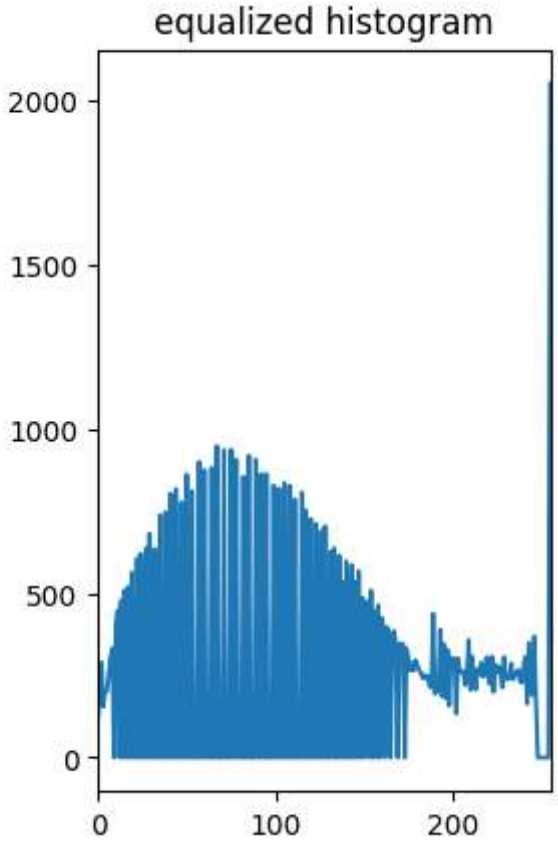


equalized image









In [5]:
```
plt.imshow(all_images[4],cmap='gray')
plt.title('Original image')
plt.xticks([])
```

```
plt.yticks([])
plt.show()
plt.imshow(eq_images[4],cmap='gray')
plt.title('Equalized image')
plt.xticks([])
plt.yticks([])
plt.show()
print('I observed more clarity in Equalized image compared to original image ')
```

I observed more clarity in Equalized image compared to original image

In [6]:
```
RGB_img=['20180112-073312-1.jpg','20170728-152434-1.jpg','20170906-100047-1.jpg','2
         '20170715-113141-1.jpg']
```

In [7]:
```
rgb_img=[cv2.imread(image) for image in RGB_img]
```

In [8]:
```
for j in range (0,5):
    plt.subplot(2,2,1)
    plt.title("Image")
    plt.imshow(cv2.cvtColor(rgb_img[j], cv2.COLOR_BGR2RGB))
    for i,col in zip(range(3),["blue","green","red"]):
        plt.subplot(2,2,2)
        h = cv2.calcHist(rgb_img[j], [i], None, [256], [0, 256])
        plt.hist(h, color=col)
#     plt.tight_layout()
    plt.title("RGB colors Histogram")
    plt.xlabel("Intensity")
    plt.ylabel("Pixel Count")
    plt.show()
```

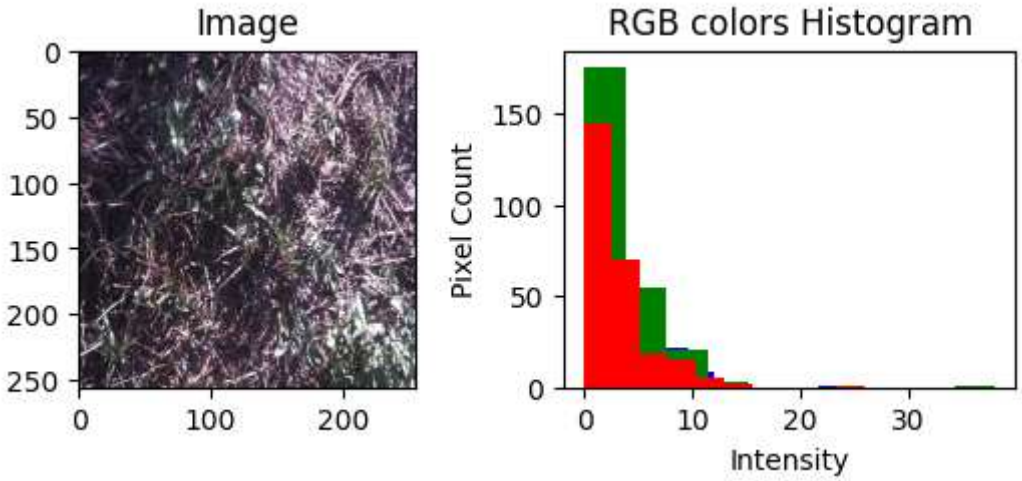


Image
RGB colors Histogram
Pixel Count
Intensity

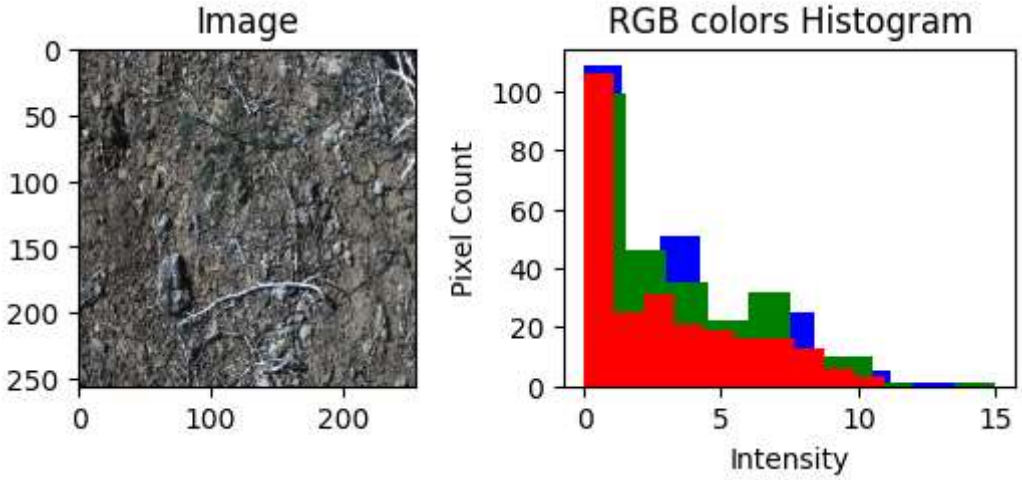


Image
RGB colors Histogram
Pixel Count
Intensity

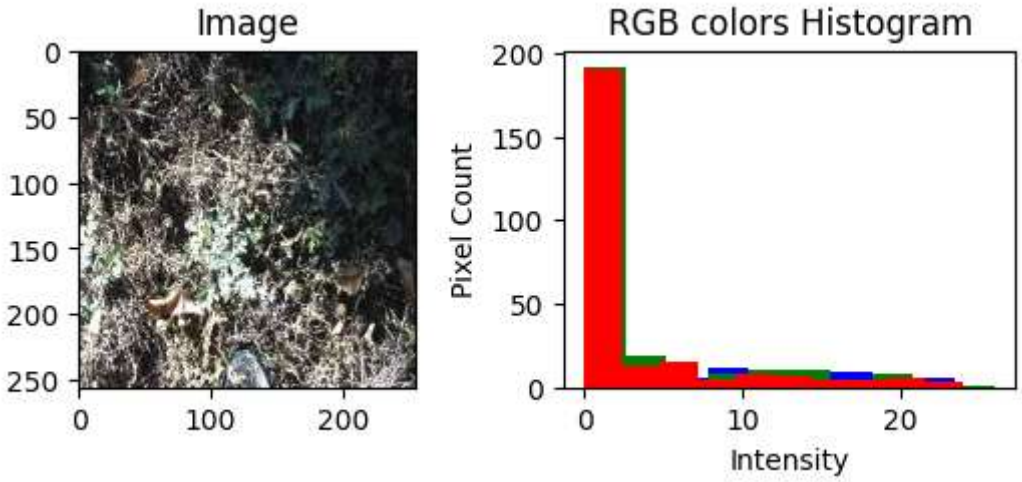
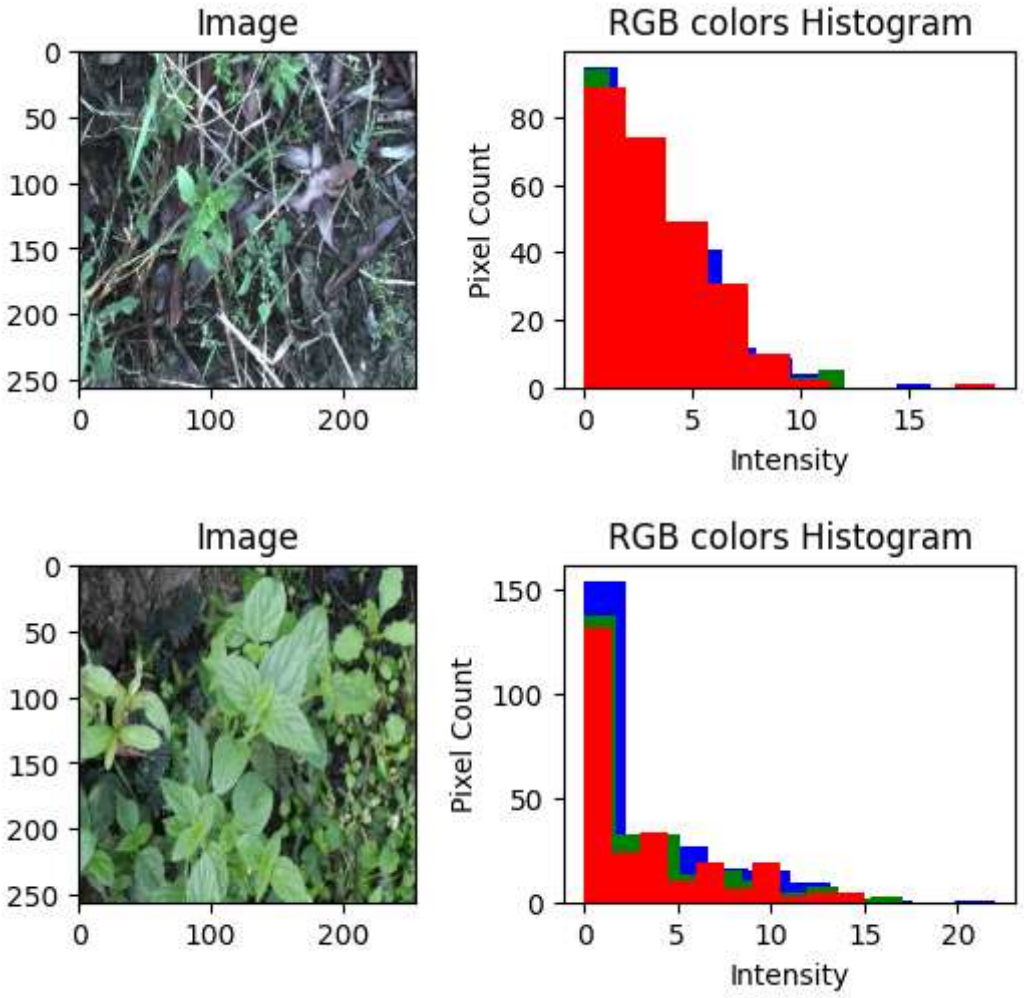
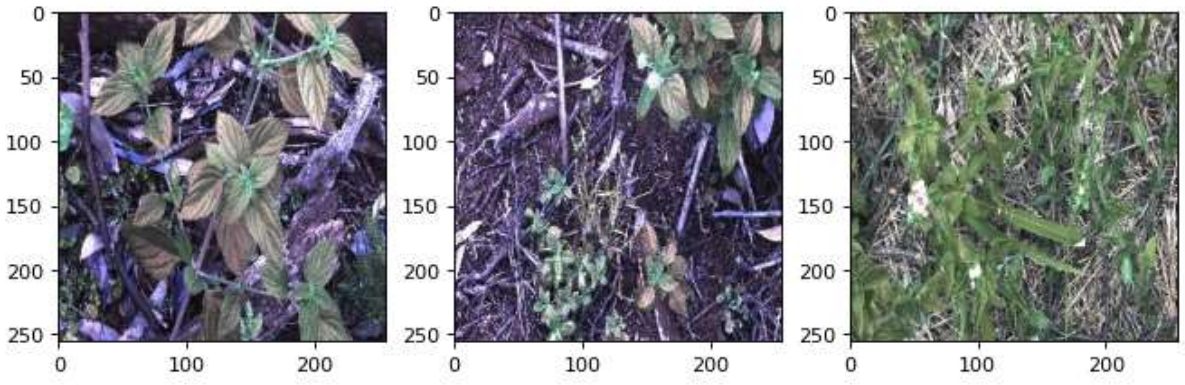


Image
RGB colors Histogram
Pixel Count
Intensity

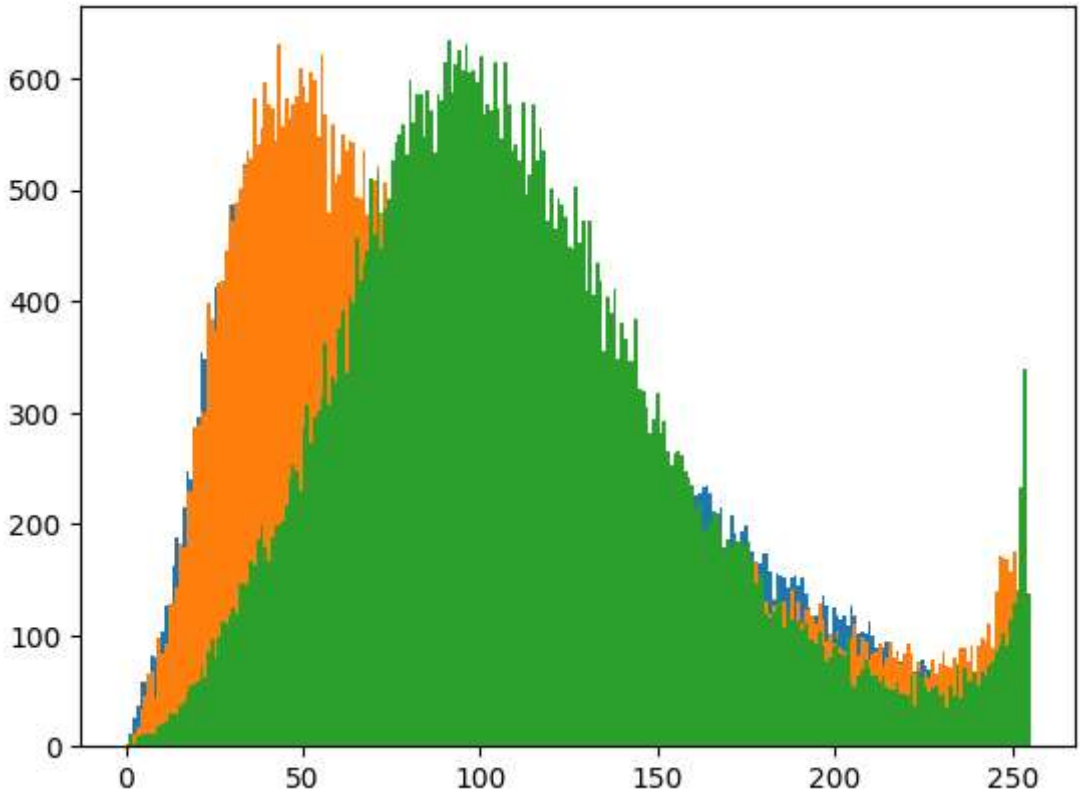


In [9]:
```
cv_img = []
fig = plt.figure(figsize=(10, 10),dpi=80)
rows = 3
columns = 3
count=1
# Lantana image
i1=cv2.imread("20170714-143530-2.jpg")
fig.add_subplot(rows, columns, count)
plt.imshow(i1)
count+=1
# Lantana image
i2=cv2.imread("20170714-143606-2.jpg")
fig.add_subplot(rows, columns, count)
plt.imshow(i2)
count+=1
# Siam weed
i3=cv2.imread("20171113-124439-1.jpg")
fig.add_subplot(rows, columns, count)
plt.imshow(i3)
count+=1
cv_img.append(i3)
```

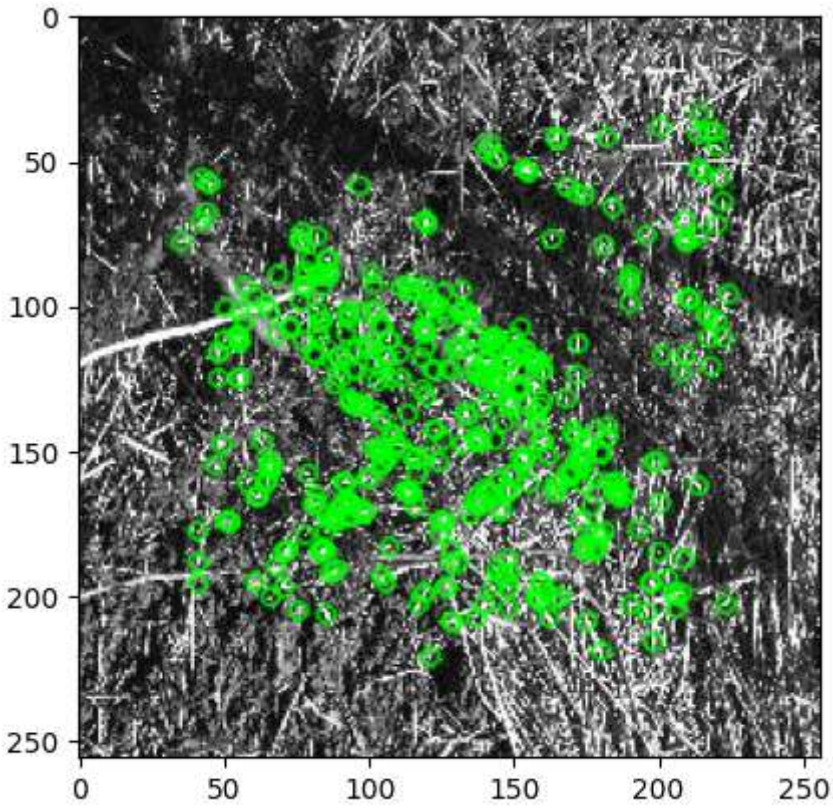


In [10]:
```
fig.add_subplot(rows, columns, count)
grayimage1 = cv2.cvtColor(i1, cv2.COLOR_BGR2GRAY)
plt.hist(grayimage1.ravel(),256,[0,255]);
count+=1
fig.add_subplot(rows, columns, count)
grayimage2 = cv2.cvtColor(i2, cv2.COLOR_BGR2GRAY)
plt.hist(grayimage2.ravel(),256,[0,255]);
count+=1
fig.add_subplot(rows, columns, count)
grayimage3 = cv2.cvtColor(i3, cv2.COLOR_BGR2GRAY)
plt.hist(grayimage3.ravel(),256,[0,255]);
count+=1
h1 = cv2.calcHist([grayimage1],[0],None,[256],[0,256])
h2 = cv2.calcHist([grayimage2],[0],None,[256],[0,256])
h3 = cv2.calcHist([grayimage3],[0],None,[256],[0,256])

#euclidean Distance calculate
dist=cv2.norm(h1,h2,normType=cv2.NORM_L2)
print("euclidean-Same Class:",dist)
dist2=cv2.norm(h2,h3,normType=cv2.NORM_L2)
print("euclidean-Different Class:",dist2)

#Manhattan Distance calculate
dist3=cv2.norm(h1,h2,normType=cv2.NORM_L1)
print("Manhattan-Same Class:",dist3)
dist4=cv2.norm(h2,h3,normType=cv2.NORM_L1)
print("Manhattan-Different Class:",dist4)

#Bhattacharyya Distance calculate
dist5=cv2.compareHist(h1,h2,cv2.HISTCMP_BHATTACHARYYA)
print("Bhattacharyya-Same Class:",dist5)
dist6=cv2.compareHist(h2,h3,cv2.HISTCMP_BHATTACHARYYA)
print("Bhattacharyya-Different Class:",dist6)

#Histogram Intersection Distance calculate
dist7=cv2.compareHist(h1,h2,cv2.HISTCMP_INTERSECT)
print("Histogram Intersection-Same Class:",dist7)
dist8=cv2.compareHist(h2,h3,cv2.HISTCMP_INTERSECT)
print("Histogram Intersection-Different Class:",dist8)
```

```
euclidean-Same Class: 992.8474203018307
euclidean-Different Class: 2872.236410882642
Manhattan-Same Class: 12344.0
Manhattan-Different Class: 34616.0
Bhattacharyya-Same Class: 0.08290651164591246
Bhattacharyya-Different Class: 0.22552588105043844
Histogram Intersection-Same Class: 59364.0
Histogram Intersection-Different Class: 48228.0
```

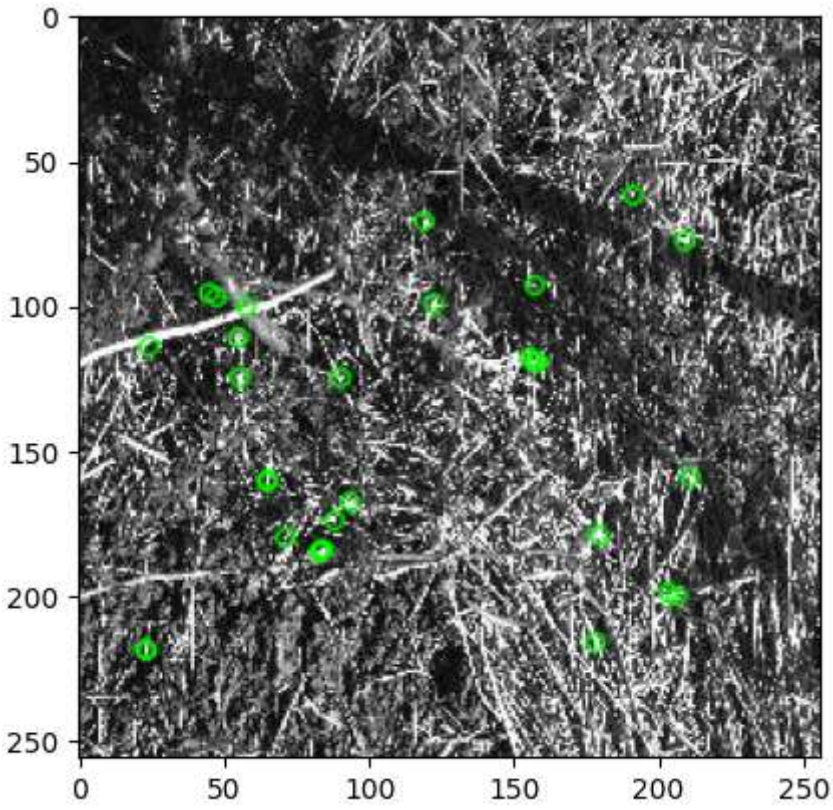


In [11]:
```python
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt
img = cv.imread('20170729-092850-2.jpg', cv.IMREAD_GRAYSCALE)
# Initiate ORB detector
orb = cv.ORB_create()
# find the keypoints with ORB
kp = orb.detect(img,None)
# compute the descriptors with ORB
kp, des = orb.compute(img, kp)
# draw only keypoints location,not size and orientation
img2 = cv.drawKeypoints(img, kp, None, color=(0,255,0), flags=0)
plt.imshow(img2), plt.show()
```



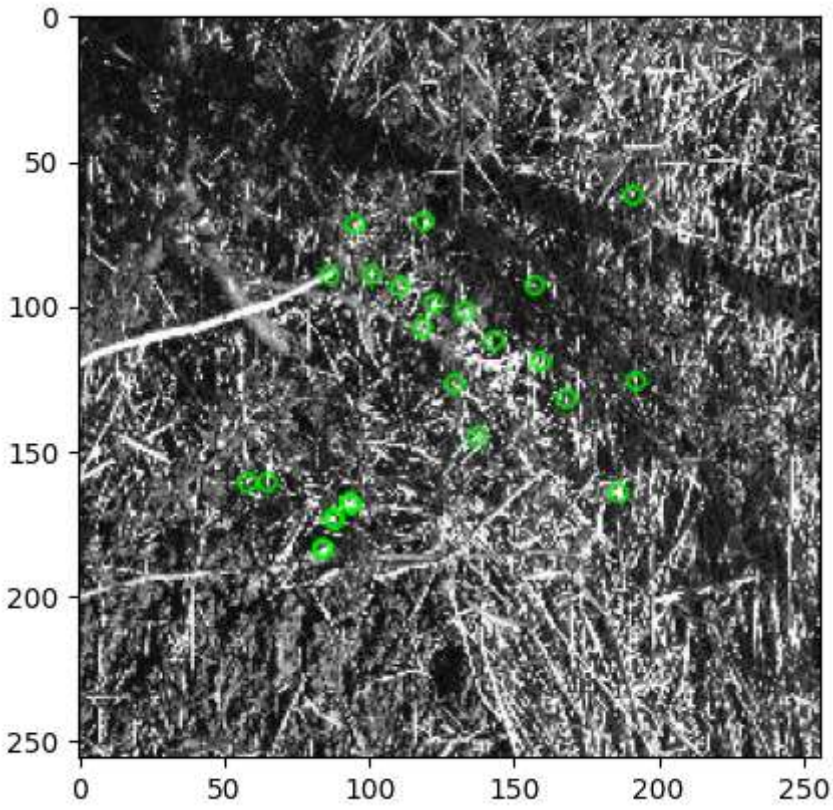
Out[11]: (<matplotlib.image.AxesImage at 0x1f405292f50>, None)

In [12]:
```
orb = cv.ORB_create(edgeThreshold=int(21/2),
patchSize=10, nlevels=8, fastThreshold=20,
scaleFactor=1.2, WTA_K=2,scoreType=cv.ORB_HARRIS_SCORE,
firstLevel=0, nfeatures=30)
# find the keypoints with ORB
kp = orb.detect(img,None)
# compute the descriptors with ORB
kp, des = orb.compute(img, kp)
# draw only keypoints location,not size and orientation
img2 = cv.drawKeypoints(img, kp, None, color=(0,255,0), flags=0)
plt.imshow(img2), plt.show()
print(len(kp))
```



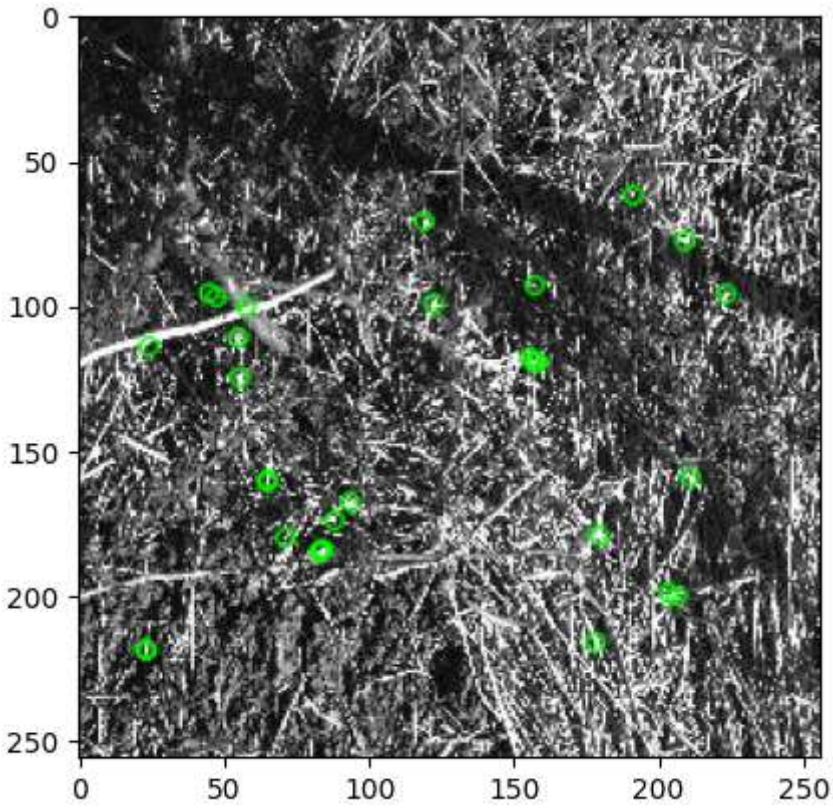

```
30
```

In [13]:
```
orb = cv.ORB_create(edgeThreshold=int(50),
patchSize=int(10), nlevels=8, fastThreshold=20,
scaleFactor=1.2, WTA_K=2,scoreType=cv.ORB_HARRIS_SCORE,
firstLevel=0, nfeatures=30)
# find the keypoints with ORB
kp = orb.detect(img,None)
# compute the descriptors with ORB
kp, des = orb.compute(img, kp)
# draw only keypoints location,not size and orientation
img2 = cv.drawKeypoints(img, kp, None, color=(0,255,0), flags=0)
plt.imshow(img2), plt.show()
print(len(kp))
```



25

In [14]:
```
orb = cv.ORB_create(edgeThreshold=int(5),
patchSize=int(10), nlevels=8, fastThreshold=20,
scaleFactor=1.2, WTA_K=2,scoreType=cv.ORB_HARRIS_SCORE,
firstLevel=0, nfeatures=30)
# find the keypoints with ORB
kp = orb.detect(img,None)
# compute the descriptors with ORB
kp, des = orb.compute(img, kp)
# draw only keypoints location,not size and orientation
img2 = cv.drawKeypoints(img, kp, None, color=(0,255,0), flags=0)
plt.imshow(img2), plt.show()
print(len(kp))
print("Banner id:916439321, The endpoints extracted for the 3 edgethreshold values
```




```
30
Banner id:916439321, The endpoints extracted for the 3 edgethreshold values are 3
0,25,30
```

In [15]:
```
spreadsheet = pd.read_csv(r"C:\Users\vitta\OneDrive\Documents\Data_mining\Weed-4cla
spreadsheet = spreadsheet[(spreadsheet['Species']=="Negative")|(spreadsheet['Specie
spreadsheet
```

Out[15]:

	Filename	Label	Species
0	20161207-112417-0.jpg	8	Negative
1	20161207-112431-0.jpg	8	Negative
2	20161207-112802-0.jpg	8	Negative
3	20161207-112812-0.jpg	8	Negative
4	20170128-101909-0.jpg	8	Negative
...	...	...	...
13323	20171025-172145-3.jpg	3	Parthenium
13324	20171025-172200-3.jpg	3	Parthenium
13325	20171025-172226-3.jpg	3	Parthenium
13326	20171025-172236-3.jpg	3	Parthenium
13327	20171025-172247-3.jpg	3	Parthenium

10128 rows × 3 columns

In [16]:
```
img=[cv2.imread(image,0) for image in spreadsheet['Filename']]
histogram=[]
for i in img:
    hist,bin = np.histogram(i.ravel(),256,[0,255])
    histogram.append(hist)
len(histogram)
```

Out[16]: 10128

In [17]:
```
pca = PCA(n_components=2)
pca.fit(histogram)
p = pca.transform(histogram)
```

In [18]:
```
plt.scatter(p[:9106,0],p[:9106,1],color='red')
plt.scatter(p[-1022:,0],p[-1022:,1],color='blue')
```

Out[18]: <matplotlib.collections.PathCollection at 0x1f402c982e0>