Import Libraries

In [29]:
```
import numpy as np
import cv2
import matplotlib.pyplot as plt
import os
import pyclustering
import pandas as pd
import random
```

Dataset Paths

In [30]:
```
folder_path = "C:\\Users\\Rajat\\Desktop\\SEM_3\\CV\\Project\\Semantic dataset50"
test_path = "C:\\Users\\Rajat\\Desktop\\SEM_3\\CV\\Project\\Semantic dataset50\\image"
ground_truth_path = "C:\\Users\\Rajat\\Desktop\\SEM_3\\CV\\Project\\Semantic dataset50\\ground-truth"
```

Function to load images from folder sorted by name

In [31]:
```
def load_images_from_folder(folder):
    images = []
    ls = os.listdir(folder)
    ls.sort()
    ID = []
    #print(ls)
    for filename in ls:
        img = cv2.imread(os.path.join(folder,filename))
        if img is not None:
            images.append(img)
            ID.append(filename)
    return images, ID
```

Function to convert grayscale image to binary

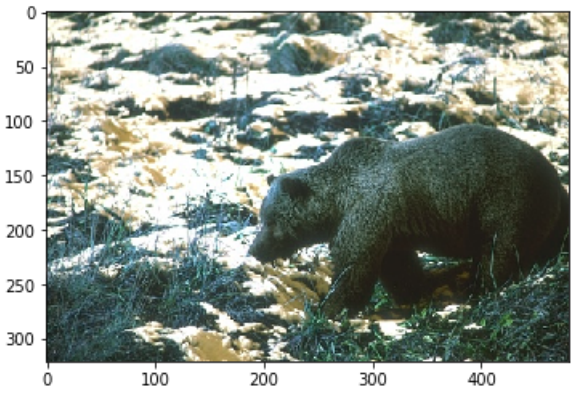
In [32]:
```
# Binary Conversion so to get precison/recall # 256/2 = 128 # 0 For <=127 , 1 else
# 0-> Black 255-> White
def convert_gray_2_binary_data(img):
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            if(img[i,j] <= 127):
                img[i,j] = 0
            else:
                img[i,j] = 1
    plt.imshow(img, cmap = 'gray')
    return img
def convert_gray_2_binary_truth(img):
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            if(img[i,j] <= 127):
                img[i,j] = 0
            else:
                img[i,j] = 1
    plt.imshow(img, cmap = 'gray')
    return img
```

In [33]:
```
images, ID_data = load_images_from_folder(test_path)
images_ground_truth, ID_truth = load_images_from_folder(ground_truth_path)
```

Original Image

In [34]:
```
plt.imshow( images[0])
```

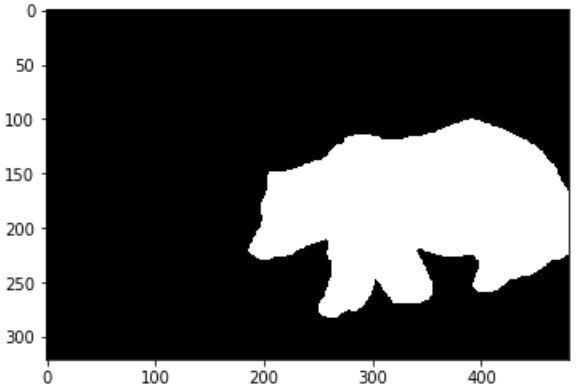
Out[34]: <matplotlib.image.AxesImage at 0x77230e62b0>



Ground Truth Image

In [35]:
```
plt.imshow(images_ground_truth[0])
```

Out[35]: <matplotlib.image.AxesImage at 0x771a287b38>



Function to calculate Precision, Recall, F1 Score, Accuracy and IoU Scores

In [45]:
```
# Precision, Recall, F1 Score, IUC
# TP = 11 (ground_truth, result_data)
# TN = 00 (actual, predicted)
# FP = 01
# FN = 10
def calc_precision_recall(results, ground_truths):
    precisions = []
    recalls = []
    F1_scores = []
    IOUs = []
    accuracies =[]

    for k in range(len(results)):
        print('for image',k, '\n')
        TP = 0
        FP = 0
        FN = 0
        TN = 0
        for i in range(results[k].shape[0]):
            for j in range(results[k].shape[1]):
                if results[k][i,j] == 0 and ground_truths[k][i,j] == 0:
                    TN = TN + 1
                elif results[k][i,j] == 0 and ground_truths[k][i,j] == 1:
                    FP = FP + 1
                elif results[k][i,j] == 1 and ground_truths[k][i,j] == 0:
                    FN = FN + 1
                else :
                    TP = TP + 1
        precision = TP / (TP + FP)
        recall = TP / (TP + FN)
        iou = TP /(TP + FN + FP )
        accuracy = (TP + TN) / (TP + TN + FP + FN)
        if (precision + recall) != 0:
            f1_score = (2 * precision * recall)/ (precision + recall)
        else:
            f1_score = 0
        precisions.append(precision)
        recalls.append(recall)
        accuracies.append(accuracy*100)
        IOUs.append(iou)
        F1_scores.append(f1_score)
    return precisions, recalls, accuracies, IOUs, F1_scores
```

In [38]:
```
def convert(img):
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            if img[i,j] == 0:
                img[i,j] = 1
            else:
                img[i,j] = 0
    return img
```

In [50]:
```
ls = [5]
for i in ls:
    results[i] = convert(results[i])
```

KMedoids from scratch for image segmentation

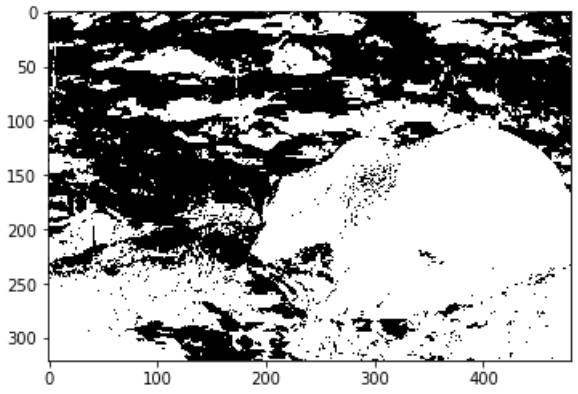
In [22]:

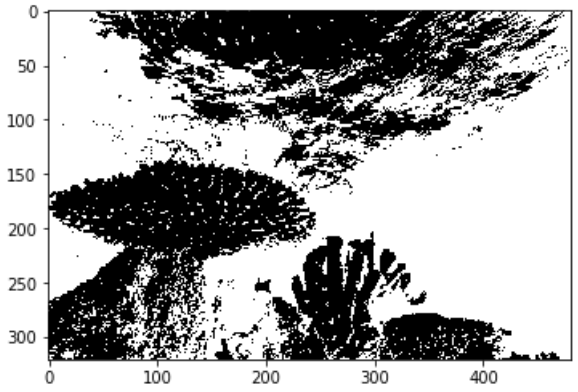
```
# Kmedoid algo
def distance(a,b):
    return np.abs(a-b)
def closest(lst, K):
    return lst[min(range(len(lst)), key = lambda i: abs(lst[i]-K))]

def mean_calc(img, labels, medoids):
    mean_1 = 0
    mean_2 = 0
    for i in range(len(img)):
        if labels[i] == medoids[0]:
            mean_1 = mean_1 + img[i]
        else:
            mean_1 = mean_1 + img[i]
    mean_1 = mean_1/len(img)
    mean_2 = mean_2/len(img)

    ## nearest mean
    mean_1 = closest(img, mean_1)
    mean_2 = closest(img, mean_2)

    return mean_1, mean_2


results = []
for p in range(50):
    print('Working on image :', p)
    img = images[p]
    img = cv2.cvtColor(img,cv2.COLOR_RGB2GRAY)
    img = img.reshape((-1,))
    print(img.shape)
    K = 2
    medoid = []
    m1 = random.choice(img)
    m2 = random.choice(img)
    medoids =   [m1,  m2]
    #plt.imshow(img, cmap = 'gray')
    iters = 10
    for i in range(iters):
        labels = np.zeros(len(img))
        # Label assignment
        for j in range(len(img)):
                if distance(img[j], medoids[0]) < distance(img[j], medoids[1]):
                    labels[j] = medoids[0]
                else:
                    labels[j] = medoids[1]
         # Mean calculation

        mean_1, mean_2 = mean_calc(img, labels, medoids)
        medoids[0] = mean_1
        medoids[1] = mean_2

    print(medoids)
    for i in range(len(img)):
        if labels[i] == medoids[0]:
            img[i] = 0
        else:
            img[i] = 1
    img = img.reshape(images[p].shape[:2])
    plt.figure()
    plt.imshow(img, cmap = 'gray')
    results.append(img)
```

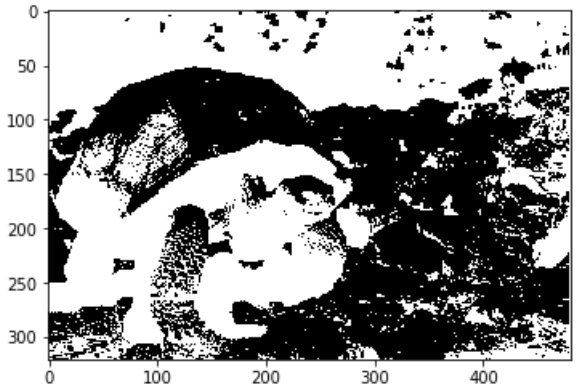
```
Working on image : 0
(154401,)

C:\Users\Rajat\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: RuntimeWarning: overflow encou
ntered in ubyte_scalars
  This is separate from the ipykernel package so we can avoid doing imports until

[131, 14]
Working on image : 1
(154401,)
[89, 1]
Working on image : 2
(154401,)
[125, 26]
Working on image : 3
(154401,)
[77, 23]
Working on image : 4
(154401,)
[106, 22]
Working on image : 5
(154401,)
[129, 31]
Working on image : 6
(154401,)
[123, 27]
Working on image : 7
(154401,)
[174, 34]
Working on image : 8
(154401,)
[70, 22]
Working on image : 9
(154401,)
[107, 20]
Working on image : 10
(154401,)
[134, 22]
Working on image : 11
(154401,)
[98, 10]
Working on image : 12
(154401,)
[113, 5]
Working on image : 13
(154401,)
[94, 0]
Working on image : 14
(154401,)
[100, 1]
Working on image : 15
(154401,)
[62, 2]
Working on image : 16
(154401,)
[153, 2]
Working on image : 17
(154401,)
[121, 1]
Working on image : 18
(154401,)
[120, 17]
Working on image : 19
(154401,)
[63, 13]
Working on image : 20
(154401,)
[99, 19]
Working on image : 21
(154401,)

C:\Users\Rajat\Anaconda3\lib\site-packages\ipykernel_launcher.py:60: RuntimeWarning: More than 20
figures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure
`) are retained until explicitly closed and may consume too much memory. (To control this warning,
see the rcParam `figure.max_open_warning`).
```

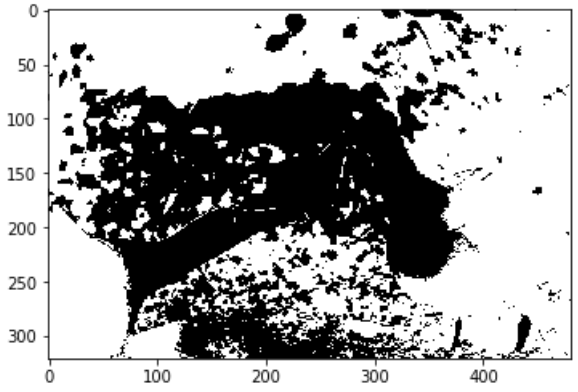
```
[114, 8]
Working on image : 22
(154401,)
[87, 1]
Working on image : 23
(154401,)
[120, 13]
Working on image : 24
(154401,)
[90, 4]
Working on image : 25
(154401,)
[128, 10]
Working on image : 26
(154401,)
[75, 1]
Working on image : 27
(154401,)
[95, 5]
Working on image : 28
(154401,)
[37, 6]
Working on image : 29
(154401,)
[169, 8]
Working on image : 30
(154401,)
[32, 3]
Working on image : 31
(154401,)
[60, 8]
Working on image : 32
(154401,)
[52, 2]
Working on image : 33
(154401,)
[60, 4]
Working on image : 34
(154401,)
[125, 3]
Working on image : 35
(154401,)
[72, 0]
Working on image : 36
(154401,)
[85, 0]
Working on image : 37
(154401,)
[112, 1]
Working on image : 38
(154401,)
[185, 11]
Working on image : 39
(154401,)
[125, 29]
Working on image : 40
(154401,)
[96, 0]
Working on image : 41
(154401,)
[70, 0]
Working on image : 42
(154401,)
[75, 0]
Working on image : 43
(154401,)
[87, 5]
Working on image : 44
(154401,)
[67, 0]
Working on image : 45
(154401,)
[86, 0]
Working on image : 46
(154401,)
[89, 1]
```

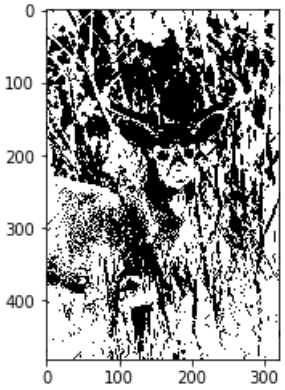
```
Working on image : 47
(154401,)
[128, 1]
Working on image : 48
(154401,)
[132, 16]
Working on image : 49
(154401,)
[134, 17]
```

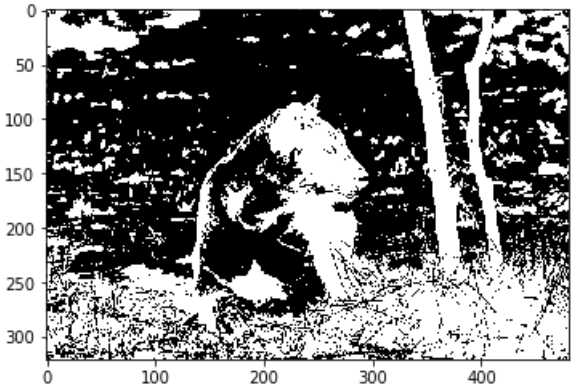


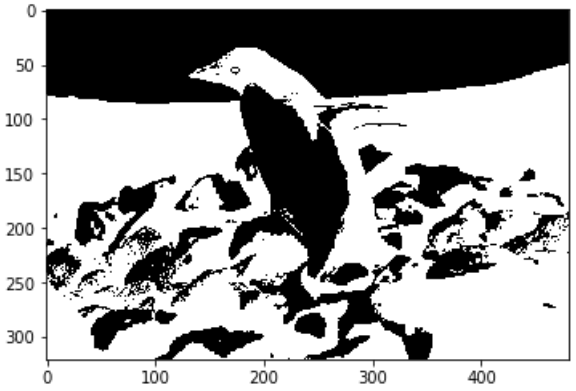


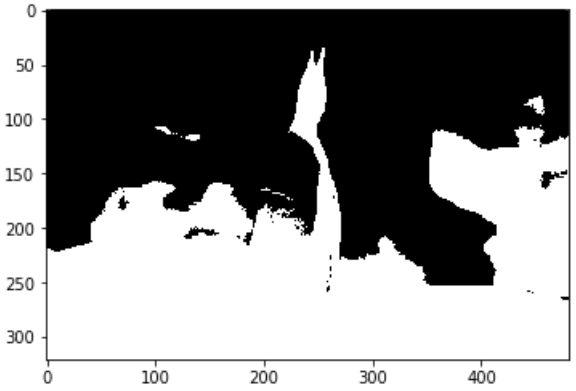


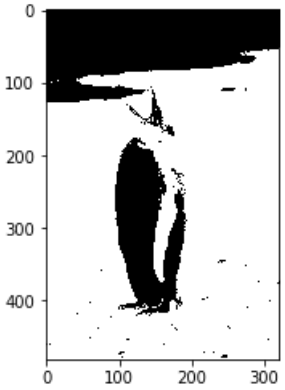


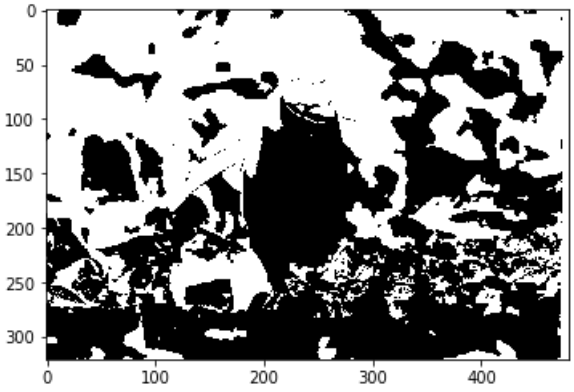

0
100
200
300
400
0
100
200
300

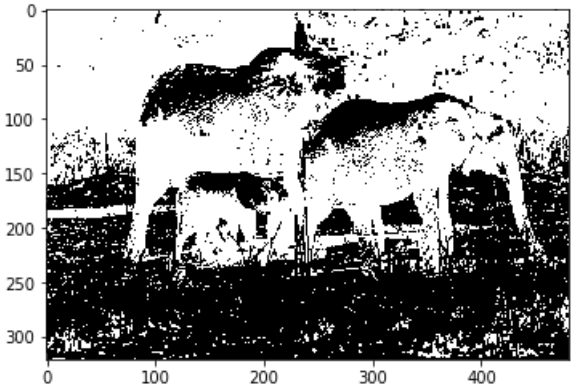

0
50
100
150
200
250
300
0
100
200
300
400

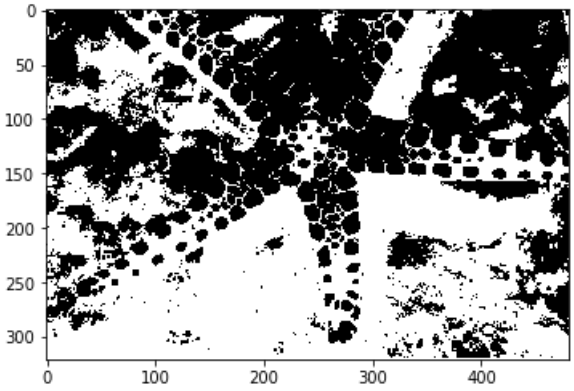

0
50
100
150
200
250
300
0
100
200
300
400

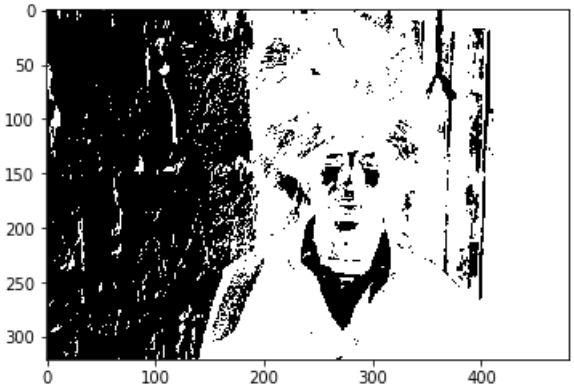

0
50
100
150
200
250
300
0
100
200
300
400

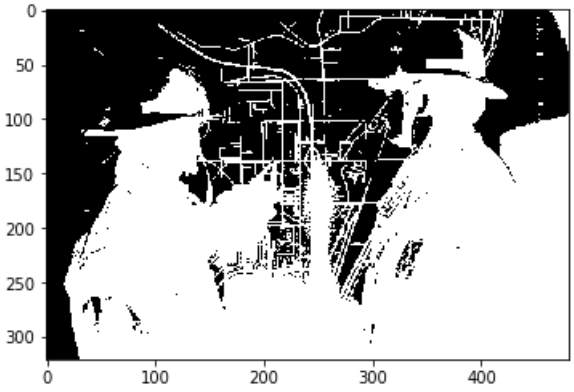

0
100
200
300
400
0
100
200
300

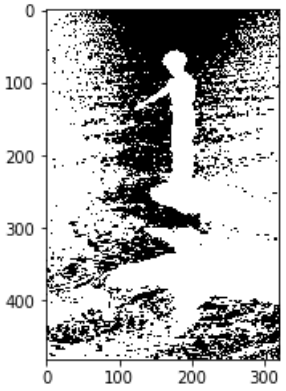

0
50
100
150
200
250
300
0
100
200
300
400

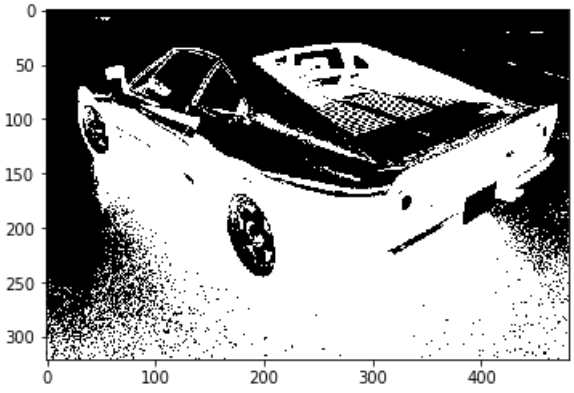

0
50
100
150
200
250
300
0
100
200
300
400

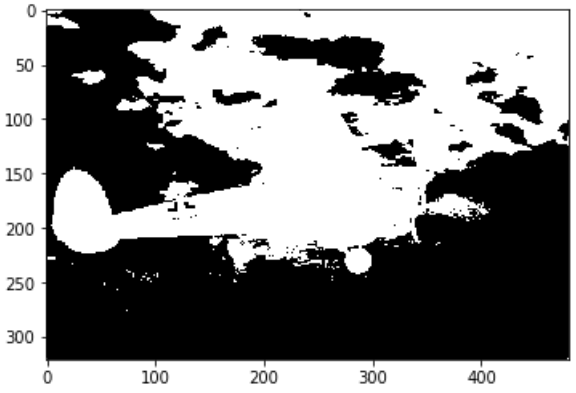

0
50
100
150
200
250
300
0
100
200
300
400

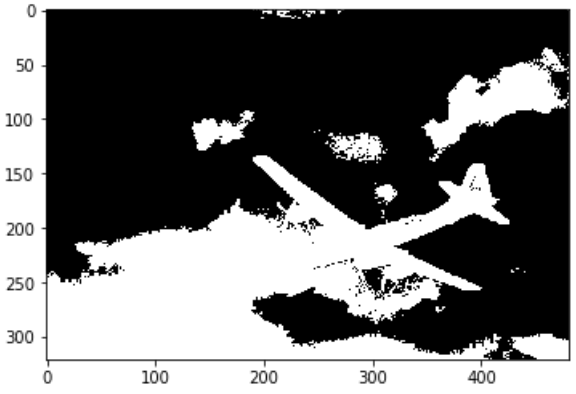

0
50
100
150
200
250
300
0
100
200
300
400

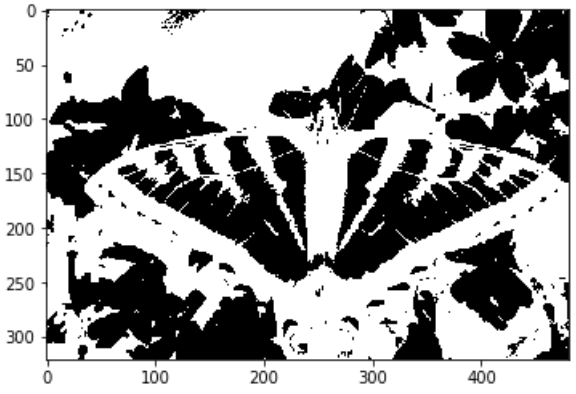

0
50
100
150
200
250
300
0
100
200
300
400

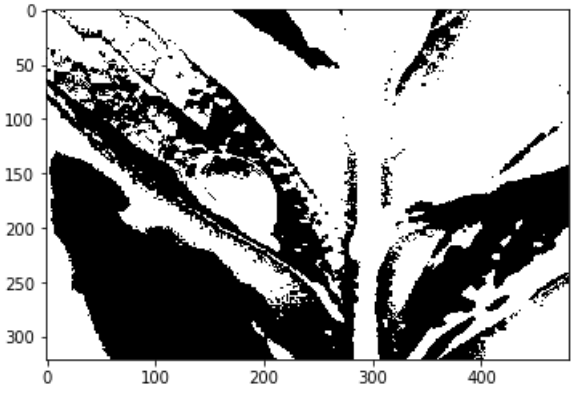

0
100
200
300
400
0
100
200
300

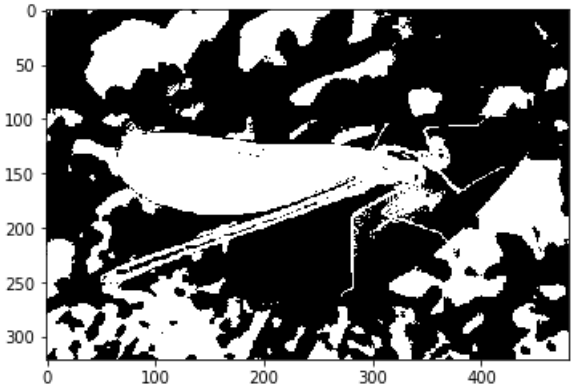

0
50
100
150
200
250
300
0
100
200
300
400

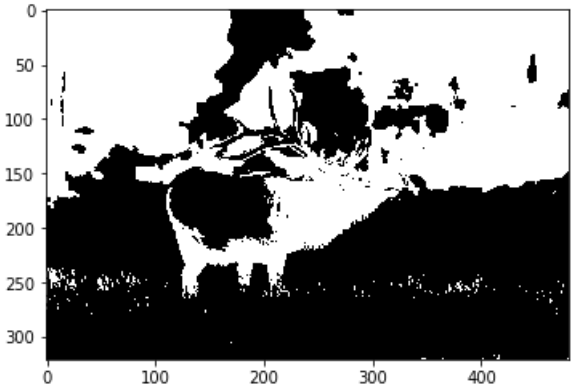

0
50
100
150
200
250
300
0
100
200
300
400

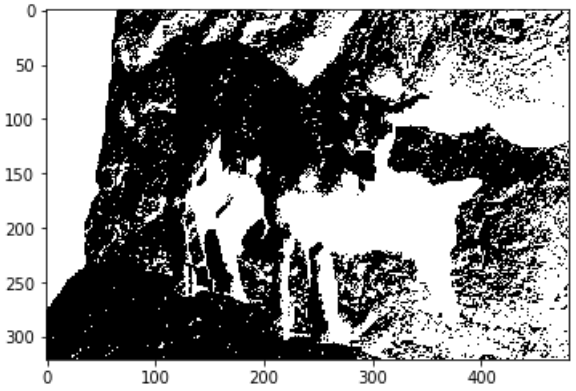

0
50
100
150
200
250
300
0
100
200
300
400

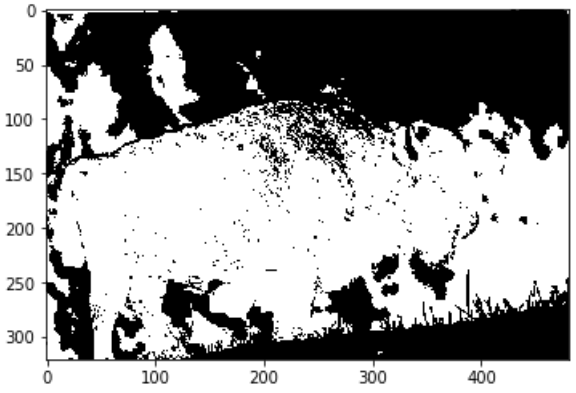

0
50
100
150
200
250
300
0
100
200
300
400

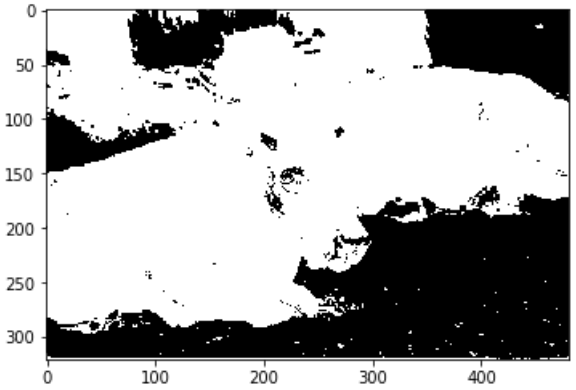

0
50
100
150
200
250
300
0
100
200
300
400

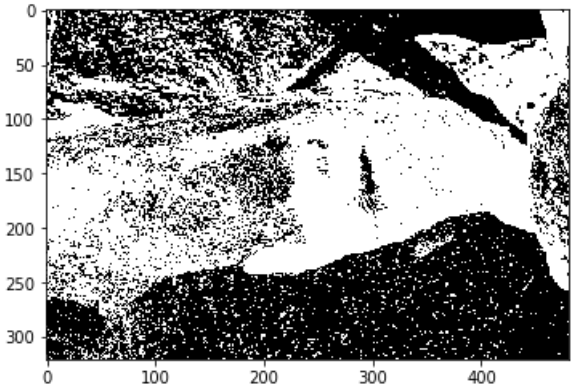

0
50
100
150
200
250
300
0
100
200
300
400

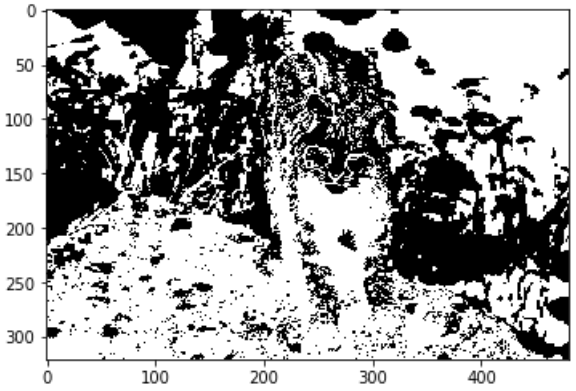

0
50
100
150
200
250
300
0
100
200
300
400

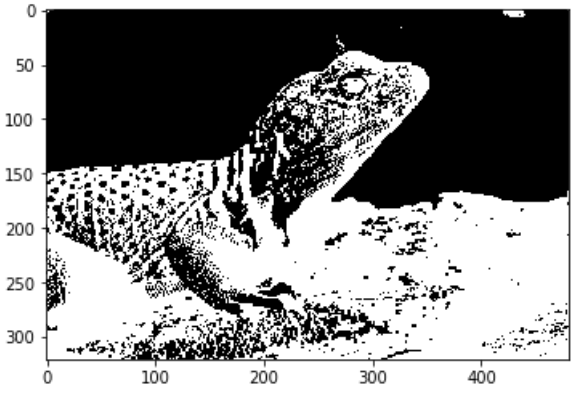

0
50
100
150
200
250
300
0
100
200
300
400

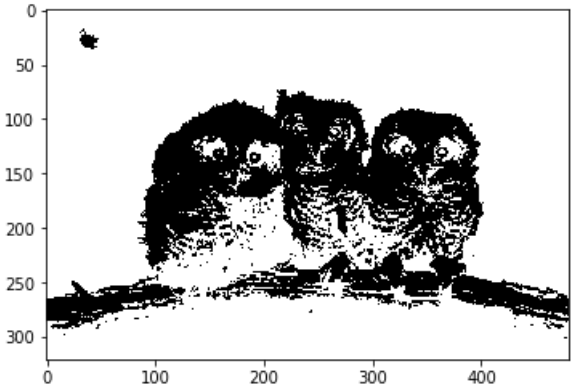

0
50
100
150
200
250
300
0
100
200
300
400

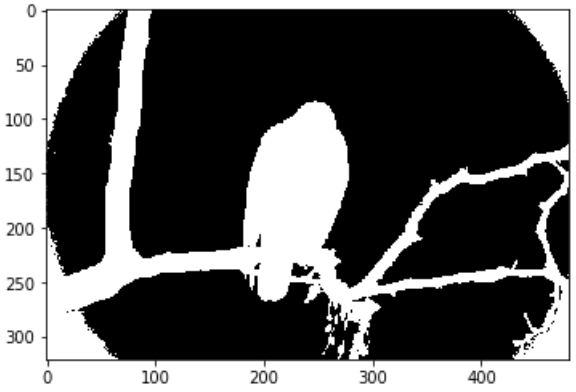

0
50
100
150
200
250
300
0
100
200
300
400

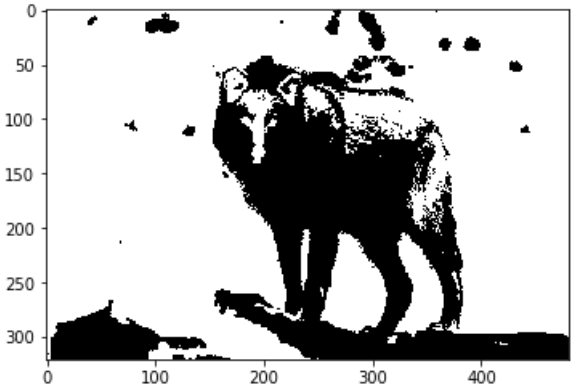

0
50
100
150
200
250
300
0
100
200
300
400

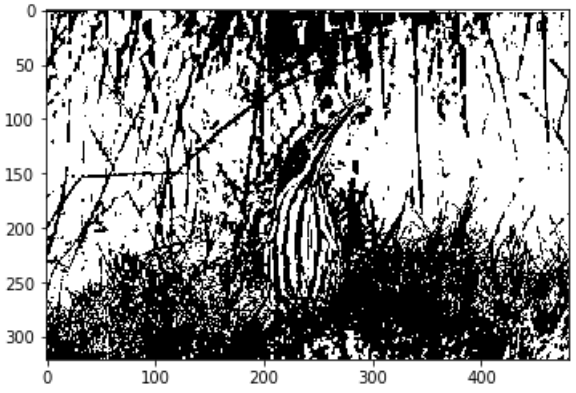

0
50
100
150
200
250
300
0
100
200
300
400

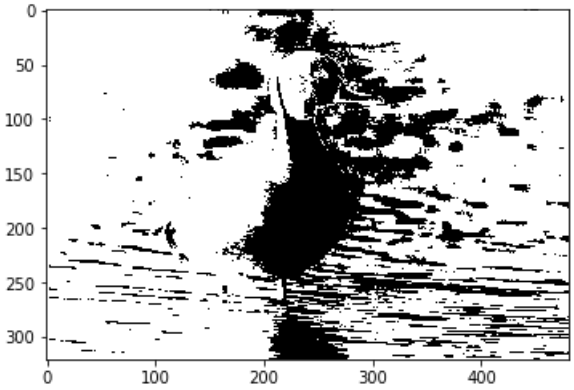

0
50
100
150
200
250
300
0
100
200
300
400

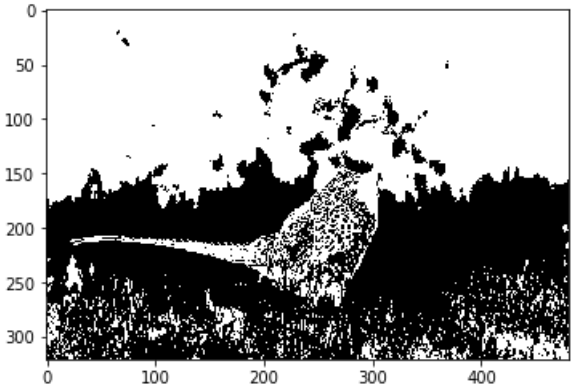

0
50
100
150
200
250
300
0
100
200
300
400

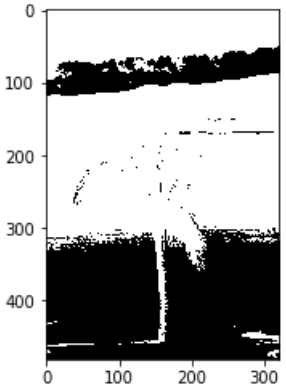

0
50
100
150
200
250
300
0
100
200
300
400

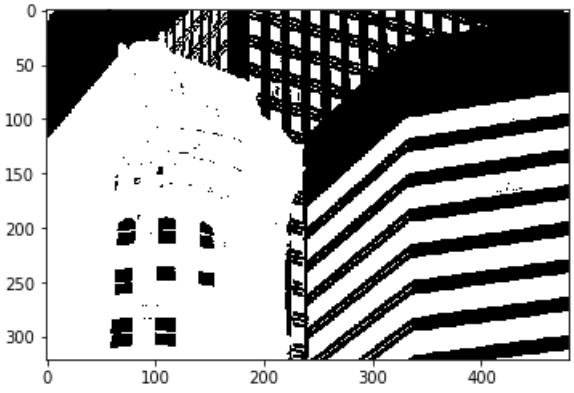

0
50
100
150
200
250
300
0
100
200
300
400

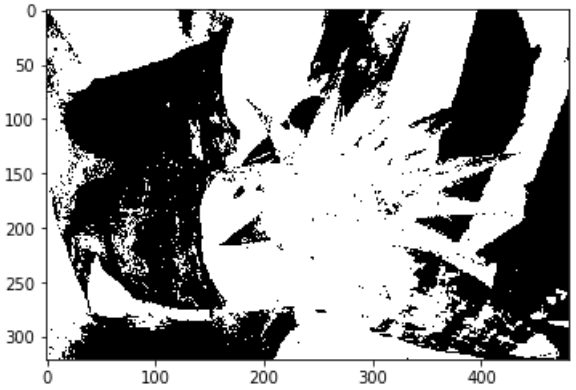

0
50
100
150
200
250
300
0
100
200
300
400

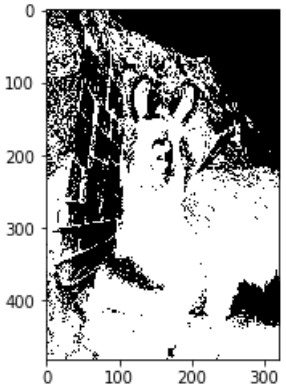

0
50
100
150
200
250
300
0
100
200
300
400

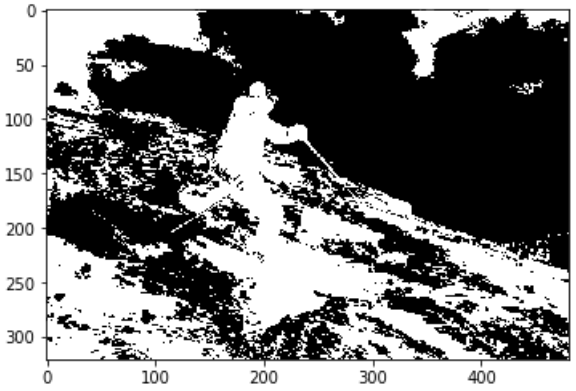

0
50
100
150
200
250
300
0
100
200
300
400

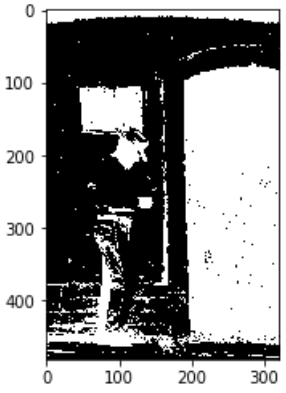

0
100
200
300
400
0
100
200
300

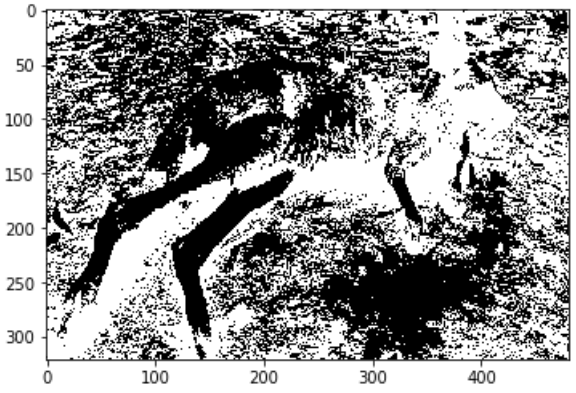

0
50
100
150
200
250
300
0
100
200
300
400

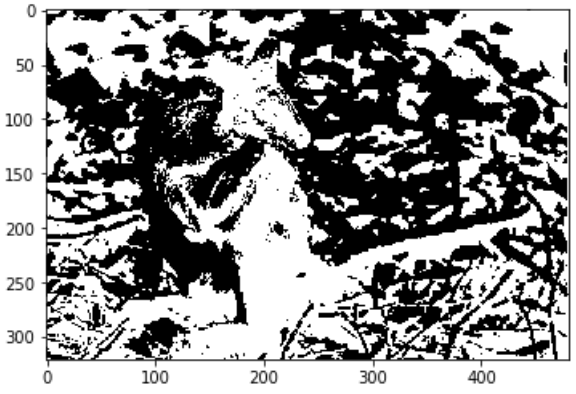

0
50
100
150
200
250
300
0
100
200
300
400

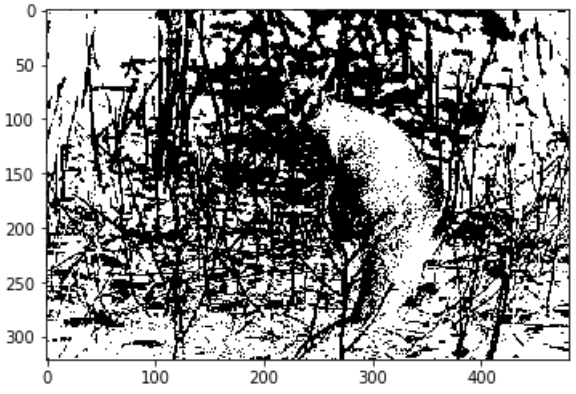

0
100
200
300
400
0
100
200
300

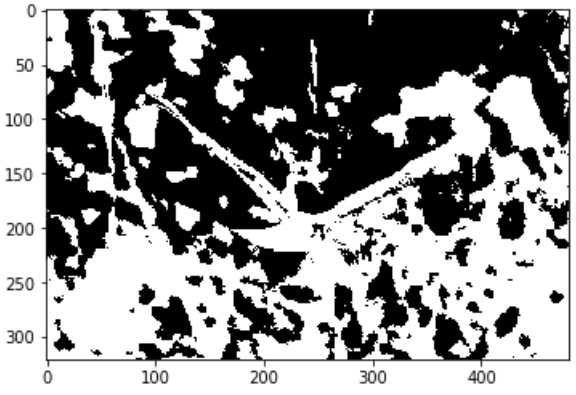

0
50
100
150
200
250
300
0
100
200
300
400

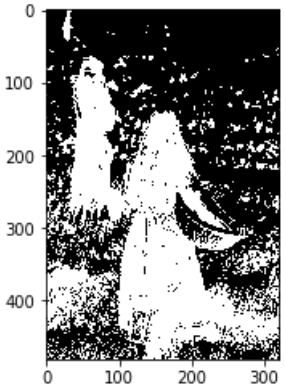

0
100
200
300
400
0
100
200
300

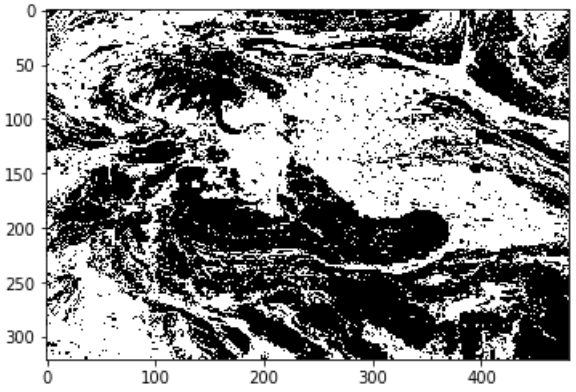

0
50
100
150
200
250
300
0
100
200
300
400

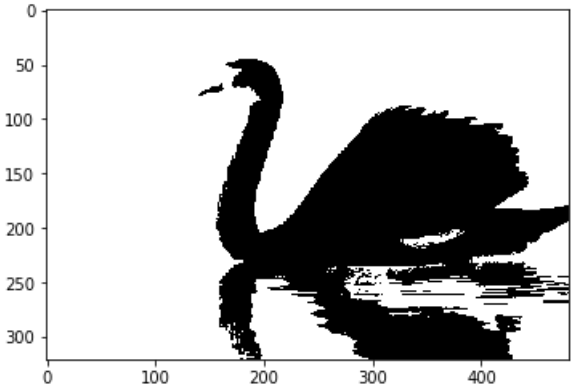

0
50
100
150
200
250
300
0
100
200
300
400

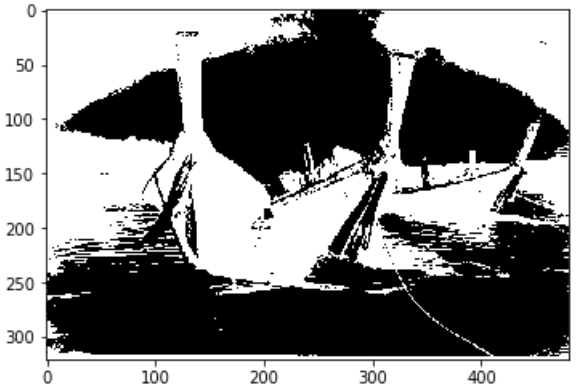

0
50
100
150
200
250
300
0
100
200
300
400

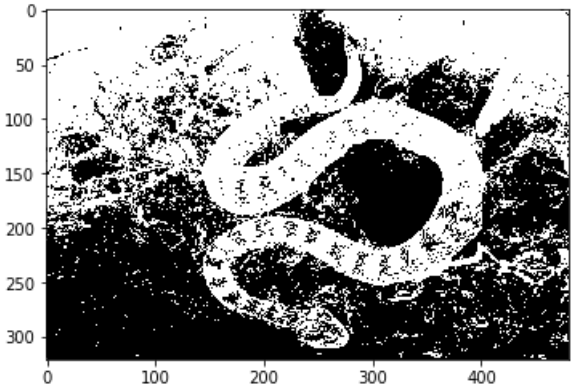

0
50
100
150
200
250
300
0
100
200
300
400

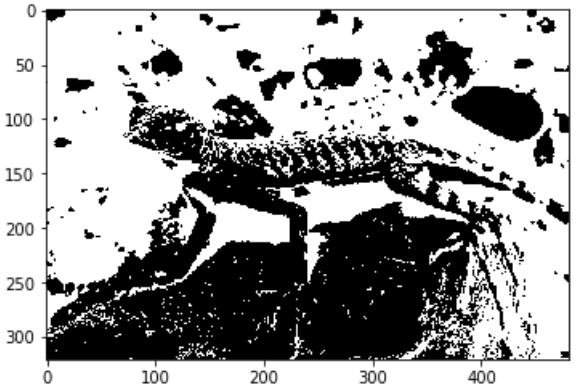

0
100
200
300
400
0
100
200
300


0
50
100
150
200
250
300
0
100
200
300
400


0
50
100
150
200
250
300
0
100
200
300
400

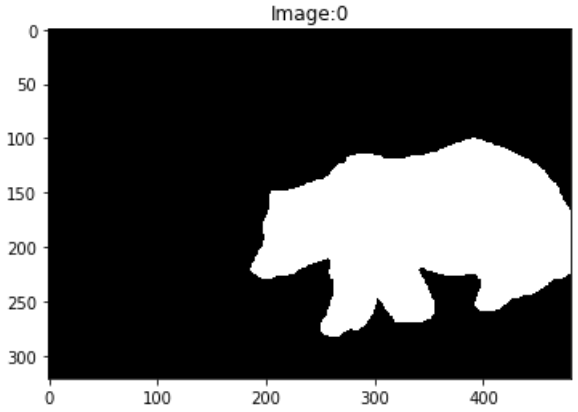

0
50
100
150
200
250
300
0
100
200
300
400

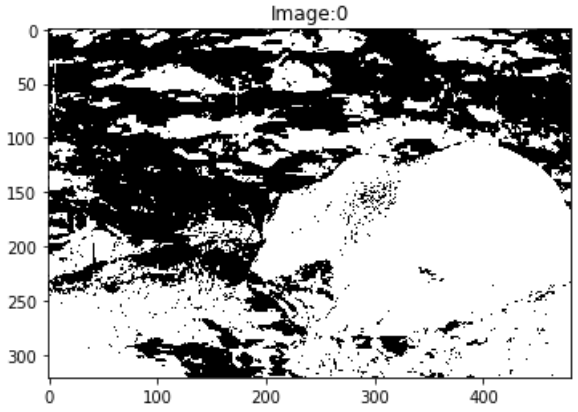

0
50
100
150
200
250
300
0
100
200
300
400

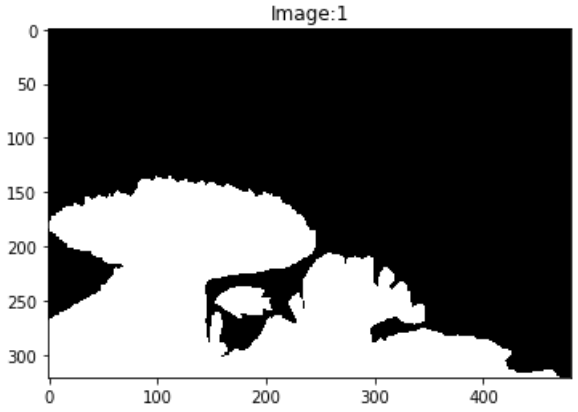

0
50
100
150
200
250
300
0
100
200
300
400

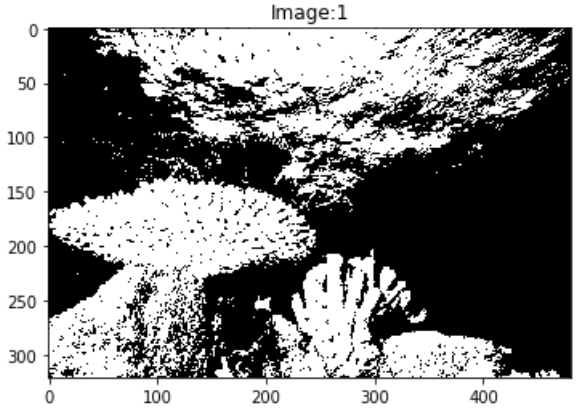
In [25]:
```
for i in range(50):
    plt.figure()
    plt.title('Image:' + str(i))
    plt.imshow(ground_truths[i], cmap = 'gray')
    plt.figure()
    plt.title('Image:' + str(i))
    plt.imshow(results[i], cmap = 'gray')
```

```
C:\Users\Rajat\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: RuntimeWarning: More than 20 f
igures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`)
are retained until explicitly closed and may consume too much memory. (To control this warning, se
e the rcParam `figure.max_open_warning`).

C:\Users\Rajat\Anaconda3\lib\site-packages\ipykernel_launcher.py:5: RuntimeWarning: More than 20 f
igures have been opened. Figures created through the pyplot interface (`matplotlib.pyplot.figure`)
are retained until explicitly closed and may consume too much memory. (To control this warning, se
e the rcParam `figure.max_open_warning`).
  """
```









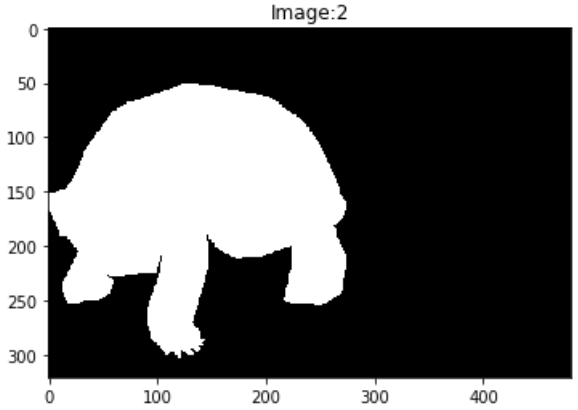


Image:2
0
50
100
150
200
250
300
0
100
200
300
400

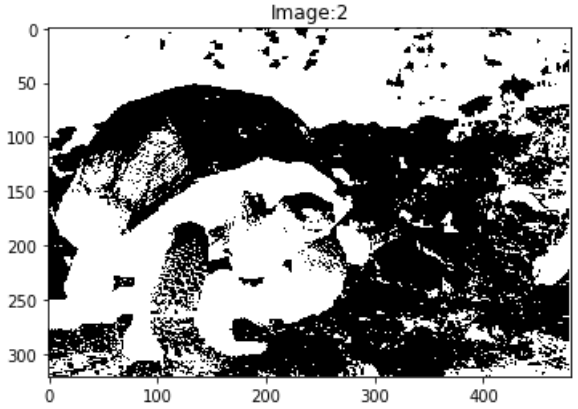


Image:2
0
50
100
150
200
250
300
0
100
200
300
400

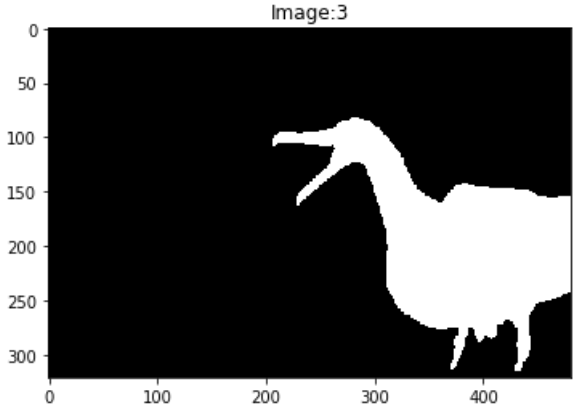


Image:3
0
50
100
150
200
250
300
0
100
200
300
400

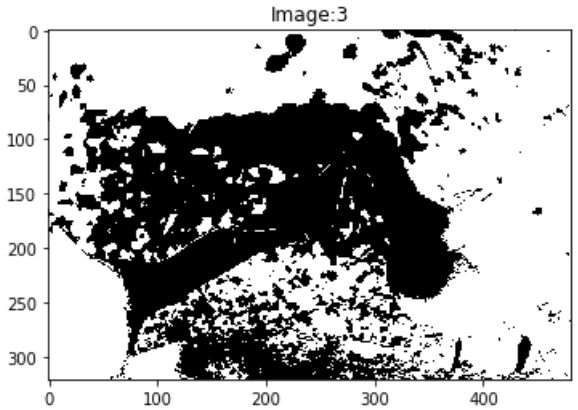


Image:3
0
50
100
150
200
250
300
0
100
200
300
400

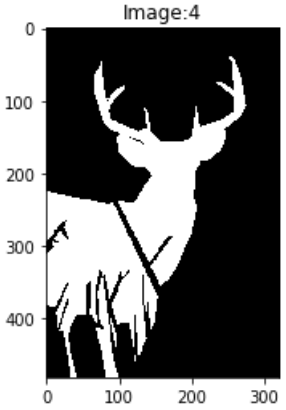


Image:4
0
100
200
300
400
0
100
200
300

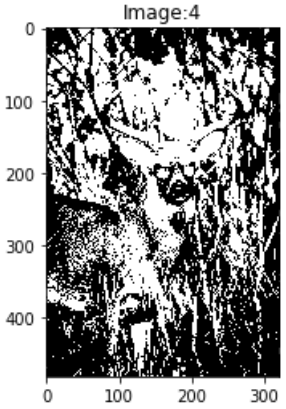


Image:4
0
100
200
300
400
0
100
200
300

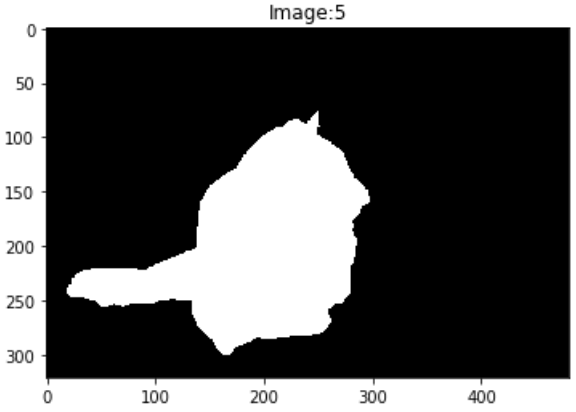


Image:5
0
50
100
150
200
250
300
0
100
200
300
400

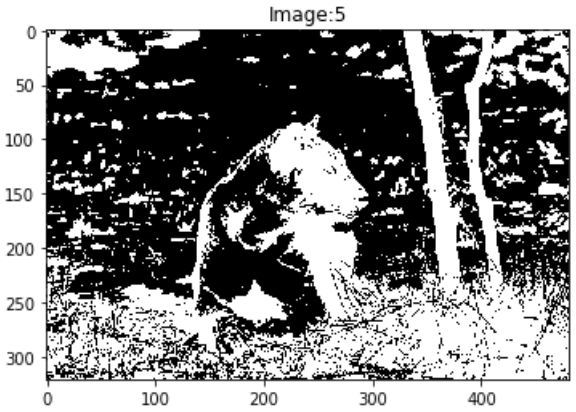


Image:5
0
50
100
150
200
250
300
0
100
200
300
400

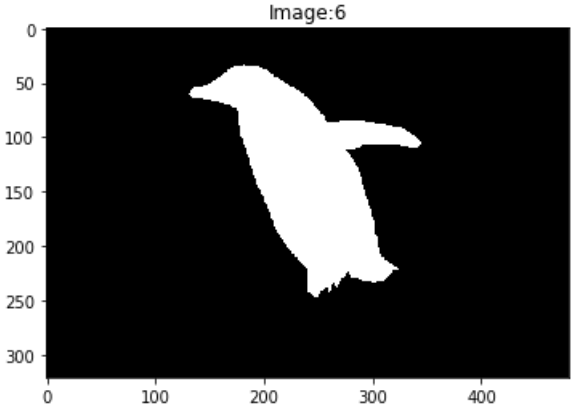


Image:6
0
50
100
150
200
250
300
0
100
200
300
400

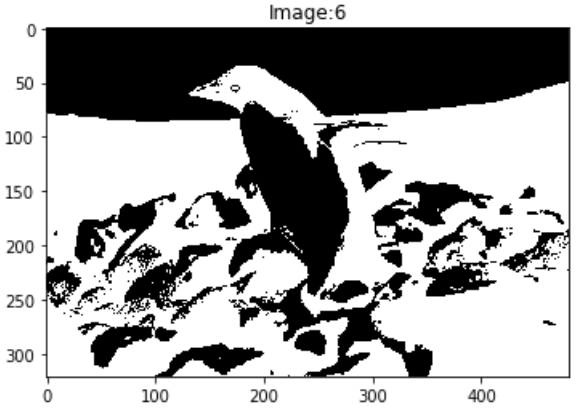


Image:6
0
50
100
150
200
250
300
0
100
200
300
400

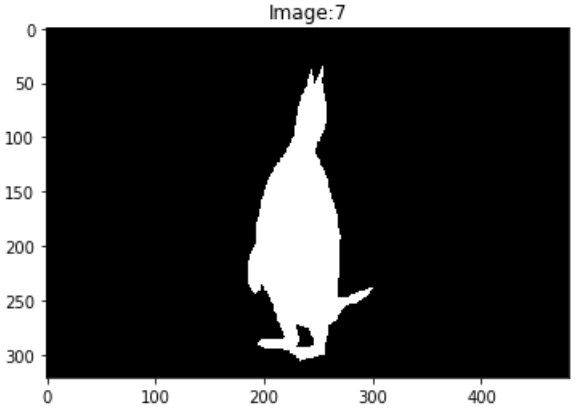


Image:7
0
50
100
150
200
250
300
0
100
200
300
400

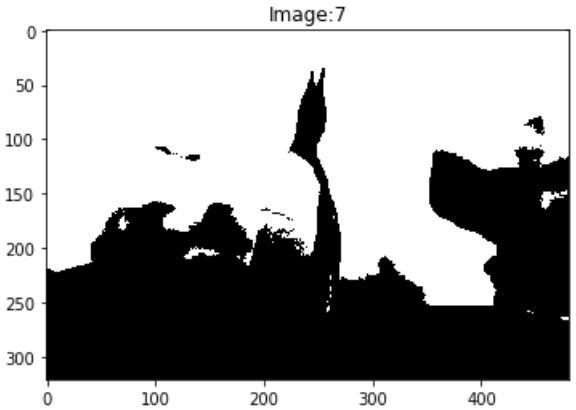


Image:7
0
50
100
150
200
250
300
0
100
200
300
400

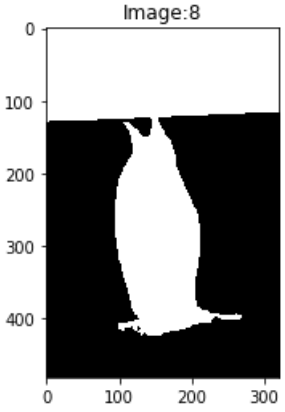


Image:8
0
100
200
300
400
0
100
200
300

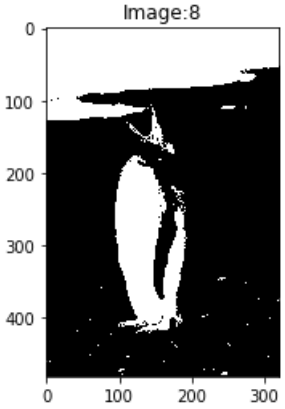


Image:8
0
100
200
300
400
0
100
200
300

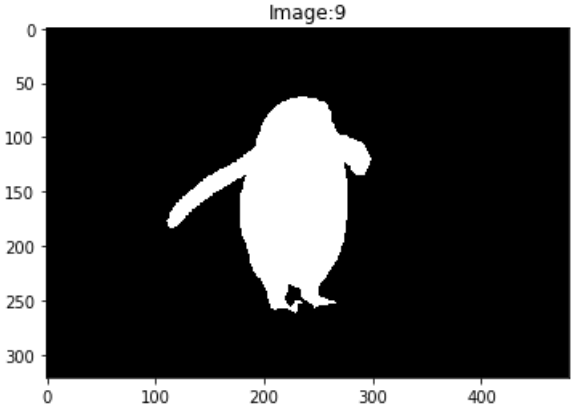


Image:9
0
50
100
150
200
250
300
0
100
200
300
400

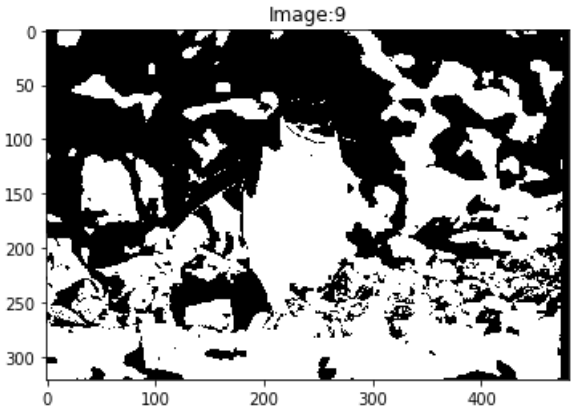


Image:9
0
50
100
150
200
250
300
0
100
200
300
400


Image:10
0
50
100
150
200
250
300
0
100
200
300
400

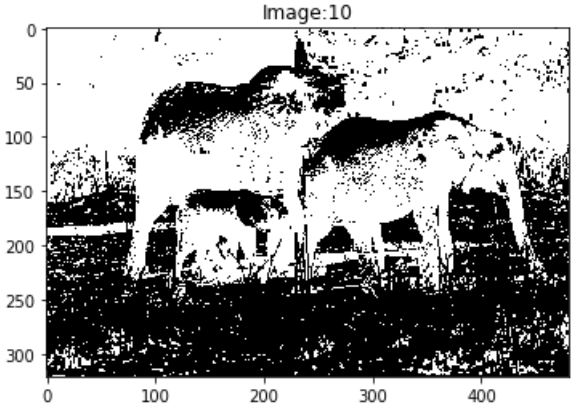


Image:10
0
50
100
150
200
250
300
0
100
200
300
400

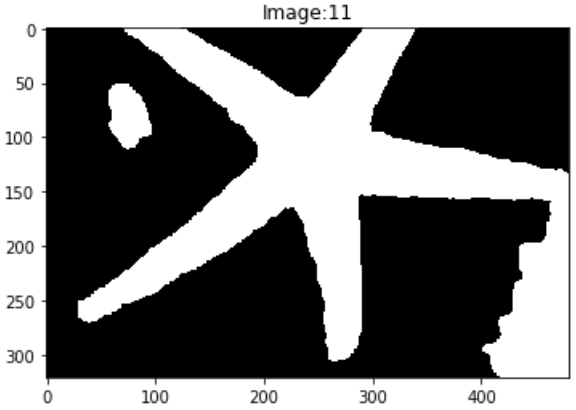


Image:11
0
50
100
150
200
250
300
0
100
200
300
400

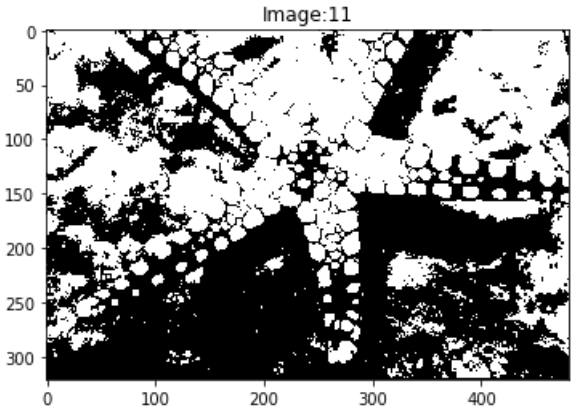


Image:11
0
50
100
150
200
250
300
0
100
200
300
400

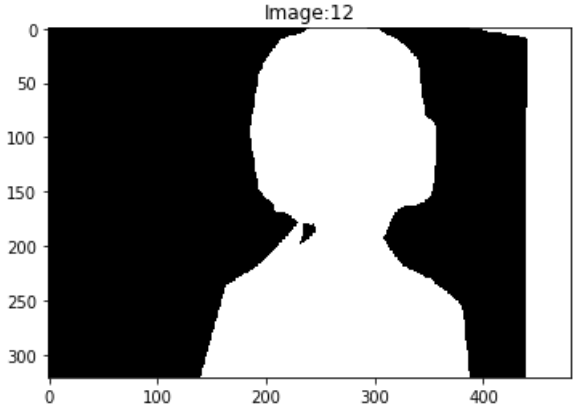


Image:12
0
50
100
150
200
250
300
0
100
200
300
400

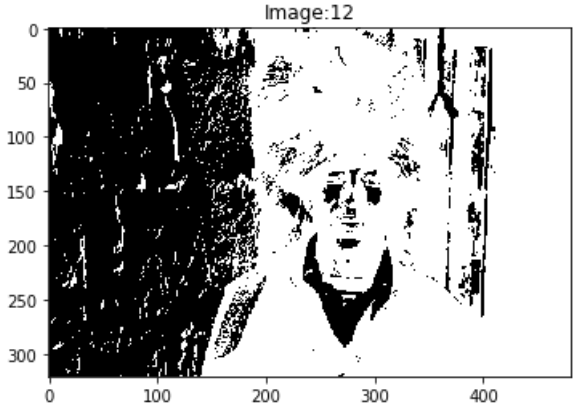


Image:12
0
50
100
150
200
250
300
0
100
200
300
400

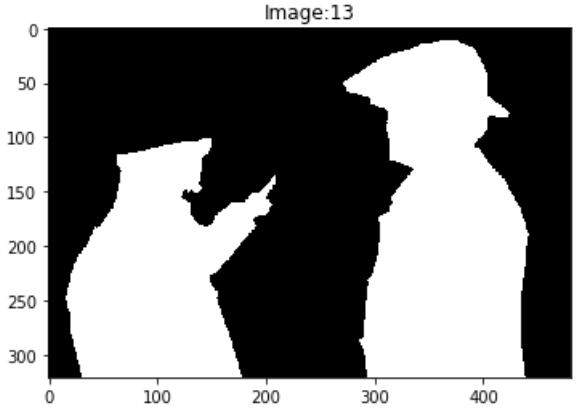


Image:13
0
50
100
150
200
250
300
0
100
200
300
400

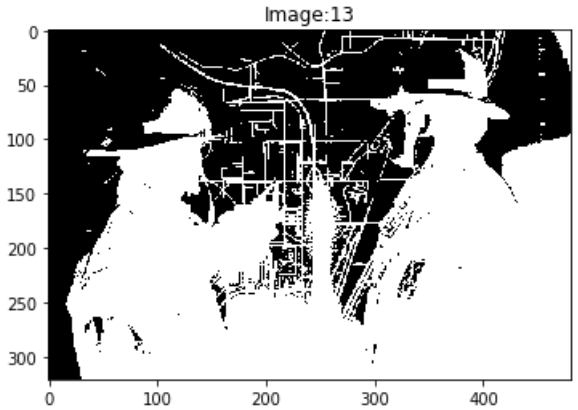


Image:13
0
50
100
150
200
250
300
0
100
200
300
400

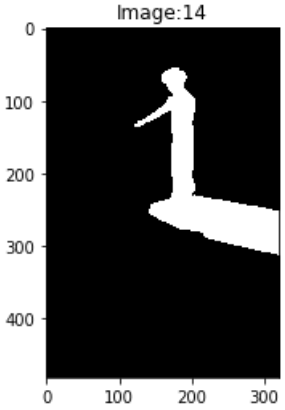


Image:14
0
100
200
300
400
0
100
200
300

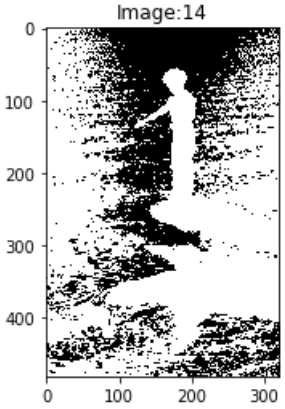


Image:14
0
100
200
300
400
0
100
200
300

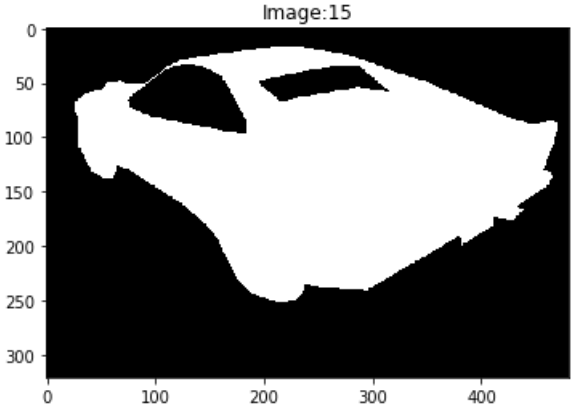


Image:15
0
50
100
150
200
250
300
0
100
200
300
400

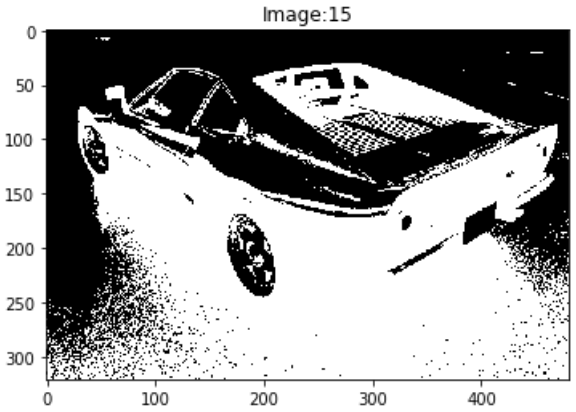


Image:15
0
50
100
150
200
250
300
0
100
200
300
400

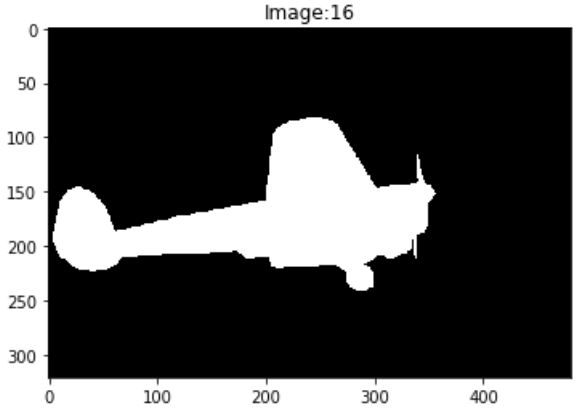


Image:16
0
50
100
150
200
250
300
0
100
200
300
400

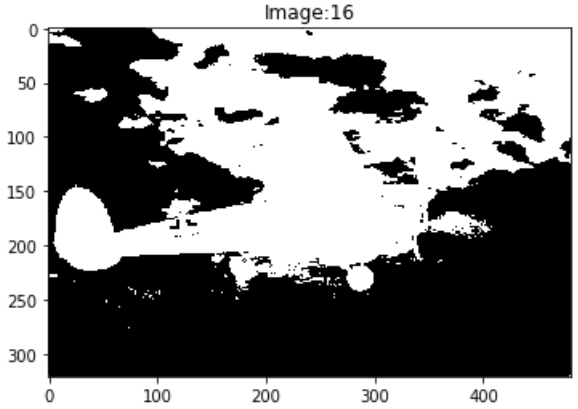


Image:16
0
50
100
150
200
250
300
0
100
200
300
400

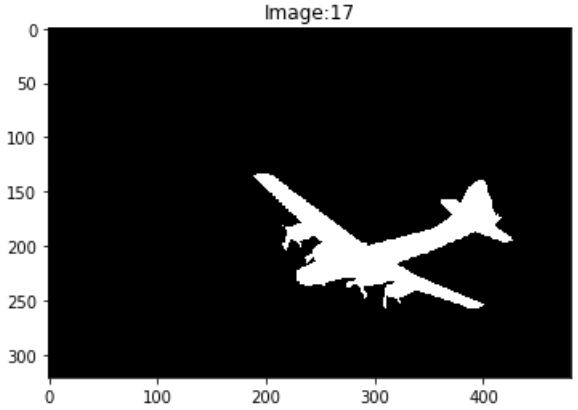


Image:17
0
50
100
150
200
250
300
0
100
200
300
400

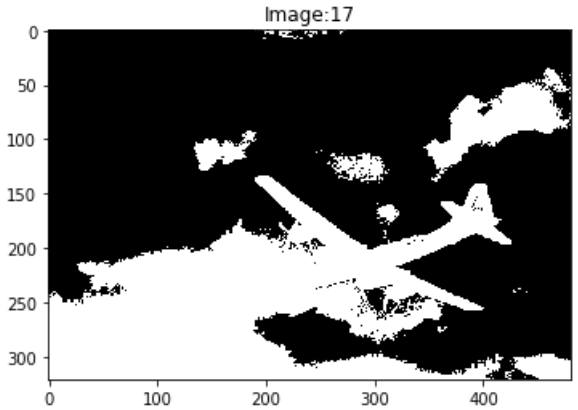


Image:17
0
50
100
150
200
250
300
0
100
200
300
400

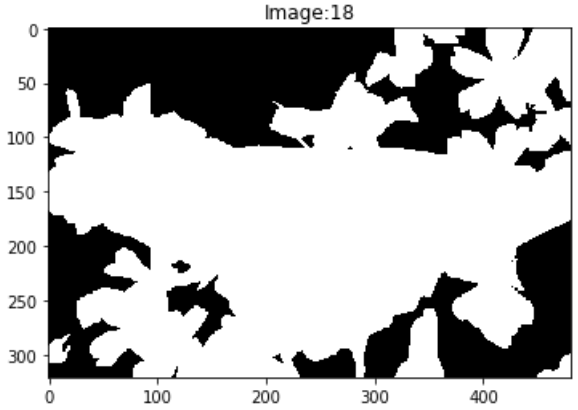


Image:18
0
50
100
150
200
250
300
0
100
200
300
400

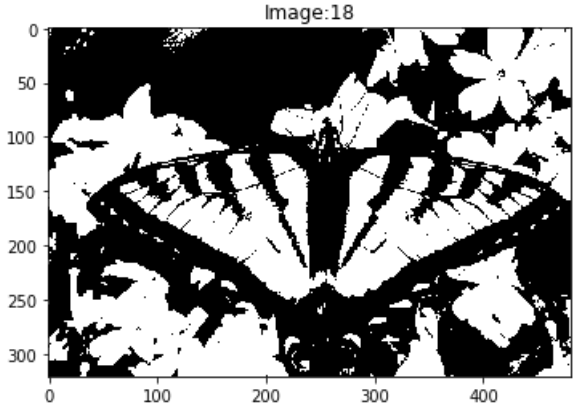


Image:18
0
50
100
150
200
250
300
0
100
200
300
400

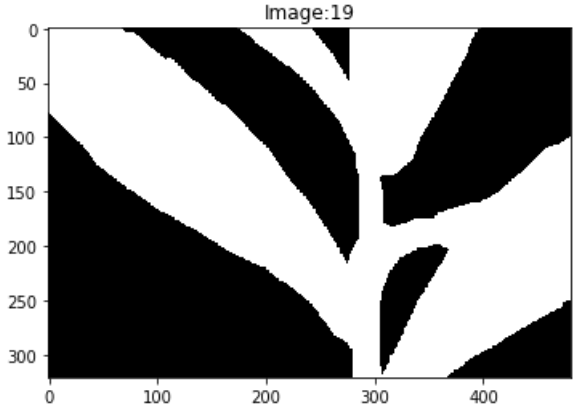


Image:19
0
50
100
150
200
250
300
0
100
200
300
400

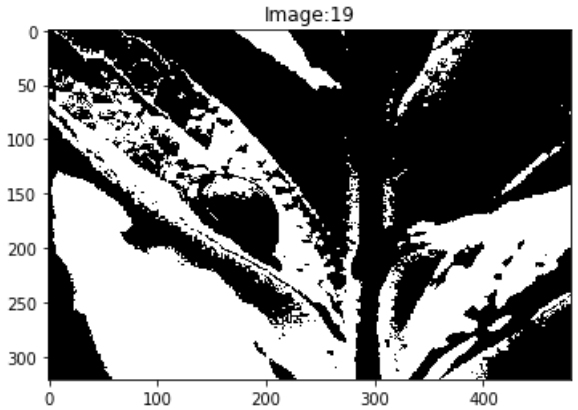


Image:19
0
50
100
150
200
250
300
0
100
200
300
400

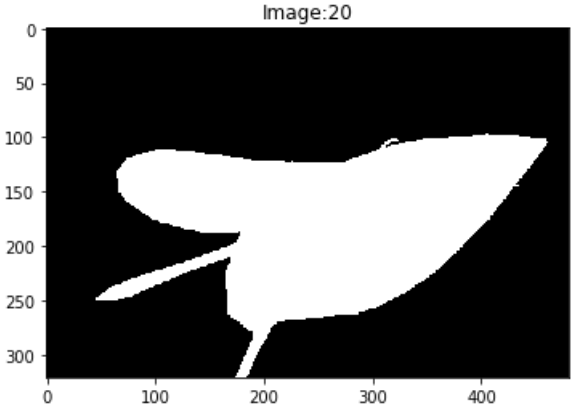


Image:20
0
50
100
150
200
250
300
0
100
200
300
400

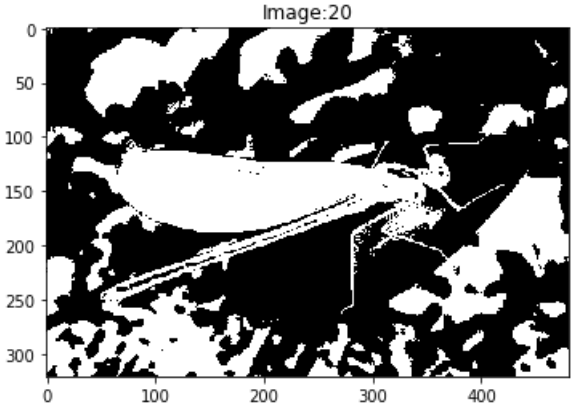


Image:20
0
50
100
150
200
250
300
0
100
200
300
400

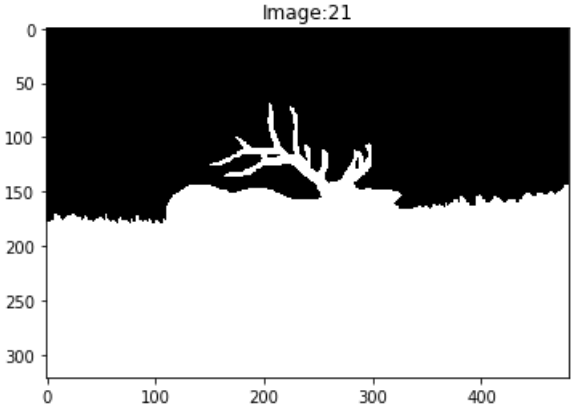


Image:21
0
50
100
150
200
250
300
0
100
200
300
400

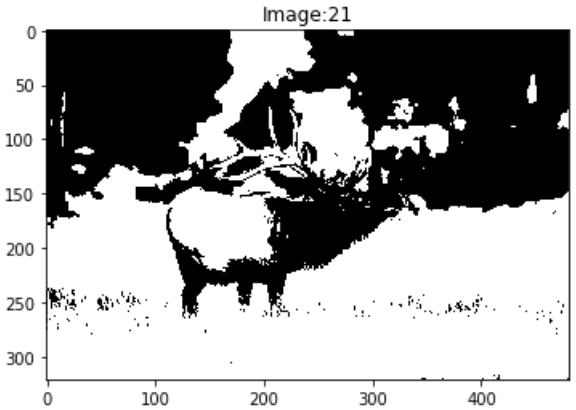


Image:21
0
50
100
150
200
250
300
0
100
200
300
400


Image:22
0
50
100
150
200
250
300
0
100
200
300
400

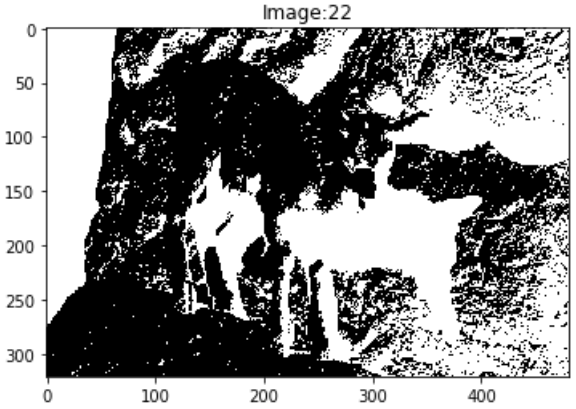


Image:22
0
50
100
150
200
250
300
0
100
200
300
400

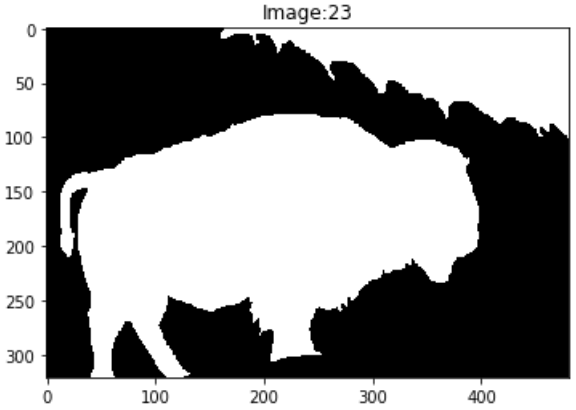


Image:23
0
50
100
150
200
250
300
0
100
200
300
400

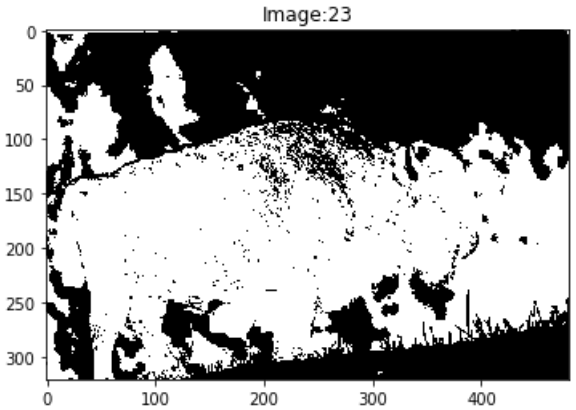


Image:23
0
50
100
150
200
250
300
0
100
200
300
400

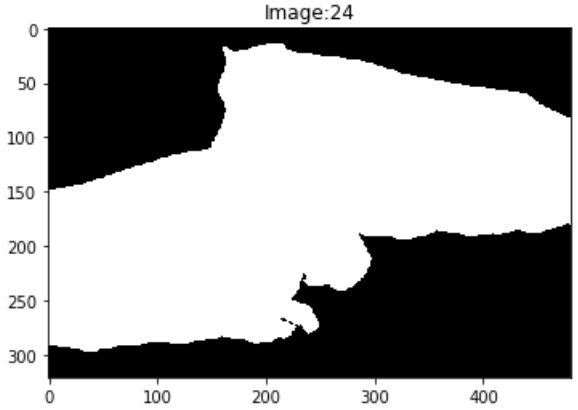


Image:24
0
50
100
150
200
250
300
0
100
200
300
400

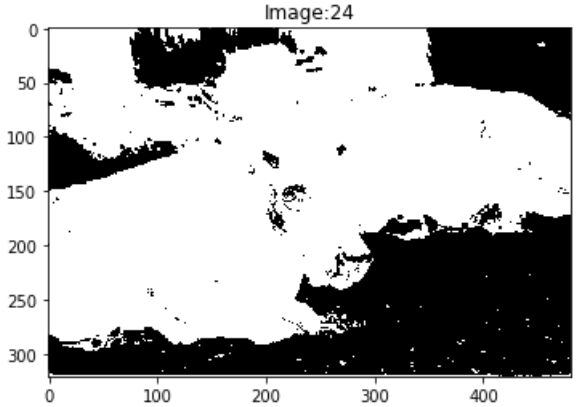


Image:24
0
50
100
150
200
250
300
0
100
200
300
400

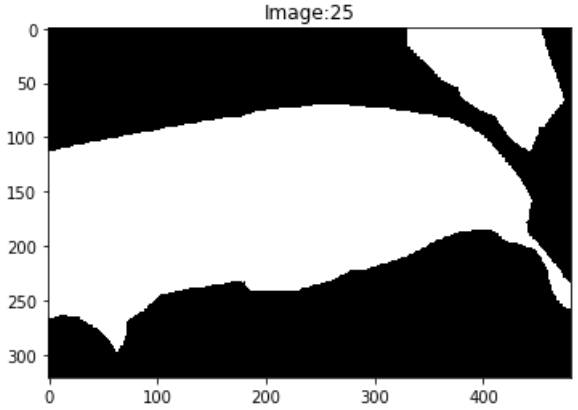


Image:25
0
50
100
150
200
250
300
0
100
200
300
400

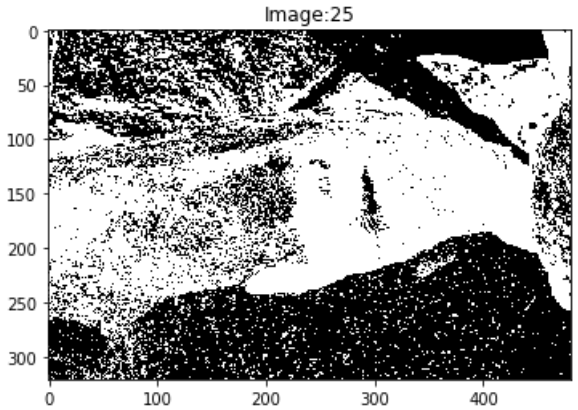


Image:25
0
50
100
150
200
250
300
0
100
200
300
400

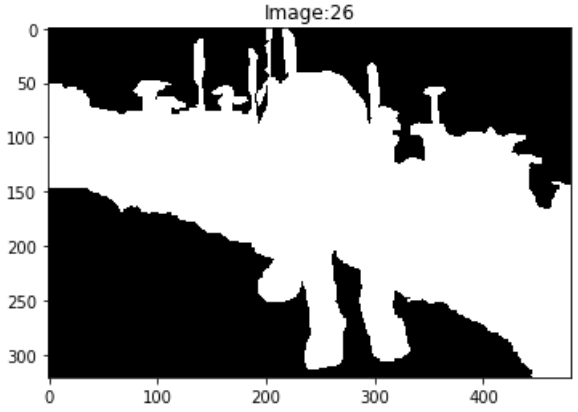


Image:26
0
50
100
150
200
250
300
0
100
200
300
400

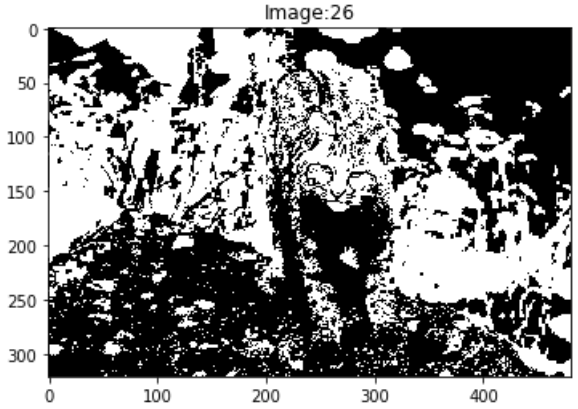


Image:26
0
50
100
150
200
250
300
0
100
200
300
400

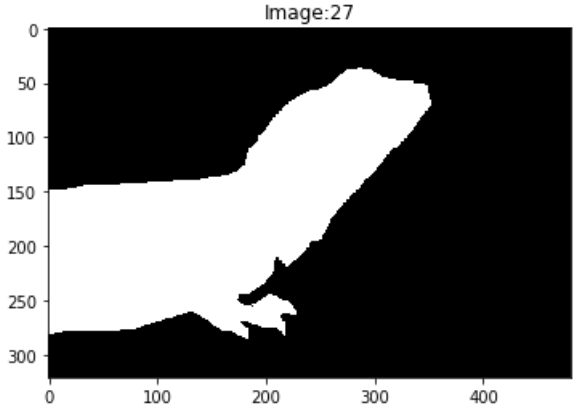


Image:27
0
50
100
150
200
250
300
0
100
200
300
400

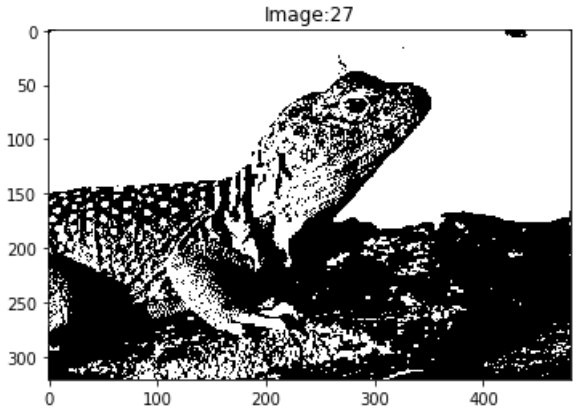


Image:27
0
50
100
150
200
250
300
0
100
200
300
400

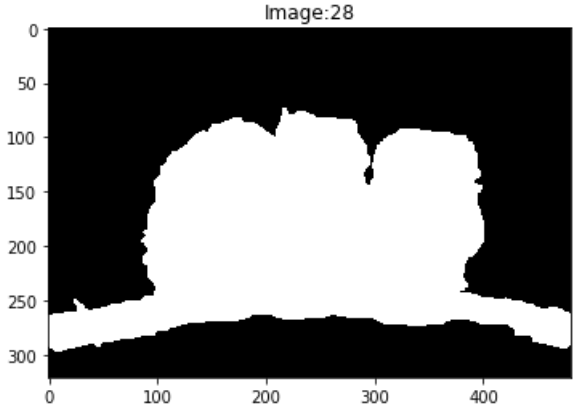


Image:28
0
50
100
150
200
250
300
0
100
200
300
400

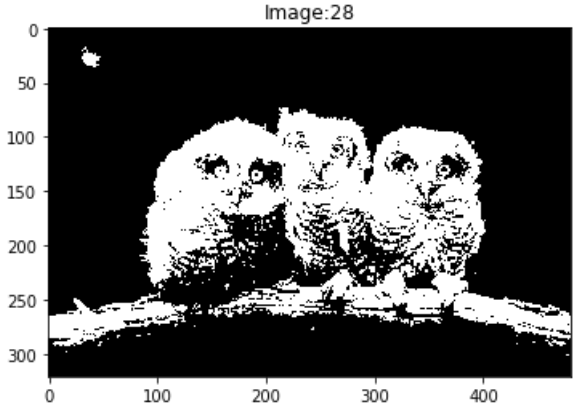


Image:28
0
50
100
150
200
250
300
0
100
200
300
400

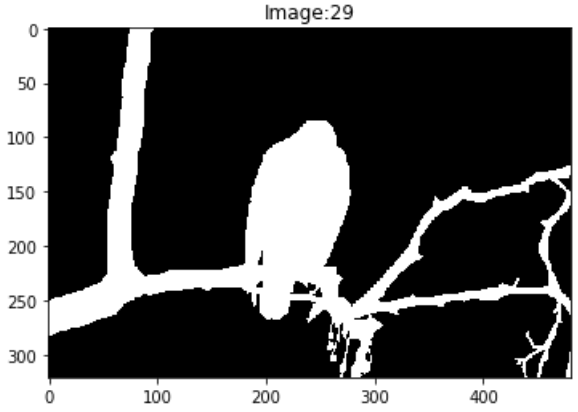


Image:29
0
50
100
150
200
250
300
0
100
200
300
400

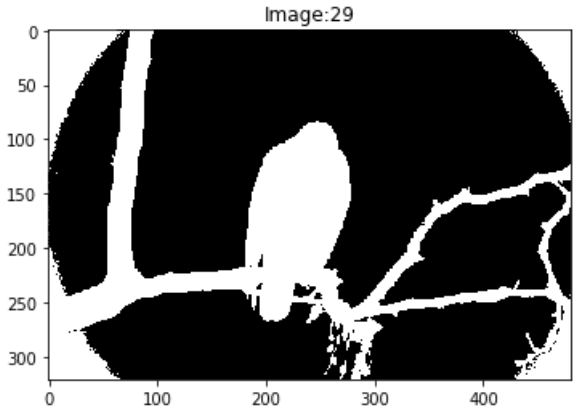


Image:29
0
50
100
150
200
250
300
0
100
200
300
400

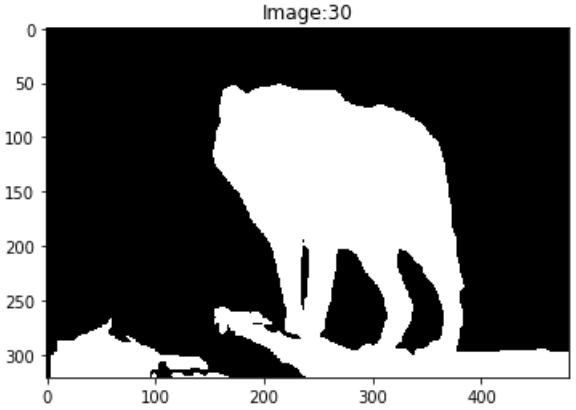


Image:30
0
50
100
150
200
250
300
0
100
200
300
400

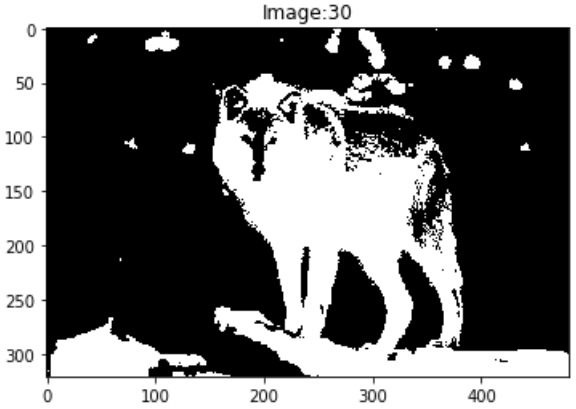


Image:30
0
50
100
150
200
250
300
0
100
200
300
400

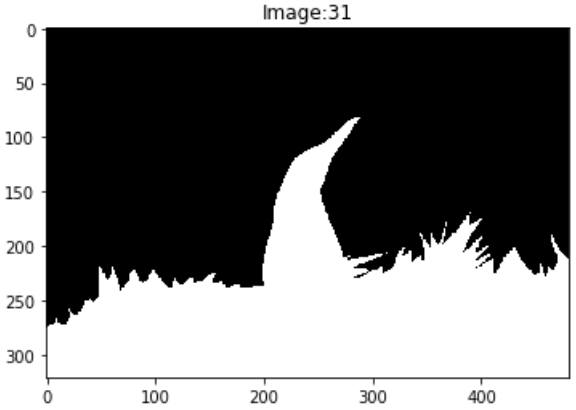


Image:31
0
50
100
150
200
250
300
0
100
200
300
400

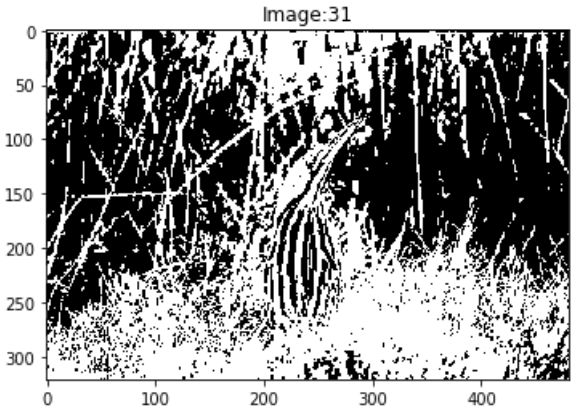


Image:31
0
50
100
150
200
250
300
0
100
200
300
400

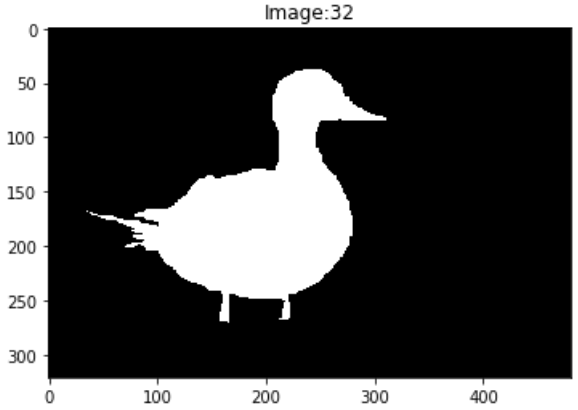


Image:32
0
50
100
150
200
250
300
0
100
200
300
400

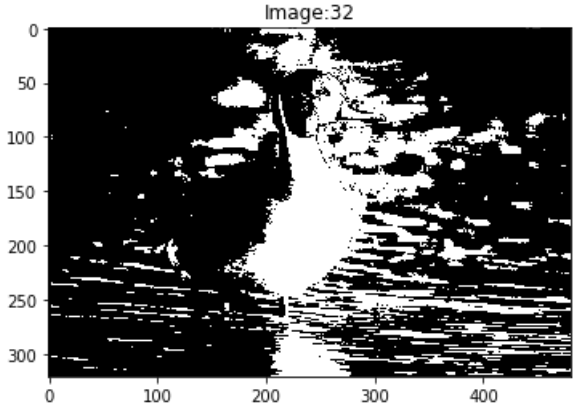


Image:32
0
50
100
150
200
250
300
0
100
200
300
400

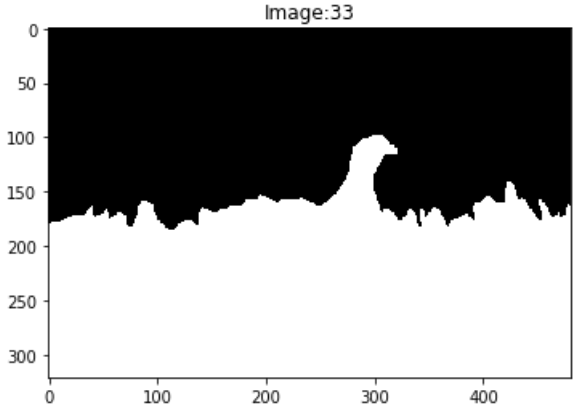


Image:33
0
50
100
150
200
250
300
0
100
200
300
400

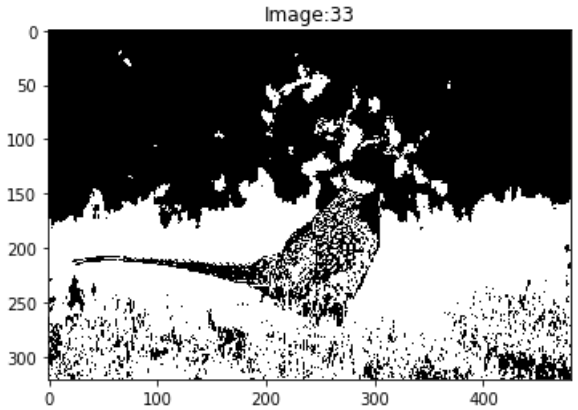


Image:33
0
50
100
150
200
250
300
0
100
200
300
400

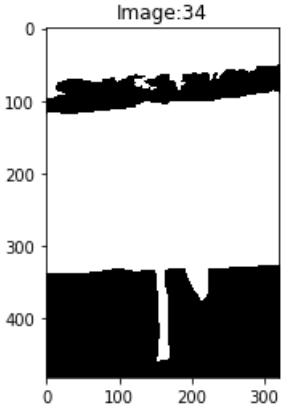


Image:34
0
100
200
300
400
0
100
200
300

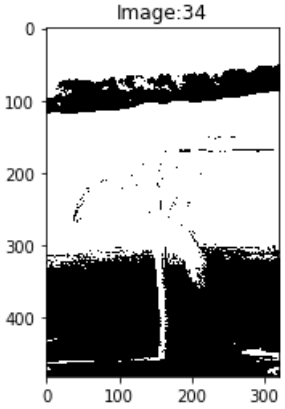


Image:34
0
100
200
300
400
0
100
200
300

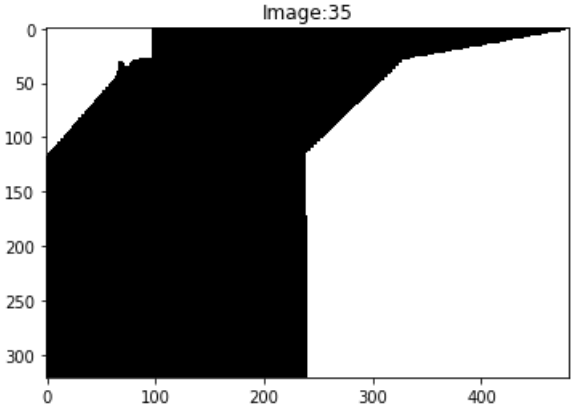


Image:35
0
50
100
150
200
250
300
0
100
200
300
400

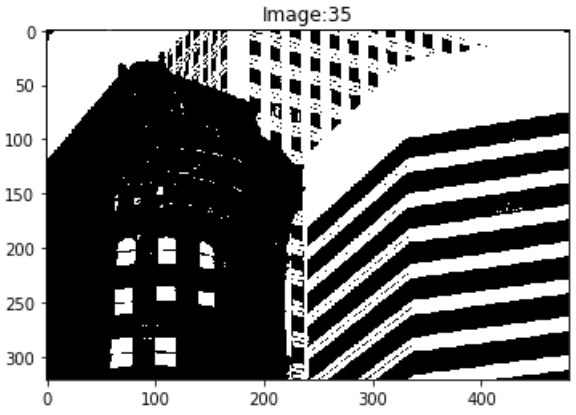


Image:35
0
50
100
150
200
250
300
0
100
200
300
400

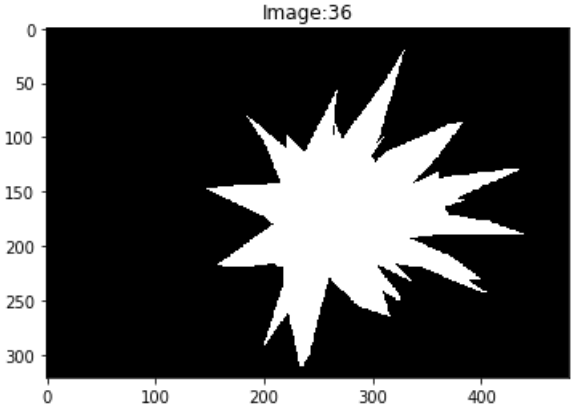


Image:36
0
50
100
150
200
250
300
0
100
200
300
400

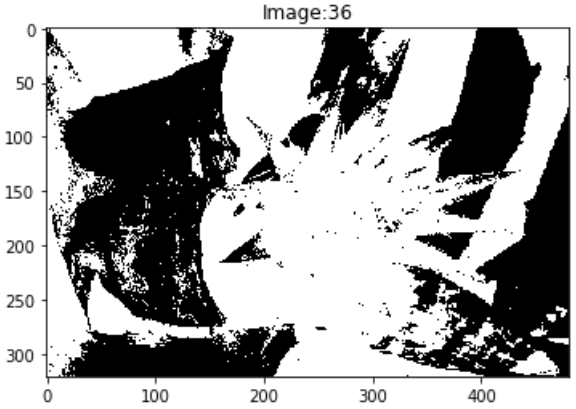


Image:36
0
50
100
150
200
250
300
0
100
200
300
400

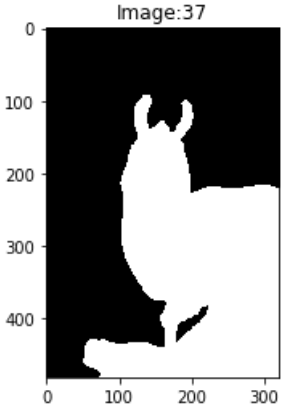


Image:37
0
100
200
300
400
0
100
200
300

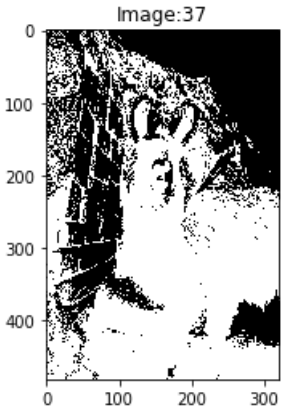


Image:37
0
100
200
300
400
0
100
200
300

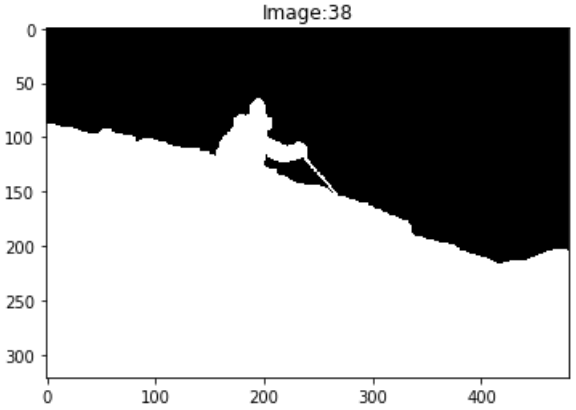


Image:38
0
50
100
150
200
250
300
0
100
200
300
400

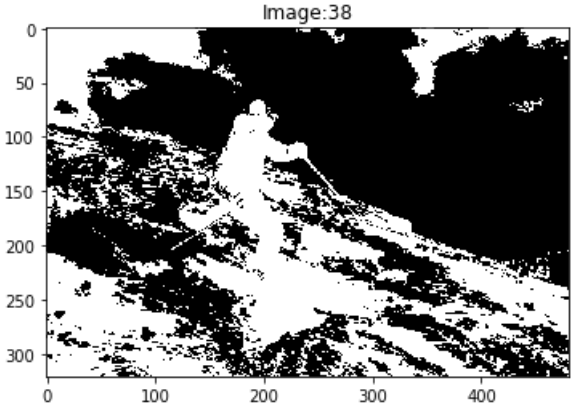


Image:38
0
50
100
150
200
250
300
0
100
200
300
400

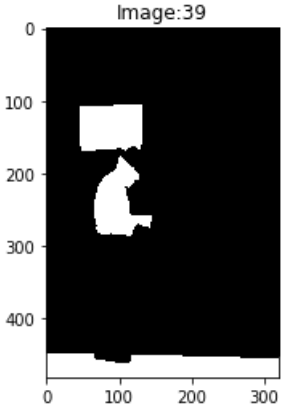


Image:39
0
100
200
300
400
0
100
200
300

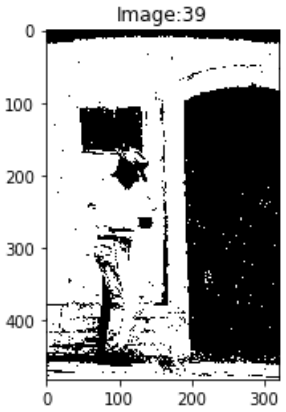


Image:39
0
100
200
300
400
0
100
200
300

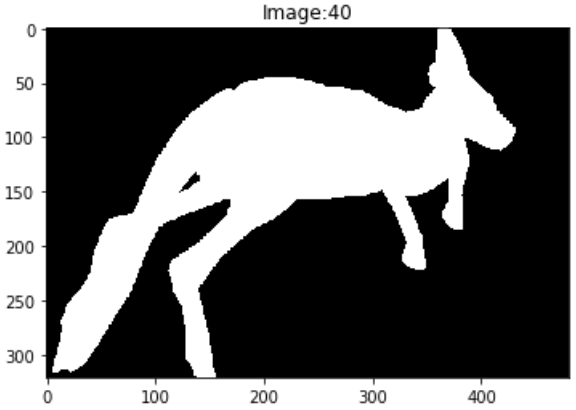


Image:40
0
50
100
150
200
250
300
0
100
200
300
400

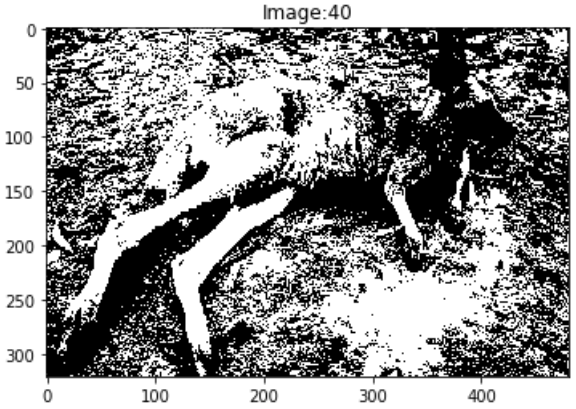


Image:40
0
50
100
150
200
250
300
0
100
200
300
400

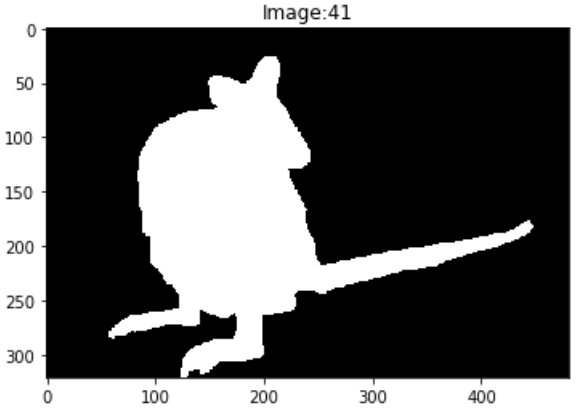


Image:41
0
50
100
150
200
250
300
0
100
200
300
400

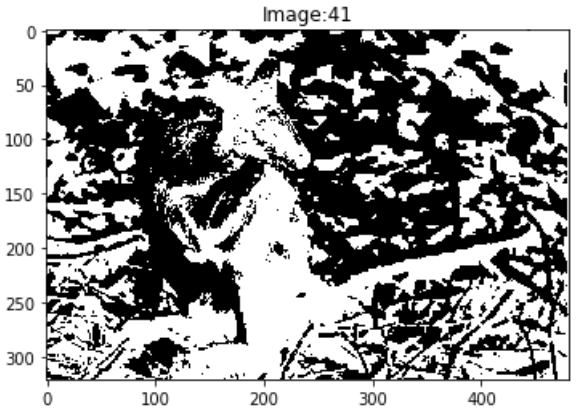


Image:41
0
50
100
150
200
250
300
0
100
200
300
400

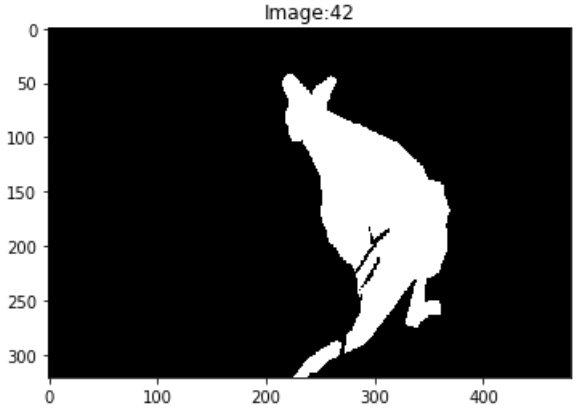


Image:42
0
50
100
150
200
250
300
0
100
200
300
400

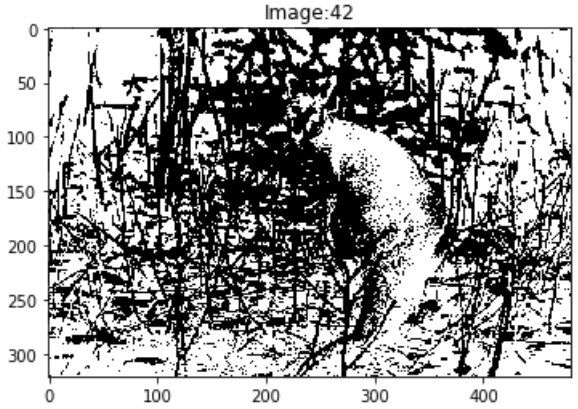


Image:42
0
50
100
150
200
250
300
0
100
200
300
400

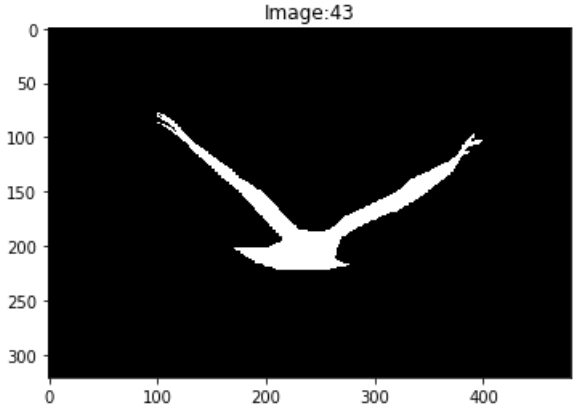


Image:43
0
50
100
150
200
250
300
0
100
200
300
400

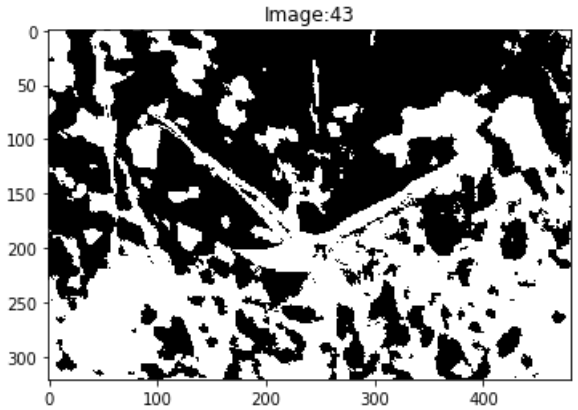


Image:43
0
50
100
150
200
250
300
0
100
200
300
400

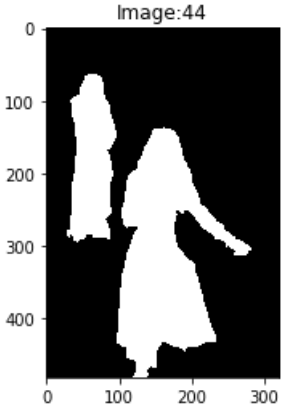


Image:44
0
100
200
300
400
0
100
200
300

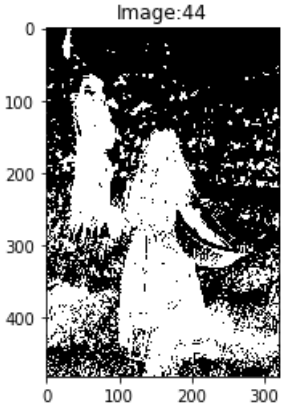


Image:44
0
100
200
300
400
0
100
200
300

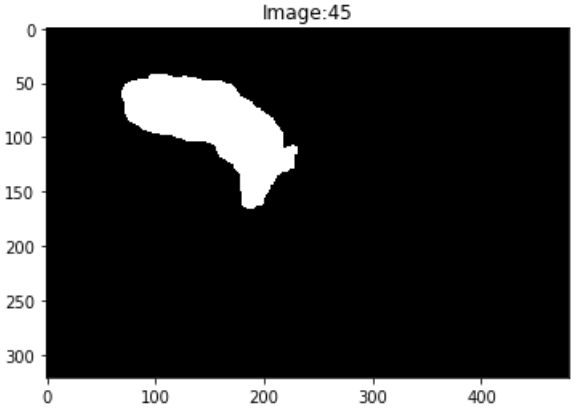


Image:45
0
50
100
150
200
250
300
0
100
200
300
400

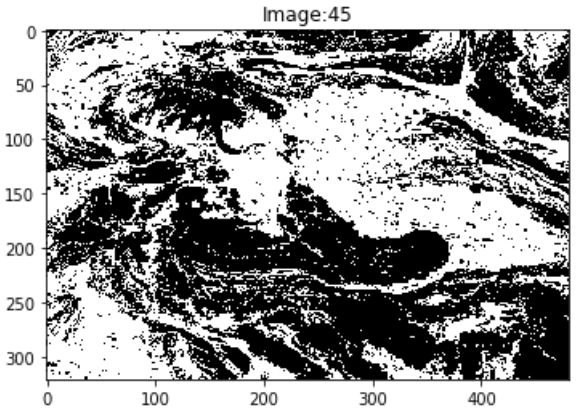


Image:45
0
50
100
150
200
250
300
0
100
200
300
400

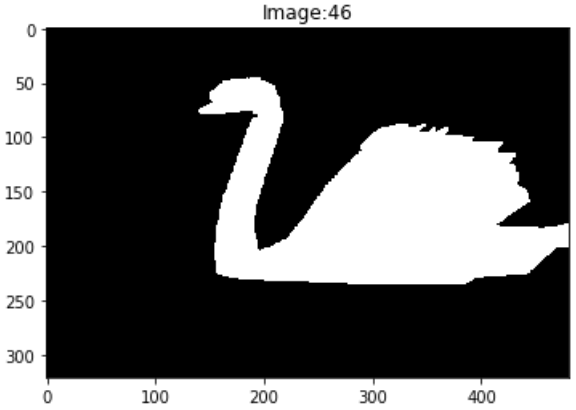


Image:46
0
50
100
150
200
250
300
0
100
200
300
400

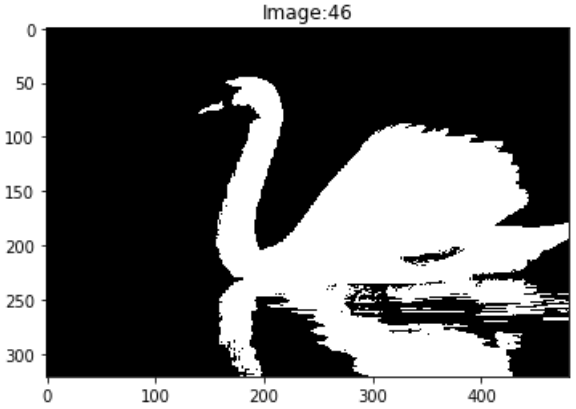


Image:46
0
50
100
150
200
250
300
0
100
200
300
400

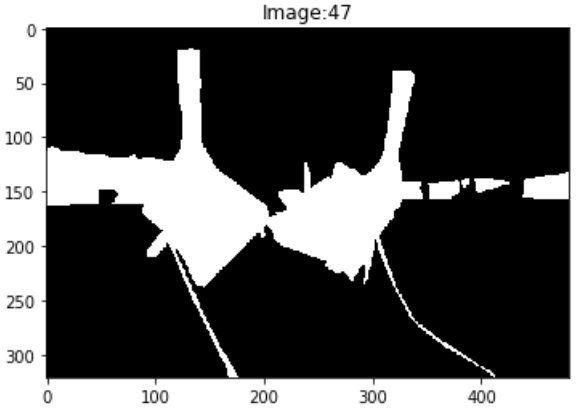


Image:47
0
50
100
150
200
250
300
0
100
200
300
400

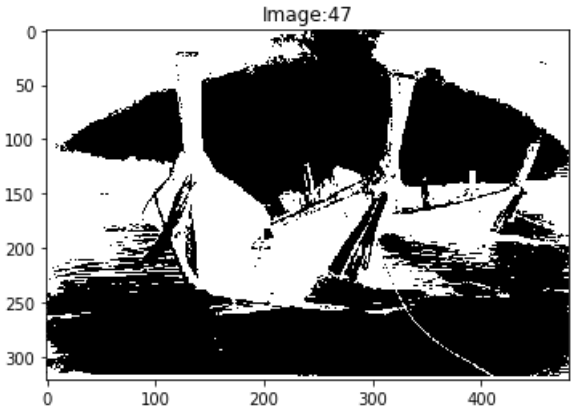
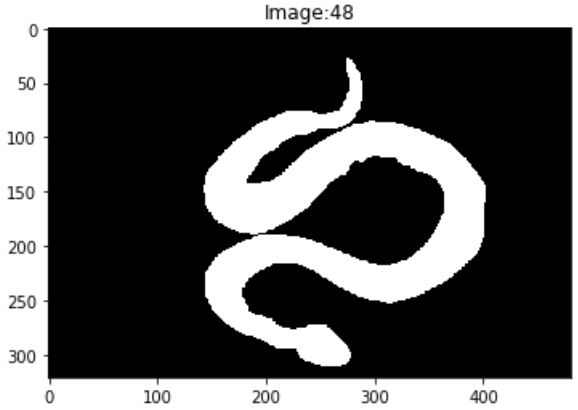
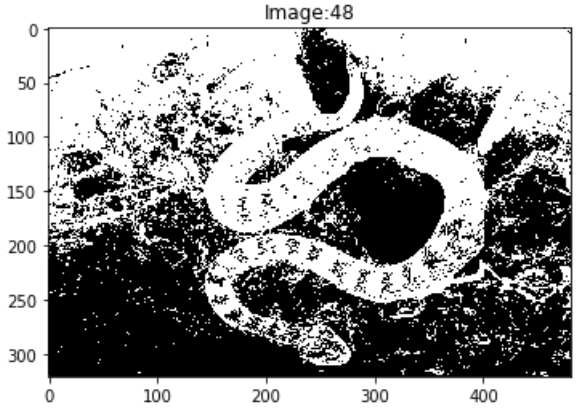
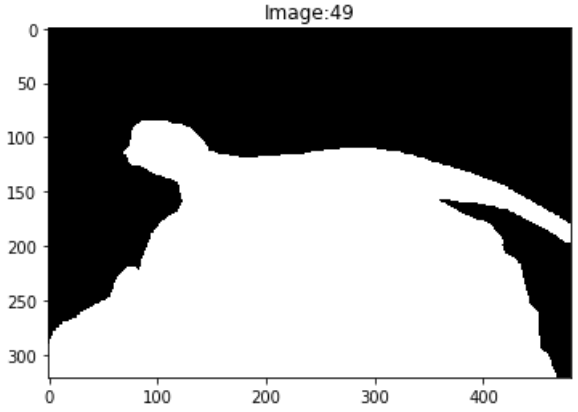
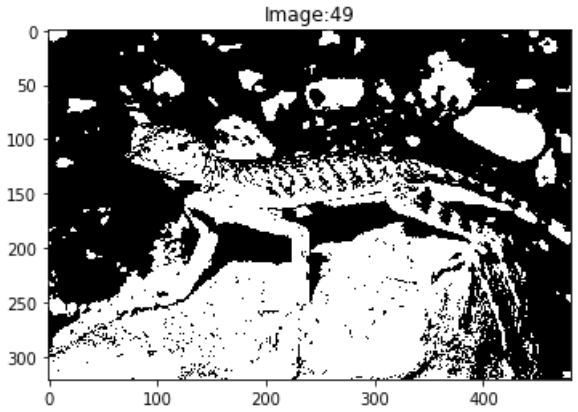


Image:47
0
50
100
150
200
250
300
0
100
200
300
400









In [24]:
```
ls = [1, 4, 7, 8, 9, 11, 18, 19, 21, 26, 27, 28, 30, 31, 32, 33, 35, 39, 40, 46, 49,   ]
for i in ls:
    results[i] = convert(results[i])
```

In [26]:
```
precisions, recalls, accuracies, IOUs, F1_scores = calc_precision_recall(results, ground_truths)
```

for image 0

for image 1

for image 2

for image 3

for image 4

for image 5

for image 6

for image 7

for image 8

for image 9

for image 10

for image 11

for image 12

for image 13

for image 14

for image 15

for image 16

for image 17

for image 18

for image 19

for image 20

for image 21

for image 22

for image 23

for image 24

for image 25

for image 26

for image 27

for image 28

for image 29

for image 30

for image 31

for image 32

for image 33

for image 34

for image 35

for image 36

for image 37

```
for image 38

for image 39

for image 40

for image 41

for image 42

for image 43

for image 44

for image 45

for image 46

for image 47

for image 48

for image 49
```

Saving results in a pandas dataframe

In [27]:
```
df = pd.DataFrame(ID_data, columns = ['Image_ID'])
df.insert(1, "Precision", precisions, True)
df.insert(2, "Recall", recalls, True)
df.insert(3, "F1_score", F1_scores, True)
df.insert(4, "Accuracy", accuracies, True)
df.insert(5, "IOU", IOUs, True)
df.head()
```

Out[27]:

	Image_ID	Precision	Recall	F1_score	Accuracy	IOU
0	100098.jpg	0.963809	0.216656	0.353785	21.490793	0.214908
1	101027.jpg	0.752592	0.249243	0.374470	23.036768	0.230368
2	103006.jpg	0.536910	0.167318	0.255129	14.621667	0.146217
3	103029.jpg	0.586290	0.101000	0.172315	9.428048	0.094280
4	104010.jpg	0.521872	0.160709	0.245742	14.008329	0.140083

In [28]:
```
df.to_csv('Kmedoids.csv', index=False)
```