Count certain type of Lego blocks in the given image

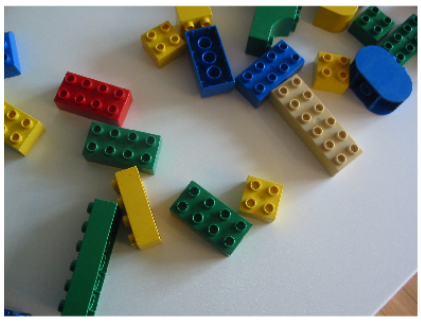
Haiwen Xiao

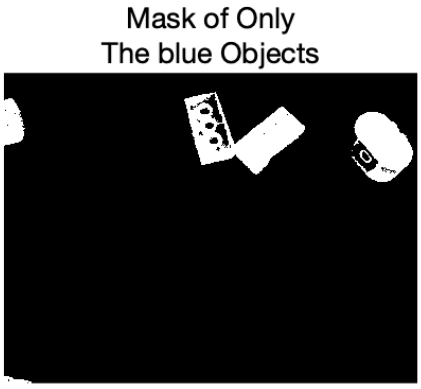
Abstract

The general idea of counting the 2*2 and 2*4 blocks in the image are based on two metrics: the number of circles in one block area and the Length to width ratio based on the Feret Properties of the given area.

Step 1: Segment the hsv-converted image

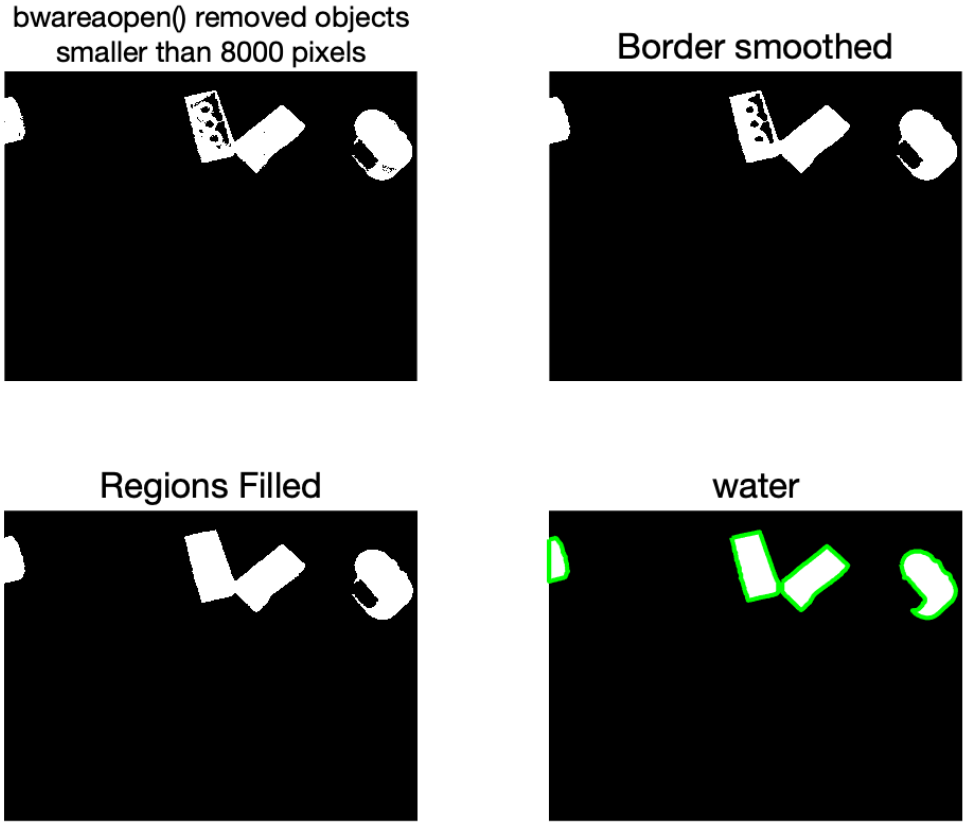
The first step is to create a mask which only cover the red/blue region in one area. The image was converted to HSV space which facilitates the color segmentation. The mask was generated automatically by the MATLAB's built-in colour thresholder app. *https://ww2.mathworks.cn/help/images/image-segmentation-using-the-color-thesholder-app.html*





Step 2: Process the mask

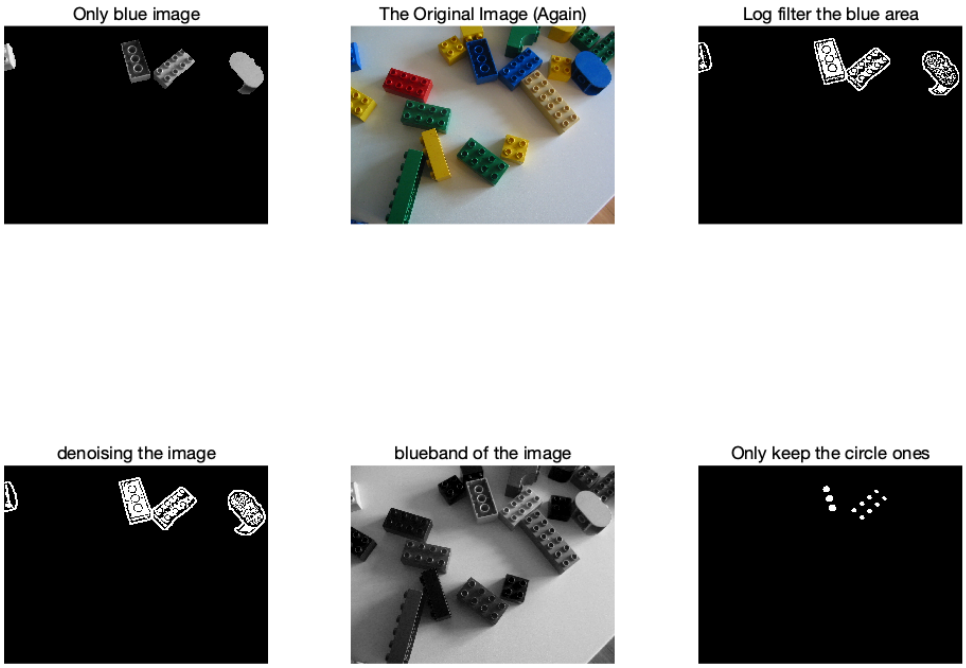
The idea is generally based on *https://ww2.mathworks.cn/matlabcentral/fileexchange/26420-simplecolordetection*. First remove those small connected areas(less than 8000 pixels here) which are most likely to be the noise in the image. Then use *imclose* to smooth the border of each area. Thirdly use *imfill* to fill the small holes in each area which makes the image more integrated. Notice that here we add a white border on one side of the image each time, and then fill the holes, this could keep the information of objects lying on the border more completed.
Then we use the watershed method to segment the regions which are joined by small bridges. It works well when the contact area is not very large. *https://uk.mathworks.com/company/newsletters/articles/the-watershed-transform-strategies-for-image-segmentation.html*



Step 3: Detect the circle area in Red/Blue block

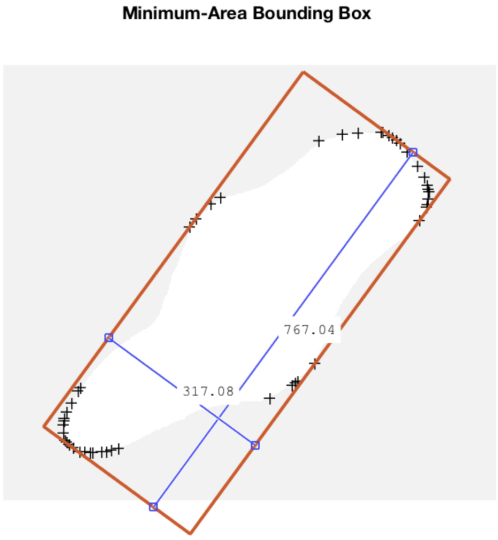
First we extract the blue/red band of the original image and we point multiply the blue/red band image with the mask created in last step. It shows a greyscale image which keeps the information we want. Then we use a log mask to filter the image which highlights the border of the greyscale image. Then we do a smoothing to the result image and use *bwboundaries* and *regionprops* to calculate the perimeter and area of each connected region. We set a metric composed of perimeter and area of each region. When the metric is greater than the

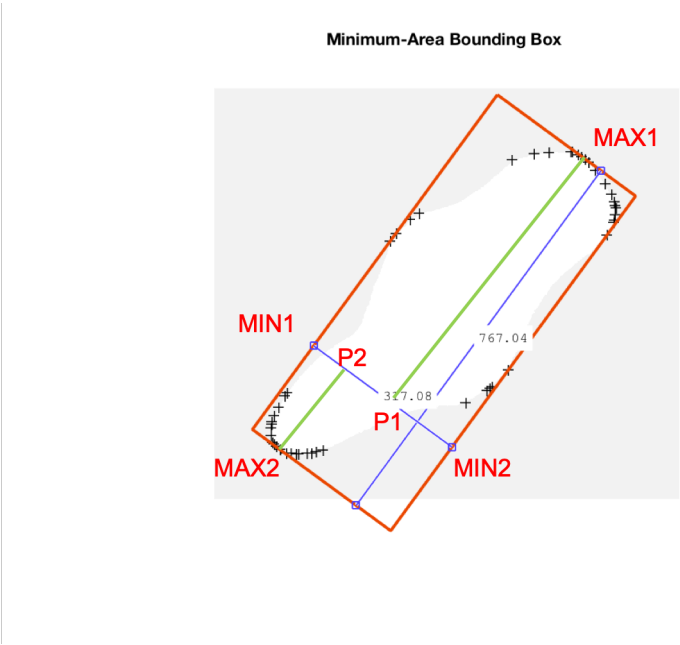
threshold, we keep the area. Then we get the circle region of the certain blocks. *https://uk.mathworks.com/matlabcentral/answers/16033-detect-rounding-objects-only-and-remove-all-other-objects. https://uk.mathworks.com/help/images/examples/identifying-round-objects.html*



Step 5: Compute the Length to width ratio of each region based on Feret Properties

The best measurement of the length to width ratio of the detected blocks is to find the length and width of Minimum-area Bounding Box of it. So we use the feret properties to compute that. *regionprops* could give us the coordinates of the end points of the minimum and maximum Feret Diameter. We make the projection of the max diameter endpoints on the line segment of the min diameter and then sum the two distance, which forms the length of the bounding box. The width is the distance between the minimum diameter endpoints. *https://blogs.mathworks.com/steve/2018/04/17/feret-properties-wrapping-up/*





Step 6: Count the qualified Lego blocks

Label each region, compute the Feret L-to-W ratio of the region and count the number of circles in that region. Through testing the training images, we can summary the metric for the two type of blocks and use this metric to count the number of blocks.

Length-to width ratio of the detected blocks in the training images

imageno	2*2 with circle(num)	2*2 without circle	2*4 with circle(num)	2*4 without circle	fakeones
1		1.56	1.688(3)		
2	1.15			2.14	
3	1.14 (1) 0.98 (2)		1.854 (2)		1.293
4	0.9071 (1)		1.8387 (3)	2.395	
5			1.922 (5)		
6			1.8892 (8)		
7					
8	1.1305 (1)	1.4361	1.8082 (3)	2.1638 (1)	1.0684
9			1.9825 (8)		
10			1.9933 (8)		
11	218 (1) 0.7383 (2)		1.9843 (3)		
12			1.6946 (7)		

imageno	2*2 with circle(num)	ithout circle(num)	2*4 with circle(num)	2*4 without circle(num)	fakeones(num)
1		1.59	1.82(5),1.76(3),1.749(8)		1.29,1.25
2	1.25(1)		1.98(8),1.93(10),1.63(3),2.02(8)		1.83
3		1.024(1)	1.887(3),2.056(8),2.11(3),2.09(2)		1.56(3)
4	0.9812(1),	1.4785	1.047(10)		1.82,1.247,1.87
5			1.911(1), 1.616(3),1.52(10)	2.05(1),1.50(2),1.6495, 0.77, 2.22	
6		1.035	2.19(2)		1.97(3)
7			1.772(7)		2.48
8		1.5, 1.638(no)	1.1079(10)*2 1.6482(9)		1.1769(2) 1.7478
9			1.955(3)		1.4792
10			1.8294(3) 1.7797(8)		
11	1.0785(1) 1.0614(1)		1.9461(3)1.5446(8)		1.1922,1.9733
12	1.1046(9)		1.1046(*2)		1.967