ECE 661 Computer Vision

Homework 6

Yi-Hong Liao | liao163@purdue.edu | PUID 0031850073

1. Logic

Use the Otsu algorithm to segment the foreground from the background of the image with RGB and texture. Implement the contour extraction algorithm using the segmented binary mask.

2. Step

- Otsu's algorithm

  The histogram of the picture is normalized and regarded as a probability distribution p_i for a gray-level i. The image is dichotomized into two classes C_0 and C_1 by a threshold at level k. The probability and mean of two classes are

$$\omega_0 = \sum_{i=1}^{k} p_i$$

$$\omega_1 = \sum_{i=k+1}^{L} p_i$$

$$\mu_0 = \sum_{i=1}^{k} ip_i / \omega_0$$

$$\mu_0 = \sum_{i=k+1}^{L} ip_i / \omega_1$$

  The goal of the Otsu's algorithm is to find the level k that maximizes the between-class variance,

$$\sigma_B^2 = \omega_0 \omega_1 (\mu_1 - \mu_0)^2$$

  in each iteration.

- RGB-based segmentation

  Apply Otsu's algorithm on each channel of the images and combine the results using the logical operator AND. I used the foreground segmentation returned by each iteration in the Otsu's algorithm to refine the threshold k. on the foreground portion of the image.

- Texture-based segmentation

  I calculated the variance of each pixel in a $N \times N$ window to create a texture-based features. For border pixels, I use the symmetry option. Repeat the feature extraction with different values of N and use them as the channels in the RGB-based segmentation.

- Contour extraction
  I dilate the binary mask and then erode it to connect the missing edge. Then the pixels with value 1 and have at least one pixel with value 0 in the 8 neighbors are consider contours.

3. Result

- **Theory question**
  The Otsu algorithm works well with images that have strong differences in pixel values but small difference in area size between foreground and background. Also, it is simple for calculation. However, it is noisy because it finds only one threshold for the entire image. The Watershed algorithm can be computed in parallel, and it always produce complete boundaries. However, the seed points have to be well chosen and it suffers from over-segmentation in areas with many local minimum points.

- **Input images**

Figure 1: Cat

Figure 2: Car

Figure 3: Dog (foreground: Dog)

Figure 4: F16 (foreground: F16)

- **RGB-based segmentation**

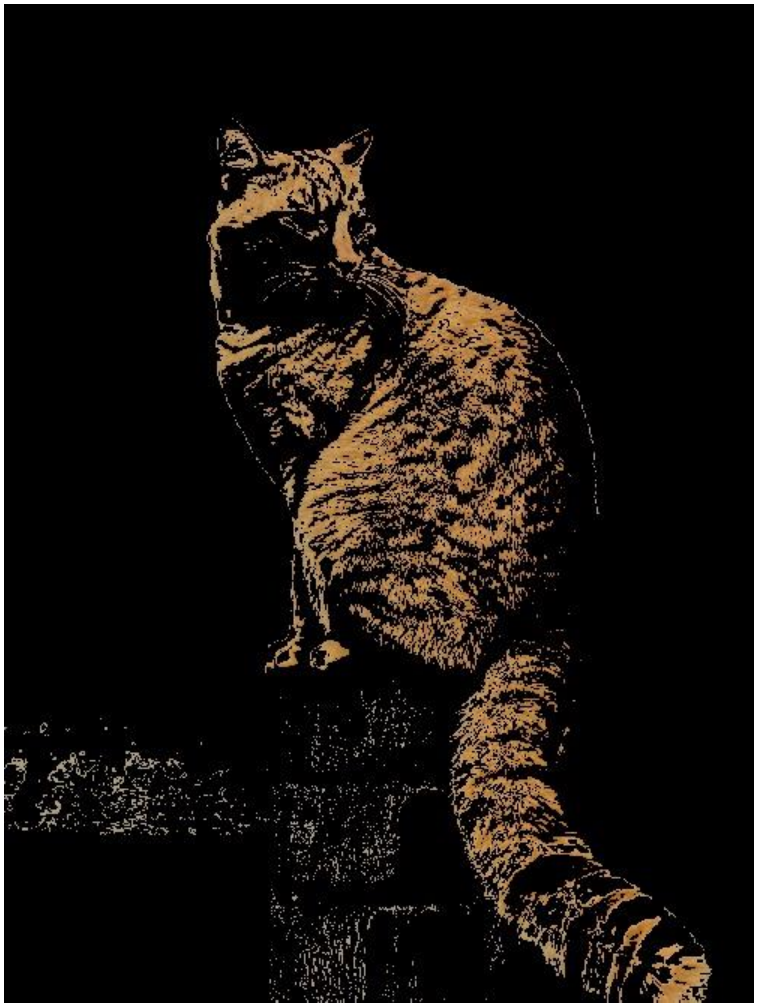

Figure 5: Cat RGB segmentation

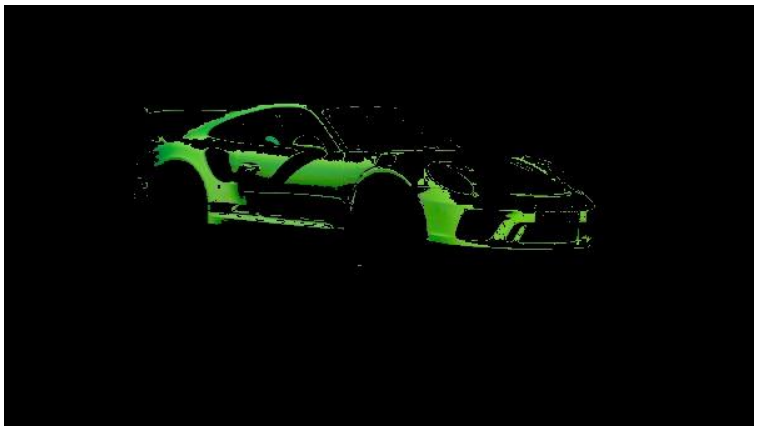

Figure 6: Car RGB segmentation

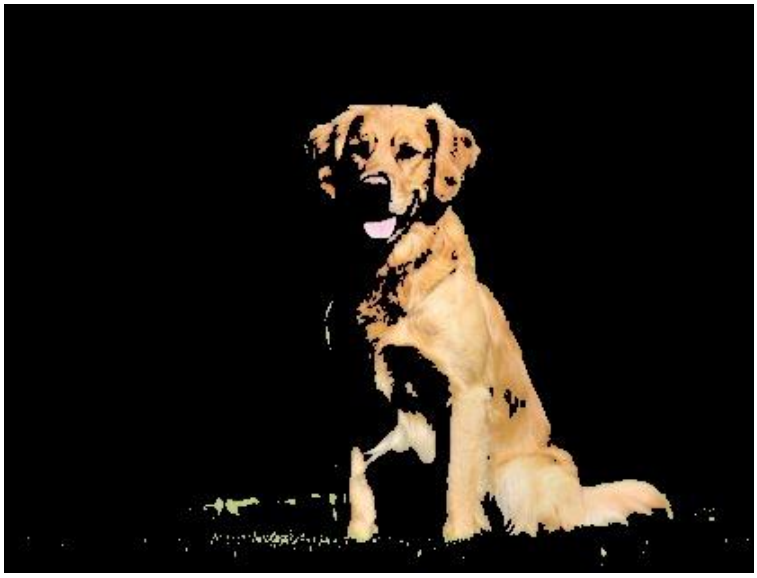

Figure 7: Dog RGB segmentation

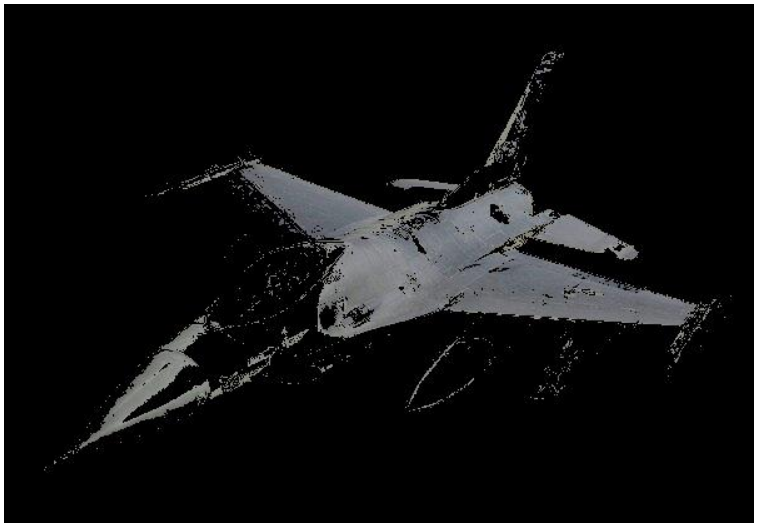

Figure 8: F16 RGB segmentation

The segmentation works well on the F16 image but not on the Dog image.

- **RGB-based contour**

Figure 9: Cat RGB contour

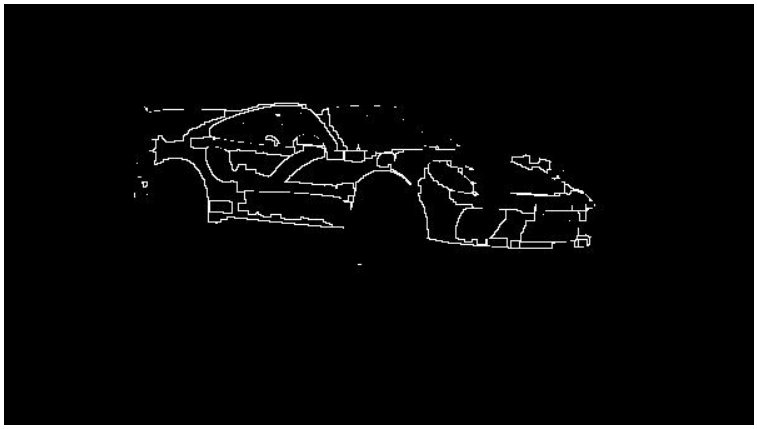

Figure 10: Car RGB contour

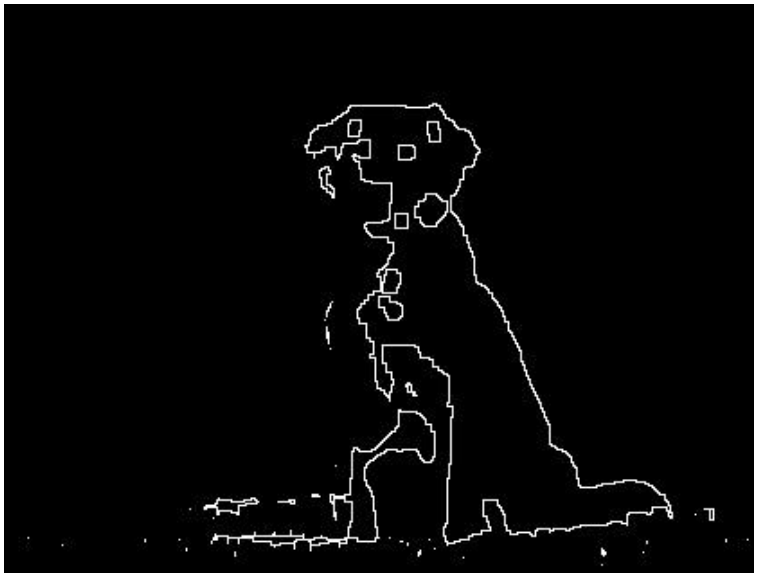

Figure 11: Dog RGB contour

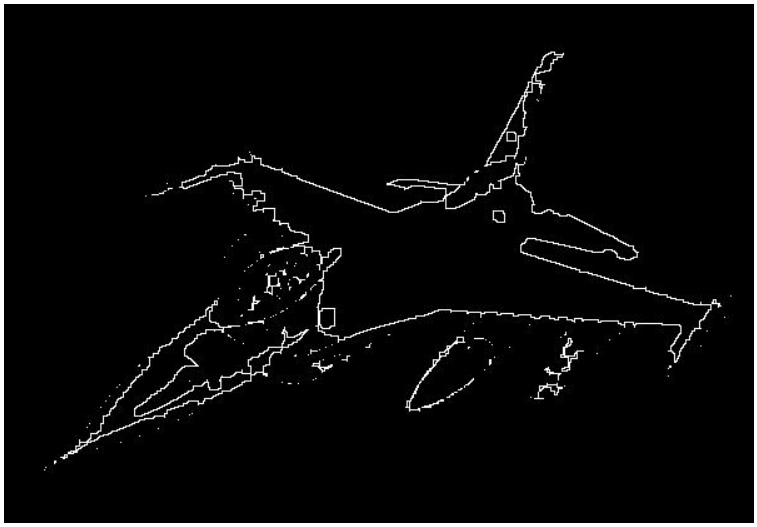

Figure 12: F16 RGB contour

The contour extraction works well on the F16 image but not on the Dog image.

- **Texture-based segmentation**

Figure 13: Cat texture segmentation

Figure 14: Car texture segmentation

Figure 15: Dog texture segmentation

Figure 16: F16 texture segmentation

The segmentation works well on the F16 image but not on the Dog image.

- **Texture-based contour**

Figure 17: Cat texture contour

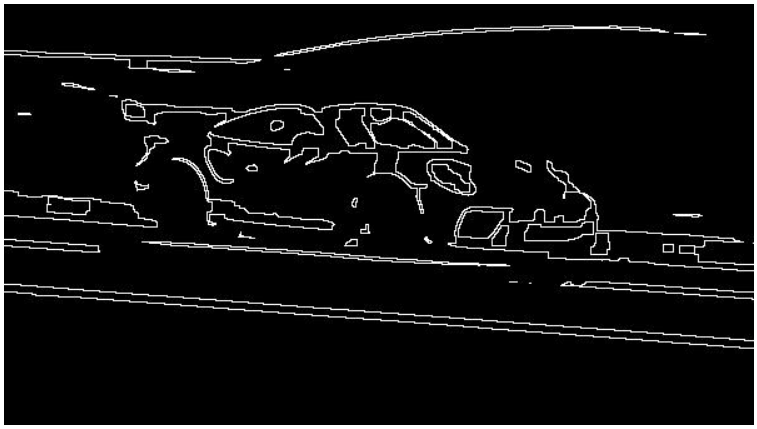

Figure 18: Car texture contour

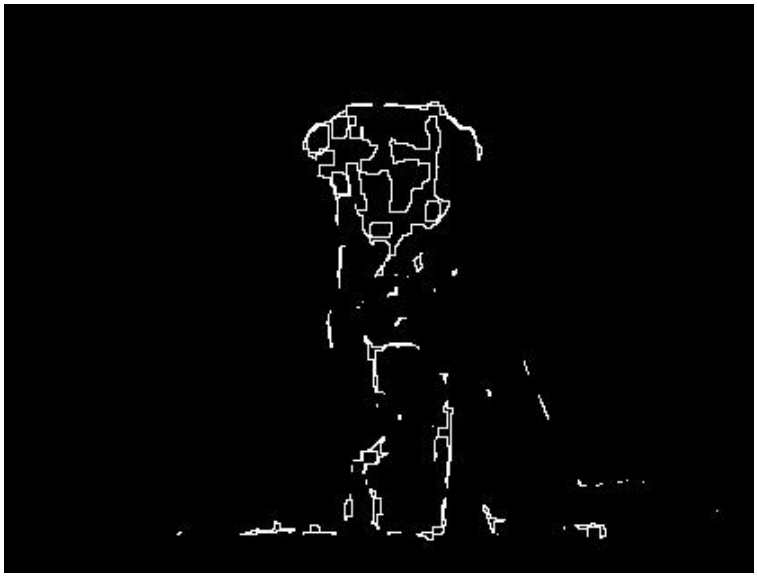

Figure 19: Dog texture contour

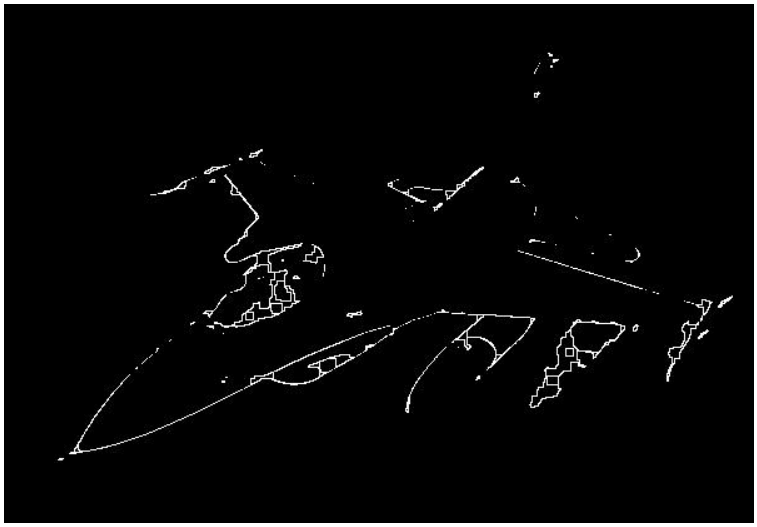

Figure 20: F16 texture contour

The contour extraction works well on the F16 image but not on the Dog image.

- **Observation**

  The RGB segmentation works well when the foreground has a monotone color and its different from the background. If the foreground has many different colors, the RGB method has difficulty segmenting the entire foreground. On the other hand, the texture feature segmentation works well when the foreground has lots of texture features inside or on the edges.

- **Parameters**

RGB Segmentation				
	Cat	Car	Dog	F16
B Iteration	1	1	1	1
G Iteration	1	1	0	1
R Iteration	1	1	2	1
B Inverted	1	1	0	0
G Inverted	1	0	1	1
R Inverted	0	1	0	1

RGB Contour				
	Cat	Car	Dog	F16
Kernel Size	5	5	5	5
Dilation Itr	1	1	1	1
Erosion Itr	1	1	1	1

Texture Segmentation				
	Cat	Car	Dog	F16
Ns	6, 8, 10	6, 8, 10	3, 4, 5	2, 3, 4
Iteration 1	1	1	1	1
Iteration 2	1	1	1	1
Iteration 3	1	1	1	1
Inverted 1	0	0	0	0
Inverted 2	0	0	0	0
Inverted 3	0	0	0	0

Texture Contour				
	Cat	Car	Dog	F16
Kernel Size	5	5	5	5
Dilation Itr	1	1	1	1
Erosion Itr	1	1	1	1

4. Source Code

```
#!/usr/bin/env python
# coding: utf-8

# In[187]:


import numpy as np
import matplotlib.pyplot as plt
import cv2
import math
from scipy import signal


# In[188]:


def otsu(img, num_itr, invert):
    mask = np.ones(img.shape, dtype=np.uint8)

    for i in range(num_itr):
        foreground = img[mask!=0]
        hist, bins = np.histogram(foreground, bins =
np.unique(foreground), density=True)
        bins = bins[:-1]

        sigmaB2 = np.zeros(len(bins))
        i = 0
        for k in bins:
            omega0 = sum(hist[bins < k])
            omega1 = sum(hist[bins >= k])
            mu0 = bins[bins < k].dot(hist[bins < k]/omega0)
            mu1 = bins[bins >= k].dot(hist[bins >= k]/omega1)
            sigmaB2[i] = omega0 * omega1 * (mu1 - mu0)**2
            i = i + 1
        k_star = bins[np.argmax(sigmaB2)]

        if invert == 0:
            mask = img > k_star
        else:
            mask = img < k_star

    return mask

def bgr_segmentation(img, iterations, inverts):
    mask = np.ones(img[:, :, 0].shape, dtype=np.uint8)
    for i in range(3):
#         i = 2
        mask_tmp = otsu(img[:, :, i], iterations[i], inverts[i])
        mask = mask*mask_tmp
        print(i)
    return mask

def texture_segmentation(img, iterations, inverts, Ns):
    img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

```
    mask = np.ones(img_gray.shape, dtype=np.uint8)
    for i, N in enumerate(Ns):
        img_tmp = get_feature_map(img_gray, N)
        mask_tmp = otsu(img_tmp, iterations[i], inverts[i])
        mask = mask*mask_tmp
        print(i)
    return mask

def get_feature_map(img, N):
    img = img.astype(float)
    sum_kernel = np.ones((N, N))
    window_sum = signal.convolve2d(img, sum_kernel, boundary='symm',
mode='same')
    window2_sum = signal.convolve2d(img**2, sum_kernel, boundary='symm',
mode='same')
    mean = window_sum/(N**2)
    mean2 = window2_sum/(N**2)
    var = mean2 - mean**2
    return var.astype(int)

def find_contour(img, k_size, dil_itr, ero_itr):

    kernel = np.ones((k_size, k_size), np.uint8)
    img_dilation = cv2.dilate(img, kernel, iterations=dil_itr)
    img_erosion = cv2.erode(img_dilation, kernel, iterations=ero_itr)

    sum_kernel = np.ones((3, 3))
    window_sum = signal.convolve2d(img_erosion, sum_kernel,
boundary='symm', mode='same')
    sum_mask = window_sum < 9

    return sum_mask*img_erosion




# In[189]:


if __name__ == '__main__':
    path = r'C:\Users\yhosc\Desktop\ECE661\HW6\HW6-Images\\'
    outputPath = r'C:\Users\yhosc\Desktop\ECE661\HW6\\'
    img_cat = cv2.imread(path+'cat.jpg')

    mask_bgr = bgr_segmentation(img_cat, [1, 1, 1], [1, 1, 0])
    img_cat_bgr = cv2.bitwise_and(img_cat, img_cat, mask = mask_bgr)

#     cv2.imwrite(outputPath+'testB.jpg', img_cat[:, :, 0])
#     cv2.imwrite(outputPath+'testG.jpg', img_cat[:, :, 1])
#     cv2.imwrite(outputPath+'testR.jpg', img_cat[:, :, 2])

    mask_texture = texture_segmentation(img_cat, [1, 1, 1], [0, 0, 0], [6,
8, 10])
    img_cat_texture = cv2.bitwise_and(img_cat, img_cat, mask =
mask_texture)
```

```
    img_cat_contour_bgr = find_contour(mask_bgr, 5, 1, 1)
    img_cat_contour_texture = find_contour(mask_texture, 5, 1, 1)

    cv2.imwrite(outputPath+'img_cat_bgr.jpg', img_cat_bgr)
    cv2.imwrite(outputPath+'img_cat_texture.jpg', img_cat_texture)
    cv2.imwrite(outputPath+'img_cat_contour_bgr.jpg',
img_cat_contour_bgr*255)
    cv2.imwrite(outputPath+'img_cat_contour_texture.jpg',
img_cat_contour_texture*255)


# In[190]:


img_car = cv2.imread(path+'car.jpg')

mask_bgr = bgr_segmentation(img_car, [1, 1, 1], [1, 0, 1])
img_car_bgr = cv2.bitwise_and(img_car, img_car, mask = mask_bgr)

#     cv2.imwrite(outputPath+'testB.jpg', img_car[:, :, 0])
#     cv2.imwrite(outputPath+'testG.jpg', img_car[:, :, 1])
#     cv2.imwrite(outputPath+'testR.jpg', img_car[:, :, 2])

mask_texture = texture_segmentation(img_car, [1, 1, 1], [0, 0, 0], [6, 8,
10])
img_car_texture = cv2.bitwise_and(img_car, img_car, mask = mask_texture)

img_car_contour_bgr = find_contour(mask_bgr, 5, 1, 1)
img_car_contour_texture = find_contour(mask_texture, 5, 1, 1)

cv2.imwrite(outputPath+'img_car_bgr.jpg', img_car_bgr)
cv2.imwrite(outputPath+'img_car_texture.jpg', img_car_texture)
cv2.imwrite(outputPath+'img_car_contour_bgr.jpg', img_car_contour_bgr*255)
cv2.imwrite(outputPath+'img_car_contour_texture.jpg',
img_car_contour_texture*255)


# In[191]:


img_f16 = cv2.imread(path+'f16.jpg')

mask_bgr = bgr_segmentation(img_f16, [1, 1, 1], [0, 1, 1])
img_f16_bgr = cv2.bitwise_and(img_f16, img_f16, mask = mask_bgr)

#     cv2.imwrite(outputPath+'testB.jpg', img_f16[:, :, 0])
#     cv2.imwrite(outputPath+'testG.jpg', img_f16[:, :, 1])
#     cv2.imwrite(outputPath+'testR.jpg', img_f16[:, :, 2])

mask_texture = texture_segmentation(img_f16, [1, 1, 1], [0, 0, 0], [2, 3,
4])
img_f16_texture = cv2.bitwise_and(img_f16, img_f16, mask = mask_texture)

img_f16_contour_bgr = find_contour(mask_bgr, 5, 1, 1)
img_f16_contour_texture = find_contour(mask_texture, 5, 1, 1)

cv2.imwrite(outputPath+'img_f16_bgr.jpg', img_f16_bgr)
```

```
cv2.imwrite(outputPath+'img_f16_texture.jpg', img_f16_texture)
cv2.imwrite(outputPath+'img_f16_contour_bgr.jpg', img_f16_contour_bgr*255)
cv2.imwrite(outputPath+'img_f16_contour_texture.jpg',
img_f16_contour_texture*255)


# In[192]:


img_dog = cv2.imread(path+'dog.jpg')

mask_bgr = bgr_segmentation(img_dog, [1, 0, 2], [0, 1, 0])
img_dog_bgr = cv2.bitwise_and(img_dog, img_dog, mask = mask_bgr)

#     cv2.imwrite(outputPath+'testB.jpg', img_dog[:, :, 0])
#     cv2.imwrite(outputPath+'testG.jpg', img_dog[:, :, 1])
#     cv2.imwrite(outputPath+'testR.jpg', img_dog[:, :, 2])

mask_texture = texture_segmentation(img_dog, [1, 1, 1], [0, 0, 0], [3, 4,
5])
img_dog_texture = cv2.bitwise_and(img_dog, img_dog, mask = mask_texture)

img_dog_contour_bgr = find_contour(mask_bgr, 5, 1, 1)
img_dog_contour_texture = find_contour(mask_texture, 5, 1, 1)

cv2.imwrite(outputPath+'img_dog_bgr.jpg', img_dog_bgr)
cv2.imwrite(outputPath+'img_dog_texture.jpg', img_dog_texture)
cv2.imwrite(outputPath+'img_dog_contour_bgr.jpg', img_dog_contour_bgr*255)
cv2.imwrite(outputPath+'img_dog_contour_texture.jpg',
img_dog_contour_texture*255)
```