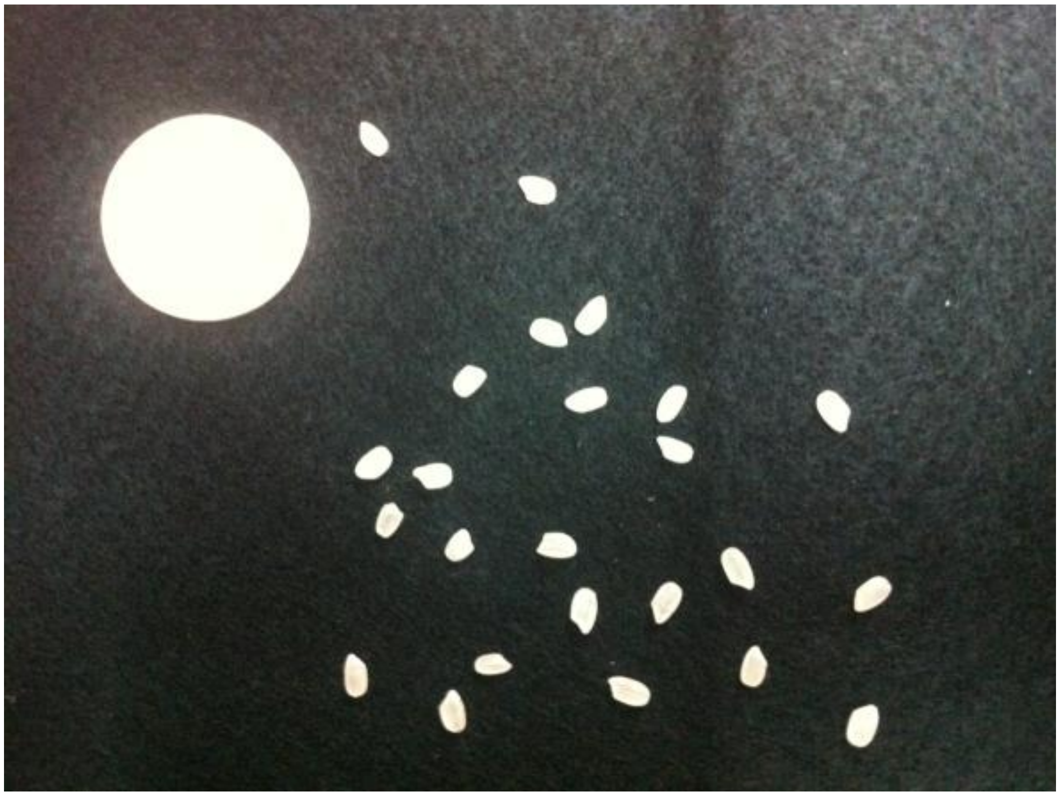
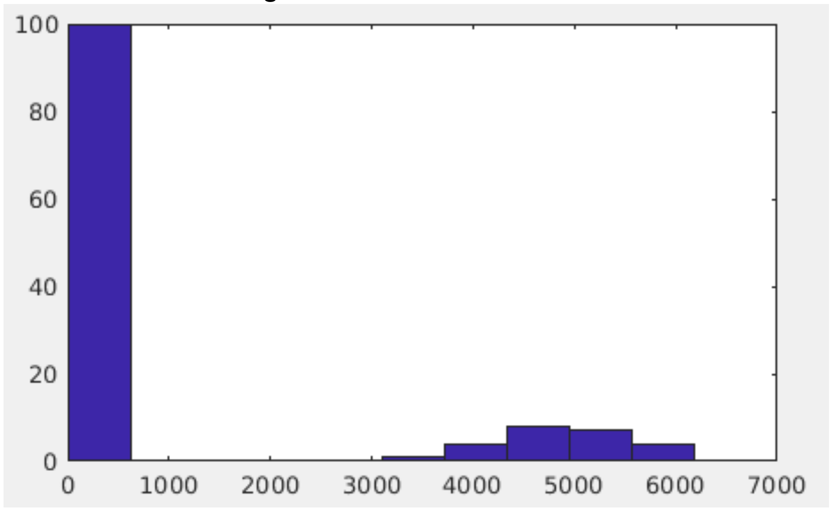
lab09_HW

1. Write a (not necessary GUI) program to calculate the number of rice grains for the given image `“09Rice.jpg”` (see below). Identify the size, centroid, perimeter, major axis length, minor axis length, and orientation for each grain. A ten-dollar coin was put with the grains when the photo was taken; hence, you can calculate the actual sizes of the grains rather than providing their sizes in pixels. Try **NOT** to use built-in function `regionprops` .



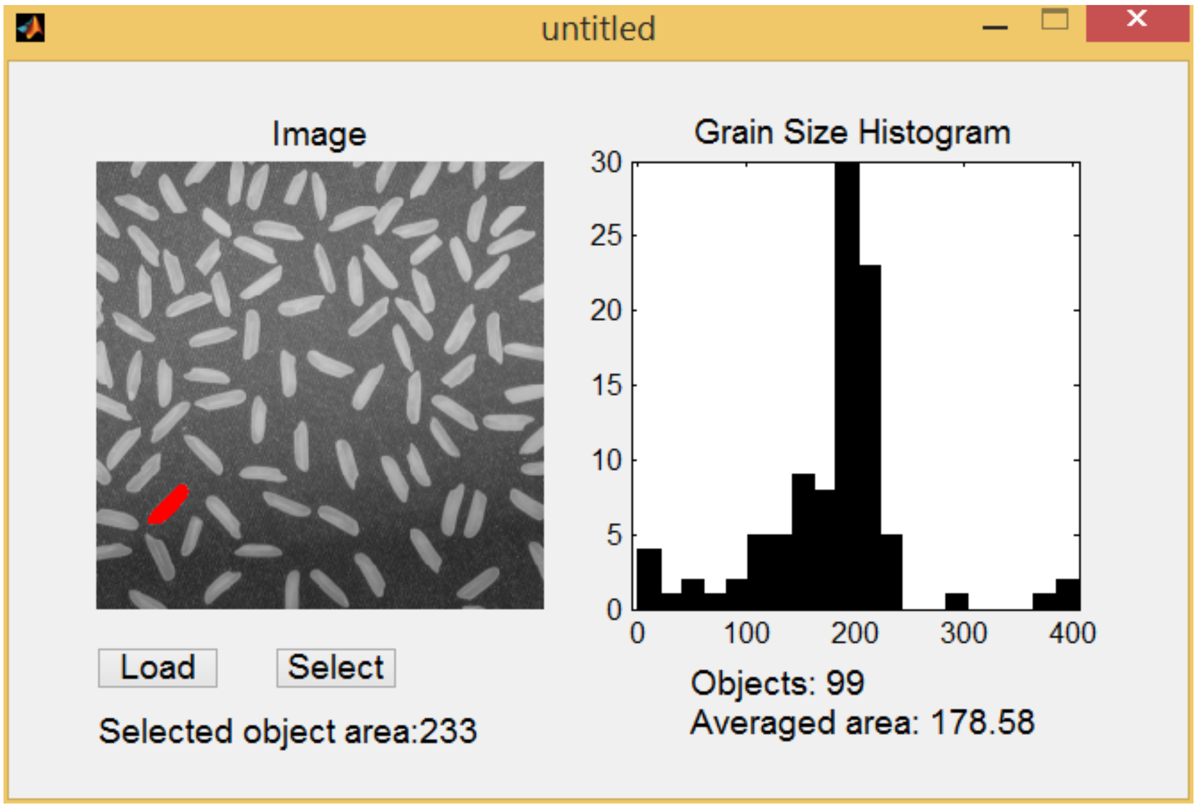
transfer to greyscale image
and use `imbinarize` to black-white image
use `bwlabel` to label white color
You can use `regionprops` now.
But, I write it by myself,
`find(bwimg == 1)` to find x-y value of labeled 1 rice
`sum(sum(bwimg == 1))` to count labeled 1
this assume that rice are well-separated
remove the max one(because it is coin)
and see the historgram



and you will see that there are some small white point
just remove it, when the size is too small
`labelsum = labelsum(labelsum ~= coin & labelsum >= coin * 0.001);`
and output
see `rice_size.m`

```
>> rice_size
mean
    0.0236
std
    0.0032
number
    24

all rice size in the ratio of coin
  Columns 1 through 5

    0.0231    0.0254    0.0199    0.0192    0.0227

  Columns 6 through 10

    0.0239    0.0187    0.0215    0.0173    0.0241

  Columns 11 through 15

    0.0241    0.0235    0.0236    0.0270    0.0255

  Columns 16 through 20

    0.0260    0.0250    0.0224    0.0197    0.0270

  Columns 21 through 24

    0.0228    0.0286    0.0298    0.0260
```

2. Design a GUI program to analyze the `“09rice.png”`. The program should allow the users to load an image and then display the histogram of the connected components in the image. The program should contain a “select” button with which the users can interactively select objects to be analyzed in the loaded image. Try NOT to use the built-in functions for calculating the grain statistics. You may need the built-in function `ginput` to allow the users for selecting objects of interest.



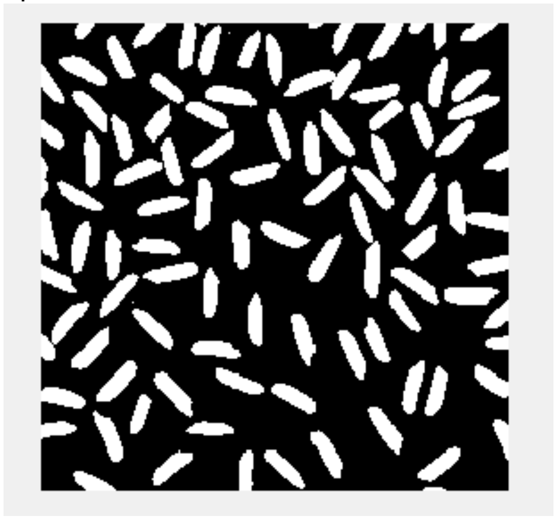
Selected object area:233

Objects: 99
Averaged area: 178.58

see `rice_cleaner.m`

`imopen` is Morphologically perform erosion followed by a dilation. In this case, large pattern will remove rice and get the bakcground image

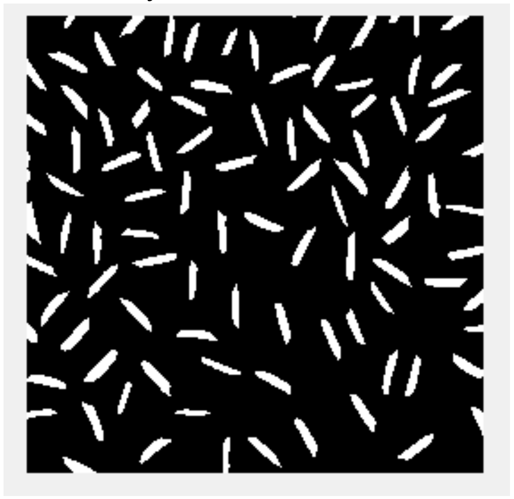
`imbinarize` and `im2bw` is same thing, it will cacluate threshold for you if you don't specific threshold value.



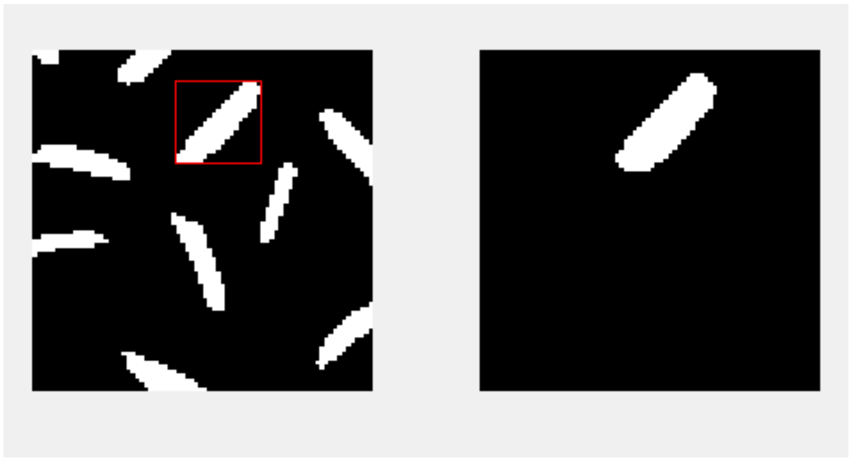
Now we have binary image with some rice connect together and some error small pixel

use Morphology erosion to calculate

This question didnot specifically saying how to deal with rice at the edges. I assume it as a very small rice



dilate each rice to get real size



then you can plot historgram or do something

see `rice_cleaner.m#riceSelect`

if you click on the image

it will check every rice that if it is on it

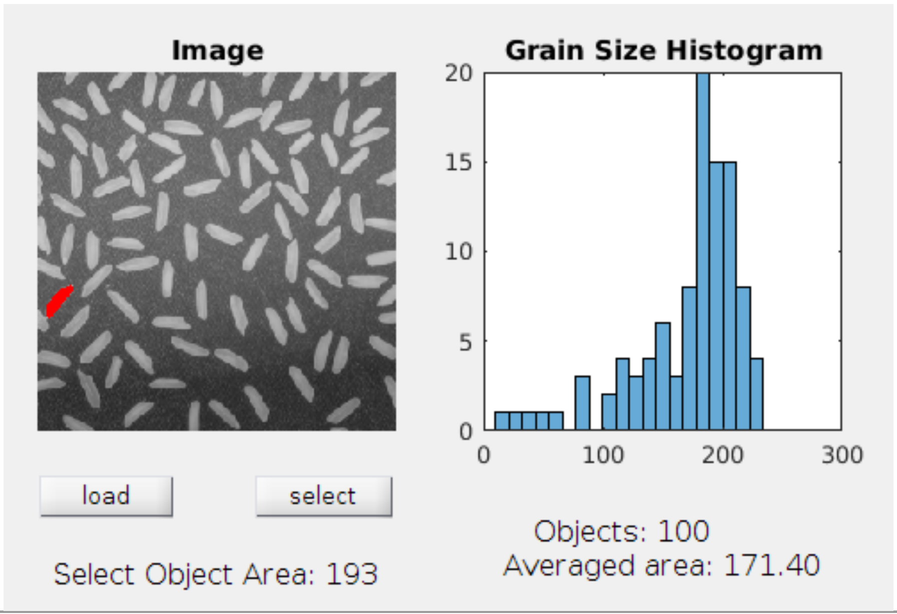
and get the index of rice and fill with color

be care for the output of `ginput` is `[y x]`

see `rice_selector.m`

see `rice_selector.fig`

see `rice_selector.png`



Select Object Area: 193

Objects: 100
Averaged area: 171.40

3. Connected-component labeling is an algorithm used to detect connected regions in binary digital images. The algorithm subsets of connected pixels in an image a unique number. The algorithm can be implemented using a two-pass approach. The first pass assigns temporary labels to the pixels. The second pass replaces each temporary label by the smallest label of its equivalence class.

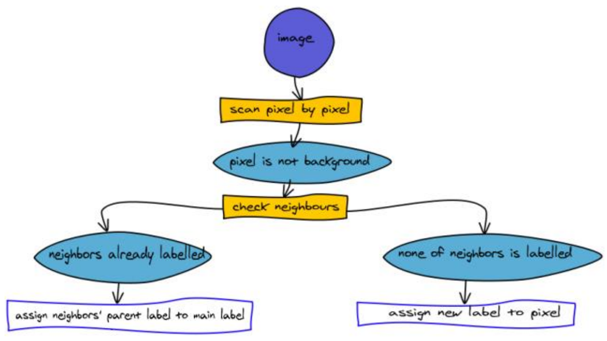
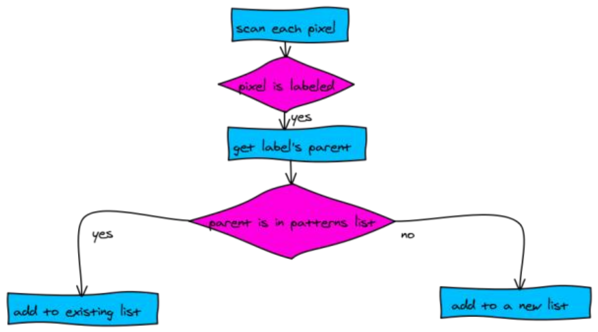
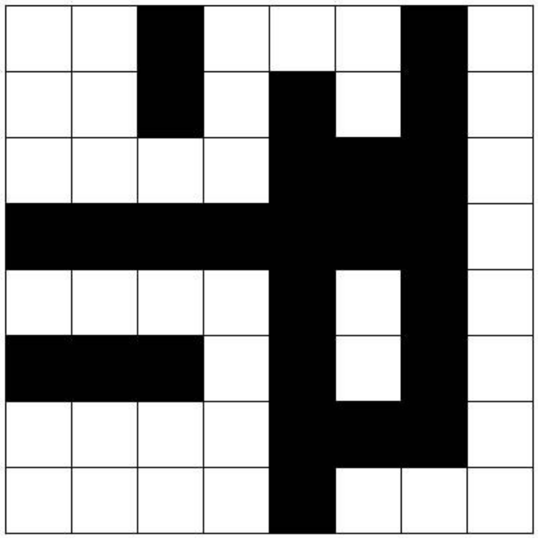
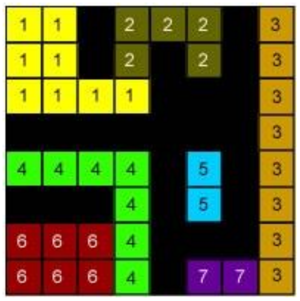
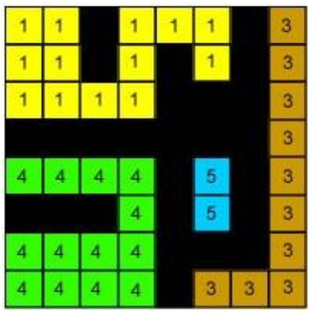
First-pass	Second-pass
	

image
scan pixel by pixel
pixel is not background
check neighbours
neighbors already labelled
none of neighbors is labelled
assign neighbors' parent label to main label
assign new label to pixel

Below find an example for the original (binary) image, first-pass image, and second-pass image. In the first pass, the algorithm searches for connected pixels row by row. If there exists a neighbor pixel in a previous row, the labels of the current pixels are assigned to the label of the neighbor pixel. Otherwise, the labels of the current pixels are assigned to a new number. In the second pass, a table contains the equivalence relationships between the labels are generated. The labels are then updated according to the table.

Original image	First-pass image	Second-pass image
		

Set ID	Equivalence label
1	1, 2
3	3, 7
4	4, 6
5	5

Write a function that implements the two-pass connected-component labeling algorithm.

> stragely : go through every pixel
>
> - if one of upper or left is labeled, label it same number
> - if none of upper and left are labeled, label new number
> - if both upper and left are labeled and dirrerent, label it with last one and relabel old one with this one and set old one in label array to 0
>
> to set label array, disjoint set is the best solution, but now, i use stupid one(use a array to store if the label is used(0 unused 1 used).
>
> if there are unused label, label the biggest one to reduce label id.
>
> see `label_component.m`
> test it

```
>> oriimage = [
    0 0 1 0 0 0 1 0 ;
    0 0 1 0 1 0 1 0 ;
    0 0 0 0 1 1 1 0 ;
    1 1 1 1 1 1 1 0 ;
    0 0 0 0 1 0 1 0 ;
    1 1 1 0 1 0 1 0 ;
    0 0 0 0 1 1 1 0 ;
    0 0 0 0 1 0 0 0 ];

>> [labelimg, num] = label_component(oriimage)
labelimg =
     1     1     0     1     1     1     0     2
     1     1     0     1     0     1     0     2
     1     1     1     1     0     0     0     2
     0     0     0     0     0     0     0     2
     3     3     3     3     0     4     0     2
     0     0     0     3     0     4     0     2
     3     3     3     3     0     0     0     2
     3     3     3     3     0     2     2     2
num =
     4
```

star me on Github

linnil1 (https://github.com/linnil1/Lab304_2017summer)

Test on ubuntu16.04 + Matlab R0217a academic use