Image Processing Technique for Automated Viral Plaque Counting

Michael Moorman, Hood College

I. Abstract

The viral plaque assay is a widely used and important procedure in biology for measuring virus titer - the concentration of viruses in a given sample stock. Though effective and reliable, this method has long relied on humans to subjectively and painstakingly count instances of viral plaques. The primary goal of this project is to devise a computerized method that automatically processes viral plaque assay images and renders counts for each sample. The technique consumes negative images of viral plaque assays that have been generated by a standard flat bed scanner. The sample cells used in the study have been stained to enhance visibility of the viral plaques. In this research, parameters are fixed for plate size, number of plates per platter, and image dimensions. Experimental results are promising for the method, with accuracy rates measured at about 90 percent for most viral plaque images. Further research is necessary to investigate its overall viability for more generalized viral plaque counting tasks.

II. Background

The basic idea behind the assay is to use several different dilutions of a given virus stock to inoculate a thin monolayer of susceptible animal tissue samples in petri dishes. These inoculated dishes are then topped off with a special agar that limits viral expansion to only neighboring cells.

Each sample is allowed to incubate for some amount of time, allowing the virus to attack the animal cells. The virus will gradually destroy portions of the cells, creating the viral plaques. Their visibility is enhanced with the usage of special dyes.

The titer of a virus stock can be calculated in plaque-forming units (PFU) per milliliter. To determine the virus titer, the plaques are counted. Each dilution is plated in triplicate to enhance accuracy.

III. Technical Approach

The overall goal for the technical approach is to identify the count of viral plaques for each plate on a JPG format platter image. This goal is divided into two sub goals: region of interest (ROI) extraction, and actual viral plaque counting. ROI extraction aims to demarcate each plate into its own sub region while zero masking areas of the image that are not relevant to the plates. In the ideal case, each plate on the platter is divided into a ROI with all areas beyond the edges discerned with a zero pixel mask. The second sub-goal is obvious: count the instances of viral plaques within each plate.

Segmentation of each plate into a small ROI is an essential step for any conceivable viral plaque counting solution. This phases addresses several issues with the plaque image and allows each plate to be processed later in a faster, more meaningful way. Additionally, segmentation provides a means for evaluating the feasibility of further processing the image data. For example, the program can know to halt if segmentation yields only 20 plates when the program expects 24 to be found.

Perhaps the most glaring problem that segmentation resolves is the platter orientation problem. When a technician scans the image of the platter, there are no guarantees as to the postion of the platter with

relation to the scanner surface. Platters may be butted up against the top left corner of the scanner, have slight offset on any axis, or have some non-zero angle orientation about them. A global solution is needed to address this problem. Furthermore, it needs to be robust enough to handle subtle variations in brightness, contrast, and of course viral plaques for all possible input data.

The technical approach for segmentation is as follows:

1) Grayscale the image
2) Otsu threshold into a binary image.
3) Dilate the image enough to close 'viral growth' holes.
4) Aggressively erode the image. Only the centers of the strongest pixel groupings will survive this erosion (the plates). Since the size of the plates is known, it can be done reliably. Iterative erosion operations using a smaller kernel are preferred to using one iteration of a large kernel.
5) Extract the center points of what's left behind by calculating the centroid for each contour that remains. These are the seeds for the next stage.
6) Grow the regions outward from the seed location using another threshold with a Flood Fill.
7) Calculate the centroids of the newly grown regions found in 6. This is the estimated center of the plate with error resultant from plaques residing on the edges of the plates.
8) Draw a perfect circle at each center found in 7. The radius of these circles should reflect the ideal plate size of the actual plate minus some error constant.

Segmentation produces clean regions of interest for each plate in the platter, however, the task of counting each viral plaque still remains. Given the noisy nature of the substrate and viral plaques therein, it's difficult to devise a method that is perfectly accurate for all varieties of input. Viral plaques can be very small, very large, malformed, or even overlapped on top of one other. Never the less, the plaque counting method for this project attempts to be simple and accurate for the most commonly occurring instances of viral plaques. It is as follows:

1) Let a masked region of interest be image A. Ideally, this is a color image containing a single plate. The mask covers all areas beyond the edges of the plate.
2) Grayscale and then Otsu's threshold transform image A.
3) Let images B,C, D, and E be zeroed copies of image A.
4) Examine every contour in A. If the contour area is larger than some constant, redraw the contour onto image B. The constant used for this study is 5 pixels.
5) Perform an imfill() (Matlab) type operation on image B to close any internal holes in the contours.
6) Do dilation morphology on image B. This study uses a 3x3 kernel as a structuring element.
7) Copy image A to image C. Apply a morphological open operation on image C.
8) Examine every contour in C. If the contour area is larger than some constant, redraw the contour onto image D. The constant used for this study is 25 pixels.
9) Binary AND the images B and D, store the result on image B. B will now contain contours that will have an outer component and one or more inner components.
10) Count the contours using the following scheme:

```
totalPlaques = 0;
For each outer contour
  totalPlaques += 1 + max(0, innerContourCount - 1)
```

IV. Experimental Results

Experiments demonstrated promising result for this study. A total of 5 plaque images were tested against the viral plaque counting program. The output of the program was compared against the truth data on a by image basis. The absolute difference of the viral plaque totals vs truth data totals was sampled. On average, the program yielded about 90 percent accuracy with respect to this metric.

The speed of the program is satisfactorily swift, with experiments demonstrating a processing time of about 1.5 seconds per image. Hardware used for this study was an Intel i5 quad core processor.

V. Conclusions

The viral plaque counting method does a very good job segmenting each plate into a clean ROI. This is because post Otsu threshold and dilation, the general shape of images is quite similar. Plates appear as most prominent, filled, circular areas. Aggressive eroding exploits this fact, and ultimately provides all important seed points for further processing to take place.

The actual plaque counting routine attempts to find a balance between finding faint and sparse plaques vs finding prominent and heavily grouped plaques. This is accomplished by coalescing dilated and eroded variants of the ROI.

VI. Future Work

The next natural steps for the project are:

1) Investigate the feasibility of the program for the general viral plaque counting case. More experimental data is needed for this.
2) Improve upon the actual plaque counting method within the program. A more sophisticated plaque identifying technique is likely necessary to improve the accuracy of the routine.
3) Parametrize the program so that it can accept various platter types, image dimensions, and plate diameters. Supporting different formats will improve the overall usefulness of the program.
4) Enhance the program with the addition of a confidence indicator. Using some metrics, perhaps some defined by the user, the program can provide the user with a level of confidence for a given result set. If it is beyond some threshold, report an error.