Binary Image Analysis: Part 1
Readings: Chapter 3: 3.1, 3.4, 3.8

- thresholding grayscale images
- connected components labeling

Binary Image Analysis

Binary image analysis

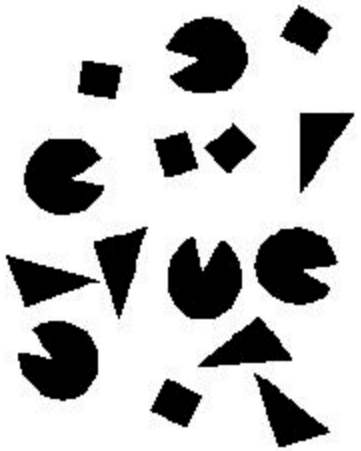
- consists of a set of image analysis operations that are used to produce or process binary images, usually images of 0's and 1's.

0 represents the background
1 represents the foreground

```
0001001000 1000
0001111000 1000
0001001000 1000
```

Binary Image Analysis

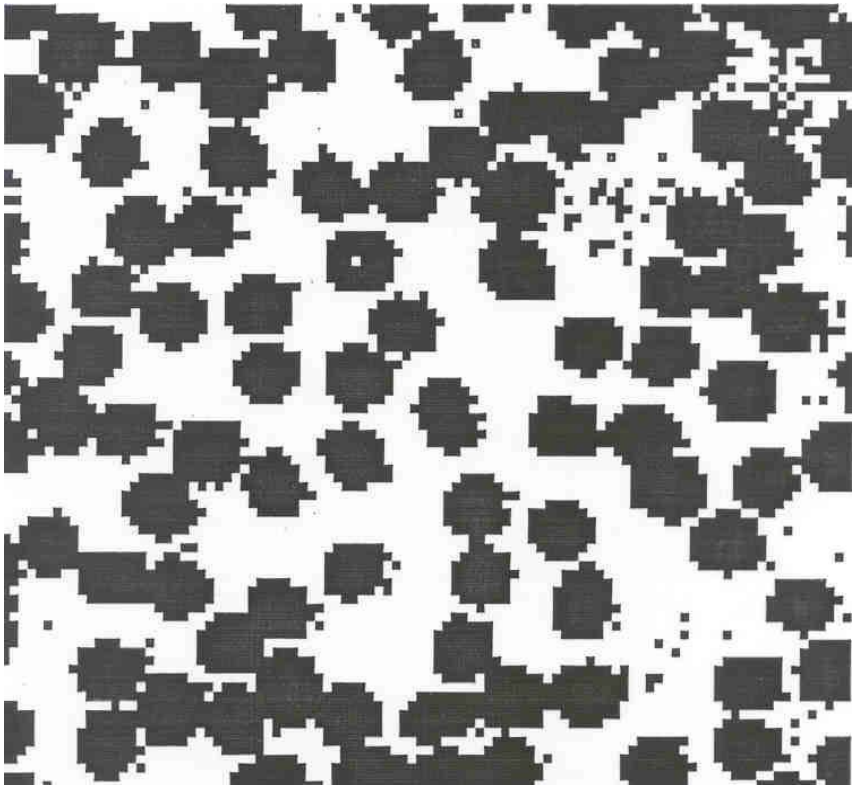
- is used in a number of practical applications, e.g.
  - part inspection
  - riveting
  - fish counting
  - document processing



What kinds of operations?

- Separate objects from background and from one another
- Aggregate pixels for each object
- Compute features for each object

Example: red blood cell image



- Many blood cells are separate objects
- Many touch – bad!
- Salt and pepper noise from thresholding
- How useable is this data?

Results of analysis

- 63 separate objects detected
- Single cells have area about 50
- Noise spots
- Gobs of cells

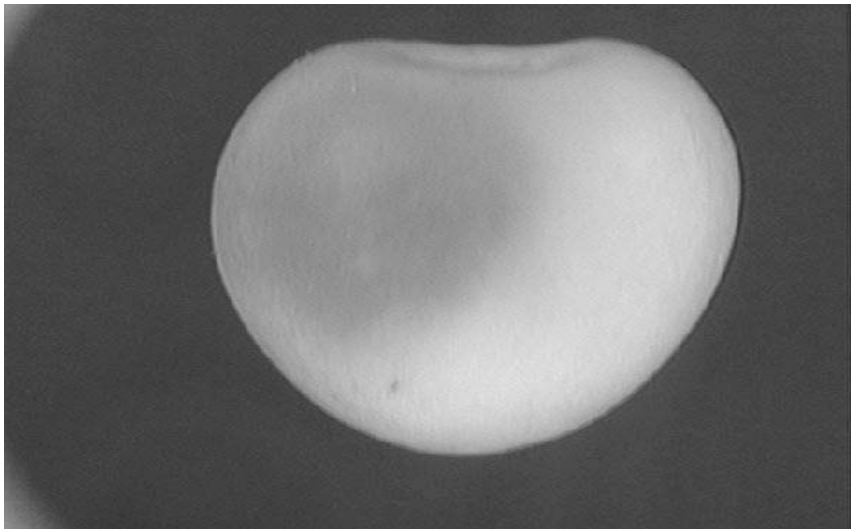
Object	Area	Centroid	Bounding Box	
1	383	(8.8 , 20)	[1 22 1 39]	
2	83	(5.8 , 50)	[1 11 42 55]	
3	11	(1.5 , 57)	[1 2 55 60]	
4	1	(1 , 62)	[1 1 62 62]	
5	1048	(19 , 75)	[1 40 35 100]	gobs
32	45	(43 , 32)	[40 46 28 35]	cell
33	11	(44 , 1e+02)	[41 47 98 100]	
34	52	(45 , 87)	[42 48 83 91]	cell
35	54	(48 , 53)	[44 52 49 57]	cell
60	44	(88 , 78)	[85 90 74 82]	
61	1	(85 , 94)	[85 85 94 94]	
62	8	(90 , 2.5)	[89 90 1 4]	
63	1	(90 , 6)	[90 90 6 6]	

Useful Operations

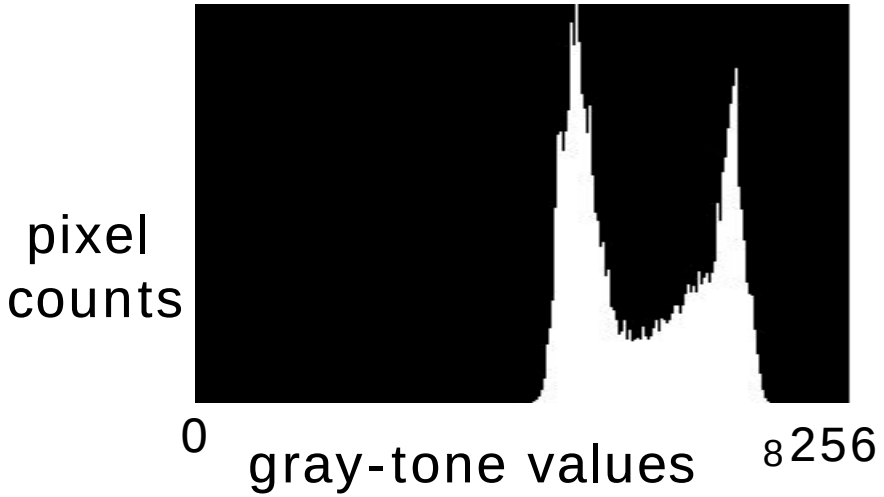
1. Thresholding a gray-tone image
2. Determining good thresholds
3. Connected components analysis
4. Binary mathematical morphology
5. All sorts of feature extractors
   (area, centroid, circularity, ...)

Thresholding

- Background is black
- Healthy cherry is bright
- Bruise is medium dark
- Histogram shows two cherry regions (black background has been removed)

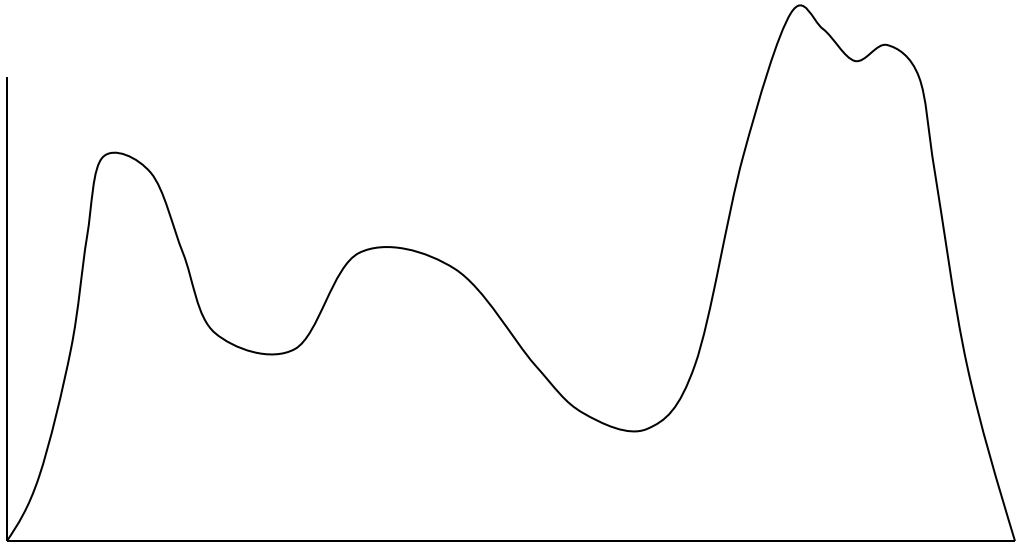


This is abstract.
How are histograms represented as data structures?



Histogram-Directed Thresholding

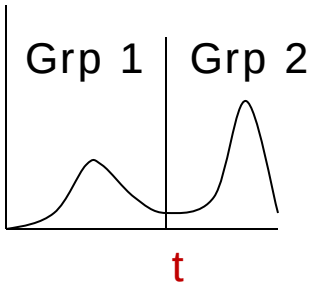
How can we use a histogram to separate an image into 2 (or several) different regions?



Is there a single clear threshold? 2? 3?

Automatic Thresholding: Otsu's Method

Assumption: the histogram is bimodal



Method: find the threshold t that minimizes the weighted sum of within-group variances for the two groups that result from separating the gray tones at value t.

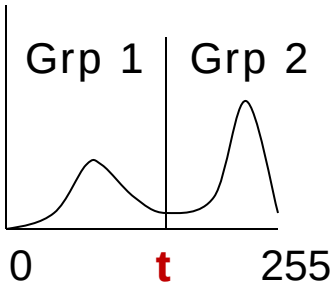
What is variance?

Computing Within-Group Variance

- For each possible threshold t
- For each group i (1 and 2)
  - compute the variance of its gray tones $\sigma_i^2(t)$
  - compute its weight $q_i(t) = \sum_{\tau \text{ in Grp i}} P(\tau)$

where $P(\tau)$ is the normalized histogram value for gray tone τ, normalized by sum of all gray tones in the image and called the probability of τ.

- Within-group variance $= q_1(t)\,\sigma_i^2(t) + q_2(t)\,\sigma_i^2(t)$



See text (Section 3.8) for the efficient recurrence relations; in practice, this operator works very well for true bimodal distributions and not too badly for others, but not the CTs.

Thresholding Example

original gray tone image

binary thresholded image

Connected Components Labeling

Once you have a binary image, you can identify and then analyze each **connected set of pixels**.

The connected components operation takes in a binary image and produces a **labeled image** in which each pixel has the integer label of either the background (0) or a component.

binary image (after morphology)

connected components
(shown in pseudo-color)

Methods for CC Analysis

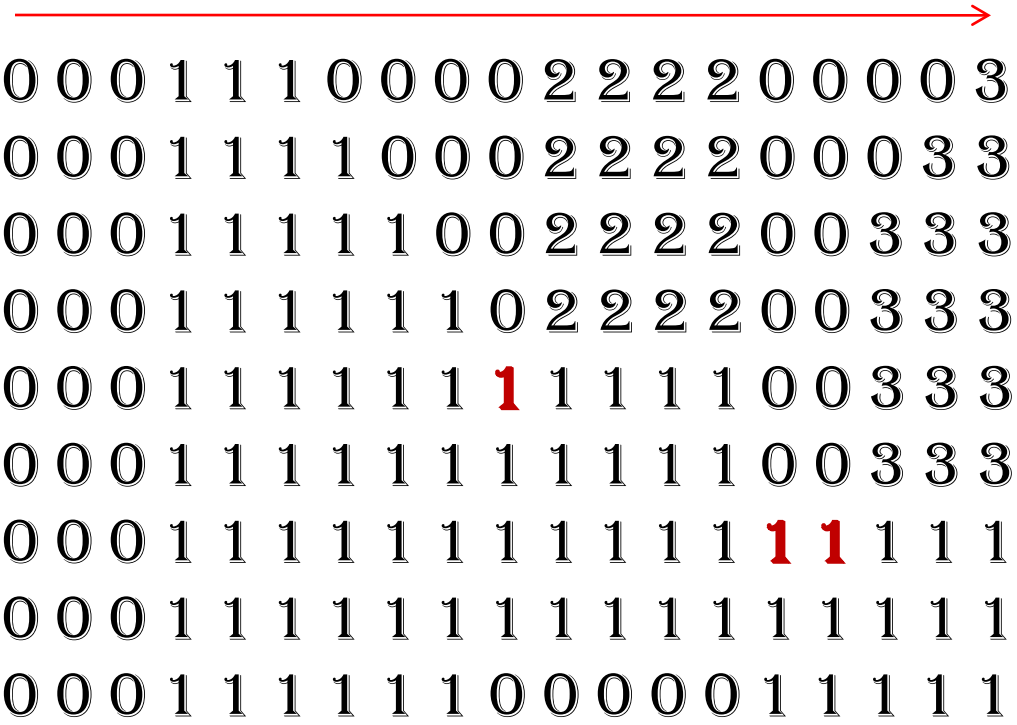
1. Recursive Tracking (almost never used)
2. Parallel Growing (needs parallel hardware)
3. Row-by-Row (most common)
    - Classical Algorithm (see text)
    - Efficient Run-Length Algorithm (developed for speed in real industrial applications)

Equivalent Labels

Original Binary Image

Equivalent Labels

The Labeling Process:
Left to Right, Top to Bottom

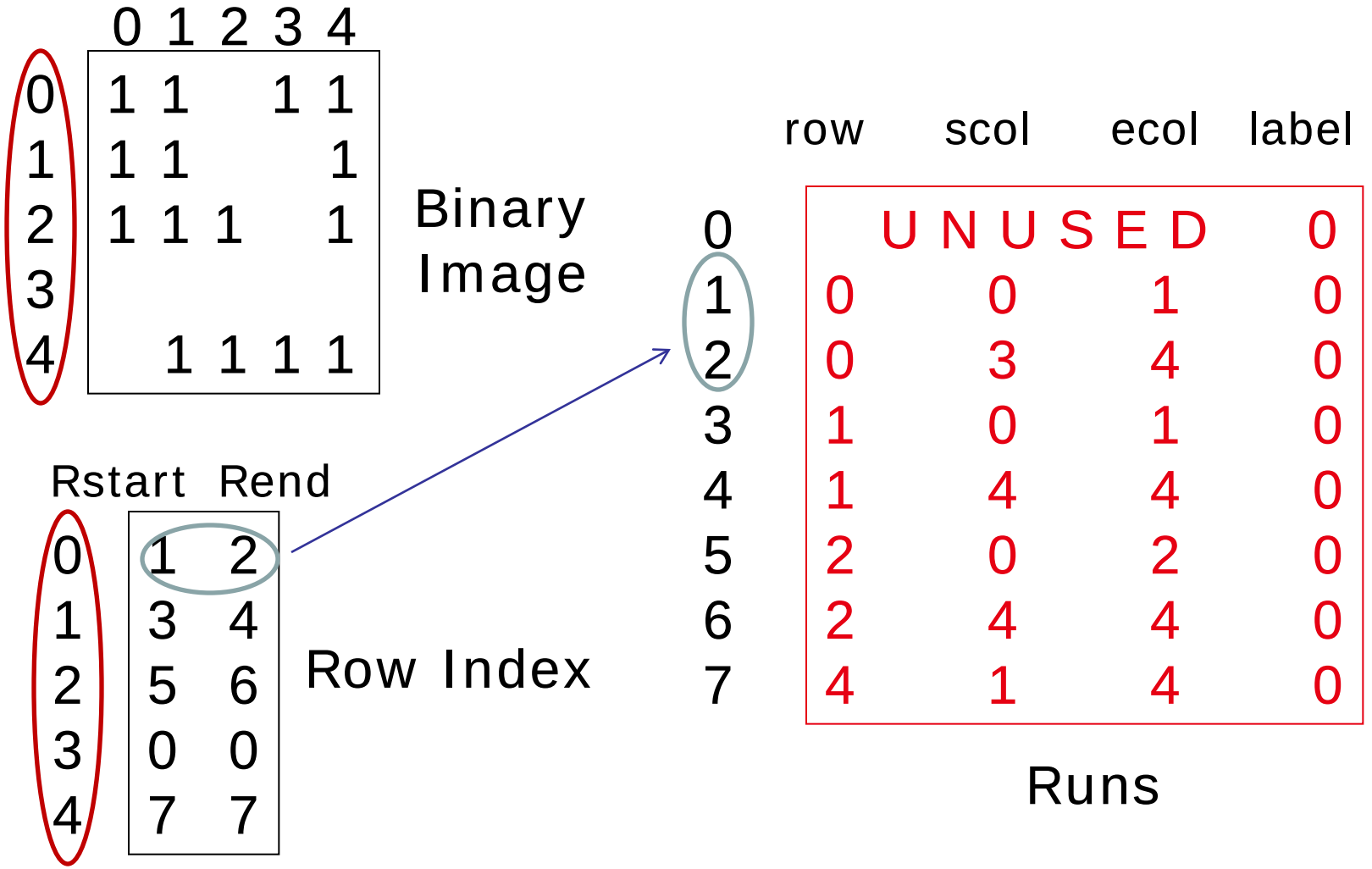


$1 \equiv 2$
$1 \equiv 3$

What CSE algorithm does this kind of thing on a different structure?

Run-Length Data Structure

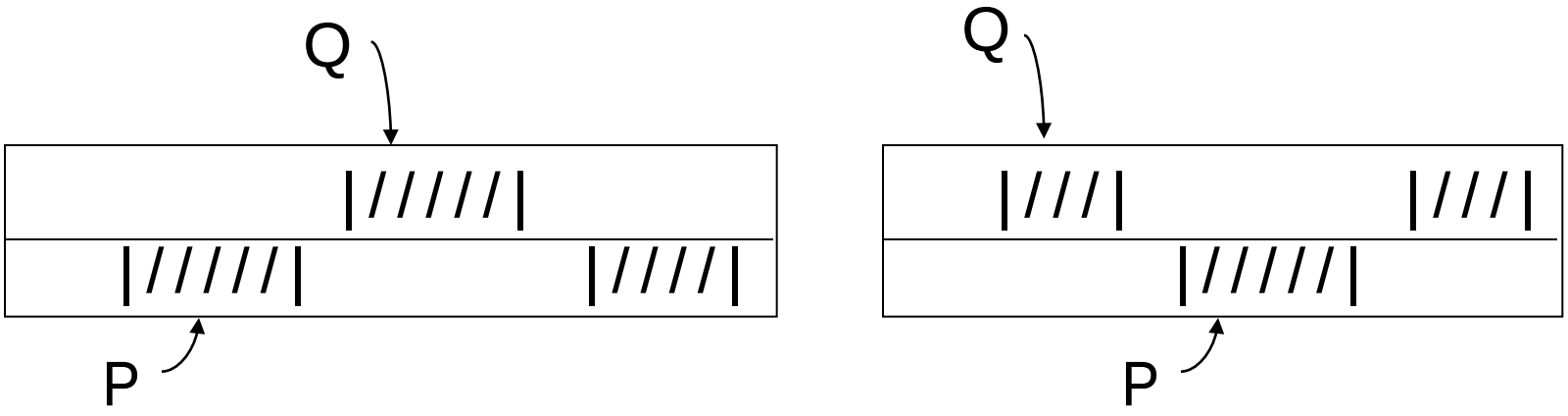
used for speed of processing and to gain from hardware



Run-Length Algorithm

```
Procedure run_length_classical
  {
   initialize Run-Length and Union-Find data structures
   count <- 0

/*  Pass 1 (by rows) */

   for each current row and its previous row
     {
      move pointer P along the runs of current row
      move pointer Q along the runs of previous row
```

Case 1: No Overlap

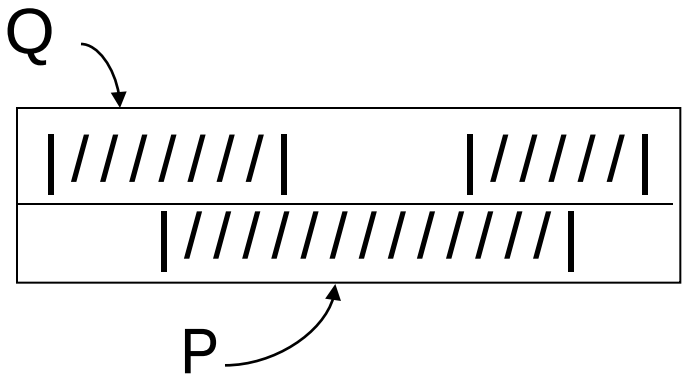



```
/* new label */
count <- count + 1
label(P) <- count
P <- P + 1
```

```
/* check Q's next run */
Q <- Q + 1
```

Case 2: Overlap

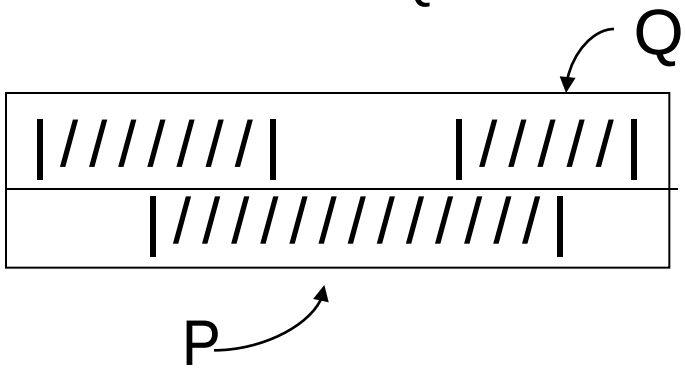
Subcase 1:
P's run has no label yet



label(P) <- label(Q)
move pointer(s)

}

Subcase 2:
P's run has a label that is different from Q's run



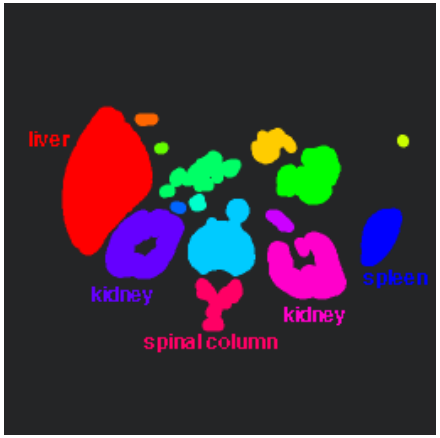
union(label(P),label(Q))
move pointer(s)

Pass 2 (by runs)

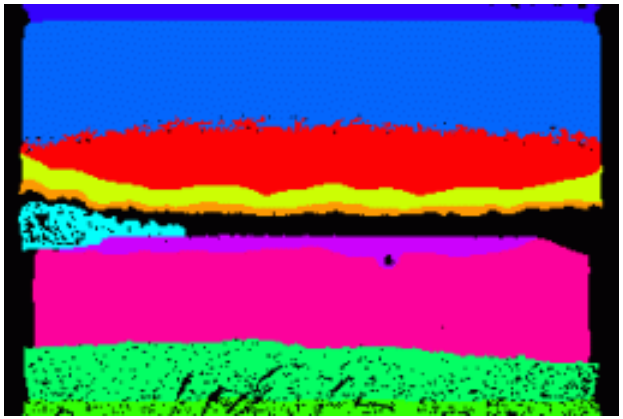
```
/* Relabel each run with the name of the
   equivalence class of its label */
For each run M
   {
    label(M) <- find(label(M))
    }

}
```

where union and find refer to the operations of the Union-Find data structure, which keeps track of sets of equivalent labels.

Labeling shown as Pseudo-Color



connected components of 1's from thresholded image



connected components of cluster labels