

Nama: Muhamad Raehan

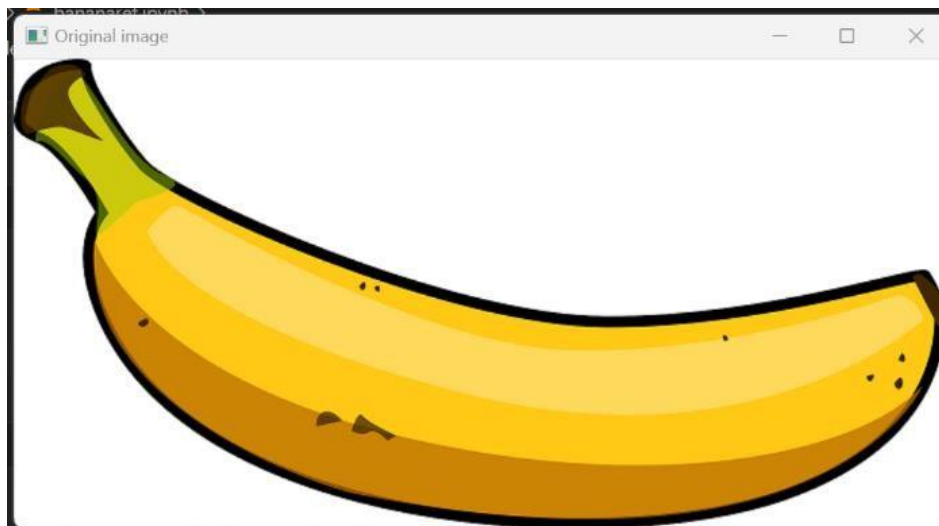
NIM: 312210266

Kelas: TI.22.A.2

---

```
1  # %%
2  import cv2
3
4  # %%
5  image = cv2.imread('data/bananaref.png')
6  imagecopy= image.copy()
7  cv2.imshow( 'Original image' , image )
8  cv2.waitKey(0)
9  cv2.destroyAllWindows()
10
11 # %%
12 gray_image = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
13 cv2.imshow( 'gray' , gray_image )
14 cv2.waitKey(0)
15 cv2.destroyAllWindows()
16
17 # %%
18 ret,binary_im = cv2.threshold(gray_image,245,255,cv2.THRESH_BINARY)
19 cv2.imshow( 'binary' , binary_im )
20 cv2.waitKey(0)
21 cv2.destroyAllWindows()
22
23 # %%
24 binary_im= ~binary_im
25 cv2.imshow( 'inverted binary' , binary_im )
26 cv2.waitKey(0)
27 cv2.destroyAllWindows()
28
29 # %%
30 #find the external contours from binary image
31 contours,hierarchy = cv2.findContours(binary_im,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
32
33 # %%
34 with_contours = cv2.drawContours(image,contours,-1,(0,0,255),3)
35 cv2.imshow( 'contours marked on RGB image' , with_contours )
36 cv2.waitKey(0)
37 cv2.destroyAllWindows()
38
39
```

Output:





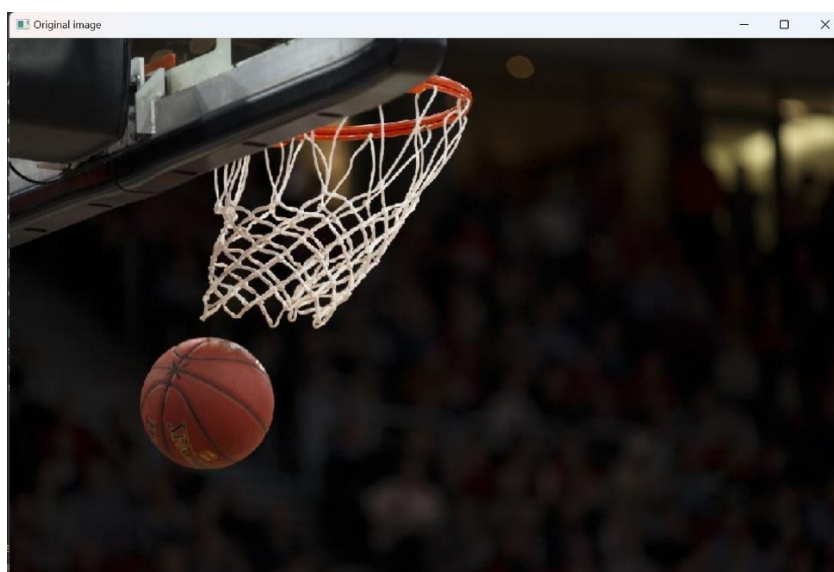
Basketball.ipynb

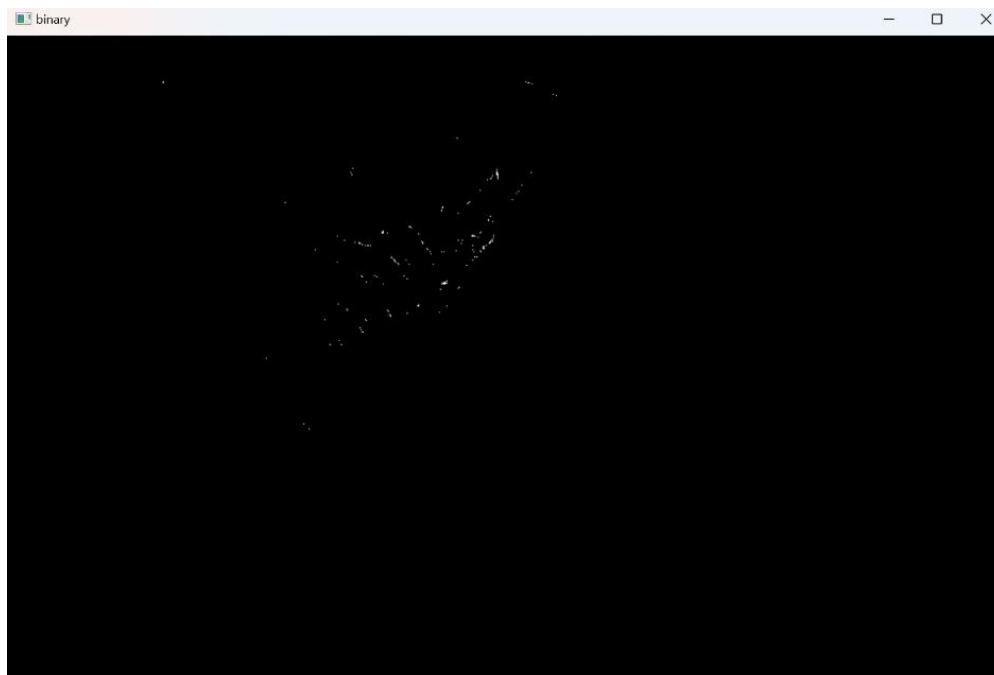
```

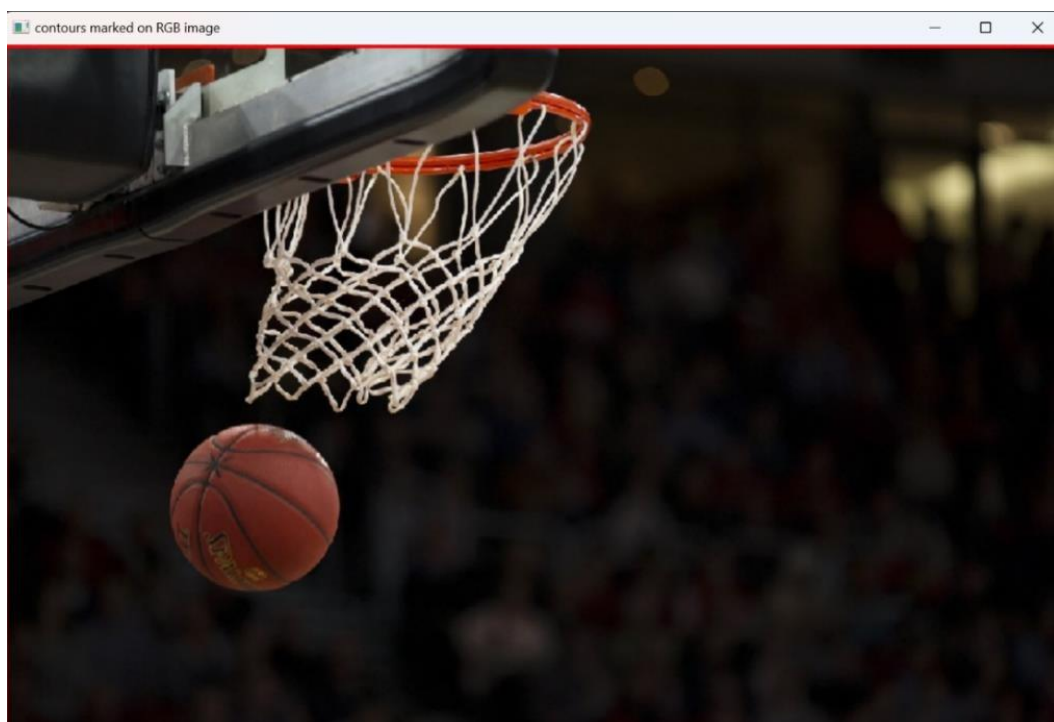
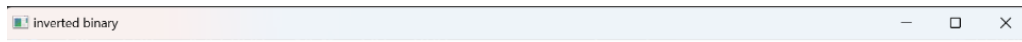
1  # %%
2  import cv2
3
4  # %%
5  image = cv2.imread('data/basketball.jpg')
6  imagecopy= image.copy()
7  cv2.imshow( 'Original image' , image )
8  cv2.waitKey(0)
9  cv2.destroyAllWindows()
10
11 # %%
12 gray_image = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
13 cv2.imshow( 'gray' , gray_image )
14 cv2.waitKey(0)
15 cv2.destroyAllWindows()
16
17 # %%
18 ret,binary_im = cv2.threshold(gray_image,245,255,cv2.THRESH_BINARY)
19 cv2.imshow( 'binary' , binary_im )
20 cv2.waitKey(0)
21 cv2.destroyAllWindows()
22
23 # %%
24 binary_im= ~binary_im
25 cv2.imshow( 'inverted binary' , binary_im )
26 cv2.waitKey(0)
27 cv2.destroyAllWindows()
28
29 # %%
30 #find the external contours from binary image
31 contours,hierarchy = cv2.findContours(binary_im,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
32
33 # %%
34 with_contours = cv2.drawContours(image,contours,-1,(0,0,255),3)
35 cv2.imshow( 'contours marked on RGB image' , with_contours )
36 cv2.waitKey(0)
37 cv2.destroyAllWindows()

```

Output:





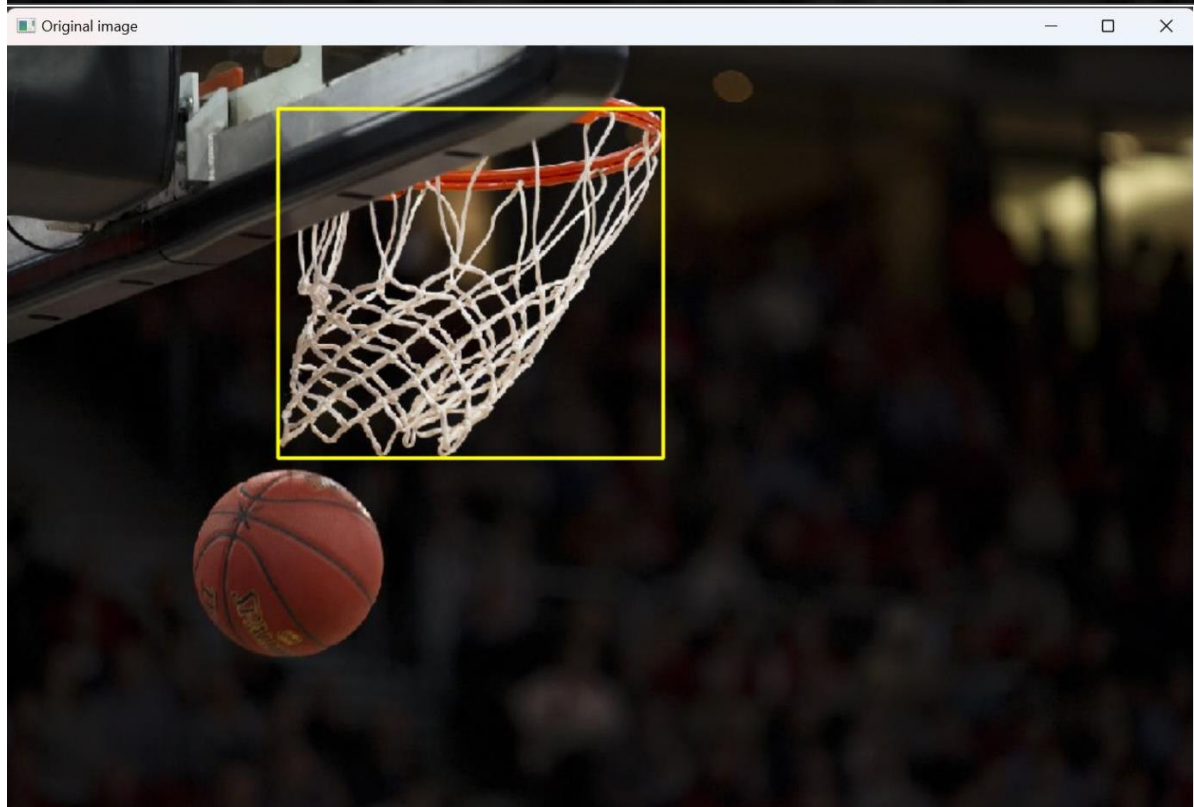
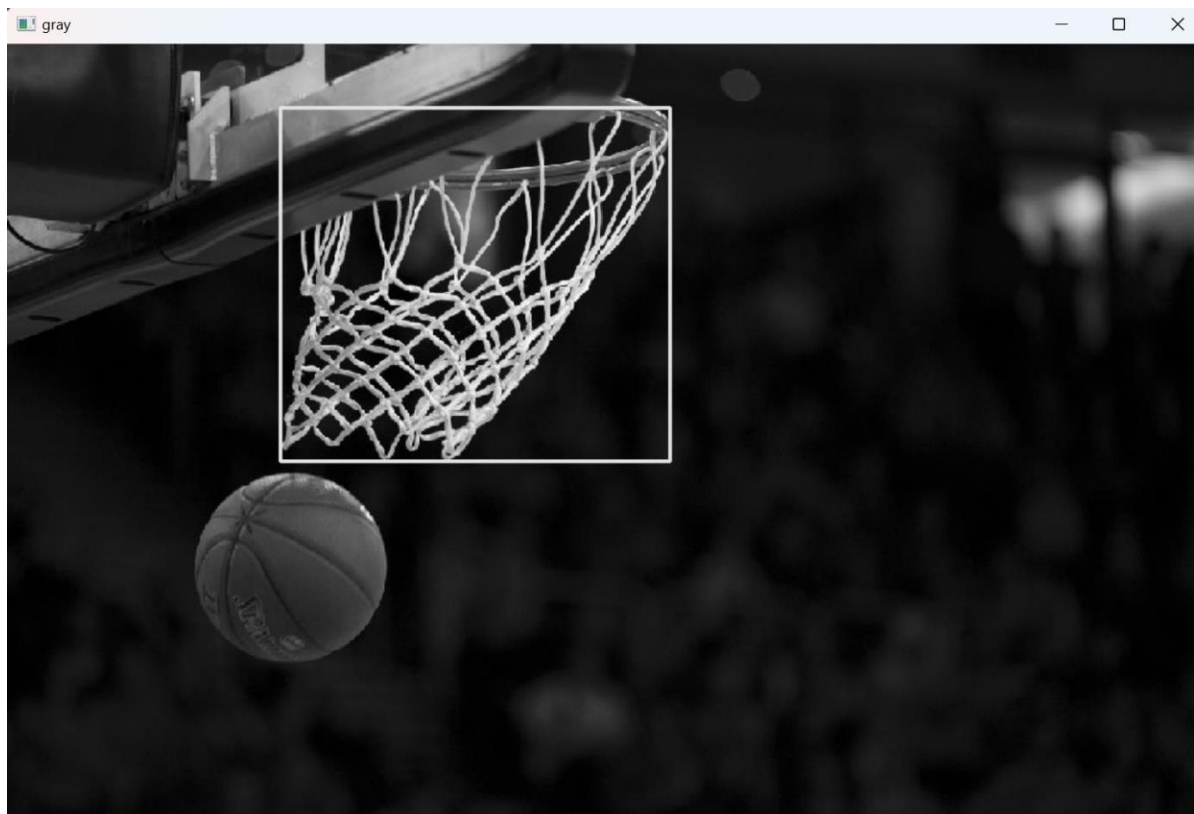


Detected basket.ipynb

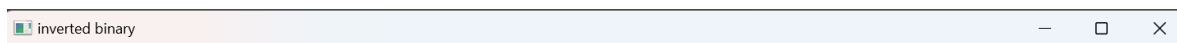
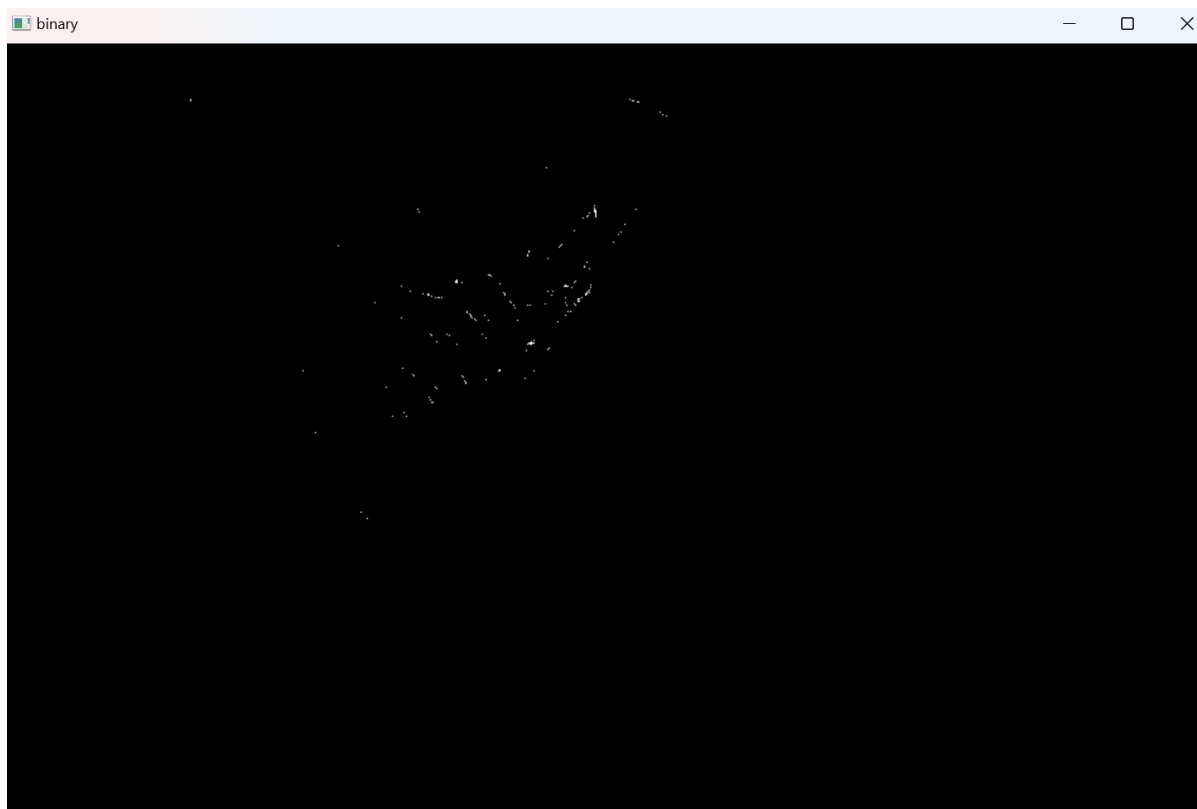
```
1  # %%
2  import cv2
3
4  # %%
5  image = cv2.imread('data/detected_basket.png')
6  imagecopy= image.copy()
7  cv2.imshow( 'Original image' , image )
8  cv2.waitKey(0)
9  cv2.destroyAllWindows()
10
11 # %%
12 gray_image = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
13 cv2.imshow( 'gray' , gray_image )
14 cv2.waitKey(0)
15 cv2.destroyAllWindows()
16
17 # %%
18 ret,binary_im = cv2.threshold(gray_image,245,255,cv2.THRESH_BINARY)
19 cv2.imshow( 'binary' , binary_im )
20 cv2.waitKey(0)
21 cv2.destroyAllWindows()
22
23 # %%
24 binary_im= ~binary_im
25 cv2.imshow( 'inverted binary' , binary_im )
26 cv2.waitKey(0)
27 cv2.destroyAllWindows()
28
29 # %%
30 #find the external contours from binary image
31 contours,hierarchy = cv2.findContours(binary_im,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
32
33 # %%
34 with_contours = cv2.drawContours(image,contours,-1,(0,0,255),3)
35 cv2.imshow( 'contours marked on RGB image' , with_contours )
36 cv2.waitKey(0)
37 cv2.destroyAllWindows()
38
39
40
```

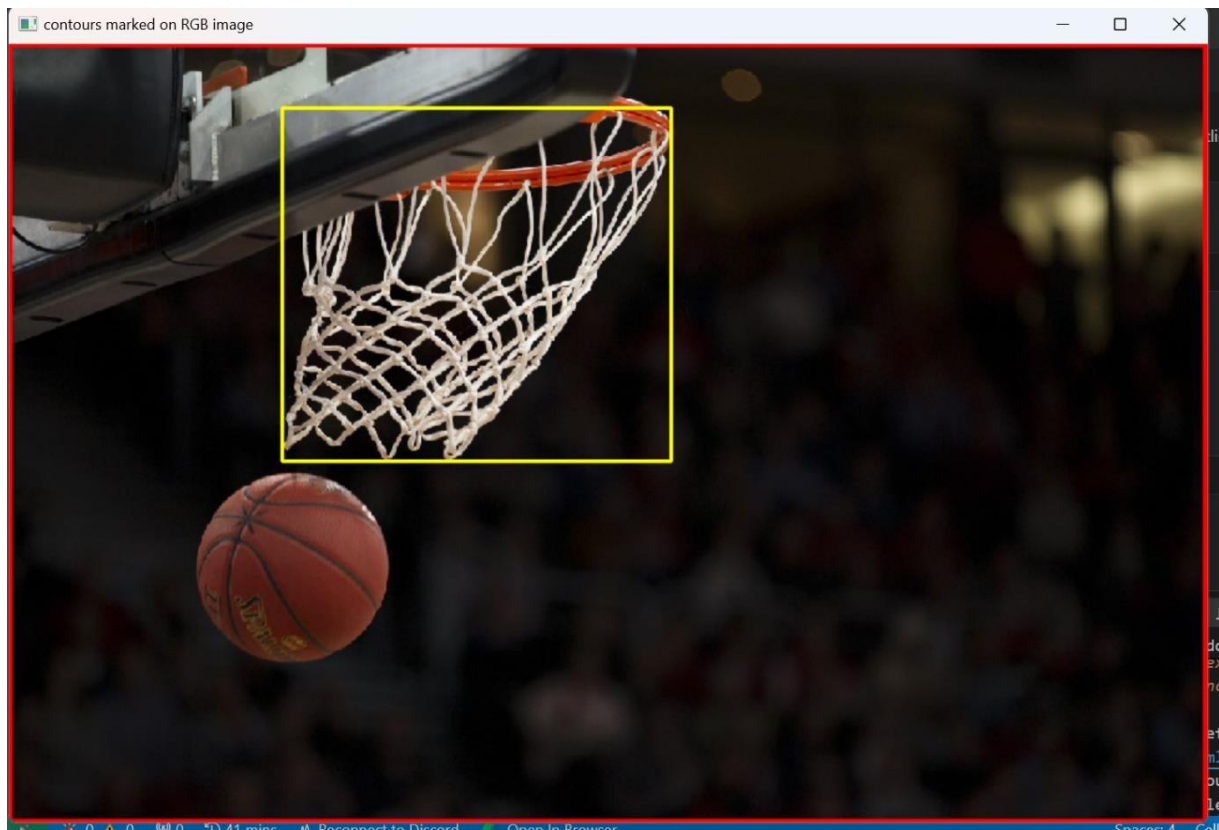
Output:







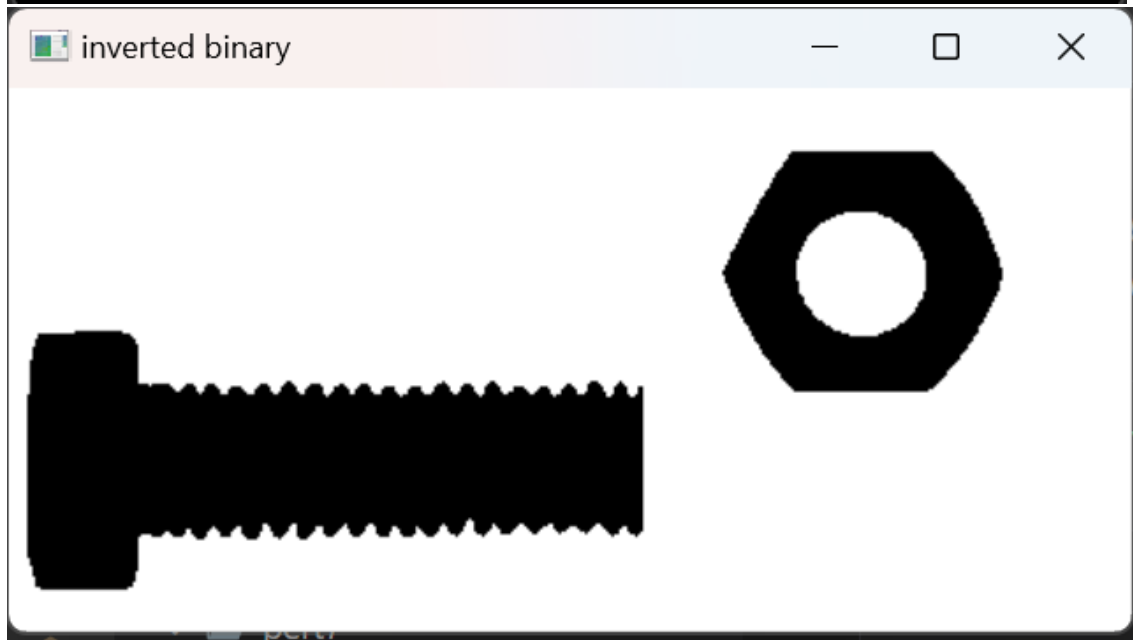


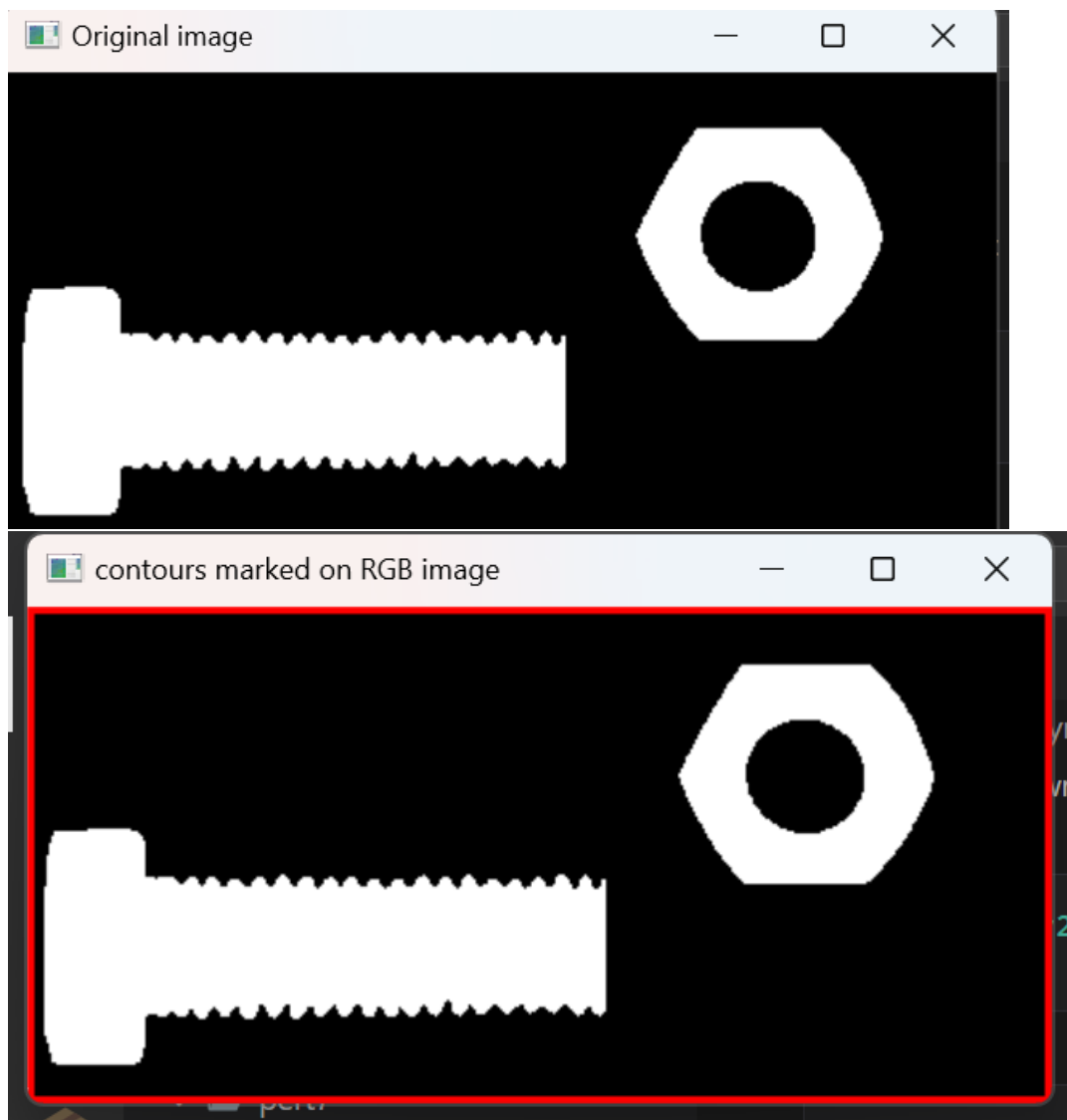


Nult\_bolt.ipynb

```
1  # %%
2  import cv2
3
4  # %%
5  image = cv2.imread('data/nut_bolt.png')
6  imagecopy= image.copy()
7  cv2.imshow( 'Original image' , image )
8  cv2.waitKey(0)
9  cv2.destroyAllWindows()
10
11 # %%
12 gray_image = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
13 cv2.imshow( 'gray' , gray_image )
14 cv2.waitKey(0)
15 cv2.destroyAllWindows()
16
17 # %%
18 ret,binary_im = cv2.threshold(gray_image,245,255,cv2.THRESH_BINARY)
19 cv2.imshow( 'binary' , binary_im )
20 cv2.waitKey(0)
21 cv2.destroyAllWindows()
22
23 # %%
24 binary_im= ~binary_im
25 cv2.imshow( 'inverted binary' , binary_im )
26 cv2.waitKey(0)
27 cv2.destroyAllWindows()
28
29 # %%
30 #find the external contours from binary image
31 contours,hierarchy = cv2.findContours(binary_im,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
32
33 # %%
34 with_contours = cv2.drawContours(image,contours,-1,(0,0,255),3)
35 cv2.imshow( 'contours marked on RGB image' , with_contours )
36 cv2.waitKey(0)
37 cv2.destroyAllWindows()
38
39
40
```

Output:

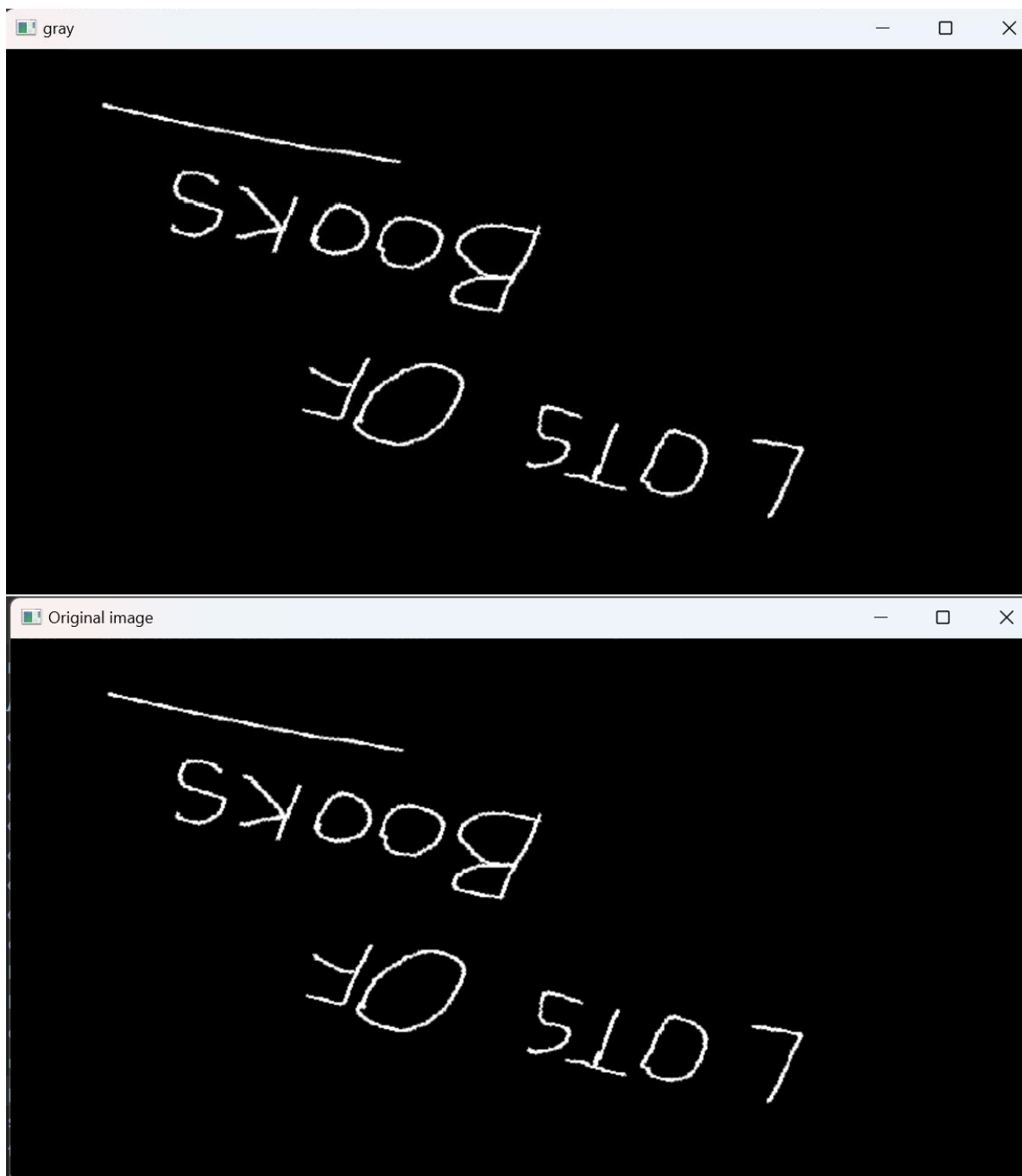




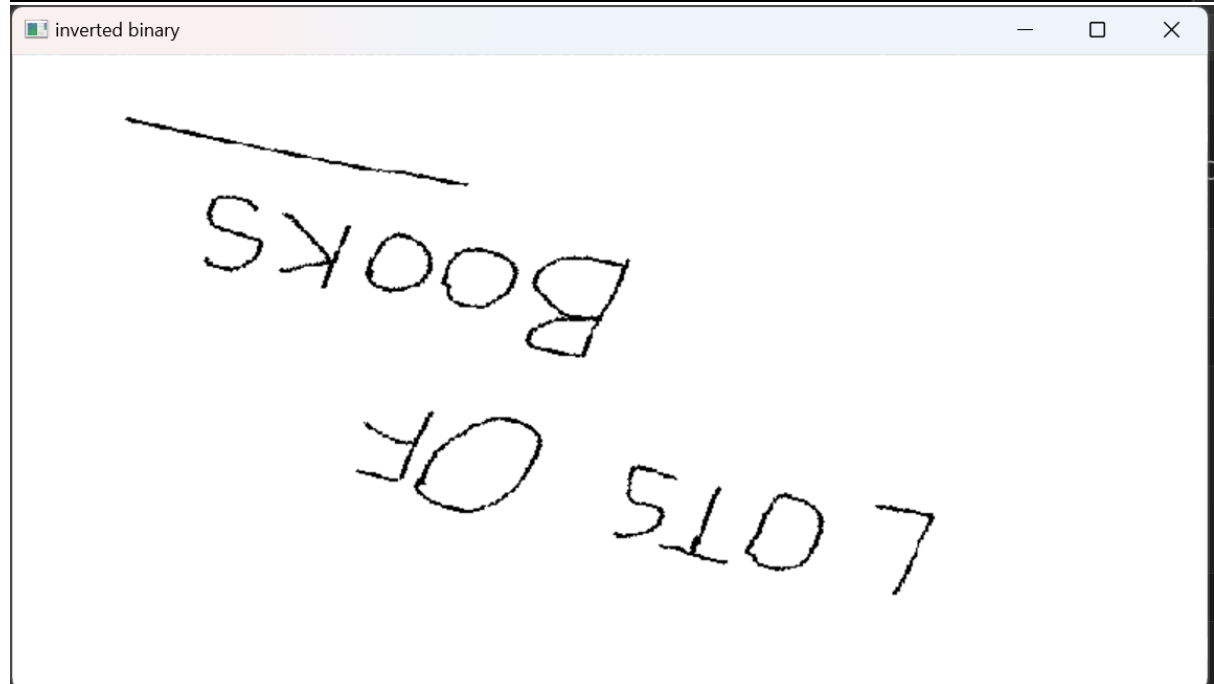
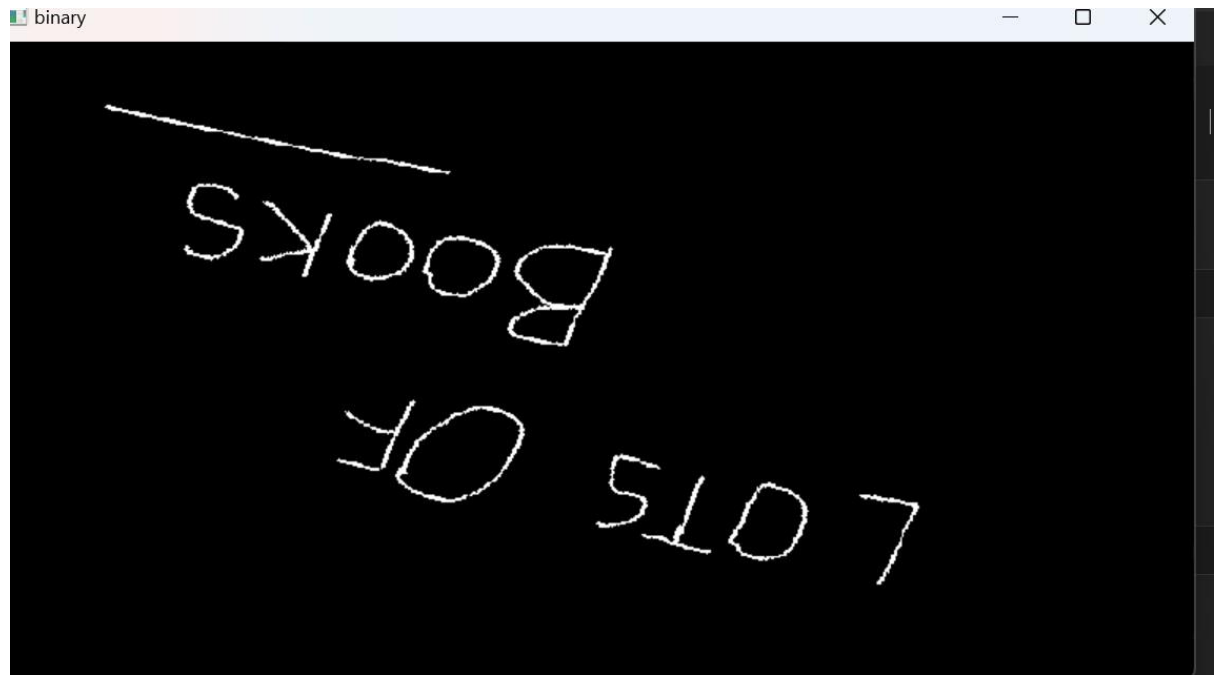
Phrase\_handwriting.ipynb

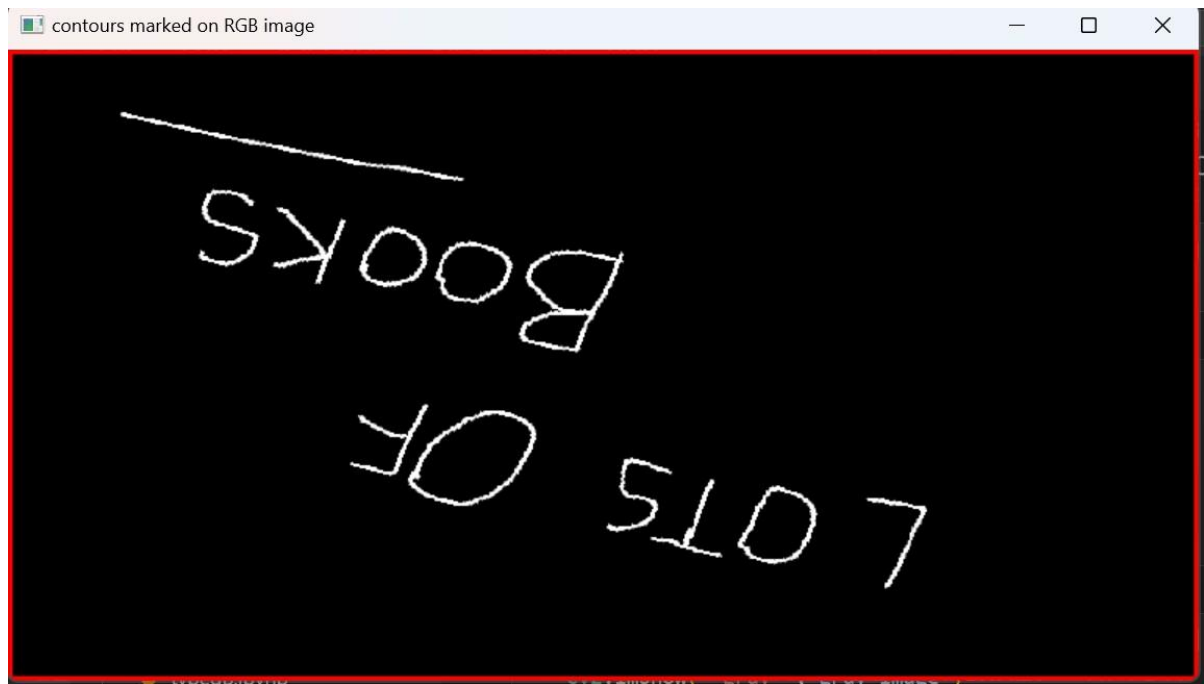
```
1  # %%
2  import cv2
3
4  # %%
5  image = cv2.imread('data/phrase_handwritten.png')
6  imagecopy= image.copy()
7  cv2.imshow( 'Original image' , image )
8  cv2.waitKey(0)
9  cv2.destroyAllWindows()
10
11 # %%
12 gray_image = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
13 cv2.imshow( 'gray' , gray_image )
14 cv2.waitKey(0)
15 cv2.destroyAllWindows()
16
17 # %%
18 ret,binary_im = cv2.threshold(gray_image,245,255,cv2.THRESH_BINARY)
19 cv2.imshow( 'binary' , binary_im )
20 cv2.waitKey(0)
21 cv2.destroyAllWindows()
22
23 # %%
24 binary_im= ~binary_im
25 cv2.imshow( 'inverted binary' , binary_im )
26 cv2.waitKey(0)
27 cv2.destroyAllWindows()
28
29 # %%
30 #find the external contours from binary image
31 contours,hierarchy = cv2.findContours(binary_im,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
32
33 # %%
34 with_contours = cv2.drawContours(image,contours,-1,(0,0,255),3)
35 cv2.imshow( 'contours marked on RGB image' , with_contours )
36 cv2.waitKey(0)
37 cv2.destroyAllWindows()
38
39
40
```

Output:





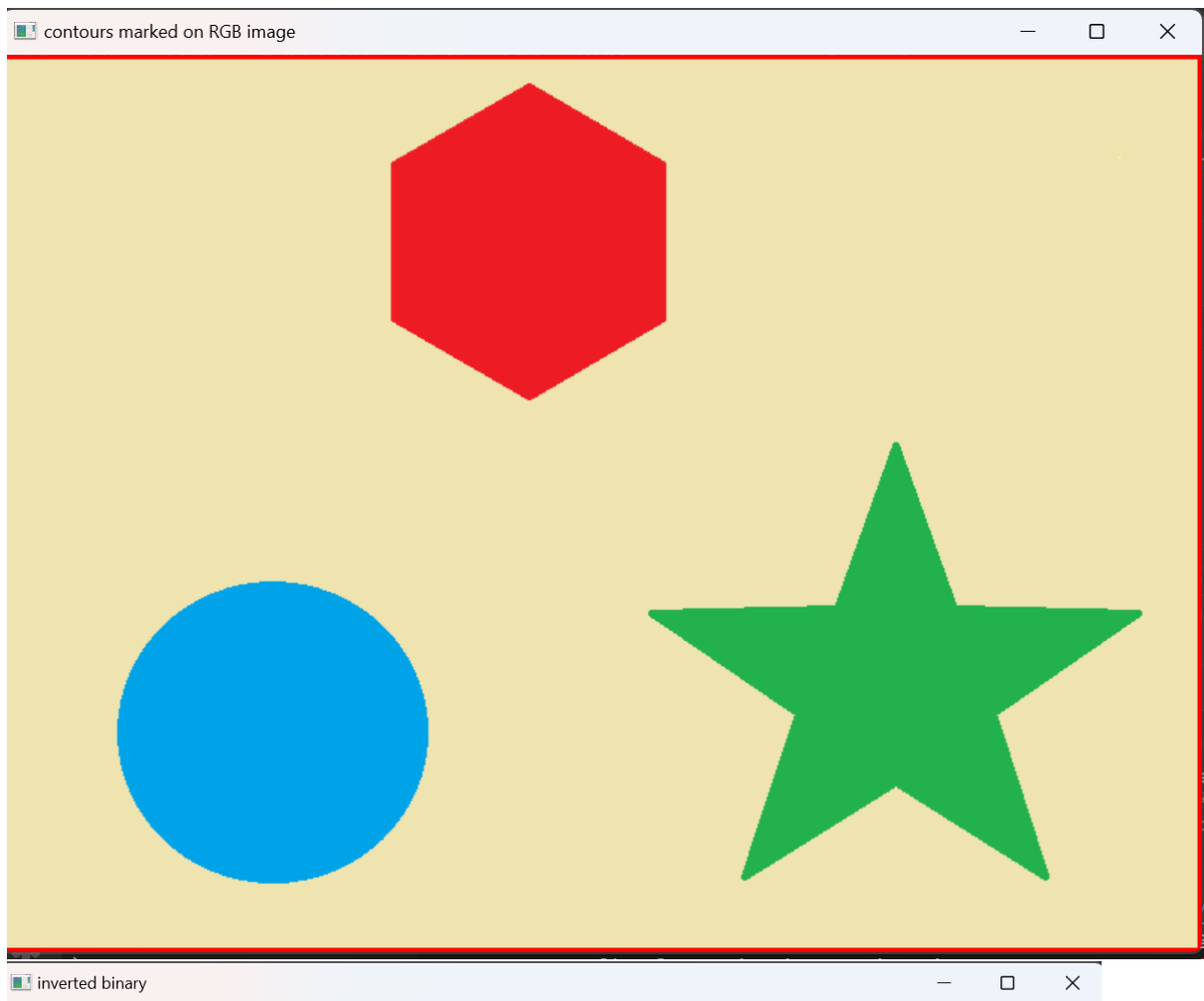


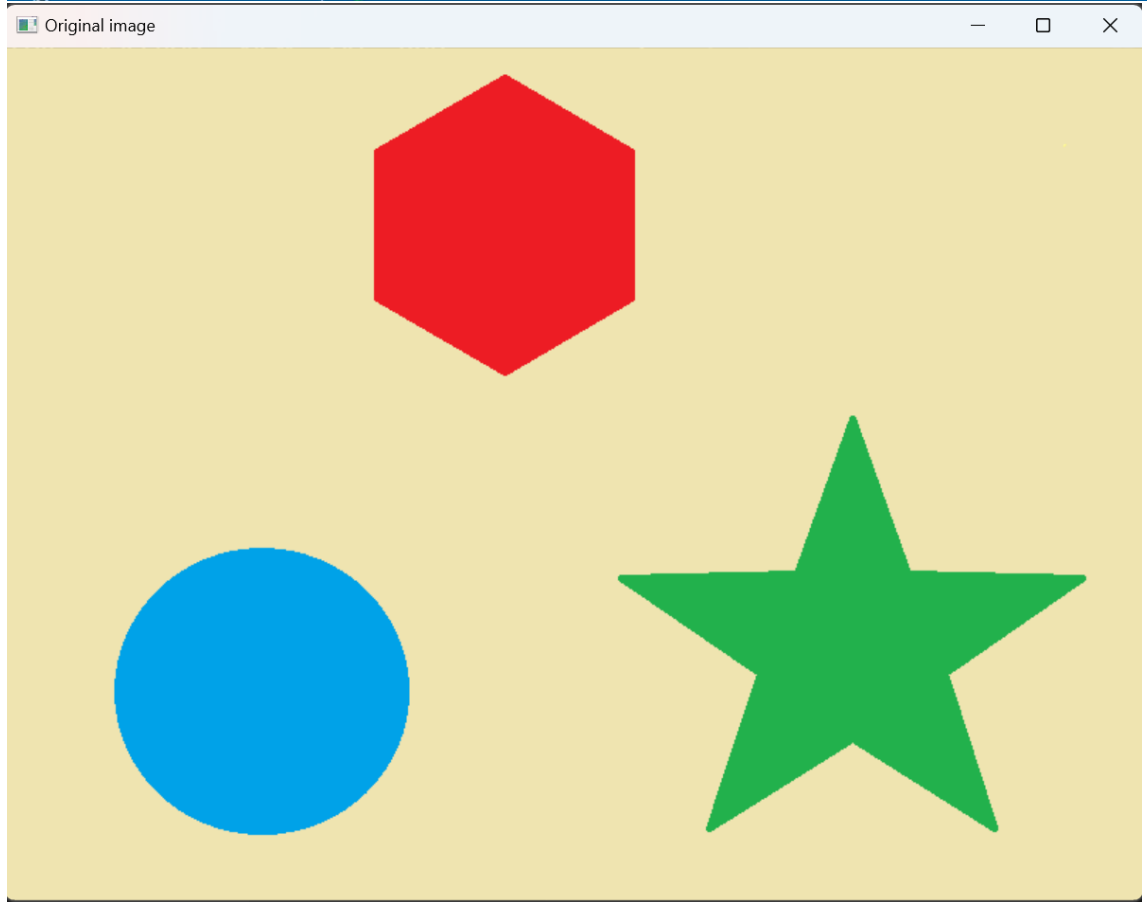
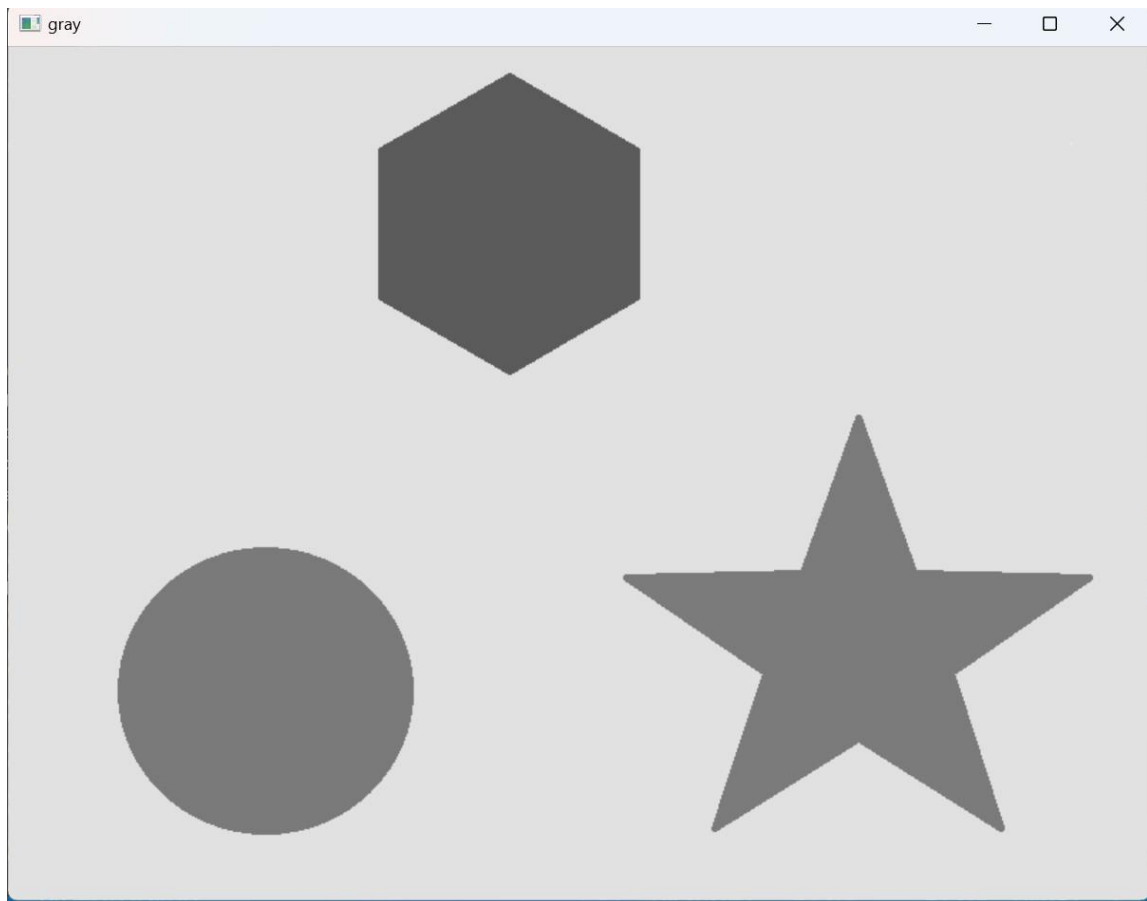


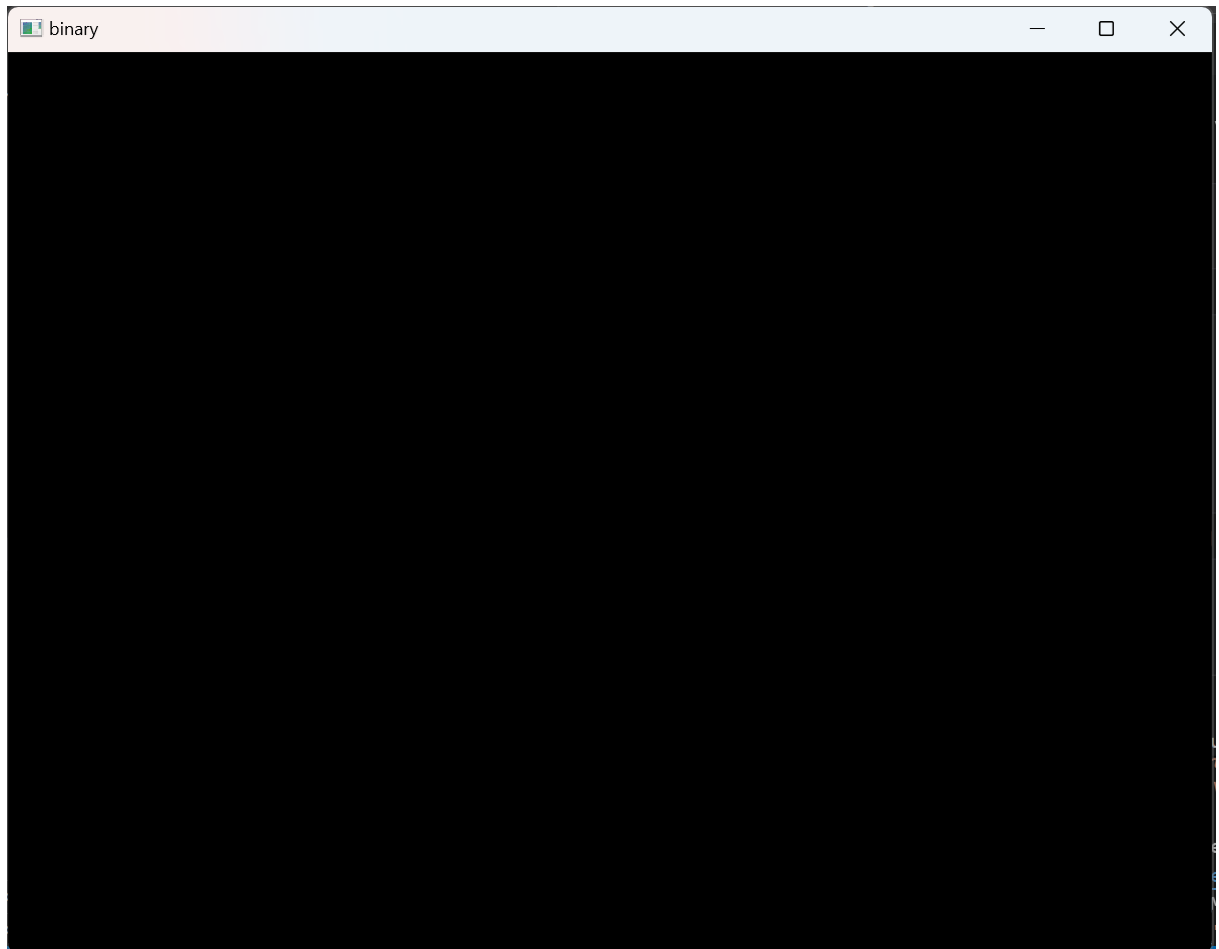
Sample\_shapes.ipynb

```
1  # %%
2  import cv2
3
4  # %%
5  image = cv2.imread('data/sample_shapes.png')
6  imagecopy= image.copy()
7  cv2.imshow( 'Original image' , image )
8  cv2.waitKey(0)
9  cv2.destroyAllWindows()
10
11 # %%
12 gray_image = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
13 cv2.imshow( 'gray' , gray_image )
14 cv2.waitKey(0)
15 cv2.destroyAllWindows()
16
17 # %%
18 ret,binary_im = cv2.threshold(gray_image,245,255,cv2.THRESH_BINARY)
19 cv2.imshow( 'binary' , binary_im )
20 cv2.waitKey(0)
21 cv2.destroyAllWindows()
22
23 # %%
24 binary_im= ~binary_im
25 cv2.imshow( 'inverted binary' , binary_im )
26 cv2.waitKey(0)
27 cv2.destroyAllWindows()
28
29 # %%
30 #find the external contours from binary image
31 contours,hierarchy = cv2.findContours(binary_im,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
32
33 # %%
34 with_contours = cv2.drawContours(image,contours,-1,(0,0,255),3)
35 cv2.imshow( 'contours marked on RGB image' , with_contours )
36 cv2.waitKey(0)
37 cv2.destroyAllWindows()
38
39
40
```

Output:







Many\_fruit.ipynb

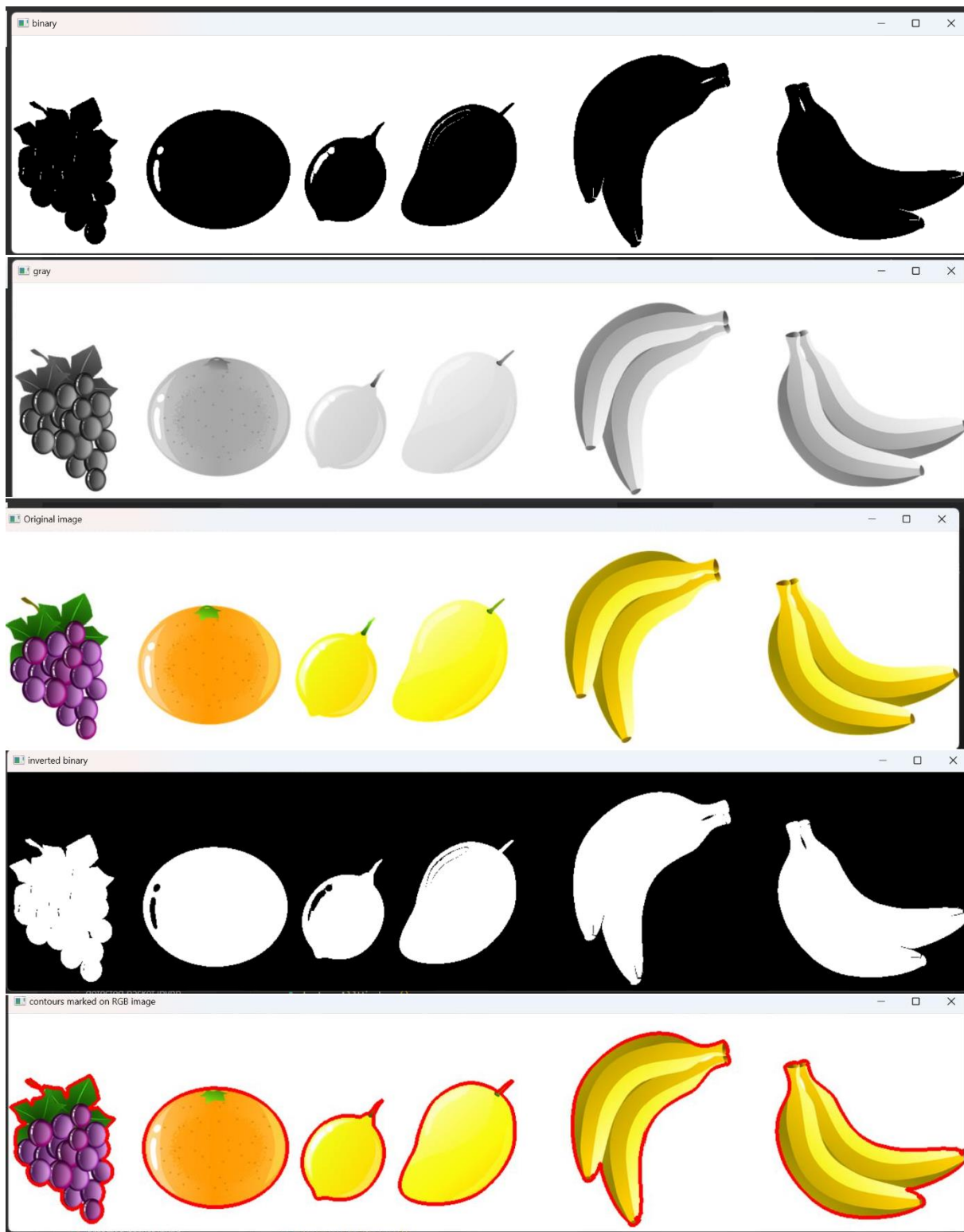
```

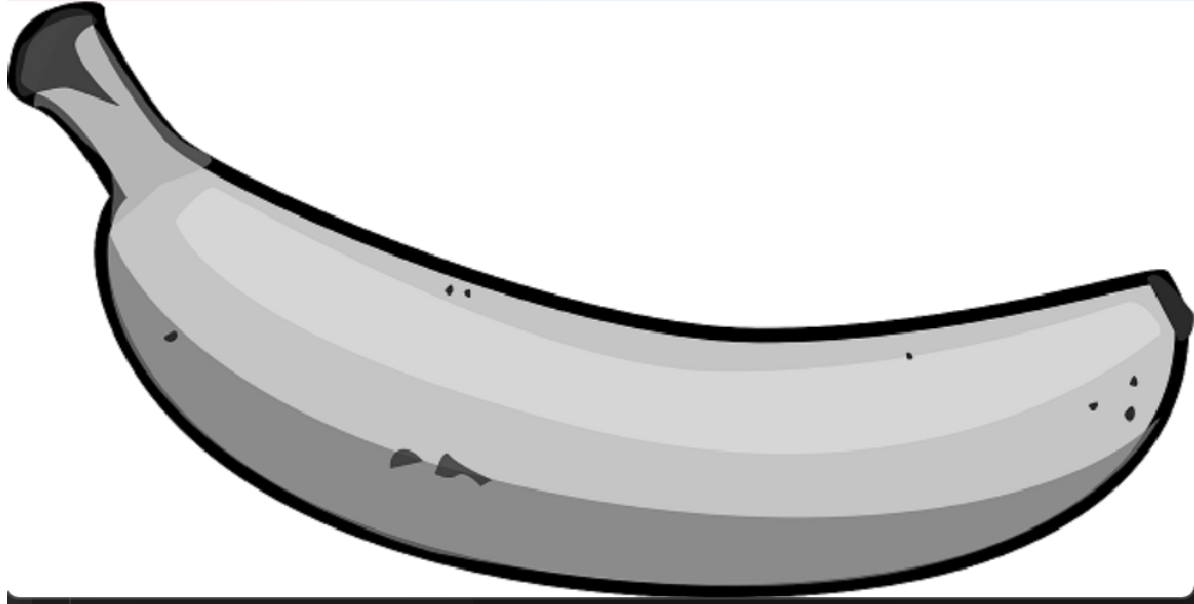
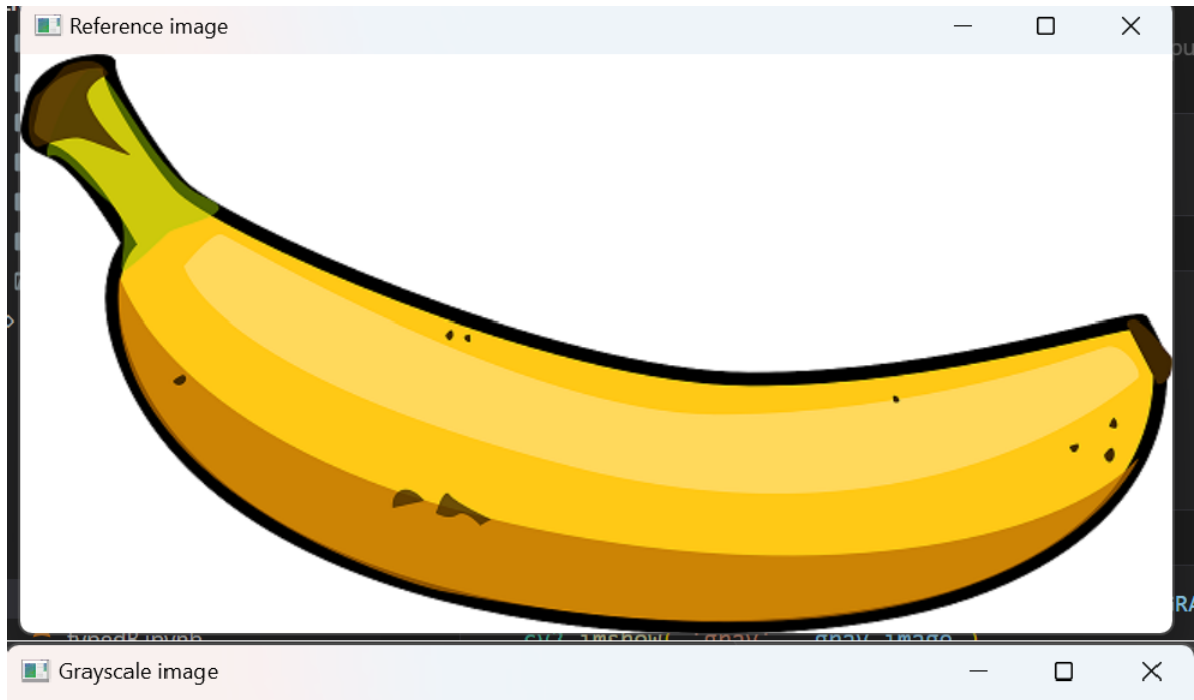
1  # %%
2  import cv2
3
4  # %%
5  image = cv2.imread('data/many_fruits.png')
6  imagecopy= image.copy()
7  cv2.imshow( 'Original image' , image )
8  cv2.waitKey(0)
9  cv2.destroyAllWindows()
10
11 # %%
12 gray_image = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
13 cv2.imshow( 'gray' , gray_image )
14 cv2.waitKey(0)
15 cv2.destroyAllWindows()
16
17 # %%
18 ret,binary_im = cv2.threshold(gray_image,245,255,cv2.THRESH_BINARY)
19 cv2.imshow( 'binary' , binary_im )
20 cv2.waitKey(0)
21 cv2.destroyAllWindows()
22
23 # %%
24 binary_im= ~binary_im
25 cv2.imshow( 'inverted binary' , binary_im )
26 cv2.waitKey(0)
27 cv2.destroyAllWindows()
28
29 # %%
30 #find the external contours from binary image
31 contours,hierarchy = cv2.findContours(binary_im,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
32
33 # %%
34 with_contours = cv2.drawContours(image,contours,-1,(0,0,255),3)
35 cv2.imshow( 'contours marked on RGB image' , with_contours )
36 cv2.waitKey(0)
37 cv2.destroyAllWindows()
38
39 # %%
40 ref_image = cv2.imread('data/bananaref.png')
41 cv2.imshow( 'Reference image' , ref_image )
42 cv2.waitKey(0)
43 cv2.destroyAllWindows()
44
45 # %%
46 gray_image = cv2.cvtColor(ref_image,cv2.COLOR_BGR2GRAY)
47 cv2.imshow( 'Grayscale image' , gray_image )
48 cv2.waitKey(0)
49 cv2.destroyAllWindows()
50
51 # %%
52 ret,binary_im = cv2.threshold(gray_image,245,255,cv2.THRESH_BINARY)
53 cv2.imshow( 'Binary image' , binary_im )
54 cv2.waitKey(0)
55 cv2.destroyAllWindows()
56
57 # %%
58 binary_im= ~binary_im
59 cv2.imshow( 'inverted binary image' , binary_im )
60 cv2.waitKey(0)
61 cv2.destroyAllWindows()
62
63 # %%
64 dist_list = []
65 for cnt in contours:
66     retval = cv2.matchShapes(cnt, binary_im, 1, 0) # reference_contour diganti dengan binary_im
67     dist_list.append(retval)
68
69 # %%
70 sorted_list= dist_list.copy()
71 sorted_list.sort() # sorts the list from smallest to largest
72 ind1_dist= dist_list.index(sorted_list[0])
73 ind2_dist= dist_list.index(sorted_list[1])
74
75 # %%
76 banana_cnts= []
77 banana_cnts.append(contours[ind1_dist])
78 banana_cnts.append(contours[ind2_dist])
79
80 # %%
81 with_contours = cv2.drawContours(image,banana_cnts,-1,(255,0,0),3)
82 cv2.imshow( 'contours marked on RGB image' , with_contours )
83 cv2.waitKey(0)
84 cv2.destroyAllWindows()
85
86 # %%
87 for cnt in banana_cnts:
88     x,y,w,h = cv2.boundingRect(cnt)
89     if h>w:
90         cv2.rectangle(imagecopy,(x,y),(x+w,y+h),(255,0,0),2)
91 cv2.imshow( 'Upright banana marked on RGB image' , imagecopy )
92 cv2.waitKey(0)
93 cv2.destroyAllWindows()
94
95
96

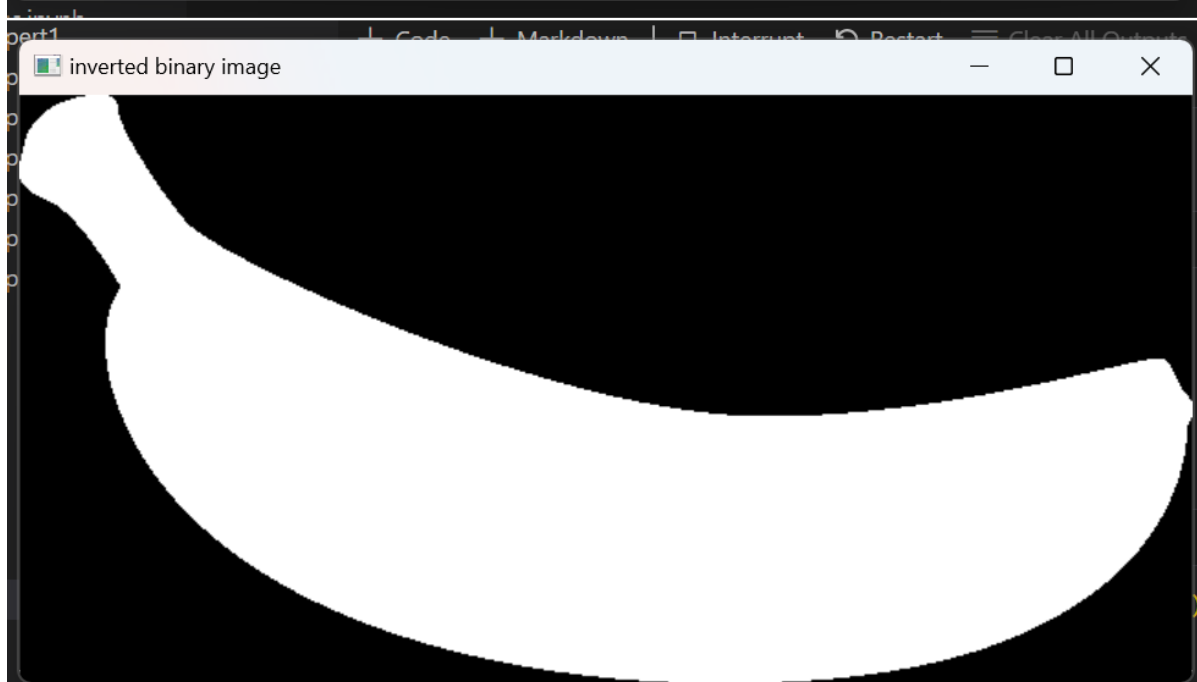
```



Output:









Typed\_b.ipynb

```
1  # %%
2  import cv2
3
4  # %%
5  image = cv2.imread('data/typed_B.png')
6  imagecopy= image.copy()
7  cv2.imshow( 'Original image' , image )
8  cv2.waitKey(0)
9  cv2.destroyAllWindows()
10
11 # %%
12 gray_image = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
13 cv2.imshow( 'gray' , gray_image )
14 cv2.waitKey(0)
15 cv2.destroyAllWindows()
16
17 # %%
18 ret,binary_im = cv2.threshold(gray_image,245,255,cv2.THRESH_BINARY)
19 cv2.imshow( 'binary' , binary_im )
20 cv2.waitKey(0)
21 cv2.destroyAllWindows()
22
23 # %%
24 binary_im= ~binary_im
25 cv2.imshow( 'inverted binary' , binary_im )
26 cv2.waitKey(0)
27 cv2.destroyAllWindows()
28
29 # %%
30 #find the external contours from binary image
31 contours,hierarchy = cv2.findContours(binary_im,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
32
33 # %%
34 with_contours = cv2.drawContours(image,contours,-1,(0,0,255),3)
35 cv2.imshow( 'contours marked on RGB image' , with_contours )
36 cv2.waitKey(0)
37 cv2.destroyAllWindows()
38
39
40
```

Output:

