Automated Graphene Sample Clustering

Neil Ghugare, Jacob Balek, Jiayin Wang, and Dr. ChunNing Lau

The Ohio State University

Abstract

Graphene is a common substance exfoliated on a Silicon/Silicon Dioxide (Si/SiO_2) substrate. On our substrate, we can manually search the surface to find a Graphene sample using an optical microscope and identify the sample's thickness using green values from the sample and a close substrate pixel [Wang et al., 2012]. However, searching the surface is time consuming, and it would be easier if a person could just place a wafer and a system 1) searched the sample and 2) clustered layers of the sample. This project provides a way to do the latter. The results of the program provide a helpful guide to seeing the clusters of the sample, but still require some physical observations to attain conclusive results.

Sample Classification Model

We created a Tensorflow machine learning model that can identify if an image contains a sample or not.

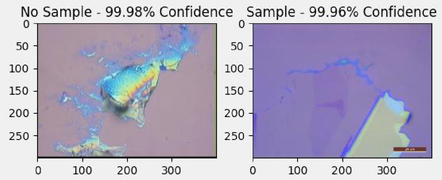

Figure 1. Model Confidence on Sample Classification

Image Pre-Processing

When an image is opened, it is immediately resized from its original size to 75x100 in order to reduce features and to save time iterating through the image's pixels. The image is converted to L*a*b* from RGB color space. The background color is extracted by looking at similar L*a*b* values from the substrate. Then the image is blurred using a median blur algorithm with a kernel size of 3. The median blur algorithm takes a pixel centered at the 3x3 window size and repeats the bordering pixels to create it, if needed. The median of the 9 pixels are calculated and then the central pixel is set to that value.

Example of entire 3x3 window median blurred:

$$\begin{matrix} 1\,9\,3 \\ 4\,5\,0 \\ 1\,7\,2 \end{matrix} \longrightarrow \begin{matrix} 4\,3\,3 \\ 4\,3\,3 \\ 4\,2\,2 \end{matrix}$$

Example of the top left pixel median blurred by making it the central pixel, and repeating the missing gaps in the kernel with the border pixels:

$$\begin{matrix} \mathbf{1}\,9\,3 \\ 4\,5\,0 \\ 1\,7\,2 \end{matrix} \longrightarrow \begin{matrix} 1\,1\,9 \\ 1\,\mathbf{1}\,9 \\ 4\,4\,5 \end{matrix} \longrightarrow 4$$

This example was done with a matrix of shape $(3, 3, 1)$ but our matrix size would be $(75, 100, 3)$. The blurring is more complicated, but the core idea is still the same. After the median blur is completed, the L* values are isolated into a grayscale-esque image. The image the undergoes Canny edge detection, and the contours are located and the sample is isolated from the substrate using a mask. Canny edge detection does the following:

- Smooth the image through Gaussian blurring
- Calculate intensity gradients between the pixels
- Apply suppression to thin out the edges
- End with Hysteresis thresholding (double-thresholding)

DBSCAN and GMM Clustering

The isolated image next undergoes DBSCAN clustering. This unsupervised clustering algorithm finds core samples in high density regions and expands clusters from them. However, clusters classified from DBSCAN are way more than actually present due to samples being separated. Therefore, we employ Gaussian Mixture Modeling, or GMM to reduce the total number of clusters by finding similarities between the clusters. An easier way to think about the GMM is by saying it takes a sample set of size n and estimates and predicts K clusters (user-defined) while taking into account the structure of the data.

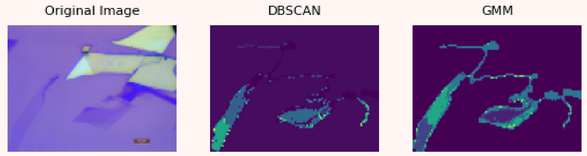

Figure 2. Original Image Undergoing DBSCAN and GMM

Behind the GMM

The math behind the GMM:

Upon initialization, the GMM initializes a random mean (μ), a random covariance (σ), and default mixing coefficients $w = \underbrace{(1/K, 1/K, \ldots, 1/K)}_{K \text{ times}}$ for K clusters we wish to find (predefined using custom thresholding).

For each data point x_i, we calculate the probability it belongs to cluster c.

$$p_{ic} = \frac{w_c N\left(x_i \mid \mu_c, \sigma_c\right)}{\sum_{k=1}^{K} w_k N\left(x_i \mid \mu_k, \sigma_k\right)}$$

where $N\left(x \mid \mu, \sigma\right)$ is the probability density function, or PDF, of a Gaussian Distribution.

Then μ, σ, and w are updated to help fit the data better, and this repeats until the model converges.

Image Cleaning and Deep Cleaning

After the DBSCAN and GMM clustering is completed, we opt to clean up the image in two ways: 1) reclassifying "stray" pixels and 2) reclassifying "far" pixels. These two words in our case have specific definitions.

Stray: Any pixel not labeled as the background label, that is not contained within the original Canny image mask.
Far: Any pixel is "far" if it's normalized RGB value in space is greater than 5% compared to all other pixels of its class.

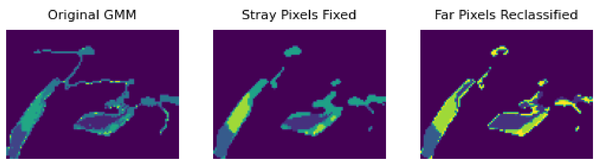

Figure 3. Comparison of the Steps of Image Cleaning

The colors changing are a side effect of simplifying the range of colors from arbitrary numbers to $[0, K]$. We are working on being able to adjust these so that way we get a more gradient-like aspect to the image.

Results

After cleaning the image, we can get an acceptable result for the clustered sample. However, without a concrete key, the student overseeing the photos will still need to do some physical observations of what colors represent thinner and thicker layers. However, one can feel more confident about what layers are what through the clustering instead of having to check each part of a sample with the green color values. We would still need to do more testing on a wider array of photos to see the robustness of the program. However, this may not end up being a problem, due to interesting samples not always being useful. Another note is that since DBSCAN and GMM are unsupervised clustering algorithms, each attempt of clustering on the same image could yield different results (even if minor).

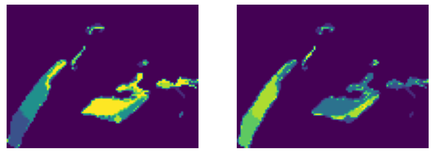

Figure 4. Alternate Results from GMM Clustering

Further Work

There is a lot of further work to be done with this project. We still need to complete the first part of the project, which automates the searching of the sample and takes photos. This clustering algorithm as well can definitely be more fine-tuned in order to become more robust. There are a lot of cases of what a sample could look like on the substrate, which could cause imprecise results with the GMM modeling. Finally, we could create a project that is more general that works with many different substrates and samples that we use.

Acknowledgements

This project acknowledges the help of Jacob Balek, Jiayin Wang, and Dr. ChunNing Lau on this project. We additionally acknowledge Max Sullivan for helping with code testing, and we also acknowledge Xuanzhi Zhang for insight into a similar project he completed for his undergraduate thesis [Zhang, 2023]. We'd also like to acknowledge graduate student Greyson Voigt for allowing us to use his Graphene images for our project. We especially acknowledge Lindsey Thaler, The Ohio State University Department of Physics, and the Bunny C. Clark Student Support Fund for the support and funding of this project through the 2023 OSU Physics Summer Research Program. Finally, we also acknowledge the Python community whose packages we used for this project, including Tensorflow, OpenCV-Python, Scikit-Learn, NumPy, Keras, Matplotlib, and the various provided internal python libraries, with their respective licenses.

References

[Masubuchi et al., 2018] Masubuchi, S., Morimoto, M., Morikawa, S., Onodera, M., Asakawa, Y., Watanabe, K., Taniguchi, T., and Machida, T. (2018).
Autonomous robotic searching and assembly of two-dimensional crystals to build van der waals superlattices.
Nature Communications, 9(1413).

[Sterbentz et al., 2021] Sterbentz, R. M., Haley, K. L., and Island, J. O. (2021).
Universal image segmentation for optical identification of 2d materials.
Scientific Reports, 11(5808).

[Wang et al., 2012] Wang, Y. Y., Gao, R. X., Ni, Z. H., He, H., Guo, S. P., Yang, H. P., Cong, C. X., and Yu, T. (2012).
Thickness identification of two-dimensional materials by optical imaging.
Nanotechnology, 23(49):495713.

[Zhang, 2023] Zhang, X. (2023).
Similar color segmentation using number of cluster algorithm.