**Counting Gold Nanoparticles**

By Bram Koobs and Michiel Meeuwisse

**Introduction**

Gold nanoparticles are a commonly used imaging technique. They are attached to a binding protein so that it becomes easily visible under an electron microscope. This way you are able to see how many of the binding proteins bind themselves to the surface of a bacteria. In this particular case images are made of bacteria that are genetically modified to have either their receptor genes turned on or off. In order to see how this effects how well the binding proteins bind to the genetically modified bacteria gold nanoparticles are attached to the binding proteins in order to see how many of them bind to the surface of the bacteria. The goal of our project is to create a web application that automatically counts the amount of gold nanoparticles on the surface of a bacteria. And displays images of the found gold nanoparticles.

**Approach**

In order to count the amount of gold nanoparticles in the given images we have created a web application. This is a detailed step by step breakdown of the pipeline behind the web application. The libraries we used are CV2, Numpy and Matplotlib. CV2 is the backbone of our pipeline. We use it for all our image processing steps and to eventually find the nanoparticles in the processed image. Numpy is used for certain numerical operations like calculating the median of a list of numbers. Matplotlib is used to show intermediate or final images. This is mostly used for testing and debugging. Our pipeline accepts a folder of images to be processed and will process these images one by one.

The first step in our pipeline is to find the cell. This is done by applying a heavy blur to the image to cancel out the effects of heavy noise. Then the image is thresholded on a custom brightness. This brightness depends on the average brightness of the entire image. After this you are left with a black and white image that only contains the cell. CV2 is used to draw a contour around the cell. This contour is later used to check if found dots fall within the area of the cell.

After this the image is preprocessed for finding the nanoparticles. This also starts with a blur. But this blur is a lot less strong than the blur that is used for finding the cell. This is so that minor noise will be eliminated without eliminating the dots themselves. After this an adaptive threshold is used on the blurred image. This is different from a regular threshold because it checks a single pixel and looks if it is a significantly different color from a small surrounding area. This finds most of the dots because they are usually significantly darker than the surrounding area. However, because the adaptive threshold checks every individual pixel it creates a very noisy image. To eliminate some of this noise another slightly stronger blur is applied. On this blurred image a regular threshold is applied. This removes some of the noise while only minimally reducing the visibility of the nanoparticles in the image.

On this preprocessed image CV2 is used to find coherent shapes. Some of these shapes will be the nanoparticles we are looking for. But initially it is only a small percentage of the found shapes. In order to eliminate most of the shapes that are not nanoparticles some filters are applied to the found shapes. First all shapes that are below a given minimum size are removed. After this all shapes that are not circular enough are removed. This removes the majority of the shapes that are not nanoparticles. However, some of the removed shapes will actually be clusters of nanoparticles that are overlapping. In order to count the number of nanoparticles in such a cluster. The Size of the shape is divided by the average size of a nanoparticle. This generally gives a good approximation of the correct number of nanoparticles in a cluster. In order to make sure less shapes that are not clusters of nanoparticles are counted as clusters. The potential clusters have to be darker than a given threshold. This eliminates a lot of slightly darker smudges that can be found in the cell.

After all of the shapes that are likely nanoparticles are found they are cross referenced with the earlier found cell. If the nanoparticles are within the boundaries of the cell, then they are counted towards the total. This leaves you with a number of found nanoparticles.

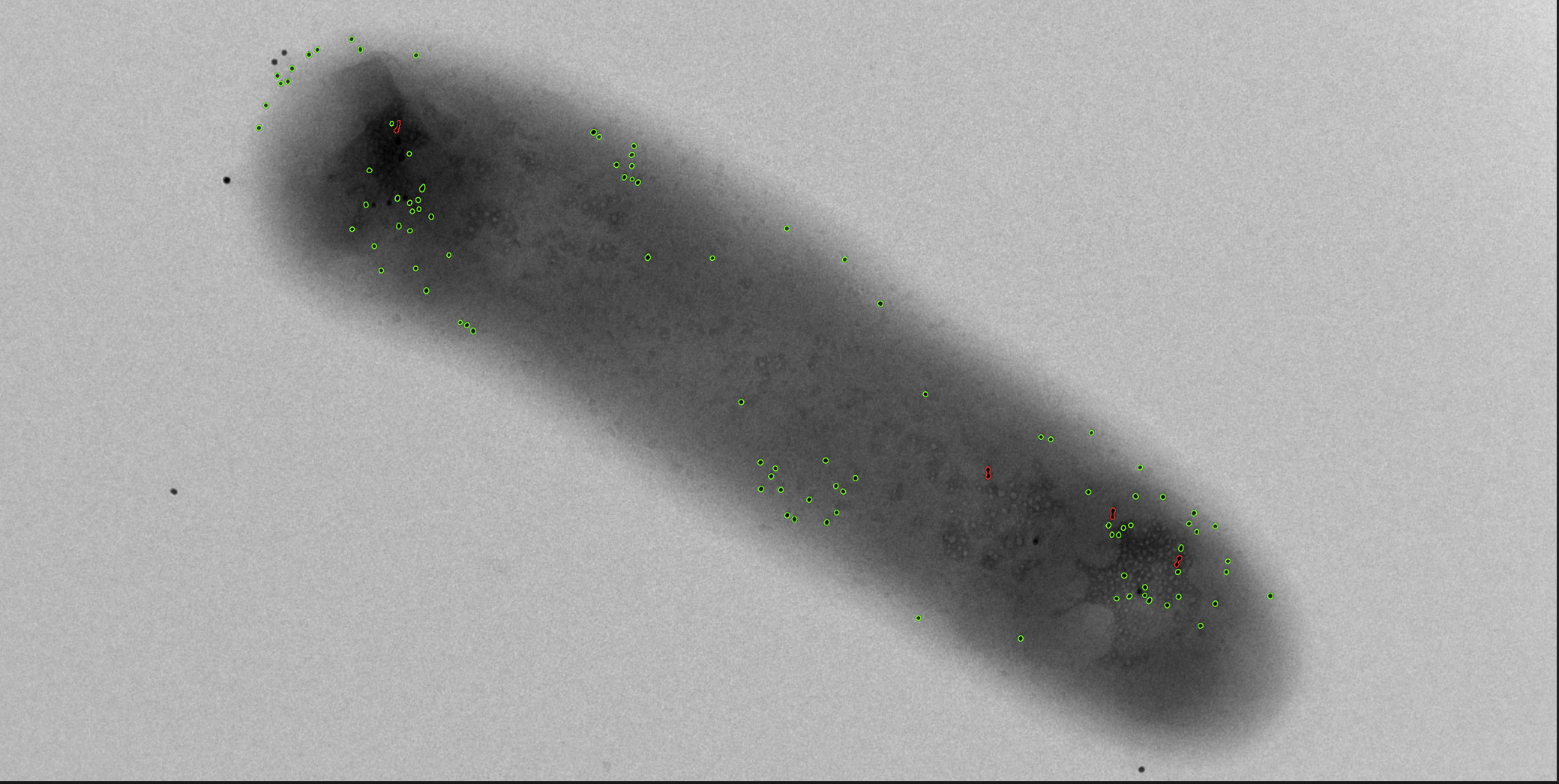
After all the nanoparticles are found their contours will be drawn onto the original image which will then be displayed on the web application together with the number of found nanoparticles.

**Preliminary results**

The backend of the application, if the settings are right, is already working nicely. Most dots and cells are found. In the example images below there are no dots that are found in places where there shouldn’t be (false positives). And clusters of dots (the red outlines) are found quite reliable.

Dots are only counted if they meet certain criteria: The dots can’t be smaller than a certain size, they must possess a certain circularity and they exist within the boundaries of the cell itself.

Unlike earlier versions, the edge of the cell is not seen as an area with high colour difference, which had previously resulted in some false positives.



**Discussion and outlook**

The biggest issue with the current application is that the settings are not well suited to all images. The settings have to be manually tweaked between images to get the application to recognize the dots. The current plan is to train a neural network to predict the best setting given some information about the image. The planned inputs for the neural network will be:

* Zoom
* Resolution
* Average colour/grey value of image
* Most common grey values in image (spikes in amount of that shade)

The settings that we hope the network will be able to predict are:

* A threshold value for finding the cell
* The size of the smallest gold nanoparticle
* Settings for the adaptive threshold
* Values for the thresholding and blurring for finding the particles

I’m not sure the neural network will be able to predict these settings from the relatively simple inputs. If this is not the case, the model will either have the image itself as input, resulting in a much slower application. Or we will have to attempt make formulas that decide the settings, which we have not been able to achieve up until now.

The next big step in the development of the application will be the front-end. It’s a new step in the development of the application, so current progress is not that far.

A minor issue we are yet to fix is that some dots in the darker areas of the cell are failing the circularity test. A possible solution is to detect the surrounding areas' greyness and make circularity less important based on its result.

Collaboration with the client has only happened at the start of the project. Currently we are attempting to plan the next meeting. We want to share our current progress and decide together on how to move forwards with the next steps of development.