Analysing Atomic Structure in Nano-scale Imagery

by

David M. Nemirovsky

A thesis submitted in conformity with the requirements
for the degree of M.Sc.
Faculty of Science (Computer Science)
University of Ontario Institute of Technology

Supervisor(s): Dr. Faisal Z. Qureshi and Dr. Isaac Tamblyn

Abstract

Analysing Atomic Structure in Nano-scale Imagery

David M. Nemirovsky

M.Sc.

Faculty of Science

University of Ontario Institute of Technology

2017

This thesis presents methods for both detecting atoms and reconstructing single layer molecular geometry depicted within high resolution nanoscale imagery. Nanoscale imaging is a common method of analysing molecular structures produced through self-assembly. The second derivatives of the nanoscale images are then computed and used to aid in analysis of the depicted molecules. Peaks from the derivative images are used as the locations of the atoms and using those locations we derive the molecular geometry. In order to efficiently determine the performance of our proposed methods, a system to synthesize HRTEM imagery and ground truth data was developed. We demonstrate and compare the effectiveness of our system to other proposed systems using both experimentally produced nanoscale images and the synthetic data that has been produced.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Molecular self-assembly has been a topic of interest for many years due to its ability to produce materials that are lightweight and microscopic but also tensile and otherwise useful. These materials are also difficult and cost ineffective to produce through means other than self-assembly, i.e. lithography [1]. Lattices are of particular importance as some forms of lattices are capable of producing nanoscale circuitry that can be used within computational units. These lattices can decrease the size of transistors within silicon chips thereby enabling the development of faster, smaller and denser processing units. However molecular self-assembly is not perfectly understood [2], often it is unclear whether specific structures or defects within self-assembled molecules will arise and the mechanism driving the defects is often misunderstood. In order to study molecular self-assembly, material scientists often must image the materials via atomic scale imagery, such as Transmission Electron Microscopy (TEM) or Atomic Force Microscopy (AFM), in order to inspect the atomic structure of the resulting material. Inspection of these materials is typically a manual process [3] and is often tedious and time consuming to extract meaningful data from the image. Fig. 1.1 is an example of a manually annotated image for the purposes of highlighting the polygons formed by tris(4-bromophenyl)benzene (TBPB). In this figure someone has manually identified all important vertices within the image and annotated the image with colour coded rings based on the number of vertices within the ring.

As the amount of data increases the time needed and costs of manual inspection may become prohibitive and slow the progress of research. Due to the possibly increasing costs of manual inspection and analysis of these image a method is needed to automatically identify the atoms and important segments within these images.

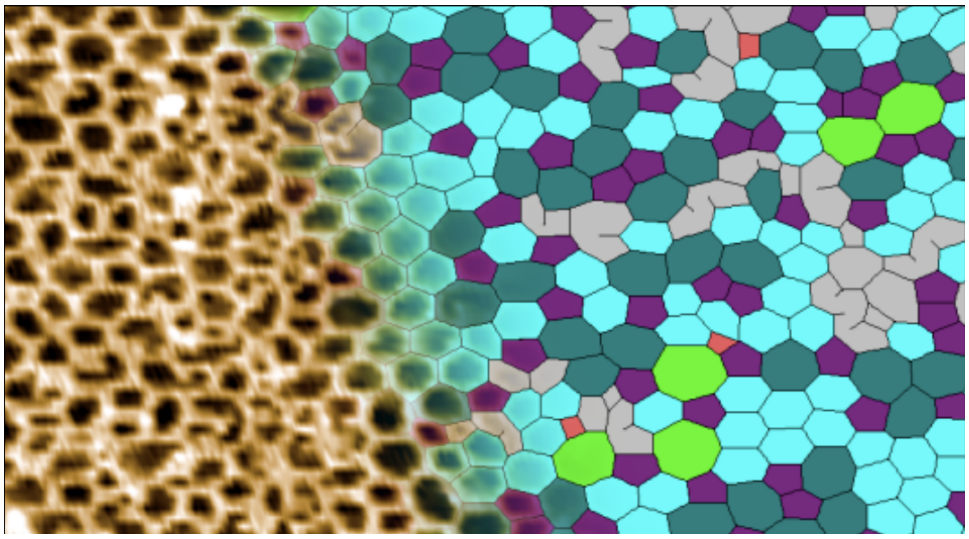


Figure 1.1: Example of a manually annotated image create in order to highlight the rings present within the tris(4-bromophenyl)benzene (TBPB) lattice with each ring being colored according to the number of vertices it has [3].

In this thesis, we have created a method of automatically detecting atoms within atomic scale imagery, produced via AFM and TEM, and extracting the molecular structure for planar lattices. This has been done using image processing techniques and computer vision methods on the atomic scale imagery. The field of computer vision offers many possible image processing techniques that can aid in the analysis of images that possess both an orthogonal view or a perspective view of the data. Since both AFM and TEM imagery are comprised of two dimensional views of the sample surface, it is possible to use image processing techniques in order to extract the necessary data.

However, many challenges exist that make it difficult to not only detect these atoms but also develop efficient testing methods. For example, the high levels of noise present in nanoscale imagery makes it difficult to use simple methods for atomic detection. It is also difficult to find

a large enough number of TEM images to perform machine learning. In this paper we propose a robust and efficient method of detecting atoms within nanoscale imagery produced through Electron Microscopy methods as well as creating a forward model, based on the appearance of atoms, to create images with ground truth data.

1.1 Self-Assembly

Self-assembly refers to "the spontaneous formation of organized structures through a stochastic process that involves pre-existing components,..."[4]. Self-assembly is interesting because the organized structures that it can produce can be used for many purposes. In biological systems, self-assembly drives the creation of several structures including DNA, micelles, ribosomes and even viruses, like the tobacco mosaic virus [5]. Self-assembly within biology can be driven by various forces. These forces include, but are not limited to, concentrations of materials, weak interacting forces, and enzymes. The process of protein folding within cells is also driven by self-assembly. The weak interacting forces between each branch on the protein cause the protein to fold into a functioning unit. The cell structure that combines amino acids to create proteins, the ribosome, also experiences self-assembly during its creation. It has been shown that the ribosome can be created in two portions with each portion created in a solution within separate test tubes. After combining the solution from each test tube, a ribosome will self-assemble from its two parts. The widespread use of self-assembly within organic systems to produce functioning microscopic structures has inspired scientists from many fields — physics, chemistry and biology — to pursue the study of self-assembly.

As described by Pelesko [4], self-assembly requires four enabling features, structured particles, binding forces, an environment, and a driving force. If we use protein folding as our example of self-assembly, the branches of the protein molecule are the structured particles. These branches of the protein when split are amino acids. The order of the amino acids within the protein dictate the shape of the final structure. The weakly interacting forces between the

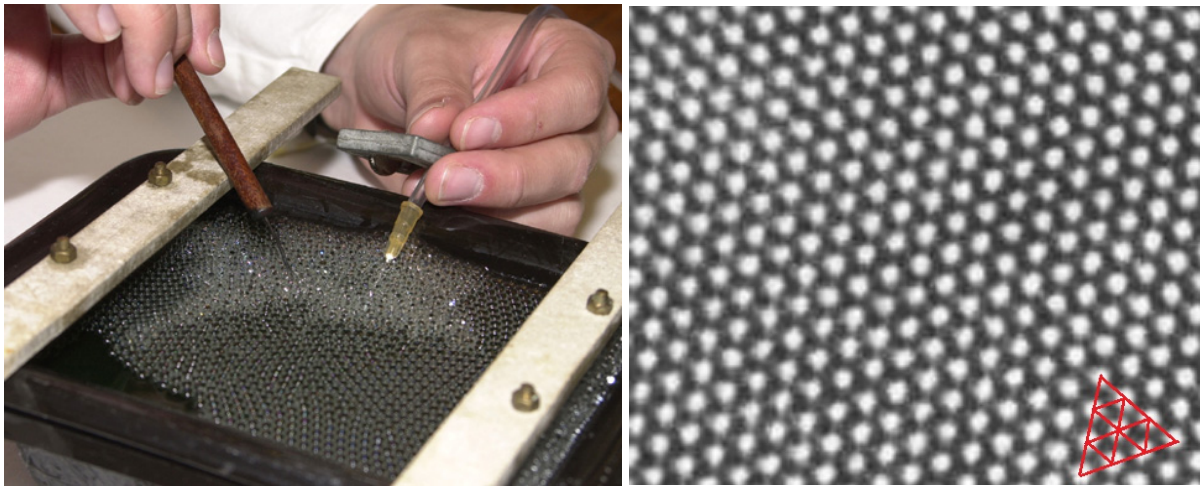
protein branches provide the driving force for the folding. The binding force in this case is the same as the driving force. The weak forces between the branches eventually enter a global minima for the system as the protein enters its final shape. The environment, the cytoplasm of the cell, allows the protein molecule to freely move into its final shape.

Self-assembly has also been shown to work on not only microscale structures, such as those within biological systems, but also on macroscale and nanoscale structures. Several experiments have been conducted on macroscale self-assembly. These experiments were conducted using entirely differing forms of the four features that enable self-assembly. The driving forces of these experiments included magnetic, electromagnetic, and the capillary forces of water, as well as manual excitation of the self-assembling system.

The bubble raft was an early experiment created in order to model nanoscale crystalline structures [6][7]. During the bubble raft experiment, bubbles were created by forcing air through a small pipette into a soapy solution by use of a motor. This method of bubble creation ensured that the bubbles created during the experiment were similar in size, which was necessary in order to properly model atoms within a crystalline structure. Due to the nature of the soapy solution and the capillary forces between the bubbles, the bubble raft would naturally form a triangular tiling as shown in Fig. 1.2. It is of note that the structure formed by the bubble raft closely resembles the structure of aluminum in the same figure.

An example of using magnetism and electromagnetism as the driving force for self-assembly can be shown by the experiments of Golosovsky et al.. In Golosovsky et al.'s experiments, disk shaped magnets were placed inside larger styrofoam disks. These disks were then placed on the surface of a liquid held within a cylindrical container. These disks would eventually form triangular packings on the surface of the liquid due to the repulsion between disks. A current carrying wire was wrapped about the container holding the liquid in order to apply a repulsive magnetic force to the disks, forcing them inwards. As more current was applied to the wire, which created a stronger repulsive force, the disks closest to the walls of the container, forming the perimeter of the disks, took the shape of a hexagon. In this experiment, the driving

forces, the magnetism from the disks, and the external force from the wire, allowed the disks to self-assemble into a hexagonal shape.



(a) Bubble Raft (b) STM of Aluminum

Figure 1.2: Example of patterns formed by self-assembly. (a) shows a bubble raft formed using soap water and a air pump [9]. (b) shows an STM image of aluminum [10] overlaid with bonds in the bottom right showing the triangular pattern. Here we can see that both the bubble raft and the aluminum form very similar tightly packed arrangements.

In the nanometre scale, self-assembly occurs very similarly to the macroscale methods presented above. However, instead of bubbles or disks, the structured particles in question are atoms or small molecules. Self-assembly on this scale is often driven by electrostatic forces. In the presence of gold or platinum plating, along with other molecules, carbon has been found to assemble into Graphene sheets, using platinum as a catalyst [11]. The resulting Graphene sheets, or lattices, have a regular pattern. An example of the hexagonal tiling can be seen in Fig 1.3. Other molecular lattices have also shown the ability to self-assemble in the presence of catalysts, such as hexagonal Boron Nitride (h-BN) on Ruthenium [12].

Designing a self-assembling system can be broken down into three problems: (i) given the final structure of the system how do we design our four features; (ii) given an initial configuration of a system can we predict the final structure; and (iii) given a self-assembling system what is the yield of the self-assembled structures. We can illustrate these 3 problems using

proteins folding as an example. The first problem can be stated as "If given a protein, what was sequence of folds that produced its shape?" and a question to represent the second problem could be "When given a sequence of amino acids, what will the shape of the resulting protein be?".

In the case of protein folding, the issue of yield cannot be posed as a meaningful question as each sequence of amino acids will produce a folded protein unique to that sequence. However, the yield problem is still of particular interest within most self-assembled systems. The yield of self-assembly in biology can be easily altered by a number of different factors present within the cell structure. For example, the concentrations of the amino acids within the cell can greatly affect the formation of a protein.

In macroscale experiments conducted by the Whitesides group [13] involving the assembly of chains of straws, the group found that when more than a single chain was allowed to self-assemble, longer assembled chains could hinder the process of self-assembly for shorter chains by blocking access to single structured particles. This is a perfect example of the yield problem. In this case, due to the design of the system, larger chains hindered the growth of smaller chains. In the case of the bubble raft, it is possible that the experiment could yield a bubble with multiple grain boundaries or none.

Understanding molecular self-assembly is of great importance to the materials science community. As mentioned previously, atoms have the ability to form regular patterns within the presence of catalysts. The resulting structures from these methods can be greatly useful. For example, Graphene, single layer graphite, has several properties which make it a valuable material. Graphene is not only one of the thinnest materials discovered, being a single atom thick, but also one of the strongest and the best conductors of electricity that is currently known [14]. It has been used, in conjunction with other materials, to produce nanoscale circuits and flash memory. Self-assembly allows for these materials to be produced in very efficient ways that would otherwise be overly costly where such costs would be prohibitive [1].

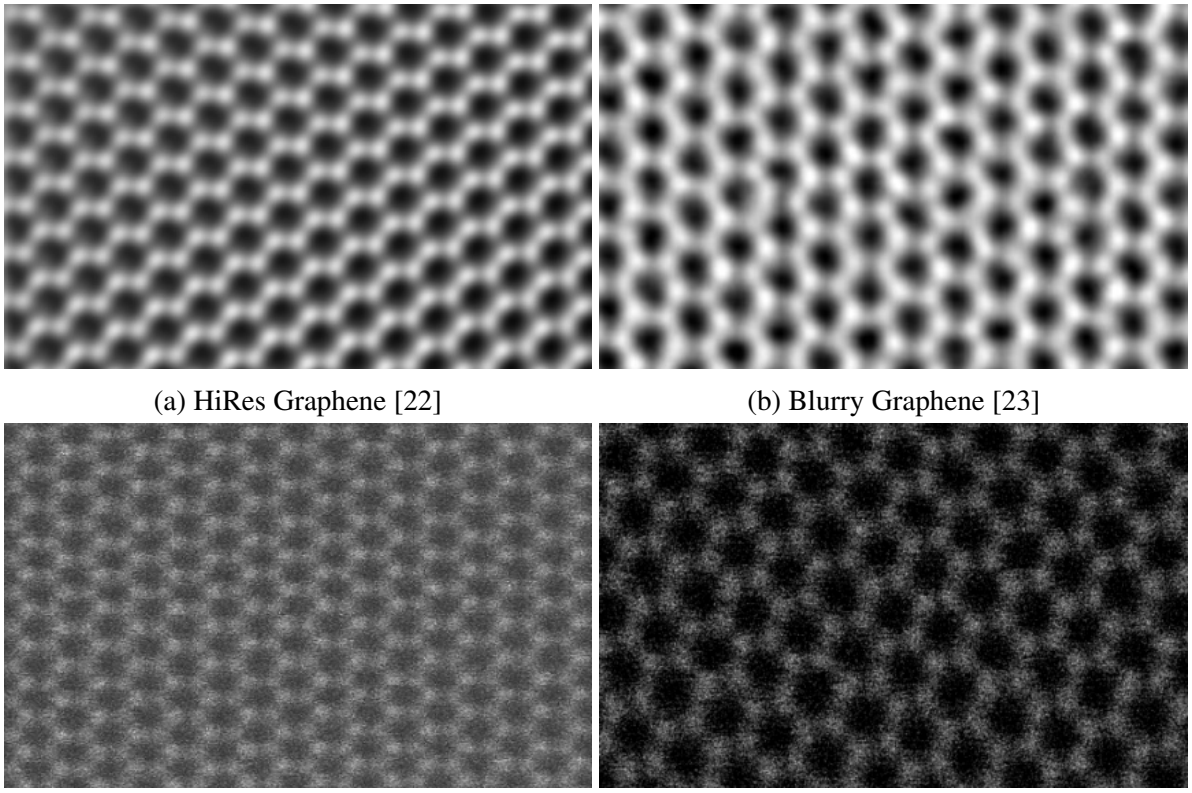
1.2 Lattices

Lattices are arrangements formed by either atoms, ions, or molecules. These arrangements can take on various shapes or patterns. Two dimensional planar lattices can be an interesting subject of investigation, especially within the context of self-assembled lattices. These lattices, created through molecular self-assembly, can form materials that are stronger or more tensile than traditionally used materials within the manufacturing industry [15]. In particular, Graphene and hexagonal boron nitride (h-BN) are of interest due to their properties and potential uses.

Graphene is a single layer of carbon atoms that can exhibit several differing properties based on the condition of the environment. It can be produced by "peeling" layers off of graphite using mechanical exfoliation [16], and via self-assembly by chemical vapor deposition on various substrates [17]. Graphene's properties are of interest in the field of materials science due to its ability to be used as a conductor and insulator in nanoscale circuitry through the application of differing voltages [16][18]. The applications of Graphene within nanoscale circuitry are not only limited to conductors, Graphene is also capable of being used as capacitors [19] and transistors, and within biomedical applications Graphene has been used for cell growth control, mass spectrometry and stem cell differentiation [20][21].

Hexagonal boron nitride (h-BN) is another material that can have various nanoscale applications due to its properties. h-BN, like Graphene can be made via chemical vapour deposition [12] and via mechanical exfoliation [25]. h-BN can be used for compact lasing devices [26], however when used as a substrate for Graphene, h-BN allows Graphene to provide improved device performance when compared to devices using other substrates [27][28].

These two monolayer lattice materials can both be used to create nanoscale circuitry and is currently the subject of research and study. It is for these reasons that monolayer lattices are of importance and worth investigation. This thesis uses these materials for testing purposes as they are relevant to further research within the field of nanoscale circuitry and they can provide real world applications of the methods proposed within this thesis.



(a) HiRes Graphene [22] (b) Blurry Graphene [23]

(c) High Background h-BN [24] (d) High Distortion Graphene [24]

Figure 1.3: Experimental images from TEM. The images in the top row and left column are of planar Graphene lattices while the image in the bottom right is a planar hexagonal Boron Nitride lattice.

1.3 Atomic Scale Imagery

Atomic scale imaging is the process of creating images of materials where the atoms comprising the materials within the image are individually distinguishable. Atomic scale imaging helps characterise molecular and other nanoscale structures by allowing people to view the exact atomic structure present within the image. These images in particular help the study of and help improve the process of self-assembly by allowing researchers to view the structures prior to and after the self-assembly process has occurred. There are many methods of creating atomic scale imagery, however, these methods can generally be divided into two “camps”, which are Scanning Probe Microscopy (SPM) and Electron Microscopy (EM). Currently, the two most popular methods of the two camps are Atomic Force Microscopy (AFM) and High-

Resolution Transmission Electron Microscopy (HRTEM). Note that from here on the "M" in SPM, EM, TEM and AFM can stand for "Microscope" and "Microscopy" interchangeably.

1.3.1 Atomic Force Microscopy

Atomic Force Microscopy (AFM) is a method of imaging materials and structures with nanoscale resolution by capturing the amount of force applied near the surface of those materials. AFMs use a probe to scan the surface of the material. This method of imaging is called Scanning Probe Microscopy. AFMs have several methods of operation, they are able to capture either the height of a surface or the force applied to the probe by the surface of the sample depending on if the user chooses to either maintain a constant level of force applied to the tip or a constant level of height from the surface.

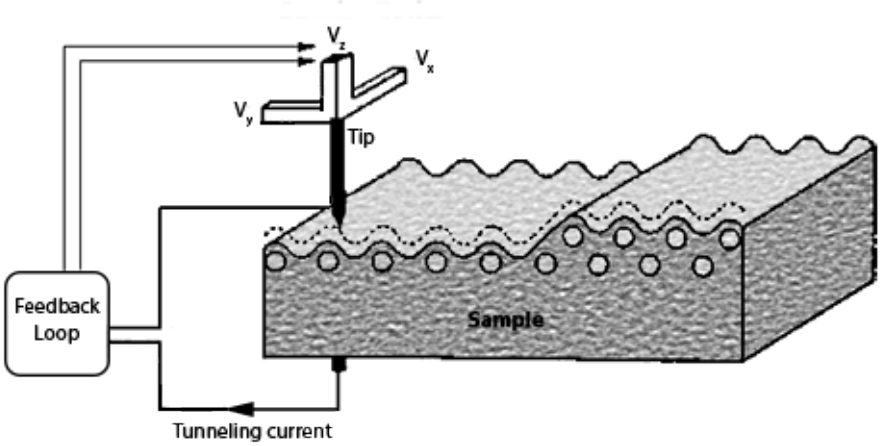


Figure 1.4: A simplified schematic of an STM [29]. A tip is scanned over the surface of a sample. The tip uses quantum tunneling and the voltage is measured at each point along the scan. The feedback loop provides the measuring device for the quantum tunneling as well as informs the microscope of how the tip should move heightwise along the sample.

The precursor to Atomic Force Microscopy was Scanning Tunneling Microscopy (STM). The STM is another SPM that used the tip of the probe to send a current through the sample to scan the surface of the material. Due to the nature of the capture process, the STM could only capture information from conductive materials. By using force to measure the surface, instead

of current, AFMs are able to measure insulators as well as conductors and can also perform surface measurement in many different mediums, which include liquid and gas mediums. Since AFMs and STMs are both SPMs and are similar in construction and operation, occasionally AFMs are equipped with the components necessary to perform the quantum tunnelling necessary for STM captures. An example of the construction and method of operation of an STM is shown in Fig. 1.4.

Construction

An AFM consists of several very key components. Most atomic force microscopes are constructed such that the probe, depicted in Fig. 1.5 sits above the sample where its fine movement is controlled by piezoelectric ceramics which lay above the probe.

Below the probe lies the sample stage. The sample stage allows for any specimens or samples to be laid upon it as well as control the coarse movements along the width and length of the sample. Below the sample stage exists the motors necessary to perform coarse movements perpendicular to the sample's surface. All these components, plus the rigid body that supports them is called the mechanical loop of an AFM.

The resolution from the AFM and the noise produced during the capture process of the AFM are both directly related to the rigidity of the mechanical loop. The more rigid the loop is the less noise is produced within the resulting image. As a result, the resolution for the AFM increases along with rigidity. It is typically easier to make the mechanical loop more rigid by making the microscope smaller. It is for this reason that AFMs are usually small, however the size of an AFM limits the maximum size of the specimen that is able to be captured. A schematic of the mechanical loop is shown in figure Fig. **??**.

Movement of the probe is controlled using piezoelectric ceramics. Piezoelectric ceramics, or piezoelectrics and piezos for short, allow the AFM to function as it does. Piezos enable the the AFM to perform the fine movements necessary to scan the probe. In order to perform the movements, piezos in the shape of cylinders can be grown or shunk through the application of

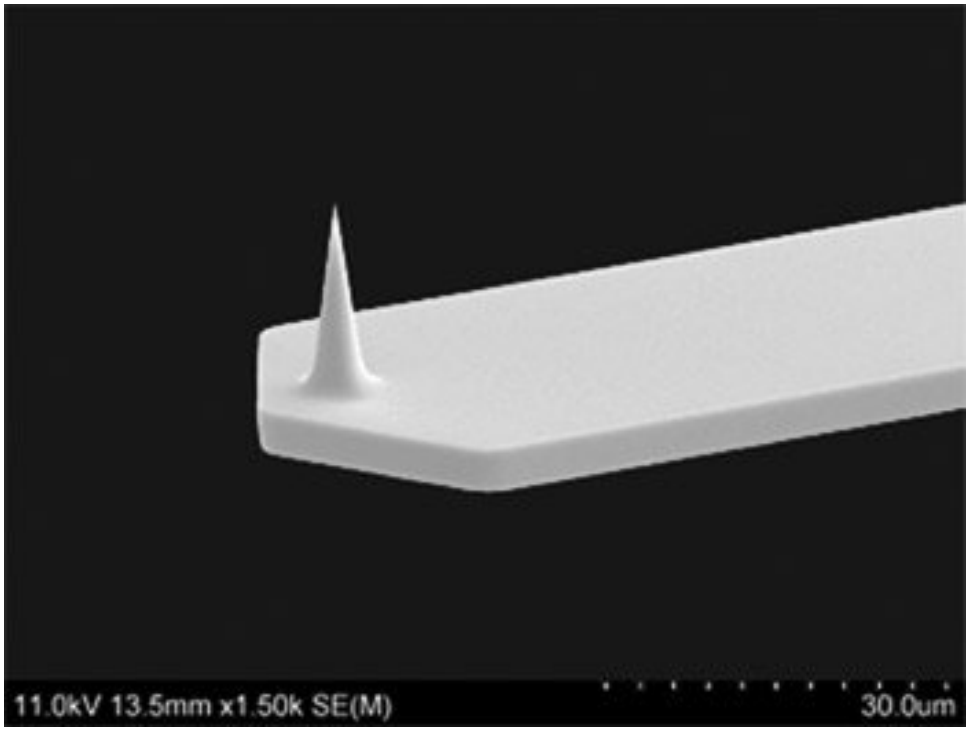


Figure 1.5: The tip of an Atomic Force Microscopy which is used to record the amount of force applied to it by a sample surface [30].

voltage through the piezoelectrics. Depending on the construction of the piezo, it is possible that for every volt applied to the piezo, the piezo will grow by one nanometer. This resolution for the growth of the piezoelectric ceramic allows for the precise movement of the probe tip across the surface. During the growing process of the piezo, the volume of the piezo will remain the same throughout the process. By measuring the amount of voltage applied to the piezo, the amount the piezoelectric has been grown or shrunk, and by extension how much the probe tip has moved, can be determined.

However, the growth of the piezoelectrics are not perfectly linearly related to the voltage applied. Often there is an amount of error which can be cause by several different factors. Two factors that cause errors are hysteresis and creep. Hysteresis is the tendency of the peizo to maintain its shape despite the changes in voltage applied. Creep is the tendency of piezos to continue growing or shrinking past when a large voltage is applied. Creep often occurs when attempting to scan a different part of the sample that is distant from the original scanning loca-

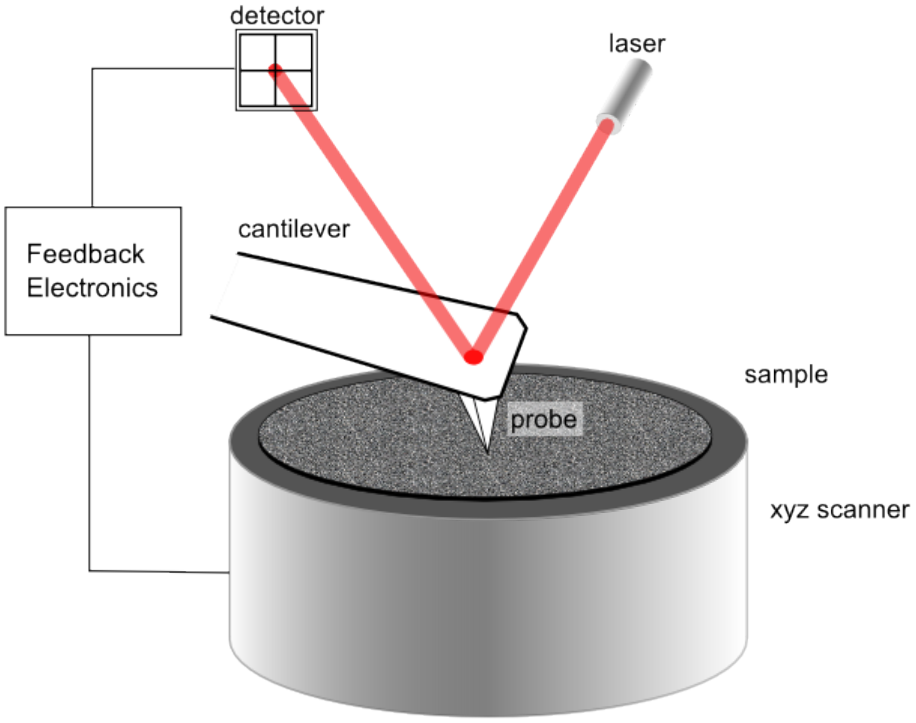


Figure 1.6: The Schematics of an Atomic Force Microscope [31]. The laser is used to measure the deflection of the cantilever caused by the forces of the sample's surface applied to the probe's tip.

tion. Calibration of the AFM using a well known material can be done to solve the problems of hysteresis however they do not solve the problem of creep. External sensors, possibly an optical sensor, can be used to direct the probe and detect both hysteresis and creep. Using an optical sensor however can create noise, have misalignment problems and can cause thermal drift.

1.3.2 High Resolution Transmission Electron Microscopy

High Resolution Transmission Electron Microscopes, or HRTEM/HREM, pictured in Fig. 1.7, are tools that are capable of imaging an object's atomic structure via the use of projected electrons. Transmission Electron Microscopy (TEM), differs from other methods of Electron Microscopy, for example Scanning Electron Microscopy, through its ability to capture electrons

that have passed through the sample instead of those that have been reflected off the sample. By transmitting the electrons through the sample, microscopists have been able to achieve higher resolutions and magnifications than other forms of electron microscopy.

Figure 1.7: A Scanning Transmission Electron Microscope on desk with a workstation to the right of the device [32].

Prior to the existance of HRTEM, biologists used phase-contrast microscopes in order to view cells and study cell division [33]. However, as the thickness of samples decreased, phase-contrast microscopy failed to portray the sample with sufficient contrast. Using improved sample preparation techniques, the electron microscope is capable of providing similar levels of constrast as earlier phase-contrast microscopes.

Most implementations of TEMs contain several common components. These include condensing and imaging lenses, in order to focus the electrons used to capture sample information, an electron gun, in order to fire the electrons at the specimen, a specimen stage to hold the samples, and an aperture in order to record the intensities of the captured electrons.

Types of HRTEM

There are several different forms of HRTEM, however only two forms will be discussed. Conventional microscopy will be described due to its historical meanings and Scanning Transmission Electron Microscopy (STEM), will be described because of its relevance to the topic of focus. It is worth it to be aware that there are other notable forms of TEM such as cryoTEM.

Conventional TEMs, Fig. 1.8 function more similarly to an optical microscope than the STEM does. In a conventional TEM, a beam of electrons pass through several lenses in order to focus the beam to illuminate a sample. The electrons which transmit through the sample projected onto a viewing screen or capture by a camera sensor. The electron beam, or primary electrons, functions similarly to the light beam of an optical microscope. Using an electron beam we can gather information from the sample in many ways. There can be electrons that do not interact with the sample and pass directly through it, and there are also electrons that are elastically and inelastically scattered. Backscattered electrons are electrons from the electron beam that have had elastic interactions with the specimen which has altered their trajectory and sent the incident electron back. Secondary electrons are produced when there is an inelastic collision with the specimen and an electron from the specimen is ejected from an atom due to acquiring energy from the incident beam. Finally, auger electrons can also be ejected from the specimen, they are caused by an atom attempting to reach equilibrium after a secondary electron has been ejected from the same atom. The electron beam can also produce x-rays when interacting with the sample. These sources of information can all be used to extract details from the sample and the primary, secondary, backscattered and auger electrons can both be used to create images. Samples being examined by a TEM must be very thin, around 100 nanometres.

The STEM uses a very fine, highly focused beam of electrons, more so than the conventional TEM, to scan over a sample. In an STEM, an image is constructed by scanning the sample slowly using the beam of electrons and capturing the transmitted electrons at each point in the scanning process. An STEM, being a variety of TEM, also produce the same

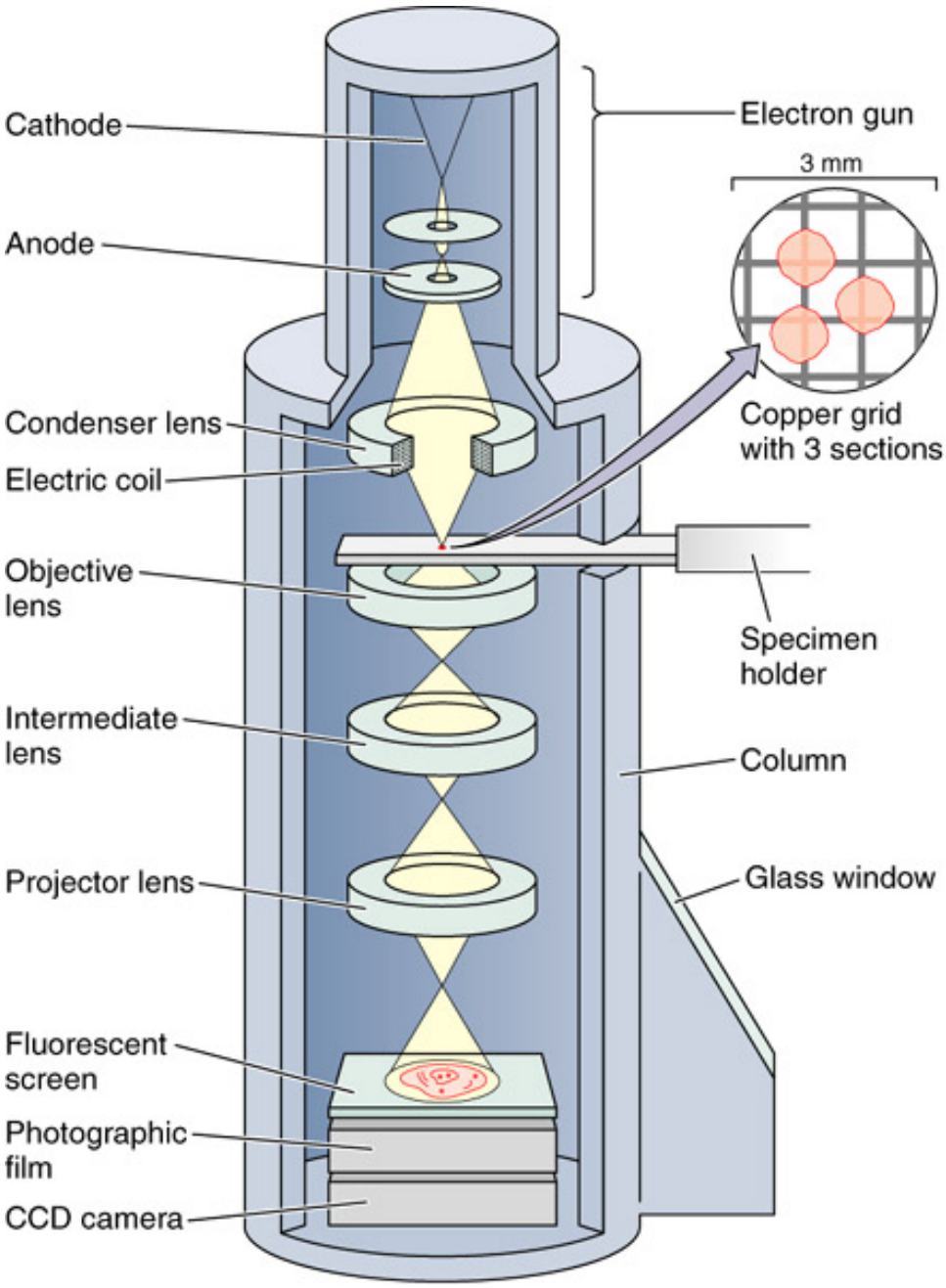


Figure 1.8: The Schematics of a Transmission Electron Microscope[34].

secondary electrons and x-rays that conventional TEMs produce, albeit on a much finer level. This confers to the STEM the same analytical abilities that a TEM has but on a larger or higher resolution. Due to the nature of the STEM, the STEM can provide higher resolutions than conventional TEMs while also being able to image thicker samples, 2 micrometres thick at

200 keV instead of 0.5 micrometres thick. The ability of the STEM to image thicker samples allows for easier sample preparation.

STEM provides many advantages over TEM, higher resolution, easier sample preparation, etc., however STEMs function much more slowly than conventional TEMs as TEMs image the entire sample at once. This means that TEM can provide more information of an ongoing process than an STEM can provide. Table 1.1 gives an overview of some of the capabilities of TEMs and the other microscopy methods.

1.3.3 Analysing Atomic Imagery

Due to the nature of the capture processes and the resulting imagery of both Atomic Force Microscopy and High-Resolution Transmission Electron Microscopy, the imagery from both methods can be analysed similarly to 2 dimensional grayscale images that are produced via optical methods such as those from an everyday digital camera. All three methods, AFM, HRTEM and optical, produce a grid of coordinates with corresponding values which can be regarded as pixels. The x and y coordinates can be likened to the width and height of an image. The third value, or z coordinates, can be compared to the pixel's intensity.

In most cases, a height map of a surface using regularly spaced points can be viewed as an image and vice versa. The output of an AFM can be likened to the height map of a surface because they map a discrete grid above the sample by measuring or maintaining the force applied to the tip of the cantilever.

A grayscale image can also be viewed more similarly to the height map of a surface if we take the z coordinate to be the height of the image and the x and y coordinates to be the width and length of the image. By viewing the output of both AFM and HRTEM imagery in this manner we can use traditional computer vision methods to analyse the image.

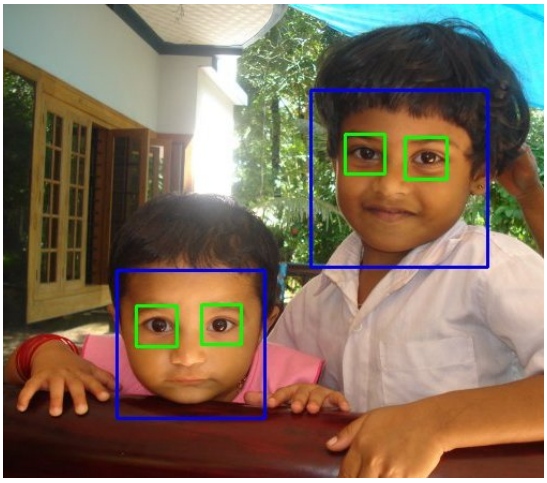
Type	STM	TEM	AFM	STEM
Capture Method	Probe Based	Electrons	Probe Based	Electrons
Media	Vacuum, Gas, Liquid	Vacuum	Vacuum, Gas, Liquid	Vacuum
Image Conductors	Yes	Yes	Yes	Yes
Image Non-Conductors	No	Yes	Yes	Yes
Image Dimensions	2D, 3D	2D	2D, 3D	2D

Table 1.1: Capabilites of each of the nanoscale imaging methods presented earlier.

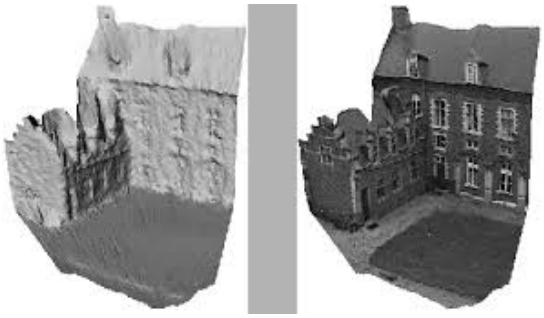
1.4 Computer Vision

Computer vision is a subfield of computer science that specializes in the application of computer science to images and video as well as the study of image processing methods. Computer vision methods are able to extract descriptions of the world, or a system, through a picture or a sequence of pictures. There are many uses of computer vision present throughout many industries, examples of which are depicted in Fig. 1.9. For example, the film industry has used digital reconstruction in order to reproduce buildings and scenes to aid in the development of computer generated imagery used in films. Motion capture is also used extensively within the film industry and video gaming industry to accurately represent realistic human movements in those media. Several industries use computer vision to perform inspection of produced products before their sales. Each of these cases use different setups in order to perform their task. Motion capture and digital reconstruction typically use several cameras in order to produce digital representations of real objects, however if data from another source is present a single camera can be used [35, 36]. For the industrial inspection of products, typically a single camera is used along with a model for the expected appearance of the product [37]. This method of industrial inspection closely resembles the requirements for the problem of detecting atoms within atomic scale imagery. We have an input image and a model for the appearance of an atom within that imagery.

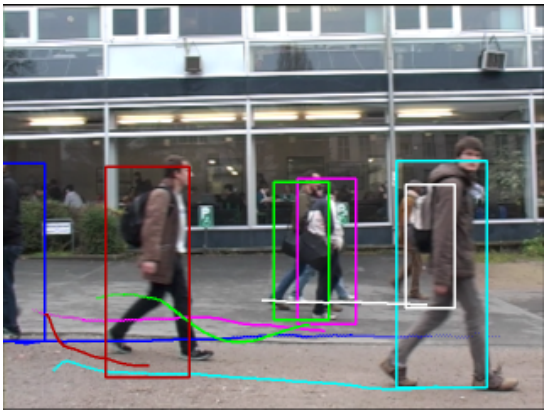
Computer vision has been a field of study since the 1960s after Larry Roberts discussed the possibilities of acquiring three dimensional information from two dimensional perspectives of polyhedra [42]. As researchers realised that images taken from non laboratory conditions were



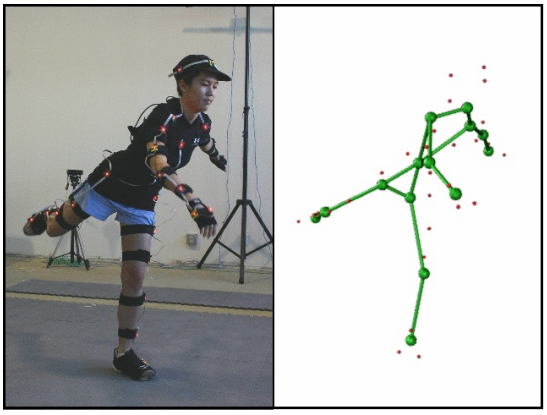
(a) Face Detection [38]



(b) 3D Scene Reconstruction [39]



(c) People Tracking [40]



(d) Motion Capture [41]

Figure 1.9: Various applications of Computer Vision. (a) uses Haar cascading classifiers in order to detect faces and its components, (b) is a 3-dimensional reconstruction of a building using stereo vision, (c) is the output of a people tracking algorithm and (d) is the outputting a 3-dimensional skeleton of the person in order to capture his motion.

often more complex than the initial block images used by Roberts, research was directed into understanding scenes on a lower level, such as detecting edges within an image and segmentation. These low level scene understandings provided the basis for a bottom up approach to better understanding scenes in both images and video. David Marr later proposed a bottom up framework for understanding scenes that still has much use today [43]. It argued that the description of an information processing system had three levels, the computational theory level, the representations and algorithms level and the hardware implementation level. The

computational theory level describes the goal and the constraints of the system. The representations and algorithms level describes how the input and output are represented as well as the algorithms used to calculate the result. The hardware implementation level describes how the representations and algorithms are mapped onto actual hardware.

Computer vision today is capable of many applications, such as facial recognition, object tracking in video across complex backgrounds [44], Optical Character Recognition, and 3D reconstruction using stereo vision. Each of these applications have several possible solutions that can be used to implement the application. Facial recognition can be done using a combination of skin color detection and simple features or it can be done using cascading classifers which are able to identify the individual components of a face, eyes, nose, etc., as well as the entire face. Object tracking also can be done using several methods. Color histograms, background subtraction, and complex features such as SIFT[45], SURF[46], and ORM[47], have all been used to perform object tracking. With so many applications having such a large variety of implementations and solutions, it is sensible to consider computer vision methods in order to perform atomic detection.

1.5 Contributions and Outline

In this thesis, we study the problem of detecting atoms within High Resolution Electron Microscopy (HREM) images and the problem of automatically and digitally reconstructing molecules from within said images. We present two techniques for segmenting the HREM images. The first is an adaptive thresholding based method. It was noticed that this method loses performance on images with a large range of varying intensities. The second is a curvature based method which uses the second order derivatives of the image. We also address the problem of creating synthetic TEM images for the purpose of testing the automated system.

The contributions of this thesis are as follows: (i) a method of quickly synthesizing TEM images from pre-existing atomic coordinates, (ii) a method of detecting atoms within HREM

images and (iii) a method of reconstructing a lattice from a set of 2D coordinates.

The remainder of this thesis is organised as follows. In the next chapter, we review background knowledge necessary to understand the work and review related work. There we focus on understanding the components used in the system in order to accurately reproduce it. Chapter 3 presents our proposed system and a method for synthesizing Electron Microscopy images. Implementation details, testing details and results are then shown in Chapter 4. Implementation details include the platform used and the interfacing of different technologies. The testing details explain how the constructed images were tested and the tests' purpose. In the results section we compare our works with the existing work in the field. Chapter 5 concludes the thesis and presents possible future work on the subject.

Chapter 2

Background and Related Work

In this chapter we will review both the background work necessary in understanding the atomic detection algorithm proposed in this thesis, as well an existing solution currently in the field. The proposed method in this thesis uses several image processing and image analysis methods that could be their own subject of discussion. After discussing important background knowledge, a brief review of the existing work will be presented.

2.1 Computing Image Derivatives

Derivatives are a method of mathematically describing the rate in which a given function will change as its input changes. Taking the derivative of an image can reveal more complex patterns within the image and make image processing easier. Due to the discrete nature of images, exact image derivatives cannot be computed and usually are approximated using finite differencing methods, i.e., methods of the form

$$\frac{df}{dt} = \frac{f(x_2) - f(x_1)}{x_2 - x_1} \tag{2.1}$$

this function only applies to 1D images and functions, most images have 2 dimensions that must be considered and therefore we must compute partial derivatives of the image in both

dimensions, i.e.

$$\frac{\partial f}{\partial x} = \frac{f(x_2, y) - f(x_1, y)}{x_2 - x_1}, \frac{\partial f}{\partial y} = \frac{f(x, y_2) - f(x, y_1)}{y_2 - y_1} \tag{2.2}$$

In computer vision, it is possible to approximate these finite differencing methods through the use of convolution. Fig. 2.1 shows a number of different forms of image derivatives for a given image.

Convolving an image with a Sobel filter is a common method of calculating the derivative of an image. A Sobel filter, S_x or S_y, convolves an image with a kernel that typically has the form

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{2.3}$$

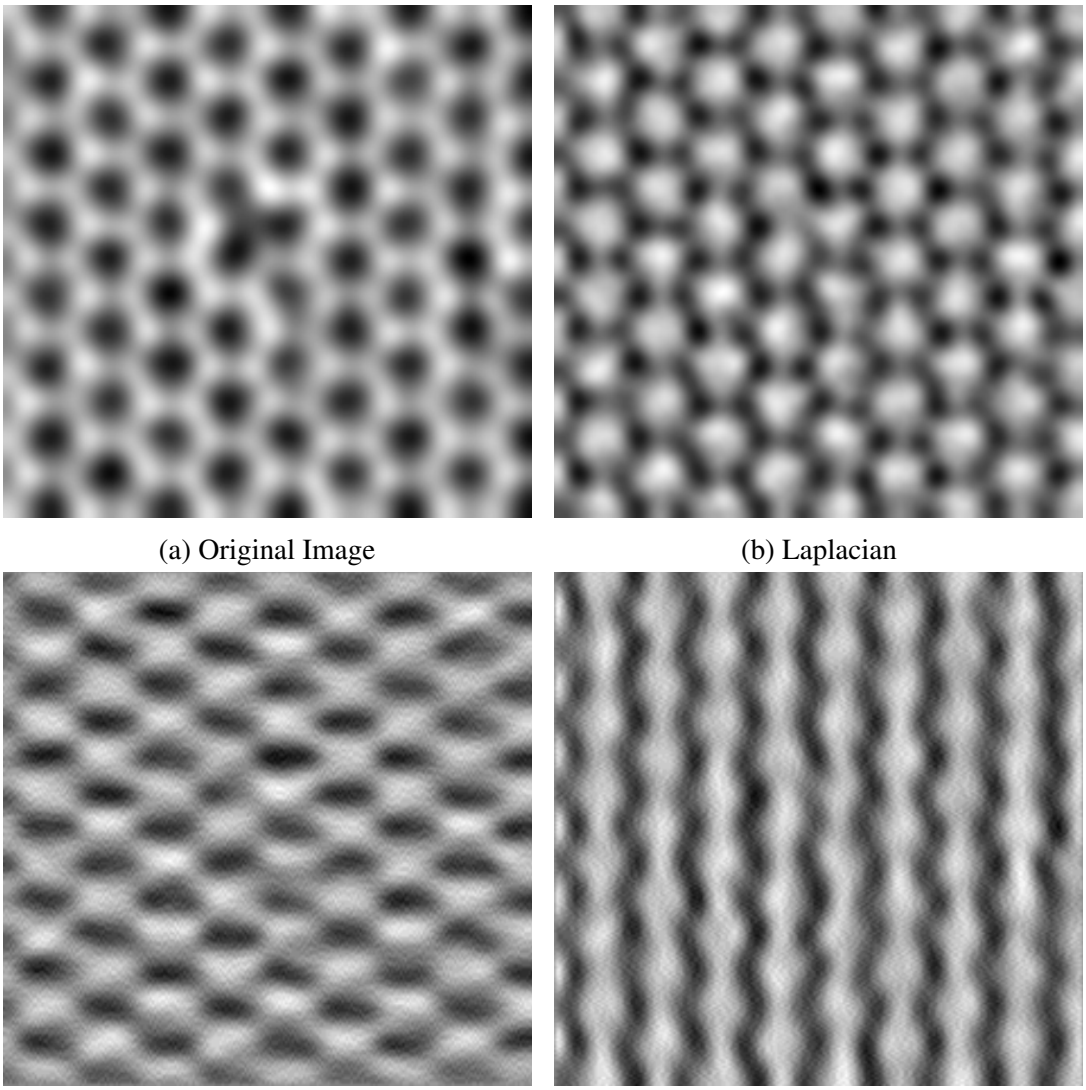
for the x direction, and

$$S_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{2.4}$$

for the y direction, assuming we are using a 3x3 filter. It is worth noting that sobel filters have a slight smoothing effect on the resulting derivative images.

A Laplacian filter, L, is a useful method for calculating the second derivative of an image. Approximations of the Laplacian filter can take the form of

$$L = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \tag{2.5}$$



(a) Original Image

(b) Laplacian

(c) 2nd Order x Derivative

(d) 2nd Order y Derivative

Figure 2.1: Possible second order derivatives of an electron microscopy image [48] with (a) being the original input image. (b) is the laplacian of the image which is a form of discrete second order derivative. The laplacian gives us an example of what information can be extracted from the second derivatives of an image. (c) and (d) are second order derivatives computed using finite difference methods with (c) and (d) being computed with respect to the y and x axes, respectively.

or of the form

$$L = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \tag{2.6}$$

which also includes the information from the diagonals.

In our work we found Sobel and Laplacian filters did not provide the smoothing necessary in order to produce suitable derivatives. Instead we used a simple discrete representation for the kernel in conjunction with convolution with a Gaussian kernel to calculate the derivative of an input image. The reason for this discrete representation is to remove the possibility of over smoothing or unwanted smoothing from the derivative images. This approximation of a derivative is based on first principles of derivatives. The kernel for this derivative takes the form

$$D_x = \begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \tag{2.7}$$

for the x direction, and

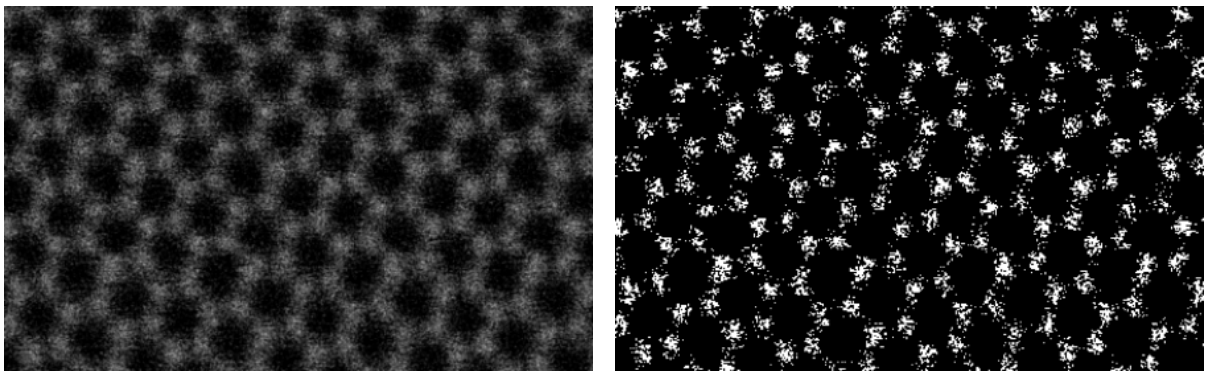
$$D_y = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{2.8}$$

for the y direction.

2.2 Segmentation

Segmentation is the act of grouping components of an object with respect to features within the image. The purpose of segmentation is that the resulting groupings will contain similar features and can provide a higher level understanding of the object as a whole [49].

In the context of image processing, segmentation often means grouping pixels of the image based on their features, such as color, intensity and location. These groupings can also provide



(a) (b)

Figure 2.2: Example segmentation of a gray scale image [24] created by flagging pixels that are higher than a single intensity value. This intensity thresholding method was applied uniformly throughout the image.

information about the objects within the image and distinguish objects from other objects and the background. Image segmentations are usually represented as images themselves as it provides the most accurate representation of the groupings. A binary segmentation image is an image segmentation that classifies objects as either belonging to an object or the background. The image only uses two values to achieve that goal. A value of 1 for an object and a value of 0 for the background. Fig. 2.2 shows a binary segmentation of a TEM image where several areas within atoms do not have values of 1. This figure highlights both the importance of selecting an appropriate method for segmentation as well as output of an inadequate method.

2.3 Finding Image Contours

The act of finding image contours is a fundamental method in image processing. From a binary segmentation image, one can use methods to derive a sequence of coordinates that represents a border between the background and an object.

There are two components to finding image contours, detecting borders and following borders. Border detection searches for borders within the binary image based on specific criteria. This criteria can change based on the algorithm used, however most border detection methods begin by scanning each individual pixel of the image sequentially and halting the algorithm

whenever a border is found [50]. Once a border pixel is detected, the rest of the border is identified by following the normal of the gradient around the border. The normal of the gradient is followed in both directions to ensure that the entire contour is followed in the case that the contour is not closed.

2.4 Calculating Moments

Contour and image moments are a gross measure of a contour, which is computed by integrating over all points in a contour. For example, these moments contain several key characteristics about the object in question. The zeroeth moment is the area of the contour and by combining the zeroeth moment and the first order moments we can compute the centroid of the contour.

The moments of an image are calculated using the integral

$$\mathcal{M}_{pq} = \iint I(x, y)\ x^p y^q\ dxdy \tag{2.9}$$

where p is the order of the x axis elements and q is the order of the y axis elements. We can derive the integral used to calculate the image moments of a contour using Green's Theorem [51]. For example, in order to calculate the zeroeth moment of a contour the integral becomes

$$\mathcal{M}_{00} = \iint I(x, y)\ dxdy = \oint_C \frac{x\ dx - y\ dy}{2} \tag{2.10}$$

where C is the contour that is being integrated. Since the images and contours are discrete objects, rather than continuous, the integrations can be treated as summations across the image or contour.

2.5 Convex Hull

The convex hull of a set of points is the smallest convex region enclosing the set of points. In mathematical terms, a subset plane $\mathcal{S}$ is considered convex if for any points $\mathbf{p}, \mathbf{q} \in \mathcal{S}$ the line segment $\overline{\mathbf{pq}}$ is contained within $\mathcal{S}$. The convex hull, $\mathcal{Q}(\mathcal{S})$ is the smallest convex set that contains $\mathcal{S}$ [50].

Eddy [52], proposed a method for computing the convex hull of a set of points using a tree structure and partitioning. Eddy partitions the input polygon based on lines between its points. These subsets of the polygon are used as nodes in the tree. The algorithm continues to partition the polygon until there are no more partitions to be made. At this point, all partitioning lines that have a null subset are considered to be lines within the convex hull. It was reported that the alogrithm has a time complexity of $O(n^4/3)$ by empirically testing the algorithm.

Graham and Yao proposed LeftHull, a method to find the convex hull of a simple polygon [53]. A simple polygon is a polygon that does not have any intersections. Graham and Yao's algorithm uses the points of the polygon directly and move counterclockwise along the polygon in order to find the convex hull. Graham and Yao reported that the LeftHull algorithm has a linear time complexity.

In 2010, Zhang et al. proposed a method of calculating the convex hull within binary images. This method begins by scanning the image for extremal points in the x and y axes of the polygon. This is in order to reduce the number of points being analysed by the algorithm. Once all extremal points have been identified, scanning continues in varying directions in order to identify other points within the convex hull and to establish a temporary convex hull. A complex scanning technique is then applied to the temporary convex hull and retrieves the convex hull from the image. Zhang et al. reported that this algorithm completes within $O(n^2 - s) + O(n)$ time, where n is the size of the image and s is the number of pixels within the polygon.

2.6 Ellipse Fitting

Fitting ellipses to contours is neccessary to determine several key characteristics about contours. These characteristics can later be statistically analysed and used to determine outlying or exceptional contours. The greatest advantage to fitting an ellipse to a contour is that it can calculate the eccentricity of the resulting ellipse parameters and use that value to perform analysis on the ellipses and detect outliers. There are several algorithms for fitting ellipses, such as the algebraic distance, geometric distance and approximate mean square [55]. In our work it was found that the algebraic distance was sufficient for our methods.

The algebraic distance algorithm minimizes the objective function, $\epsilon^2(\mathbf{x}) = \sum_{i=1}^{n} \delta(C(\mathbf{x}), \mathbf{p_i})$ where $\mathbf{x}$ is a vector that contains parameters for a family of curves $C(\mathbf{x})$ and $\mathbf{p_i} = (x_i, y_i)$ is a point within the contour. In the case of the algebraic distance algorithm the objective function can be rewritten as

$$\epsilon^2(\mathbf{x}) = \sum_{i=1}^{n} F(\mathbf{x}, \mathbf{p_i}) = \|A\mathbf{x}\|^2 \tag{2.11}$$

where

$$F(\mathbf{x}, \mathbf{p_i}) = [A_{xx}\ A_{xy}\ A_{yy}\ A_x\ A_y\ 0] \cdot [x_i^2\ x_i y_i\ y_i^2\ x_i\ y_i\ 1] \tag{2.12}$$

$$= \mathbf{a_i} \cdot \mathbf{x} \tag{2.13}$$

and where A is an nx6 matrix where the rows are the individual $\mathbf{a_i}$ vectors and has the constraint $\|\mathbf{x}\|^2 = 1$. We can constrain the objective function, $E = \|A\mathbf{x}\|^2 - \lambda(\|\mathbf{x}\| - 1) = \mathbf{x}^T A^T A\mathbf{x} - \lambda(\mathbf{x}^T\mathbf{x} - 1)$, and minimize it to form the eigenvector problem

$$\nabla E = 0 <> 2A^T A\mathbf{x} - 2\lambda\mathbf{x} = 0 \tag{2.14}$$

where λ is a Lagrange multiplier. The best fit for the ellipse would be the eigenvector of $A^T A$ corresponding to the smallest eigenvalue, which can be easily found by finding the single value decomposition of $A^T A$.

2.7 Connectivity and Neighbourhoods

The terms connectivity and neighbourhood varies depending on the object in question. This paper uses the word neighbourhood in the context of both pixels and atoms. In the context of pixels, a neighbourhood can either be the 4-connected neighbourhood about a pixel or the 8-connected neighbourhood. Fig. 2.3 contains a visual representation of the concepts.

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{matrix} \qquad \begin{matrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{matrix}$$

(a) 4-connected neighbourhood (b) 8-connected neighbourhood

Figure 2.3: Shows the appearance of the two kinds of connected neighbourhoods. Within each image the pixel with the value 2 is the pixel of focus, and pixels with a value of 1 are within the neighbourhood.

A 4-connected neighbourhood about a pixel includes only the pixel's immediately adjacent pixels along the x or y axis of the image while an 8-connected neighbourhood includes the diagonals as well. In the context of a molecular lattice, the neighbourhood of an atom refers to the atoms that are bonded to the atom in question. Knowing the neighbourhood of each atom within a lattice is important for inferring the structure of the lattice within a nanoscale image.

2.8 N-Patch Disk Model

The N-Patch Disk Model is a method of simulating atomic behaviour during molecular self-assembly [3][2]. This model demonstrates the creation and growth of varying forms of planar lattices by modelling atoms as disks. These disks are given a uniform number of areas in which they can connect to other disks. By varying the number of connective areas and their sizes, the disks can form several different kinds of planar lattices, as shown in Fig. 2.4. Using this technique it is possible to prepare an output set of coordinates and connectivity representing a lattice.

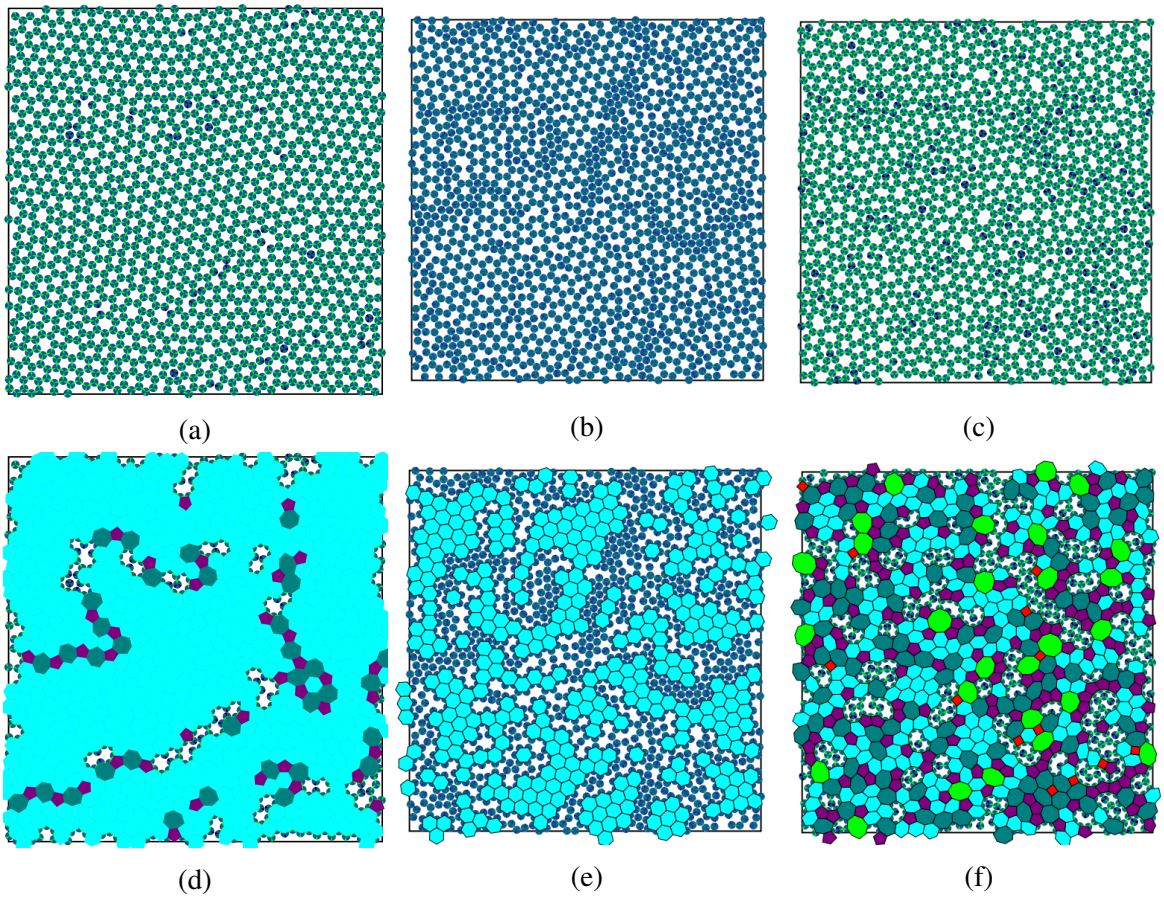


Figure 2.4: This figure shows several outputs for the N-Patch Disk simulation. (a), (b), and (c) show the exact output from the simulations. (d), (e) and (f) show the same output from the above rows but where rings within the lattice are indicated and color coded according to number of disks in the rings.

2.9 Image Analysis on the Atomic Scale

Advances in both atomic scale imaging and computer vision have made this problem only recently capable of being relevant and addressed. As this problem has only been recently addressed, investigation into the issue has been limited.

An existing method by Ophus et al. [56] for inferring the structure of a planar lattice within an electron microscopy image begins by first identifying all the minima within the image. Once the minima are identified the Voronoi tessellation of the minima is computed [57]. A weighted version of the Voronoi tessellation is then reapplied to the minima but excludes any minima within a boundary cell, a cell that is open and extends beyond the frame of image. This version of the Voronoi algorithm weighs intermediate tessellations based on length of the edges and ensures that all edges are similar in length. The method used by Ophus et al has the benefit of being able to quickly compute the atomic locations for a large number of atoms, as is shown in Fig. 2.5. While this method is sufficient for analysing grain boundaries, it excludes atoms bordering the edge of the image which can provide more information about the system in general.

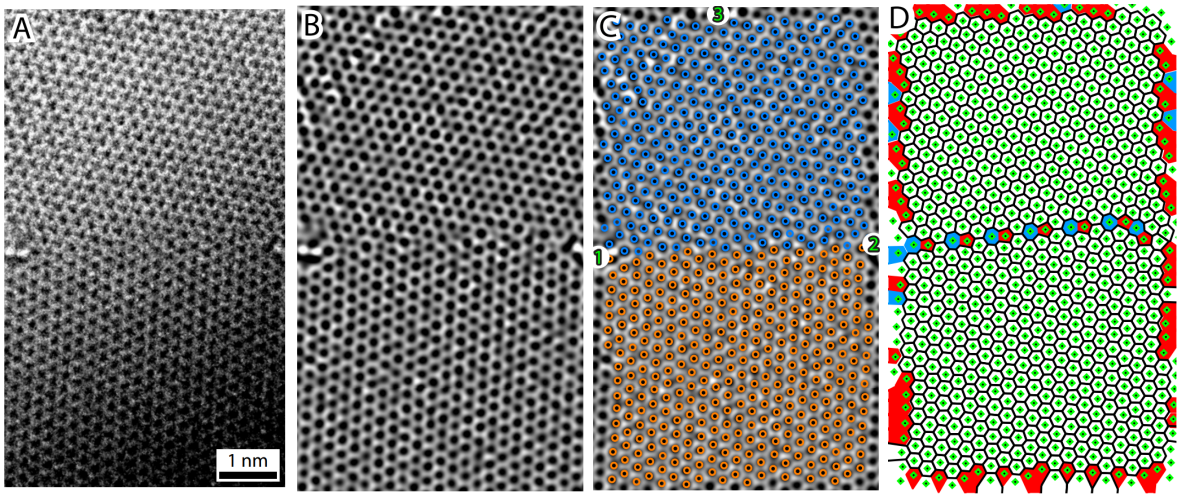


Figure 2.5: A visual representation of the steps involved in Ophus et al.'s [56] algorithm. (a) is the original image of the graphene grain boundary. (b) is the same image as in (a) but after applying image processing methods. (c) contains the graphene ring centers and which grain each ring belongs to. (d) is a graphical representation of the image which includes the graphene ring centers and highlight rings which are part of any defects.

Chapter 3

Method

In this chapter, an algorithm for analysing and inferring structure from a Transmission Electron Microscopy (TEM) image is introduced. We first describe a method of determining an initial set of atomic locations from a nanoscale image and of detecting and removing potential false detections from the initial set of atomic locations. We then describe how atomic bonds, and neighbourhoods, are inferred from the locations of detected atoms and how to find rings within the generated neighbourhoods. Towards the end of this chapter, a forward model is presented in order to generate images for the purpose of evaluating our algorithm.

3.1 Atomic Detection and False Detection Removal

In order to accurately detect atoms within a TEM image, we use a process that includes three steps. The first step is to perform initial segmentation of the image. The second step is to analyse the detected regions and identify preliminary atom locations. The final step removes any erroneous detections found within the crude detections. During the second step several kinds of errors can occur. There can be errors where there are several detections in a large area that are part of a single atom, which are called fractured detections. There can be detected regions which are not part of any atoms, which are false positives. There can be detections where two atoms are encapsulated within a single detected region, which are called merged detections.

The third and final step in the process rectifies all three of these errors by constraining the detections and refining them into more accurate approximations.

3.1.1 Segmentation

Upon initial inspection of a TEM image one might believe that the local maxima within the image could be used to locate atoms. However, as shown in Fig. 3.1, using the local maxima is not enough to accurately locate atoms. A more complex method, called segmenation, is needed to retrieve the atomic locations.

A segmentation of a nanoscale image is the separation of the image into regions that represent whether a pixel belongs to an atom or the background. The image is segmented in this fashion in order to later identify atomic locations. Segmentation of the image is achieved by smoothing the image, then creating a binary mask of the image and flagging each pixel that belongs to an atom. In this subsection we present two methods used for the segmentation process. The first is an intensity based method which is loosely based on an adaptive thresholding method implemented by Bradski [58]. The second method uses the second derivative of an image calculated using finite differencing methods. Fig. 3.2 shows the output of both these methods along with the original image for comparison.

Let I denote an input TEM image, where $I(x, y)$ is a pixel within the image where $x \in [0..w]$ and $y \in [0..h]$ where w and h are the width and height, respectively. We convolve I with a Gaussian filter $g(\mu, \sigma)$ that smooths the image. This is in order to remove noise from the image and ensure that the input image is as continuous as possible. For clarity, we will let I_g be a shorthand notation for $I * g(\mu, \sigma)$.

Intensity based algorithm

The adaptive threshold method segments a greyscale image on a per pixel basis based on the intensity value of the pixel and its surrounding 8-connected neighbourhood. The image is transformed into a binary image by determining if the mean of the pixel and its neighbourhood

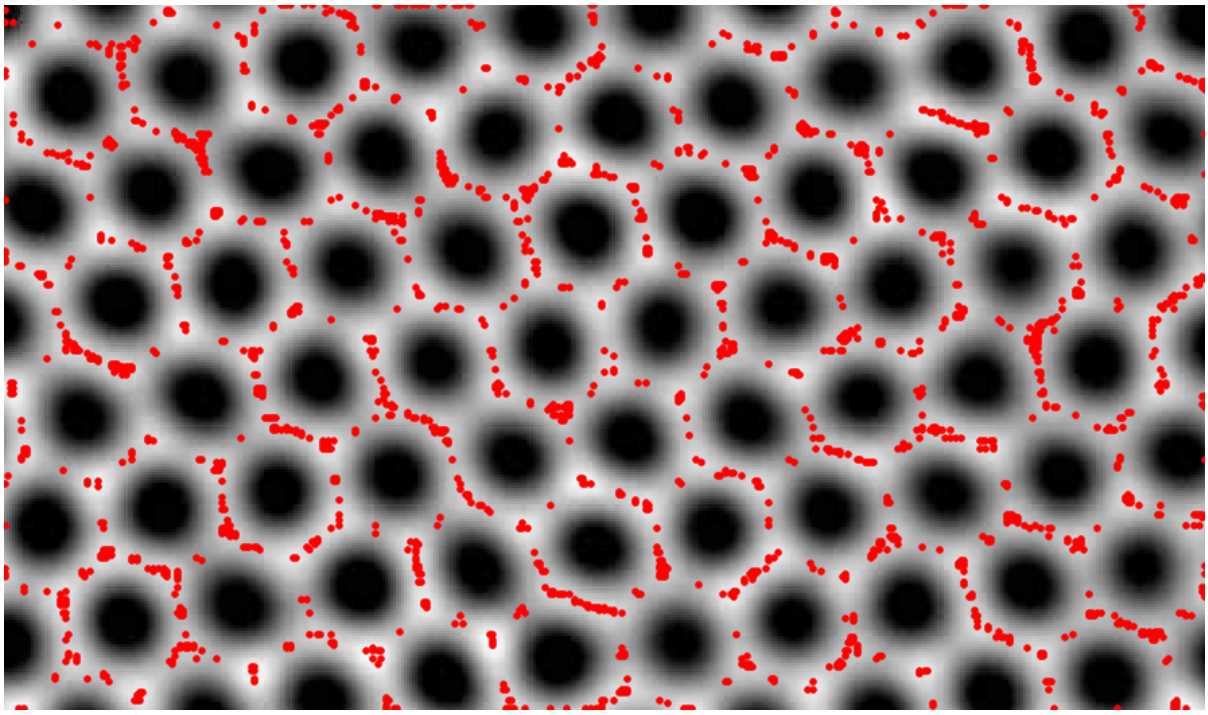


Figure 3.1: Displayed above is a TEM image [23] annotated with local maxima in red. The image is upscaled using a nearest neighbour method to more accurately portray pixel values. Here we can see that attempting to use the local maxima to find atoms is insufficient as there are too many local maxima in within the image.

is greater than the value of the pixel itself. This can be shown using the equation

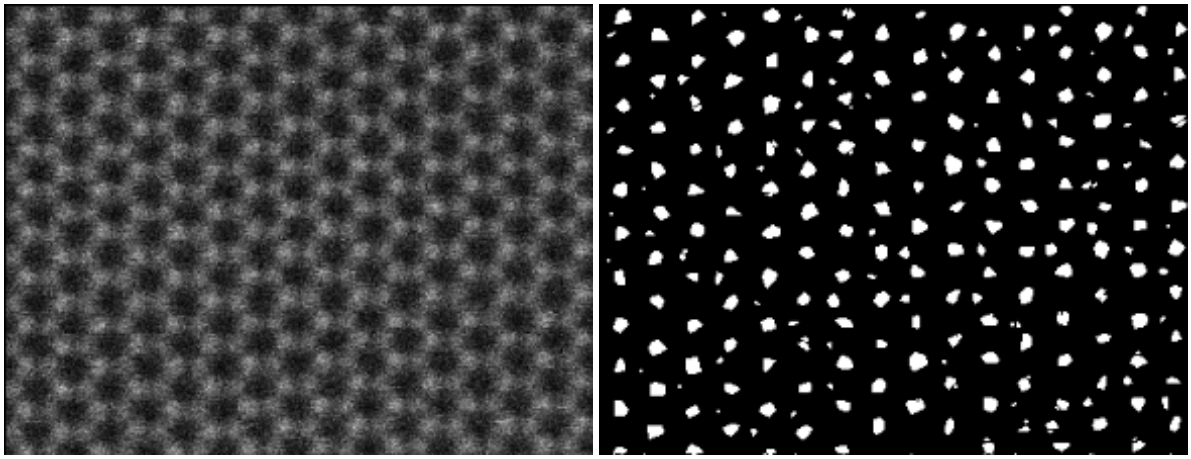
$$S(x,y) = \begin{cases} 1 & I_g(x,y) > T(x,y) \\ 0 & otherwise \end{cases} \tag{3.1}$$

where

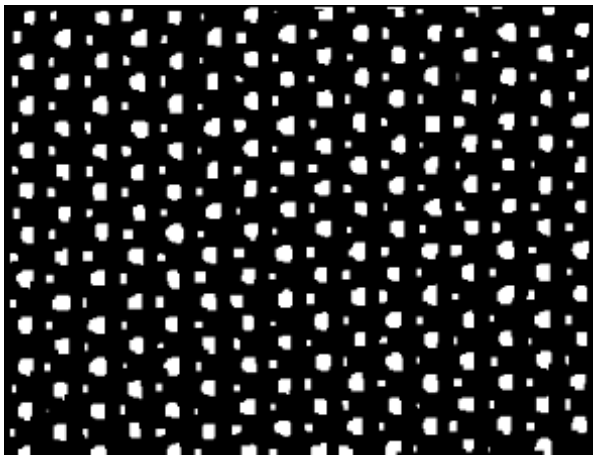
$$T(x,y) = \frac{1}{9} \sum_{j=y-1}^{y+1} \sum_{i=x-1}^{x+1} I_g(i,j) \tag{3.2}$$

$T(x,y)$ can be simply stated to be the mean of the 8-connected neighbourhood of $I_g(x,y)$.

The resulting binary image is taken to be the segmentation image for the later parts of the algorithm. This method assumes that any point on a 3D surface with positive curvature should be equal to or higher than the mean of its surrounding points. However with many TEM images, it was found that this is not the case. Noise presents a difficult obstacle for this



(a) Experimental TEM image for analysis (b) Intensity Based Segmentation



(c) Second Derivative Based Segmentation

Figure 3.2: Sample segmentations of a TEM image. (a) is the TEM image [24] that is being segmented. (b) Segments the image using an intensity based method and (c) segments the image using the second order derivative of the image. The TEM image is of hexagonal Boron Nitride (h-BN). One can see the differences between the different kinds of atoms within the lattice in the second derivative based segmentation. Segmentation is the first step in the atomic detection process.

algorithm and in order for the algorithm to function an extreme amount of smoothing is needed. The advantage of this method is that the segmentation image is simple to compute with time complexity of $O(N * K)$ where N is the number of pixels in the image and K is the size of the kernel. Since this method only uses the 8 connected neighbourhood for a kernel, K is fixed to a value of 9. This means that this algorithm has a linear time complexity.

Second Derivative Based Algorithm

The second derivative based method also segments a greyscale image on a per pixel basis. However, segmentation is based on the computed discrete second derivative of the target pixel's intensity instead of the original intensity value. The second derivative of the smoothed I_g is taken along the direction of I_g's axes and diagonals, $\frac{\partial^2(I_g)}{\partial x^2}$, $\frac{\partial^2(I_g)}{\partial y^2}$, ... The first derivative images are calculated by convolving the image with a derivative kernel, as explained in Section 2.2. In order to calculate the second derivative, the image is convolved a second time with the same kernel. In order to compute for the x axis ($\frac{\partial^2(I_g)}{\partial x^2}$), the kernel

$$D_x = \begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \tag{3.3}$$

would be used and to compute the kernel for the xy diagonal ($\frac{\partial^2(I_g)}{\partial xy}$), the kernel

$$D_{xy} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} \tag{3.4}$$

would be used. We compute second derivative images for each direction, including diagonals, and construct binary images from them.

These binary images indictate any areas of the derivative images that have negative second derivatives, i.e. $B_{xx} = \frac{\partial^2 I_g}{\partial x^2} < 0$. Per pixel information from the binary images are then aggregated using the $\wedge$ operator, $B_{xx} \wedge B_{yy} \wedge ...$, to keep only the areas that have negative second derivatives across all directions. Any remaining noise in the resultant binary image is removed by means of a morphological opening, $\circ$, of the segments in the image using a small 3x3 or 5x5 structuring element, E. The resultant binary image S is taken to be the segmentation of pixels that belong to atoms from background pixels, where $S(x, y) = 1$ are

pixels that belong to atoms and $S(x, y) = 0$ are pixels that belong to the background. The entire equation for the process is as follows:

$$S = \left(B_{xx} \wedge B_{yy} \wedge B_{xy} \wedge B_{-xy} \right) \circ E \tag{3.5}$$

where

$$B_{nm}(x, y) = \begin{cases} 1 & \frac{\partial^2(I_g)}{\partial n \partial m}(x, y) < 0 \\ 0 & \frac{\partial^2(I_g)}{\partial n \partial m}(x, y) \geq 0 \end{cases} \tag{3.6}$$

where n and m can represent either the y or x axis.

3.1.2 Initial Atomic Locations

In order to locate a set of potential atoms, $\mathcal{A}$, from the segmentation image S, we need to first retrieve the set of contours, $\mathcal{C}$, from S. The contours within $\mathcal{C}$ define a set of closed borders between the background and the atomic regions. We take the centre of the potential atoms to be the centroids of the contours. These centroids are calculated using contour moments. While calculating the moments may be computationally expensive, they are reused in the stage of this algorithm in order to prune incorrect detections within $\mathcal{A}$.

The set of contours within the segmentation image are retrieved using Suzuki and Abe's method [59]. We define the set of retrieved contours as $\mathcal{C} = \{c_i | i = 1, ..., N\}$ where c_i is an individiual contour defined as $c_i = \{(x_j, y_j) \, | j = 1, ..., J \wedge x_j \in [0..w] \wedge y_j \in [0..h] \wedge x_0 = x_J \wedge y_0 = y_J\}$ where N is the number of contours in the retrieved set and J is the length of the ith contour in the set. Moments $\mathcal{M}^i$ are then calculated from the ith contours in $\mathcal{C}$, using Green's Theorem [51]. The centroids are then calculated from the spatial moments of the contours such that $\mathcal{A} = \{\mathbf{a}_i | \mathbf{a}_i = \left(\frac{\mathcal{M}^i_{01}}{\mathcal{M}^i_{00}}, \frac{\mathcal{M}^i_{10}}{\mathcal{M}^i_{00}} \right) \wedge i = 1, ..., N\}$ where $\mathcal{A}$ is the set of points that represents the set of centres of possible atoms.

3.1.3 Merging Fractured Detections

Fractured detections are a group erroneous detections within the set of potential atoms $\mathcal{A}$ that should be represented as a single atom. These fractured detections can arise as a consequence of incorrect parameters being chosen during the initial segmentation. In order to fix fractured detections, we locate all pairs of detections where the distance between centroids is smaller than $\frac{d}{2}$ where d is a scalar value representing the distance between atomic centres. Once all pairs have been found, we then merge the pairs using the convex hull [53] of the pairs' contours. If a detection is fractured into more than two separate detections, they are merged as a consequence of the merging process.

It is assumed that the distance, d, between any two adjacent atoms within the lattice is relatively similar for all adjacent pairs in the lattice. It is also assumed that the size of d, in pixels, is known *a priori* and that $\frac{d}{2}$ is the approximate radius of an atom in the image. The method begins by finding all pairs of contours where the distance between their centroids is smaller than $\frac{d}{2}$. We define this set of pairs, the set of fractured detections and incorrect detections, as $\mathcal{V} = \{v_i | v_i = (\mathbf{a}_i, \mathbf{a}_j) \wedge \mathbf{a}_i, \mathbf{a}_j \in A \wedge |\mathbf{a}_i - \mathbf{a}_j| < \frac{d}{2} \wedge i \neq j\}$. If $f(\mathbf{x})$ is a function that retreives a contour from $\mathcal{A}$ using its centroid as input, we can then define a fractured point within $\mathcal{V}$ to be any point that satisfies the equation

$$\frac{f(v_{i_0})}{f(v_{i_0}) + f(v_{i_1})} < T \wedge \frac{f(v_{i_1})}{f(v_{i_0}) + f(v_{i_1})} < T\,, \tag{3.7}$$

where $T \in [0, 1]$ and is a scalar value that represents the threshold that controls the maximum percentage of the total area between the pair of contours that one of the contours may possess. If the equation evaluates to false we assume that the detection with the smaller contour area is likely to be noise and is removed. Though T can exist within the interval $[0, 1]$, T should not be set within several subintervals. One should not use the $[0, 0.5)$ interval as, due to the nature of the equation, it will ensure that the equation will evaluate to false. Higher values of T, on the scale $[0.8, 1]$, will cause the evaluation of equation to present a bias towards true. In

these cases, with higher values of T, it is more likely that detections arising from noise will be preserved. Upon identification of a fractured detection, the convex hull [53] of both contours is drawn onto S. After all pairs of detections have been evaluated we recompute $\mathcal{A}$ from S. The resulting $\mathcal{A}$ from the computation is a set of potential atoms that is devoid of fractured detections. Fig 3.3 illustrates correct, fractured, and merged detections.

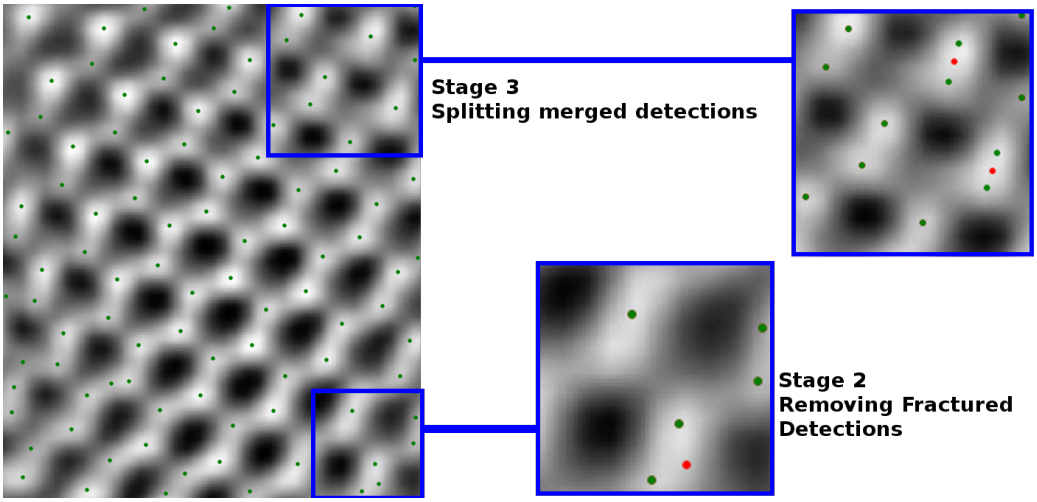


Figure 3.3: Example of a TEM image of graphene being analyzed by the system. The image on the left shows the initial detections from the atomic detection system [60]. The bottom right image shows the output from the second stage of the system where points that are red have been removed from the previous set of detections. The top right image shows the output from the third stage of the system.

3.1.4 Splitting Merged Detections

Merged detections are considered to be any contours within $\mathcal{C}$ that contain two or more real atoms. Though it is impossible to know for sure whether or not a contour contains more than one atom, it is possible to use statistical analysis and several criteria in order to determine the likelihood of a contour being a merged atom.

In our method, three criteria are used for the purpose of determining merged detections. The first criterion is the area of the contour. This can be retreived from the zeroeth moment of a contour, $\mathcal{M}_{00}$. This criterion is employed because we expect merged detections to be larger

than correctly identified detections. It is assumed that all atoms are of similar size with a small amount of variance between them. If there is a possible atom that is significantly larger than others within the image we expect the atom to be incorrectly detected and possibly a merged detection.

The eccentricity, ε, of the contour is used as the second criterion, where ε is calculated using the central moments, μ of the contour, $\varepsilon = 1 - \left(\frac{\mu_{20}+\mu_{02}-\sqrt{4\mu_{11}^2+(\mu_{20}+\mu_{02})^2}}{\mu_{20}+\mu_{02}+\sqrt{4\mu_{11}^2+(\mu_{20}+\mu_{02})^2}}\right)^2$. The central moments of a contour are moments that have been centred with respect to the contour's centroids and are calculated from the contour moments using the equation,

$$\mu_{pq} = \int\int \left(x - \frac{\mathcal{M}_{01}^i}{\mathcal{M}_{00}^i}\right)^p \left(y - \frac{\mathcal{M}_{10}^i}{\mathcal{M}_{00}^i}\right)^q I(x,y)\, dxdy \tag{3.8}$$

ε can be used to show the elongation of a contour and it is assumed that the more elongated a contour, the more likely it is to be a merged detection. Our last criteria was the Euclidean distance between the highest intensity pixel within the contour and the centroid of the contour. We expect the centroid of the contour to be near the maximum intensity pixel within the contour due to the nature of nanoscale imagery. Within nanoscale imagery, the centre of an atom is expected to have the highest intensity value of the entire atom. If the calculated value from the contour has a positive difference of more than one standard deviation from each of the means of the values detailed earlier, we assume that the contour is a merged detection.

If a merged detection has been found, we fit an ellipse to the contour and split the contour in half by drawing a line on S along the fitted ellipse's minor axis, in order to split the contour into two separate regions. These seperate regions will represent the likely contours of the atoms existing within the image. Similar to step 2, we then recompute $\mathcal{A}$ from S to get our final set of atoms within the image. Fig. 3.4 shows a comparison between the output of our method and the local maxima used as a baseline comparison.

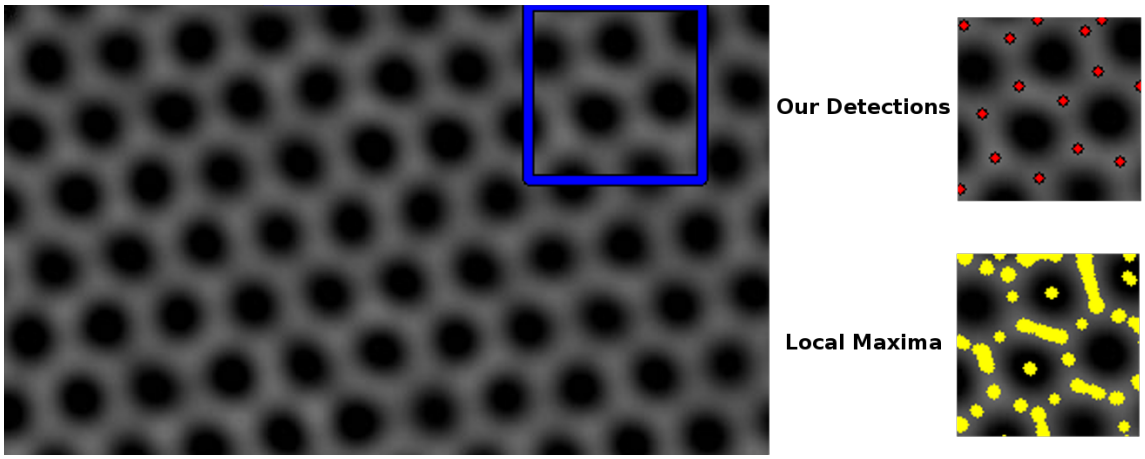


Figure 3.4: Comparing Local Maxima to the system's initial detections. On the left is an experimentally produced image of graphene (non synthetic) [23] overlayed with a square representing a region of interest. The bottom right image is the region of interest overlayed with the local maxima in yellow and the top right is the region of interest overlayed with the output set of points from our system.

3.2 Generating Neighbourhoods and Finding Rings

3.2.1 Generating Neighbourhoods

The neighbourhood of an atom defines the bonds between a detection and other detections in the set. The bonds within each atom's neighbourhood are important for inferring the structure of the molecule depicted within the input nanoscale image. In order to generate an initial neighbourhood of atoms, our algorithm first begins by selecting a single atom, and finding the closest atom to our selected atom. These will be called the target atom and the closest neighbour respectively. The closest neighbour is retrieved by calculating the euclidean distance between the target atom and all other atoms in the set of atomic coordinates and finding the atom with the smallest distance.

If r is a scalar value representing the euclidean distance between the target atom and its closest neighbour, then all atoms within $1.5r$ are considered to be possible nieghbours of the target atom for the initial neighbourhood. It was found that within TEM images, neighbouring atoms often do not occupy the exact same radius as other neighbours, using $1.5r$ as the maximal distance for potential neighbours, instead of r, allows our the neighbourhood generation

Algorithm 1: An Algorithm to generate a set of neighbourhoods for a set of atoms.

Input : An ordered set of 2 dimensional points A
Output: An ordered set of neighbourhoods N with indices matching A

Function `GetInitialNeighbourhood(`A`)`**:**
  $N \leftarrow \varnothing$
  foreach $\vec{a_i} \in A$ **do**
    $r \leftarrow \infty$
    foreach $\vec{a_j} \in A \mid i \neq j$ **do**
      $d \leftarrow \|\vec{a_i} - \vec{a_j}\|$
      if $d < r$ **then**
        $r \leftarrow d$
      end
      $N \leftarrow N \cup \{N_i \mid N_i = \{(\vec{a_i}, \vec{a_j}) \mid \|\vec{a_i} - \vec{a_j}\| < 1.5r\}\}$
    end
  end
  return N

algorithm to account for any small errors in the locations of the atoms. Algorithm 1 shows the pseudocode for neighbourhood generation.

To account for any incorrect neighbours within the initial neighbourhood, we eliminate possible neighbours based on the distance between the target atom and its neighbours. To eliminate possible incorrect neighbours based on distance the mean and the standard deviation of the possible neighbours for all possible target atoms is calculated. Using the mean and standard deviation, we check to see if the distance from each target atom to its possible neighbours are within three standard deviations of the mean of all distances between target atoms and their neighbours. If they are not within three standard deviations they are then removed from the set of possible neighbours. This prevents the possibility of a neighbourhood being generated for an atom that is unbonded to any others.

To ensure that there are no incorrect neighbours that happen to exist within the remaining possible neighbours, we evalute the angles formed between possible neighbours using the target atom. Since we are working with a 2-dimensional image, a connecting vector between a target atom and a neighbour will have two angles associated with it. One on the clockwise side of the connecting line between a target atom and a possible neighbour and one on the counter

Algorithm 2: The Algorithm to calculate the angles between atomic neighbours for all atoms, A, and neighbourhoods, N.

Input : An ordered set of 2D points A and associated neighbourhoods N
Output: A set of angles representing the angles between two neighbours in the neighbourhoods of N

Function `GetInnerAngles(`A, N`)`**:**
  angles $\leftarrow \varnothing$
  foreach $N_i \in N$ **do**
    $P \leftarrow \varnothing$
    foreach $(\vec{a_i}, \vec{a_j}) \in N_i$ **do**
      foreach $(\vec{a_i}, \vec{a_k}) \in N_i \mid \vec{a_j} \neq \vec{a_k}$ **do**
        $ang \leftarrow$ `GetAngle` $(\vec{a_i} - \vec{a_j}, \vec{a_i} - \vec{a_k})$
        $P \leftarrow P \cup P_i \mid P_i = \{ang, \vec{a_i}, \vec{a_j}, \vec{a_k}\}$
      end
    end
    $sn, n \leftarrow N_{i_0}$ `// sn, n has the form` $(\vec{a_i}, \vec{a_j})$
    while *True* **do**
      `// This loop reorders the angles for later usage`
      $Q \leftarrow \{Q_i \mid Q_i \in P, P_{i_3} = n_1\}$ `//` Q_i `has the form` $\{ang, \vec{a_i}, \vec{a_j}, \vec{a_k}\}$
      $t \leftarrow \text{argmin}_x f(x) = \{x | Q_{x_0}\}$
      angles $\leftarrow$ angles $\cup Q_t$
      $n \leftarrow (Q_{t_1}, Q_{t_2})$
      if $n = sn$ **then**
        break
      end
    end
  end
  return angles
Function `GetAngle(`l, s`)`**:**
  ang $\leftarrow \arctan(l_y/l_x) - \arctan(s_y/s_x)$
  if $ang < 0$ **then**
    return $2\pi +$ ang
  end
  return ang

Algorithm 3: The Algorithm to remove incorrect neighbours from a set of neighbourhoods.

Input : An ordered set of 2D points A and associated neighbourhoods N
Output: N with all incorrect neighbours removed

Function `Prune(`A, N`)`**:**
  angles $\leftarrow$ `GetInnerAngles(`A, N`)`
  $\mu \leftarrow \frac{\sum_{v \in angles} v_{i_0}}{|angles|}$
  $\sigma \leftarrow \sqrt{\frac{\sum_{v \in angles}(v_0 - \mu)^2}{|angles|}}$
  edgesToRemove $\leftarrow \varnothing$
  foreach $v \in$ *angles* **do**
    if $v_0 < 2\sigma - \mu \wedge \mu + 2\sigma < v_0$ **then**
      if $v_2 \notin$ *edgesToRemove* **then**
        edgesToRemove $\leftarrow$ edgesToRemove $\cup v_2$
      else
        remove (v_1, v_2) from N
      end
      if $v_3 \notin$ *edgesToRemove* **then**
        edgesToRemove $\leftarrow$ edgesToRemove $\cup v_3$
      else
        remove (v_1, v_3) from N
      end
    end
  end
  return N

clockwise side. As in Algorithm 2, the mean and standard deviation of the angles is calculated. We define an outlier angle to be three standard deviations from the mean. All connecting vectors which are found to have both of their associated angles, clockwise and counter clockwise sides, considered to be outliers are removed from the set of possible neighbours, as shown in Algorithm 3. What is left is considered to be the final set of neighbours for the lattice.

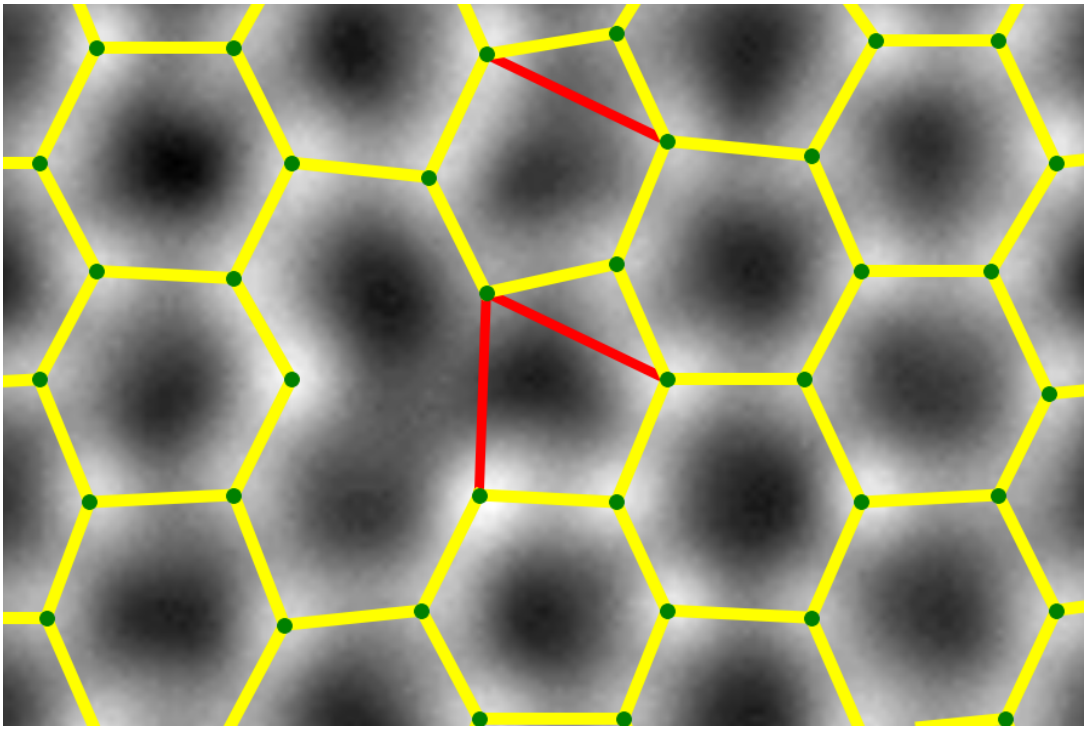


Figure 3.5: Sample output from the neighbourhood generation process for a TEM images [48]. In green, the detected atoms. The yellow lines connecting atoms represent the neightbours. The red lines are the connections from the initial neighbours that have been removed.

Fig. 3.5 shows sample output for the neighbourhood generation algorithm. The final set of neighbours are represented using yellow lines. As stated previously, neighbours are removed based on the angles between them. The neighbours that have been removed are outlined in red. Upon inspection, one can ascertain that most of the angles, in Fig. 3.5, between the final set of connections are approximately 120 (degrees) and that the neighbours which have been removed, have angles that are either much larger or smaller than the angles formed by neighbours. This demonstrates the principles on which our method of eliminting possible neighbours is based.

3.2.2 Finding Rings

A ring is a connecting path of three or more atoms that begins and ends with the same atom. Identifying rings allows a higher level view of the nanoscale image that can highlight defects within the lattice. Rings are found using a modified version of the floodfill algorithm to identify atomic detections that belong to the ring we are currently attempting to find. This is followed by a validation step that verifies that the atomic detections that have been found create a valid ring.

Algorithm 4: A modified version of the floodfill algorithm used to find atoms belonging to rings.

Input : I, A binary image with lines drawn to represent neighbours. A, all atomic locations. $\vec{x_0}$, a starting point.
Output: vertsFound, the verteces found that are part of the ring.

```
Function FindInitialVertices(I, A, x⃗₀):
    vertsFound ← ∅
    pointsToCheck ← ∅
    pointsToCheck.push(x⃗₀)
    while |pointsToCheck| ≠ 0 do
        x⃗ ← pointsToCheck.pop()
        if I(x⃗) = 1 then
            skip
        else
            I(x⃗) ← 1
            foreach p in 8-connected neighbourhood of x⃗ do
                if p ∈ A then
                    vertsFound.push(p) break
                end
            end
            foreach p in 3-connected neighbourhood of x⃗ do
                pointsToCheck.push(p)
            end
        end
    end
    return vertsFound
```

The ring finding algorithm begins by creating a binary image, the same size as the input TEM image, and setting all pixel values to zero. Lines are then drawn on the binary image that match the vectors connecting atoms to their neighbours.

After the preparation of the binary image, the ring finding algorithm then applies the modified floodfill algorithm to the image. The ring finding algorithm first scans the binary image for any zero pixels. When a zero pixel is found the location of the pixel is given as input to the modified floodfill algorithm. The input pixel is then used as the first pixel to be inspected, as is shown in Algorithm 4. The floodfill algorithm first checks to see if the pixel being inspected has been set to one. If a pixel has been set to one, the pixel has either already been inspected or is part of a line separating rings. If the pixel being inspected is set to one, we then end this iteration of the algorithm. If the inspected pixel is zero, we then check the 8-connected neighbourhood of pixels about the inspected pixel and compare their locations to the set of atomic detections. If any detections match the locations in the 8-connected neighbourhood, we record that detection within a set. We then set value of the inspected pixel to one and all pixels in its 4-connected neighbourhood are added to the set of pixels to be inspected. The floodfill portion of this algorithm then repeats until zero pixels can no longer be found.

Algorithm 5: Reorders a set of vertices found by the floodfill algorithm to determine whether or not a valid ring has been found.

Input : N, the set of Neighbours. V, A set of atoms representing a potential ring
Output: path, An ordered set of atoms representing a ring or an empty set in the case of an invalid potential ring.

```
1  Function ReorderVertices(N, V):
2      path ← {v_0}
3      remain ← {v_i | v_i ∈ V, v_i ≠ v_0}
4      while |remain| ≠ 0 do
5          if last atom in path has a neighbour in remain then
6              Add neighbour to path
7              Remove neighbour from remain
8          else
9              return ∅
10         end
11     end
12     if last atom in path is not a neighbour of the first atom then
13         return ∅
14     end
15     return path
```

Once there are no longer any zero pixels left to be inspected, the algorithm then checks to

see if the set of recorded detections form a ring. Algorithm 5 shows the pseudocode for the the ring validation and reordering code. In the code, a detection is removed from the input set of detections and an attempt is made to match any neighbours of the removed detection to other atoms within the recorded detections. If a match is made we remove the match and look for its neighbours in the recorded detections. This process is continued until there are no longer any recorded detections in the set. We then ensure that the last detection removed from the set matches is a neighbour of the first detection removed. If all matches are found, the set of detections is flagged as a cycle. This method of verifing the validity of this set as a ring also has the advantage of ordering the detections for display.

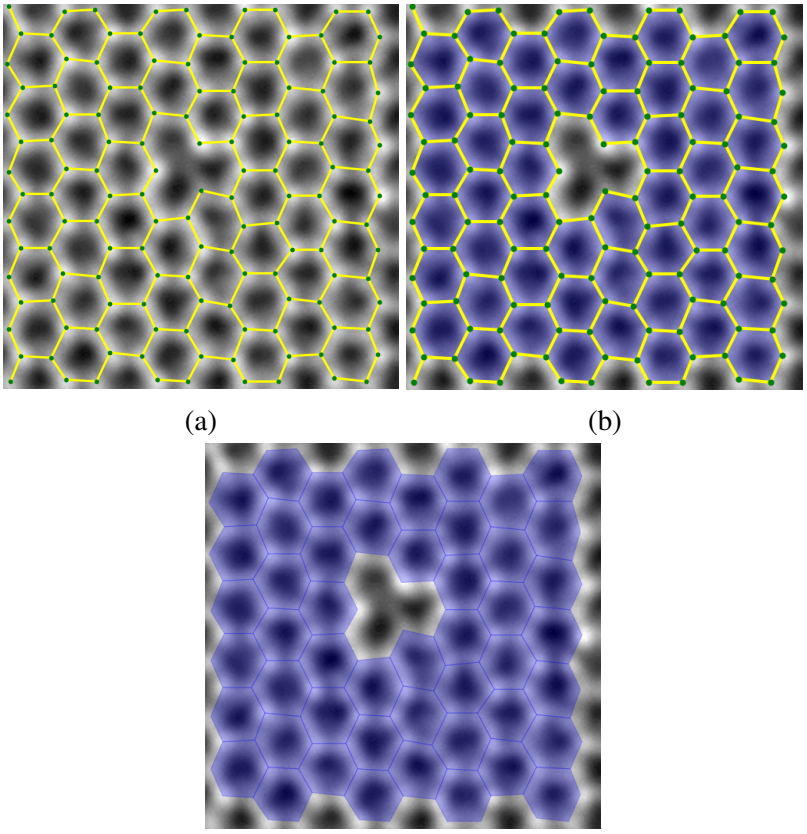


Figure 3.6: This figure shows the input and output for the modified floodfill algorithm used to find rings using the neighbourhood of each atom. (a) visualises the input for the system overlayed on the TEM image [48], and (b) visualises the associated output. Only rings with 6 atoms are coloured in this image. (c) is the visualised output from our algorithm but with the neighbourhood and atoms removed for clarity.

The algorithm then checks for zero pixels and iterates continuing to apply the floodfill algorithm until there are no remaining zero pixels left. What is now produced by the algorithm is all rings found within the lattice. Fig. 3.6 shows the output rings of our system for the given input. There we see that each ring has been colour coded in order to identify it as a ring. However, since there are no rings with less than 6 atoms all rings acquire the same colour. The large ringless area within the center of the image is detected as a ring though it is not coloured for illustrative purposes as it includes a large number of atoms within itself. Large rings, like in the figure, could be highlighted in order to illustrate possible defects within lattices.

3.3 Parameter Learning

It was found that the output of our method relied heavily upon the parameters used during the segmentation process. A correct or near correct set of parameters would allow the segmentation process to produce a set of results that was as close to ground truth as possible, with little to no split or merged detections. However an incorrect set of parameters would create a binary image where the correct set of points are irretrievable. It is for this reason that it was decided that parameter learning should be used within our method.

The parameter learning process consists of testing a range of possible input parameters for the Gaussian smoothing and the morphological operations. These parameters are the σ and k of the Gaussian smoothing function and the kernel size for the morphological operations. For each test a set of parameters is selected from the range. The set is then used to create a binary image using one of the segmentation processes details earlier in subsection 3.1.1. Contours are then extracted from the binary image and used as input to a cost function.

Two separate cost functions have been tested for parameter learning. The first cost function used the root mean squared error of of the zeroth moment of the contours. As a reminder, the zeroth moment of a contour is the area of the contour. The purpose of this cost function was to ensure the area of all contours were as similar as possible, which would be the expected

appearance for a lattice. We discovered that this measure was insufficient and did not properly learn the correct parameters.

The second cost function used bond lengths generated using the same method that generates our initial neighbourhoods. The error is taken to be the root mean squared of the set of retrieved bond lengths. It was found that this measure not only produced an ideal cost function for producing output results with minimal error but also had the advantage of producing an approximate value for d to be used in merging split detections. By taking d to be the mean of the bond lengths generated where the cost function is at a minima and using the value as the input for merging split detections we are able to add more automation to the atomic detection process.

This cost function however is not without its drawbacks. Our bondlength based cost function uses the assumption that in the given lattice all bondlengths are of similar lengths. The cost function also has a large runtime for each execution of the function, which greatly increases the overall execution time for the algorithm. While this method does work for the given test set there is no guarantee that it will work for all given planar lattices.

3.4 Forward Model

Due to the lack of available experimental nanoscale images, a method of synthesizing TEM images was necessary in order to test our system. In Fig. 3.7, we see a TEM image of graphene 3.7(a), in addition to 3-dimensional, 3.7(b), 3.7(c), and 2-dimensional, 3.7(d) plots of the image. There are several distinct notable features in the plots that are neccessary to duplicate when attempting to synthesize a TEM image: (I) atoms within the lattice do not share identical peak intensities and can vary greatly; (II) as we move further away from the centre of an atom intensity values fall in a non-linear fashion; (III) due to errors within the electron microscopy capture process, atoms can have multiple smaller peaks. Feature (I) can be seen by inspecting Fig. 3.7, particularly in 3.7(a,b,c). In these images you can see that the peaks are not all iden-

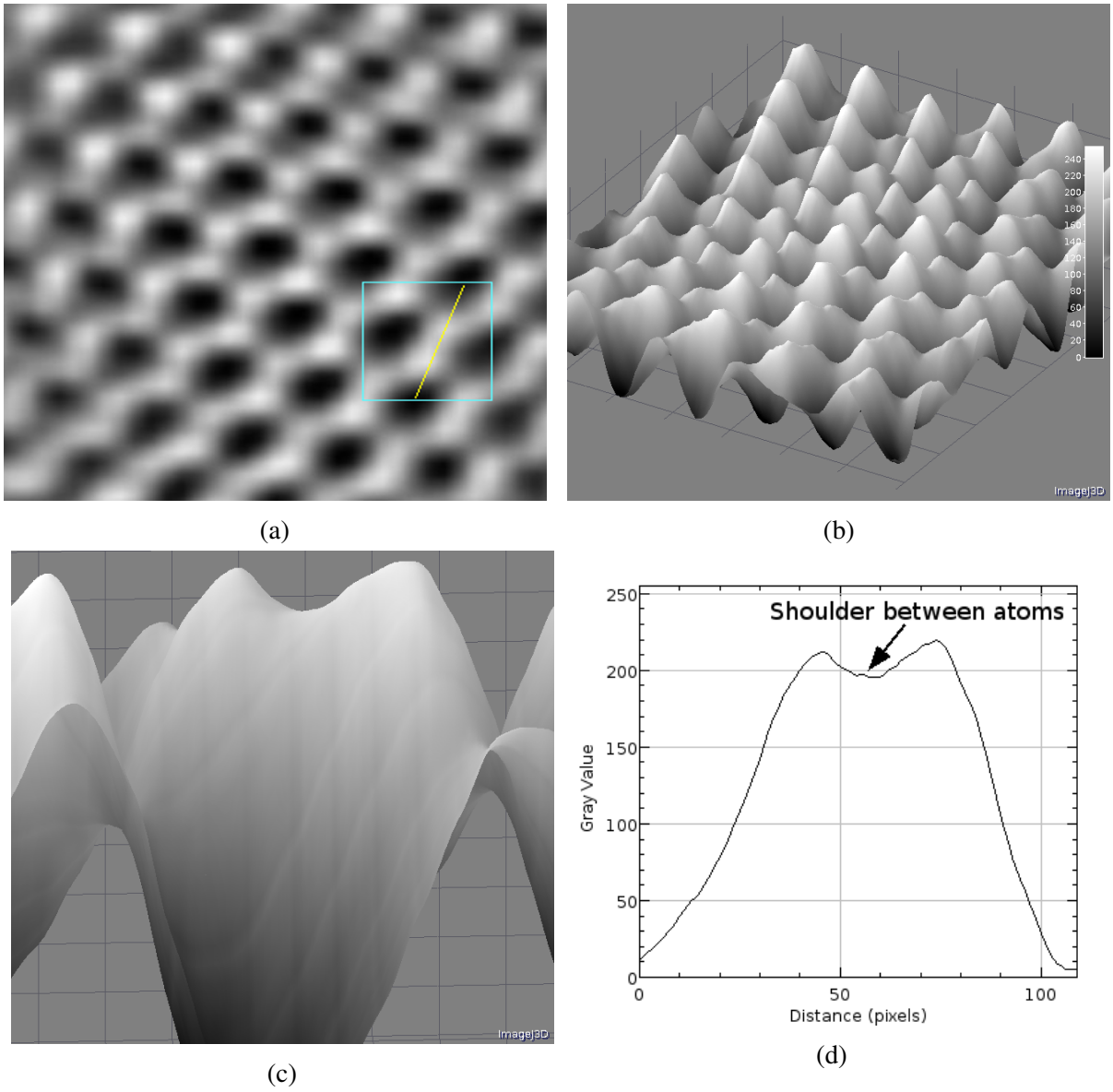


Figure 3.7: Example TEM image along with corresponding 3D plots and Line Plot. (a) is the original experimental image of aberration corrected (AC) Graphene[60] overlaid with the region of interest that is plotted by (c) in cyan and the line plot of (d) in yellow. (b) is the image projected to into a 3D mesh using intensity as the z axis. (c) is the projected image displaying that smaller local maxima exist on an atom.

tical and in 3.7b you can see that peaks have a large range in size and height. We can see that feature (II) exists by inspecting 3.7(d), where we see that the intensity drop off as a function of distance from the atom is non-linear and resembles more closely a Gaussian function. The third feature, (III), is shown in 3.7(c) and 3.7(d) where we see a slight change in the rate of

intensity increase as we go closer to the peak on the right of the images. 3.7(d) shows this more clearly as it indicates where the smaller peak is on the image. In order to accurately synthesize a lattice each of these characteristics need to be replicated.

Our process of synthesizing a TEM image of a lattice has three steps. The first step is a process of inputting a set of 2D or 3D coordinates. The second step is the replication of the coordinates to fit within our image at a given input resolution. The third step is the addition of noise and defects into the lattice prior to drawing.

Coordinates are first input into our synthesizing program by means of an XYZ file. This file contains the 3D coordinates, in angstroms, of a set of points that represent a set of atoms. These atoms can be either free atoms or part of a larger molecule. In the case of the image synthesizer, the coordinates should represent a planar (or near planar) lattice. The assumption is made that the lattice to be synthesized is planar with respect to the Z axis. The Z axis is then removed from the 3D coordinates to convert our 3D lattice to a 2D lattice which allows us easily represent the 2D atomic coordinates within an image.

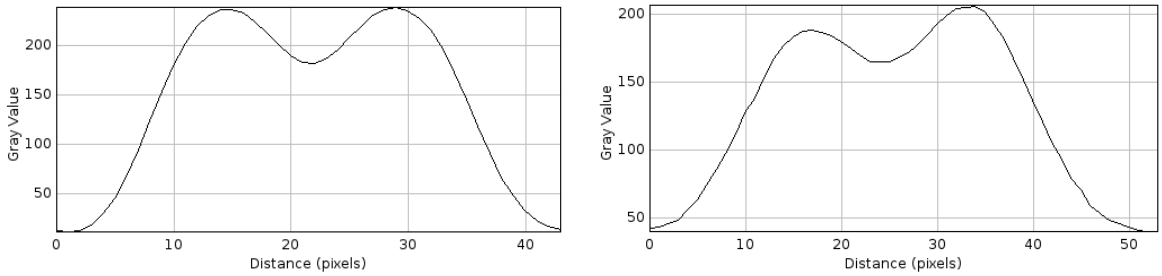
The coordinates are scaled to our desired resolution using a given Angstrom to pixel ratio. If the coordinates represent a lattice we can choose to duplicate the coordinates in other areas of the image in order to fill the image. Defects within the generated lattice, i.e. missing atoms, can be given to the lattice by omitting or shifting coordinates before drawing.

Using the resulting image coordinates we represent the atoms by drawing overlapping concentric filled circles about each of the atomic coordinates. Each of these concentric circles are drawn with an intensity dictated by the function

$$f(r, R) = \frac{1}{R\sqrt{2\pi}} e^{-\frac{r^2}{2R^2}} \tag{3.9}$$

which is loosely based off of the Gaussian function, where r is the distance from the center of the circle and R is the radius of the circle. This will ensure that intensity values fall as the points are further away from the atoms. The height of the peak intensity of the atom can

be controlled by limiting the intensity value of the smallest circle. To ensure the simulation of charge between two atoms, the concentric circles are drawn to be overlapping with their neighbours. To add the smaller peaks detailed earlier, we add noise into the image and then apply smoothing. Our justification for using this function is illustrated in Fig. 3.8. In Fig. 3.8 you can see that when plotting the output of the function, the synthetic data closely resembles the experimental data provided.



(a) Plot of grayscale data from a synthesized image (b) Plot of grayscale data from a captured image

Figure 3.8: Comparing a line plot of real data from a grayscale image to the synthetic data produced by our foreward model. The line is of two bonded atoms in a graphene structure. The two peaks represent the centres of carbon atoms and the minima at the extremes of the plot are the holes in the hexagonal structure of graphene. Even though the peak values for the data vary, the shape of the peak and data are visually similar.

We can also supplement this system with N-Patch Disk Model simulations[3][2], which was incorporated during the testing phase of our system. The N-Patch Disk Model uses Monte Carlo methods to generate 2 dimensional structures that mimic self-assembled lattices. Using this model, we can create lattices with irregular patterns similar to those that exist in experimental TEM images. Fig. 3.9 shows the process of generating one of these lattices and shows the given synthesized image for a 3-Patch Disk Simulation.

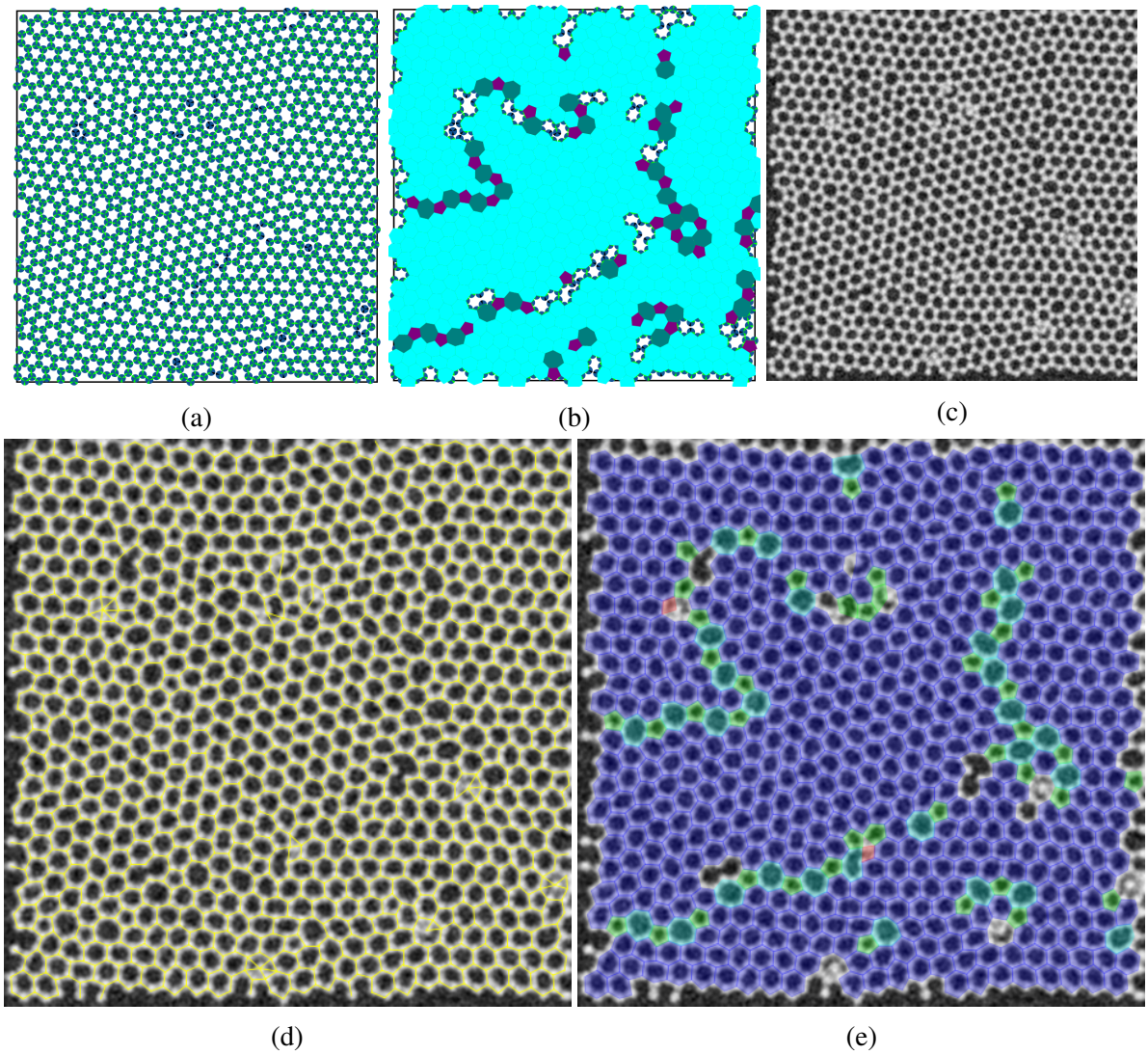


Figure 3.9: Using a 3-Patch Disk model to simulate a hexagonal lattice and compare our image synthesis output with an experimental image. (a) contains an image of a complete 3-Patch Disk simulation of a lattice and (b) contains a color coded version where complete rings are colored. (c) is a synthesized TEM image using the coordinates from (a). (d) shows the bonds output by our system and (e) is an annotated version of the image with color coded rings.

Chapter 4

Experiments

This chapter will discuss implementation details of the proposed system as well as discuss the testing methods used and the results of the tests. Implementation details include the programming languages used as well as the libraries used in order to facilitate the implementation. The constructed testing methods will be described and explained as well as the composition and importance of the selected images used within the tests. Results from the proposed solution and other solutions to the atomic detection problem will be provided and discussed.

4.1 Implementation

Leucippus, our system, is implemented as a plugin for the ImageJ image processing package. The front end of the program was written using Java which interfaces with a Python process running in the background that starts at the initialisation of the Java plugin. The Python process contains all the necessary functions and methods needed to implement the proposed algorithms and perform the atomic detection and the molecular reconstruction algorithms.

The main algorithms of our system were implemented in Python in order to rapidly prototype and develop the system. The algorithms leverage many common mathematical and scientific libraries available for the Python language. These include NumPy [61] and the SciPy [62] software stack supplemented with the OpenCV library [58].

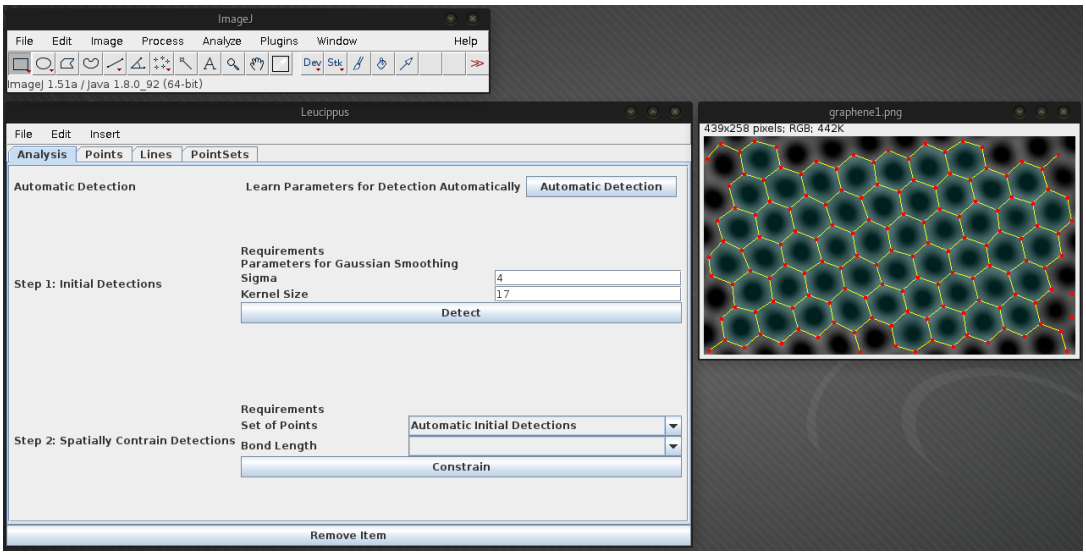


Figure 4.1: The complete implementation of the proposed system as a plugin within ImageJ. At the top left of the image is the main ImageJ toolbar and menu. At the right is the image being analysed overlaid with the output of our proposed system. At the left is the systems main interface.

Due to the implementation of the system as a ImageJ plugin, the possible programming languages that could be used to write the plugin was limited. It was decided that Java would be used because of extensive API and the available documentation for Java within the ImageJ API. The Java front-end for this plugin leverages many Java libraries in order to create a Graphical User Interface (GUI) for the user in order to communicate with the main algorithms written in Python. Results from the main algorithms are then displayed in the main ImageJ GUI. Fig. 4.1 shows an example of the Leucippus GUI along with ImageJ.

4.2 Testing

The detection system was evaluated on several experimentally produced high-resolution TEM images and synthesized high-resolution TEM images. The experimentally produced images were divided into two sets. The first set is composed of planar graphene lattices which is comprised of carbon atoms. The images of graphene have varying amounts of defects within the lattice and the images themselves varied from clear to noisy. Since graphene is composed of

carbon atoms each atom within the lattice should have similar properties allowing us to acquire a baseline performance of the system. The second set is composed of planar hexagonal boron nitride (h-BN). Like the images of graphene, the h-BN have varying amounts of defects and noise within them. The second set, the h-BN images, was designed to determine if the system could recognise both atoms within the lattice as atoms. Ground truth data was produced by hand for each image and is used as a basis of comparison for the performance of the system. In order to compare the results of Ophus et al.'s [56] method with our own an implementation of the method was created in Python so it could be integrated with the testing system. While the results from this method may not be the exact same as those presented in [56] they should still provide similar results to the original implementation.

The synthesized TEM images are created using our forward model, which was detailed in Section 3.4. Using the forward model we can create different sorts of lattices with varying conditions within them. The test images can be generated to have several kinds of defects and variations. These variations can include missing atoms, noise and atoms with variable intensities. Since we are generating the lattice programmatically we can easily output the ground truth for each output image. This allows us to quickly and efficiently create tests to evaluate the performance of our system.

The performance of each atomic detection method was evaluated using three metrics, the precision, the recall, and the F_1 score. The precision of a method's results is the ratio of the number of correct detections divided by the total number of detections by the system. The recall of a method's results is the ratio of correct detections identified by the method divided by the total number of atoms within the image. Both the precision and recall give ranges from zero to one, where a value of one means a method is performing well and a value of zero means the method is failing entirely. If both the precision and recall of a method has a value of one, that means that the method has correctly identified all the correct detections while not reporting any incorrect ones. The F_1 score combines the precision and recall in a manner that represents the performance of the system as a single value. For most tables we use the F_1 score as it can

compare the results of each method in a meaningful way.

4.3 Results

Images within the performance tables are separated into two sets. The first set is a group of experimental images that have been collected experimentally. Each experimental image is listed within the tables demonstrating the results. The second set is the performance of the system on a large set of synthesized images, —50 images per image type. These values are averaged to condense the results into one table.

Table 4.1 shows the performance of the raw detections of the system and of the local maxima when given different input images and in Fig. 3.4 we see example output of both the local maxima and our system within a region. We use the local maxima in order to provide a baseline comparison for all methods being used. Here we see that our system has a relatively high performance values for most input images. The table also shows that using local maxima tends to outperform our system in recall. This is due to the simple fact that the centre of an atom should coincide with a local maxima. However we also see that the precision of the local maxima is very low and shows that very few of the local maxima found within the image belong to atoms. This outlines the advantage of using the F_1 score as a metric. Values of the F_1 score takes into account the precision and recall value and is limited by lowest value present which gives a more accurate representation of the performance of our system. This table also serves to prove that the naive approach of using local maxima is insufficient for the detection of atoms within TEM images.

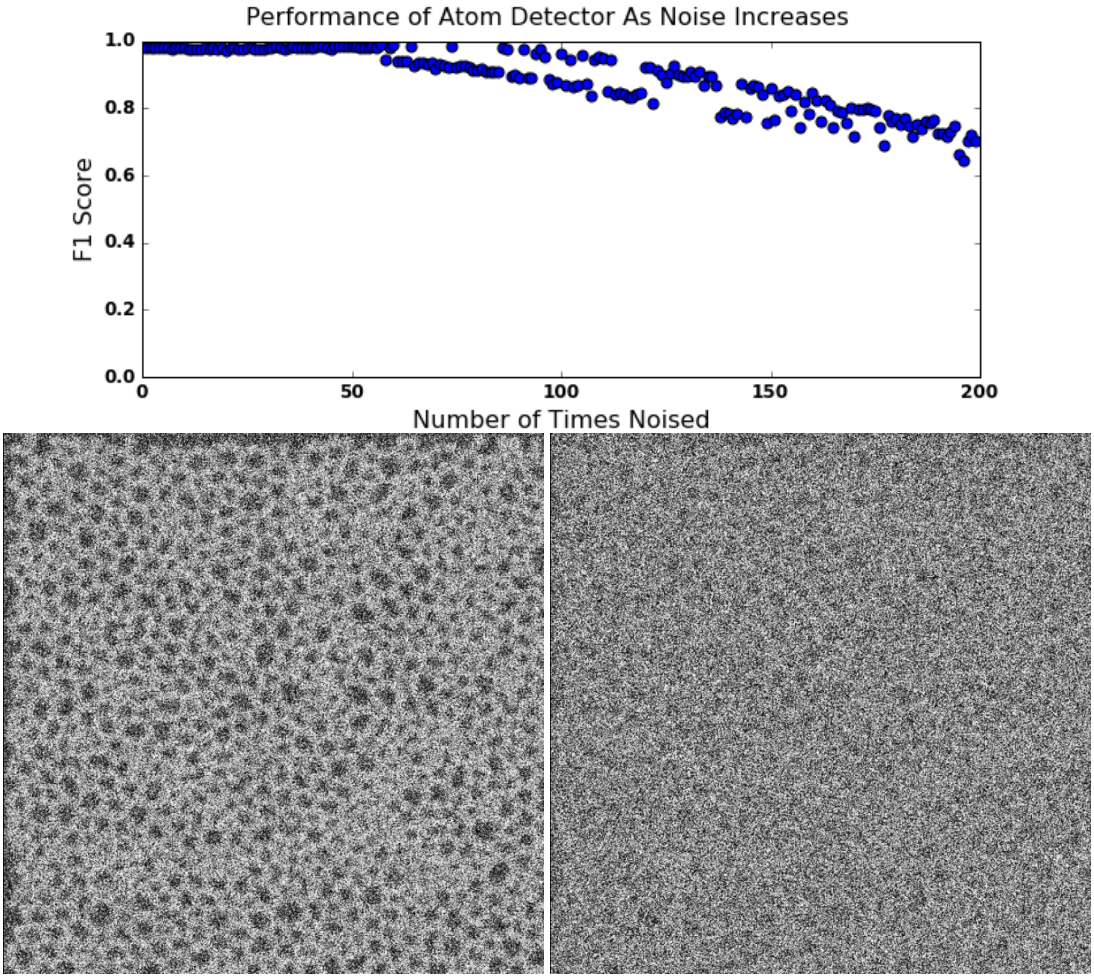
The inclusion of noise within testing was clearly extremely important. In Table 4.2, if we exclude all real nanoscale images and include only the synthetic images, we see that if not for the inclusion of noise within the tests we may conclude that our intensity based method out performs the second derivative based method that our system uses. This noise was added to existing images by first creating a “noise” image where values were randomly generated on a

Image	LM P	System P	LM R	System R	LM F_1	System F_1
Noisy Synthetic Images	0.2492	1.0000	0.9964	0.9814	0.3987	0.9906
Synthetic Images	0.5597	1.0000	0.9976	0.9805	0.7170	0.9902
High Background h-BN	0.1186	1.0000	1.0000	0.9702	0.2121	0.9849
h-BN Defects	0.3164	1.0000	0.9850	0.9624	0.4790	0.9808
High Distortion Graphene	0.0976	0.9620	1.0000	1.0000	0.1779	0.9806
HiRes Graphene	0.2558	0.9785	1.0000	0.9785	0.4073	0.9785
Blurry Graphene	0.3714	0.9860	0.9828	0.9691	0.5391	0.9775
Small Peak Graphene	0.0582	0.9635	1.0000	0.9778	0.1099	0.9706
AC Graphene	0.0764	0.8283	0.8000	0.9111	0.1395	0.8677

Table 4.1: Comparing results between the local maxima of the image and our System. P is the precision of the system, where the value is the ratio of ground truth points detected vs not detected. R is recall, which is the ratio of ground truth points detected vs the number of detected points. A value of one means that all points are ground truth and a value of zero means none of the detected points belong to the ground truth.

uniform distribution. In Fig. 4.2 we see the effects of increasing the amount of noise of an image used as input into our system. The values within the noise image would then be applied to the original image by performing matrix addition upon the original image and the noise image. The exact range of values for this noise image was from negative 25 to positive 25. In order to encompass a larger range of synthetic images we included the ability to ramp the values within images. This ramping was in part inspired by both Ophus et al.'s [56] method and to reduce the uniformity of peak intensities within images. Omitting atoms within the synthesizing process allows us to further introduce defects within the images and more greatly test the capabilities of our system.

If we compare all methods, as shown in Table 4.2, we see that every method, aside from local maxima, has decent performance for non-noisy synthetic images, $F_1 > 0.7$. This is beacuse these images are the ideal conditions for any atomic detection algorithm. As there is no noise present within the images, each method can more easily identify the maxima that represents the atomic locations. We also see that the intensity based version of our method outperforms our proposed system. This is due to the method having been designed for the ideal case, a noiseless image. However, as we introduce noise and real data into the tests we



(a) Image that has noised 70 times (b) Image that has noised 140 times

Figure 4.2: Performance Plot of Noised Images with example images. The example images, (a) and (b), show possible outputs for the noising being done to the original image. As we can see (b) still has a decent F_1 score even though the image is barely recognisable.

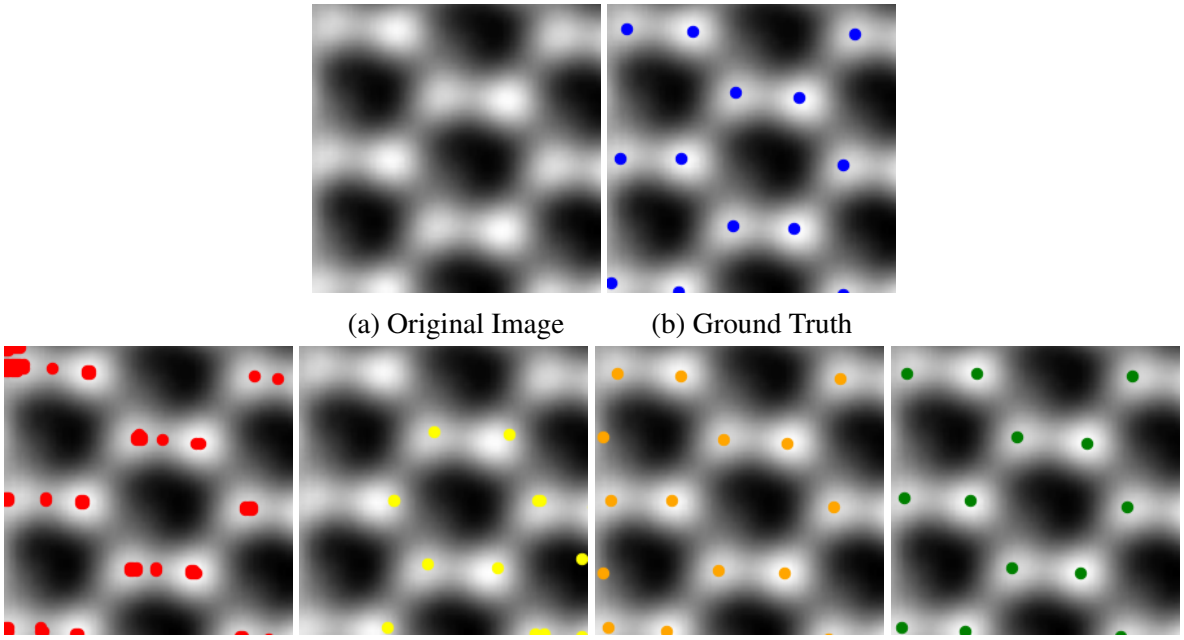
Image	LM F_1	Ophus et al.'s F_1	Intensity Based F_1	System F_1
Noisy Synthetic Images	0.3992	0.0989	0.0165	0.9940
Synthetic Images	0.7159	0.8797	0.9993	0.9939
High Background h-BN	0.2121	0.1851	0.7943	0.9849
h-BN Defects	0.4790	0.7511	0.4422	0.9808
High Distortion Graphene	0.1779	0.0673	0.9264	0.9806
HiRes Graphene	0.4073	0.6574	0.6075	0.9785
Blurry Graphene	0.5391	0.8872	0.0068	0.9775
Small Peak Graphene	0.1099	0.2977	0.6207	0.9706
AC Graphene	0.1395	0.1243	0.6311	0.8677

Table 4.2: Comparing the F_1 scores of all the methods tested.

see the intensity based method fail, in addition to all other methods, excluding our proposed system.

Fig. 4.3 contains a visual representation of sample output for each of the presented systems with the Blurry Graphene TEM image as input along with the ground truth data for the image. Each output has notable characteristics that can be explained. In the local maxima output, we see that there is an excess of detected atoms. These incorrect atoms detections mostly occur along the areas of overlapping charge between atoms. This is because when viewing the image as a surface map, the charge between the atoms form saddle points. These saddle points are incredibly sensitive to noise. Any noise within these saddle points tend to create miniature peaks. This can be better seen in Fig. 3.7. Notice that within the saddle point between the atoms we see small peaks. Fig. 4.3d shows that Ophus et al.'s [56] method output can offset from the ground truth. This offset is caused by the location of the local minima that represents the centre of a ring of atoms. When this centre is not equidistant from the surrounding atoms it causes the aformentioned offset. The output from the intensity based method and the second derivative based method closely resemble the ground truth points. However the intensity based method outputs incorrect detections along the left edge of the image. This is because just outside the image's left boundary there are more atoms, however they should not be detected as they are not complete atoms and the resulting points are not accurate atomic locations.

Table 4.4 shows the performance of the stages of the system using the second derivative



(a) Original Image (b) Ground Truth

(c) Local maxima (d) Ophus et al.'s method (e) Intensity based (f) System

Figure 4.3: Showing the output for all methods presented within the thesis along with the ground truth for the TEM image [22]. (a) shows the original image and (b) is (a) annotated with its ground truth data. (c), (d), (e), and (f) are the output for each particular method with (f) being the output for the second derivative based method.

Image	I	I+M	I+S	I+M+S
Synthetic Images	0.9937	0.9937	0.9937	0.9937
Noisy Synthetic Images	0.9866	0.9867	0.9866	0.9867
High Background h-BN	0.9849	0.9849	0.9849	0.9849
h-BN Defects	0.9808	0.9808	0.9808	0.9808
High Distortion Graphene	0.9806	0.9806	0.9806	0.9806
HiRes Graphene	0.9785	0.9785	0.9785	0.9785
Blurry Graphene	0.9775	0.9775	0.9775	0.9775
Small Peak Graphene	0.9706	0.9706	0.9706	0.9706
AC Graphene	0.8511	0.8511	0.8677	0.8677

Table 4.3: F_1 scores for each stage of the detection system when using parameter learning. For clarification: Stage 1 is the initial atomic locations, Stage 2 is after fixing split detections, and Stage 3 is after fixing merged detections.

Image	I	I+M	I+S	I+M+S
h-BN Defects	0.9963	0.9963	0.9963	0.9963
High Background h-BN	0.9956	0.9956	0.9956	0.9956
Synthetic Images	0.9937	0.9937	0.9937	0.9937
Noisy Synthetic Images	0.9869	0.9871	0.9869	0.9871
Blurry Graphene	0.9684	0.9684	0.9684	0.9684
HiRes Graphene	0.9648	0.9648	0.9648	0.9648
Small Peak Graphene	0.9247	0.9408	0.9247	0.9408
High Distortion Graphene	0.9075	0.9383	0.9129	0.9383
AC Graphene	0.8083	0.7937	0.8308	0.8229

Table 4.4: F_1 scores for each stage of the detection system when not using parameter learning. For clarification: Stage 1 is the initial atomic locations, Stage 2 is after fixing split detections, and Stage 3 is after fixing merged detections.

based method without the use of parameter learning. In Table 4.4 we see that for most images, there is little to no improvment made to the F_1 score of the system in subsequent stages. This is due to the initial detections for many images often present a small amount of error. In some cases there is a decrease in F_1 score during stages 1 and 2. If there is a decrease in the F_1 score during stage 2 this means that either an incorrect bondlength was used for the method or the resolution of the image is too low to distinguish between areas that do and do not belong to atoms. If there is a decrease in stage 3 it is incredibly likely that the blobs contained within the segmentation image are very similar in appearance. Essentially if the features used to distinguish the erroneous contours have a small standard deviation then the method can produce incorrect atomic locations. However this problem can be solved by checking the standard deviation ahead of time and if it is too small we do not perform the third step.

In comparing values from Table 4.3 and Table 4.4 we see that when using parameter learning in our process, we acheive better results on average. This behaviour is expected due to the fact that we are attempting to learn the best parameters for the specific data set. However the increased accuracy of the results is not without its disadvantages. Table 4.5 shows us that run times increase significantly when using the parameter learning process. Another caveat of using the parameter learning process is that for some images we see a decrease in the F_1 score. This is most likely due to the parameters used during the testing process fitting the data more

Image	With Learning	Without Learning	Ophus et al.	Intensity Based
Noisy Synthetic Images	32.3426	0.6392	0.1908	0.1770
Synthetic Images	39.8235	0.6255	0.0946	0.1204
High Background h-BN	2.9992	0.1189	0.0189	0.0332
HiRes Graphene	4.3782	0.1036	0.0128	0.0371
Blurry Graphene	1.6788	0.1006	0.0109	0.0017
High Distortion Graphene	2.9880	0.0974	0.0207	0.0278
Small Peak Graphene	2.4643	0.0830	0.0101	0.0457
h-BN Defects	1.2994	0.0686	0.0045	0.0108
AC Graphene	3.1059	0.0655	0.0101	0.0339

Table 4.5: Comparing the runtimes, in seconds, of all valid methods discussed within this paper.

accurately than the results of the parameter learning process. In this case, it can be said that the algorithm used for determining error during the learning process can be improved upon or a new cost function can be introduced.

The cost function used during the parameter learning process can greatly affect the results, F_1 score, of our system. In Table 4.6, the results from both created cost functions is displayed for each of the experimental images along with the image size and elapsed. The F_1 scores for the system may be the best according to the cost functions used, however it is very likely that the cost functions that are currently being used are not optimal cost functions for the system and report less than accurate parameters. For example, even though the bond length based cost function generally outperforms the area based cost function there are some images where the performance of both cost functions is equal and there is a case where the area based cost function outperforms the bond length based cost function by a small margin.

In the same table, Table 4.6, it can be seen that the time cost scales in a roughly linearly with respect to the number of pixels. The linear nature of the time costs is likely due to the segmentation portion of the algorithm as it contains the largest number of computations of entire system. However since the time cost for a single image is so low the time results of this system may be inconsequential.

Table 4.7 shows the F_1 score of the second derivative based atomic detection method and

		Area Based		Bond Length Based	
Image	Pixels	Time	F_1	Time	F_1
Synthesized	480000	5.5543	0.0044	19.9553	0.9941
HiRes Graphene	203648	2.3232	0.9785	2.5049	0.9785
AC Graphene	144550	1.5523	0.8256	1.6335	0.8511
High Distortion Graphene	114380	1.2497	0.0128	1.4373	0.9806
Small Peak Graphene	113262	1.2306	0.9778	1.3516	0.9742
High Background h-BN	110871	1.2292	0.0118	1.4306	0.9864
Blurry Graphene	62500	0.7238	0.0136	0.9066	0.9775
h-BN Defects	53580	0.6246	0.8125	0.6234	0.9808

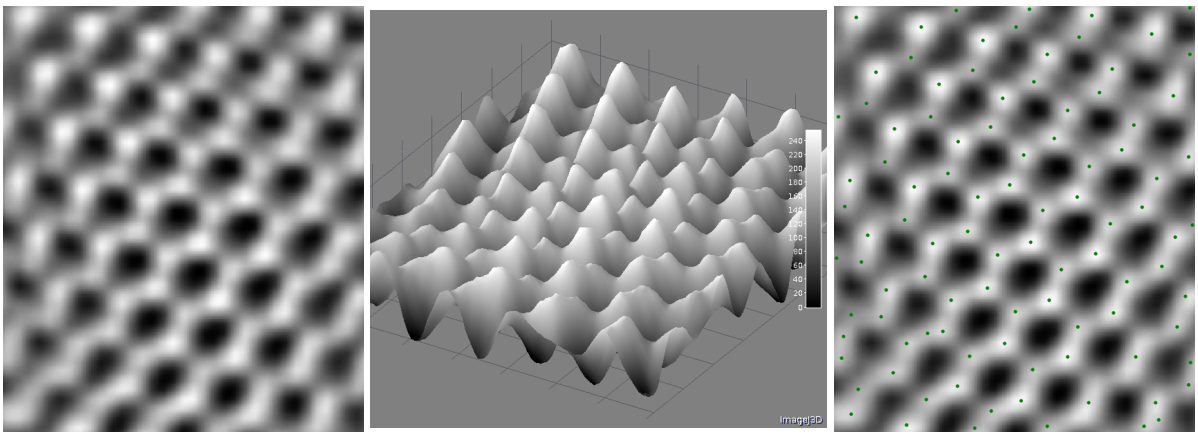
Table 4.6: Evaluating the performance of both cost function presented, in both time and F_1 score. We see that generally the bond length based cost function generally outperforms the area based method, however the area based method has less of a time cost.

the Voronoi based detections proposed by Ophus et al. along with the F_1 scores of the neighbourhoods produced by both methods. These tests were performed by first matching each detection to a ground truth point. The match criteria was based on the euclidean distance between the detection's coordinates and the ground truth coordinates on the image. If the distance between a single detection and a atomic location in the ground truth data is smaller than some predetermined threshold in pixels, we then consider the detection and the ground truth to match. This matching step is normally performed when generating scores for our data. After matching the detections to the ground truth data we then check to see if a connection within the ground truth neighbourhood exists between the detected points. By checking the existance of matching connections between the ground truth neighbourhood and those generated during detection, we can produce an F_1 score for the generated neighbourhoods. In Table 4.7, the F_1 score for the neighbourhoods produced using both our neighbourhood generation method and our atomic detection method generally outscore the neighbourhoods generated by the Voronoi based approach. However, whether or not this is due to the initial set of points into the neighbourhood generation portions of each algorithm is arguable.

It is notable that all tables showing the performance, F_1, of each method show that the image "AC Graphene" performs the least favorably compared to all other images by a large margin. Upon inspection of the "AC Graphene" image it is noticeable that there is distortion in

	System		Ophus et al.	
Image	Point F_1	Line F_1	Point F_1	Line F_1
HiRes Graphene	0.9785	0.9985	0.7409	0.6109
h-BN Defects	0.9808	0.9718	0.6762	0.6237
Small Peak Graphene	0.9706	0.9644	0.6265	0.4700
High Distortion Graphene	0.9806	0.9623	0.3785	0.0760
High Background h-BN	0.9849	0.9597	0.7162	0.5116
Blurry Graphene	0.9775	0.9448	0.8270	0.8267
AC Graphene	0.8677	0.8452	0.1864	0.0620

Table 4.7: Comparing the F_1 Score of the resulting generated Neighbourhoods to the F_1 Score of the points.



(a) (b) (c)

Figure 4.4: Example TEM image along with corresponding 3D plots and Line Plot. (a) is the original experimental image of aberration corrected (AC) Graphene[60]. (b) is (a) projected into a surface using the intensity as height. (c) is the same image as (d) with overlaid with initial uncorrected system output.

the extremities of the image, as one might be able to see in Fig. 4.4(a). This is likely due to the aberration correction needed to achieve TEM images at this resolution. Upon inspecting the image as a surface map this becomes more clear as we see that the peak intensities for the outer edges of the image are unlikely to be at the locations depicted for a clear image of Graphene, Fig. 4.4(b,c). Furthermore, we see that many of the errors within the detections of the image coincide with the peaks of the surface map. Testing on this image is useful due to the challenge it poses and the possibility of using *a priori* knowledge order to properly reconstruct the lattice despite the lack of information represented within the image.

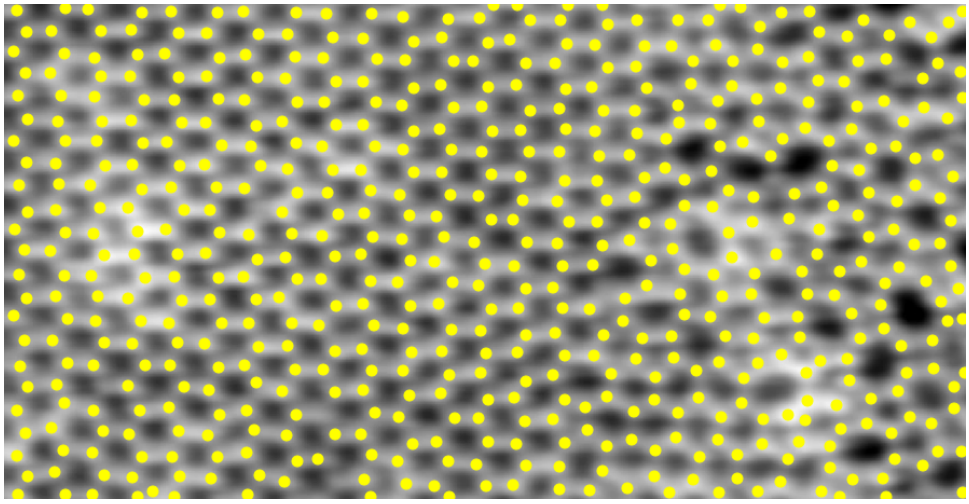
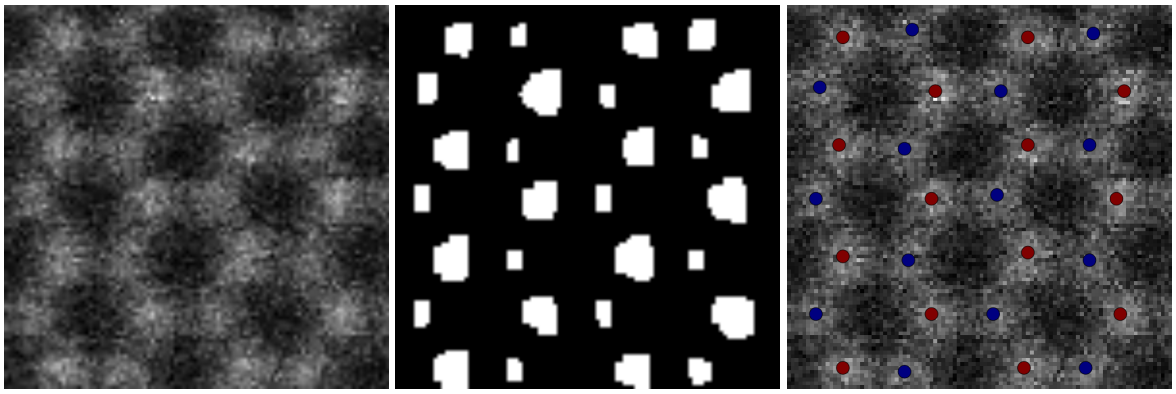


Figure 4.5: Sample output for a STM image [63] of a crystalline-vitreous interface annotated with the output of the second derivative based atomic detection system. In this image our system is able to identify most Si atoms within it, however the system is not able to identify O atoms. Notice that there are areas within the image, especially on the right where the lattice is irregular, that lack appropriate output.

There are situations where all algorithms can fail to identify atoms completely. Fig. 4.5 demonstrates the conditions under which the atomic detection algorithm fails. The figure depicts a STM image of a crystalline-vitreous interface that is composed of both Si atoms and O atoms. However, the output of the system is only able to identify the Si atoms. This is because the oxygen atoms do not present a large enough increase in intensity within the STM image. There is also clearly missing output for Si atoms. Upon inspecting the image we found that there was not enough of a change within the image to locate the atoms. Whether or not this is caused by an error within the STM capture process, the nature of the STM image capture process, or limitations within the atomic detection algorithms is unknown. However, it is unlikely that the latter is the case as upon visual inspection of the STM image it can be determined that there is very little indication of the oxygen atoms present within the image.

Using the images of h-BN there has been some preliminary research into the topic of determining the type of atom depicted within an image. The lattice h-BN, has a very similar appearance to graphene. In graphene, each carbon atom is bonded to three other carbon atoms and forms hexagonal rings. In h-BN, there is also hexagonal rings, however each boron atom is



(a) (b) (c)

Figure 4.6: Example of grouping species of atoms based on the size of their segmentations. (a) is a cropped image of hexagonal Boron Nitride, (b) is the segmentation of of that cropped image. (c) highlights the species within the image. The results within (c) were found by using principal component analysis and k-means clustering.

bonded to three nitrogen atoms and each nitrogen atom is bonded to three boron atoms. Using this information it may be possible to group each of the atoms in the image with other atoms of the same species. In Fig. 4.6(b), it is shown that based on the species of atom, the size and shape of the segmentation for atoms can vary greatly. There we see that atoms corresponding to one species have a very different appearance, in size, from the other species in the image. Using this appearance we performed some investigations into using size and shape of the segmentations in order to determine groupings of atoms. Fig. 4.6(c) shows more on this, where atoms have been grouped simply based on their size. Here we have used both principal component analysis on a number of variables calculated from the segmentations along with k-means clustering in order to find the likely species within the image. However we could not find a method that was able to find the proper groupings with a large degree of accuracy.

Chapter 5

Conclusion and Future Work

The conclusion will be organised as follows. First a brief summary of the contributions made within this thesis will be presented. Following the contributions, the evalutions of the performance of our system compared to other existing works and other possible systems will be compared and shortly discussed. The programming libraries used and other implementation specifics are described along with their performance. Lastly, the direction of future work into the problem will be discussed. The discussion will include possible optimisation methods and possible features to add to the system.

5.1 Contributions

In this thesis, a solution has been presented for the problem of detecting atoms and inferring molecular structure in electron microscopy images that should be applicable to other forms of nanoscale imagery. We presented two methods of segmenting and analysing electron microscopy images for atoms. These methods either use the intensity of an image or second order derivative of the image in order to perform segementation. Using the segmented images, we derived the locations for all atoms within the image. From the atomic locations we infer the structure of the molecule and reconstruct the bonds for each atom. We also demonstrated that with supervised learning we can improve the performance of the proposed system.

5.2 Performance Evaluation

To evaluate the performance of our approach we have created a system to synthetically produce high resolution transmission electron microscopy images that mimic those that have been experimentally produced. All presented algorithms have been implemented using the Python programming language in conjunction with the established mathematical libraries NumPy [61] and SciPy [62] and the computer vision library OpenCV [58].

Our experiments have compared the performances of not only our intensity based and second derivative based methods but also to the state-of-art approach by Ophus et al. [56] and the local maxima of the image. We evaluated all methods using the output of both experimentally and synthetically produced images.

Our experiments have shown that our second derivative based method tend to outperform all other methods that have been tested. However, results show that the second derivative based solution is slower than other solutions and can possibly take more than 10 times the amount of time for a single image. This can be considered acceptable as the time for the completion of the curvature based method is still smaller than a second. This changes once supervised learning is applied, as times increase drastically.

5.3 Future Work

There are a number of extensions to this thesis that we would like to explore in future investigations. These extensions include both extensions in function and design and in implementation.

1. Identifying Types of Atoms

   The detection method proposed in this thesis focuses solely on the detection of the existence of an atom and determines the atom's location. However in Fig. 3.2 where we have segmented a Transmission Electron Microscopy image of hexagonal boron nitride we see that the regions representing the two different kinds of atoms are distinct in size.

By visually inspecting the image we can differentiate between the two types. One could extend the approach in this thesis by adding the functionalilty neccessary to perform the task of differentiating between the atoms within electron microscopy images.

2. Supervised Learning using GPU

   Currently the most time consuming portion of the proposed system is where learning occurs. It may be possible that the time taken for the learning process to complete can be reduced by implementing the learning process on a graphics proccessing unit (GPU). However, we forsee that there could exist difficulties with implementing the learning process on GPU as each learning process has several complex stages included within it, such as image segmentation and neighbourhood generation.

3. Probabilities of Correct Detection

   The system does not take into account the blurriness of the image or other factors and will perform the entire algorithm even if the image is impossible to process correctly. The system makes the assumption that not only is there a lattice within the image but also that any detected atom simply exists. However, it is known that this is not always the case. Errors exist within the detections. It would be useful to users to present a probability for each detection that details how likely a detection is to exist. An extension to this thesis might be the ability to present the likelihood of a correct detection for each atom. A simple method of implementing this method is to assign all original detections a score of 1 and decrease that score each time the detection fails during one of the correction stages. It would also be possible to extend this method to bonds where we could assign each bond a value of 1 and decrease the value if the atoms do not agree that they are bonded.

4. Multilayer Analysis

   Most lattices do not exist in single layers. Within this thesis we used graphene and hexagonal boron nitride for testing as they provided a challenge that was simple to per-

form with the human eye but difficult to perform using computation. The extension of the thesis in order to analyse atomic lattices with multiple layers would be very useful. However this extension would not be easy to perform and would not be without it's own set of challenges, especially for the analysis of two dimensional nanoscale imagery. The author of this paper is unsure how to perform multilayer analysis of nanoscale without the inclusion of a third dimension provided by an Atomic Force Microscope. The inclusion of the third dimension would allow similar methods to those in our system to be performed in 3 dimensions and should be able to perform detection.

The stated future work into the system would aid in the general applicability of the system as it would be able to provide more information to the user and researchers as well as decrease the time needed to analyse the specimens.

Bibliography

[1] Douglas Philp and J Fraser Stoddart. Self-assembly in natural and unnatural systems. *Angewandte Chemie International Edition in English*, 35(11):1154–1196, 1996.

[2] Stephen Whitelam, Isaac Tamblyn, Juan P. Garrahan, and Peter H. Beton. Emergent rhombus tilings from molecular interactions with m-fold rotational symmetry. *Phys. Rev. Lett.*, 114:115702, Mar 2015. doi: 10.1103/PhysRevLett.114.115702. URL `http://link.aps.org/doi/10.1103/PhysRevLett.114.115702`.

[3] Stephen Whitelam, Isaac Tamblyn, Thomas K. Haxton, Maria B. Wieland, Neil R. Champness, Juan P. Garrahan, and Peter H. Beton. Common physical framework explains phase behavior and dynamics of atomic, molecular, and polymeric network formers. *Phys. Rev. X*, 4:011044, Mar 2014. doi: 10.1103/PhysRevX.4.011044. URL `http://link.aps.org/doi/10.1103/PhysRevX.4.011044`.

[4] John A Pelesko. *Self assembly: the science of things that put themselves together*. CRC Press, 2007.

[5] Heinz Fraenkel-Conrat and Robley C Williams. Reconstitution of active tobacco mosaic virus from its inactive protein and nucleic acid components. *Proceedings of the National Academy of Sciences*, 41(10):690–698, 1955.

[6] Lawrence Bragg. A model illustrating intercrystalline boundaries and plastic flow in metals. *Journal of Scientific Instruments*, 19(10):148, 1942. URL `http://stacks.iop.org/0950-7671/19/i=10/a=302`.

[7] Lawrence Bragg and J. F. Nye. A dynamical model of a crystal structure. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 190 (1023):474–481, 1947. ISSN 0080-4630. doi: 10.1098/rspa.1947.0089. URL http://rspa.royalsocietypublishing.org/content/190/1023/474.

[8] M Golosovsky, Y Saado, and D Davidov. Self-assembly of floating magnetic particles into ordered structures: A promising route for the fabrication of tunable photonic band gap materials. *Applied Physics Letters*, 75(26):4168–4170, 1999.

[9] DoITPoMS, University of Cambridge. Bubble raft, 2016. URL http://www.doitpoms.ac.uk/tlplib/dislocations/images/creatingraft2.jpg. [Online; accessed October 19, 2016].

[10] Electron Microscopy Group, University of Cambridge. High-resolution image of a sigma 19 grain boundary in al, 2016. URL http://www-hrem.msm.cam.ac.uk/gallery/pics/allarge.jpg. [Online; accessed October 19, 2016].

[11] Yongchao Si and Edward T. Samulski. Exfoliated graphene separated by platinum nanoparticles. *Chemistry of Materials*, 20(21):6792–6797, 2008. doi: 10.1021/cm801356a. URL http://dx.doi.org/10.1021/cm801356a.

[12] Andrii Goriachko, Yunbin He, Marcus Knapp, Herbert Over, Martina Corso, Thomas Brugger, Simon Berner, Juerg Osterwalder, and Thomas Greber. Self-assembly of a hexagonal boron nitride nanomesh on ru (0001). *Langmuir*, 23(6):2928–2931, 2007.

[13] Ned Bowden, Andreas Terfort, Jeff Carbeck, and George M Whitesides. Self-assembly of mesoscale objects into ordered two-dimensional arrays. *Science*, 276(5310):233–235, 1997.

[14] Jesus de La Fuente. Properties of graphene. http://www.graphenea.com/pages/graphene-properties.

[15] George M Whitesides and Bartosz Grzybowski. Self-assembly at all scales. *Science*, 295 (5564):2418–2421, 2002.

[16] Kostya S Novoselov, Andre K Geim, Sergei V Morozov, D Jiang, Y_ Zhang, Sergey V Dubonos, Irina V Grigorieva, and Alexandr A Firsov. Electric field effect in atomically thin carbon films. *science*, 306(5696):666–669, 2004.

[17] Peter Sutter, Jerzy T. Sadowski, and Eli Sutter. Graphene on pt(111): Growth and substrate interaction. *Phys. Rev. B*, 80:245411, Dec 2009. doi: 10.1103/PhysRevB.80.245411. URL http://link.aps.org/doi/10.1103/PhysRevB.80.245411.

[18] A. H. Castro Neto, F. Guinea, N. M. R. Peres, K. S. Novoselov, and A. K. Geim. The electronic properties of graphene. *Rev. Mod. Phys.*, 81:109–162, Jan 2009. doi: 10.1103/RevModPhys.81.109. URL http://link.aps.org/doi/10.1103/RevModPhys.81.109.

[19] Yi Huang, Jiajie Liang, and Yongsheng Chen. An overview of the applications of graphene-based materials in supercapacitors. *Small*, 8(12):1805–1834, 2012. ISSN 1613-6829. doi: 10.1002/smll.201102635. URL http://dx.doi.org/10.1002/smll.201102635.

[20] He Shen, Liming Zhang, Min Liu, and Zhijun Zhang. Biomedical applications of graphene. *Theranostics*, 2(3):283–294, 2012.

[21] Chul Chung, Young-Kwan Kim, Dolly Shin, Soo-Ryoon Ryoo, Byung Hee Hong, and Dal-Hee Min. Biomedical applications of graphene and graphene oxide. *Accounts of chemical research*, 46(10):2211–2224, 2013.

[22] Wataru Norimatsu, Koichiro Hirata, Yuta Yamamoto, Shigeo Arai, and Michiko Kusunoki. Epitaxial growth of boron-doped graphene by thermal decomposition of

b 4 c. *Journal of Physics: Condensed Matter*, 24(31):314207, 2012. URL http://stacks.iop.org/0953-8984/24/i=31/a=314207.

[23] D. Alloyeau, E. Riccardi, J. Lagoutte, C. Ricolleau, G. Wang, and Y. Gallais. Ms-2-p-2699 multi-scale investigations of nitrogen doping in graphene. Presented at the 18th International Microscopy Congress in Prague, 2014.

[24] Recep Zan, Quentin M. Ramasse, Rashid Jalil, and Ursel Bangert. Atomic structure of graphene and h-bn layers and their interactions with metals. 2013-07-31. doi: 10.5772/56640. URL http://www.intechopen.com/books/export/citation/BibTex/advances-in-graphene-science/atomic-structure-of-graphene-and-h-bn-layers-and-their-interactions-w

[25] Nasim Alem, Rolf Erni, Christian Kisielowski, Marta D. Rossell, Will Gannett, and A. Zettl. Atomically thin hexagonal boron nitride probed by ultrahigh-resolution transmission electron microscopy. *Phys. Rev. B*, 80:155425, Oct 2009. doi: 10.1103/PhysRevB.80.155425. URL http://link.aps.org/doi/10.1103/PhysRevB.80.155425.

[26] Kenji Watanabe, Takashi Taniguchi, and Hisao Kanda. Direct-bandgap properties and evidence for ultraviolet lasing of hexagonal boron nitride single crystal. *Nature materials*, 3(6):404–409, 2004.

[27] Zheng Liu, Lulu Ma, Gang Shi, Wu Zhou, Yongji Gong, Sidong Lei, Xuebei Yang, Jiangnan Zhang, Jingjiang Yu, Ken P Hackenberg, et al. In-plane heterostructures of graphene and hexagonal boron nitride with controlled domain sizes. *Nature nanotechnology*, 8(2): 119–124, 2013.

[28] Jiamin Xue, Javier Sanchez-Yamagishi, Danny Bulmash, Philippe Jacquod, Aparna Deshpande, K Watanabe, T Taniguchi, Pablo Jarillo-Herrero, and Brian J LeRoy. Scan-

ning tunnelling microscopy and spectroscopy of ultra-flat graphene on hexagonal boron nitride. *Nature materials*, 10(4):282–285, 2011.

[29] nanoScience Instruments. Schematic of scanning tunneling microscopy (stm), 2016. URL `http://www.nanoscience.com/files/7114/5590/6061/STM_schematic_labeled_w.png`. [Online; accessed October 19, 2016].

[30] Aspire. Aspire ct170 conical tapping mode afm probes, 2016. URL `http://store.nanoscience.com/store/pc/catalog/03_oem-probe-series_tn_165_detail.jpg`. [Online; accessed October 19, 2016].

[31] Peter Eaton and Paul West. Schematic of afm operation, 2016. URL `http://afmhelp.com/images/stories/afm_scheme.png`. [Online; accessed October 19, 2016].

[32] Imaging And Microscopy. Highest resolution scanning/ transmission electron microscope ((s)tem), 2016. URL `http://www.imaging-git.com/sites/imaging-git.com/files/images/special/2600755__original.jpg`. [Online; accessed October 20, 2016].

[33] John CH Spence. *High-resolution electron microscopy*. OUP Oxford, 2013.

[34] . Schematic view of a transmission electron microscope with its lenses and the pathway of the electrons. ccd, charged coupled device., 2016. URL `http://intranet.tdmu.edu.ua/data/kafedra/internal/histolog.../classes_stud/en/stomat/ptn/1/01%20Microscope.%20Microscopic%20equipment.%20Histologic%20technique.%20Cytology.%20General%20structure%20of%20the%20cell.%20Superficial%20complex.files/image034.jpg`. [Online; accessed October 20, 2016].

[35] Thomas B Moeslund and Erik Granum. A survey of computer vision-based human motion capture. *Computer vision and image understanding*, 81(3):231–268, 2001.

[36] Thomas B Moeslund, Adrian Hilton, and Volker Krüger. A survey of advances in vision-based human motion capture and analysis. *Computer vision and image understanding*, 104(2):90–126, 2006.

[37] Gerald J Agin. Computer vision systems for industrial inspection and assembly. *IEEE Computer*, 13(5):11–20, 1980.

[38] OpenCV. image, 2016. URL `http://docs.opencv.org/trunk/face.jpg`. [Online; accessed October 20, 2016].

[39] Marc Pollefeys. 3d surface model obtained automatically from an uncalibrated image sequence, shaded (left), textured (right)., 2016. URL `http://www.cs.unc.edu/~marc/tutorial/img778.png`. [Online; accessed October 20, 2016].

[40] Computer Vision Lab. tr_breitenstein162, 2016. URL `https://www.vision.ee.ethz.ch/showroom/tracking/pics/tr_Breitenstein162.png`. [Online; accessed October 20, 2016].

[41] Adam G Kirk, James F O'Brien, and David A Forsyth. Skeletal parameter estimation from optical motion capture data. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 782–788. IEEE, 2005.

[42] Lawrence G Roberts. Machine perception of three-dimensional solids. 1963.

[43] David Marr. Vision: A computational investigation into the human representation and processing of visual information. 1982. *San Fransisco, CA: Henry Holt & Company*, 1982.

[44] Hedvig Sidenbladh, Michael J Black, and David J Fleet. Stochastic tracking of 3d human figures using 2d image motion. In *Proceedings of the 6th European Conference on Computer Vision-Part II*, pages 702–718. Springer-Verlag, 2000.

[45] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[46] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[47] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International conference on computer vision*, pages 2564–2571. IEEE, 2011.

[48] Nasim Alem, Oleg V. Yazyev, Christian Kisielowski, P. Denes, Ulrich Dahmen, Peter Hartel, Maximilian Haider, Maarten Bischoff, Bin Jiang, Steven G. Louie, and A. Zettl. Probing the out-of-plane distortion of single point defects in atomically thin hexagonal boron nitride at the picometer scale. *Phys. Rev. Lett.*, 106:126102, Mar 2011. doi: 10.1103/PhysRevLett.106.126102. URL `http://link.aps.org/doi/10.1103/PhysRevLett.106.126102`.

[49] Azriel Rosenfeld and Avinash C Kak. *Digital picture processing*, volume 1. Elsevier, 2014.

[50] Mark De Berg, Marc Van Kreveld, Mark Overmars, and Otfried Cheong Schwarzkopf. Computational geometry. *Computational Geometry*, pages 1–17, 2000.

[51] George Green. An essay on the application of mathematical analysis to the theories of electricity and magnetism. *Journal fr die reine und angewandte Mathematik*, 44:356–374, 1852. URL `http://eudml.org/doc/147512`.

[52] William F Eddy. A new convex hull algorithm for planar sets. *ACM Transactions on Mathematical Software (TOMS)*, 3(4):398–403, 1977.

[53] Ronald L Graham and F Frances Yao. Finding the convex hull of a simple polygon. *Journal of Algorithms*, 4(4):324–331, 1983.

[54] Xianquan Zhang, Zhenjun Tang, Jinhui Yu, and Mingming Guo. A fast convex hull algorithm for binary image. *Informatica*, 34(3), 2010.

[55] Andrew W Fitzgibbon, Robert B Fisher, et al. A buyer's guide to conic fitting. *DAI Research paper*, 1996.

[56] Colin Ophus, Ashivni Shekhawat, Haider Rasool, and Alex Zettl. Large-scale experimental and theoretical study of graphene grain boundary structures. *Phys. Rev. B*, 92: 205402, Nov 2015. doi: 10.1103/PhysRevB.92.205402. URL `http://link.aps.org/doi/10.1103/PhysRevB.92.205402`.

[57] C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. The quickhull algorithm for convex hulls. *ACM TRANSACTIONS ON MATHEMATICAL SOFTWARE*, 22(4):469–483, 1996.

[58] G. Bradski. The opencv library. *Dr. Dobb's Journal of Software Tools*, 2000.

[59] Satoshi Suzuki and KeiichiA be. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32 – 46, 1985. ISSN 0734-189X. doi: 10.1016/0734-189X(85)90016-7. URL `http://www.sciencedirect.com/science/article/pii/0734189X85900167`.

[60] reprinted from http://britishcarbon.org/images.shtml.

[61] Stfan van der Walt, S. Chris Colbert, and Gal Varoquaux. The numpy array: A structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–

30, 2011. doi: http://dx.doi.org/10.1109/MCSE.2011.37. URL http://scitation.aip.org/content/aip/journal/cise/13/2/10.1109/MCSE.2011.37.

[62] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. URL http://www.scipy.org/. [Online; accessed 2016-06-06].

[63] Leonid Lichtenstein, Markus Heyde, and Hans-Joachim Freund. Crystalline-vitreous interface in two dimensional silica. *Phys. Rev. Lett.*, 109:106101, Sep 2012. doi: 10.1103/PhysRevLett.109.106101. URL http://link.aps.org/doi/10.1103/PhysRevLett.109.106101.