A Method for Single-Image Identification of Graphene

Andi Gu
University of California, Berkeley

Abstract
The aim of this paper is to present an algorithm for the automatic labelling of monolayer graphene from a single RGB image taken from a standard microscope. We are thus faced with the challenge of analyzing a noisy low contrast image with significant visual artifacts. The methods described attempt to compensate for these imperfections in the data with minimal ad-hoc information, allowing them to generalize quite naturally to a wide array of applications. Chief among these methods are the image segmentation method, and a novel artifact removal method.

1 Introduction

Since its discovery in 2004, graphene has become a very popular area of research in materials science. As such, graphene is increasingly sought after by research labs across the world – so sought after, in fact, that many labs have collected large stockpiles of graphene samples. However, due to the sheer size of these collections, it is often impractical to build a useful catalogue, resulting in samples that go unused, as well as wasted time spent searching the large collection for graphene samples that have certain properties. The objective of this paper is to develop an algorithm that automatically identifies and labels images of graphene, enabling labs to programatically and quickly extract useful characteristics of a sample. More specifically, this algorithm will be able to draw a tight border around individual pieces of monolayer graphene, enabling queries about properties such as graphene dimension, shape regularity, or area. For labs with unwieldy, large collections of graphene, this algorithm could be run on hundreds of images at once to find samples with certain desired properties, reducing the quantity of wasted samples as well as improving productivity.

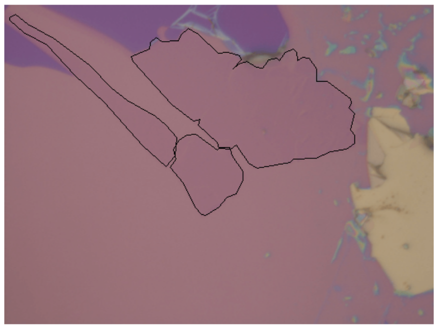


Fig. 1: Graphene on SiO_2 under 50x Magnification[1]
Monolayers highlighted with black border

It is well documented that monolayer graphene on a substrate of SiO_2 can be characterized by its color contrast[2] with the substrate. For thin graphene (less than 10 layers), the green contrast serves as the strongest differentiating factor, with a contrast of approximately -0.06 for monolayer graphene [1] [2], and thus we conduct analysis solely over the green channel of the image.

[1]Original image downloaded from `https://www.acsmaterial.com/media/catalog/product/optimized/9/0/90a0a1c6222895e7e5acbe252cb52f2a/monolayer_graphene_on_300nm_sio2.png` in July 2019.

[2]Contrast of a pixel with value R with respect to a background of value R_0 is defined as $\frac{R-R_0}{R_0}$.

There are two steps to identifying monolayers: first, to identify the substrate and its color, and second to find regions which have a contrast anywhere between -0.07 and -0.04 (depending on the variant of SiO_2 substrate) with respect to the substrate. Although seemingly simple, a number of imperfections in the image data often renders these two steps nontrivial. Most notably, images taken under a microscope exhibit strong vignetting (illumination falloff towards image edges) due to a combination of mechanical and geometric factors. Although it is a common problem in image analysis, the effects of vignetting are much more pronounced for monolayer detection – an illumination decrease of just 5% can result in misidentification of certain regions as graphene. The primary goal of this paper is to discuss an approach to attenuating these data imperfections, and applying this to reliably identify monolayers with just a single image.

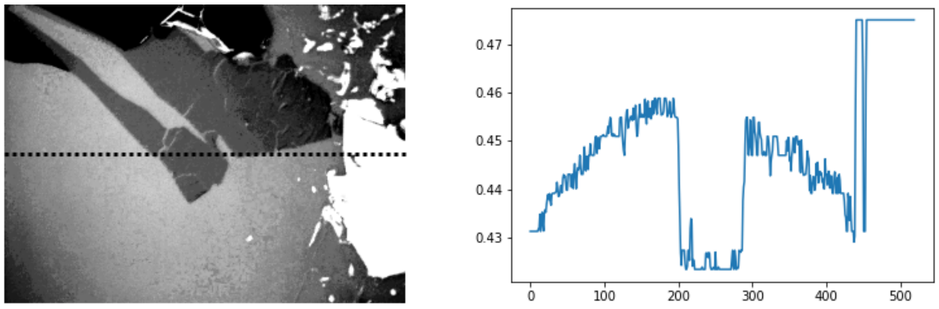


Fig. 2: Vignetting across the horizontal medial axis.
Image clipped to range [0.4, 0.5] to emphasize vignetting

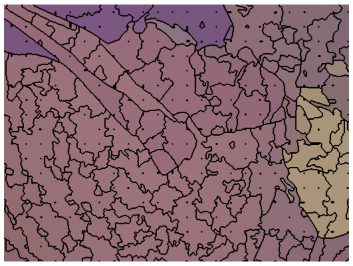
2 Segmentation

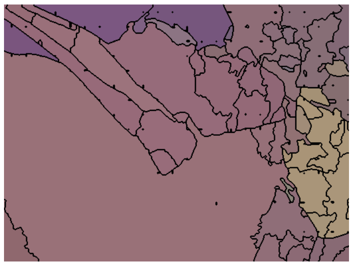
The first step in identifying monolayer graphene is to group the image into similar regions. Perhaps the most straightforward way to do this is to find the perimeter (i.e. edges) of each region and group together connected non-edge pixels. With a perfect image, thresholding the gradient (as found by the Sobel operator) around a constant value would suffice as an edge detection algorithm. However, the presence of noise and vignetting necessitates a more dynamic approach.

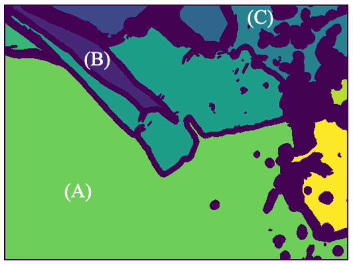
Given the robustness of the Canny edge detector [3], we find it useful to apply it to this case. We are thus left with the problem of finding the appropriate low and high hysteresis thresholds for the Canny detector. As there is a wide range of possible values that characterize an edge gradient and a comparatively small range of gradient values for non-edges, we define the hysteresis thresholds in relation to *non*-edges. To identify non-edges, we begin by finding an oversegmentation of the image by running a Priority-Flood watershed [4] on the gradient of the image. A boundary region adjacency graph (RAG) [5] can then be constructed from this segmentation, such that the weight between two nodes is the average value of the gradient on the boundary between their corresponding regions on the image. The oversegmentation is then merged via a greedy process by combining regions with edge weight below a small threshold. However, due to vignetting and various other imperfections, this result is inevitably still oversegmented – albeit slightly less so than before. Given this oversegmentation, we observe the gradient values for the interior of each segment, taking the top 70th and 90th percentile of interior gradient values as the low and high hysteresis thresholds, respectively.

After finding edges via the Canny detector, regions were labelled by 4-connectivity to split the image into areas of roughly similar color. Pixels within the same group can be considered the same class (e.g. substrate, monolayer graphene, bilayer graphene, etc.) – but it is not true that all the pixels in a class belong to the same group. For example, the groups labelled (A), (B), and (C) all belong to the substrate class, yet they are separate groups.

[2]The interior of a segment is defined as the subset of pixels in the segment that are at least a distance ϵ away from the rest of the image (typically with $1 \leq \epsilon \leq 10$)


(a) Watershed oversegmentation


(b) Result of greedy merging



(c) Labelled and pruned[3] groups

Fig. 3: Process for image segmentation

3 Removing Vignette Effect

Having split the image into various regions of approximate similarity, it is possible to estimate the effects of vignetting and remove them to reveal the ground truth color of a given region. A well-established model for illuminance falloff is the $\cos^4$ law [6], however, in practice, lens aberrations can cause up to a 31% variation in brightness over an image [7]. This certainly represents a nontrivial fraction of the vignetting, and thus our vignetting model must incorporate a variety of factors that include tilt effects in the scene: we develop a model that extends the Kang-Weiss model [8] to estimate the vignetting falloff over an image as a function of a pixel's position. We can assume the graphene lies on a two dimensional plane in a coordinate system we label $\mathcal{G}$. However, it is possible that the plane does not lie perfectly normal to the axis of the camera, and thus we allow for it to be tilted with respect to the plane normal to the camera's axis (termed the 'normal plane'). This tilt is characterized by the parameters θ and φ – more specifically, we assume the object plane is rotated by an angle φ about the unit vector at an angle θ (with respect to the x-axis) in the normal plane.

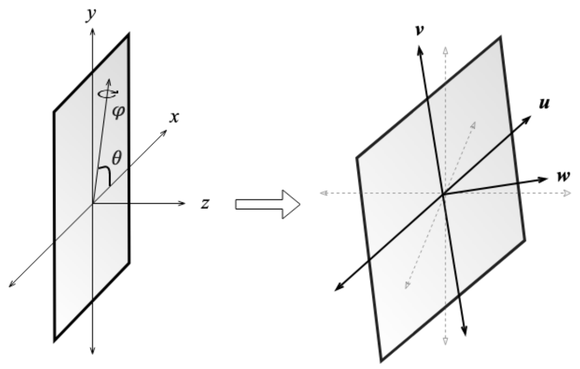


Fig. 4: Rotation about $\hat{R}_\theta$ by φ

This allows us to create a mapping from image coordinates[4] in the basis $\mathcal{G}$ to three-dimensional space coordinates in the camera's perspective, which we call $\mathcal{C}$. To do so, we use the Rodrigues rotation formula:

$$\underset{\mathcal{G}\to\mathcal{C}}{T} = I + W\sin\varphi + W^2(1-\cos\varphi) = \begin{bmatrix}\vec{v}_1 & \vec{v}_2 & \vec{v}_3\end{bmatrix} \tag{1}$$

$$\text{with } W = \begin{bmatrix} 0 & 0 & \sin\theta \\ 0 & 0 & -\cos\theta \\ -\sin\theta & \cos\theta & 0 \end{bmatrix}$$

[3]Groups below a certain area threshold are discarded and ignored in future analysis.

[4]Although any point on the image can be uniquely identified with two coordinates u and v, we introduce a third coordinate w which is set to a constant value of zero for convenience (keeping the transformation $\underset{\mathcal{G}\to\mathcal{C}}{T} : \mathbb{R}^3 \to \mathbb{R}^3$).

Now, instead of separating vignetting effects into the illumination factor A and tilt factor T as in the Kang-Weiss model, we combine the two into a single 'off-axis' term. To begin our analysis, we define a few important terms.[5]

- The principal point $[\vec{x}]_{\mathcal{G}} = \begin{bmatrix} u_0 & v_0 & 0 \end{bmatrix}^T$, the point on the image to which the camera is focused
- $\vec{n} = \vec{v}_3$, the unit normal to the object plane
- The focus f, the distance from the camera to the principal point
- $\vec{C}_0 = \begin{bmatrix} 0 & 0 & f \end{bmatrix}^T$, the vector to the camera from the principal point
- $[\vec{r}]_{\mathcal{G}}$, the vector from the principal point to any point on the image in $\mathcal{G}$ coordinates (consequently $\vec{r} = [\vec{r}]_{\mathcal{C}} = \underset{\mathcal{G}\to\mathcal{C}}{T}[\vec{r}]_{\mathcal{G}}$)
- $\vec{C} = \vec{C}_0 - \vec{r}$, the vector from any point in the image to the camera
- $\bar{I}$, the illuminance at the principal point

There are two factors included in illuminance falloff: distance and projected areas. First, illuminance falls off with $1/D^2$, where D is the distance from the source to the receiver. Thus, the distance term is $\propto \|\vec{C}\|^{-2}$. Second, assuming the surface of the graphene acts as a diffuse Lambertian reflector, the luminous intensity emitted is proportional to the projected area of the graphene as seen from the lens, so the luminous intensity emitted is proportional to $\vec{n} \cdot \widehat{C}$. Finally, the projected area of the lens as seen from the object is proportional to $\widehat{C} \cdot \widehat{C}_0$. Thus the off-axis term, η, (with tilt effects included) is:

$$\begin{aligned}\eta &= \frac{f^2}{\left\|\vec{C}\right\|^2} \frac{\vec{n} \cdot \vec{C}}{\left\|\vec{C}\right\|} \frac{\vec{C} \cdot \vec{C}_0}{\left\|\vec{C}\right\|\left\|\vec{C}_0\right\|} \\ &= \frac{f\left(\vec{n} \cdot \vec{C}\right)\left(\vec{C} \cdot \vec{C}_0\right)}{\left\|\vec{C}\right\|^4} \qquad (2)\end{aligned}$$

Of course, this is the most general case. When $\varphi = 0$, η is equal to $\frac{f^4}{R^4}$, that is, it reduces to the the $\cos^4$ law for falloff of illumination. We also introduce a linear geometric term $G = 1 - \sum_{i=1}^{p} \alpha_i r^i$ used in the Kang-Weiss model, with $p = 2$. In summary, the vignetting function is:

$$I = \eta G \bar{I} = \vartheta \bar{I} \qquad (3)$$

3.1 Identifying Parameters

There are eight parameters to be found for each group: u_0, v_0, f, θ, φ, α_1, α_2, and $\bar{I}$. Although it is possible to fit eight new parameters for each group, the physical nature of this model allows us to impose more rigorous restrictions – by definition, it is clear that the first seven parameters should be the same for the entire image. Only the last parameter, $\bar{I}$, should vary across different groups – however, since it is a linear term, it can easily be fit for each individual group once the global vignetting function ϑ is determined. In effect, our model has seven global parameters and n 'secondary' hidden parameters (where n is the number of groups), namely $\left\{\bar{I}_1, \ldots, \bar{I}_n\right\}$. We can declare a function $\mathcal{I}(f, \theta, \varphi, \alpha_1, \alpha_2)$ which approximates the image's vignetting function. The goal, then, is to minimize the error between $\mathcal{I}$ and the image to fit, M.

$$u_0, v_0, f, \theta, \varphi, \alpha_1, \alpha_2 = \operatorname{argmin} \|M - \mathcal{I}(u_0, v_0, f, \theta, \varphi, \alpha_1, \alpha_2)\| \qquad (4)$$

The function $\mathcal{I}$ calculates the global vignetting function ϑ, and multiplies the appropriate constant factor $\bar{I}_i$ to each group. More formally, if the image we are trying to fit the vignetting function to is M, we let M_i and ϑ_i denote the pixel values of the image and global vignetting function for the ith group, respectively. Then:

$$\bar{I}_i = \underset{C}{\operatorname{argmin}} \|M_i - C\vartheta_i\| = \frac{\vartheta_i \cdot M_i}{\|\vartheta_i\|^2} \qquad (5)$$

[5]We use $\hat{v}$ to denote the normalized form of $\vec{v}$.

```
**Data:** Global parameters $u_0$, $v_0$, $f$, $\theta$, $\varphi$, $\alpha_1$, $\alpha_2$, and the image to fit $M$
Center image coordinates in $\mathcal{G}$ around the principal point $[\vec{x}]_{\mathcal{G}}$;
$\vartheta \leftarrow \eta(f, \theta, \varphi) * G(\alpha_1, \alpha_2)$;
**for** $i \leftarrow 1$ **to** $n$ **do**
    $\bar{I}_i \leftarrow \frac{\vartheta_i \cdot M_i}{\|\vartheta_i\|^2}$;
    **for** $[\vec{r}]_{\mathcal{G}}$ *in ith group* **do**
        $\mathcal{I}$ at $[\vec{r}]_{\mathcal{G}} \leftarrow \bar{I}_i * (\vartheta$ at $[\vec{r}]_{\mathcal{G}})$
    **end**
**end**
```

Algorithm 1: Evaluating the vignetting function $\mathcal{I}$

Using the Levenberg–Marquardt algorithm [9] for nonlinear least squares, it is possible to solve equation (4), finding the optimal parameters. Once the model parameters are identified, it is easy to 'flatten' the image by subtracting out $\mathcal{I}_{best}$.

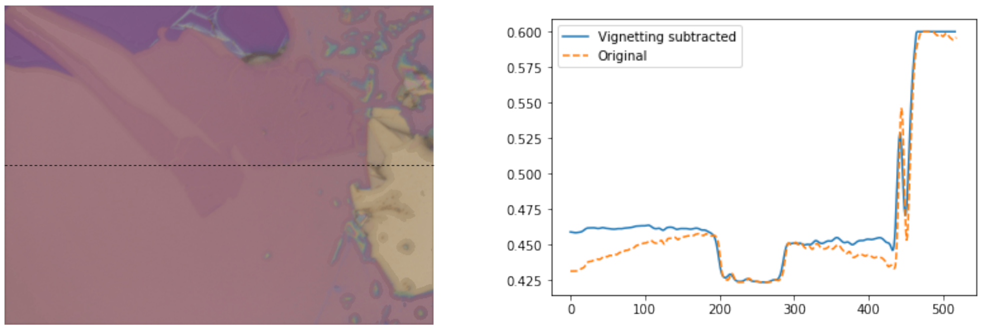
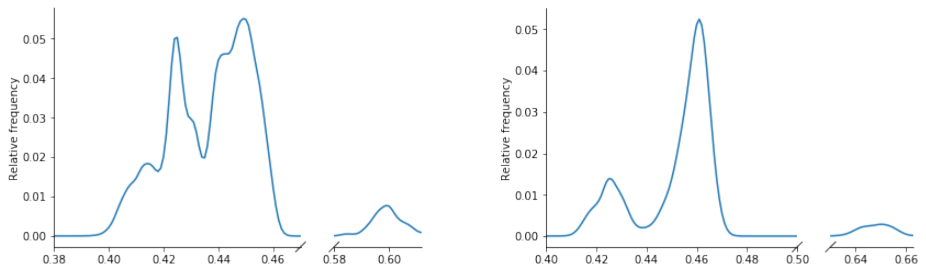


Fig. 5: Subtracting vignetting function

4 Identifying the Substrate and Monolayers

Although we have a 'flattened' version of the image with vignetting removed, due to noise, there may be slight variation in the distribution of pixel values for each class. Fortunately, the noise follows a roughly normal distribution centered around each classes' ground truth color. As such, the distribution of pixel values should be a collection of superimposed normal distributions, centered around unique peaks for each 'class' (one peak for monolayers, one peak for substrate, and so forth). With a simple peak-finding algorithm, we treat the highest peak as the substrate color and the peak with contrast ≈ -0.06 as the monolayer. This reliably identifies all areas that might be considered monolayer graphene, allowing us to create a boolean mask that marks all pixels belonging to the monolayer class.



(a) Histogram with vignetting **(b)** Ground truth histogram

Fig. 6: De-vignetting acting to separate peaks corresponding to each class

4.1 Separating Overlapping Monolayers

A final problem is that separate pieces of graphene can tend to be narrowly connected, making it impossible to identify the dimensions of each individual piece. As with most segmentation algorithms for overlapping objects, the watershed algorithm was used. Applying a Frangi vesselness filter to the distance transform of the monolayer boolean mask allows us to find an approximate skeleton of each individual piece of graphene by identifying ridges or the natural 'axes' of the graphene. These skeletons are taken to be source markers for each individual piece and serve as the basins drain for the Priority-Flood watershed algorithm [4]. Finally, running the watershed algorithm segments overlapping monolayers unique basins (i.e. individual pieces of graphene).

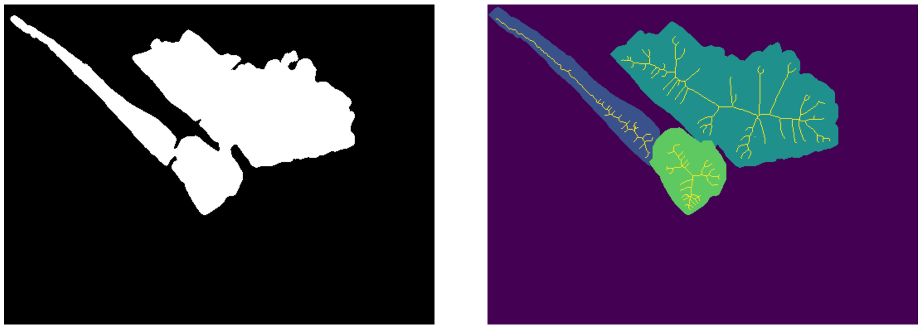

(a) Monolayer graphene boolean mask **(b)** Separated graphene with basin drains, 'skeletons', highlighted

Fig. 7: Separating connected monolayers

4.2 Extracting Graphene Characteristics

With a tight bound drawn around each individual piece of graphene, it is possible to extract a wide variety of physical characteristics. One of the most useful features to extract might be the physical dimensions of each individual piece of graphene – however, without auxiliary information about the scale of the image, this would be impossible to quantify. However, it is certainly possible to find 'thin-ness' of a piece of graphene, quantified by its axial ratio. After the graphene has been separated into individual pieces, we have all the necessary information to calculate this. Assuming a roughly convex shape, simply running principal component analysis and identifying the spread in the direction of both principal components suffices to find the axes of each graphene. Yet another useful characteristic might be the shape regularity, or compactness of each graphene, as quantified by the isoperimetric ratio (the ratio between the squared perimeter and area of a shape).

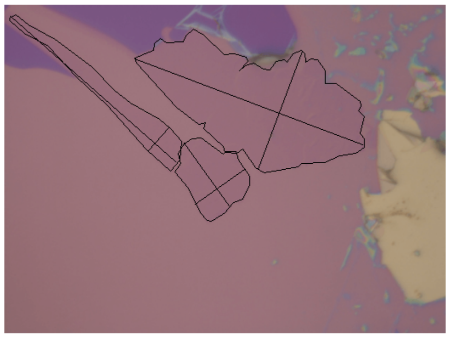


Fig. 8: Perimeter and principal axes of each piece of graphene

5 Discussion

To review, the algorithm may be broken down into six steps.

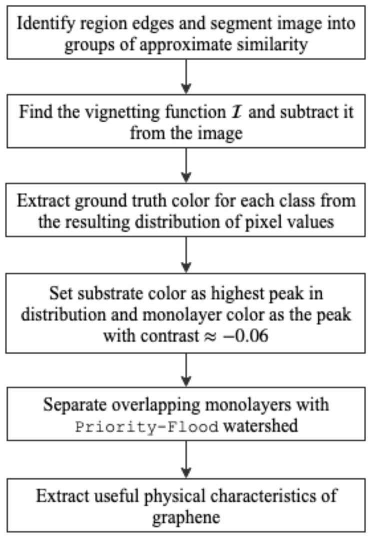

Identify region edges and segment image into groups of approximate similarity
Find the vignetting function $\mathcal{I}$ and subtract it from the image
Extract ground truth color for each class from the resulting distribution of pixel values
Set substrate color as highest peak in distribution and monolayer color as the peak with contrast ≈ −0.06
Separate overlapping monolayers with Priority-Flood watershed
Extract useful physical characteristics of graphene

Fig. 9: An overview of the monolayer graphene identification algorithm

This process was designed with generality in mind, requiring only a bare minimum input to receive useful results. As such, there are a number of opportunities to inject more setup-specific information to improve the accuracy of results. For example, one of the most obvious openings for auxiliary information is the focal length parameter f in the vignetting model, which can be manually entered for any given camera setup. However, it is the full generality of the model that allows it to be extended to a wide array of cases. Perhaps the most natural extension would be to find n-layer graphene, given the contrast in the green channel varies linearly [2] with the number of layers of graphene. This could be done by finding the contrast of the labelled monolayer, C, and searching for peaks with contrast that are positive integer multiples of C. This process might also be extended beyond graphene by adjusting the target contrast. For example, searching for thin hexgonal boron nitride (hBN) may be much easier than searching for graphene, as the target contrast is a much larger range than that of graphene – oftentimes, a contrast that is relatively low (i.e. $-0.1 \leq C \leq -0.03$) will suffice.

However, there are a number of areas in which this process may be improved. For one, the use of a physical model is possibly unnecessary and imposes needless constraints on the shape of the vignetting function. In practice, it is possible for the camera to have non-ideal lenses or an imperfect light source, in which case these effects will most likely outweigh the natural vignetting expressed in the model. As such, perhaps a more robust way of attenuating vignetting may be to fit a polynomial of the form $\bar{I}\big(1 - \sum_{i=1}^{p} \alpha_i u^i + \beta_i v^i\big)$, with u and v being pixel coordinates in $\mathcal{G}$. Although this comes at the cost of the benefits associated with a physical model, it allows for a much more robust expression of vignetting. Yet another area of improvement might be in the method for the identification of the monolayer color. For extremely noisy images, the deviation in substrate and monolayer colors may be large enough to cause their normal distributions to overlap, potentially masking the peak of the monolayer distribution. Although applying a Gaussian before histogram analysis may attenuate noise, this does not preclude the possibility of an underlit image in which the color of the monolayer graphene is quite similar to that of the substrate (i.e. an absolute difference in color of less than 0.02) – in this case, the two normal distributions would inevitably overlap. Given this, it is quite possible that the algorithm yield inaccurate results on noisy, underlit images; to remedy this, we propose running further preprocessing on the image to normalize the illumination (thereby artificially increasing the ISO, potentially amplifying noise), and applying Wang and Fan's self-adaptive filter [10] to then attenuate noise at no cost to edge sharpness.

5.1 Validation

Although there exists no data against which this methodology may be validated against, it was run on a number of examples with widely varying characteristics to test its accuracy and robustness. Shown on the following page are just three cases that a more ad-hoc algorithm might have difficulty labelling. For further validation, an implementation of the algorithm is provided along with additional examples at `www.github.com/andigu/graphene-identification`.

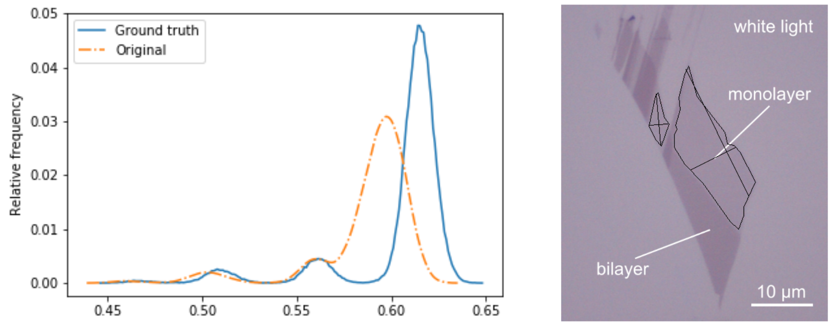


Fig. 10: Graphene on 285 nm SiO_2 with added text and annotation prior to processing[6]

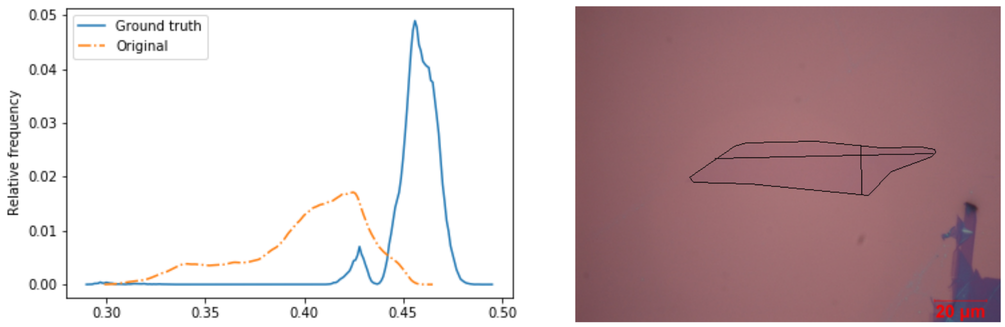


Fig. 11: Graphene on 90 nm SiO_2 with very strong vignetting[7]

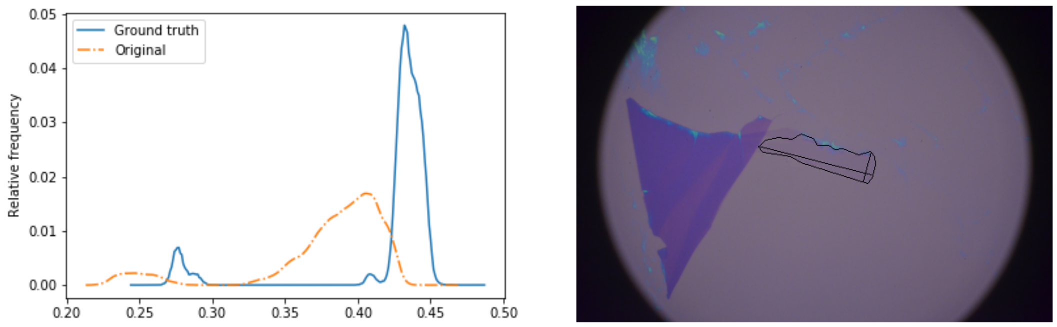


Fig. 12: Small graphene on 285 nm SiO_2 with a mixture of mechanical and natural vignetting

[6]Original image downloaded from `https://ocw.mit.edu/courses/mechanical-engineering/2-674-micro-nano-engineering-laboratory-spring-2016/lecture-notes/MIT2_674S16_Lec10graphene.pdf` in June 2019

[7]Original image downloaded from `www.acsmaterial.com/media/catalog/product/cache/1/image/600x600/85e4522595efc69f496374d01ef2bf13/m/o/monolayer_graphene_on_90nm_sio2.png` in July 2019

References

[1] Xuefeng Wang, Ming Zhao, and David D. Nolte. Optical contrast and clarity of graphene on an arbitrary substrate. *Applied Physics Letters*, 95(8):081102, 2009.

[2] Zhen hua Ni, H. M. Wang, Johnson Kasim, Hongkuan Fan, Tongxi Yu, Yong Hua Wu, Y. P. Feng, and Zhongyao Shen. Graphene thickness determination using reflection and contrast spectroscopy. *Nano letters*, 7 9:2758–63, 2007.

[3] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, Nov 1986.

[4] Richard Barnes, Clarence Lehman, and David Mulla. Priority-flood: An optimal depression-filling and watershed-labeling algorithm for digital elevation models. *CoRR*, abs/1511.04463, 2015.

[5] A. Tremeau and P. Colantoni. Regions adjacency graph applied to color image segmentation. *IEEE Transactions on Image Processing*, 9(4):735–744, April 2000.

[6] Douglas A. Kerr. Derivation of the “cosine fourth” law for falloff of illuminance across a camera image, 2007. [Online; accessed July 11, 2019].

[7] M. Aggarwal, H. Hua, and N. Ahuja. On cosine-fourth and vignetting effects in real lenses. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 1, pages 472–479 vol.1, July 2001.

[8] Sing Bing Kang and Richard Weiss. Can we calibrate a camera using an image of a flat textureless lambertian surface? In *European Conference on Computer Vision*, volume 2, pages 640–653, June 2000.

[9] Jorge J. Moré. The levenberg-marquardt algorithm: Implementation and theory. In G. A. Watson, editor, *Numerical Analysis*, pages 105–116, Berlin, Heidelberg, 1978. Springer Berlin Heidelberg.

[10] B. Wang and S. Fan. An improved canny edge detection algorithm. In *2009 Second International Workshop on Computer Science and Engineering*, volume 1, pages 497–500, Oct 2009.