Developing a GUI and Image Processing Tools for Lunar Dust Adhesion Testing

Student: Davin Baal

Mentors: Christopher Wohl

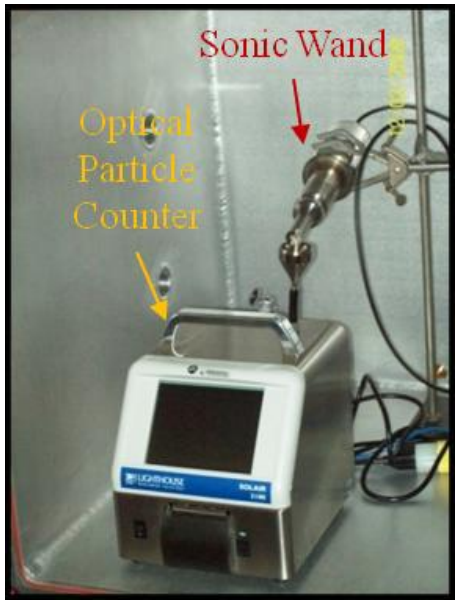
Lopamudra Das

2020 Fall NIFS Research Symposium

*All images in presentation are credited to NASA

Two Main Challenges

- Compilation and analysis of optical particle counter data for Lunar dust simulant adhesion characterization
- Image analysis of number of particulates and areal coverage of simulant on surfaces

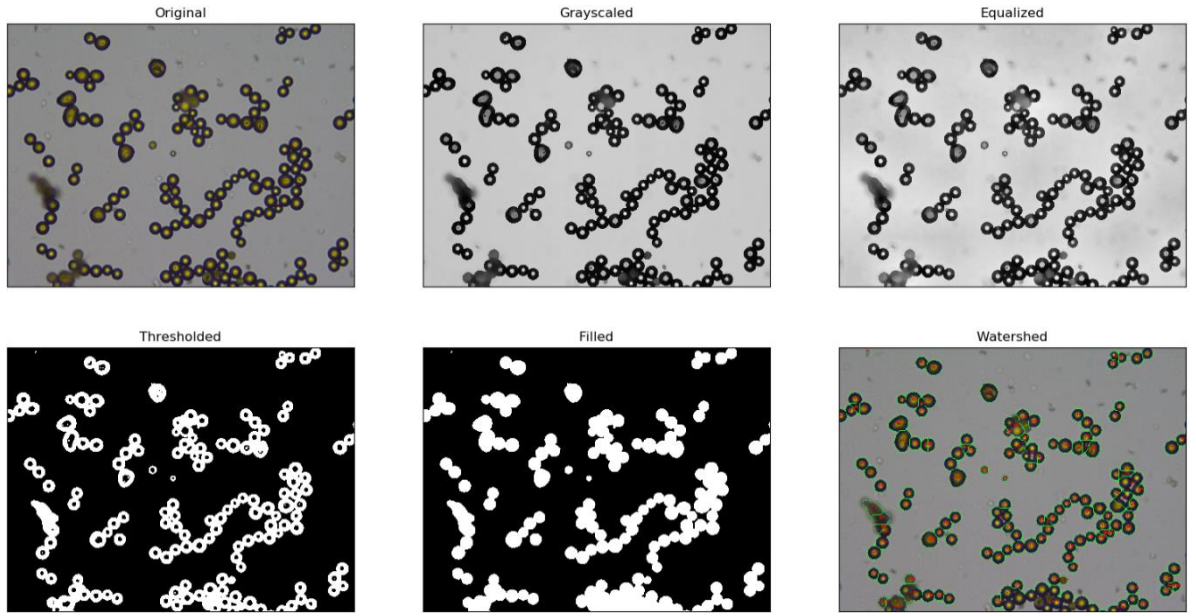


Optical particle counter and sonic wand
(From *Method and Apparatus for the Quantification of Particulate Adhesion Forces on Various Substrates* by Christopher J. Wohl, Brad M. Atkins, and John W. Connell (October 2010))

Microscope image of dust particulates
Image Credit: NASA

History of Challenges

- Optical Particle Counter
  - Prior code was developed to control the sonic wand and read data off of the optical particle counter
- Image Processing
  - ImageJ originally used, but it takes too long
  - Prior code was developed to measure and count dust particulates from images; however, it was only optimized for (roughly) spherical particulates



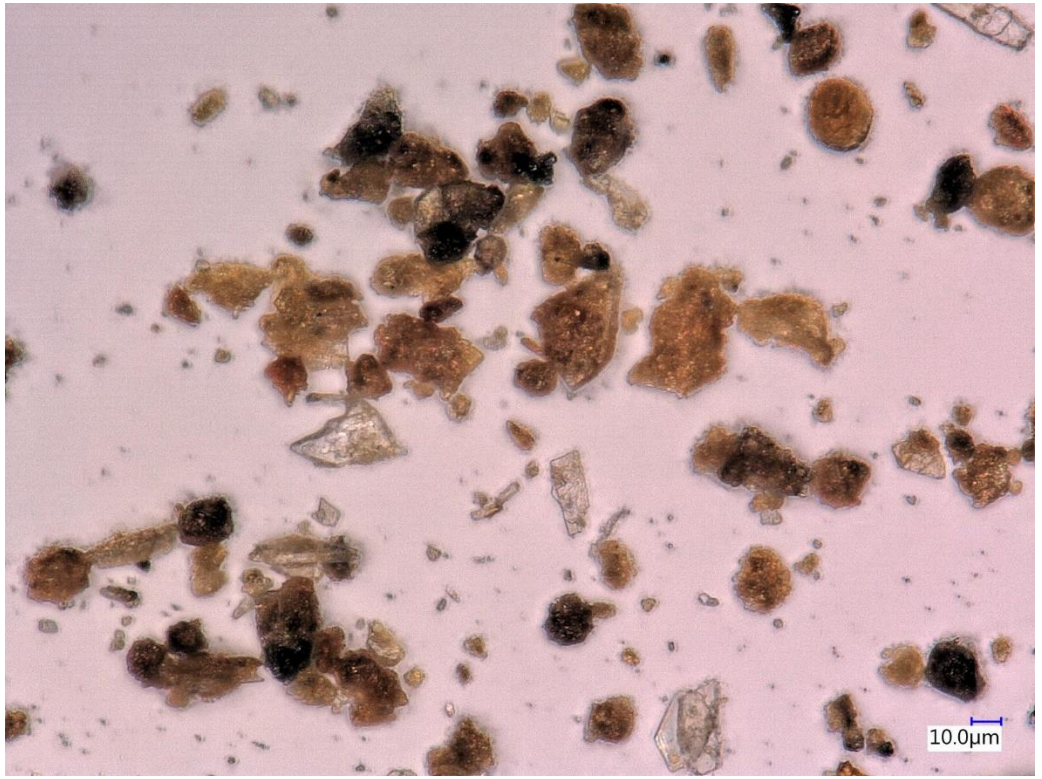
Prior image processing program counting and measuring (roughly) spherical particulates
(From *Technical Review Paper* by George Blackwell, Nicolas Fransen, and Dylan Lew (August 7, 2020))

Overall Objective

- The overall objective of the project is to measure the adhesion forces between various lunar dust simulant particulates and surfaces
  - Dust-covered sample material placed on tip of sonic wand
  - Sonic wand shakes at varying amplitudes, applying varying forces to the dust particulates
  - The amplitude at which the dust falls off indicates the adhesion force of the particulates
  - The optical particle counter counts and measures the sizes of the dust particulates that fall off
  - Microscope images of the dust on the sonic wand are taken to monitor how much dust has fallen off
    - SOURCE: *Method and Apparatus for the Quantification of Particulate Adhesion Forces on Various Substrates* by Christopher J. Wohl, Brad M. Atkins, and John W. Connell (October 2010)

Specific Objectives

- Graphical User Interface (GUI)
  - Develop a GUI using Python for plotting particle counts from the optical particle counter
  - Integrate this GUI with the rest of the optical particle counter code
  - This will provide a quick and easy way to visualize data from tests
- Image Processing
  - Enable the particle counting and measuring program to handle varying shapes and sizes
  - Make the image processing sequence as automated as possible
  - This will provide a quick way to count and measure dust particulates for testing

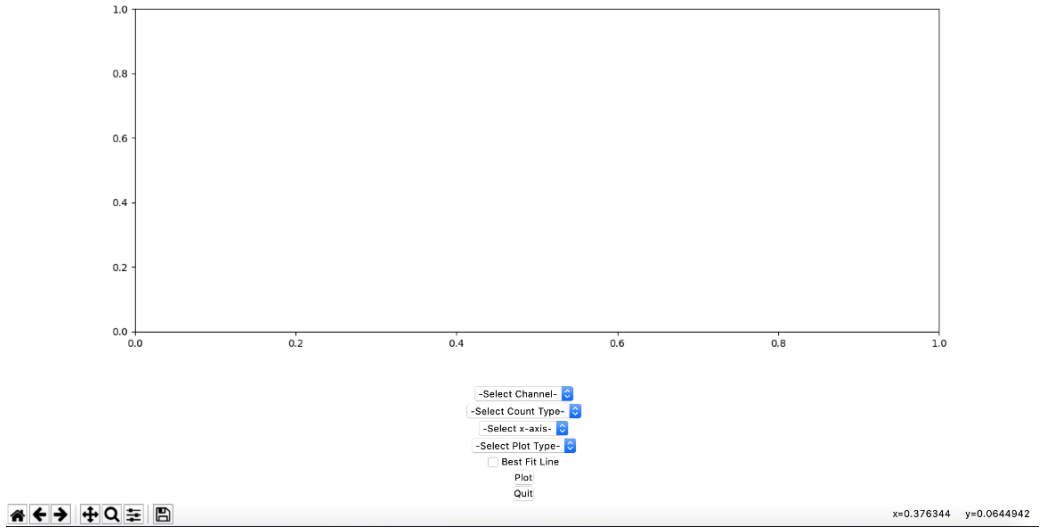


Dust particulates of varying shapes and sizes
Image Credit: NASA

Graphical User Interface

Approach

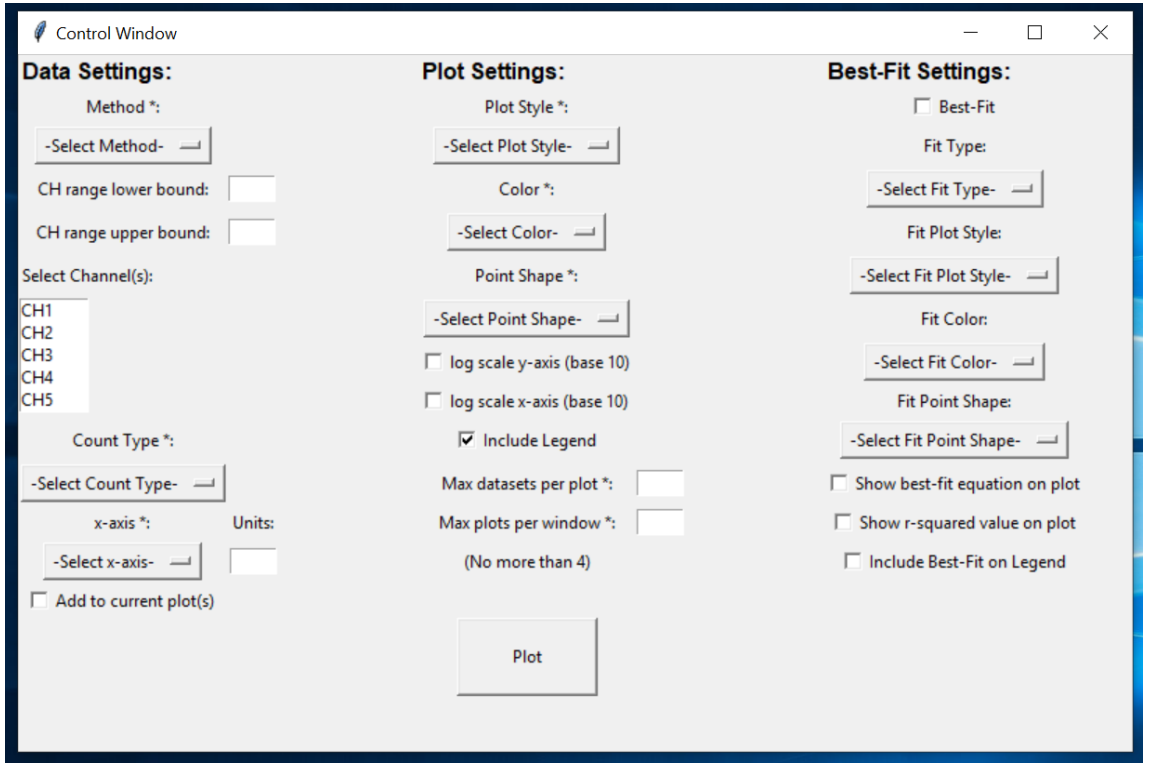
- First step was to just get a basic plot working in a GUI window (image on right)
- Features gradually added:
  - Best-fit
  - Plotting multiple sets of data at once
  - Displaying multiple plots in one window
  - A legend
  - Different datapoint shapes and colors
- Created ability to plot particle counts in many different ways, establishing an efficient way to plot and fit count data



Early version of GUI

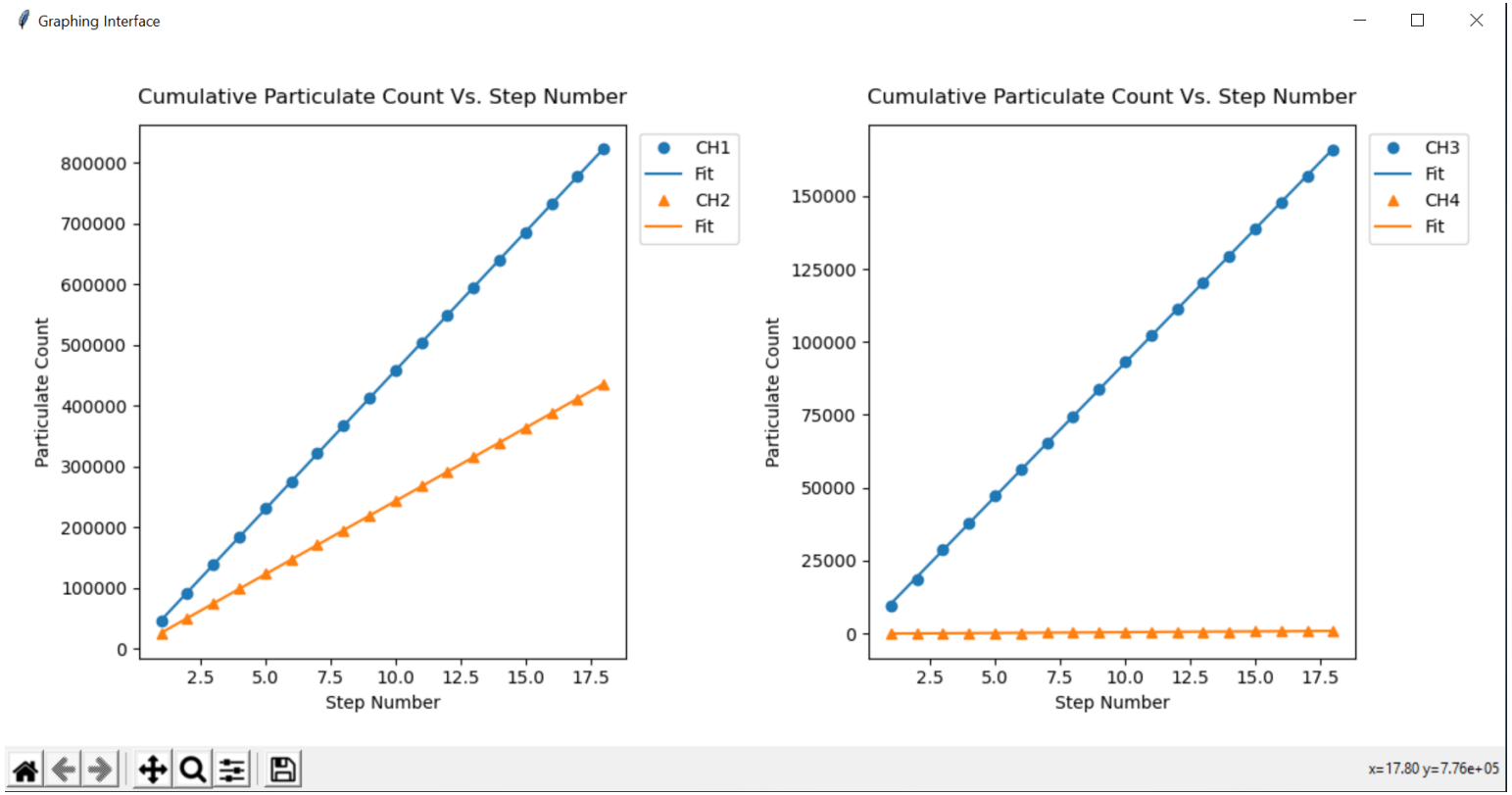
Results

- Final version of GUI allows for plotting all data from optical particle counter and sonic wand, including:
  - Particle count
  - Measurement number
  - Time
  - Sonic wand amplitude
  - Sonic wand voltage
- Also allows for:
  - Multiple datasets to be plotted simultaneously
  - Multiple plots to be plotted in one window
  - Automatic and manual best-fits



GUI Settings Window

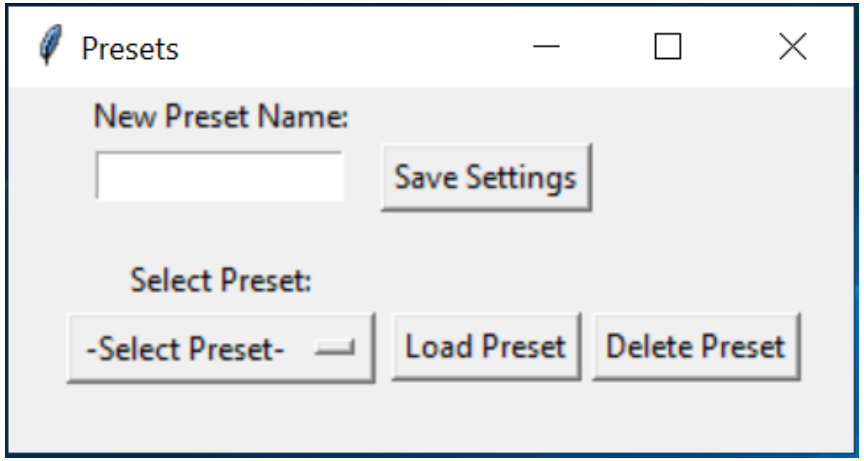
Results (cont.)



Plots of Cumulative Particulate Count vs. Step Number from sample data

Analysis/Summary

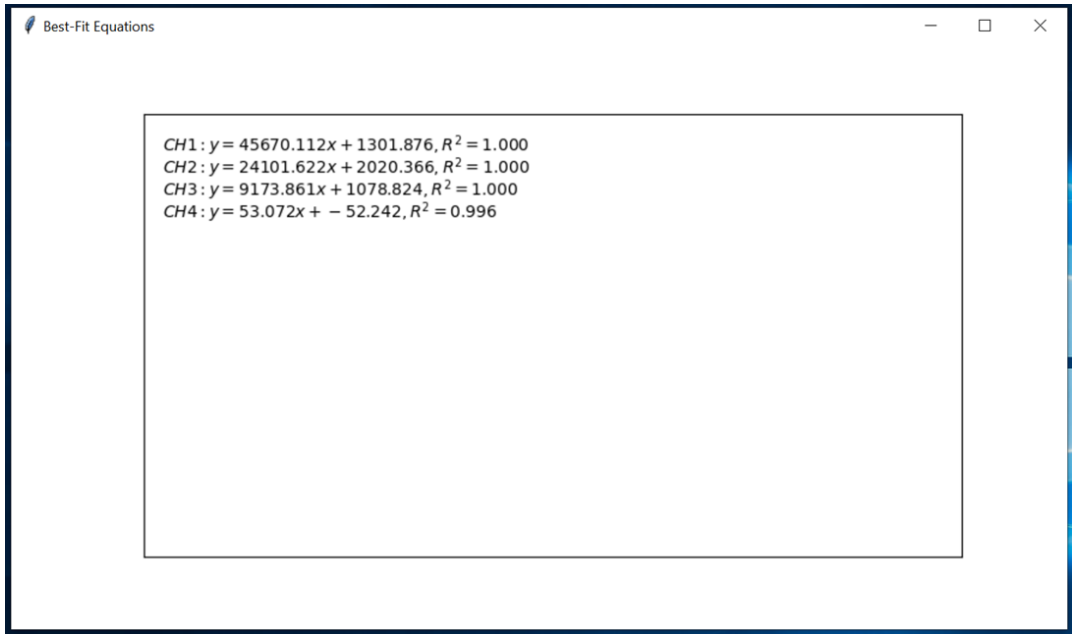
- GUI performs as expected
- Reads data files
- Calculates cumulative particulate counts and step durations
- Creates best-fit models
- Stores presets (image on right)
- Displays various sets of data in various ways



GUI presets window

Next Steps/Outlook/Future Work

- GUI would be a lot more valuable and versatile if different files could automatically work with the code
- All best-fit equations and r-squared values are displayed in a separate window
- Working on a way to get more equations and r-squared values on the plots themselves could be helpful



Best-fit equations from the GUI

Image Processing

Approach

- Review of existing image processing code
- Testing with different images to figure out what the main factors were in preventing measurement of non-spherical particulates
- Main problem factors:
  - Threshold process (divides the image into foreground and background)
  - Watershed process (finds the boundaries between particulates)

Approach (cont.)

- Thresholding
  - The main problem is that some images have large variance in the foreground and background pixel values and overlap between the two
  - Attempts to enable efficient thresholding of images with wide range of pixel values:
    - Development of code that allows the user to threshold by clicking background pixels in the image
    - Addition of ability to look at the threshold image before moving on, at which point the user can choose to redo the threshold as many times as desired

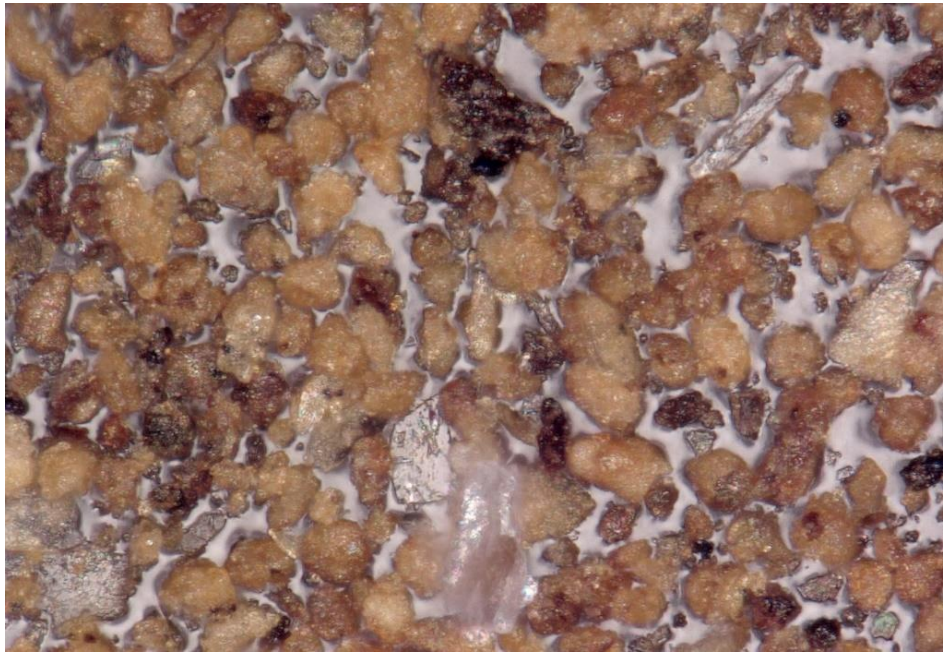
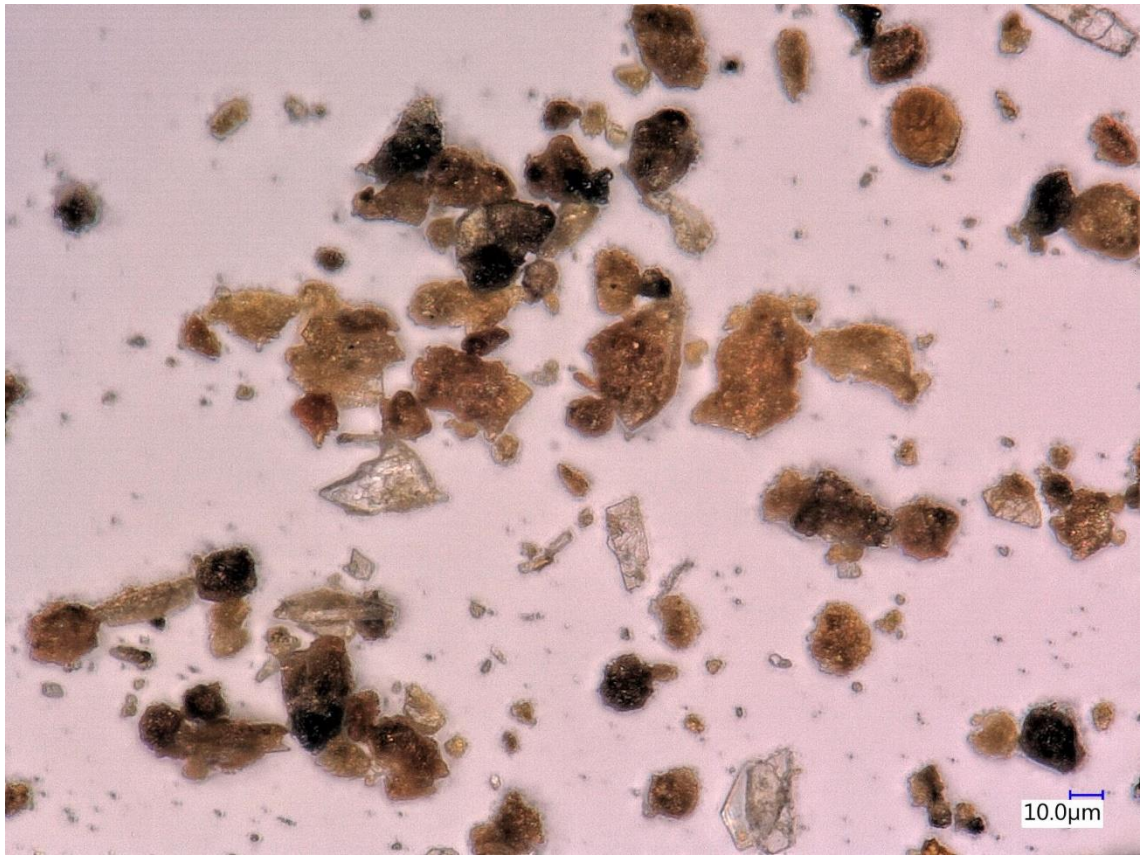
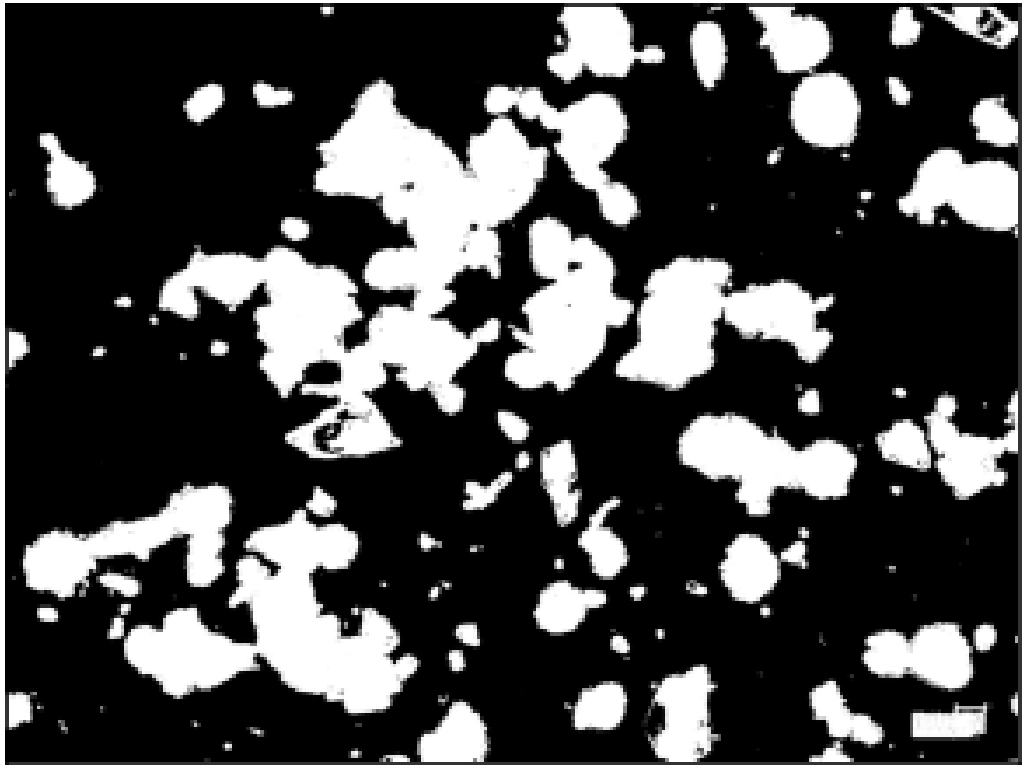


Image with problematic pixel values
Image Credit: NASA



Original image

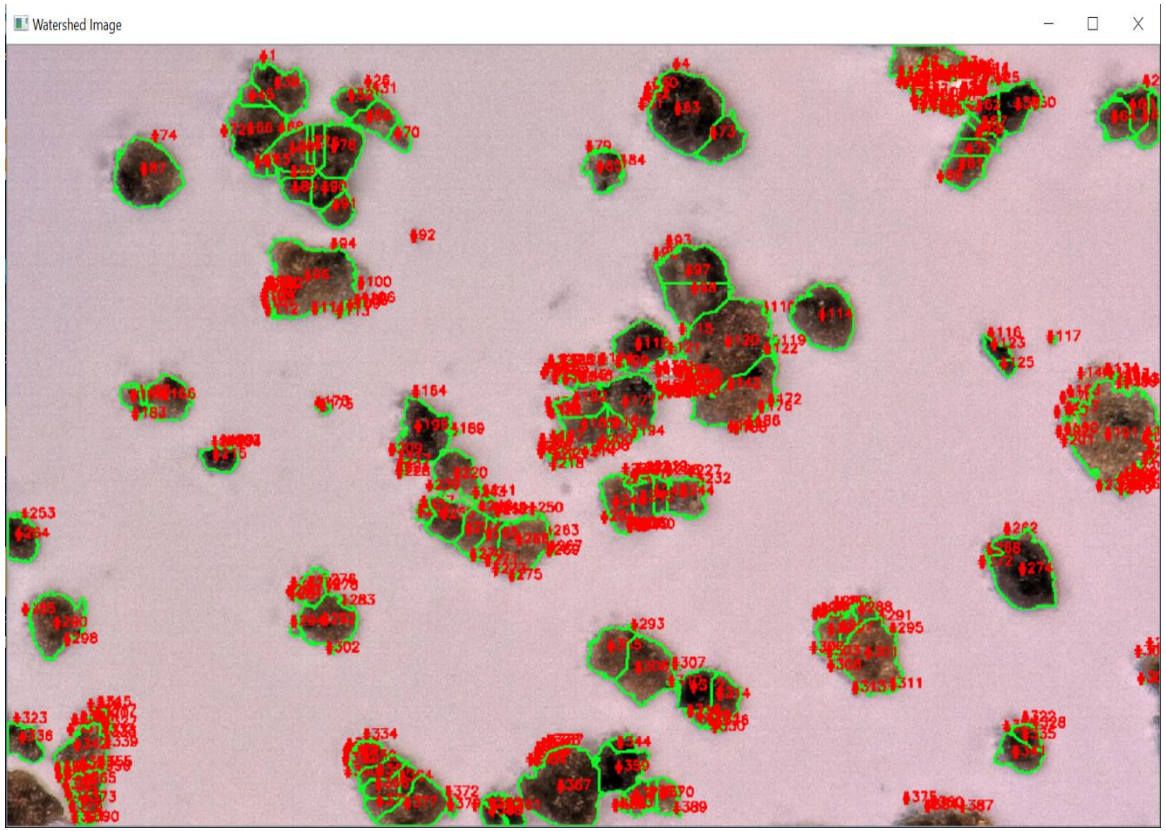


Threshold image

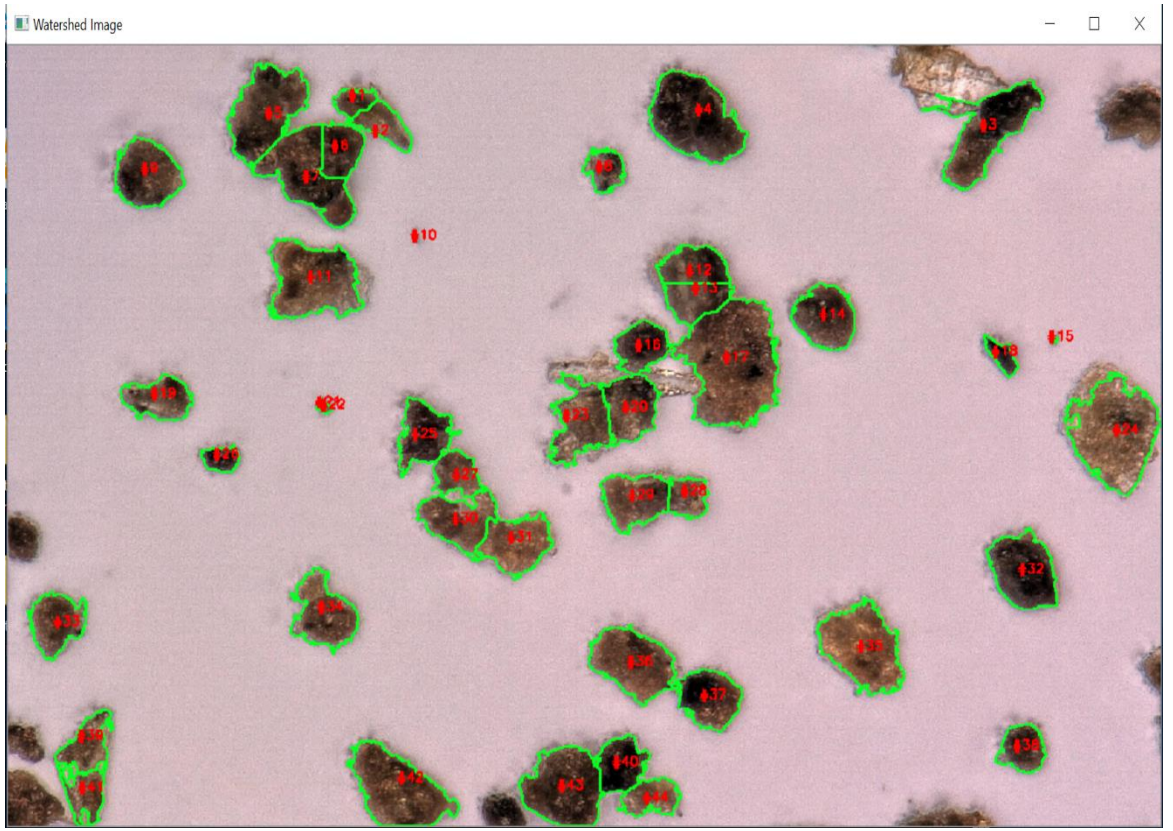
Image Credit: NASA

Approach (cont.)

- Watershed
  - Smaller minimum distances in the peak_local_max function from scikit-image yield more segmentation, and large minimum distances can exclude particulates and under-segment
  - Attempts to prevent missing particulates and over-segmentation:
    - Development of code that allows the user to draw particulate regions and background regions manually, as well as click to connect and remove regions
    - Allow user to manually set minimum distance for peak_local_max function



Over-segmentation due to small peak_local_max minimum distance



Exclusion and under-segmentation of some particulates due to large peak_local_max minimum distance

Image Credit: NASA

Results

- Optimal threshold through clicking background
- Optimal segmentation through setting minimum distance for peak_local_max function
- Manually correct for under-segmentation and over-segmentation

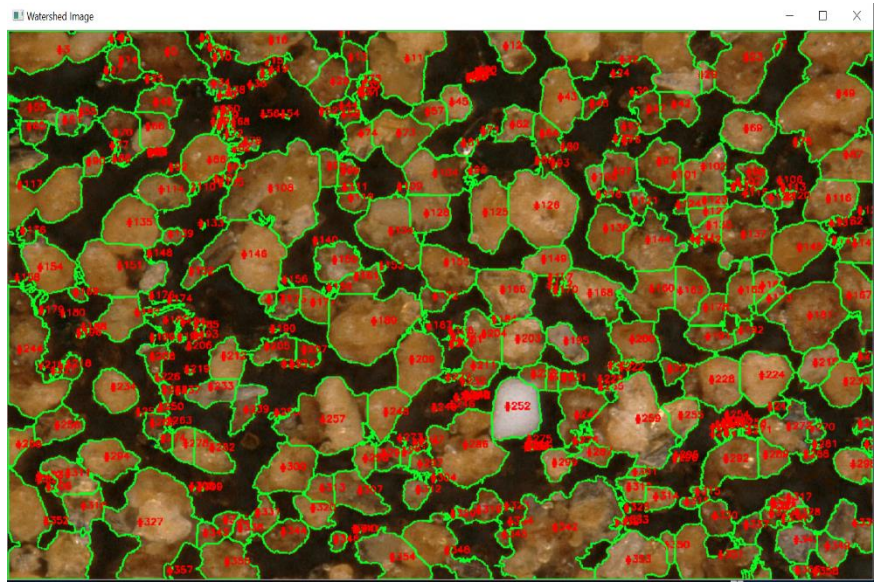
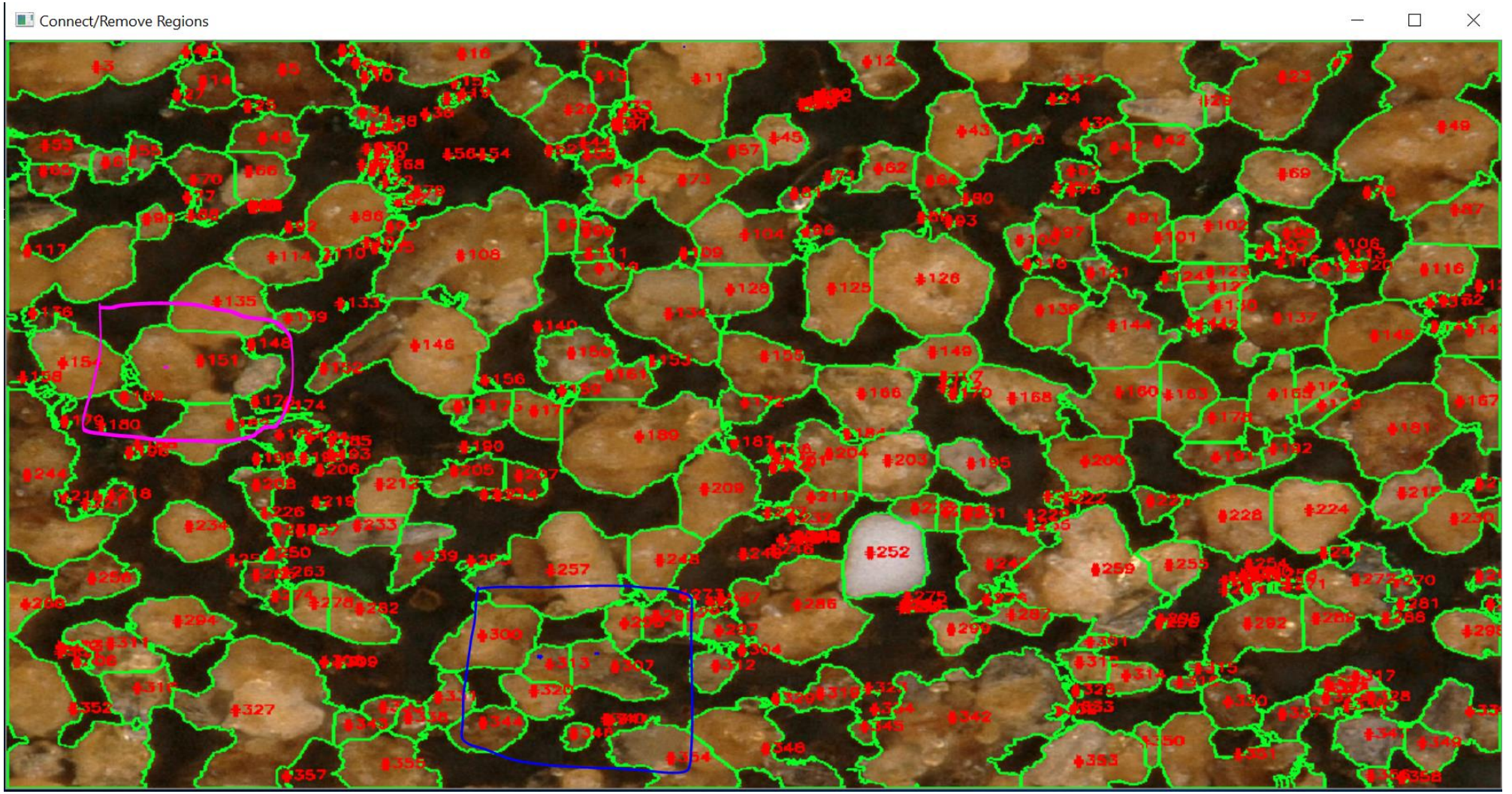
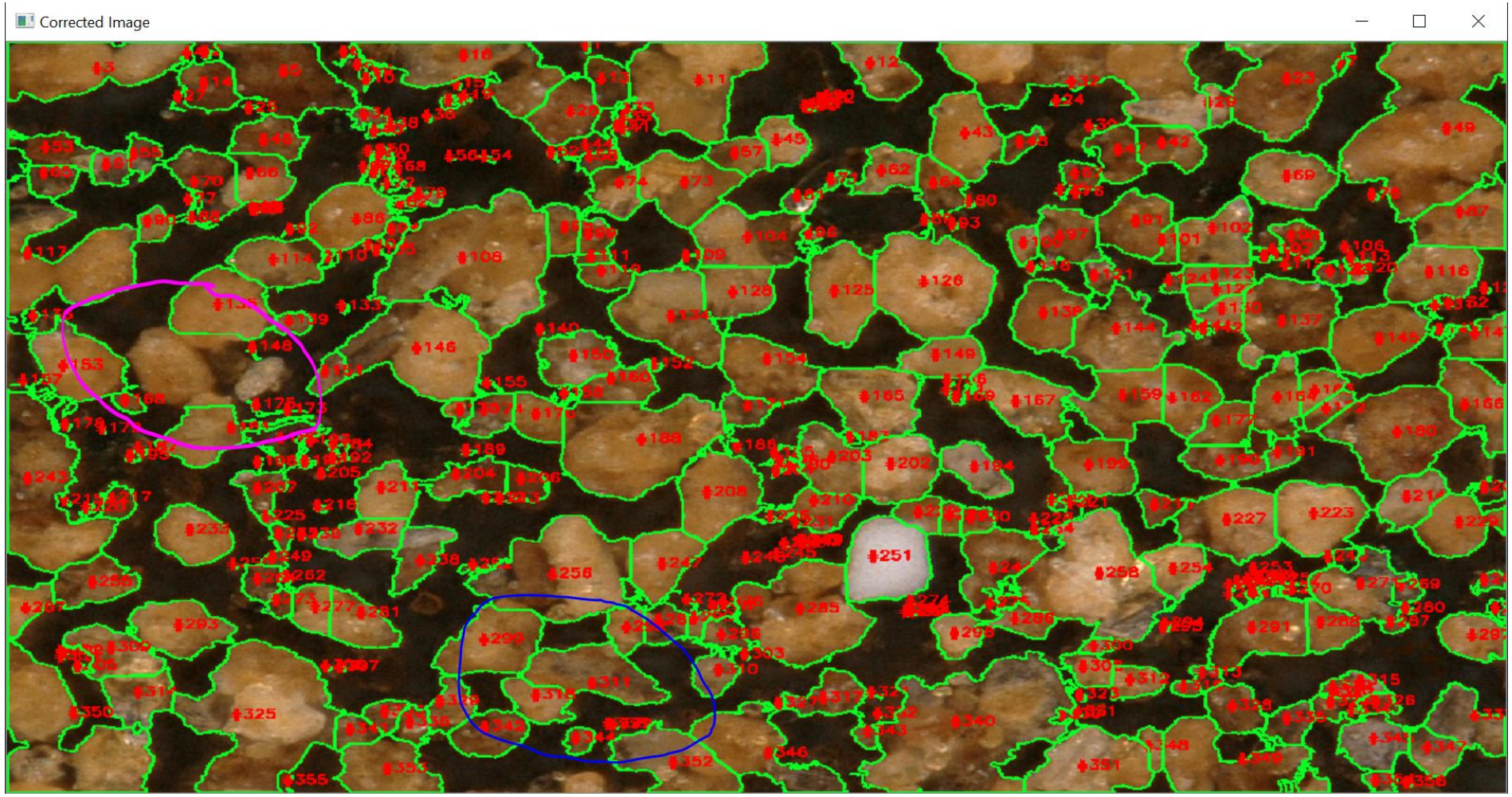


Image after watershed
Image Credit: NASA

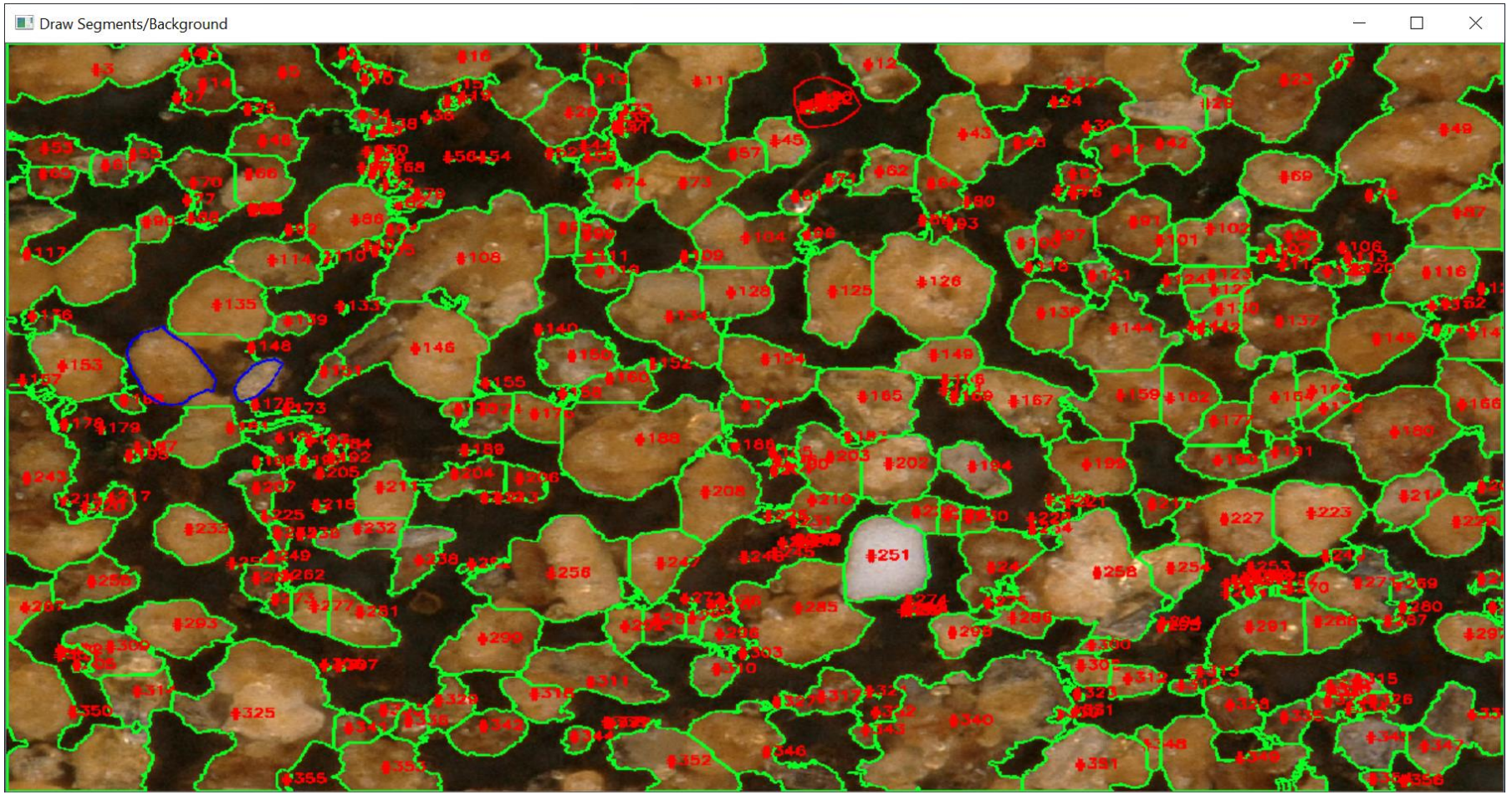


Clicking to connect(blue) and remove(pink) regions
Image Credit: NASA

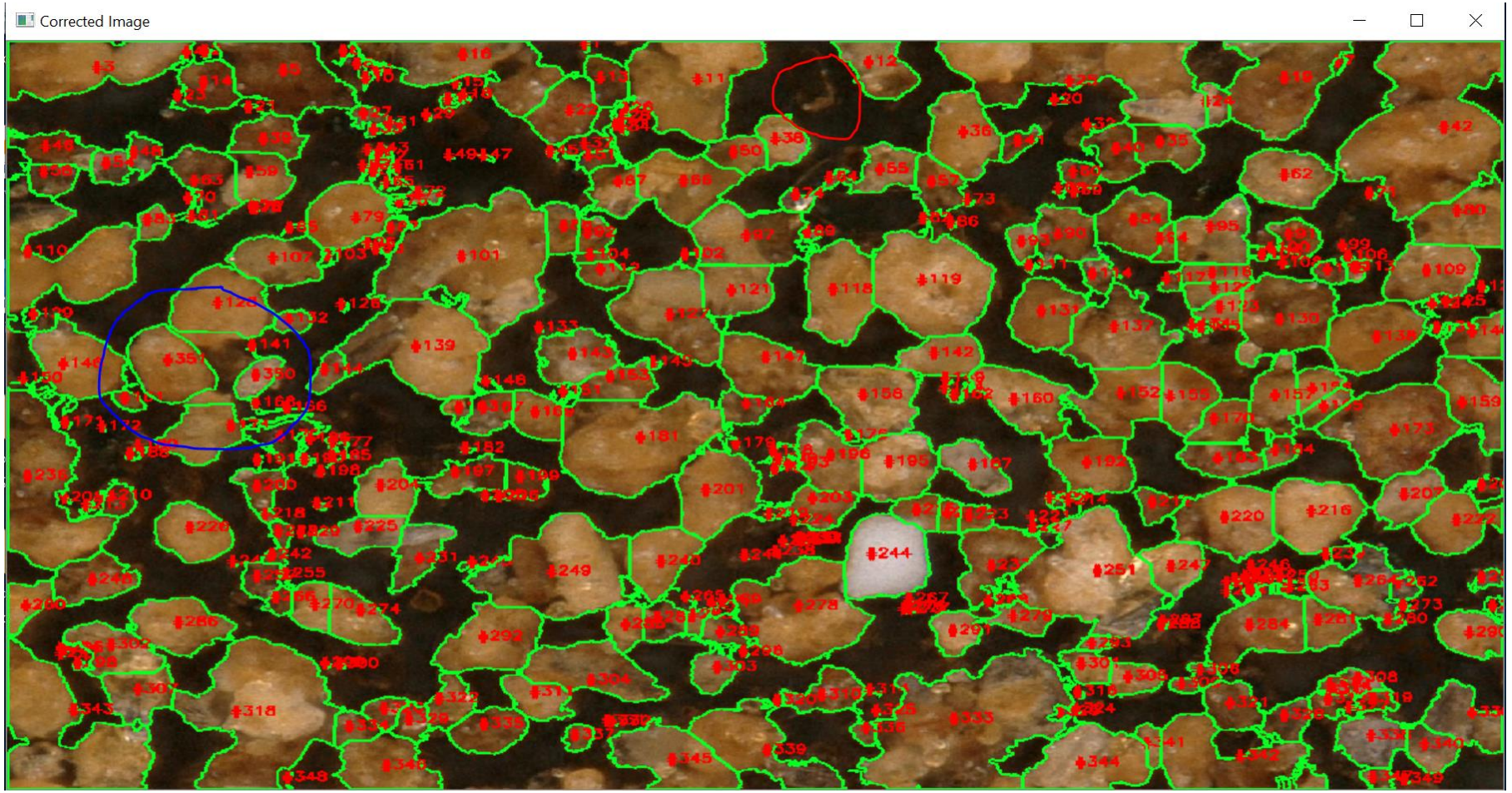


Results of clicking

Image Credit: NASA



Drawing to add (blue) and remove (red) regions
Image Credit: NASA



Results of drawing

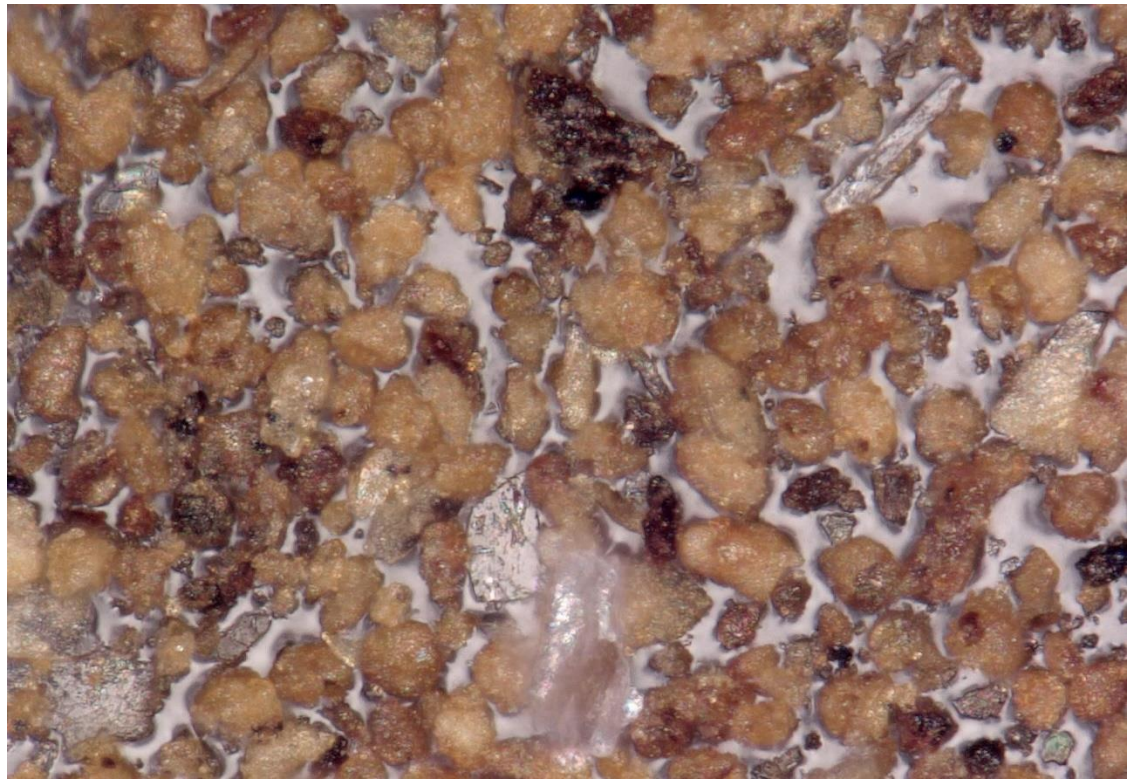
Image Credit: NASA

Analysis/Summary

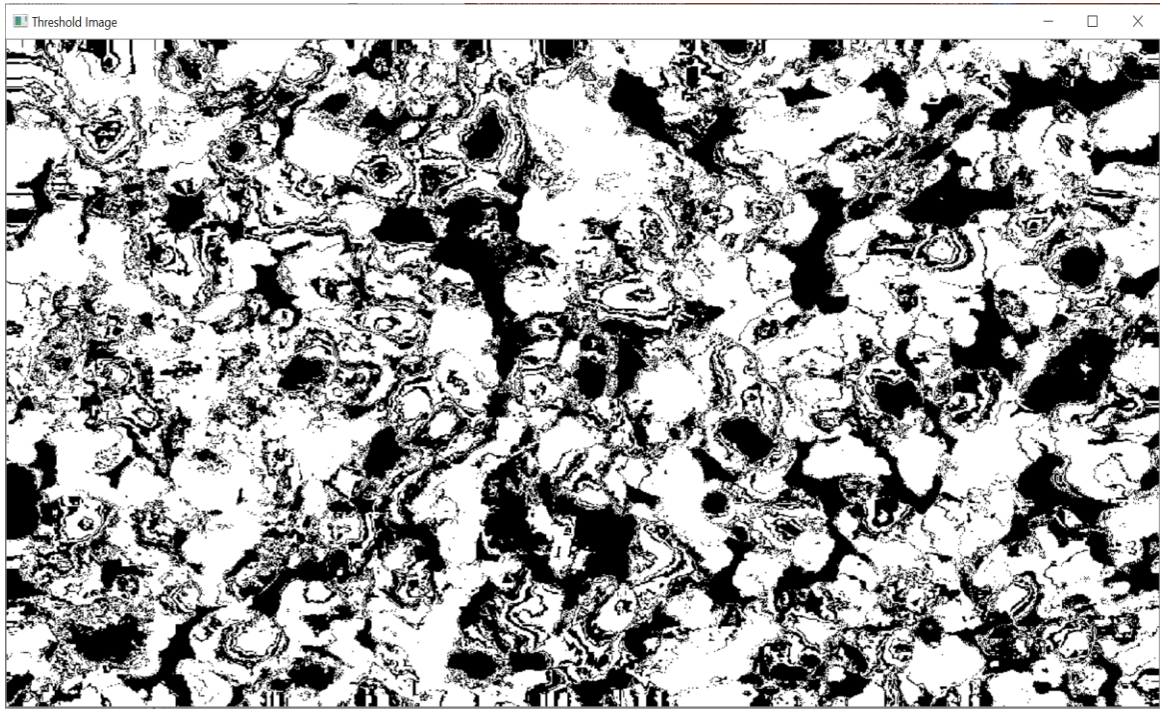
- The optimized image processing program has the capabilities of the previous version
- However, the new version allows for more flexibility in processing images with varying colors, particulate sizes, and particulate shapes
- The new manual thresholding and segmentation features require moderate user-input, but the process is still faster than manually drawing with ImageJ

Next Steps/Outlook/Future Work

- Machine learning or looping through multiple peak_local_max minimum distances promising for increased automation
- Could be useful to attempt other segmentation methods besides watershed
- Different thresholding methods could be attempted to deal with pixel value overlap between background and particulates



Pixel value overlap



Bad click thresholding

Image Credit: NASA

Acknowledgements

- **Langley Lunar Dust Brigade**
  - Christopher Wohl
  - Lopa Das
  - Glen King
  - Valerie Wiesner
  - Keith Gordon
  - Sang Choi
  - Amy Huynh
  - Erica Montbach
- **Langley Internships Team**
  - Patricia Sanchez
  - Jalisa Thomas
  - Jessica Gangitano
  - Valerie Ellis
- **The University of Waterloo**
- **Additional Thanks**
  - Samuel Hocker for assistance with the GUI
  - Beverly W. Kemmerer for providing images of Martian regolith simulant