atividade-6

November 13, 2017

```
In [116]: import os

          import cv2
          import numpy as np
          import imutils

          from imutils import paths
          from sklearn.neighbors import KNeighborsClassifier
          from skimage import exposure
          from skimage import measure
          from skimage import feature

          import pylab as plt
          %matplotlib inline
```

```
In [2]: def exibir_imagens(list_process=None):
            if isinstance(list_process, list):
                for obj in list_process:
                    plt.figure()
                    plt.title(s=obj.get('title', ''))
                    plt.xticks(obj.get('xticks', []))
                    plt.yticks(obj.get('yticks', []))
                    plt.imshow(X=obj.get('X'), cmap=obj.get('cmap', None))
                    plt.tight_layout()
            else:
                obj = list_process
                plt.figure()
                plt.title(s=obj.get('title', ''))
                plt.xticks(obj.get('xticks', []))
                plt.yticks(obj.get('yticks', []))
                plt.imshow(X=obj.get('X'), cmap=obj.get('cmap', None))
                plt.tight_layout()
```

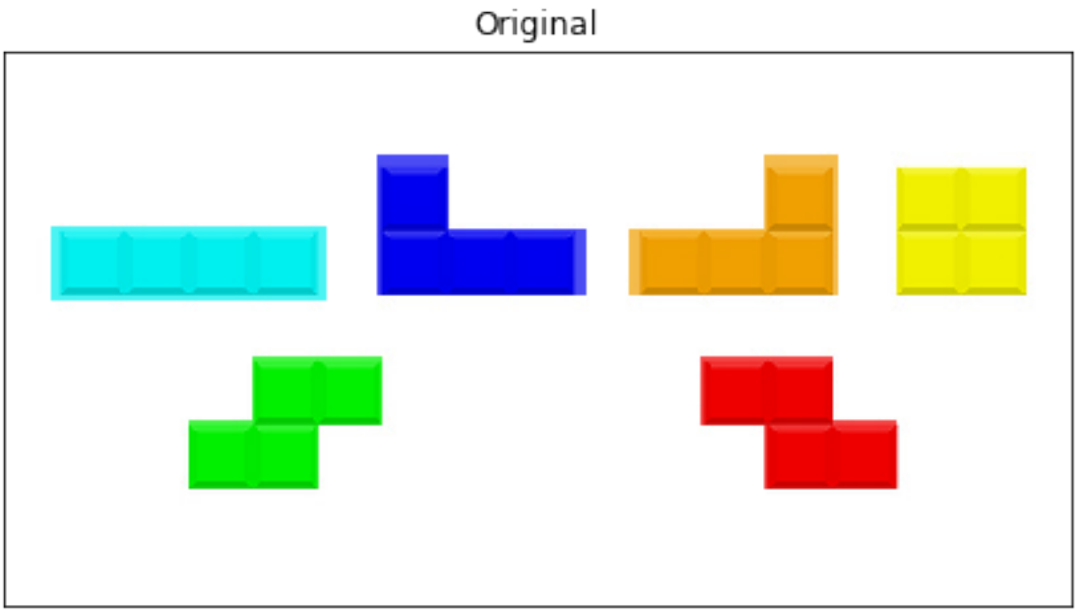
1 1 - Detectar formas geométricas

```
In [3]: path = os.getcwd() + os.sep
        png = path + '../../db_images/png/captcha.png'
```

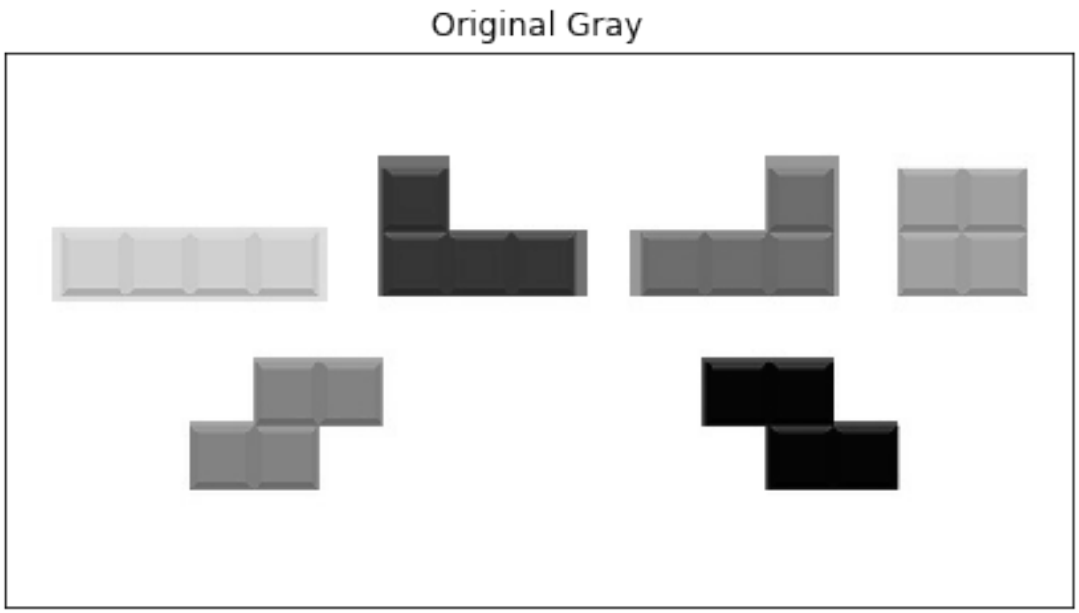
```
jpeg = path + '../../db_images/jpeg/captcha.jpeg'
other = path + '../../db_aulas/Imagens/tetris_blocks.png'

path_images = [other]
```

In [4]:
```
images = []
for path_img in path_images:
    img = cv2.imread(path_img)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    it = {'title': 'Original',
          'X': img}
    images.append(it)

grays = []
for obj in images:
    gray = cv2.cvtColor(obj['X'], cv2.COLOR_BGR2GRAY)
    it = {'cmap': 'gray',
          'title': 'Original Gray',
          'X': gray}
    grays.append(it)
```

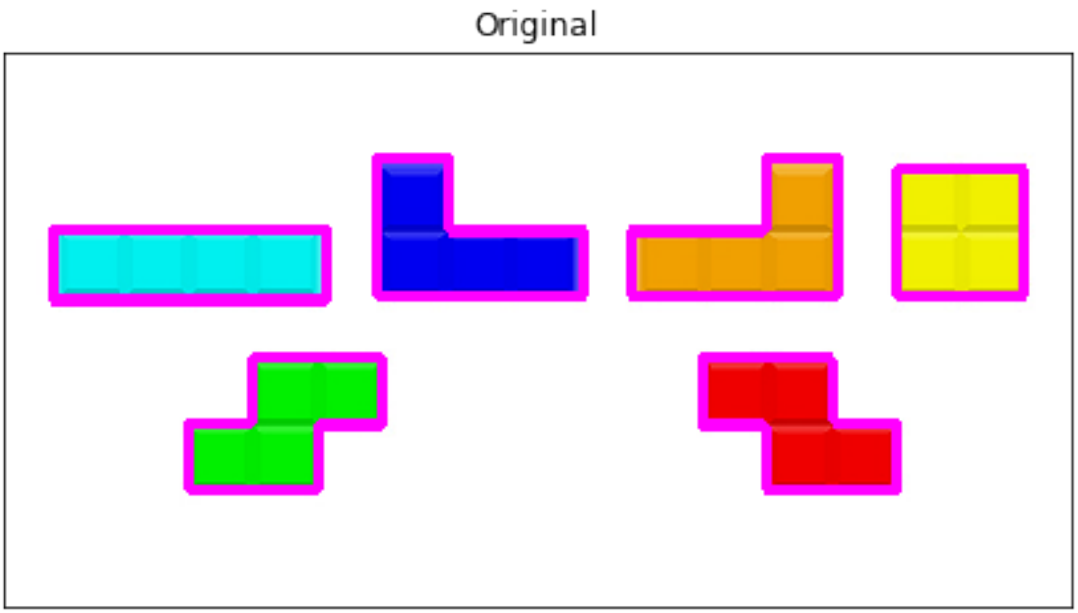
In [5]:
```
exibir_imagens(images)
```

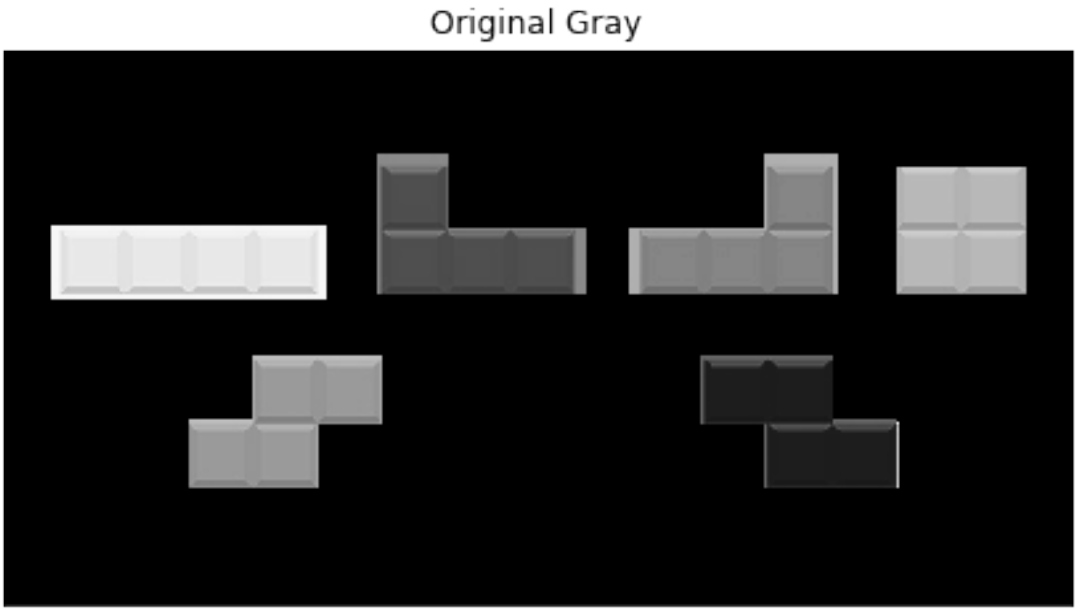


In [6]:
```
exibir_imagens(grays)
```




```
In [7]: contours = []
        for i in range(len(images)):
            gray = grays[i].copy()
            img = images[i].copy()

            img_gray = gray.get('X').copy()
            origin = img.get('X').copy()

            #img_gray = cv2.medianBlur(img_gray, 5)
            #img_gray = cv2.GaussianBlur(img_gray, (5,5), 0.)

            img_gray[(img_gray[:,:] > 233)] = 0

            (_, cnts, hierarquia) = cv2.findContours(img_gray,
                                                     cv2.RETR_EXTERNAL,
                                                     cv2.CHAIN_APPROX_SIMPLE)
            clone = img.get('X').copy()
            cv2.drawContours(clone, cnts, -1, (255,0,255), 3)

            img.update({'X': clone})
            contours.append(img)

            gray.update({'X': img_gray, 'cnts': cnts, 'origin': origin, 'gray': img_gray})
            contours.append(gray)

In [8]: exibir_imagens(contours)
```

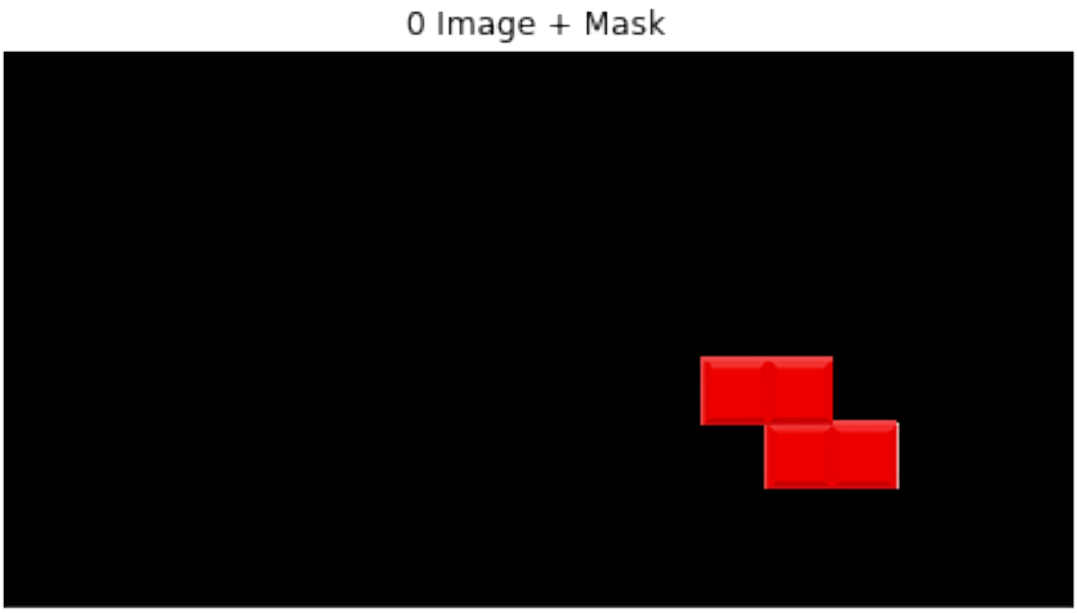




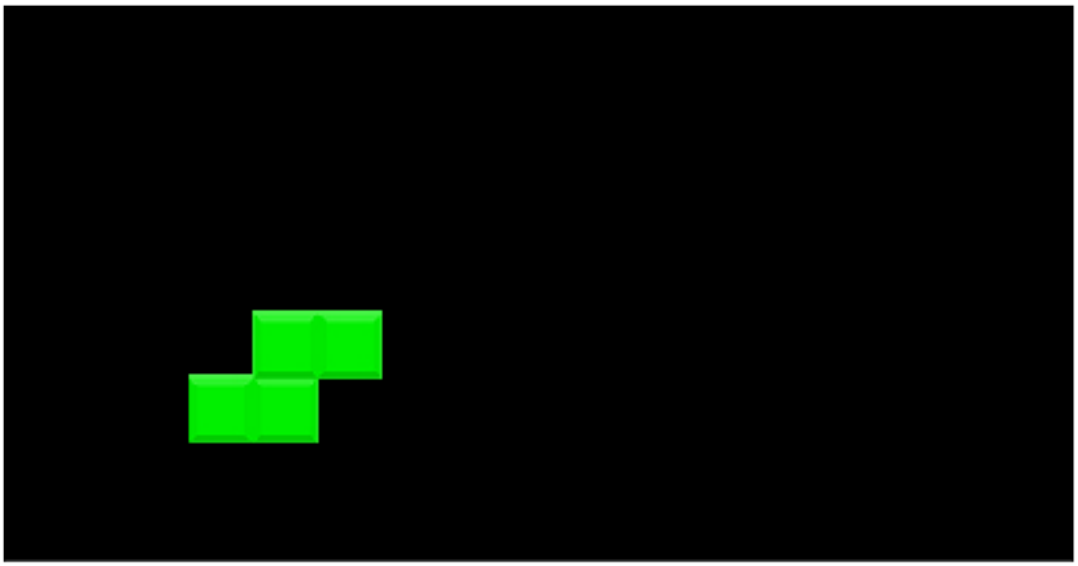

```
In [9]: objetos = []
        for img_contour in contours:
            if img_contour.get('cnts'):
                cnts = img_contour.get('cnts')
                for i, c in enumerate(cnts):
```

```
                    img = img_contour.copy()

                    clone = img.get('origin').copy()

                    mask = np.zeros(img.get('X').shape, dtype='uint8')
                    cv2.drawContours(mask, [c], -1, 255, -1)

                    clone_gray = img.get('gray').copy()

                    mask_gray = np.zeros(img.get('X').shape, dtype='uint8')
                    cv2.drawContours(mask_gray, [c], -1, 255, -1)

                    img.update(
                        {
                            'title': str((i))+' Image + Mask',
                            'X': cv2.bitwise_and(clone, clone, mask=mask),
                            'index': i,
                            'gray':  cv2.bitwise_and(clone_gray, clone_gray, mask=mask_gray),
                        }
                    )
                    objetos.append(img)
```

In [10]:
```
exibir_imagens(objetos)
```



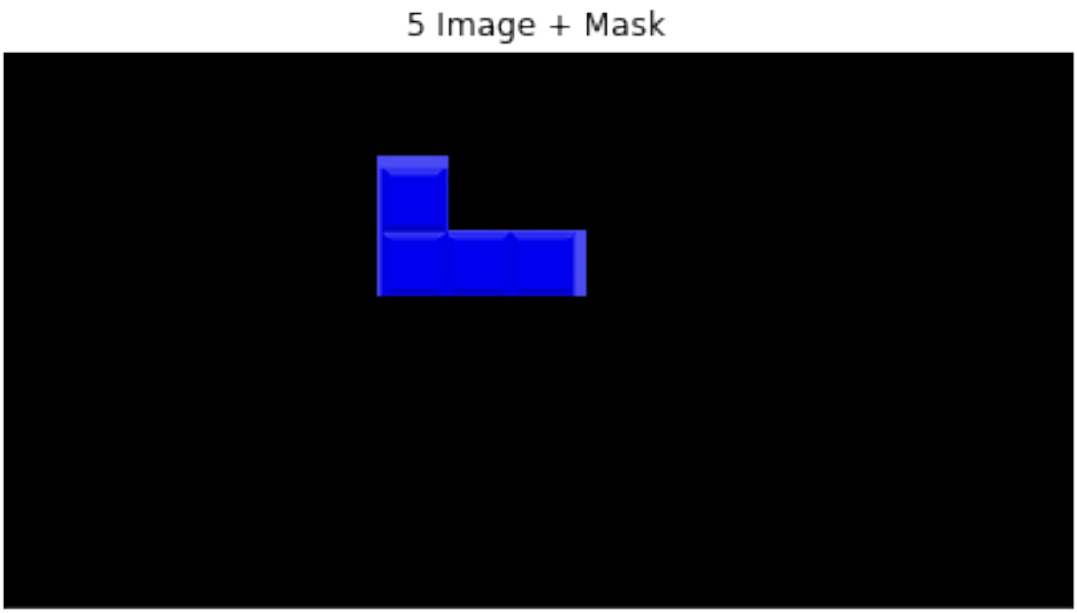
1 Image + Mask



2 Image + Mask

3 Image + Mask

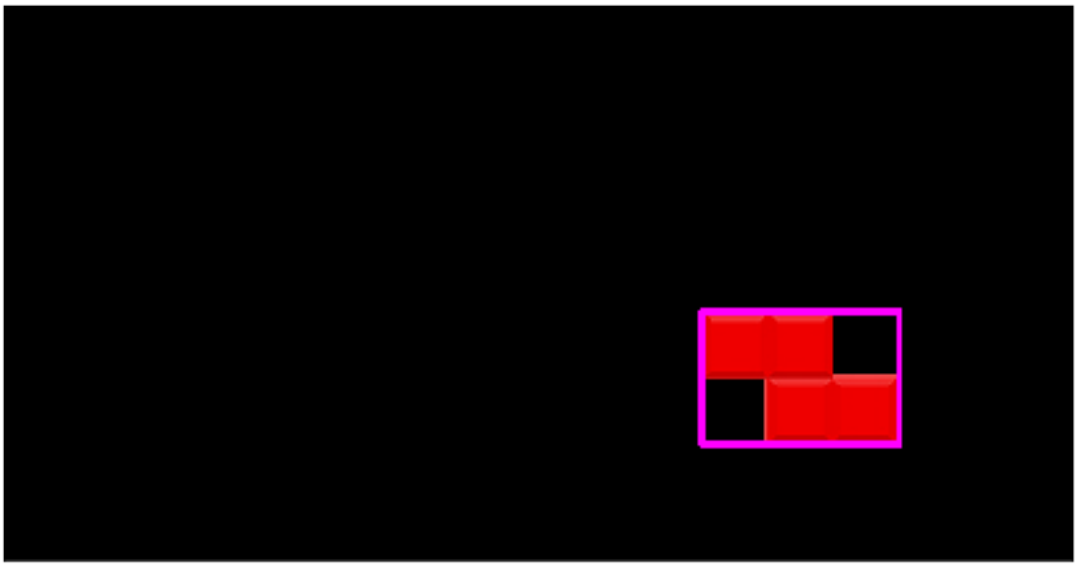
4 Image + Mask



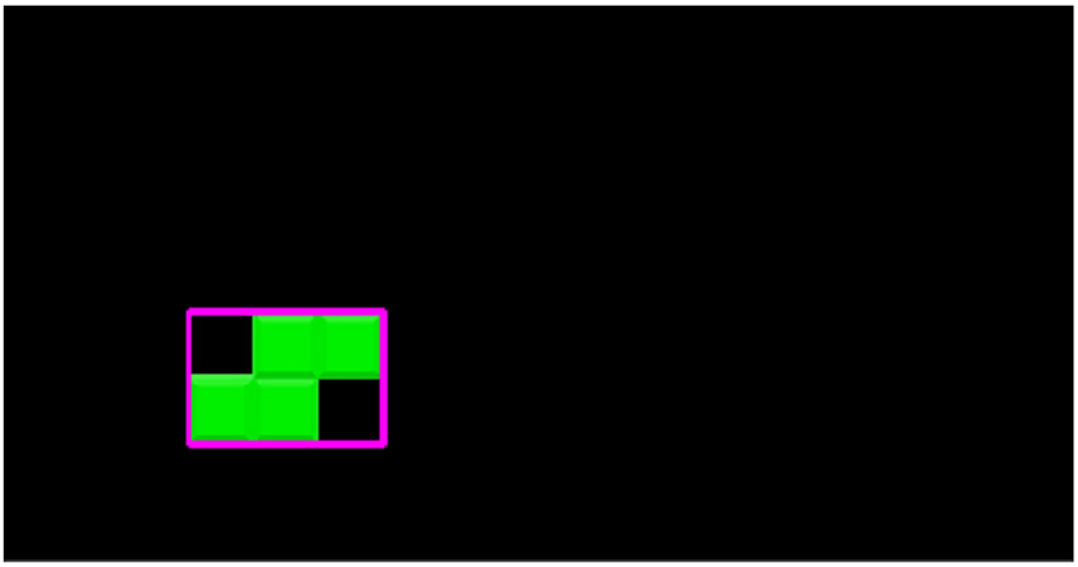

```
In [11]: rects_objs = []
         for obj in objetos:
             it = obj.copy()
             cnts = it.get('cnts').copy()
             for i, c in enumerate(cnts):
                 if it.get('index') == i:
                     it = obj.copy()
                     clone = it.get('X').copy()

                     (x, y, w, h) = cv2.boundingRect(c)
                     cv2.rectangle(clone, (x,y), (x+w,y+h), (255,0,255), 2)

                     it.update({'X': clone})
                     rects_objs.append(it)
```

```
In [12]: exibir_imagens(rects_objs)
```

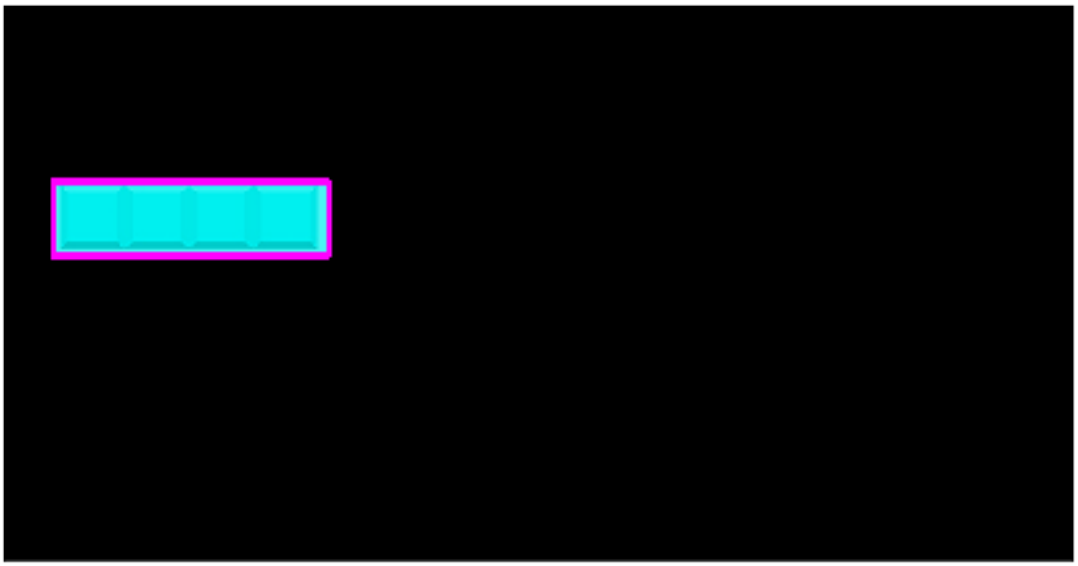
0 Image + Mask



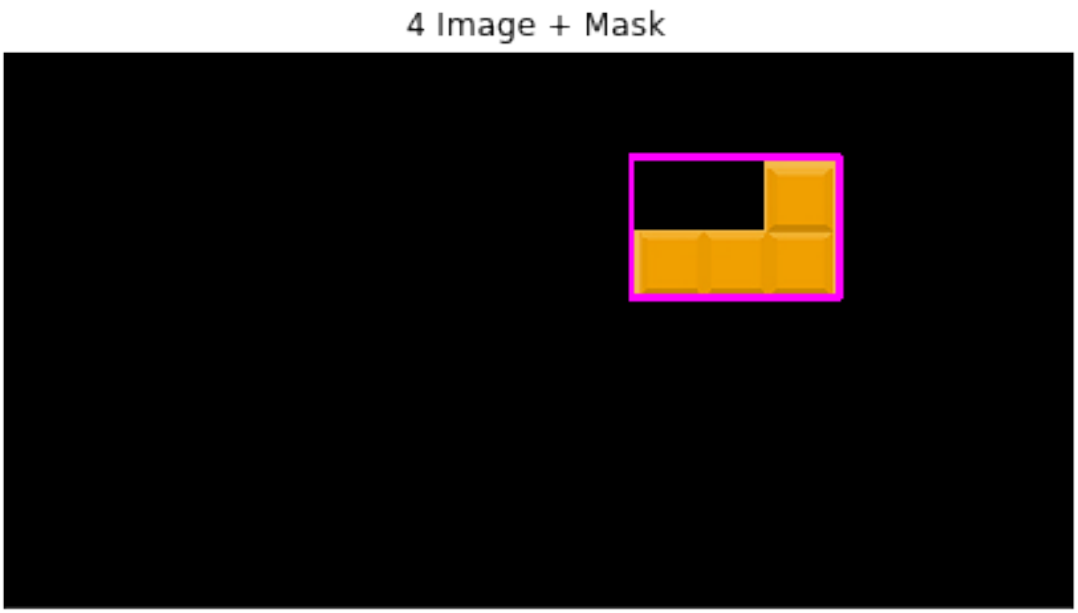
1 Image + Mask

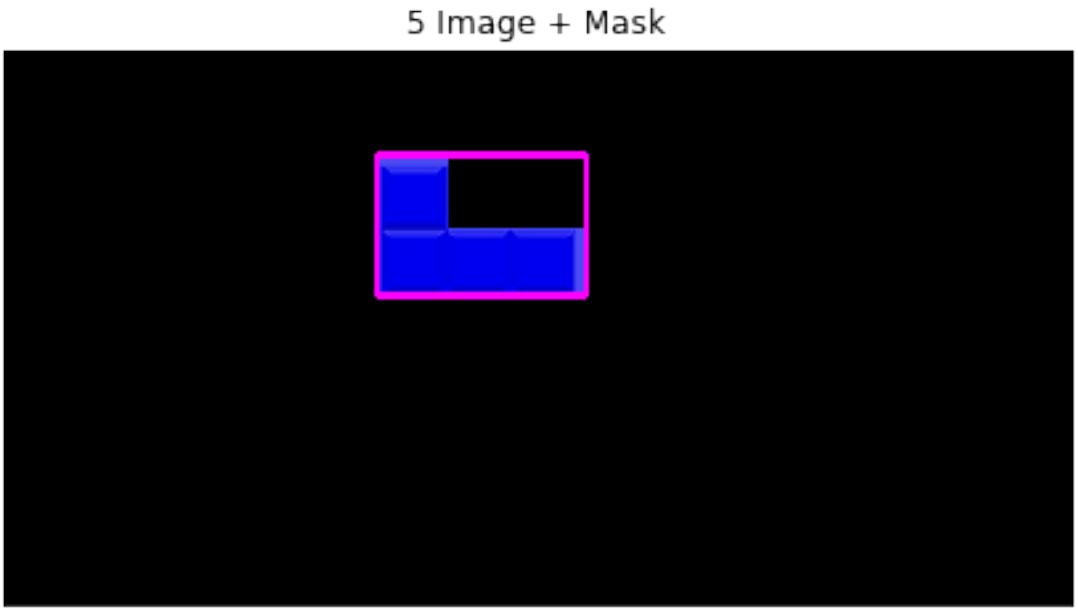


2 Image + Mask



3 Image + Mask






```
In [19]: for obj in objetos:
             it = obj.copy()

             gray = it.get('gray').copy()
             gray[(gray != 0)] = 1 # binary
```

```
it.update({'X': gray})

sample_image = it.get('X')
label_img = measure.label(sample_image, neighbors=8) # neighbors 4 or 8
props = measure.regionprops(label_img,  intensity_image=sample_image)

print('id:\n{}\n'.format(it.get('index')))
print('H:\n{}\n'.format(measure.shannon_entropy(it.get('X'), np.e)))
print('area:\n{}\n'.format(props[0].area))
print('bbox:\n{}\n'.format(props[0].bbox))
print('bbox_area:\n{}\n'.format(props[0].bbox_area))
print('centroid:\n{}\n'.format(props[0].centroid))
print('convex_area:\n{}\n'.format(props[0].convex_area)) # not 3d
print('convex_image:\n{}\n'.format(props[0].convex_image)) # not 3d
print('coords:\n{}\n'.format(props[0].coords))
print('eccentricity:\n{}\n'.format(props[0].eccentricity))  # not 3d
print('equivalent_diameter:\n{}\n'.format(props[0].equivalent_diameter))
print('euler_number:\n{}\n'.format(props[0].euler_number))  # not 3d
print('extent:\n{}\n'.format(props[0].extent))
print('filled_area:\n{}\n'.format(props[0].filled_area))
print('filled_image:\n{}\n'.format(props[0].filled_image))
print('image:\n{}\n'.format(props[0].image))
print('inertia_tensor:\n{}\n'.format(props[0].inertia_tensor)) # not 3d
print('inertia_tensor_eigvals:\n{}\n'.format(props[0].inertia_tensor_eigvals)) #
print('intensity_image:\n{}\n'.format(props[0].intensity_image))
print('local_centroid:\n{}\n'.format(props[0].local_centroid)) # not 3d
print('major_axis_length:\n{}\n'.format(props[0].major_axis_length)) # not 3d
print('max_intensity:\n{}\n'.format(props[0].max_intensity))
print('mean_intensity:\n{}\n'.format(props[0].mean_intensity))
print('min_intensity:\n{}\n'.format(props[0].min_intensity))
print('minor_axis_length:\n{}\n'.format(props[0].minor_axis_length)) # not 3d
print('moments:\n{}\n'.format(props[0].moments))  # not 3d
print('moments_central:\n{}\n'.format(props[0].moments_central))  # not 3d
print('moments_hu:\n{}\n'.format(props[0].moments_hu))  # not 3d
print('moments_normalized:\n{}\n'.format(props[0].moments_normalized))  # not 3d
print('orientation:\n{}\n'.format(props[0].orientation)) # not 3d
print('perimeter:\n{}\n'.format(props[0].perimeter)) # not 3d and obrigatório bin
print('solidity:\n{}\n'.format(props[0].solidity)) # not 3d
print('weighted_centroid:\n{}\n'.format(props[0].weighted_centroid)) # not 3d
print('weighted_local_centroid:\n{}\n'.format(props[0].weighted_local_centroid))
print('weighted_moments:\n{}\n'.format(props[0].weighted_moments)) # not 3d
print('weighted_moments_central:\n{}\n'.format(props[0].weighted_moments_central))
print('weighted_moments_hu:\n{}\n'.format(props[0].weighted_moments_hu)) # not 3d
print('weighted_moments_normalized:\n{}\n'.format(props[0].weighted_moments_normal

print()
```

```
id:
```

```
0

H:
8.204124932574041

area:
3656

bbox:
(138, 314, 198, 403)

bbox_area:
120500

centroid:
(167.5082056892779, 357.98386214442013)

convex_area:
4440

convex_image:
[[ True  True  True ..., False False False]
 [ True  True  True ..., False False False]
 [ True  True  True ..., False False False]
 ...,
 [False False False ...,  True  True  True]
 [False False False ...,  True  True  True]
 [False False False ...,  True  True  True]]

coords:
[[138 314]
 [138 315]
 [138 316]
 ...,
 [197 400]
 [197 401]
 [197 402]]

eccentricity:
0.8631982223315521

equivalent_diameter:
68.22729494529118

euler_number:
1

extent:
```

```
0.6846441947565544

filled_area:
3656

filled_image:
[[ True  True  True ..., False False False]
 [ True  True  True ..., False False False]
 [ True  True  True ..., False False False]
 ...,
 [False False False ...,  True  True  True]
 [False False False ...,  True  True  True]
 [False False False ...,  True  True  True]]

image:
[[ True  True  True ..., False False False]
 [ True  True  True ..., False False False]
 [ True  True  True ..., False False False]
 ...,
 [False False False ...,  True  True  True]
 [False False False ...,  True  True  True]
 [False False False ...,  True  True  True]]

inertia_tensor:
[[ 517.14935664 -214.05770901]
 [-214.05770901  295.08855411]]

inertia_tensor_eigvals:
(647.25885831584094, 164.97905243247274)

intensity_image:
[[1 1 1 ..., 0 0 0]
 [1 1 1 ..., 0 0 0]
 [1 1 1 ..., 0 0 0]
 ...,
 [0 0 0 ..., 1 1 1]
 [0 0 0 ..., 1 1 1]
 [0 0 0 ..., 1 1 1]]

local_centroid:
(29.508205689277901, 43.983862144420129)

major_axis_length:
101.76513024142137

max_intensity:
1
```

```
mean_intensity:
1.0

min_intensity:
1

minor_axis_length:
51.37766867929649

moments:
[[  3.65600000e+03   1.07882000e+05   4.26224800e+06   1.89439892e+08]
 [  1.60805000e+05   5.52766200e+06   2.33647578e+08   1.07283821e+10]
 [  8.96352300e+06   3.33357930e+08   1.45057490e+10   6.74638415e+11]
 [  5.60573741e+08   2.19541001e+10   9.71959121e+11   4.55344489e+13]]

moments_central:
[[  3.65600000e+03  -1.52951429e-09   1.07884375e+06  -8.85471890e+02]
 [ -1.53465862e-09   7.82594984e+05  -8.49798388e+03   3.52539589e+08]
 [  1.89069805e+06   1.73498212e+04   5.50644172e+08   3.59973450e+06]
 [  2.97645149e+03   8.68307410e+08   9.45015790e+06   3.90977778e+11]]

moments_hu:
[  2.22165731e-01   1.74014637e-02   5.53092164e-09   4.61685088e-10
  -7.85044797e-20  -5.49676757e-11   7.33575441e-19]

moments_normalized:
[[             nan              nan   8.07134995e-02  -1.09561838e-06]
 [             nan   5.85497016e-02  -1.05147859e-05   7.21421918e-03]
 [  1.41452231e-01   2.14674043e-05   1.12681465e-02   1.21828477e-06]
 [  3.68284415e-06   1.77686710e-02   3.19828684e-06   2.18840312e-03]]

orientation:
-0.5461528919251896

perimeter:
291.65685424949237

solidity:
0.8234234234234235

weighted_centroid:
(167.5082056892779, 357.98386214442013)

weighted_local_centroid:
(29.508205689277901, 43.983862144420129)

weighted_moments:
[[  3.65600000e+03   1.07882000e+05   4.26224800e+06   1.89439892e+08]
```

```
 [  1.60805000e+05   5.52766200e+06   2.33647578e+08   1.07283821e+10]
 [  8.96352300e+06   3.33357930e+08   1.45057490e+10   6.74638415e+11]
 [  5.60573741e+08   2.19541001e+10   9.71959121e+11   4.55344489e+13]]

weighted_moments_central:
[[  3.65600000e+03  -1.52951429e-09   1.07884375e+06  -8.85471890e+02]
 [ -1.53465862e-09   7.82594984e+05  -8.49798388e+03   3.52539589e+08]
 [  1.89069805e+06   1.73498212e+04   5.50644172e+08   3.59973450e+06]
 [  2.97645149e+03   8.68307410e+08   9.45015790e+06   3.90977778e+11]]

weighted_moments_hu:
[  2.22165731e-01   1.74014637e-02   5.53092164e-09   4.61685088e-10
  -7.85044797e-20  -5.49676757e-11   7.33575441e-19]

weighted_moments_normalized:
[[             nan              nan   8.07134995e-02  -1.09561838e-06]
 [             nan   5.85497016e-02  -1.05147859e-05   7.21421918e-03]
 [  1.41452231e-01   2.14674043e-05   1.12681465e-02   1.21828477e-06]
 [  3.68284415e-06   1.77686710e-02   3.19828684e-06   2.18840312e-03]]


id:
1

H:
8.18813341451048

area:
3598

bbox:
(138, 83, 198, 171)

bbox_area:
120500

centroid:
(167.5, 126.5)

convex_area:
4410

convex_image:
[[False False False ...,  True  True  True]
 [False False False ...,  True  True  True]
 [False False False ...,  True  True  True]
 ...,
 [ True  True  True ..., False False False]
```

```
 [ True  True  True ..., False False False]
 [ True  True  True ..., False False False]]

coords:
[[138 112]
 [138 113]
 [138 114]
 ...,
 [197 139]
 [197 140]
 [197 141]]

eccentricity:
0.8636803529877504

equivalent_diameter:
67.68394109356454

euler_number:
1

extent:
0.6814393939393939

filled_area:
3598

filled_image:
[[False False False ...,  True  True  True]
 [False False False ...,  True  True  True]
 [False False False ...,  True  True  True]
 ...,
 [ True  True  True ..., False False False]
 [ True  True  True ..., False False False]
 [ True  True  True ..., False False False]]

image:
[[False False False ...,  True  True  True]
 [False False False ...,  True  True  True]
 [False False False ...,  True  True  True]
 ...,
 [ True  True  True ..., False False False]
 [ True  True  True ..., False False False]
 [ True  True  True ..., False False False]]

inertia_tensor:
[[ 507.34282935  213.75611451]
 [ 213.75611451  295.08602001]]
```

```
inertia_tensor_eigvals:
(639.86671309853921, 162.56213626221674)

intensity_image:
[[0 0 0 ..., 1 1 1]
 [0 0 0 ..., 1 1 1]
 [0 0 0 ..., 1 1 1]
 ...,
 [1 1 1 ..., 0 0 0]
 [1 1 1 ..., 0 0 0]
 [1 1 1 ..., 0 0 0]]

local_centroid:
(29.5, 43.5)

major_axis_length:
101.1823473219347

max_intensity:
1

mean_intensity:
1.0

min_intensity:
1

minor_axis_length:
50.99994294306091

moments:
[[  3.59800000e+03   1.06141000e+05   4.19287900e+06   1.86331381e+08]
 [  1.56513000e+05   3.84803900e+06   1.37013661e+08   5.75121681e+09]
 [  8.63373500e+06   1.87783961e+08   6.10591610e+09   2.41638304e+11]
 [  5.34378969e+08   1.05674080e+10   3.15139630e+11   1.16799276e+13]]

moments_central:
[[  3.59800000e+03   0.00000000e+00   1.06171950e+06   0.00000000e+00]
 [  0.00000000e+00  -7.69094500e+05   0.00000000e+00  -3.46284799e+08]
 [  1.82541950e+06   0.00000000e+00   5.31131560e+08   0.00000000e+00]
 [  0.00000000e+00  -8.30814334e+08   0.00000000e+00  -3.74074154e+11]]

moments_hu:
[ 0.2230208   0.01759823  0.          0.          0.          0.          0.        ]

moments_normalized:
[[        nan         nan  0.0820139   0.        ]
```

```
 [        nan -0.0594097   0.          -0.00743447]
 [ 0.1410069   0.          0.01140299  0.        ]
 [ 0.         -0.01783695  0.          -0.0022321 ]]

orientation:
0.5549791443742736

perimeter:
290.8284271247462

solidity:
0.8158730158730159

weighted_centroid:
(167.5, 126.5)

weighted_local_centroid:
(29.5, 43.5)

weighted_moments:
[[  3.59800000e+03   1.06141000e+05   4.19287900e+06   1.86331381e+08]
 [  1.56513000e+05   3.84803900e+06   1.37013661e+08   5.75121681e+09]
 [  8.63373500e+06   1.87783961e+08   6.10591610e+09   2.41638304e+11]
 [  5.34378969e+08   1.05674080e+10   3.15139630e+11   1.16799276e+13]]

weighted_moments_central:
[[  3.59800000e+03   0.00000000e+00   1.06171950e+06   0.00000000e+00]
 [  0.00000000e+00  -7.69094500e+05   0.00000000e+00  -3.46284799e+08]
 [  1.82541950e+06   0.00000000e+00   5.31131560e+08   0.00000000e+00]
 [  0.00000000e+00  -8.30814334e+08   0.00000000e+00  -3.74074154e+11]]

weighted_moments_hu:
[ 0.2230208   0.01759823  0.          0.          0.          0.          0.        ]

weighted_moments_normalized:
[[        nan         nan  0.0820139   0.        ]
 [        nan -0.0594097   0.          -0.00743447]
 [ 0.1410069   0.          0.01140299  0.        ]
 [ 0.         -0.01783695  0.          -0.0022321 ]]


id:
2

H:
8.31678912707152

area:
```

```
4092

bbox:
(80, 22, 113, 146)

bbox_area:
120500

centroid:
(96.0, 83.5)

convex_area:
4092

convex_image:
[[ True  True  True ...,  True  True  True]
 [ True  True  True ...,  True  True  True]
 [ True  True  True ...,  True  True  True]
 ...,
 [ True  True  True ...,  True  True  True]
 [ True  True  True ...,  True  True  True]
 [ True  True  True ...,  True  True  True]]

coords:
[[ 80  22]
 [ 80  23]
 [ 80  24]
 ...,
 [112 143]
 [112 144]
 [112 145]]

eccentricity:
0.9639687610901733

equivalent_diameter:
72.18099623208512

euler_number:
1

extent:
1.0

filled_area:
4092

filled_image:
```

```
[[ True  True  True ...,  True  True  True]
 [ True  True  True ...,  True  True  True]
 [ True  True  True ...,  True  True  True]
 ...,
 [ True  True  True ...,  True  True  True]
 [ True  True  True ...,  True  True  True]
 [ True  True  True ...,  True  True  True]]

image:
[[ True  True  True ...,  True  True  True]
 [ True  True  True ...,  True  True  True]
 [ True  True  True ...,  True  True  True]
 ...,
 [ True  True  True ...,  True  True  True]
 [ True  True  True ...,  True  True  True]
 [ True  True  True ...,  True  True  True]]

inertia_tensor:
[[ 1281.25          -0.        ]
 [   -0.            90.66666667]]

inertia_tensor_eigvals:
(1281.25, 90.666666666666742)

intensity_image:
[[1 1 1 ..., 1 1 1]
 [1 1 1 ..., 1 1 1]
 [1 1 1 ..., 1 1 1]
 ...,
 [1 1 1 ..., 1 1 1]
 [1 1 1 ..., 1 1 1]
 [1 1 1 ..., 1 1 1]]

local_centroid:
(16.0, 61.5)

major_axis_length:
143.17821063276352

max_intensity:
1

mean_intensity:
1.0

min_intensity:
1
```

```
minor_axis_length:
38.08761828556188

moments:
[[  4.09200000e+03   6.54720000e+04   1.41856000e+06   3.45692160e+07]
 [  2.51658000e+05   4.02652800e+06   8.72414400e+07   2.12600678e+09]
 [  2.07198420e+07   3.31517472e+08   7.18287856e+09   1.75041225e+11]
 [  1.91914391e+09   3.07063025e+10   6.65303221e+11   1.62129277e+13]]

moments_central:
[[  4.09200000e+03   0.00000000e+00   3.71008000e+05   0.00000000e+00]
 [  0.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00]
 [  5.24287500e+06   0.00000000e+00   4.75354000e+08   0.00000000e+00]
 [  0.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00]]

moments_hu:
[ 0.335268    0.08465417  0.          0.          0.          0.          0.        ]

moments_normalized:
[[        nan         nan  0.02215705  0.        ]
 [        nan  0.          0.          0.        ]
 [ 0.31311095  0.          0.00693762  0.        ]
 [ 0.          0.          0.          0.        ]]

orientation:
-0.0

perimeter:
310.0

solidity:
1.0

weighted_centroid:
(96.0, 83.5)

weighted_local_centroid:
(16.0, 61.5)

weighted_moments:
[[  4.09200000e+03   6.54720000e+04   1.41856000e+06   3.45692160e+07]
 [  2.51658000e+05   4.02652800e+06   8.72414400e+07   2.12600678e+09]
 [  2.07198420e+07   3.31517472e+08   7.18287856e+09   1.75041225e+11]
 [  1.91914391e+09   3.07063025e+10   6.65303221e+11   1.62129277e+13]]

weighted_moments_central:
[[  4.09200000e+03   0.00000000e+00   3.71008000e+05   0.00000000e+00]
 [  0.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00]
```

```
 [  5.24287500e+06   0.00000000e+00   4.75354000e+08   0.00000000e+00]
 [  0.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00]]

weighted_moments_hu:
[ 0.335268    0.08465417  0.          0.          0.          0.          0.        ]

weighted_moments_normalized:
[[        nan         nan  0.02215705  0.        ]
 [        nan  0.          0.          0.        ]
 [ 0.31311095  0.          0.00693762  0.        ]
 [ 0.          0.          0.          0.        ]]


id:
3

H:
8.120886021092836

area:
3364

bbox:
(53, 402, 111, 460)

bbox_area:
120500

centroid:
(81.5, 430.5)

convex_area:
3364

convex_image:
[[ True  True  True ...,  True  True  True]
 [ True  True  True ...,  True  True  True]
 [ True  True  True ...,  True  True  True]
 ...,
 [ True  True  True ...,  True  True  True]
 [ True  True  True ...,  True  True  True]
 [ True  True  True ...,  True  True  True]]

coords:
[[ 53 402]
 [ 53 403]
 [ 53 404]
 ...,
```

```
 [110 457]
 [110 458]
 [110 459]]

eccentricity:
0.0

equivalent_diameter:
65.44599169153973

euler_number:
1

extent:
1.0

filled_area:
3364

filled_image:
[[ True  True  True ...,  True  True  True]
 [ True  True  True ...,  True  True  True]
 [ True  True  True ...,  True  True  True]
 ...,
 [ True  True  True ...,  True  True  True]
 [ True  True  True ...,  True  True  True]
 [ True  True  True ...,  True  True  True]]

image:
[[ True  True  True ...,  True  True  True]
 [ True  True  True ...,  True  True  True]
 [ True  True  True ...,  True  True  True]
 ...,
 [ True  True  True ...,  True  True  True]
 [ True  True  True ...,  True  True  True]
 [ True  True  True ...,  True  True  True]]

inertia_tensor:
[[ 280.25   -0.  ]
 [  -0.    280.25]]

inertia_tensor_eigvals:
(280.25, 280.25)

intensity_image:
[[1 1 1 ..., 1 1 1]
 [1 1 1 ..., 1 1 1]
 [1 1 1 ..., 1 1 1]
```

```
 ...,
 [1 1 1 ..., 1 1 1]
 [1 1 1 ..., 1 1 1]
 [1 1 1 ..., 1 1 1]]

local_centroid:
(28.5, 28.5)

major_axis_length:
66.96267617113283

max_intensity:
1

mean_intensity:
1.0

min_intensity:
1

minor_axis_length:
66.96267617113283

moments:
[[  3.36400000e+03   9.58740000e+04   3.67517000e+06   1.58479722e+08]
 [  9.58740000e+04   2.73240900e+06   1.04742345e+08   4.51667208e+09]
 [  3.67517000e+06   1.04742345e+08   4.01512322e+09   1.73139096e+11]
 [  1.58479722e+08   4.51667208e+09   1.73139096e+11   7.46605894e+12]]

moments_central:
[[  3.36400000e+03   0.00000000e+00   9.42761000e+05   0.00000000e+00]
 [  0.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00]
 [  9.42761000e+05   0.00000000e+00   2.64208770e+08   0.00000000e+00]
 [  0.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00]]

moments_hu:
[ 0.16661712  0.          0.          0.          0.          0.          0.        ]

moments_normalized:
[[        nan         nan  0.08330856  0.        ]
 [        nan  0.          0.          0.        ]
 [ 0.08330856  0.          0.00694032  0.        ]
 [ 0.          0.          0.          0.        ]]

orientation:
0.7853981633974483

perimeter:
```

```
228.0

solidity:
1.0

weighted_centroid:
(81.5, 430.5)

weighted_local_centroid:
(28.5, 28.5)

weighted_moments:
[[  3.36400000e+03   9.58740000e+04   3.67517000e+06   1.58479722e+08]
 [  9.58740000e+04   2.73240900e+06   1.04742345e+08   4.51667208e+09]
 [  3.67517000e+06   1.04742345e+08   4.01512322e+09   1.73139096e+11]
 [  1.58479722e+08   4.51667208e+09   1.73139096e+11   7.46605894e+12]]

weighted_moments_central:
[[  3.36400000e+03   0.00000000e+00   9.42761000e+05   0.00000000e+00]
 [  0.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00]
 [  9.42761000e+05   0.00000000e+00   2.64208770e+08   0.00000000e+00]
 [  0.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00]]

weighted_moments_hu:
[ 0.16661712  0.          0.          0.          0.          0.          0.        ]

weighted_moments_normalized:
[[        nan         nan  0.08330856  0.        ]
 [        nan  0.          0.          0.        ]
 [ 0.08330856  0.          0.00694032  0.        ]
 [ 0.          0.          0.          0.        ]]


id:
4

H:
8.271036865792956

area:
3909

bbox:
(48, 282, 111, 376)

bbox_area:
120500
```

```
centroid:
(86.724481964696849, 336.99693016116652)

convex_area:
4899

convex_image:
[[False False False ...,  True  True  True]
 [False False False ...,  True  True  True]
 [False False False ...,  True  True  True]
 ...,
 [ True  True  True ...,  True  True  True]
 [ True  True  True ...,  True  True  True]
 [ True  True  True ...,  True  True  True]]

coords:
[[ 48 343]
 [ 48 344]
 [ 48 345]
 ...,
 [110 373]
 [110 374]
 [110 375]]

eccentricity:
0.8619855117512796

equivalent_diameter:
70.54851791759874

euler_number:
1

extent:
0.660081053698075

filled_area:
3909

filled_image:
[[False False False ...,  True  True  True]
 [False False False ...,  True  True  True]
 [False False False ...,  True  True  True]
 ...,
 [ True  True  True ...,  True  True  True]
 [ True  True  True ...,  True  True  True]
 [ True  True  True ...,  True  True  True]]
```

```
image:
[[False False False ...,  True  True  True]
 [False False False ...,  True  True  True]
 [False False False ...,  True  True  True]
 ...,
 [ True  True  True ...,  True  True  True]
 [ True  True  True ...,  True  True  True]
 [ True  True  True ...,  True  True  True]]

inertia_tensor:
[[ 743.35685934  193.0883362 ]
 [ 193.0883362   278.72352701]]

inertia_tensor_eigvals:
(813.12319328540207, 208.95719306351708)

intensity_image:
[[0 0 0 ..., 1 1 1]
 [0 0 0 ..., 1 1 1]
 [0 0 0 ..., 1 1 1]
 ...,
 [1 1 1 ..., 1 1 1]
 [1 1 1 ..., 1 1 1]
 [1 1 1 ..., 1 1 1]]

local_centroid:
(38.724481964696857, 54.996930161166539)

major_axis_length:
114.06126026204704

max_intensity:
1

mean_intensity:
1.0

min_intensity:
1

minor_axis_length:
57.82140684051429

moments:
[[  3.90900000e+03   1.51374000e+05   6.95141000e+06   3.41529822e+08]
 [  2.14983000e+05   7.57032300e+06   3.34754925e+08   1.61617328e+10]
 [  1.47291870e+07   4.93140747e+08   2.13269647e+10   1.01863852e+12]
 [  1.09314168e+09   3.55450977e+10   1.51641012e+12   7.19392881e+13]]
```

```
moments_central:
[[  3.90900000e+03  -3.95871069e-09   1.08953027e+06  -1.20429195e+07]
 [ -1.59080571e-09  -7.54782306e+05   1.09058223e+07  -4.92744094e+08]
 [  2.90578196e+06   5.78203023e+06   7.26365451e+08  -5.17750740e+09]
 [ -3.65365612e+07  -8.91344633e+08   2.69540726e+09  -4.69333485e+11]]

moments_hu:
[  2.61468505e-01   2.38880787e-02   6.20119470e-03   7.62718335e-04
   7.22579752e-07   4.57095982e-05  -1.49310697e-06]

moments_normalized:
[[        nan         nan  0.07130303 -0.01260572]
 [        nan -0.04939584  0.01141548 -0.00824944]
 [ 0.19016548  0.00605224  0.01216069 -0.00138641]
 [-0.03824402 -0.01492274  0.00072176 -0.00201011]]

orientation:
0.34672206948219986

perimeter:
309.4142135623731

solidity:
0.7979179424372321

weighted_centroid:
(86.724481964696849, 336.99693016116652)

weighted_local_centroid:
(38.724481964696857, 54.996930161166539)

weighted_moments:
[[  3.90900000e+03   1.51374000e+05   6.95141000e+06   3.41529822e+08]
 [  2.14983000e+05   7.57032300e+06   3.34754925e+08   1.61617328e+10]
 [  1.47291870e+07   4.93140747e+08   2.13269647e+10   1.01863852e+12]
 [  1.09314168e+09   3.55450977e+10   1.51641012e+12   7.19392881e+13]]

weighted_moments_central:
[[  3.90900000e+03  -3.95871069e-09   1.08953027e+06  -1.20429195e+07]
 [ -1.59080571e-09  -7.54782306e+05   1.09058223e+07  -4.92744094e+08]
 [  2.90578196e+06   5.78203023e+06   7.26365451e+08  -5.17750740e+09]
 [ -3.65365612e+07  -8.91344633e+08   2.69540726e+09  -4.69333485e+11]]

weighted_moments_hu:
[  2.61468505e-01   2.38880787e-02   6.20119470e-03   7.62718335e-04
   7.22579752e-07   4.57095982e-05  -1.49310697e-06]
```

```
weighted_moments_normalized:
[[        nan         nan  0.07130303 -0.01260572]
 [        nan -0.04939584  0.01141548 -0.00824944]
 [ 0.19016548  0.00605224  0.01216069 -0.00138641]
 [-0.03824402 -0.01492274  0.00072176 -0.00201011]]


id:
5

H:
8.271036865792956

area:
3909

bbox:
(48, 168, 111, 262)

bbox_area:
120500

centroid:
(86.724481964696849, 206.00306983883345)

convex_area:
4899

convex_image:
[[ True  True  True ..., False False False]
 [ True  True  True ..., False False False]
 [ True  True  True ..., False False False]
 ...,
 [ True  True  True ...,  True  True  True]
 [ True  True  True ...,  True  True  True]
 [ True  True  True ...,  True  True  True]]

coords:
[[ 48 168]
 [ 48 169]
 [ 48 170]
 ...,
 [110 259]
 [110 260]
 [110 261]]

eccentricity:
0.8619855117512796
```

```
equivalent_diameter:
70.54851791759874

euler_number:
1

extent:
0.660081053698075

filled_area:
3909

filled_image:
[[ True  True  True ..., False False False]
 [ True  True  True ..., False False False]
 [ True  True  True ..., False False False]
 ...,
 [ True  True  True ...,  True  True  True]
 [ True  True  True ...,  True  True  True]
 [ True  True  True ...,  True  True  True]]

image:
[[ True  True  True ..., False False False]
 [ True  True  True ..., False False False]
 [ True  True  True ..., False False False]
 ...,
 [ True  True  True ...,  True  True  True]
 [ True  True  True ...,  True  True  True]
 [ True  True  True ...,  True  True  True]]

inertia_tensor:
[[ 743.35685934 -193.0883362 ]
 [-193.0883362   278.72352701]]

inertia_tensor_eigvals:
(813.12319328540207, 208.95719306351708)

intensity_image:
[[1 1 1 ..., 0 0 0]
 [1 1 1 ..., 0 0 0]
 [1 1 1 ..., 0 0 0]
 ...,
 [1 1 1 ..., 1 1 1]
 [1 1 1 ..., 1 1 1]
 [1 1 1 ..., 1 1 1]]

local_centroid:
```

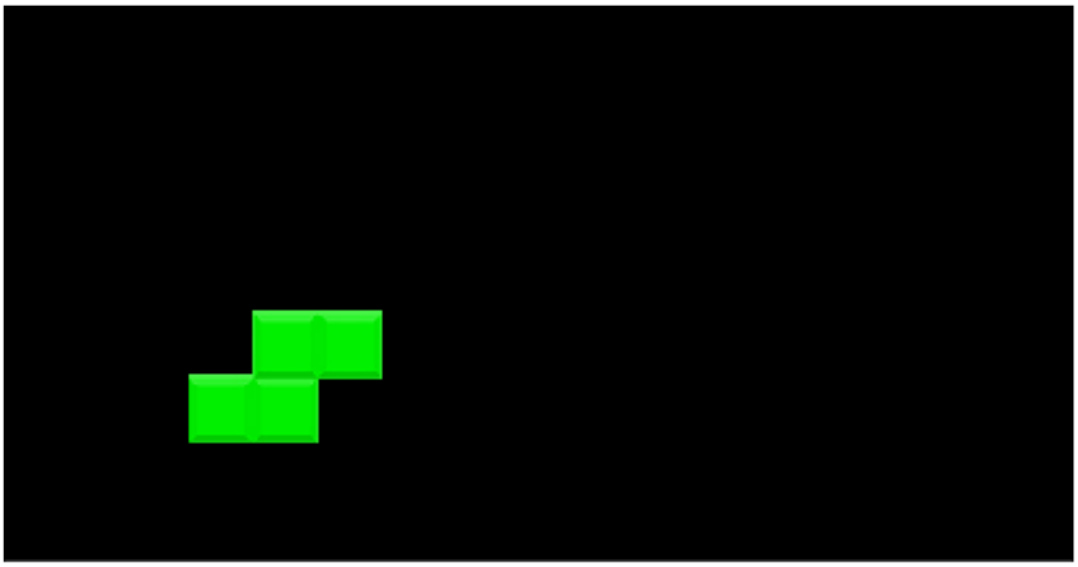
```
(38.724481964696857, 38.003069838833461)

major_axis_length:
114.06126026204704

max_intensity:
1

mean_intensity:
1.0

min_intensity:
1

minor_axis_length:
57.82140684051429

moments:
[[  3.90900000e+03   1.51374000e+05   6.95141000e+06   3.41529822e+08]
 [  1.48554000e+05   6.50745900e+06   3.11726205e+08   1.56005406e+10]
 [  8.55129000e+06   3.94294395e+08   1.91852938e+10   9.66447649e+11]
 [  5.82369102e+08   2.73727364e+10   1.33934229e+12   6.76242814e+13]]

moments_central:
[[  3.90900000e+03  -3.95871069e-09   1.08953027e+06  -1.20429195e+07]
 [  1.72254033e-09   7.54782306e+05  -1.09058223e+07   4.92744094e+08]
 [  2.90578196e+06   5.78203023e+06   7.26365451e+08  -5.17750740e+09]
 [  3.65365612e+07   8.91344633e+08  -2.69540726e+09   4.69333485e+11]]

moments_hu:
[  2.61468505e-01   2.38880787e-02   6.20119470e-03   7.62718335e-04
   7.22579752e-07   4.57095982e-05   1.49310697e-06]

moments_normalized:
[[        nan         nan  0.07130303 -0.01260572]
 [        nan  0.04939584 -0.01141548  0.00824944]
 [ 0.19016548  0.00605224  0.01216069 -0.00138641]
 [ 0.03824402  0.01492274 -0.00072176  0.00201011]]

orientation:
-0.3467220694821998

perimeter:
309.4142135623731

solidity:
0.7979179424372321
```

```
weighted_centroid:
(86.724481964696849, 206.00306983883345)

weighted_local_centroid:
(38.724481964696857, 38.003069838833461)

weighted_moments:
[[  3.90900000e+03   1.51374000e+05   6.95141000e+06   3.41529822e+08]
 [  1.48554000e+05   6.50745900e+06   3.11726205e+08   1.56005406e+10]
 [  8.55129000e+06   3.94294395e+08   1.91852938e+10   9.66447649e+11]
 [  5.82369102e+08   2.73727364e+10   1.33934229e+12   6.76242814e+13]]

weighted_moments_central:
[[  3.90900000e+03  -3.95871069e-09   1.08953027e+06  -1.20429195e+07]
 [  1.72254033e-09   7.54782306e+05  -1.09058223e+07   4.92744094e+08]
 [  2.90578196e+06   5.78203023e+06   7.26365451e+08  -5.17750740e+09]
 [  3.65365612e+07   8.91344633e+08  -2.69540726e+09   4.69333485e+11]]

weighted_moments_hu:
[  2.61468505e-01   2.38880787e-02   6.20119470e-03   7.62718335e-04
   7.22579752e-07   4.57095982e-05   1.49310697e-06]

weighted_moments_normalized:
[[        nan         nan  0.07130303 -0.01260572]
 [        nan  0.04939584 -0.01141548  0.00824944]
 [ 0.19016548  0.00605224  0.01216069 -0.00138641]
 [ 0.03824402  0.01492274 -0.00072176  0.00201011]]
```

In [14]: `exibir_imagens(objetos)`

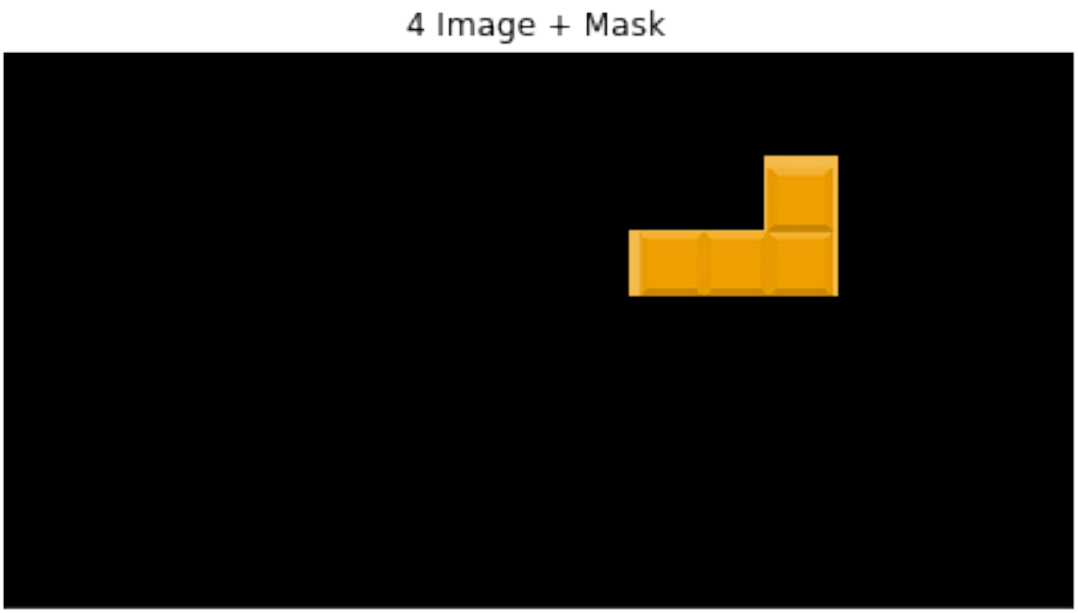
0 Image + Mask

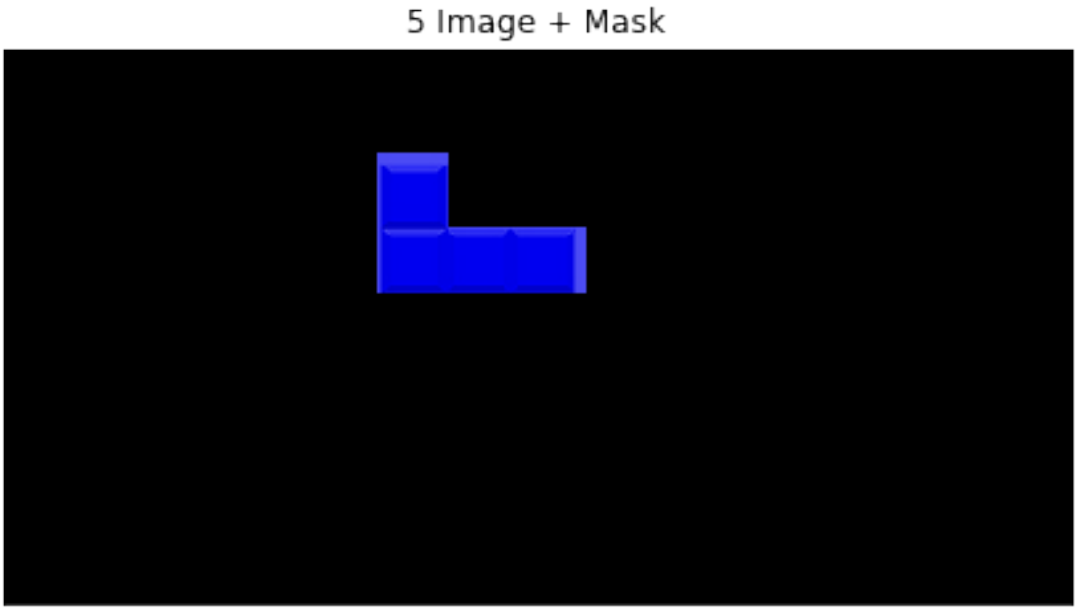
1 Image + Mask



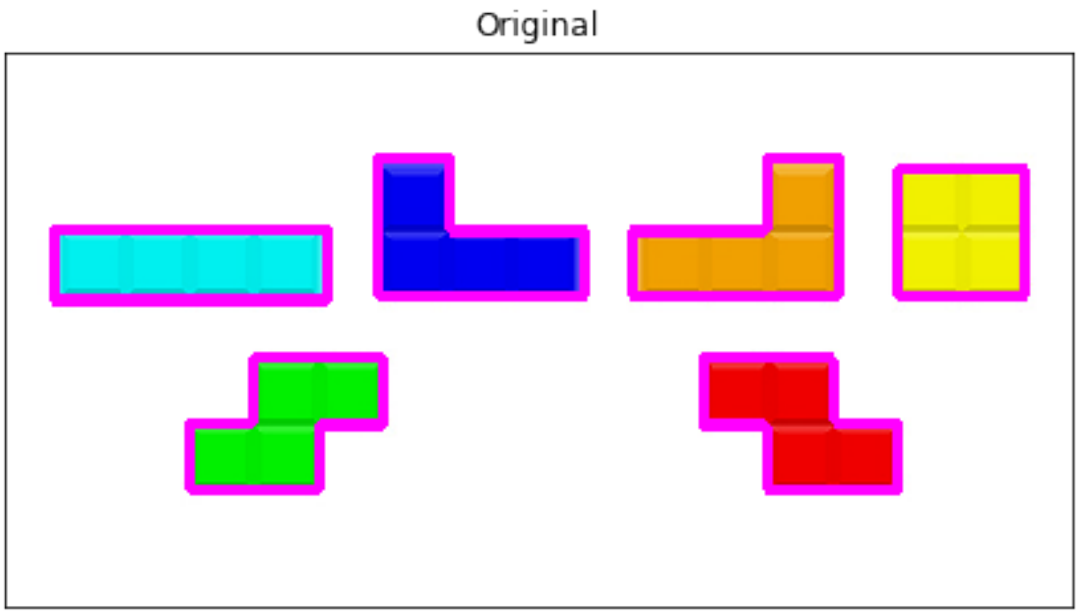
2 Image + Mask

3 Image + Mask






```
In [15]: exibir_imagens(contours[0])
```



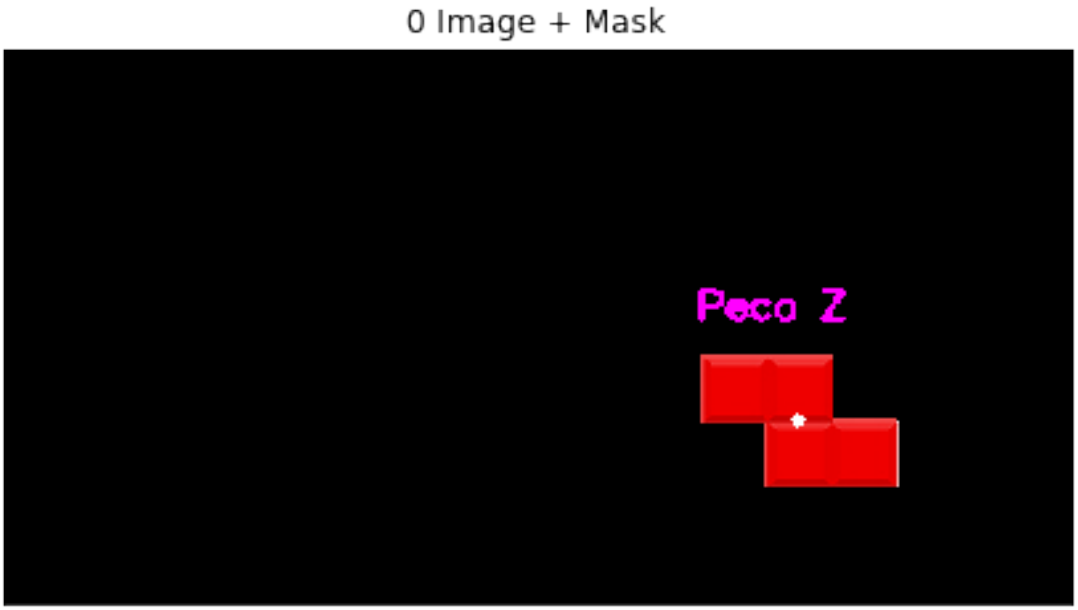

```
In [114]: text_objects = []
          for img_contour in contours:
              if img_contour.get('cnts'):
                  cnts = img_contour.get('cnts').copy()
                  for i, c in enumerate(cnts):
                      img = img_contour.copy()

                      clone = img.get('origin').copy()

                      mask = np.zeros(img.get('X').shape, dtype='uint8')
                      cv2.drawContours(mask, [c], -1, 255, -1)

                      clone_gray = img.get('gray').copy()

                      mask_gray = np.zeros(img.get('X').shape, dtype='uint8')
                      cv2.drawContours(mask_gray, [c], -1, 255, -1)

                      it = objetos[i].copy()

                      gray = it.get('gray').copy()
                      gray[(gray != 0)] = 1 # binary
                      it.update({'X': gray})

                      sample_image = it.get('X')
                      label_img = measure.label(sample_image, neighbors=8) # neighbors 4 or 8
                      props = measure.regionprops(label_img,  intensity_image=sample_image)
```

```
                    area = cv2.contourArea(c)
                    perimeter = cv2.arcLength(c, True)
                    (x, y, w, h) = cv2.boundingRect(c)
                    eccentricity = props[0].eccentricity
                    orientation = props[0].orientation

                    M = cv2.moments(c)
                    cx = int(M["m10"]/M["m00"])
                    cy = int(M["m01"]/M["m00"])

                    print(
                        (
                            "Contorno #{}\narea: {:.2f}, perimetro: {:.2f}\n"
                            "eccentricity: {:.2f}\n"
                            "orientation: {:.2}\n"
                        ).format(i, area, perimeter, eccentricity, orientation)
                    )

                    result_X = cv2.bitwise_and(clone, clone, mask=mask)

                    cv2.circle(result_X, (cx,cy), 3, (255, 255,255), -1)

                    label_name = i

                    if eccentricity == 0:
                        label_name = 'Quardrado'
                    elif abs(orientation) == 0:
                        label_name = 'Retangulo'
                    elif abs(eccentricity) > 0.80 and 0.40 < abs(orientation) < 60:
                        label_name = 'Peca Z'
                    elif abs(eccentricity) > 0.80 and 0.0 < abs(orientation) < 40:
                        label_name = 'Peca L'

                    cv2.putText(result_X, "{}".format(label_name), (cx+(-w//2), cy+(-h*3//4))

                    img.update(
                        {
                            'title': str((i))+' Image + Mask',
                            'X': result_X,
                            'index': i,
                        }
                    )
                    text_objects.append(img)
```

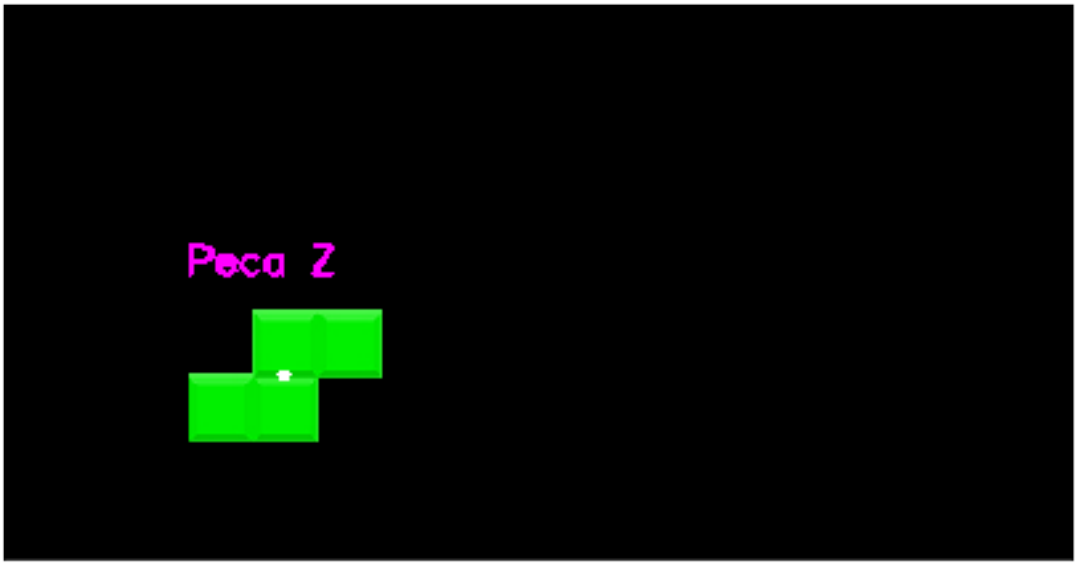
```
Contorno #0
area: 3510.00, perimetro: 291.66
eccentricity: 0.86
```

```
orientation: -0.55

Contorno #1
area: 3452.00, perimetro: 290.83
eccentricity: 0.86
orientation: 0.55

Contorno #2
area: 3936.00, perimetro: 310.00
eccentricity: 0.96
orientation: -0.0

Contorno #3
area: 3249.00, perimetro: 228.00
eccentricity: 0.00
orientation: 0.79

Contorno #4
area: 3753.50, perimetro: 309.41
eccentricity: 0.86
orientation: 0.35

Contorno #5
area: 3753.50, perimetro: 309.41
eccentricity: 0.86
orientation: -0.35
```

```
In [115]: exibir_imagens(text_objects)
```



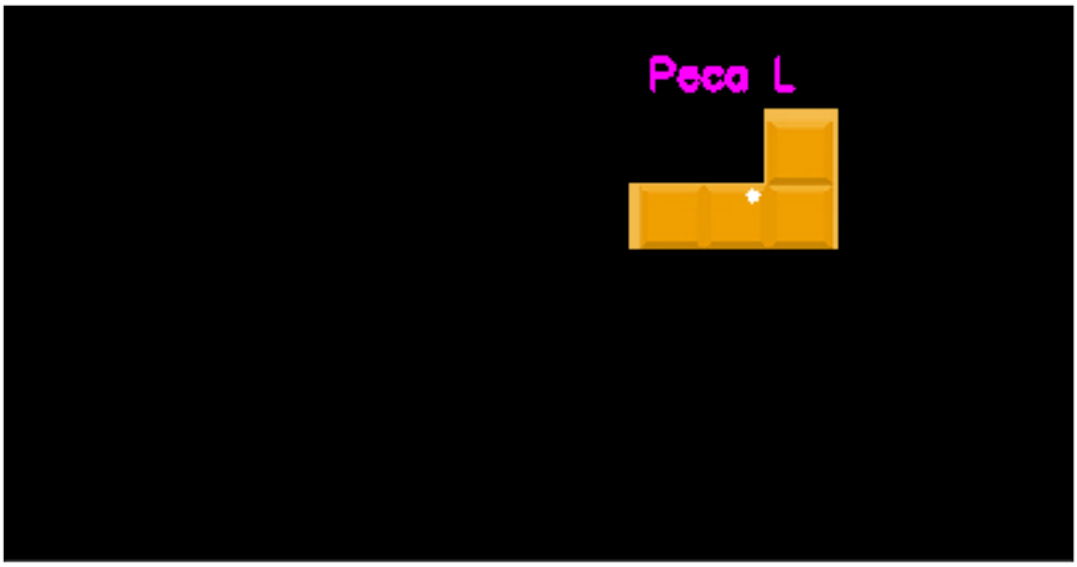
1 Image + Mask

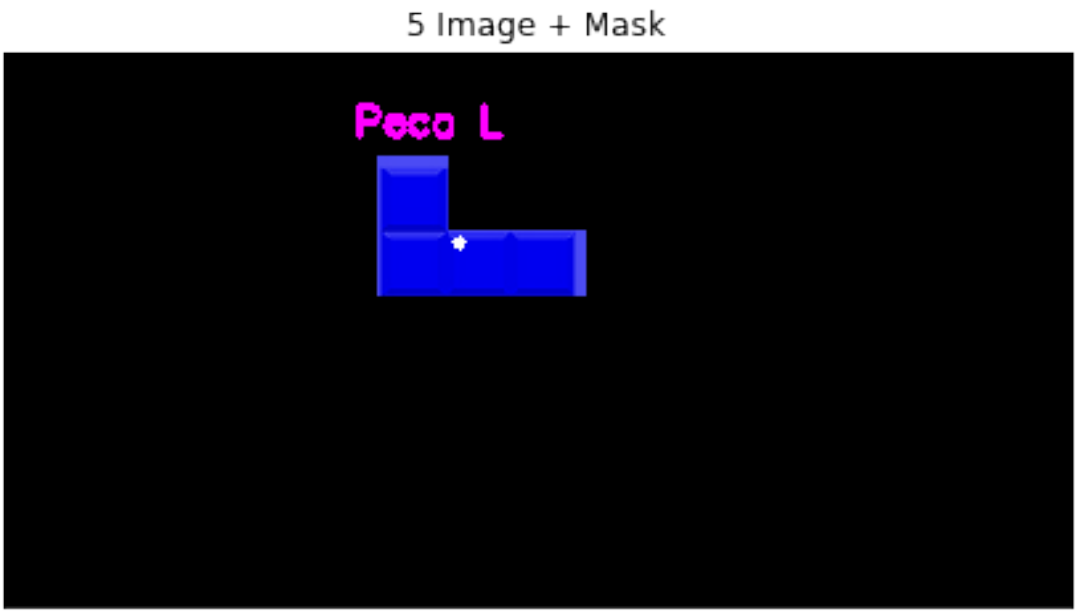


2 Image + Mask

3 Image + Mask

4 Image + Mask





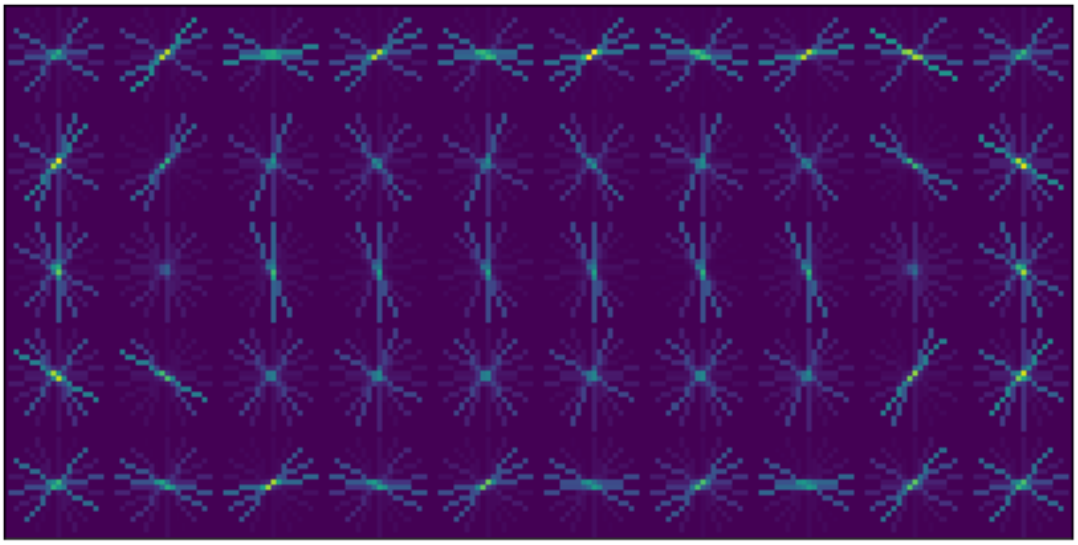
2 2 - Detectar logos de carro teste script

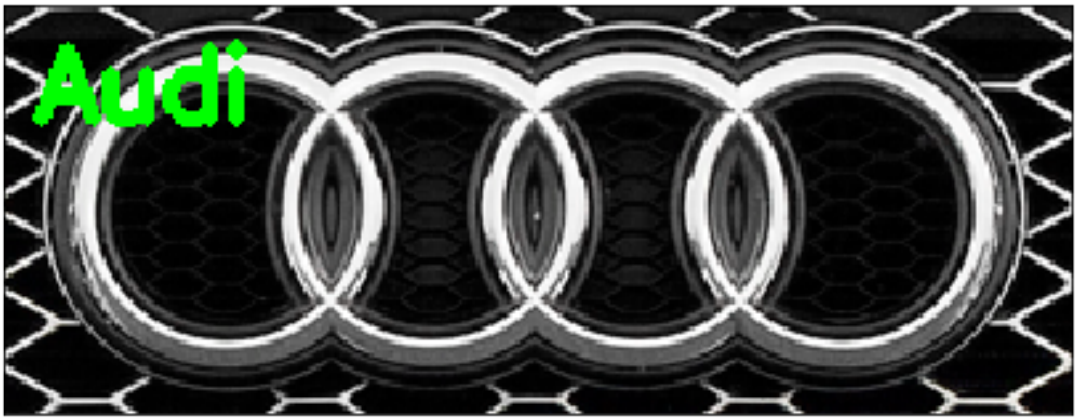
```
In [141]: def hog_detectar_texture(args):
              data = []
              labels = []

              for imagePath in paths.list_images(args["training"]):
                  # extract the make of the car
                  make = imagePath.split("/")[-2]

                  # load the image, convert it to grayscale, and detect edges or binarize

                  ######[CODE HERE]#######
                  img = cv2.imread(imagePath, 0)
                  gray = img
                  ######[END CODE]#######

                  # find contours in the edge map, keeping only the largest one which
                  # is presumed to be the car logo

                  #####[CODE HERE]#######
                  ret,thresh = cv2.threshold(gray,127,255,0)
                  im2,cnts,hierarchy = cv2.findContours(thresh, 1, 2)
                  c = max(cnts, key=cv2.contourArea)
                  #####[END CODE]#######
```

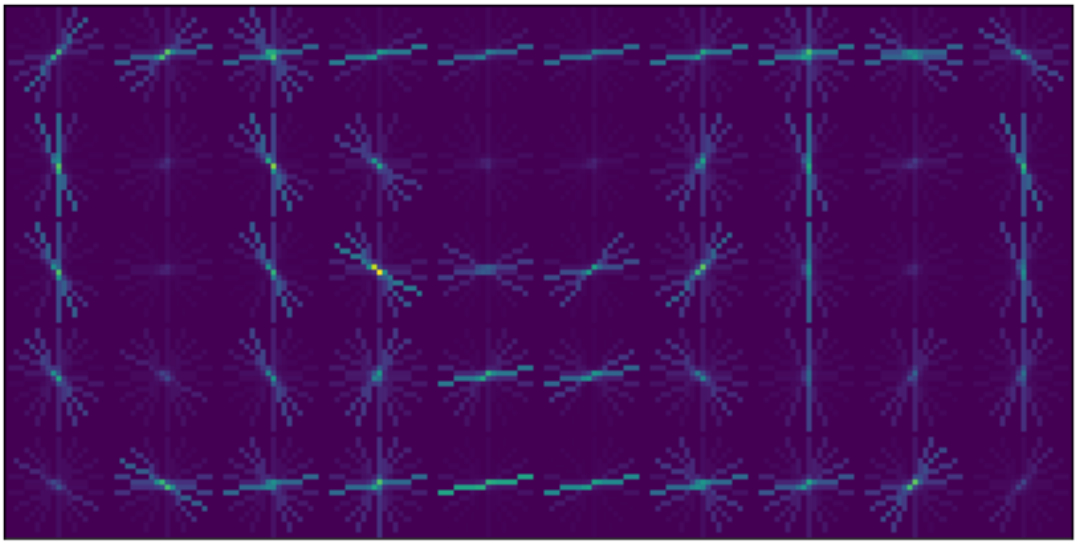
```
        # extract the logo of the car and resize it to a canonical width
        # and height
        (x, y, w, h) = cv2.boundingRect(c)
        logo = gray[y:y + h, x:x + w]
        logo = cv2.resize(logo, (200, 100))

        # extract Histogram of Oriented Gradients from the logo
        #####[CODE HERE]#######
        (H, hogImage) = feature.hog(logo, orientations=9, pixels_per_cell=(20, 20),
                                    cells_per_block=(2,2), transform_sqrt=True, visua

        #####[END CODE]#######
        # update the data and labels
        data.append(H)
        labels.append(make)

# "train" the nearest neighbors classifier
print("[INFO] training classifier...")
model = KNeighborsClassifier(n_neighbors=1)
model.fit(data, labels)
print("[INFO] evaluating...")

for imagePath in paths.list_images(args["test"]):
    # load the test image, convert it to grayscale, and resize it to
    # the canonical size
    image = cv2.imread(imagePath)
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    logo = cv2.resize(gray, (200, 100))

    # extract Histogram of Oriented Gradients from the test image and
    # predict the make of the car
    (H, hogImage) = feature.hog(logo, orientations=9, pixels_per_cell=(20, 20),
            cells_per_block=(2, 2), transform_sqrt=True, visualise=True)
    pred = model.predict(H.reshape(1, -1))[0]

    # visualize the HOG image
    hogImage = exposure.rescale_intensity(hogImage, out_range=(0, 255))
    hogImage = hogImage.astype("uint8")
    obj = {'X': hogImage}
    exibir_imagens(obj)

     # draw the prediction on the test image and display it
    cv2.putText(image, pred.title(), (10, 35), cv2.FONT_HERSHEY_SIMPLEX, 1.0, (0
    obj = {'X': image}
    exibir_imagens(obj)
```

```
In [142]: args = {
              'training': '../../db_aulas/Imagens/minibases/logos/car_logos',
              'test': '../../db_aulas/Imagens/minibases/logos/test_images',
          }

          hog_detectar_texture(args)
```

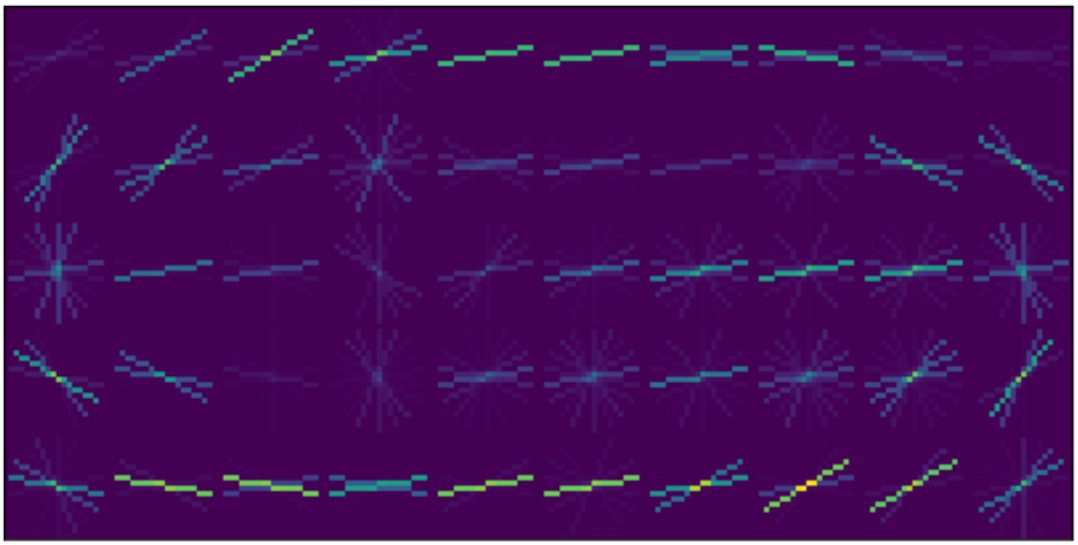
```
/home/iury/.pyenv/versions/3.6.2/envs/lab/lib/python3.6/site-packages/skimage/feature/_hog.py:
  'be changed to `L2-Hys` in v0.15', skimage_deprecation)


[INFO] training classifier...
[INFO] evaluating...
```



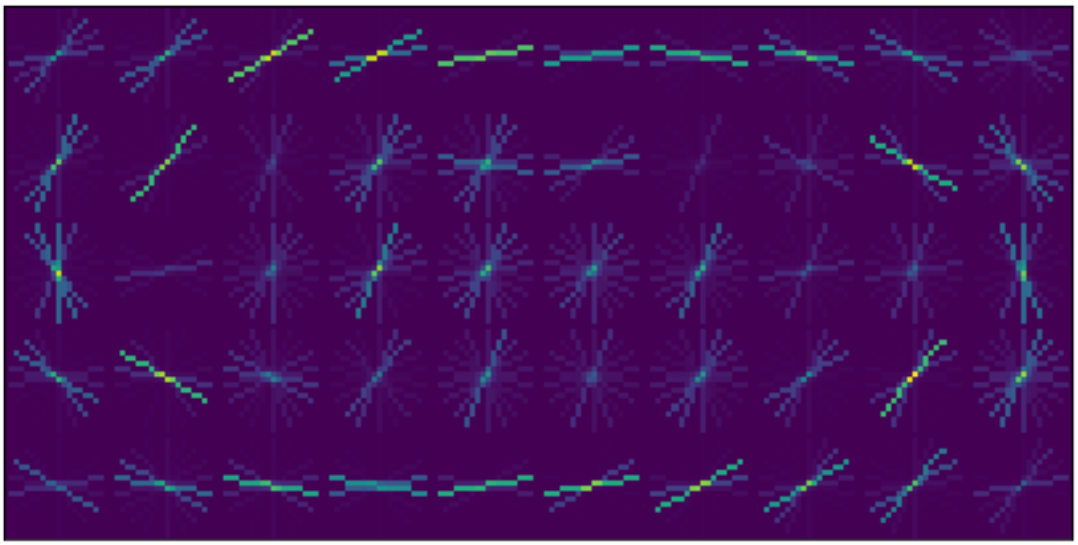




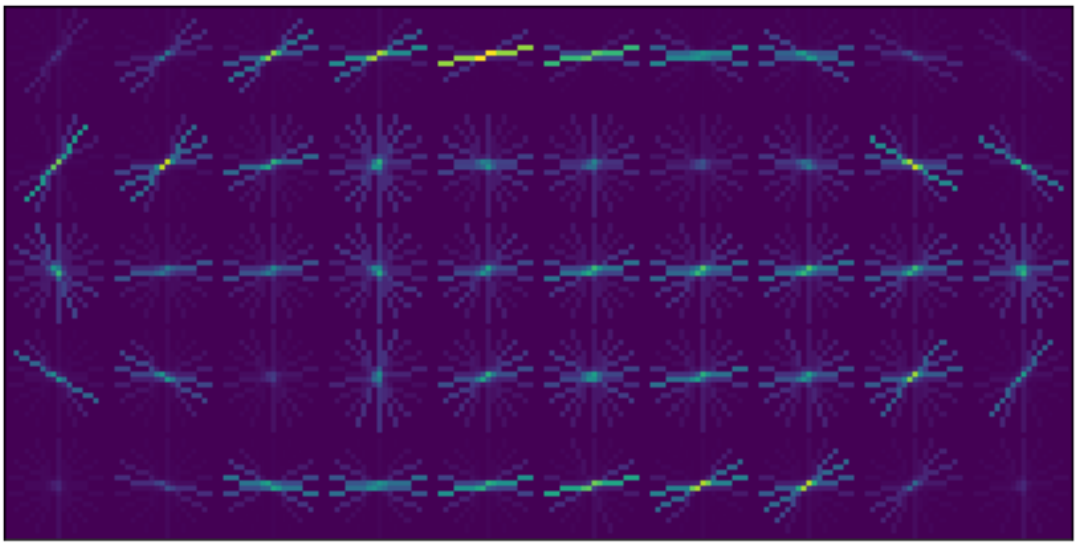

Audi



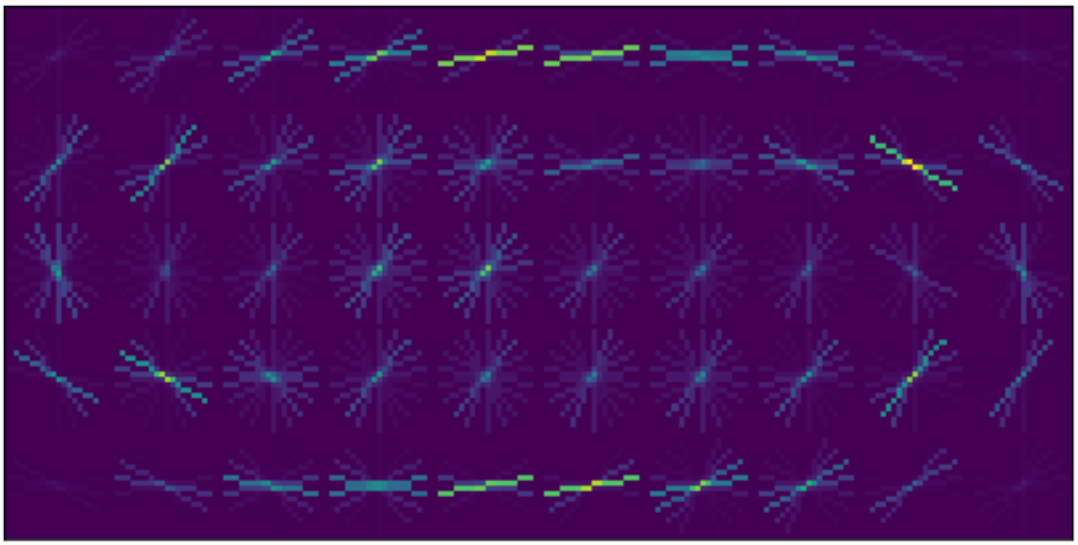

Subaru



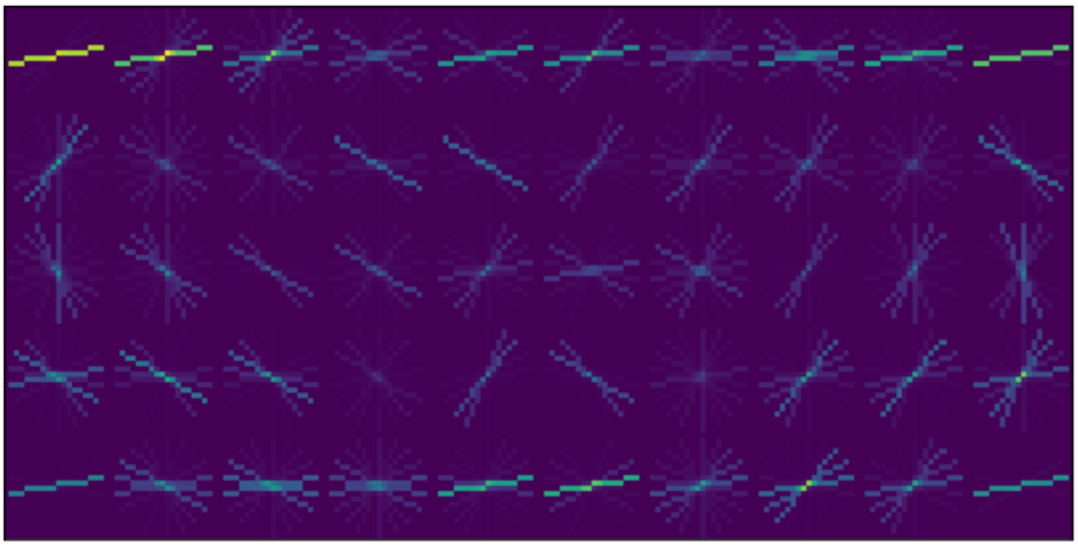

Ford
Ford




Subaru




Ford




```
In [167]: def lbp_detectar_texture(args):
              data = []
              labels = []
```

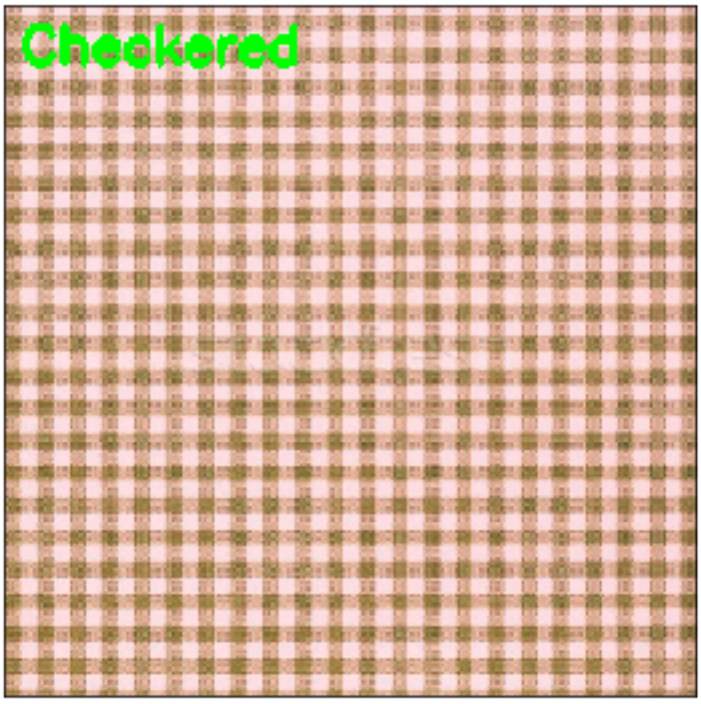
```
    for imagePath in paths.list_images(args["training"]):
        # extract the make of the car
        make = imagePath.split("/")[-2]

        # load the image, convert it to grayscale, and detect edges or binarize

        ######[CODE HERE]#######
        img = cv2.imread(imagePath, 0)
        gray = img
        ######[END CODE]#######

        # find contours in the edge map, keeping only the largest one which
        # is presumed to be the car logo

        #####[CODE HERE]#######
        ret,thresh = cv2.threshold(gray,127,255,0)
        im2,cnts,hierarchy = cv2.findContours(thresh, 1, 2)
        c = max(cnts, key=cv2.contourArea)
        #####[END CODE]#######


        # extract the logo of the car and resize it to a canonical width
        # and height
        (x, y, w, h) = cv2.boundingRect(c)
        logo = gray[y:y + h, x:x + w]
        logo = cv2.resize(logo, (200, 100))

        # extract Histogram of Oriented Gradients from the logo
        #####[CODE HERE]#######
#         (H, hogImage) = feature.hog(logo, orientations=9, pixels_per_cell=(20, 20)
#                                     cells_per_block=(2,2), transform_sqrt=True, vi

        numPoists = 50
        radius = 3

        lbp = feature.local_binary_pattern(logo, numPoists, radius, method="uniform"
        (H, _) = np.histogram(lbp.ravel(), bins=range(0, numPoists + 3), range=(0, nu

        eps = 1e-7
        H = H.astype("float")
        H /= (H.sum() + eps)

        #####[END CODE]#######
        # update the data and labels
        data.append(H)
        labels.append(make)
```

```
            # "train" the nearest neighbors classifier
            print("[INFO] training classifier...")
            model = KNeighborsClassifier(n_neighbors=1)
            model.fit(data, labels)
            print("[INFO] evaluating...")

            for imagePath in paths.list_images(args["test"]):
                # load the test image, convert it to grayscale, and resize it to
                # the canonical size
                image = cv2.imread(imagePath)
                gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
                logo = cv2.resize(gray, (200, 100))

                # extract Histogram of Oriented Gradients from the test image and
                # predict the make of the car
        #         (H, hogImage) = feature.hog(logo, orientations=9, pixels_per_cell=(20, 20)
        #                 cells_per_block=(2, 2), transform_sqrt=True, visualise=True)

                numPoists = 50
                radius = 3

                lbp = feature.local_binary_pattern(logo, numPoists, radius, method="uniform")
                (H, _) = np.histogram(lbp.ravel(), bins=range(0, numPoists + 3), range=(0, nu

                eps = 1e-7
                H = H.astype("float")
                H /= (H.sum() + eps)

                pred = model.predict(H.reshape(1, -1))[0]

                # visualize the HOG image
        #         hogImage = exposure.rescale_intensity(hogImage, out_range=(0, 255))
        #         hogImage = hogImage.astype("uint8")
        #         obj = {'X': hogImage}
        #         exibir_imagens(obj)

                 # draw the prediction on the test image and display it
                cv2.putText(image, pred.title(), (10, 35), cv2.FONT_HERSHEY_SIMPLEX, 1.0, (0
                obj = {'X': image}
                exibir_imagens(obj)
```

In [168]:
```
args = {
        'training': '../../db_aulas/Imagens/minibases/textures/train',
        'test': '../../db_aulas/Imagens/minibases/textures/test/',
    }

    lbp_detectar_texture(args)
```

```
[INFO] training classifier...
```

```
[INFO] evaluating...
```




Grass