Process for aLigo Particle Imaging Using a Nikon D7100 Camera and MATLAB

Sabrina Waller

April 2, 2015

Purpose:

The design for LIGO (Laser Interferometer Gravitational Wave Observatory) centers around the use of high intensity lasers aimed at a series of optics to detect gravitational waves. However, when the laser beams hit the optics, particulate contamination on the surface of the optics results in interference with the beam and damage to the optics. Therefore, there is a need to be able to efficiently and accurately identify the contamination level of the optics within Advanced LIGO. Toupview, the program currently used to process images and measure the pictured dust particles, forces the user to go through every step of processing the image by hand. The development of a straightforward, automated process is essential for enabling quick, accurate determinations of what optics need cleaning and if there is an unacceptable level of contamination remaining after cleaning.

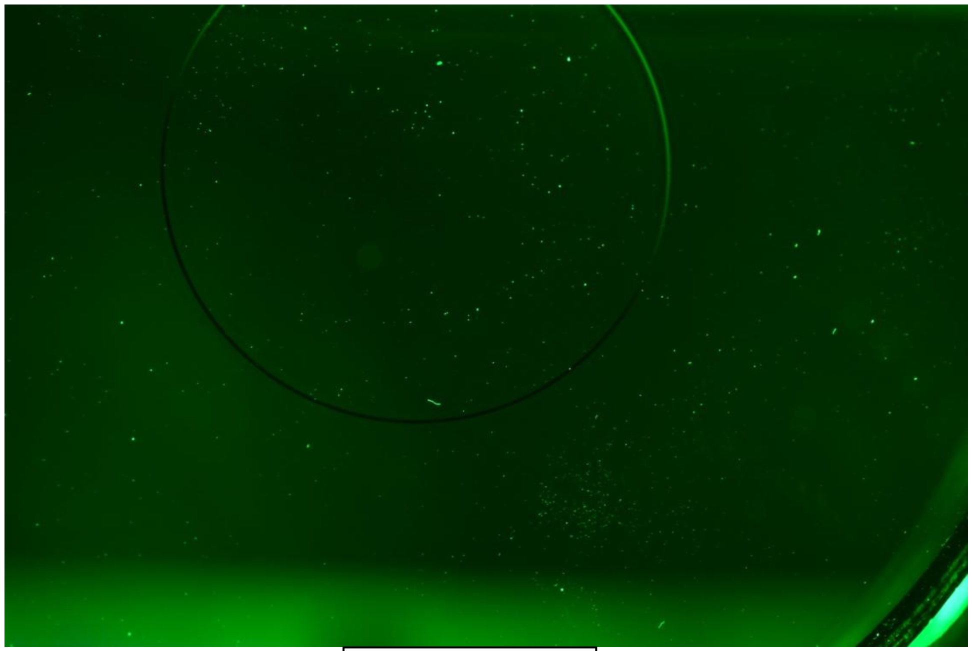
To meet this need, I developed the following process. The test camera I used was a Nikon D7100 with an AF Micro Nikkor 200mm 1:4 D lens and a Nikon TC-201 2x Teleconverter , but this process can be used for any camera. I wrote code to process the images in MATLAB utilizing MATLAB's Image Processing Toolbox. The program counts the number of particles and measures their various sizes, providing immediate information about the contamination on the optics - with the test camera, MATLAB detects and measures the particles on the images down to 10.0066881 microns with a maximum margin of error of 4 microns.

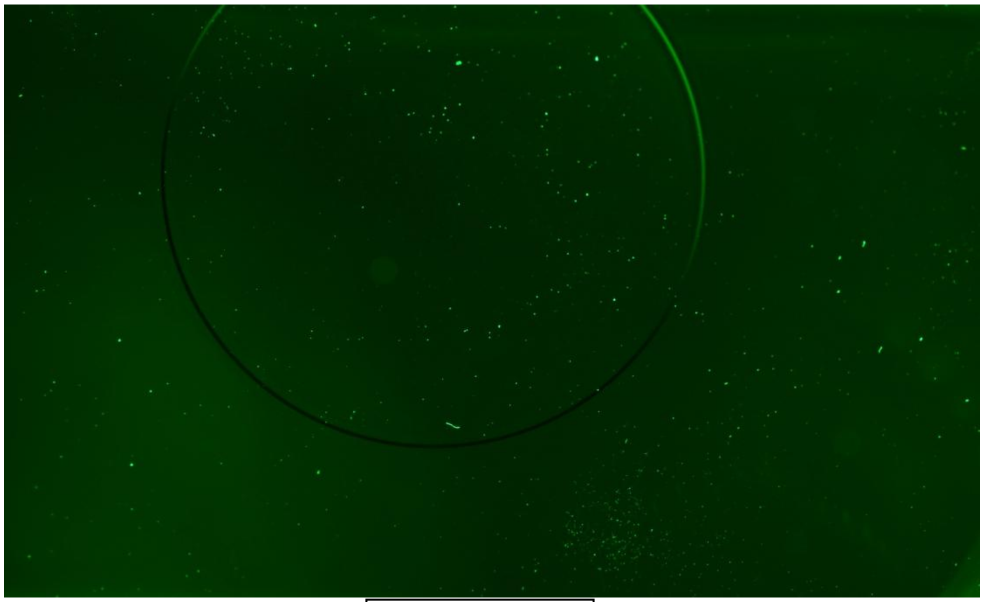
The following process is based on tests images of a 3 inch diameter optic surrounded by a ring of green LEDs that illuminate the dust on the surface of the optic, all of which is inside a chamber.

A detailed account on early development of this process can be found in my two progress reports, the first of which is located in the document LIGO-T1400474-v1 and second of which is located in the document LIGO-T1400506-v1.

<u>Processing Images</u>

1) Open the image in Paint and crop the image so only the optic itself is in the image; no part of the ring of green LEDs should be present in the final image because it will make it harder to threshold the image and will throw off the particle count.


Original Image


Cropped Image

2) Open the folder containing this image in MATLAB. Make sure that you have MATLAB Image Processing Toolbox Version 9.0 or greater because previous versions do not have the necessary functions for processing the images.

3) Open the MATLAB code to process the image and count the particles, titled *Matlab Code for Processing Images* that accompanies this document on the DCC.

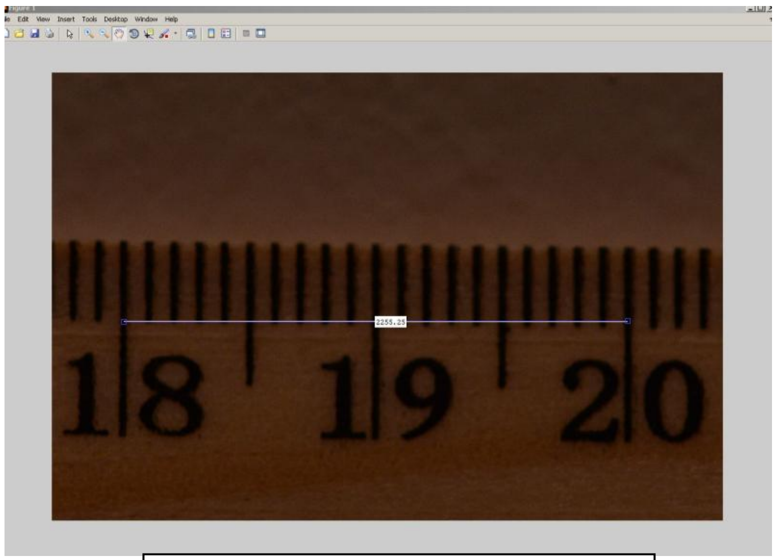
4) The code uses the pixel/meter ratio of the camera to calculate the particle size, so when changing cameras, these numbers will need to be calculated and changed in the code as follows:

    1) Tape a ruler to the wall or other flat, vertical surface and set up the camera at a distance from the ruler equivalent to the intended distance from the optic. Take an image of the ruler and then insert the name of the image into the following code, which also accompanies this document on the dcc:

    ```
    %read image
    rgb = imread('name of image');
    figure
    imshow(rgb)

    %distance line
    d = imdistline;
    ```

    2) The code will generate a distance line you can move to measure a specific distance in pixels. Place the distance line across the ruler and record the number of pixels there are over a specific distance. The larger the distance, the more accurate the ratio will be, so choose a distance of at least 20 mm to measure.



Using Distance Line to Measure 20 mm in Pixels

3) Convert the distance in millimeters to meters and divide the number of pixels the distance is by the number of meters the distance is. This will give you a pixel/meter ratio that you can swap with the pixel/meter ratio written into the code for processing the images and counting the particles in the following lines.

```
%finding areas of particles in mm^2 and microns^2
area_mm = (areatemp/12715381410)*1e6; %pixels to mm^2
area_microns = (areatemp/12715381410)*1e12; % pixels to microns^2
```

Replace the highlighted numbers with **the square of** the new pixel/meter ratio

```
%finding perimeter of particles in mm and microns
perim_mm = (perimtemp/112762.5)*1e3; %pixels to mm
perim_microns = (perimtemp/112762.5)*1e6; %pixels to microns
```

Replace the highlighted numbers with the new pixel/meter ratio

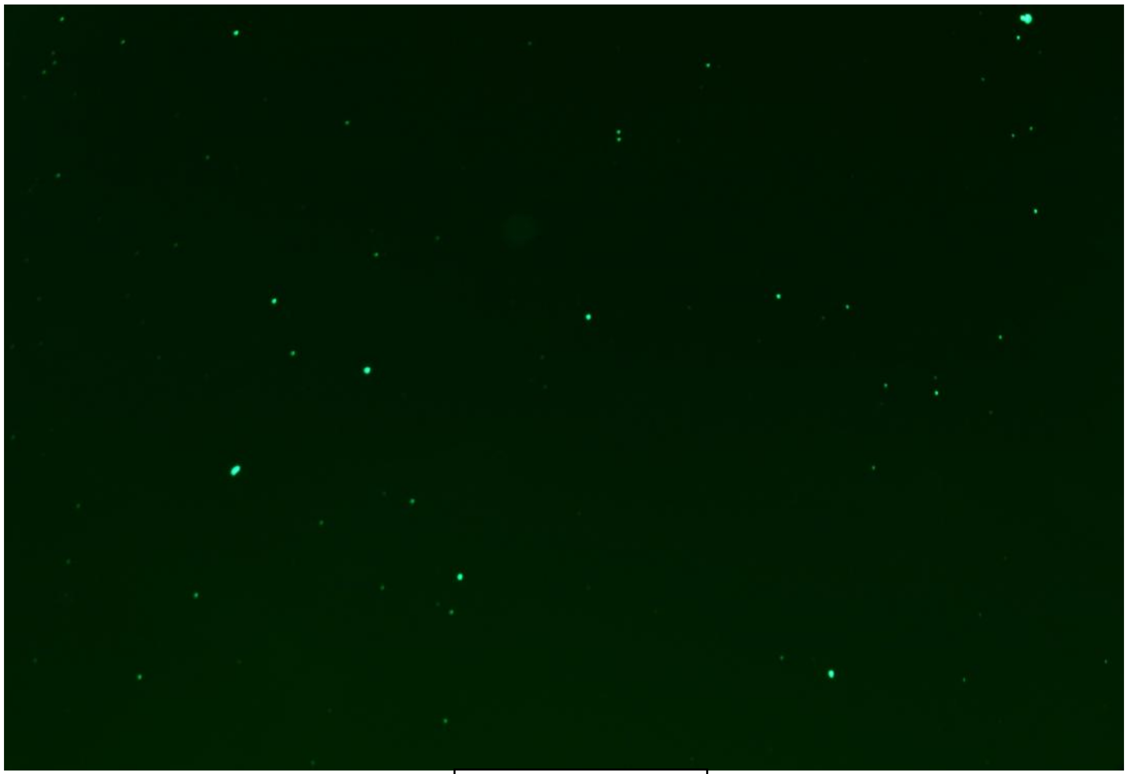
5) Insert the name of the cropped, original image into the following line of code, which reads the image.

```
%read image
rgb = imread('name of cropped image');
```

6) Next, adjust the threshold level the code uses. It is necessary to set this level by hand because each image requires unique settings; therefore, the automatic settings create inaccurate results. If the threshold is too high, MATLAB will read certain particles or sections of the particles as part of the background and color them black, effectively shrinking them or erasing them from the image. However, if the threshold is too low, MATLAB will read brighter parts of the background as particles and detect ghost particles that do not actually exist.
Adjust the highlighted value the following line to a number between 0 and 1 - the larger the number, the higher the threshold.

```
%greyscale to binary image
BW = im2bw(I, .15);
```

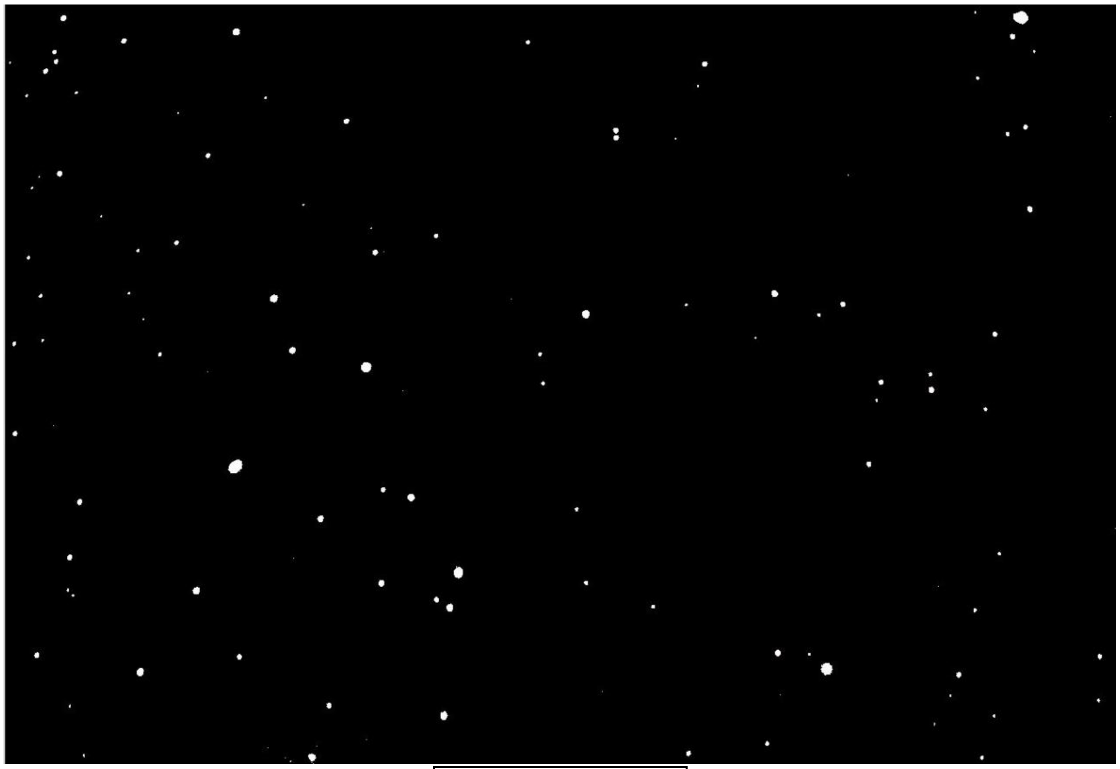
Here is an example of an image thresholded at different levels.



Original Image

0.15 Threshold Level

0.5 Threshold Level

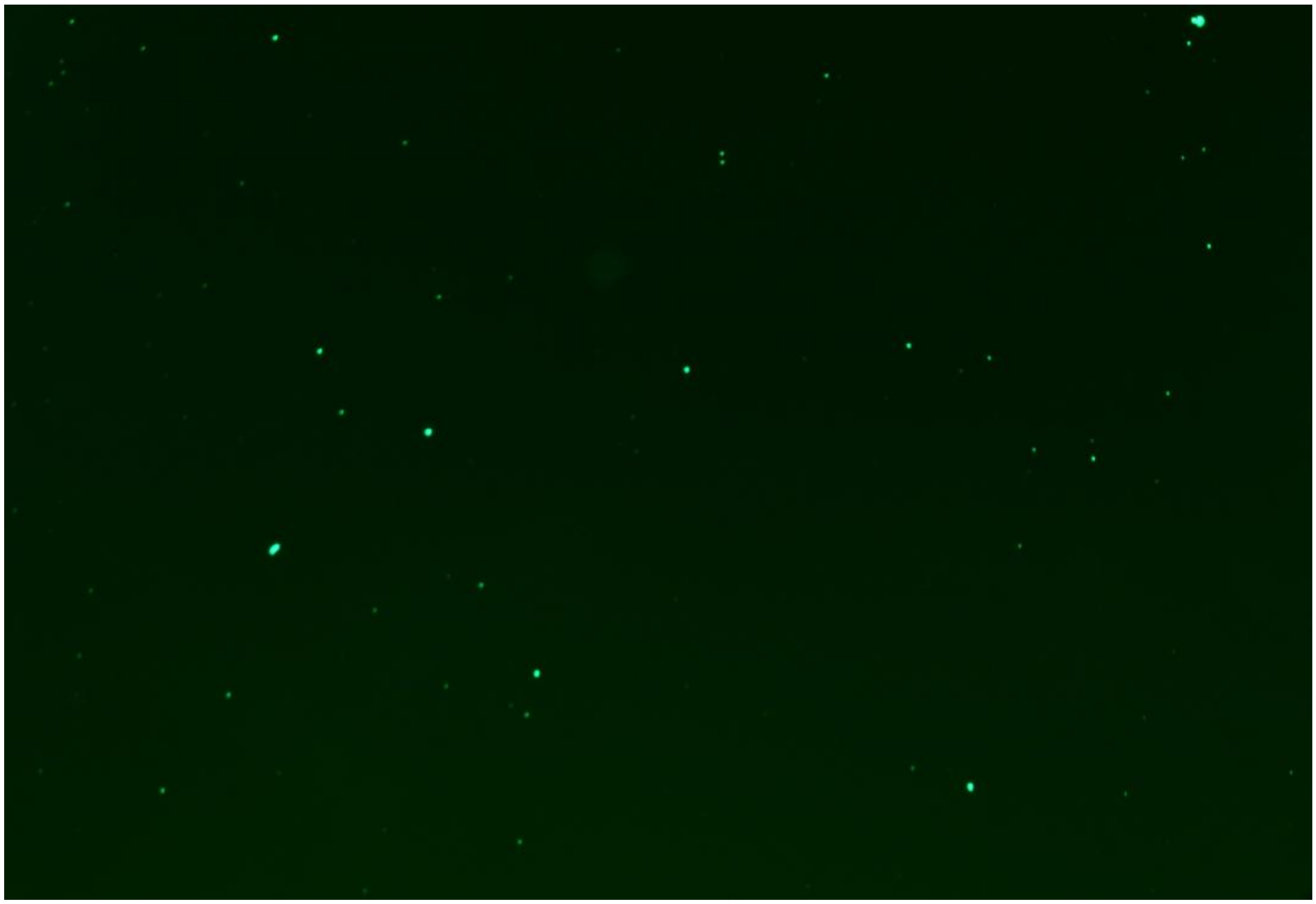

0.08 Threshold Level

By comparing the images processed at different levels to the original image, it is evident that the 0.15 threshold is the correct level for this image. By trying a variety of levels and comparing them to the original image, it is fairly easy to determine which one is right one to use.

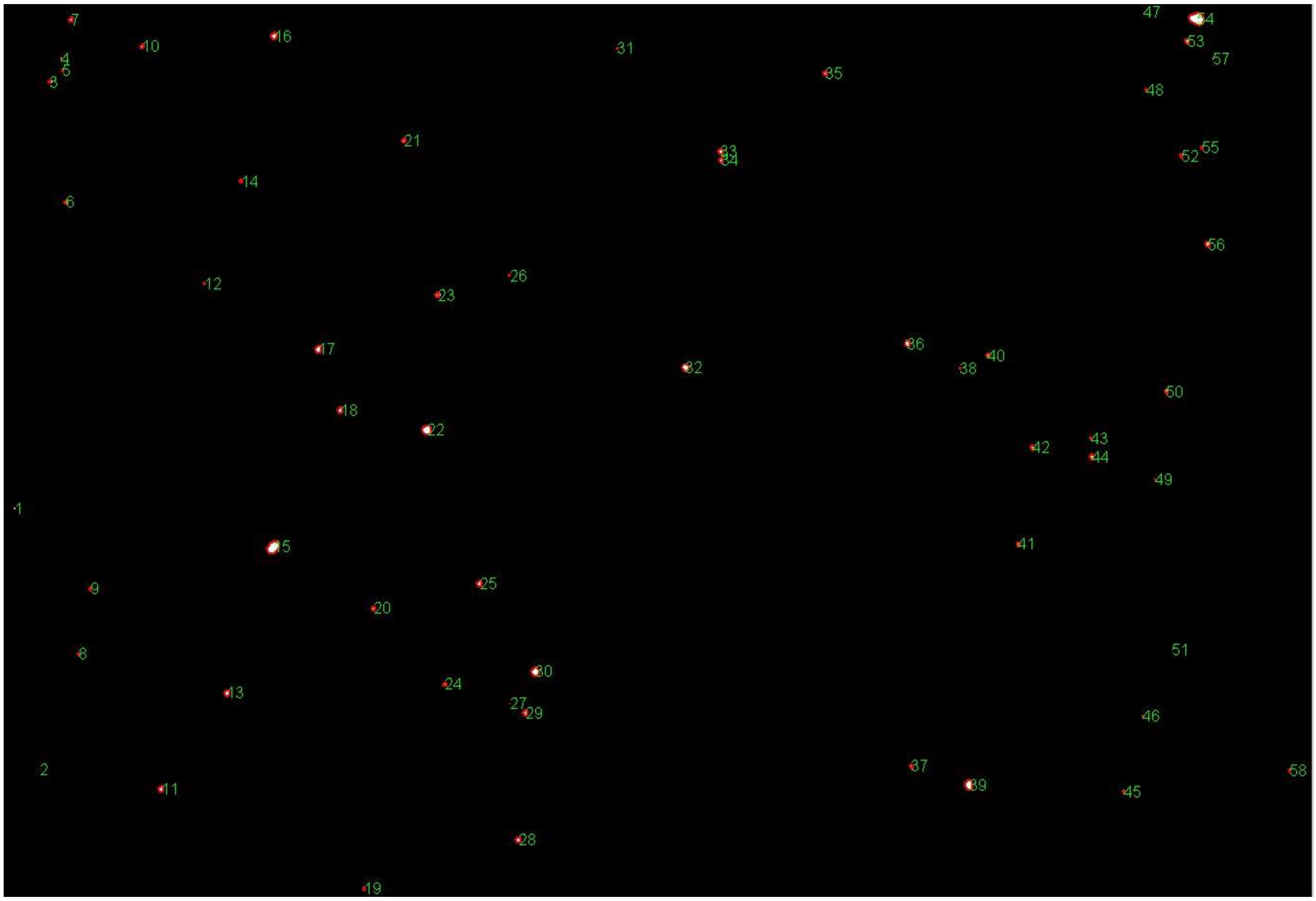
About The Results:

The code measures the diameter, perimeter, and area of each of the particles in microns and millimeters. MATLAB also outlines the detected particles, which allows you to see which particles MATLAB detects, and therefore, catch any errors in detection that may occur, such as if it misses particles, detects ones that do not exist, or under or over estimates their sizes. It also brings up both the processed image with the particles circled and tagged with serial numbers.

Shown below are the original test image and the processed test image.



Original Image



Processed Image

MATLAB outlines particles by drawing a line though the centers of the outer pixels; therefore, according to MATLAB, particles of the smallest diameter, which resolve as a single pixel, have a perimeter of zero. As seen in the image below, MATLAB's attempt to outline the particle resulted only in a single red dot in its center of the particle, which shows that MATLAB cannot find the perimeter of a particle only a single pixel in size.

The larger the size, the fewer particles there are of that same size, which occurs for several reasons. First of all, there could simply be more small particles than large particles on the optic, but it is also because there is a larger gap between particle sizes when they are smaller than when they are larger; in the test images, the smallest two sizes are 10.0066881um and 14.15159403um with a gap of 4.15 microns between them, while two of the adjacent larger sizes were 93.33617298um and 93.87105514um with only a .5 micron difference. Therefore, the system resolves the particles that are in these large gaps between the smaller sizes to the nearest pixel and places them in the category closest to their size, which inflates the number of pixels of that particular size. While this grouping does not mean the overall number of particles detected is incorrect, it does mean that the smaller particles may not necessarily be the exact size they are measured as, their true size within a couple of microns of the measured particle. The number of particles there are of the same size in the test image steadily dropped from 229 at 10.0066881 um to 72 at 24.51127986 um to 34 at 42.45478208 um and finally down to single digits around 60 um. Also, the gaps between the sizes went from multiple microns to around one micron at around 33 um and to fractions of microns at around 50 um. Therefore, while the lower sizes should not be completely discounted, it is important to remember that for particles smaller than 33 microns, the particles may, in reality, have a diameter within several microns of the size detected and for particles smaller than 50 um, the particles may have a diameter within a micron of the size detected. More information about the test results can be found in Progress Report 2 under document number LIGO-T1400506-v1 and the document titled "Data for aLigo Particle Imaging: Resolution of Nikon D7100 with 2x Teleconverter" under document number LIGO-T1400491-v1.

Related Documents:

LIGO-T1400474-v1: LIGO SURF Progress Report 1: Particle Imaging and Analysis for Advanced LIGO

LIGO-T1400506-v1: LIGO SURF Progress Report 2: Particle Imaging and Analysis for Advanced LIGO

LIGO-T1400494-v1: Data Comparing the Abilities of Matlab and Toupview to Detect Particles

LIGO-T1400491-v1: Data for aLigo Particle Imaging: Resolution of Nikon D7100 with 2x Teleconverter

LIGO-T1400467-v1: Data for aLigo Particle Imaging: Resolution of Nikon D7100

LIGO-T1300665-v9: The LIGO Particulate Evaluation Tool (PET)