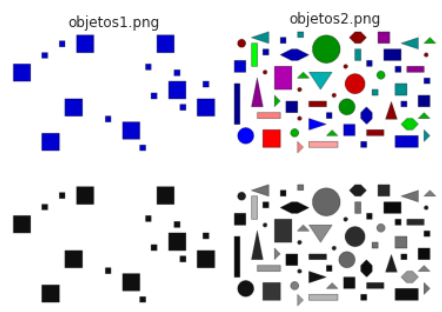
MO443 | Assignment 3 | 2021-07-26
Image Segmentation
Lucas David, 188972 | lucas.david@ic.unicamp.br

1 Introduction

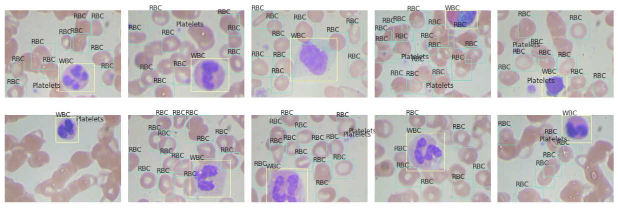
This assignment was submitted to the class of 2021/1 of course Introduction to Image Processing (MO443) at Universidade Estadual de Campinas. Its goal is to apply segmentation algorithms over images and extract characteristics of the objects using Python programming language and assess its results.

1.1 Dataset and Setup

I employed Scikit-Image and Google Colaboratory [1] to develop this assignment. The notebook produced is available for direct access[1]. All activities were answered using scikit-image implementations. To demonstrate the results, the algorithms were applied to three images containing simple geometric shapes — which were provided during class —, and additional random images from the BCCD[2] dataset. These two sets of images are illustrated in Fig. 4.



(a)



(b)

Figure 1: Examples of images (a) containing simple geometric shapes and (b) in the BCCD dataset, containing blood cells observed through a microscope.

[1]Iterative report available at colab/mo-443-assignment-3
[2]BCCD dataset is available at tensorflow.org/datasets/catalog/bccd

2 Borders Extraction

Scikit-image implements many of the border extraction methods available in the literature, such as Sobel, Prewitt and Scharr. The can be trivially used as described in List. 1. In my program, I sub-sampled a few of these methods and added to a dictionary of available border methods. The user can then select which method to use by passing the argument `--border`.

```
image = skimage.color.rgb2gray(image)
methods = [
  skimage.filters.sobel,
  skimage.filters.roberts,
  skimage.filters.prewitt,
  skimage.filters.farid,
  skimage.filters.scharr,
  partial(skimage.feature.canny, sigma=0.),
  partial(skimage.feature.canny, sigma=0.1),
  partial(skimage.feature.canny, sigma=3.),
]

segments = [m(image) for m in methods]
```

Listing 1: The application of the sobel .

Fig. 2 contains a comparison between the available border extraction methods. Roberts seems to produce the thinnest borders amongst all methods, while Farid produces thicker ones. Sobel, Prewitt and Scharr have similar results over samples that contain simple geometric shapes, but slightly differ when applied over samples from the BCCD dataset. Finally, Canny with a small σ parameter (used to set the standard deviation of the Gaussian filter) results in accurate borders for simple shapes. However, the original gray intensity of the object's border is lost, with all borders presenting the same gray intensity. For samples in the BCCD dataset, Canny is successful in removing small variations in the background, but fails to generate closed borders for objects with shade similar to the background's. Canny with a very large sigma parameter ($\sigma = 3$) still manages to extract borders from simple objects, though corners become round. On the other hand, this same method fails completely when applied over the cell images from the BCCD dataset.

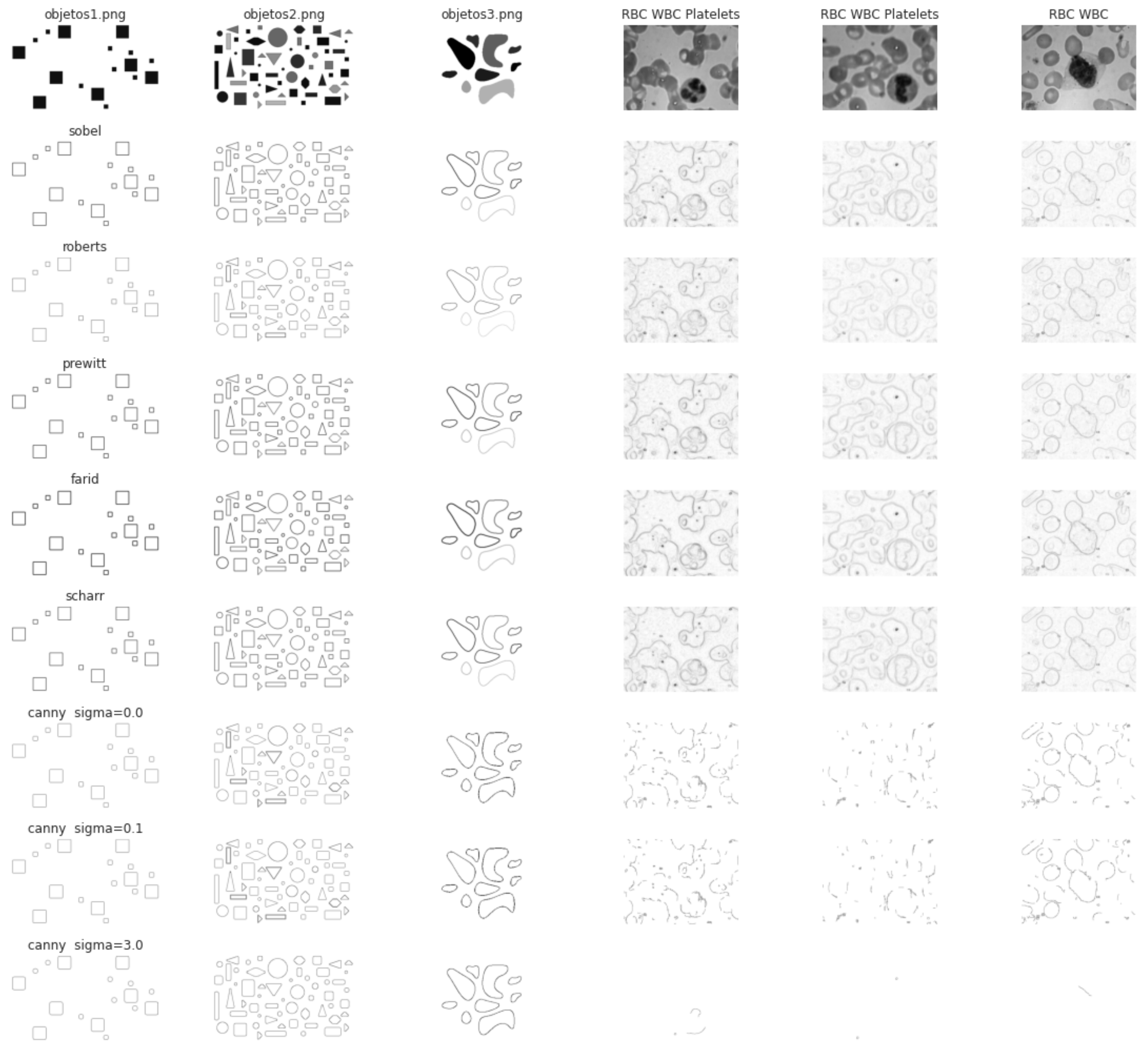


Figure 2: Comparison between multiple border extraction methods available in scikit-image library. From top to bottom: original images, Sobel, Roberts, Prewitt, Farid, Scharr, Canny $\sigma = 0.0$, Canny $\sigma = 0.1$ and Canny $\sigma = 3$.

3 Properties Extraction from Objects

In order to extract properties from the objects in an image, we must first segment each object. In this section, we first detailed our segmentation strategy, followed by the reporting of the objects properties.

3.1 Object Segmentation

Using scikit-image, multiple segmentation strategies are available. For instance, one can extract borders and label the connected regions; or find central regions and apply label expansion methods such as Watershed.

3.1.1 Felzenszwalb

I opted to employ the segmentation method described in "Efficient graph-based image segmentation", by Felzenszwalb et. al. [2]. This segmentation strategy consists of representing the pixel-color intensity in an image as a grid and find N partitions representing similarity. As the algorithm progresses, the closest partitions (with respect to a connection predicate) are iteratively merged. The final number of partitions is minimum (optimal) while still respecting the connection predicate.
Lst. 2 exemplifies the application of Felzenszwalb's segmentation method implemented in scikit-image over images.

```
segments = [
  skimage.segmentation.felzenszwalb(
    g,
    scale=1e6,
    sigma=0.1,
    min_size=10
  )
  for g in images
]
```

Listing 2: The application of the Felzenszwalb's segmentation method over images using scikit-image.

The second column in Fig. 3 illustrates the Felzenszwalb's segmentation results over images containing simple geometric shapes. I used the parameters `scale=1e6, sigma=0.1, min_size=10` to detect the simple geometric shapes, which enforces large objects (high `scale`) and low Gaussian filtering (low `sigma`). Decreasing `scale` had little influence in the results of the last sample (containing 9 objects). However, this resulted in an incorrect segmentation for the objects in the first and second images, in which the objects and their borders were being classified as distinct coinciding objects. I found `min_size` to be of little importance in these examples, as it is applied as a post-processing stage and can be disregarded when `scale` is sufficiently large.
The segmentation results of samples in the BCCD dataset are shown in Fig. 4a. The method has correctly segmented many of the blood cells in some samples (such as in the third and fifth images). However, a few drawbacks are noticeable as well: this method is strongly affected by small grains, and will often recognize small microorganisms that were captured by the microscope while photographing the blood cells. Furthermore, cells that were smashed together seem to have been classified as a single object (first and eighth images). Finally, Felzenszwalb's method will indistinguishably segment the background sections of the image into regions, as these sections also present color information. It is therefore necessary to employ heuristics that discriminate foreground/background in order to separate it.

3.1.2 Morphological Segmentation

Going in a different direction, morphology can also be used to segment objects in images. I developed a second segmentation strategy, which consists of applying the Ostu's threshold method [3] to separate objects from background and employ morphology to close small holes. The cleaned mask can then be labeled according to its connected regions. Lst. 3 describes the necessary steps to implement this strategy. Notice that the binary mask that will serve as input to the morphological operations is created by retaining the positions in which pixel intensity is below the threshold (`image < t`), as the objects in this problem present a lower pixel intensity than the background's. This differs from the examples in scikit-image documentation, in which silver coins were being compared to a dark background.

```python
from skimage.measure import label
from skimage.filters import threshold_otsu
from skimage.morphology import opening,
     closing, square

def morphological_segmentation(
     image,
     so=square(5)):
   bw = image < threshold_otsu(image)

   return label(opening(closing(bw, so), so))
```

Listing 3: Image segmentation using Otsu's thresholding and morphology.

The segmentation results over samples in the BCCD dataset are shown in Fig. 4b. Through inspection, we notice this strategy is robust against small grains in the image, and correctly segments red blood cells presenting light-shaded interiors. It also automatically ignores the background and does not interpret its regions as new objects. Notwithstanding, long cell chains (which are smashed against each other) are incorrectly segmented as a single object, regardless of their color differences (an effect observed in the first, second, third, fifth, ninth and tenth images).

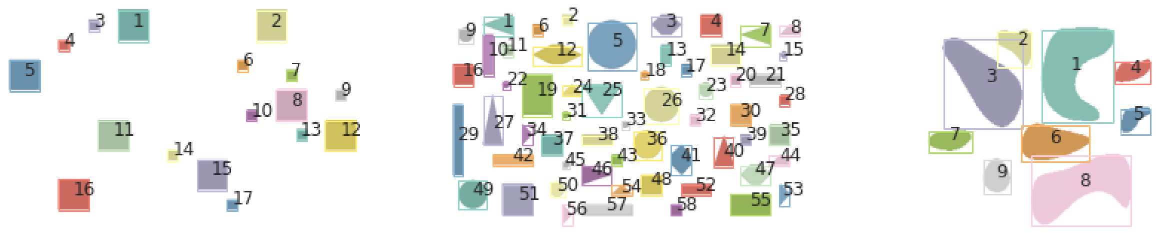
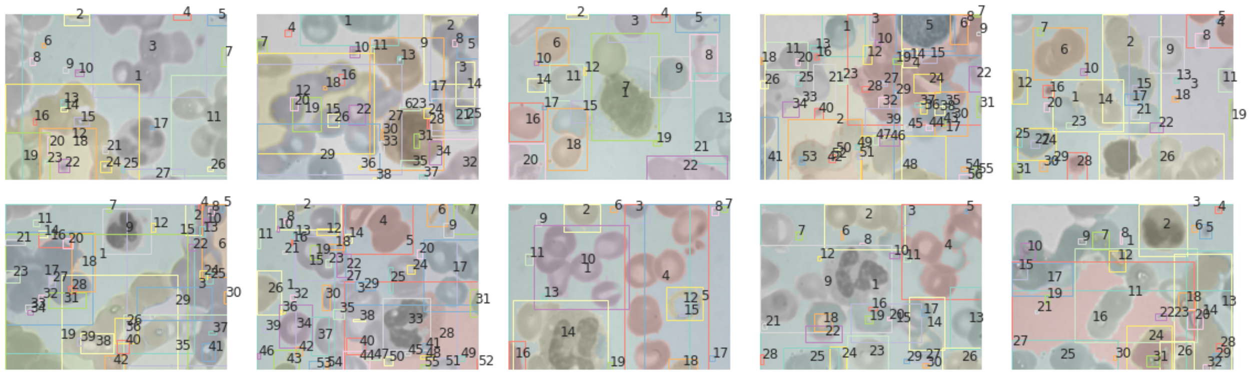
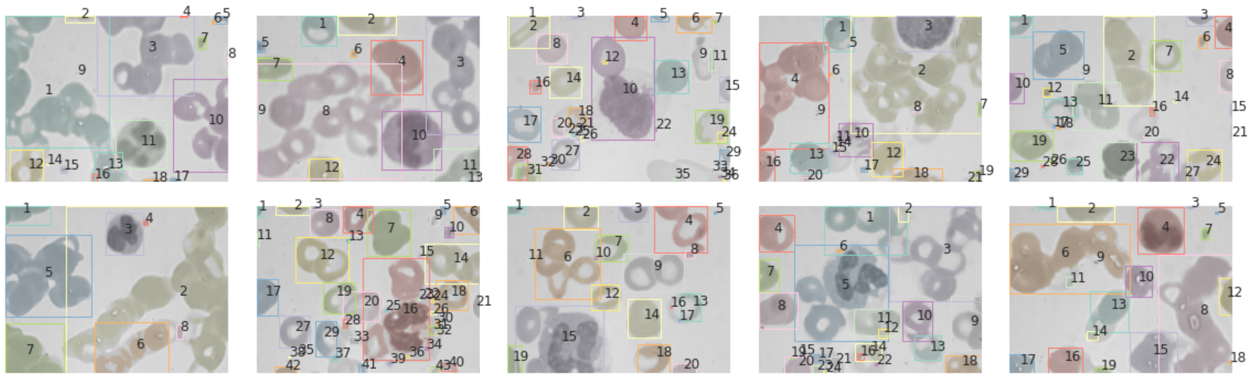


Figure 3: Results of Felzenszwalb's segmentation method, applied over simple shapes. Parameters used were `scale=1e6, sigma=0.1, min_size=10`.



(a) Results of Felzenszwalb's segmentation method, applied over samples in the BCCD dataset. Parameters used were `scale=1e4, sigma=0.01, min_size=75`.



(b) Results of the segmentation of samples in the BCCD dataset, using Otsu Thresholding, Morphology operations and labeling of connected regions. The structuring object used is a 5×5 matrix of 1's.

Figure 4: Segmentation results using (a) Felzenszwalb's segmentation method and (b) Morphological operators.

3.2 Object Properties

Once segmentation maks are available, properties can be trivially extracted using the `regionprops_table` function. Tables 1, 2 and 3 list the properties of each object in each one of the image containing simple geometric shapes.
Finally, objects can be counted considering their overall area. The histograms in Fig. 6 present the object count for each one of the images in the BCCD dataset.

label	area	convex_area	eccentricity	solidity	perimeter	centroid-0	centroid-1
1	2352	2352	0.20	1.0	190.0	29	202
2	2352	2352	0.20	1.0	190.0	29	422
3	272	272	0.34	1.0	62.0	29	139
4	272	272	0.34	1.0	62.0	60	92
5	2304	2304	0.00	1.0	188.0	107	29
6	272	272	0.34	1.0	62.0	92	375
7	289	289	0.00	1.0	64.0	108	454
8	2352	2352	0.20	1.0	190.0	155	454
9	289	289	0.00	1.0	64.0	139	533
10	289	289	0.00	1.0	64.0	171	391
11	2352	2352	0.20	1.0	190.0	202	171
12	2352	2352	0.20	1.0	190.0	202	533
13	289	289	0.00	1.0	64.0	202	470
14	272	272	0.34	1.0	62.0	234	265
15	2304	2304	0.00	1.0	188.0	265	328
16	2352	2352	0.20	1.0	190.0	296	108
17	272	272	0.34	1.0	62.0	312	360

Table 1: Properties of objects in image 1.

label	area	convex_area	eccentricity	solidity	perimeter	centroid-0	centroid-1
1	816	846	0.82	0.96	138.01	21	84
2	272	272	0.34	1.00	62.00	13	187
3	1024	1024	0.63	1.00	118.85	21	344
4	1056	1056	0.24	1.00	126.00	21	415
5	4616	4668	0.07	0.99	251.28	52	257
6	272	272	0.34	1.00	62.00	29	139
7	785	823	0.82	0.95	136.22	37	493
8	289	289	0.82	1.00	77.25	31	541
9	419	429	0.12	0.98	73.94	37	25
10	1088	1088	0.96	1.00	158.00	68	61
11	289	289	0.00	1.00	64.00	61	92
12	1552	1582	0.89	0.98	181.44	68	170
13	544	544	0.85	1.00	94.00	68	344
14	1536	1536	0.75	1.00	156.00	68	438
15	113	115	0.15	0.98	36.97	68	530
16	1056	1056	0.24	1.00	126.00	100	21
17	272	272	0.34	1.00	62.00	92	375
18	113	113	0.29	1.00	36.38	100	310
19	3072	3072	0.66	1.00	220.00	131	139
20	289	289	0.00	1.00	64.00	108	454
21	816	816	0.94	1.00	126.00	108	501
22	111	116	0.23	0.96	37.80	115	89
23	417	429	0.04	0.97	73.94	123	407
24	304	304	0.81	1.00	76.43	126	194
25	1568	1598	0.50	0.98	183.10	131	241
26	2402	2441	0.06	0.98	181.34	147	336
27	1328	1388	0.94	0.96	200.01	183	68
28	272	272	0.34	1.00	62.00	139	533
29	1792	1792	0.99	1.00	252.00	202	13
30	1089	1089	0.00	1.00	128.00	163	462
31	113	113	0.29	1.00	36.38	163	184
32	289	289	0.00	1.00	64.00	171	391
33	112	116	0.26	0.97	37.80	178	278
34	272	272	0.82	1.00	74.43	194	121
35	1056	1056	0.24	1.00	126.00	194	525
36	1583	1620	0.12	0.98	147.54	210	313
37	1056	1056	0.24	1.00	126.00	210	163
38	784	784	0.95	1.00	126.00	202	234
39	272	272	0.34	1.00	62.00	202	470
40	816	846	0.82	0.96	138.01	228	435
41	1039	1039	0.64	1.00	119.44	234	367
42	1088	1088	0.96	1.00	158.00	234	100
43	272	272	0.34	1.00	62.00	234	265
44	289	289	0.82	1.00	77.25	236	525
45	112	116	0.28	0.97	36.97	241	184
46	815	844	0.82	0.97	137.43	257	225
47	1040	1040	0.64	1.00	120.27	257	485
48	1024	1024	0.00	1.00	124.00	273	320
49	1576	1607	0.10	0.98	146.71	289	37
50	419	428	0.13	0.98	73.94	281	170
51	2401	2401	0.00	1.00	192.00	297	108
52	816	816	0.94	1.00	126.00	281	391
53	289	289	0.82	1.00	77.25	289	530
54	272	272	0.82	1.00	74.43	284	273
55	2112	2112	0.86	1.00	190.00	305	477
56	272	272	0.82	1.00	74.43	320	184
57	1360	1360	0.98	1.00	190.00	313	249
58	289	289	0.00	1.00	64.00	313	360

Table 2: Properties of objects in image 2.

label	area	convex_area	eccentricity	solidity	perimeter	centroid-0	centroid-1
1	3969	5308	0.82	0.75	313.76	68	147
2	791	880	0.74	0.90	119.98	43	94
3	3584	3665	0.90	0.98	259.46	80	63
4	540	593	0.89	0.91	99.25	70	206
5	438	478	0.86	0.92	88.77	116	208
6	1684	1732	0.87	0.97	174.12	140	126
7	642	662	0.89	0.97	103.01	137	27
8	3934	5081	0.91	0.77	305.42	182	156
9	675	691	0.62	0.98	96.33	173	73

Table 3: Properties of objects in image 3.

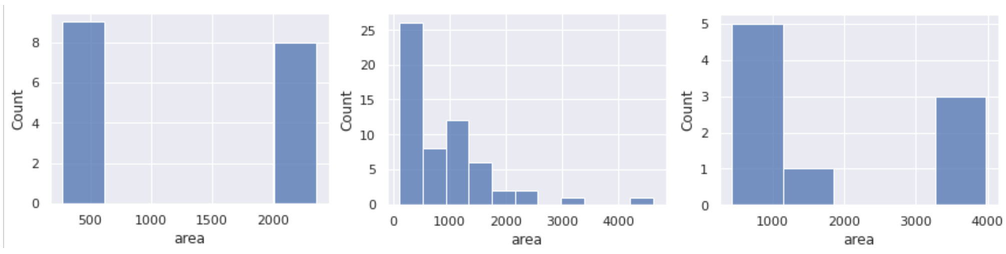
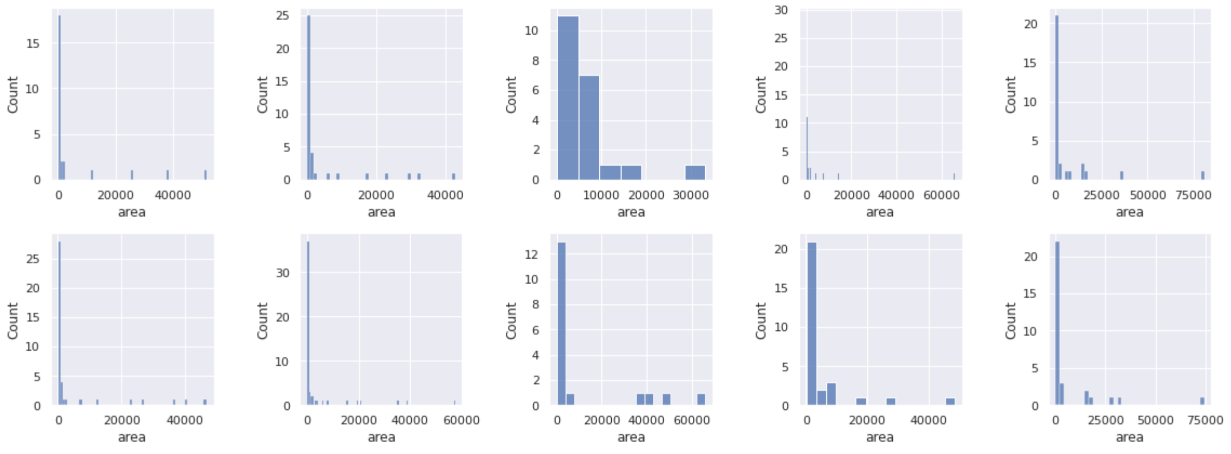
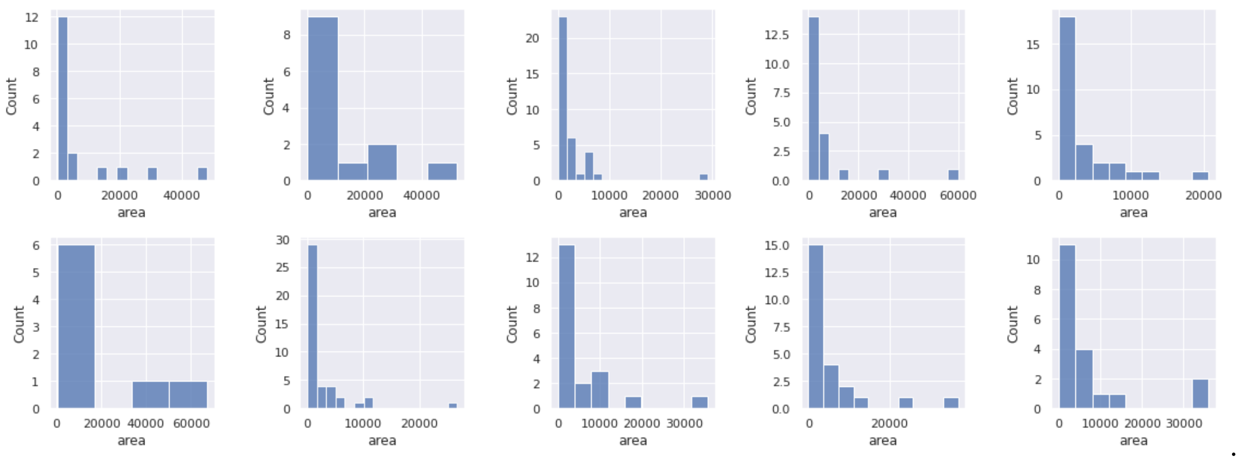


Figure 5: Histogram of object areas for each one of the images containing simple geometric shapes. Objects were labeled through the Felzenszwalb's segmentation method.



(a)



(b)

Figure 6: Histogram of object areas for each one of the images in the BCCD dataset. (a) Objects segmented with Felzenszwalb's method; and (b) objects segmented with morphological operators.

References

[1] Tiago Carneiro et al. “Performance analysis of google colaboratory as a tool for accelerating deep learning applications”. In: *IEEE Access* 6 (2018), pp. 61677–61685.

[2] Pedro F Felzenszwalb and Daniel P Huttenlocher. “Efficient graph-based image segmentation”. In: *International journal of computer vision* 59.2 (2004), pp. 167–181.

[3] Nobuyuki Otsu. “A Threshold Selection Method from Gray-Level Histograms”. In: *IEEE Transactions on Systems, Man, and Cybernetics* 9.1 (1979), pp. 62–66. DOI: 10.1109/TSMC.1979.4310076.