INDIANA UNIVERSITY

School of Informatics and Computing

Interdisciplinary Department of Biochemistry

I519 Introduction to Bioinformatics

Class Project 2010: Final

"Toward a blood-agar-plate reader"

Submitted by: Zheng Dong, Erkin Kuru, Doga Tuncay, Kehuan Zhang

Submitted to: Dr. Yuzhen Ye

Submission date: Tuesday, November 14th, 2010

Abstract

In this report and over the web (http://www.madmed.net/), we report our final progress in development of a blood-agar-plate reader, which will be used to assess effects of various mutations on the growth rate of the nasty gram-positive bacteria *Streptococcus pneumoniea*. Different growth rates are manifested as colony sizes are different on the blood plate, and our software is supposed to effectively calculate average colony size for each input image. Utilizing MatLab we have reached to a point where reliable assessment of the colony size in an input image is achieved, if the input image offers a high contrast between subject, colony, and the background.

1. Introduction

Streptococcus pneumonia is a highly contagious gram positive bacteria causing still millions of deaths annually[1]. As an infamous human pathogen, *S.pneumoniae* (pneumococcus) induces a variety of diseases from meningitis to pneumonia once it enters into the body through the respiratory track. Continuous increase of death rates related to the ever increasing antibiotic resistance [2] suggests that we have to know more about the essential proteins involved in pneumococcal lifecycle as soon as possible, which will help us develop new drug molecules inhibiting growth.

One high-throughput way to measure essentiality of a given protein is to knock-out the gene encoding that protein (i.e. generation of a single mutant) and compare growth rate of this mutant to the wild-type. Assessment of the growth rates is usually done by spreading the mutant and the wild-type cells onto sheep blood-agar plates, the media, which mimics the native nutritional condition of the host environment. When one colony forming unit (i.e. basically one or few viable cells with the potential to form one colony) of wild-type bacteria holds on to a spot on the blood plate, it starts to divide in a binary manner (~40 min per one cycle of fission) and in ~16 hours or so **a colony** of ~0.5mm diameter consisting millions of cells formed. When the duration of growth is kept constant, it is very likely that single mutations will affect the size of the colony because of the changed growth rates.

Once the essentiality of a large set of proteins is screened through the single mutation approach, one can also consider to generate double mutants (i.e. the 2-combination of proteins in the set), which can be used to determine interactions of the two proteins under investigation in vivo. For example, if the growth rates of the single mutants of protein A and protein B are not effected, but the growth rate of the double mutant of protein A and protein B is significantly decreased, one would argue that these proteins somehow interact with each other in vivo, maybe having an overlapping function rescuing the cells viability in the absence of one of the two. Such a data would be an amazing starting point for detecting functions and interactions of hundreds of proteins yet waiting to be characterized.

Although this is an efficient approach, when this approach is applied to study essentiality of hundreds of proteins, the need to automate and to standardize growth rate measurements manifests itself, since the sample sizes arrive to thousands with the combinations of double mutants. Thus, the purpose of this project is to develop a software, which will both automate but also standardize colony size measurements in input blood-agar-plate images.

Although there are existing software referring to this issue[3-6], they are either very expensive, requires co-purchase of expensive hardware or restricted their functionalities on transparent plates only, while red colored blood-agar-plates cannot be applied. In addition, these software either come with their own expensive image acquisition devices[3, 5] or appeal the user to adjust parameters heuristically and to provide high contrast images [4, 6].

Here we report development of a free, highly customized blood agar plate reader, which can report average colony size in square millimeters for input images. Similar to the commercially available options, our program too requires high contrast input images, in order to calculate colony sizes reliably. We acknowledge this to be one of our limitations.

2. Overview of the Program

This program takes a standardized image (See Figure 1) of a blood-agar-plate containing colonies as input (See section 3.1 for more on the properties of the input) and report the average size of colonies for each input image in square millimeters as output.

The input of this program should be a high contrast, zoomed image of a blood agar plate preferably showing more than one distinct colonies (See Figure 1 A). All the input images will be required to have the same zoom factor.

Our program first converts an input image to the black-white scale, and detects high contrast changes in the image. Continuous pixels of same color are marked, as shown in Figure 2 A and 2 B. If these continuous pixels constitute a rough circle with radius in a certain range, that pixel cluster is identified as a potential colony, and pixels in that cluster are then counted in the final result. In the case several clusters (i.e. colonies) are detected in one input image, our program calculates the average area of colonies by a division of the total areas of colonies by the number of distinct colonies we observed.

Our program ignores the situation when multiple colonies merged or there are incomplete colonies at the edges of the input image (See Figure 1 A, the upper left colony), since in these

3. Results and Discussion

3.1. Properties of the Input and Image Acquisition

Properties of the input:

As mentioned in the introduction, *Streptococcus pneumoniae* can thrive on sheep blood agar plates. A blood agar plate is basically a light-red jello made out of sheep blood, which contains mainly blood cells providing basic nutritional supplements for the *S.pneumoniae* to grow (Figure 1). Since the *S.pneumoniae* is not motile, it grows on a blood-agar plate spherically. As pneumococcus grows, it lyses sheep blood cells in order to acquire nutrients and this activity manifests itself as a dark-red hemolytic halo around the colony. If initially there are too many viable pneumococcal cells spread on a plate, it is very likely that some merged colonies will grow, which complicates any type of analysis (e.g. genetic) considerably. The ideal input image should show multiple distinct colonies, so that a reliable average size can be calculated (See Figure 1 A versus 1 B)

Image Acquisition:

We determined that highly standardized high contrast images can be achieved with a camera attached to a microscope in Jordan Hall 5th floor. With this microscope depending on the chosen lighting, it is possible to get both very high contrast pictures (Figure 1-A), but also relatively low-contrast pictures (Figure 1-B). In order to make our program function reliably, the user should take necessary steps to acquire high contrast input pictures. The user should put emphasis on acquiring images of only well separated colonies, since the program might fail otherwise. Additionally the user should make use to add a green 1mm scale anywhere on the input picture, which is crucial for a reliable report of colony size in square millimeter

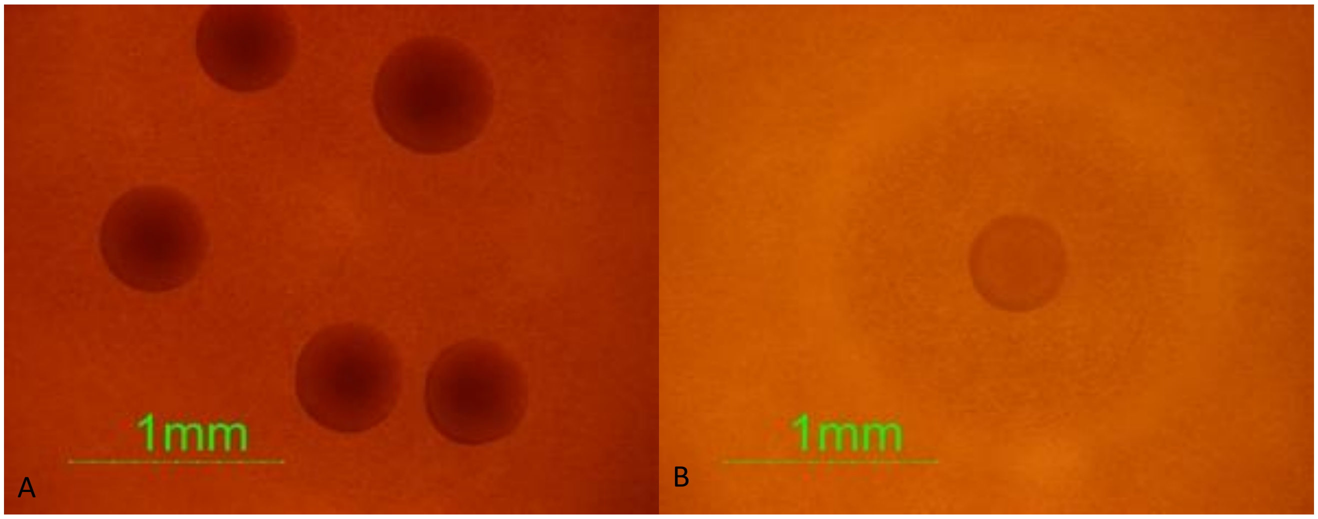


Figure 1: Sample input images of pneumococcal colonies grown on sheep blood agar plates . In thess images what is referred as a colony is the intensely grey colored dots, whereas the dark red halo around those grey spots are hemolytic regions. A- An ideal high contrast image. Note that the colonies are separate and the image has high contrast. B- An undesirable input image, because it is both a low contrast image and it shows only one colony. The size of only one colony can not be used as the representation of the whole population. Also our program will fail to distinguish the colony in this low contrast set-up and give inaccurate measurements.

3.2.The program

In order to get accurate size of colonies in the images, we need to take four steps. The first step is image preprocessing, which will convert the image from Red-Green-Blue color space into Black-White space, improve the image contrast, and also remove the noise as much as possible. The second step is counting the colony numbers. The third step is counting the pixel number of colonies in the image. And the final step is get the image scale factor and convert the colony size from pixel into square millimeter. Each step will be introduced in the following subsections.

3.2.1. Image preprocessing

The purpose of image preprocessing is to get a transformed image that will make colony identification and measuring easier and more accurate than the original form. There are three basic transforms: conversion into black-white format, contrast improvement, and edge detection.

(A) Conversion into black-white format

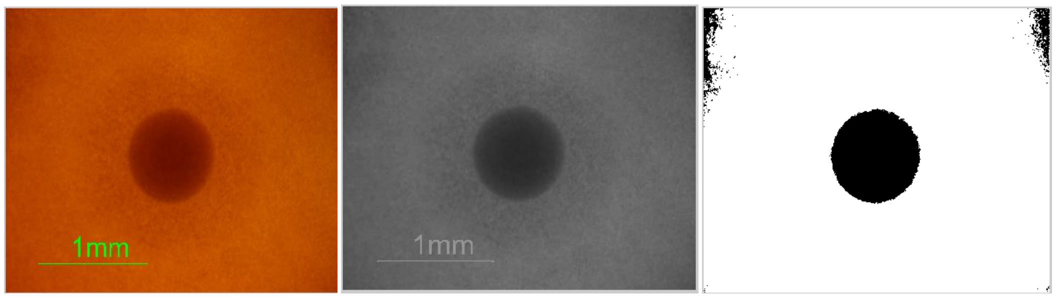
This is done using the existing API functions provided by Matlab. And the code we are using is given below:

```
rgb=imread(filename); % load data from image file
gray = rgb2gray(rgb);  % convert into gray level image first
threshold = graythresh(gray); % get the conversion threshold
im1=im2bw(gray, threshold); % convert into black-white image
im2=1-im1; % inverse the black and white color
```

The reason why need black-white image is because the analysis in following steps, for example, the edge detection, is based on computations on 2-dimension matrix with value of 0 or 1, but a color image is a 3-dimension matrix, and each pixel in gray image has values other than 0 or 1.

It is worth to mention that the last statement, “im2 = 1 – im1”, actually do a black-white color inversion transformation, which means: the while pixels in image im1 would become black in image2. This is also due to the requirements of operations in following steps, like edge detection and colony size estimation.

Figure 2(a) (b) and (c) shows a sample original input image, converted gray and black-white image respectively.



(a) the original input   (b) converted into gray scal   (3) converted into BW scale

Figure 2 Sample output of pre-processing stage

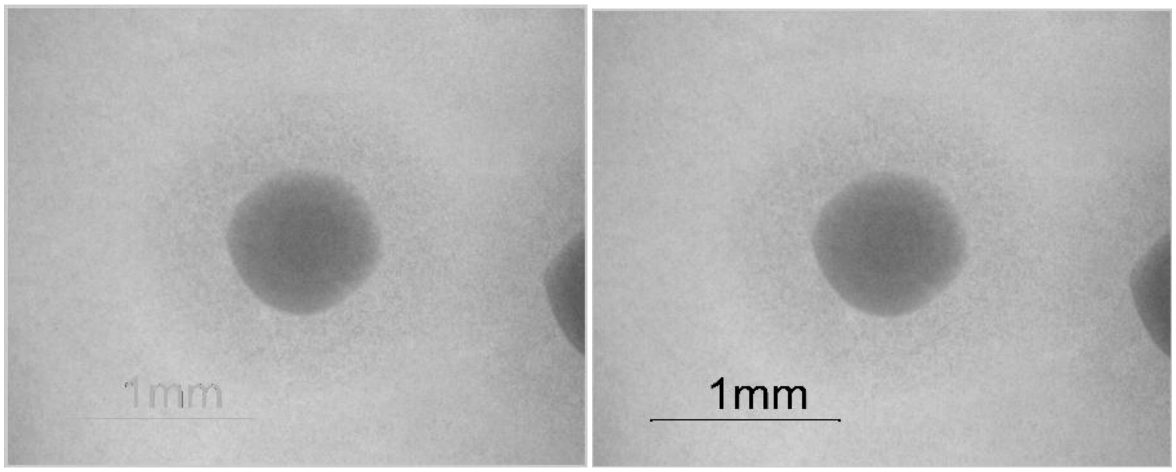
(B) Contrast improvement

Due to some reasons, some input images have very low contrast, which will affect the accuracy of the colony size estimation. So it is necessary to improve the contrast. There are several existing function in Matlab image processing toolbox, but we also designed a new algorithm and key part of the code is given below:

```
[yy,xx]=find(rgb(:,:,2)>=200);
green=rgb(:,:,1);
m = mean(mean(gg))
for i = 1:length(yy)
  y=yy(i);      x=xx(i);      gg(y,x) = m;
end
```

```
 7   ... ...
 8
 9   p1=20;   p2=20;
10   g1=gray.^p1;
11   g2=mat2gray(g1);
12   g3=1-g2;
13   g4=g3.^p2;
```

Code between Line 1 and L6 is used to lower the effect of green scale that gives the conversion factor between the size of pixel and the size of millimeter. The green scale has such a bright green color that it has negative effect on the quality of converted black-white images. The code here is trying to replace those pixels that have bright green color with the average color of the whole image. Why the average, not the minimum or even 0? This is because only the average can have the minimum effect on the output results. As an example, Figure 3 gives a comparison results between the average and the value of 0. Figure 3(a) is using the average color, and Figure 3(b) uses 0.



(a) Using the average color (b) Using color 0

Figure 3 Comparison of contrast using average color and 0

(C) Edge detection

Based on the black-white image generated from previous steps, we can use the API functions provided in Matlab to do the edge detection. The code is pretty simple that given below, and one sample output is shown in Figure 4.

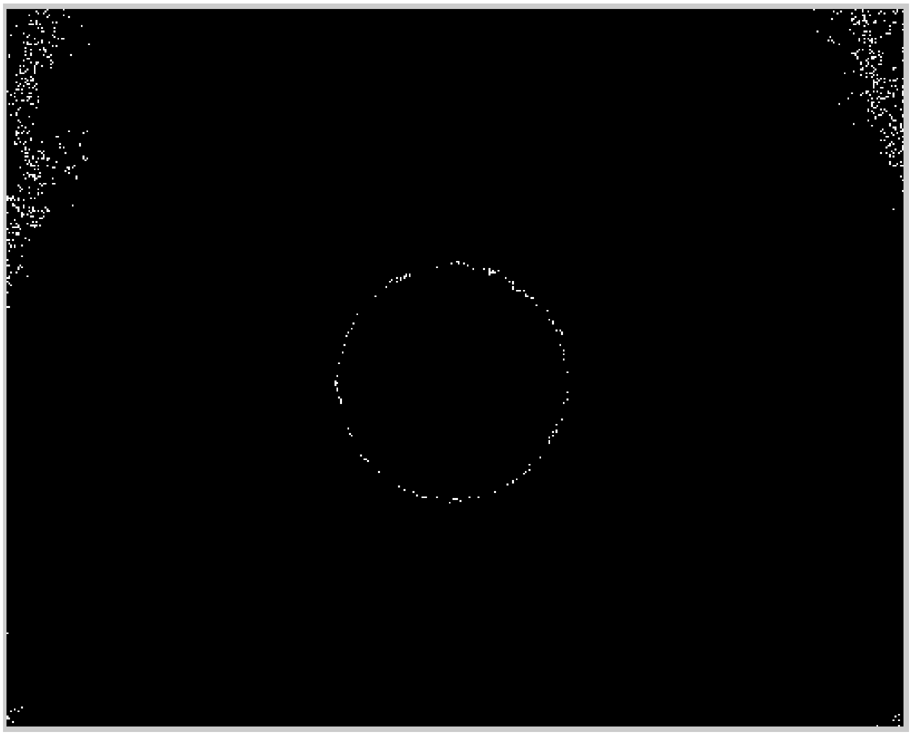


Figure 4 One sample output of edge detection.

3.2.2. Get the colony number

We have designed two different approaches to get the number of colonies within a given image. The first approach is using Hough transform, and the other approach is using clusterings.

(A) Hough transform approach

Generally, the border of colonies will appear as circles, so the task of getting the number of colonies is converted into counting the number of circles within a given image. This can be done with Hough transformation [8, 9, 10]. Hough transformation is a generally shape recognition algorithm that is widely used in computer vision areas. The basic steps of using Hough transformation to identify circles with given radius R is described below:

(1) Create a 2-D matrix A that has the same size with the input image

(2) For each point (a, b) in the given image, build a circle C using that point as origin and R as the radius. So, the coordinates of points (x, y) on circle C should satisfy following equations:

```
x = a + R*cosθ
y = b + R*sinθ
```

(3) For each points (x, y) on the circle C, increase the value of cell A[x, y] by 1

(4) Repeat (2) and (3) until all image pixels have been processed.

(5) Find the cell in matrix A that has the maximum value, and the position of that cell, say (x0, y0) is most likely the origin of the circle in the given image.

This is the algorithm to identify circles with known radius. In order to identify circles with unknown radius, we need to build a 3-dimension matrix A, and the added dimension is used to record the calculation results of different radius values. And finally, we need to find out a maximum value in this 3-D matrix, and the coordinates of that element will tell us the position of origin (x,y) as well as the radius.

During the implementation, we found that the detection speed is very slow if we translate previous algorithm into code directly. So, we have done some improvements, and part of the code is show below.

```
  1   [yy, xx] = find(img); % find the white points
  2   total_pixel = length(xx);
  3   A = zeros(sy,sx, r_size); % the accumulation matrix
  4   template = build_circle_template(rmin, rmax);
  5   for i =1:total_pixel % travel for each white points
  6     for r_index = 1:r_size % iterate all possible radius
  7       r = r_range(r_index);
  8       x0 = xx(i)-r; y0 = yy(i)-r; x1=xx(i)+r; y1=yy(i)+r;
  9       if (x0 > 0 && y0 > 0 && x1 <= sx && y1 <= sy )
 10         A(y0:y1, x0:x1, r_index)=A(y0:y1,x0:x1, r_index) +
 11     template(rmax+1-r:rmax+1+r,rmax+1-r:rmax+1+r,
r_index);
 12       end % if boundary check
 13     end % for r_index = 1:r_size
 14   end % for i=1:total_pixel
```

The first improvement is using the find() function of Matlab to filter out those points that has value 1 to avoid processing every pixels in the image (shown in Line 1). Although from functionality point, it is equivalent to use a loop to iterate all pixels and do the same filtering operation in our code, the performance is completely different, because the underlying implementation of find() function is highly optimized and its execution speed is much faster than the code written in .m files.

The second improvement is to build circle templates before the processing of pixels (show in Line 3) instead of calculating them in each loop. This is based on the observation that: calculating points of circle Y in the original algorithm is repeated unnecessarily, because essentially we can get such a circle by simply shifting its origin point from (0, 0) to (a, b). And again, such shifting can be done efficiently by adding two matrixes, as is shown by the code of Line 10-11, which is much faster by the for-loop by our own code.

With these two optimizations, our code can detect the circle in the given image quickly. And Figure 5 gives a sample output, where the green cross within the circle is the detected origin.

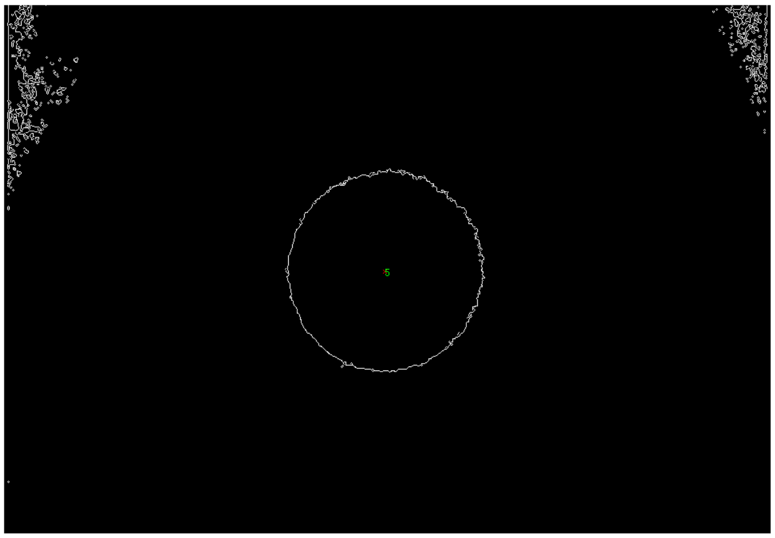
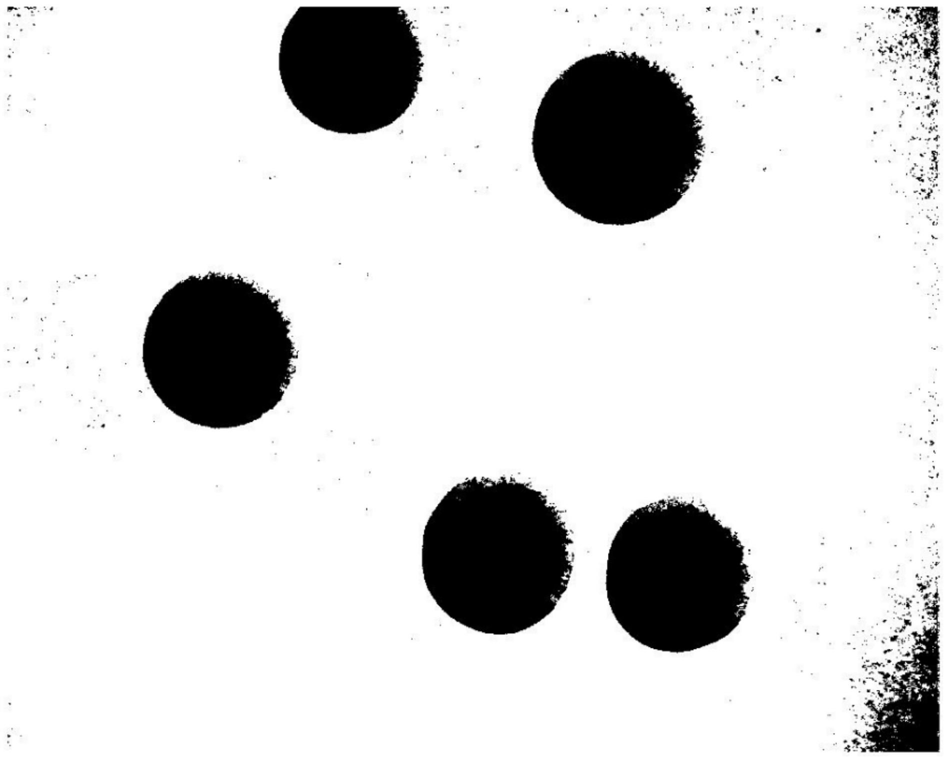
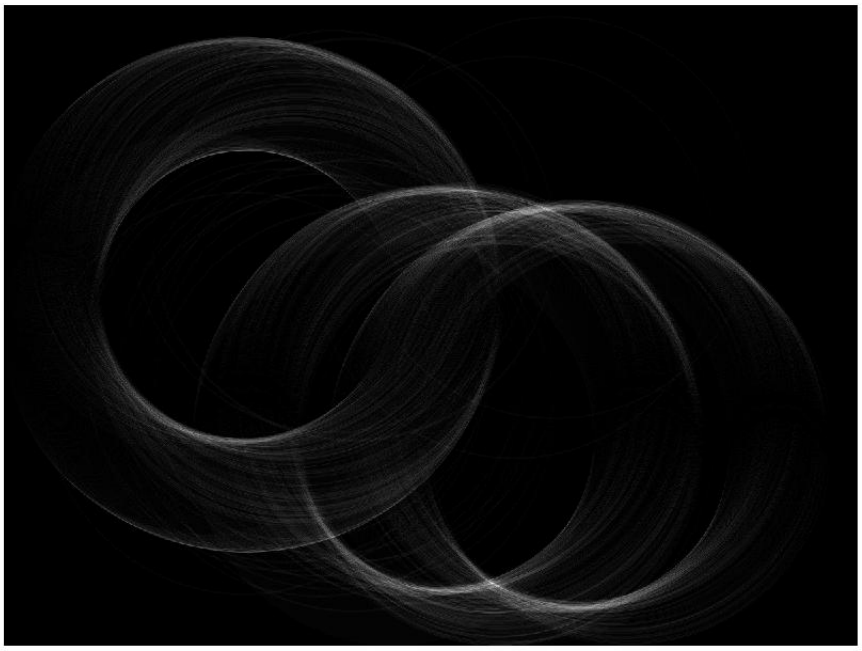


Figure 5 the detected origin of one sample image

However, when testing with a image with multiple colonies, we met a big trouble: it is very difficult to get origins of multiple circles from the output of Hough transformation, as is shown in Figure 6. To address this problem, we proposed a method that uses the clustering techniques, which is introduced in next subsection.



(a) The image before Hough transformation



(b) the image after Hough transform

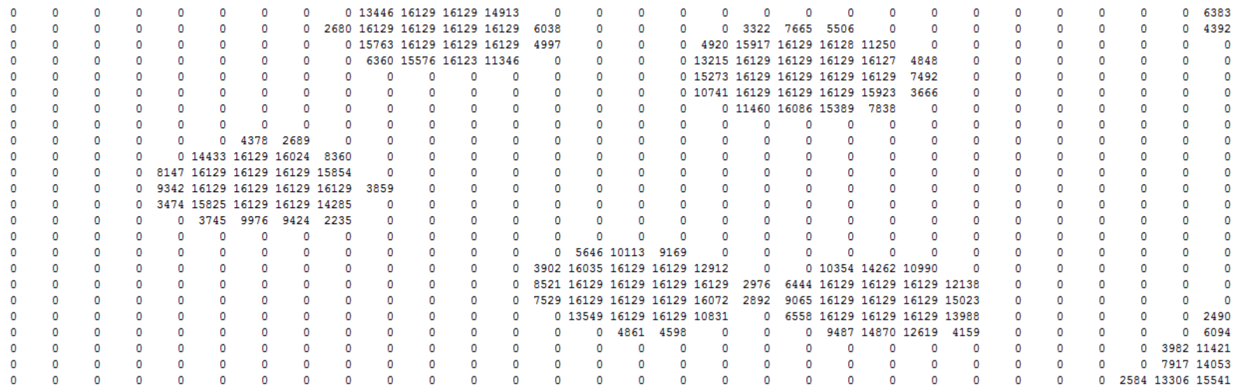
Figure 6 Hough transform output of a image that has multiple colonies.

(B) Clustering approach

This approach takes following steps:

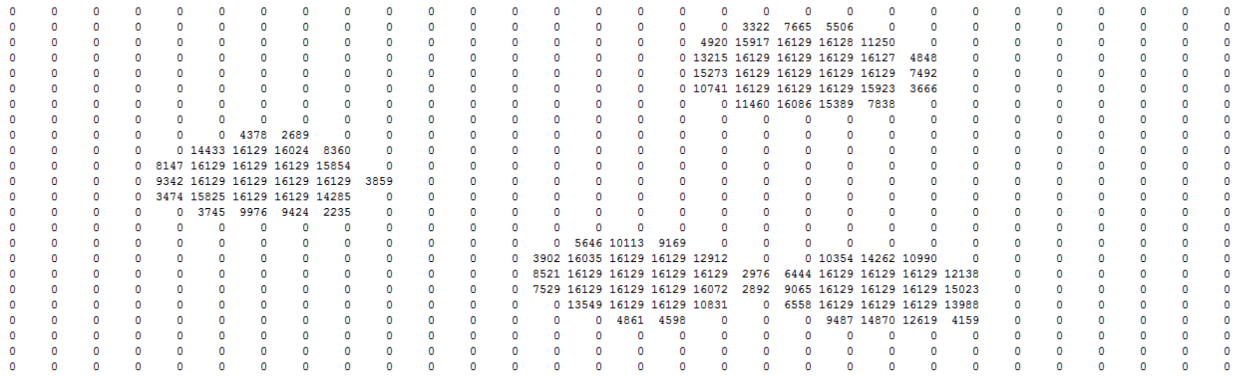
(1) Divide the image into small square blocks, with each square block has edge size of "r_min", where r_min is a predefined minimal value. Suppose the image has the width of W and height of H, then the total block numbers we get is: (W/r_min+1)x(H/r_min+1). And also create a matrix A of size M rows and N columns, where M and N is (W/r_min+1) and (H/r_min+1) respectively. In other words, each block has a corresponding matrix element.
(2) For each block, summer up the pixel values within that block. Since the image has already been transformed into black-white format, so the pixel value can be either 0 or 1. Save this sum to the position in matrix A that is mapped to this block.
(3) Clear the element values to 0 for those elements that (a) it is nonzero and is on the border, or (b) it has nonzero neighbors that is on the border or the neighbors' neighbor are on the border.
(4) Find the regional maximum value of A, and get a new matrix B, with all elements with value 0 except those positions that have the regional max values.
(5) Merge multiple nonzero elements of matrix B into a single one by clearing adjacent nonzero values.
(6) The summer of all elements in matrix B and output it as the number of circles.

Figure 7 shows the output of each steps when detecting the circles numbers of the image Figure 6(a), which is cannot be solved by Hough transformation. Figure 7(a) gives the output after step (2), Figure 7(b) shows the output after step (3), and Figure 7(c) shows the output after step (4), and Figure 7(d) shows the final results, from which we can see that there is a single nonzero value exists, thus there four circle in the image.



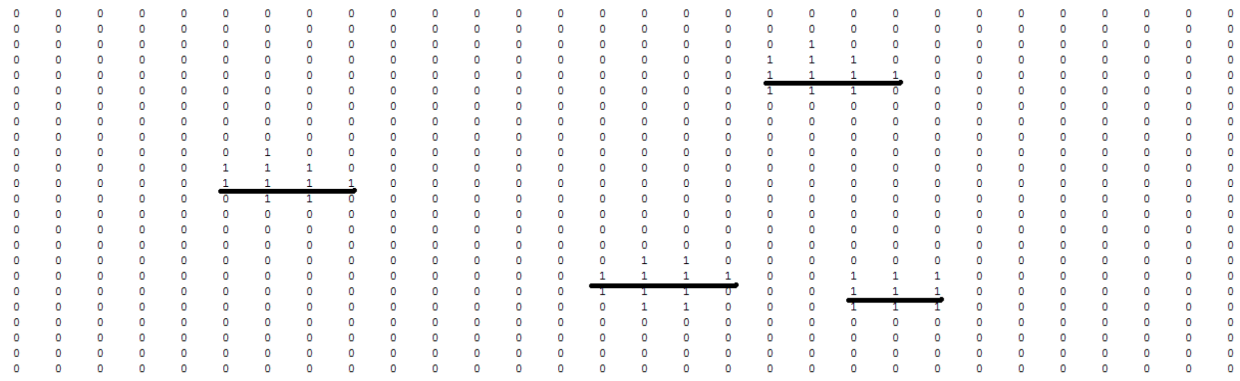
```
     0     0     0     0     0     0     0     0     0 13446 16129 16129 14913     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0  6383
     0     0     0     0     0     0     0     0  2680 16129 16129 16129 16129  6038     0     0     0     0  3322  7665  5506     0     0     0     0     0     0     0     0  4392
     0     0     0     0     0     0     0     0     0 15763 16129 16129 16129  4997     0     0     0  4920 15917 16129 16128 11250     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0  6360 15576 16123 11346     0     0     0     0 13215 16129 16129 16129 16127  4848     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0 15273 16129 16129 16129 16129  7492     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0 10741 16129 16129 16129 15923  3666     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0 11460 16086 15389  7838     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0  4378  2689     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0 14433 16129 16024  8360     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0  8147 16129 16129 16129 15854     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0  9342 16129 16129 16129 16129  3859     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0  3474 15825 16129 16129 14285     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0  3745  9976  9424  2235     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0  5646 10113  9169     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0  3902 16035 16129 16129 12912     0     0 10354 14262 10990     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0  8521 16129 16129 16129 16129  2976  6444 16129 16129 16129 12138     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0  7529 16129 16129 16129 16072  2892  9065 16129 16129 16129 15023     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0 13549 16129 16129 10831     0  6558 16129 16129 16129 13988     0     0     0     0     0  2490
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0  4861  4598     0     0     0  9487 14870 12619  4159     0     0     0     0     0  6094
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0  3982 11421
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0  7917 14053
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0  2584 13306 15541
```

(a)



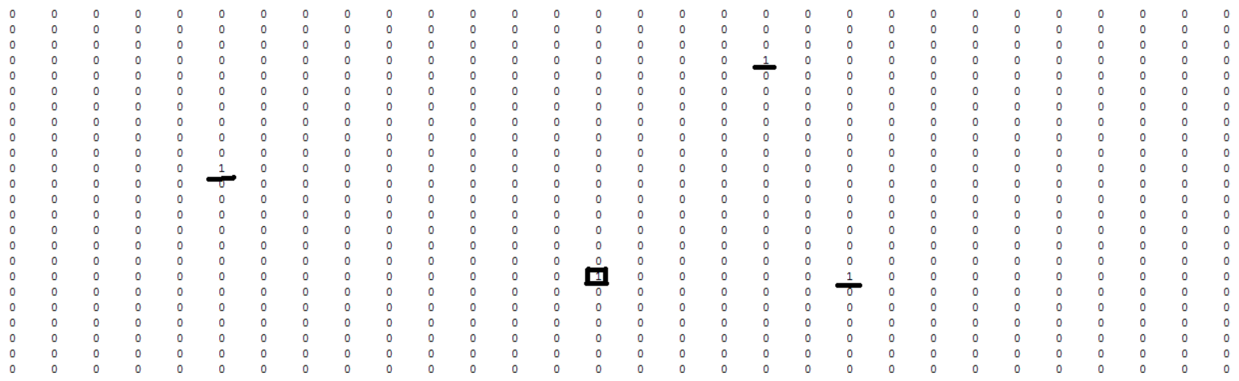
```
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0  3322  7665  5506     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0  4920 15917 16129 16128 11250     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0 13215 16129 16129 16129 16127  4848     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0 15273 16129 16129 16129 16129  7492     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0 10741 16129 16129 16129 15923  3666     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0 11460 16086 15389  7838     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0  4378  2689     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0 14433 16129 16024  8360     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0  8147 16129 16129 16129 15854     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0  9342 16129 16129 16129 16129  3859     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0  3474 15825 16129 16129 14285     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0  3745  9976  9424  2235     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0  5646 10113  9169     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0  3902 16035 16129 16129 12912     0     0 10354 14262 10990     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0  8521 16129 16129 16129 16129  2976  6444 16129 16129 16129 12138     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0  7529 16129 16129 16129 16072  2892  9065 16129 16129 16129 15023     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0 13549 16129 16129 10831     0  6558 16129 16129 16129 13988     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0  4861  4598     0     0     0  9487 14870 12619  4159     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
```

(b)



```
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1  1  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1  1  1  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1  1  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  1  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  1  1  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1  1  1  0  0  1  1  1  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1  1  0  0  0  1  1  1  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1  0  0  0  1  1  1  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

(c)



(d)

Figure 7 the output of clustering algorithm to count the circle numbers

3.2.3. Get the colony size in pixels

This can be done with the API function bwarea() provided by Matlab. It estimates the area of the objects in binary image BW using pixel as the measuring unit. It will only count pixels that that has value 1 (note that we have already converted the image into black-white format).

3.2.4. Get the scale factor and convert the size into square millimeters.

In this project, one of the most important questions to ask is: what is the real size of the colony? Diverse filming conditions and different image processing tools may affect the scale of colony images. It is therefore unreliable to use a fixed relation to calculate. In reality, the staff member who takes the pictures is required to include a horizontal line indicating the length of a millimeter in each image. Therefore, the real size of the colony may be calculated by the following formula:

$$Real\ Size\ of\ a\ colony = \left(\frac{Number\ of\ pixels\ in\ colony}{Number\ of\ pixels\ in\ the\ 1mm\ scale}\right) \times 1\text{mm}$$

Note that the real size in the formula refers to length or width only, not the area of the colonies. The 1mm horizontal bar is green, and may appear at any position in the image. Above the green bar, there are often some words, such as "1 mm", also in the color of green. The arbitrary location as well as neighboring words brings the most challenging part in recognizing accurate length of the 1mm bar.

We implemented this approach in our program in the following steps:

(a) Convert the image to green scale. In other words, remove the red, and green in the RGB colors. This step ensures the human operator can also observe and evaluate the transformation result.

(b) For each horizontal row in the image, we record the number of colors that similar to "white". By observing all current images, we considered a white pixel if the color index of that pixel is in [252,255]. Note that "white" pixels we discussed here are actually green pixels before step 1.

(c) Among all horizontal rows, we obtained the length of green bar by consider the row with the most white pixels. After that, we calculate the area of the colony by its real size which is inferred from the equation mentioned above.

3.3. Properties of output

The output of our program is in square millimeter, which is the calculated average colony size in the given blood-agar-plate image. A sample output report can be found at http://www.madmed.net/.

Conclusion

Here we report a highly customized blood-agar-plate reader for measuring sizes *Streptococcus pneumonia* colonies. The program function and report average colony size reliably if the input images are acquired as mentioned above. The complete code of our program together with various material is available in the http://www.madmed.net/.

References

1. World_Lung_Foundation. *Total number of deaths from pneumonia among children under five 2008*
2010 [cited 2010 November 19]; Available from: http://www.ariatlas.org/maps?id=0003.
2. [Anon], *Pneumonia and Influenza Death Rates - United-States, 1979-1994 (Reprinted from Mmwr, Vol 44, Pg 535-537, 1995).* Jama-Journal of the American Medical Association, 1995. **274**(7): p. 532-532.
3. NEUTECGROUP. *Flash & Grow - Economy Automatic Colony Counter*. 2010 [cited 2010 November 19]; Available from: http://www.neutecgroup.com/ecc.htm?gclid=CK_n45ThpqUCFQG6Kgod7wm9Ig.
4. ASPIRE_SOFTWARE_INTERNATIONAL. *Agar Plate Colony Counting and Size Determination*. 2010 [cited 2010 November 16]; Available from: http://www.aspiresoftwareintl.com/html/sigmascan_agar_plate_colony.html.
5. UVP. *Colony Counting Just Got Easier*. 2010 [cited 2010 November 16]; Available from: http://www.uvp.com/colony.html?gclid=CK3r7JPhpqUCFULNKgodgSpwYg.
6. Michelle Cayouette, J.M., Michael Steege, Alan Orr. *Counting Colonies Or Plaques Directly From Images Of Agar Plates*. 2010 [cited 2010 November 16]; Available from: http://www.biocompare.com/Articles/ApplicationNote/52/Counting-Colonies-Or-Plaques-Directly-From-Images-Of-Agar-Plates.html.