HƯỚNG DẪN THỰC HÀNH XỬ LÝ ẢNH CONTOURS

Nguyễn Hải Triều

Khoa CNTT-Trường ĐH Nha Trang

trieunh@ntu.edu.vn

Ngày 25 tháng 11 năm 2023

Mục lục

1 Tìm và vẽ contour xuất hiện trong ảnh **2**

2 Contour Features **4**

2.1 Moments 4

2.2 Contour Area 6

2.3 Straight Bounding Rectangle 9

CONTOURS

A contour **is a curve of points, with no gaps in the curve**.

- OpenCV provides methods to find "curves" in an image, called contours.
- Contours are extremely useful for such things as shape approximation and analysis.

In order to find contours in an image:

1. you need to first obtain a binarization of the image
2. using either edge detection methods or thresholding, for example: using Canny edge detector **to find the outlines of the objects** and then **find the actual contours** of the objects

1 Tìm và vẽ contour xuất hiện trong ảnh

Tìm contours trong OpenCV

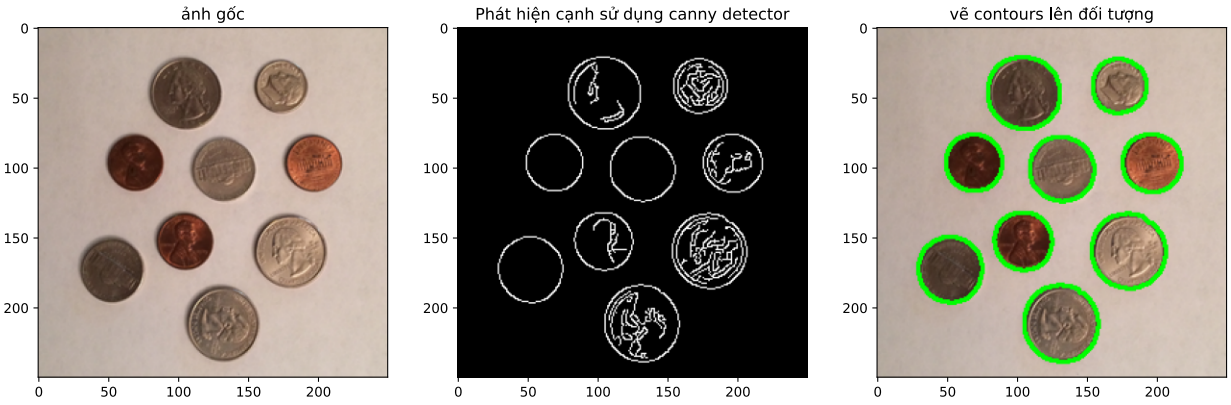
Contours can be explained simply as a curve joining all the continuous points (along the boundary), having same color or intensity. The contours **are a useful tool for shape analysis and object detection and recognition**.

- For better accuracy, use binary images. So before finding contours, apply threshold or canny edge detection.
- Since OpenCV 3.2, findContours() no longer modifies the source image.
- In OpenCV, finding contours is like finding white object from black background. So remember, **object to be found should be white** and **background should be black**.

Code minh hoạ tìm contours cho ảnh *coins.png*

```
import numpy as np
import matplotlib.pyplot as plt
import cv2

def canny_edge_detection(img, threshold1, threshold2):
    '''
    NOTE: phát hiện cạnh bằng canny_edge.
    '''
    # slightly to remove high frequency edges that we aren't interested in
    image = cv2.GaussianBlur(img, (5, 5), 0)
    canny = cv2.Canny(image, threshold1, threshold2)
    return canny

if __name__=='__main__':
    orignal_img=cv2.imread('./images/coins.png')
    assert orignal_img is not None, "file could not be read, check with
        os.path.exists()"
    # chuyển ảnh về RGB
    orignal_img=cv2.cvtColor(orignal_img,cv2.COLOR_BGR2RGB)
    # chuyển ảnh về ảnh xám
    img=cv2.cvtColor(orignal_img,cv2.COLOR_BGR2GRAY)

    # We obtain the edged image by applying the Canny edge detector
    edged = canny_edge_detection(img, 30, 150)
    # find contours of a binary image:
    contours, hierarchy = cv2.findContours(edged.copy(), cv2.RETR_EXTERNAL,
        cv2.CHAIN_APPROX_SIMPLE)
    print(f"số lượng đối tượng có trong ảnh là {len(contours)}")
    # vẽ contour lên đối tượng trong một ảnh mới
    coins=orignal_img.copy()
    cv2.drawContours(coins,contours, -1, (0,255,0), 2)

    # vẽ kết quả bằng matplotlib
    fig=plt.figure(figsize=(16,9))
```

```
    ax1,ax2, ax3=fig.subplots(1,3)
    ax1.imshow(orignal_img)
    ax1.set_title("ảnh gốc")

    ax2.imshow(edged, cmap='gray')
    ax2.set_title("Phát hiện cạnh sử dụng canny detector")

    ax3.imshow(coins)
    ax3.set_title("vẽ contours lên đối tượng")

    plt.savefig("contours.pdf",bbox_inches='tight')
    plt.show()
```



Hình 1: Ví dụ tìm và vẽ contours cho các đồng tiền trong ảnh *coins.png*.

Giải thích hàm tìm Contour

There are three arguments in cv2.findContours() function,

- first one is **source image**: ảnh cần tìm contours
- second is **contour retrieval mode**: **cv2.RETR_EXTERNAL** (chỉ lấy đường viền ngoài cùng), **cv2.RETR_TREE** (lấy tất cả contours xuất hiện trong hình)
- third is **contour approximation method**: **cv2.CHAIN_APPROX_SIMPLE** (chỉ lưu những điểm bắt đầu và kết thúc các line của các contours, nên tiết kiện được bộ nhớ lưu trữ), **cv2.CHAIN_APPROX_NONE** (lưu tất các các điểm–toạ độ (x,y) của contours)

And it outputs:

- the contours: contours is a Python tuple of all the contours in the image. **Each individual contour is a Numpy array of (x,y) coordinates of boundary points of the object**.
- hierarchy

Giải thích hàm vẽ Contour

To draw the contours, cv.drawContours function is used. It can also be used to draw any shape provided you have its boundary points. , second argument is the contours which should be passed as a Python list, third argument is index of contours (useful when drawing individual contour. To draw all contours, pass -1) and remaining arguments are color, thickness etc.

- Its first argument is **source image**
- second argument is **the contours** which should be passed as a Python tuple
- third argument is **index of contours** (useful when drawing individual contour. **To draw all contours, pass -1**)
- fourth arguments are color
- remaining arguments **thickness** etc.

Có thể vẽ các contours bằng thư viện matplotlib với phương thức xấp xỉ contour là cv2.CHAIN_APPROX_NONE

```
...
contours, hierarchy = cv2.findContours(edged, cv2.RETR_EXTERNAL,
    cv2.CHAIN_APPROX_NONE)
    # cv2.CHAIN_APPROX_NONE: lưu tất cả các điểm
    print(f"số lượng đối tượng có trong ảnh là {len(contours)}")
    fig=plt.figure(figsize=(4,4))
    for i in range(len(contours)):
        x=contours[i][:,0,0]
        y=contours[i][:,0,1]
        plt.plot(x,y)
    plt.show()
...
```

2 Contour Features

In this section, we will learn

- To find the **different features of contours, like area, perimeter, centroid, bounding** box etc.
- and plenty of functions related to contours.

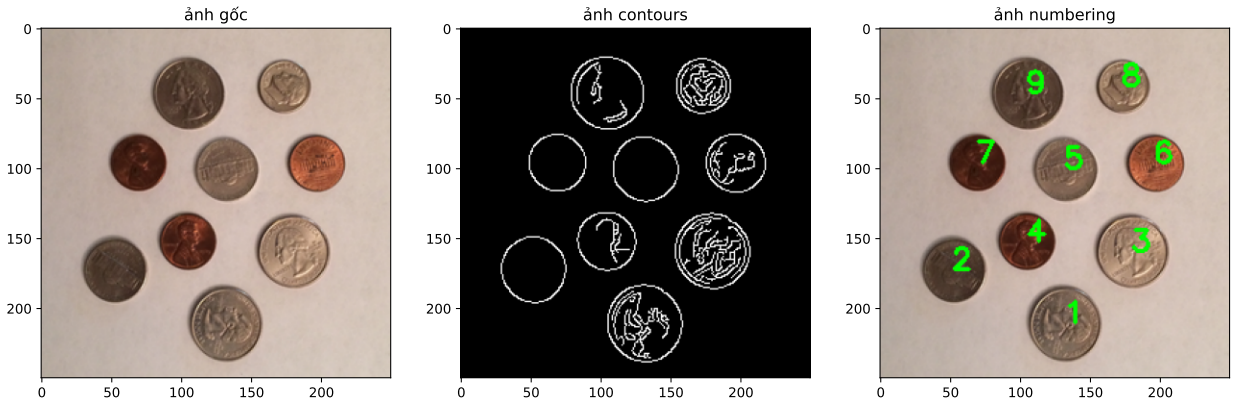
Image moments help you to calculate some features like center of mass of the object, area of the object etc.

2.1 Moments

Code minh hoạ sử dụng **cv2.moments(contours[i])** để **tìm toạ độ tâm khối của các vật thể**. Sau đó **đánh số thứ tự các vật thể lên hình** bằng hàm **cv2.putText()**.

```python
import numpy as np
import matplotlib.pyplot as plt
import cv2

def canny_edge_detection(img, threshold1, threshold2):
    '''
    NOTE: phát hiện cạnh bằng canny_edge.
    '''
    # slightly to remove high frequency edges that we aren't interested in
    image = cv2.GaussianBlur(img, (5, 5), 0)
    canny = cv2.Canny(image, threshold1, threshold2)
    return canny
def centroid_objects(contours):
    '''Hàm tìm tâm khối của các đối tượng sử dụng OpenCV'''
    number_of_contour=len(contours)
    list_of_centroid = [[0,0] for _ in range(number_of_contour)] # khởi tạo
        list lưu toạn độtâm khối
    for i in range(number_of_contour):
        M=cv2.moments(contours[i])
        cx = int(M['m10']/M['m00']) #công thức tính tâm khối toạ độx
        cy = int(M['m01']/M['m00']) #công thức tính tâm khối toạ độy
        list_of_centroid[i][0]=cx
        list_of_centroid[i][1]=cy
    return list_of_centroid

def draw_numbering_of_object(image, centroids):
    '''Hàm đánh số thứ tự của các objects tìm được trong hình tại vị trí tâm
        khối
        sử dụng opencv.
    '''
    # khởi tạo các thông số đểđánh số
    font = cv2.FONT_HERSHEY_SIMPLEX
    scale = 0.7
    color = (255, 0, 0) # Blue color in BGR
    for i in range(len(centroids)):
        cv2.putText(image, str(i+1), (centroids[i][0], centroids[i][1]), font,
            scale, color, thickness=2)
    return image


if __name__=='__main__':
    orignal_img=cv2.imread('./images/coins.png')
    assert orignal_img is not None, "file could not be read, check with
        os.path.exists()"
    # chuyển ảnh về RGB
    orignal_img=cv2.cvtColor(orignal_img,cv2.COLOR_BGR2RGB)
    # chuyển ảnh về ảnh xám
    img=cv2.cvtColor(orignal_img,cv2.COLOR_BGR2GRAY)

    # We obtain the edged image by applying the Canny edge detector
    edged = canny_edge_detection(img, 30, 150)
```

```
    # find contours of a binary image:
    contours, hierarchy = cv2.findContours(edged, cv2.RETR_EXTERNAL,
        cv2.CHAIN_APPROX_SIMPLE)
    number_of_contour=len(contours)
    print(f"số lượng đối tượng có trong ảnh là {number_of_contour}")
    # Tìm tâm khối cho tất cả các contours
    centroids= centroid_objects(contours)
    # Đánh số thứ tự lên contour tại tâm khối
    img_numbering= draw_numbering_of_object(orignal_img.copy(),centroids)

    #Vẽ ảnh
    fig=plt.figure(figsize=(16,9))
    ax1,ax2,ax3=fig.subplots(1,3)
    ax1.imshow(orignal_img)
    ax1.set_title("ảnh gốc")

    ax2.imshow(edged, cmap="gray")
    ax2.set_title("ảnh contours")

    ax3.imshow(img_numbering)
    ax3.set_title("ảnh numbering")

    plt.show()
```



Hình 2: Ví dụ đánh số thứ tự cho các contours tại tâm khối của chúng *coins.png*.

2.2 Contour Area

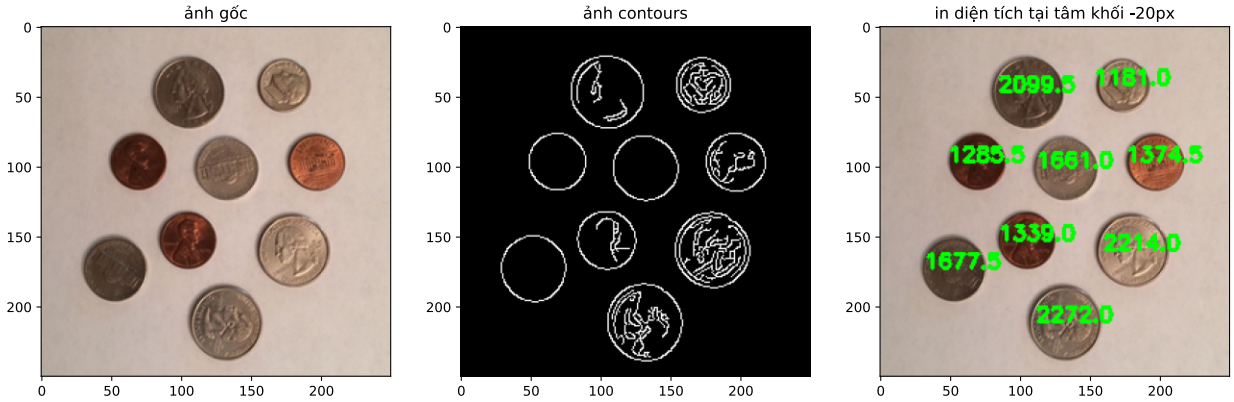
Tính diện tích của đối tượng

Contour area is given by the function **cv.contourArea()** or **from moments, M['m00']**. The area is computed using the Green formula.

```
import numpy as np
import matplotlib.pyplot as plt
import cv2

```

```
5  def canny_edge_detection(img, threshold1, threshold2):
6      '''
7      NOTE: phát hiện cạnh bằng canny_edge.
8      '''
9      # slightly to remove high frequency edges that we aren't interested in
10     image = cv2.GaussianBlur(img, (5, 5), 0)
11     canny = cv2.Canny(image, threshold1, threshold2)
12     return canny
13 def centroid_objects(cnts):
14     '''Hàm tìm tâm khối của các đối tượng sử dụng OpenCV'''
15     number_of_contour=len(cnts)
16     list_of_centroid = [[0,0] for _ in range(number_of_contour)] # khởi tạo
           list lưu toạn độtâm khối
17     for i in range(number_of_contour):
18         M=cv2.moments(cnts[i])
19         cx = int(M['m10']/M['m00'])#công thức tính tâm khối toạ độx
20         cy = int(M['m01']/M['m00'])#công thức tính tâm khối toạ độy
21         list_of_centroid[i][0]=cx
22         list_of_centroid[i][1]=cy
23     return list_of_centroid
24
25 def area_objects(cnts):
26     '''Hàm tính diện tích của tất cả các đối tượng xuất hiện trong hình'''
27     list_of_area=[cv2.contourArea(i) for i in cnts]
28     return list_of_area
29
30
31 def draw_numbering_of_object(image, centroids):
32     '''Hàm đánh số thứ tự của các objects tìm được trong hình tại vị trí tâm
           khối
33         sử dụng opencv.
34     '''
35     # khởi tạo các thông số đểđánh số
36     font = cv2.FONT_HERSHEY_SIMPLEX
37     scale = 0.7
38     color = (0, 255, 0) # Blue color in BGR
39     for i in range(len(centroids)):
40         cv2.putText(image, str(i+1), (centroids[i][0], centroids[i][1]), font,
               scale, color, thickness=2)
41     return image
42
43 def print_area_of_object(image, centroids, areas):
44     '''Hàm in diện tích lên vật thể sử dụng opencv.
45     '''
46     # khởi tạo các thông số đểđánh số
47     font = cv2.FONT_HERSHEY_SIMPLEX
48     scale = 0.5
49     color = (0, 255, 0) # Blue color in BGR
50     for i in range(len(centroids)):
51         cv2.putText(image, str(areas[i]), (centroids[i][0]-20, centroids[i][1]),
               font, scale, color, thickness=2)
52     return image
```

```
53
54 if __name__=='__main__':
55     orignal_img=cv2.imread('./images/coins.png')
56     assert orignal_img is not None, "file could not be read, check with
           os.path.exists()"
57     # chuyển ảnh về RGB
58     orignal_img=cv2.cvtColor(orignal_img,cv2.COLOR_BGR2RGB)
59     # chuyển ảnh về ảnh xám
60     img=cv2.cvtColor(orignal_img,cv2.COLOR_BGR2GRAY)
61
62     # We obtain the edged image by applying the Canny edge detector
63     edged = canny_edge_detection(img, 30, 150)
64     # find contours of a binary image:
65     contours, hierarchy = cv2.findContours(edged, cv2.RETR_EXTERNAL,
           cv2.CHAIN_APPROX_SIMPLE)
66     number_of_contour=len(contours)
67     print(f"số lượng đối tượng có trong ảnh là {number_of_contour}")
68     # Tìm tâm khối cho tất cả các contours
69     centroids= centroid_objects(contours)
70     # Tìm diện tích cho tất cả các đối tượng
71     areas=area_objects(contours)
72     # In diện tích của vật thể tại tâm khối -20px
73     img_printarea=print_area_of_object(orignal_img.copy(),centroids, areas)
74
75     #Vẽ ảnh
76     fig=plt.figure(figsize=(16,9))
77     ax1,ax2,ax3=fig.subplots(1,3)
78     ax1.imshow(orignal_img)
79     ax1.set_title("ảnh gốc")
80
81     ax2.imshow(edged, cmap="gray")
82     ax2.set_title("ảnh contours")
83
84     ax3.imshow(img_printarea)
85     ax3.set_title("in diện tích tại tâm khối -20px",)
86     plt.savefig("area_of_contour.pdf",bbox_inches='tight')
87     plt.show()
```



Hình 3: Ví dụ diện tích của các contours tại tâm khối -20px của chúng *coins.png*.

2.3 Straight Bounding Rectangle

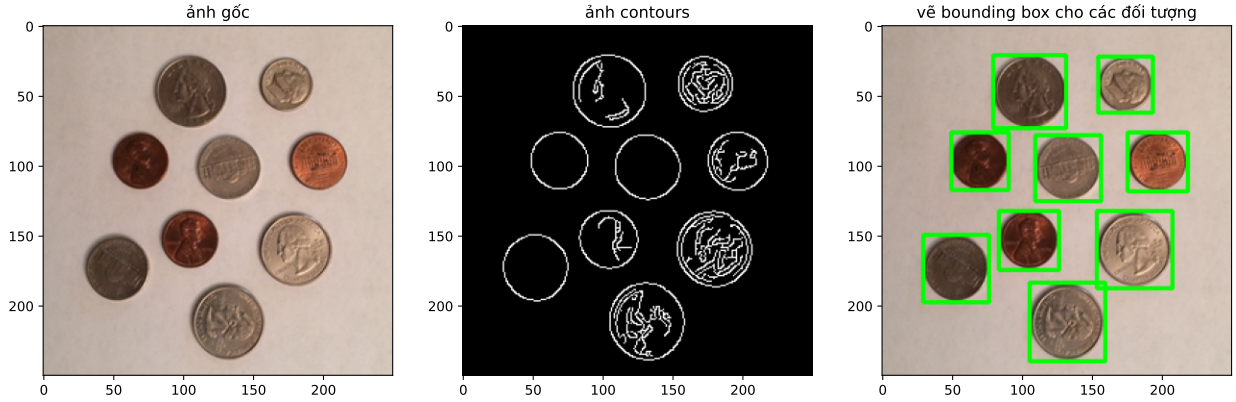
It is a straight rectangle, it doesn't consider the rotation of the object. So area of the bounding rectangle won't be minimum. It is found by the function **cv.boundingRect()**. Let **(x,y) be the top-left coordinate of the rectangle** and **(w,h) be its width and height**.

ví dụ vẽ bounding box hình chữ nhật cho các đối tượng

```
import numpy as np
import matplotlib.pyplot as plt
import cv2

def canny_edge_detection(img, threshold1, threshold2):
    '''
    NOTE: phát hiện cạnh bằng canny_edge.
    '''
    # slightly to remove high frequency edges that we aren't interested in
    image = cv2.GaussianBlur(img, (5, 5), 0)
    canny = cv2.Canny(image, threshold1, threshold2)
    return canny
def centroid_objects(cnts):
    '''Hàm tìm tâm khối của các đối tượng sử dụng OpenCV'''
    number_of_contour=len(cnts)
    list_of_centroid = [[0,0] for _ in range(number_of_contour)] # khởi tạo
        list lưu toạn độtâm khối
    for i in range(number_of_contour):
        M=cv2.moments(cnts[i])
        cx = int(M['m10']/M['m00'])#công thức tính tâm khối toạ độx
        cy = int(M['m01']/M['m00'])#công thức tính tâm khối toạ độy
        list_of_centroid[i][0]=cx
        list_of_centroid[i][1]=cy
    return list_of_centroid

def area_objects(cnts):
    '''Hàm tính diện tích của tất cả các đối tượng xuất hiện trong hình'''
    list_of_area=[cv2.contourArea(i) for i in cnts]
```

```
28      return list_of_area
29
30
31  def draw_numbering_of_object(image, centroids):
32      '''Hàm đánh số thứ tự của các objects tìm được trong hình tại vị trí tâm
            khối
33          sử dụng opencv.
34      '''
35      # khởi tạo các thông số đểđánh số
36      font = cv2.FONT_HERSHEY_SIMPLEX
37      scale = 0.7
38      color = (0, 255, 0) # Blue color in BGR
39      for i in range(len(centroids)):
40          cv2.putText(image, str(i+1), (centroids[i][0], centroids[i][1]), font,
                scale, color, thickness=2)
41      return image
42
43  def print_area_of_object(image, centroids, areas):
44      '''Hàm in diện tích lên vật thể sử dụng opencv.
45      '''
46      # khởi tạo các thông số đểđánh số
47      font = cv2.FONT_HERSHEY_SIMPLEX
48      scale = 0.5
49      color = (0, 255, 0) # Blue color in BGR
50      for i in range(len(centroids)):
51          cv2.putText(image, str(areas[i]), (centroids[i][0]-20, centroids[i][1]),
                font, scale, color, thickness=2)
52      return image
53  def draw_bounding_box(image,cnts):
54      for i in cnts:
55          x,y,w,h=cv2.boundingRect(i)
56          cv2.rectangle(image,(x,y),(x+w,y+h),(0,255,0),2)
57      return image
58
59  if __name__=='__main__':
60      orignal_img=cv2.imread('./images/coins.png')
61      assert orignal_img is not None, "file could not be read, check with
            os.path.exists()"
62      # chuyển ảnh về RGB
63      orignal_img=cv2.cvtColor(orignal_img,cv2.COLOR_BGR2RGB)
64      # chuyển ảnh về ảnh xám
65      img=cv2.cvtColor(orignal_img,cv2.COLOR_BGR2GRAY)
66
67      # We obtain the edged image by applying the Canny edge detector
68      edged = canny_edge_detection(img, 30, 150)
69      # find contours of a binary image:
70      contours, hierarchy = cv2.findContours(edged, cv2.RETR_EXTERNAL,
            cv2.CHAIN_APPROX_SIMPLE)
71      number_of_contour=len(contours)
72      print(f"số lượng đối tượng có trong ảnh là {number_of_contour}")
73      # Vẽ bounding box cho các đối tượng contours
74      bounding_box=draw_bounding_box(orignal_img.copy(),contours)
```

```
    #Vẽ ảnh
    fig=plt.figure(figsize=(16,9))
    ax1,ax2,ax3=fig.subplots(1,3)
    ax1.imshow(orignal_img)
    ax1.set_title("ảnh gốc")

    ax2.imshow(edged, cmap="gray")
    ax2.set_title("ảnh contours")

    ax3.imshow(bounding_box)
    ax3.set_title("vẽ bounding box cho các đối tượng",)
    plt.savefig("boundingbox_of_contour.pdf",bbox_inches='tight')
    plt.show()
```



Hình 4: Ví dụ vẽ bounding box hình chữ nhật quanh các đối tượng.

Tài liệu

[1] OpenCV. What are contours? (2024), https://docs.opencv.org/4.x/d4/d73/tutorial_py_contours_begin.html.

[2] Rafael C. Gonzalez, Richard E. Woods. Digital Image Processing (2018), Fourth Edition, Global Edition, Pearson.

[3] Adrian Rosebrock. Practical Python and OpenCV: An Introductory, Example Driven Guide to Image Processing and Computer Vision, Third Edition.