Software Development Sample

Andrew Kim

University of Pennsylvania

GitHub Repo Publication

As part of the Summer Engineering Apprenticeship Program (SEAP), a competitive, paid, and full-time research position awarded by the Department of Defense, I created the software platform CrystalEyes to accelerate the analysis of nano-ice crystals. In the two summers I worked for Dr. Eunkeu Oh at the U.S. Naval Research Laboratory in Washington, D.C., I created a software platform that fully automated the extraction of data from ice crystal images and even decreased processing time by 95%.

My mentor, Dr. Oh, aims to create antifreeze proteins (AFPs) that regulate the size of ice crystals in an attempt at Ice Recrystallization Inhibition. This research has countless applications, from long-term organ and tissue cryopreservation to environmental protections. One of the ways she would access the efficacy of her AFPs was to track changes in the size and shape of microscopic ice crystals. With existing methods, this would have entailed manually labeling and outlining hundreds of ice crystals per image. This was not only time and labor intensive; it also limited tracking changes in shape and size over time, a critical metric.

My task was creating CrystalEyes, a versatile software platform that could automate this process. The final product was able to rapidly analyze nano-ice crystal images to extract key metrics such as change in ice crystal size, shape, frequency, density, and more over time. By the end of my 2 summers working at the NRL, Dr. Oh and our team had analyzed thousands of images and collected gigabytes of ice crystal morphology data, tremendously accelerating the data analysis process. CrystalEyes' contributions were included in one of Dr. Oh's papers as well, published in March of 2025:

Systematic Development of a Plasmon uler Using Dithiolate-Grafted Gold Nanoparticle for High-Throughput Screening of Anti-Icing Activity

To Create CrystalEyes, I used the following frameworks and libraries:

1. **Python**: The core language of the CrystalEyes app, including logic, user interface, and analysis. The app logic included errors that the user would be notified of.

2. **NumPy + Pandas**: Storing, analyzing, and formatting data, recording user app preferences, and formatting output to be exported into the .xlsx format. I stored and analyzed metrics such as size, shape, and frequency changes over time, constituting a very large dataset. I also created a custom binary file parser. The lab machine used by my mentors stored experimental metadata in a proprietary binary file that was unreadable except for a proprietary software. So, I went through the hex values in the proprietary files and created a decoding scheme to extract all images and metadata from the files.

3. **Matplotlib**: Displaying previews of analyzed data and projecting images in the UI.

4. **Tkinter**: User interface and interactive components. Used Ttk for enhanced visuals and complex components, including a debugging console, text inputs, and tabs. I added events to the Matplotlib graphs, enabling the user to manipulate the UI by clicking on desired points on a graph. I also added a video play feature that enabled the user to browse through and play frames and live data.

5. **OpenCV**: Used to store images and project analyzed NumPy contours onto images. Outputs included both raw images extracted from the proprietary file and individual contours projected onto the raw images.

6. **Cellpose**: A Python library meant to extract morphology data from cell cytoplasms. I adopted a pretrained Cellpose machine learning model to identify ice crystal contoursand extrat them as NumPy arrays. These were then analyzed as contours to provide shape, size, and frequency data.

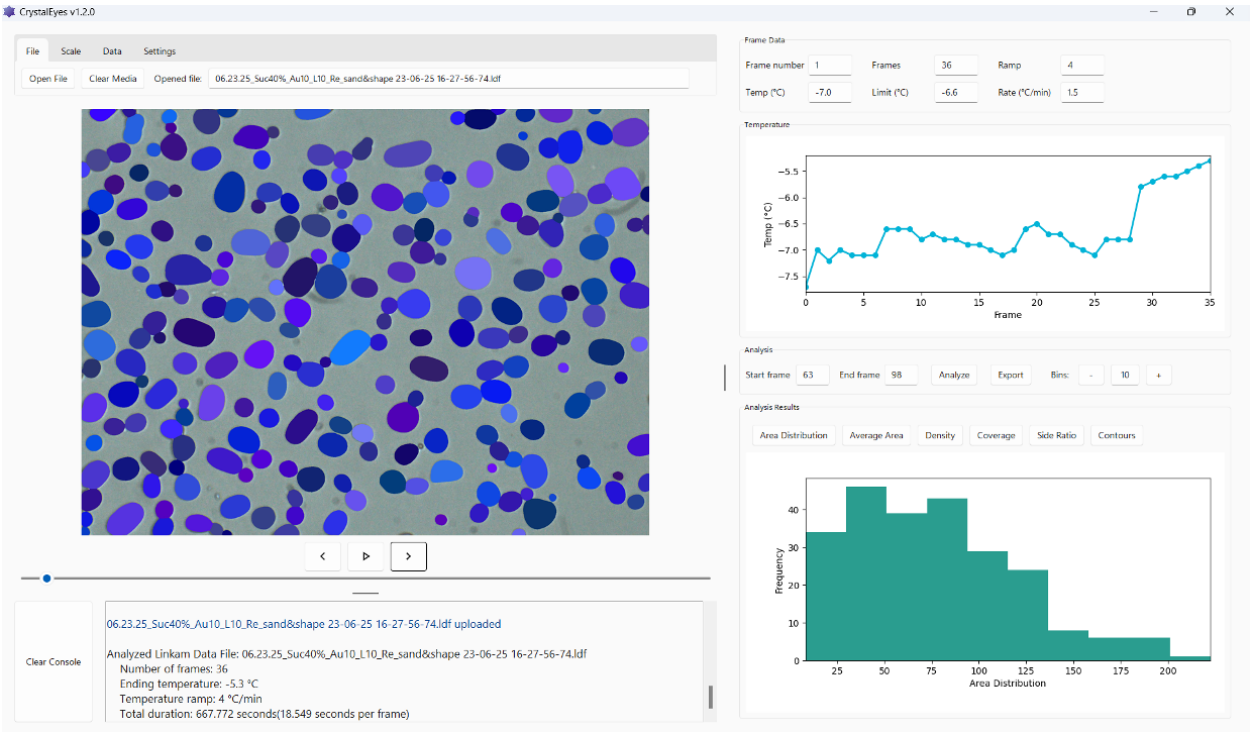

Figure 1: The CrystalEyes user interface. From top left, going counterclockwise: The tab navigation feature, metadata extracted at the current frame, interactive and customizable graphs showing dynamic changes in data, a debugging console, the video player.

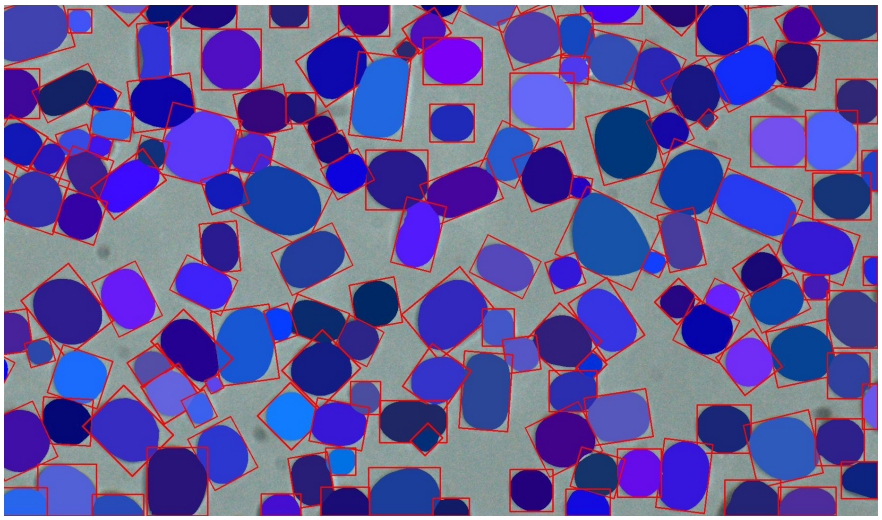

Figure 2: Contours analyzed with CrystalEyes. The contours are shown in different shades of blue for differentiation, while the bounding boxes show the dimensions used to calculate height and width ratios of shapes. Note the high degree of accuracy despite bordering and contained contours.