Final Project

Samantha Hao, Jun Ha Song, Roma Patterson, Jaegun Lee, Ting Chih Lin

December 11, 2017

Introduction

Most processes in living organisms are governed by proteins, long polymers of organic molecules called amino acids. The biological material that encodes genetic information, DNA, codes for amino acids. The 20 amino acids encoded by mRNA (transcribed from DNA) are distinguished by their side chain structure, which imparts specific chemical properties to the amino acid. The interactions between the amino acids within a protein (also known as residues) determine its structure. Its structure, in turn, determines the chemical properties of the protein.

Proteins are essential to most biological processes. Most enzymes, which are organic catalysts, are proteins. Without enzymes, many of the chemical processes that organisms rely on to survive would not happen without unfeasible amounts of energy. Enzymes speed up reactions by lowering the energy barrier required for a chemical reaction to proceed. The chemical mechanism of this activation energy reduction can often be deduced from the structure of the enzyme. The site of reaction, known as the active site, will contain the appropriate residues that facilitate a reaction.

Additionally, many structural elements in organisms are protein-based. The extracellular matrix, which gives a cell its shape, is comprised of proteins. The keratin in human hair and nails is a fibrous structural protein. In the genome, DNA is wrapped around protein complexes called histones, which allows for very tight packing of genetic information inside a cell. In general, various biological molecules are associated with proteins to control structure and availability.

Therefore, a few good images of a protein can answer many questions about the mechanisms underlying life.

Protein Structure

Fundamentally, the conformation (three-dimensional structure) of a protein is related to its function, and one of the primary goals in biology involves understanding how various proteins interact with a larger pathway or system in an organism. The basic structure of a protein is classified into three levels: primary, secondary, and tertiary. There is also a quaternary structure which involves a complex of multiple protein subunits, but it does

not exist for every protein. The primary structure refers to the sequence of amino acids within the polymer, which is called a polypeptide chain. The secondary structure refers to the regular organization of local substructures on the polypeptide backbone chain. It is comprised of two main types of structures: α-helix and β-sheet. The tertiary structure refers the three-dimensional structure that involves the folding of α-helices and β-sheets into an overall protein structure, mediated by interactions between the side chains of residues.

We may chemically sequence the amino acid chain to decipher primary structure, but secondary and tertiary structure, which rely on the interactions between the protein backbone and residue side chains, cannot be readily deduced from the primary structure. Most polypeptide chains are hundreds to thousands of residues long, resulting in an astronomical number of possible conformations due to different possible choices of angles between residues, along with interactions between residues. For very small polypeptide chains, there exist computational methods to decipher overall structure; however, these fail when the polypeptide is of any significant length. Electron microscopy allows us to see these atomic structures and decipher the overall structure of a protein, giving insight into its function and mechanisms of interest.

Transmission Electron Microscopy

One method of obtaining images, Transmission Electron Microscopy(TEM), can be used to answer questions such as "How do our brain cells communicate with one another?" by a single electron micrograph, shown in Figure 1.

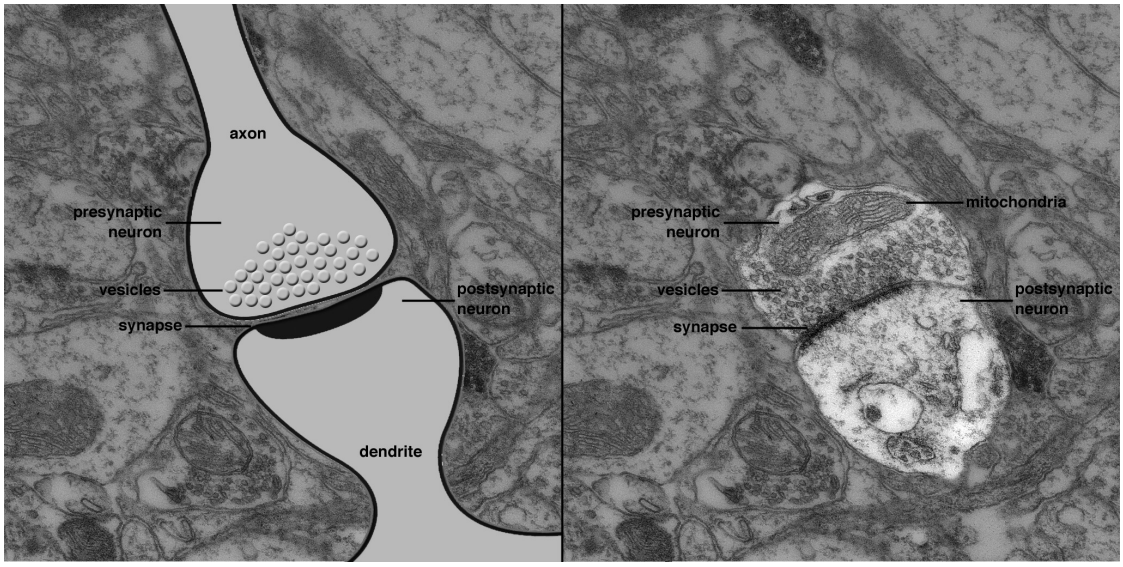


Figure 1: Nerve synapse where two neurons meet one another can easily be visualized by TEM [1]

TEM bombards the sample with an electron beam such that electrons either pass

through, absorb, or scatter. The result produces a 2 dimensional black and white image, called a micrograph. Light microscopes are limited by the diffraction of visible light, but TEM uses electrons, which have a much smaller wavelength than visible light, so it has the ability to image samples with much greater resolution. This makes it the optimal technique for imaging samples on nanometer scale.

Fundamental Challenges of TEM for Biological Samples

Despite TEM's high resolution, there are fundamental problems for imaging biological samples.

1. Due to the high reactivity of electrons, the electron beam must be operated in a vacuum to prevent electrons from interacting with air molecules. Thus the biological sample of interest must be dehydrated beforehand.

2. TEM projects 2 dimensional images, but the original sample is a three dimensional object.

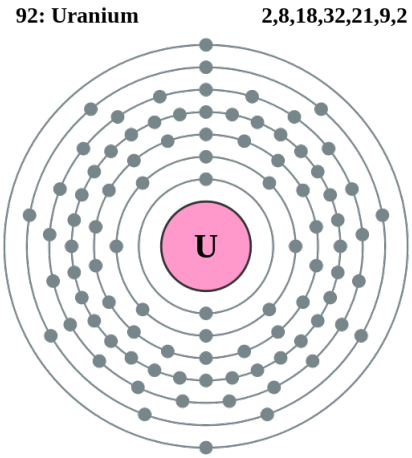


Figure 2: Uranium atom has enough electrons in its electron shell to deflect the incoming electron beam [2]

Traditional TEM for Biological Samples

The sample must be treated to withstand the high vacuum of TEM using the following processes;

1. Fixation, where protein molecules are linked together by a chemical fixative.

2. Dehydration, where the water of the cell is replaced by an organic solvent, such as ethanol.

3. Resin Embedding, where the cell is put into resin (hard plastic) and sectioned (sliced).

4. Staining, where heavy-metal ions bind with cells in resin sections.

These steps create a heavy-metal cast that surrounds the original protein structure. Heavy metals used for staining (typically Uranium or Lead) contain a high number of electrons as shown in Figure 2, which deflects the incoming electron beam. As a result, cells that have been treated with uranium yield high contrast images and the edges of the cell can easily be identified, as shown in Figure 3. The heavy metal staining can also allow imaging of the structure of the protein structures even when the actual protein inside the metal cast is gone.

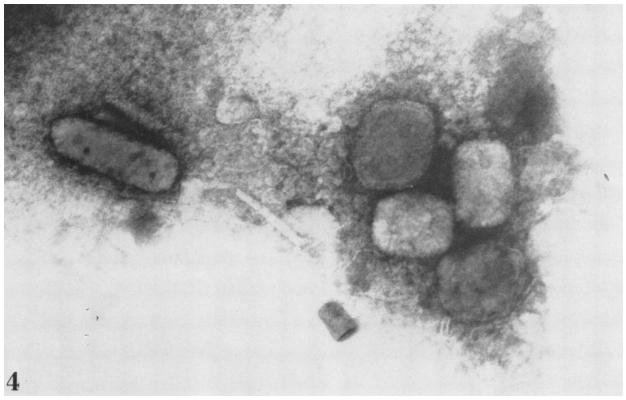


Figure 3: Smallpox infection was most commonly verified by EM image of the virus [3]

A single TEM image can provide biological information, such as smallpox virus verification (Figure 3). However, in some cases it is necessary to find the 3 dimensional shape. For example, a bacterium called *Myxococcus xanthus* was imaged using EM, and then the 3 dimensional volume of the bacterial cell was modeled (Figure 4). Through 3 dimensional volume analysis, intercellular communication tunnels were seen. This discovery led to understanding of how Bacteria communicate without other species detecting its presence, and is currently thought of as potential target for antibacterial drugs.

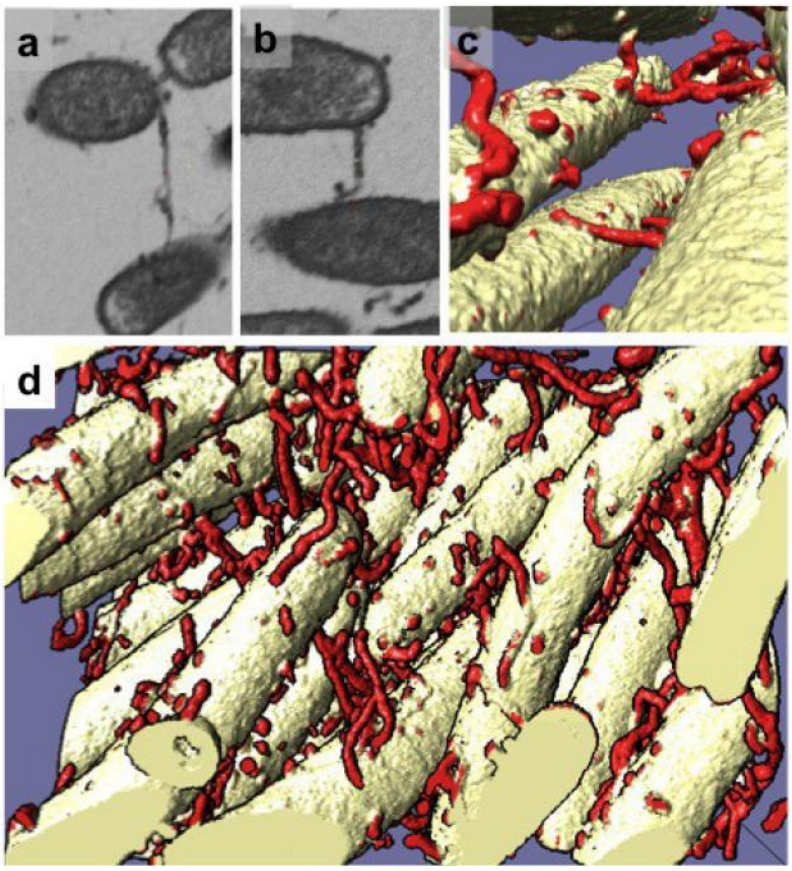


Figure 4: Myxococcus xanthus showing intercellular communication tunnels [4]

Cryo-EM : Protection from Vacuum and Electron

In 1974, Ken Taylor and Bob Glaeser successfully showed that freezing protein crystals preserved protein structures in TEM.[5] The development of Cryo Electron Microscopy (Cryo-EM) circumvented the obstacle of dehydrating samples.

Taylor and Glaeser instantaneously froze a catalase protein to prevent sample damage from ice crystal formation; a state called "vitreous ice". Using this method, and examining the electron diffraction pattern, they were able to yield a resolution of 3.4Å (figure 5). The vitreous ice layer protected the structure from electron radiation damage while keeping the cell in a fully-hydrated state.

This procedure yielded a fully hydrated, unstained, and unfixed protein that could be placed in vacuum, comparable to the protein's natural state.

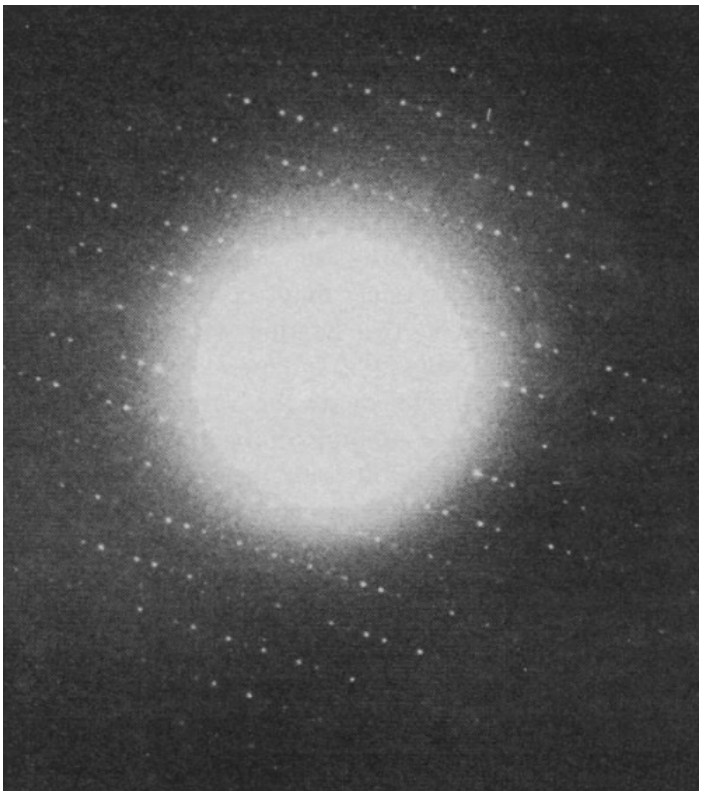


Figure 5: Electron diffraction pattern of Catalase crystal, showing 3.4Å resolution [5]

Three-Dimensional Reconstruction

The advancements of Single Particle Analysis (SPA) and Tomography allowed the 2-dimensional images to be reconstructed into a 3-dimensional volume. Both techniques utilize the Fourier slice theorem, but each technique fulfills a specific niche.

Single Particle Analysis

Single Particle Analysis (SPA) is used for the structure determination of a specific protein. SPA's general workflow (Figure 6) is the following:

1. Sample preparation: extracts the protein of interest from the source and concentrates it
2. Freezing: traps the samples in vitreous ice
3. Imaging: yields 2D projection images of various protein molecules in random orientation
4. Alignment: sorts protein molecules of the same orientation together

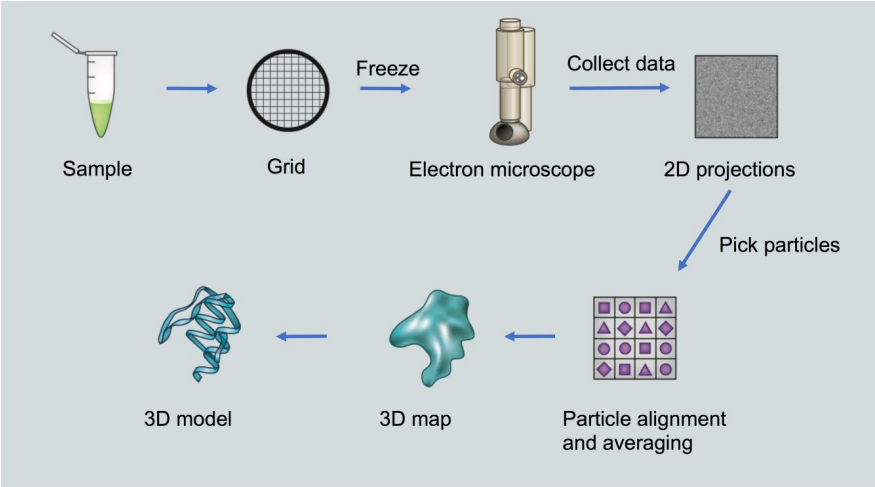


Figure 6: SPA workflow from sample preparation to reconstruction [6]

5. Averaging: averages proteins of the same orientation over one another to increase the signal to noise ratio

6. Reconstruction: averaged projections are used to reconstruct the original 3 dimensional protein (again utilizing the Fourier slice theorem)

7. Structural Analysis via Modelling: inserts the known amino acid sequence to fit into the reconstructed density map

Until the imaging step, SPA follows the standard Cryo-EM imaging sequence. However, image averaging and reconstruction results in the exceptional resolution of 3Å.

Image Averaging

Image averaging is a simple technique used to reduce random noise. It utilizes the fact that the noise location is random while the target object remains stationary. If 100 sequential images are averaged, its output overlays 100 target images on top of one another, creating a target object with 100/100 intensity on the resulting image.

However, since the noise on 100 images is random, it is unlikely the noise shares the identical location on all 100 images. Thus, the signal to noise ratio has 1/100 intensity on the resulting image compounded image. An example of this kind can be seen on Figure 7.

As previously stated, in order for image averaging to work, the target object must be identical over all image sequences. Once proteins are properly aligned, proteins imaged at identical orientations can be averaged to increase the signal to noise ratio. This also resolves the problem of loss of contrast due to lack of staining in Cryo-EM.

Single particle analysis is a strong tool for structure determination; however, averaging limits its application to a single type of molecular structure that always retains its confor-

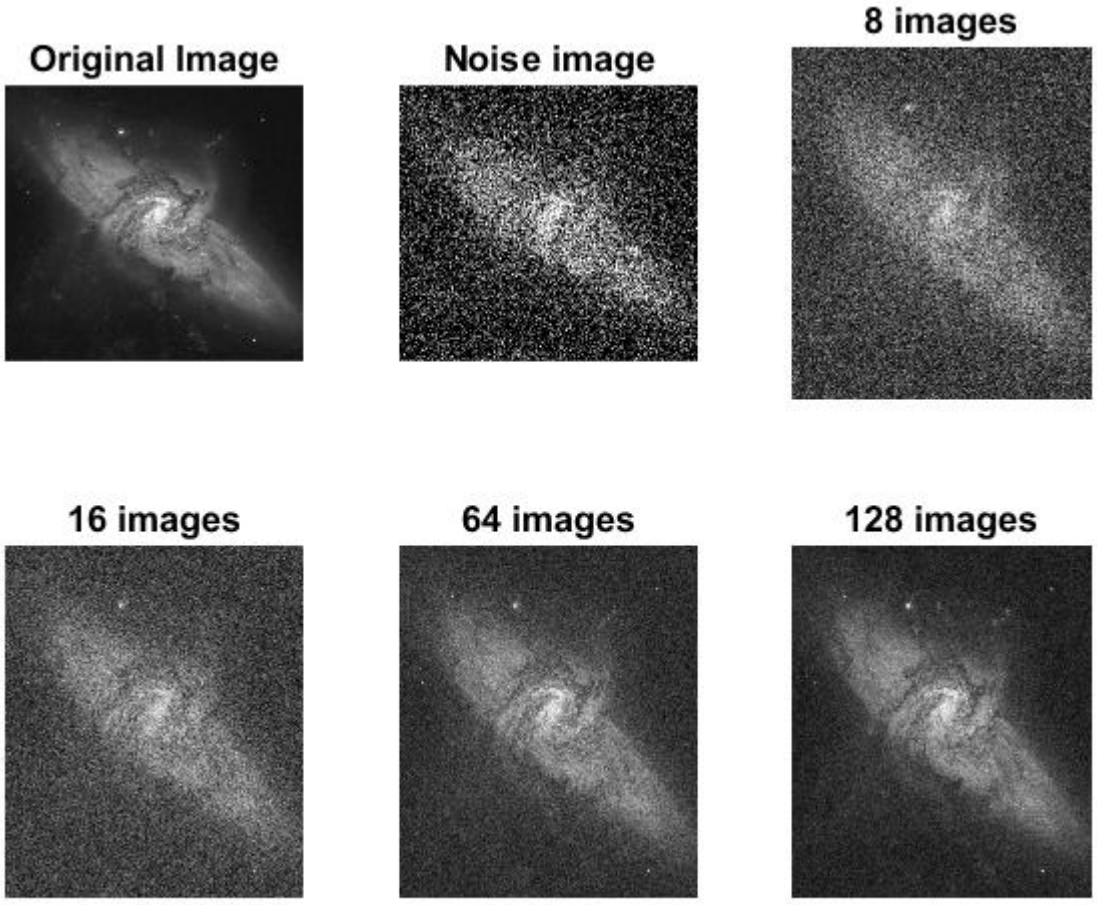


Figure 7: Random noise is diluted while the target object intensifies over image averaging [7]

mation. Unfortunately, many cellular ultrastructures are not identical in shape. In order to create a three-dimensional image of such samples, tomography can be used.

Tomography

Electron Cryo-Tomography (ECT) is used for the structure determination of a heterogeneous environment. The basic work flow of tomography is nearly identical to that of SPA. The tomography portion comes at the very last step, when the frozen sample is inserted into a Cryo-TEM. Inside, the sample is tilted at certain tilt increments, creating sequential images of the sample at varying angles. These varying two-dimensional projections are then

used to reconstruct the original three-dimensional structure.

The advantage of ECT is that it can create the 3 dimensional volume of any material placed on the grid. This allows 3D imaging of heterogeneous sample, such as cell or multiple proteins interacting with one another. Moreover, tomography avoids possible artifacts that may arise from the assumption that molecules in SPA are in identical conformation. In reality, biological samples are flexible and may be in slightly different shapes or conformations across multiple samples. Additionally, since orientations are well-known, aligning images is comparatively easier.

The greatest disadvantage of ECT is that the the sample cannot be tilted beyond certain tilt angles, approximately ± 75 degrees or so. This is because the ice layer that electrons must penetrate gets too thick as the tilt angle increases (Figure 8). This limits how high the sample can be tilted, as shown in Figure 9. This incomplete tilt angle results in incomplete 2D projections, and a void area of the reciprocal space, called the **missing wedge**, as shown in figure 10.

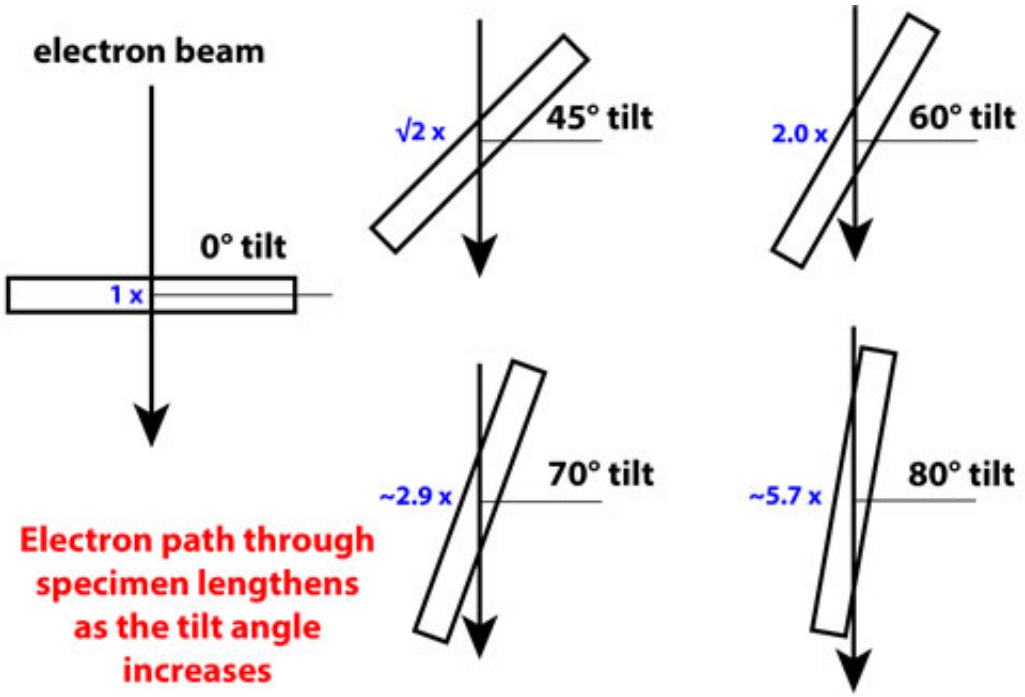


Figure 8: Excessive tilting forces electron to travel further through the ice; whereas electron can only penetrate thin layer of ice. [8]

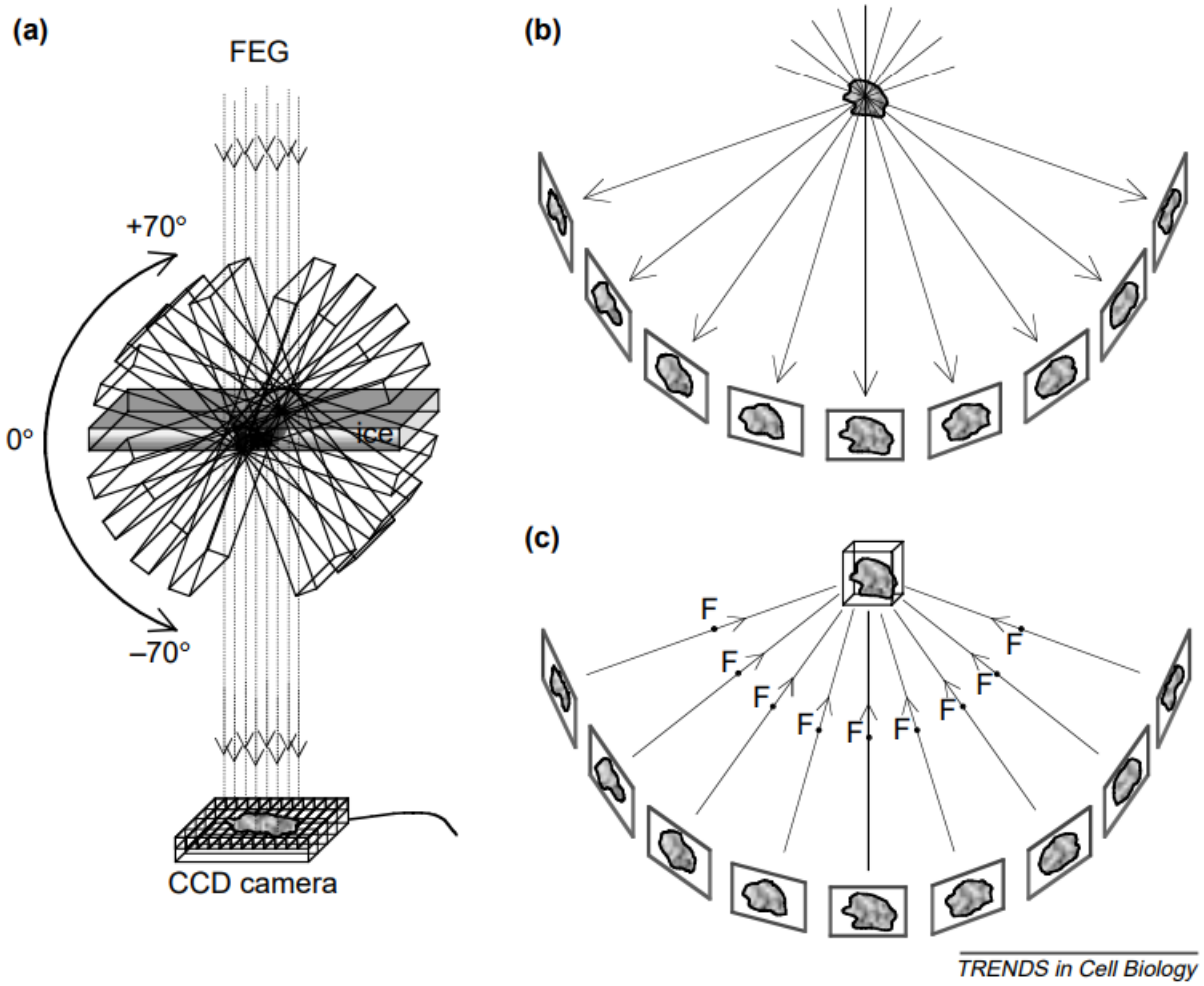


Figure 9: Original sample structure is reconstructed with imperfect 2D projection images [9]

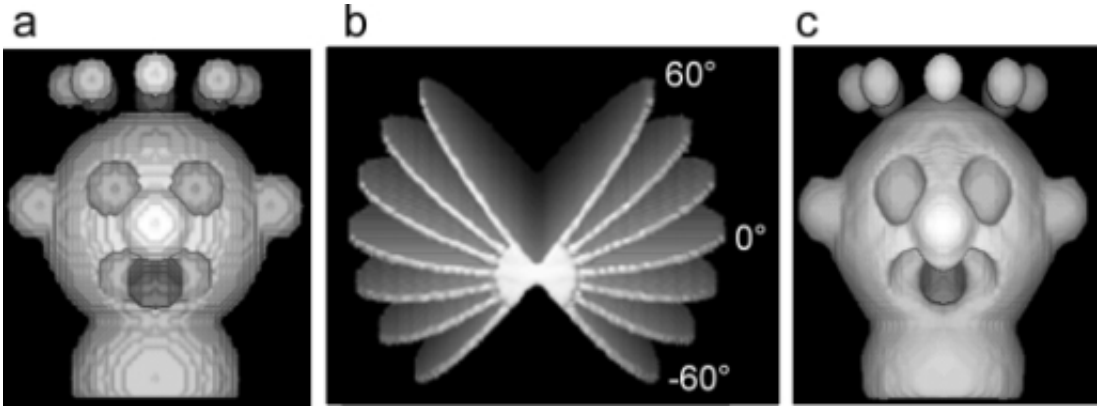


Figure 10: Tomographic projections of original object, a., is obtained only upto +-60 degrees of the tilt angle, returns reconstruction which is deformed along the location of both missing wedges. [10]

The image acquisition method of tomography inherently undersamples the sample's structure due to varying ice thickness, resulting in directionally-dependent (anisotropic) resolution. Resolution along the tilt axis is often the highest, while resolution along the axis perpendicular to the tilt axis is reduced in comparison since a discrete number of images is taken. An estimation of the best possible resolution along this axis for a reconstruction of diameter D over N images is

$$d = \frac{\pi D}{N}$$

which is derived from the geometric sampling of Fourier space.[12]

Due to the missing wedge, information along the original projection direction are stretched by an elongation factor in the back projection. This is defined by $e_{\gamma z}$: [12]

$$d_z = e_{\gamma z} d_\gamma = d_\gamma \sqrt{\frac{\alpha + sin(\alpha)cos(\alpha)}{\alpha - sin(\alpha)cos(\alpha)}}$$

where α is the maximum tilt angle. Maximal resolution along this axis is important since this projection contains the desired information. In practice, although the elongation factor cannot be eliminated, reconstructions are useful with elongation factors < 1.3. [12]

Dual-axis tomography can mitigate some of these effects that arise from lost information in the missing wedge. By rotating and tilting about another axis, a missing cone is produced. However, this is practically intensive and not always useful, so much of the work in improving tomography has gone into trying to fill in the missing wedge computationally.

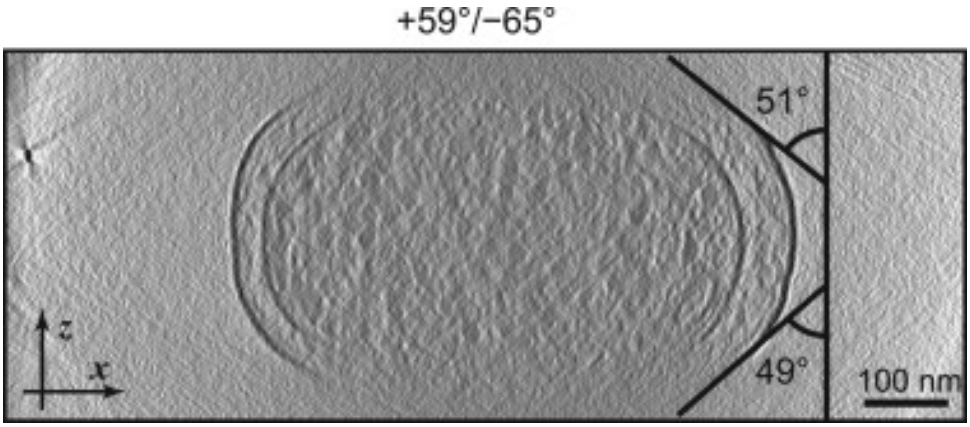


Figure 11: Cellular membrane that should be at the top and the bottom is not distinguishable from the noise as those regions are where the missing wedge occupies [11]

It is worth noting that there is not only Electron Tomography, but also soft X-ray tomography (SXT). This technique combines aspects of ECT and X-ray crystallography. Instead of using high energy "hard" x-rays, this technique uses "soft" lower energy x-rays. Therefore, they are able to penetrate through thin layers of ice but are absorbed by carbon. This results in SXT having an exceptionally low depth of field, which can create inaccurate images. Luckily, there are mathematical models that can describe and resolve these inaccuracies, partly described in the proof section of the text. The resulting image produces a tomogram, so the missing wedge remains an issue. Cryo-SXT has worse resolution than ECT, so it is not particularly useful in uncovering macromolecule structures. However, it provides good data for imaging structures that are too small to see with a light microscope and too large to see fully with an electron microscope.

Tomography is a useful tool to recover three dimensional heterogeneous structures, but the missing wedge prevents ECT from being applied to small molecules like the samples used commonly in single particle analysis. As such, each technique may be used on the appropriate biological sample. There is also a significant amount of technical and computational development in improving SPA, ET, and other EM imaging techniques. One such development combines SPA and ET by imaging multiple homologous samples at different tilt series. The tilt series gives known orientations, which can be used to constrain refinement with a reference model, which is normally a major challenge in traditional single particle analysis. This method also reduces some of the noise that is commonly an issue in ET. However, it lies under the same limitation as traditional single particle analysis in its assumption that all samples are practically identical.

References

[1] Megan Dowie. *A Nervous Encounter: scientists and artists in collaboration.* University of Arts London, University of Oxford, http://blog.nervousencounter.com/?p=215 2012

[2] Greg Robson *Electron shell 092 Uranium*. Wikimedia Commons, 2017

[3] Gary W. Long, John Noble, Jr., Frederick A. Murphy, Kenneth L. Herrmann, and Bernard Lourie. *Experience with Electron Microscopy in the Differential Diagnosis of Smallpox*. Department of Pathology, University of New Mexico School of Medicine 1970

[4] Jonathan P. Remis, Doug Wei, Amita Gorur, Marcin Zemla, Jessica Haraga, Simon Allen, H. Ewa Witkowska, J. William Costerton, James E. Berleman, and Manfred Auer *Bacterial Social Networks: Structure and composition of Myxococcus xanthus outer membrane vesicle chains*. Dept. of Biology, St. Mary's College, Life Sciences Division, Lawrence Berkeley National Laboratory. 2014

[5] Taylor KA, Glaeser RM. *Electron diffraction of frozen, hydrated protein crystals*. Division of Medical Physics, Donner Laboratory, and Lawrence Berkeley Laboratory, University of California, Berkeley 94720 1974

[6] Creative Biostructure *Cryo-EM Services*. 45-1 Ramsey Road Shirley, NY 2017

[7] Solve-Maths *Noise Removal in Digital Images by Averaging*. http://www.solvemaths.in/noise-removal-in-digital-images-by-averaging/ 2017

[8] Indiana University *Electron Tomography (ET)*. The Trustees of Indiana University 2015

[9] Alasdair C. Steven, Ueli Aebi *The next ice age: cryo-electron tomography of intact cells*. TRENDS in Cell Biology 2017

[10] Orlova EV, Saibil HR *Structural analysis of macromolecular assemblies by electron microscopy.*. Chem. Rev. 2011

[11] Colin M.Palmer, Jan Lwe *A cylindrical specimen holder for electron cryo-tomography*. Ultramicroscopy 2014

[12] Peter Ercius, Osama Alaidi, Matthew J. Rames, and Gang Ren. *Electron Tomography: A Three-Dimensional Analytic Tool for Hard and Soft Materials Research*. Advanced Materials 2015

1 Definitions for the Mathematical Model

Let $\rho : \mathbb{R}^3 \to \mathbb{R}$ be the 3D image (density) of the molecule. For the mathematical model, we shall make the following simplifying assumptions.

We assume ρ is smooth and compactly supported on interval of width L on all three axes. This allows us to recover ρ by applying the Fourier inversion theorem, which holds for continuous, absolutely integrable functions. We also assume the molecule is asymmetric, which allows us to deduce orientations of images by the method of common lines.

Our goal is to develop a method to recover ρ. We will rely on the **Fourier transform** to achieve the reconstruction.

Definition (Fourier Transform)**.** *Let $f : \mathbb{R} \to \mathbb{R}$. The Fourier transform of f, denoted by $\mathcal{F}\{f\}$ or $\hat{f}$, is another function $\hat{f} : \mathbb{R} \to \mathbb{R}$*

$$\hat{f}(\omega) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i x \omega} dx$$

In n-dimensions, the Fourier transform becomes

$$\hat{f}(\omega_{x_1}, \omega_{x_2}, ..., \omega_{x_n}) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} ... \int_{-\infty}^{\infty} f(x_1, x_2, ..., x_n) e^{-2\pi i(\omega_{x_1} x_1 + \omega_{x_2} x_2 + ... + \omega_{x_n} x_n)}$$

If f is continuous and decays to 0 as $x \to \pm\infty$ such that f is absolutely integrable, we can use the inverse Fourier transform to recover f from $\hat{f}$.

$$\mathcal{F}^{-1}\{\hat{f}\}(x) = \int_{-\infty}^{\infty} \hat{f}(w) e^{2\pi i x \omega} dx$$

Note that images are actually discretely represented by densities, not continuously. Let us also define the discrete Fourier transform.

Definition (Discrete Fourier Transform)**.** *Given an array y of N samples of f, the discrete Fourier transform of y at frequency ω is*

$$DFT(y)(\omega) = \frac{1}{N} \sum_{k=0}^{N-1} y[k] e^{-2\pi i \frac{k\omega}{N}}$$

Although we only observe samples of f, the theorems we rely on in this model utilize the continuous Fourier transform of f, so we must appeal to the following equation to reconstruct the molecule in practice.

Theorem (Poisson Summation Formula)**.** *Define the periodic summation of $\hat{f}$ with period Q to be*

$$\hat{f}_Q(\omega) = \sum_{-\infty}^{\infty} \hat{f}(\omega + mQ)$$

The Poisson summation formula gives

$$\hat{f}_Q(\omega) = \frac{1}{Q} \sum_{-\infty}^{\infty} f(\frac{m}{Q}) e^{-2\pi i \frac{\omega M}{Q}}$$

Using the Poisson Summation Formula, we may arrive at the following result.

Theorem.

$$\hat{\rho}(\frac{\omega}{L}) + \sum_{h\neq 0}(-1)^{hN}\hat{\rho}\bigg(\frac{\omega+mN}{L}\bigg) = \frac{L}{N}e^{\pi i\omega}\sum_{m=0}^{N-1}\rho\bigg(-\frac{L}{2}+\frac{mL}{N}\bigg)e^{-2\pi i\frac{\omega m}{N}}$$

$$\hat{\rho}\bigg(\frac{\omega}{L}\bigg) + \sum_{h\neq 0}(-1)^{hN}\hat{f}\bigg(\frac{\omega+mN}{L}\bigg) = Le^{\pi i\omega}DFT(y)(\omega)$$

The PSF says that the discrete Fourier transform, scaled by a constant, is equivalent to the continuous Fourier transform plus some **aliasing error**. However, since we assume $\hat{f}$ decays rapidly, the aliasing error will be negligible for large N. Note that although this is the one-dimensional version of the PSF, we will be using the three-dimensional PSF, which preserves the relation described above.

We assume we have sampled ρ enough such that N is large and the aliasing error is negligible.

2 Mathematical Model

In order to deduce the 3D structure of the molecule the set of 2D images needs to be transformed into a 3D object. Therefore, we need align these images in their correct relative orientations and reconstruct a volume.

Each image is the projection of the molecule onto a plane of some orientation. Call this plane $span\{\vec{a},\vec{b}\}$, $\vec{a},\vec{b}\in\mathbb{R}^3$. The image is the Radon (X-ray) projection of ρ onto $span\{\vec{a},\vec{b}\}$. The density is given by

$$I_{span\{\vec{a},\vec{b}\}}(x,y) = \int_{-\infty}^{\infty}\rho(\vec{a}x+\vec{b}y+\vec{c}z)dz$$

where $\vec{c}$ is unit vector orthogonal to the plane. That is, the density is given by the line integral along the line orthogonal to the plane.

Our reconstruction relies on the following theorem.

Theorem (Fourier Slice Theorem)**.**

$$\mathcal{F}\{I_{span\{\vec{a},\vec{b}\}}\} = \mathcal{F}\{\rho\}|_{span\{\vec{a},\vec{b}\}}$$

The Fourier Slice Theorem states that the Fourier transform of ρ restricted to $span\{\vec{a},\vec{b}\}$ agrees with the Fourier transform of the projection of ρ onto $span\{\vec{a},\vec{b}\}$. Thus, the Fourier transforms of the images can be built up to produce a complete Fourier transform of the density. The inverse Fourier transform can then be used to recover ρ, the 3D density of the molecule.

The images will agree on **common lines** where they intersect. For asymmetric molecules, common lines will be unique between planes. The total set of common lines allows us to deduce the correct orientation of the plane with respect to the whole molecule. This is known as the van Heel Algorithm.

2.1 Simulations Using the Mathematical Model

For this project, rather than real EM images, we have simulated two-dimensional images from an existing three-dimensional model. Using the Fourier Slice Theorem, we take the Fourier transform of ρ, restrict it to $span\vec{a}, \vec{b}$, then perform the inverse Fourier transform to produce an image. In this simulation, we know the orientations of the images we construct. However, this does not hold true for actual EM images.

2.2 Recovering Orientations by Method of Common Lines

The images we obtain have unknown orientations, but we may recover their relative orientations and reconstruct the 3D image up to a global rotation.

Let us construct a unit sphere centered at the origin in $\mathbb{R}^3$. We shall use this sphere to find the angles between planes.

Consider the intersections of three images, I_1, I_2, I_3 in space. The intersections of these lines, L_{12}, L_{13}, L_{23}, with our constructed sphere will form a spherical triangle.

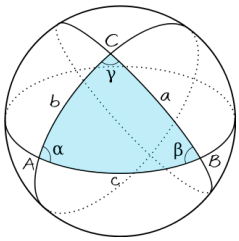


Figure 12: A spherical triangle on the unit sphere with angles α, β, γ

Denote the intersection of L_{ij} with the sphere l_{ij}. Let A be l_{12}, B be l_{23}, and C be l_{13}. Note since points A and C lie in I_1, which passes through the origin, arc b (the shortest arc between A and C on the sphere) lies in I_1 (Figure 2).

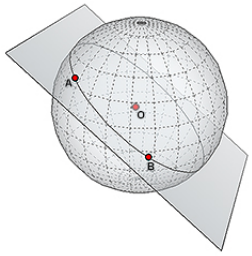


Figure 13: Intersection of plane passing through the origin and a sphere

Similarly, arc c lies in I_2 and arc a lies in I_3. Since each of the edges of this triangle lie in a unique plane, the angles of the spherical triangle are the angles between each pair of planes.

Since we have constructed a unit sphere, the angle subtended by the arc is the arc length. We may find this angle by calculating the angle between lines of intersection on each plane (using the dot product of direction vectors defining the lines). With these lengths, we can use the spherical law of cosines to determine angles α, β, and γ.

Theorem (The Spherical Law of Cosines)**.** *Let a, b, c be the sides of a triangle on the sphere and γ be the angle opposite arc c.*

$$cos(c) = cos(a)cos(b) + sin(a)sin(b)cos(\gamma)$$

Using this method, we may determine the relative orientations of any three images in space. By interchanging I_3 with $I_j, 3 < j < N$, where N is the total number of images, we may completely determine the relative orientations of all images.

By fixing I_1 to lie in the xy-plane, we can describe $\vec{a}, \vec{b}$ for all images, up to a global rotation.

2.3 3D Reconstruction by Weighted Back Projection

Given a 2D image is the projection of ρ onto a plane in space, a very rough approximation of ρ could be obtained by smearing out the densities in the direction normal to the plane. With many planes of many orientations, the intersection of the smears would produce an increasingly better approximation of ρ. Mathematically, this smearing is achieved by **convolution** with

$$l_j(x^j, y^j, z^j) = \delta(x^j, y^j) rect(z^j)$$

where $rect(z^j) = \begin{cases} 1 & -\frac{L}{2} \leq z^j \leq \frac{L}{2} \\ 0 & else \end{cases}$.

Definition (Convolution)**.** *The convolution of two functions f and g is an integral that expresses the amount of overlap as one function is shifted over the other, denoted by $f \star g$.*

$$(f \star g)(t) = \int_{-\infty}^{\infty} f(t - \tau) g(\tau) d\tau$$

Suppose we observe N images. Let

$$I_j(x, y) = \int_{-\infty}^{\infty} \rho(\vec{a}_j x + \vec{b}_j y + \vec{c}_j z) dz$$

be the Radon projection of ρ onto $span\{\vec{a}, \vec{b}\}$. Let x^j, y^j, z^j be coordinates in the basis $\{\vec{a}, \vec{b}, \vec{c}\}$.

Definition (Back Projection)**.** *The back projection of a set of images $I_1, ..., I_N$ is*

$$b(x, y, z) = \sum_{j=1}^{N} b_j(x^j, y^j, z^j)$$

We define the back projection b_j of a single image I_j to be the convolution of I_j and l_j:

$$b_j(x^j, y^j, z^j) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} I_j(x'^j, y'^j) l_j(x^j - x'^j, y^j - y'^j, z^j) dx'^j dy'^j$$

$$b_j(x^j, y^j, z^j) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \rho(x'^j, y'^j, z'^j) dz'^j l_j(x^j - x'^j, y^j - y'^j, z^j) dx'^j dy'^j$$

In order to isolate ρ, we will take the Fourier transform of b and use the **convolution theorem**.

Theorem (Convolution Theorem)**.**

$$\mathcal{F}(f \star g) = \mathcal{F}(f) \cdot \mathcal{F}(g)$$

The equation then becomes

$$\mathcal{F}\{b_j\} = \mathcal{F}\{\rho\}\mathcal{F}\{l_j\}$$

In order to approximate ρ, we can apply this result to b, taking advantage of the linearity of the Fourier transform.

$$\rho \approx \mathcal{F}^{-1}\left\{\frac{\mathcal{F}\{\sum b_j\}}{\mathcal{F}\{\sum l_j\}}\right\}$$

This recovers ρ by method of weighted back projection.

3 Statement of the Problem

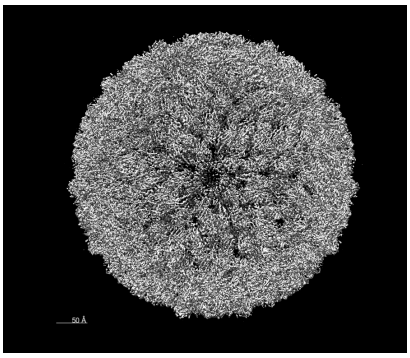
Cryo-EM produces two-dimensional projections of a three-dimensional biological sample. However, in order to understand the function of and chemical mechanisms used by a protein, we must decipher its three-dimensional conformation.

Given $\{I_1, I_2, ..., I_N\}$, the Radon projections of ρ, we will recover ρ by the method of weighted back projection.

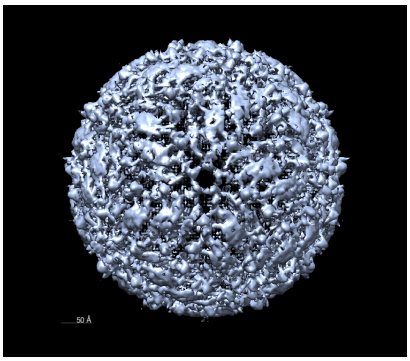
4 Introduction to the Computer Simulation

With the mathematical model established, we proceeded to translate the model into a functional program that simulates a typical EM Reconstruction pipeline. The conventional protocol begins with thousands of 2D EM images and, aside from predictive models and other similar structures that have already been reconstructed, there is little information about the 3D reconstructed model. For this simulation, the 3D reconstructed model is known and is used as a benchmark for the simulation result. Moreover, a number of assumptions and simplifications are made in various steps of the simulation.

The simulation outlined in this paper is based on Zika virus data downloaded from the EM Data Bank. The original data is a MAP file and has been down-sampled to 153x153x153 and saved as a MRC file.



(a) Original Zika Virus Data



(b) Down-sampled Zika Virus Data

Figure 14: Zika Virus EM Data

5 Fourier Slice Theorem

Using the Fourier Slice Theorem, 2D EM images were simulated with a projection algorithm that takes a 3D volume, defined here as ρ, and a $span\{\vec{a}, \vec{b}\}$ as its input and outputs the projection as a 2D image. The first step of the algorithm is to take the Fourier transform of ρ using the n-dimensional discrete Fourier transform (`fftn`) routine from NumPy (`np`), a package for scientific computing with Python. Note that ρ is greater than 1 or 2 dimensions so we use the n-dimensional routine. In addition, a fast algorithm, the fast Fourier transform, is used for efficient computation.

The output of `fftn`(ρ) is a 3D matrix of frequencies defined as a spectrum containing elements in "standard order". The definition of "standard order" is that for `A = fft(a, n)` where `a` is a linear array, `A[0]` contains the zero-frequency term, `A[1:n/2]` contains the positive frequency terms and `A[n/2+1:]` contains the negative frequency terms in the order of decreasingly negative frequencies. For an even number of input points, `A[n/2]` represents both positive and negative frequency and for an odd number of input points, `A[(n-1)/2]` contains the largest positive frequency, while `A[(n+1)/2]` contains the largest negative frequency. However, this "standard order" does not make sense for images, and the results need to be shifted using `fftshift` so that the zero-frequency component is centered at the spectrum. We define the result of the discrete Fourier transformed and shifted ρ as $\hat{\rho}$.

In addition to shifting the spectrum, it is also necessary to convert the values from the discrete Fourier transform into values from the continuous Fourier transform prior to sampling. This relation is shown by the Poisson Summation Formula (defined in the previous section). This is achieved by multiplying the results by a scaling factor. Note that we make the assumption that the aliasing error is negligible.

After obtaining $\hat{\rho}$, a 2D sampling grid needs to be generated in order to sample the volume along a plane. The sampling grid is generated by Numpy's `meshgrid` routine. Note that the sampling grid should be at least the same lengths as the 3D volume. There are two

options that can be taken at this point: (1) rotate the sampling grid so that it $spans\{\vec{a}, \vec{b}\}$ or (2) rotate $\hat{\rho}$ and keep the sampling grid in its original basis. This simulation follows the first option of rotating the sampling grid. Once the orientation of the grid and $\hat{\rho}$ are set, sampling is done using the function `RegularGridInterpolator` (`RGI`) from SciPy. Note that additional parameters such as `bounds_error=False` and `fill_value=0` need to be inputted so that corners that protrude pass the boundaries of $\hat{\rho}$ will not cause an error and that all the values outside of the boundaries will be set to 0. This parameter does not apply to all data because not all data are like the Zika virus which is spherical and is consistently confined in the the volume. For data that involves structure that isn't as uniform and symmetrical as Zika virus, additional measures need to be taken. Since these are beyond the scope of this topic, it will not be discussed in this paper.

Lastly, prior to taking the inverse Fourier transform, the result needs to be scaled again, this time with only two axes and N^2, where N is the length of one side of the image. Once the results have been scaled, it is also necessary to run the routine `ifftshift` which is the inverse of the earlier shift. After shifting, the inverse fast Fourier transform can be applied to generate a 2D array that contains both imaginary and real elements. For this simulation, the imaginary part is negligible so only the real part is taken to generate a single image I_j.

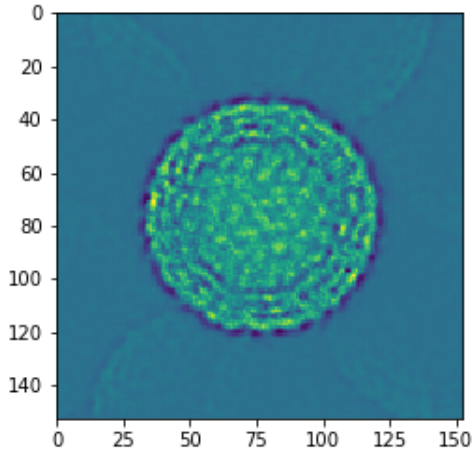


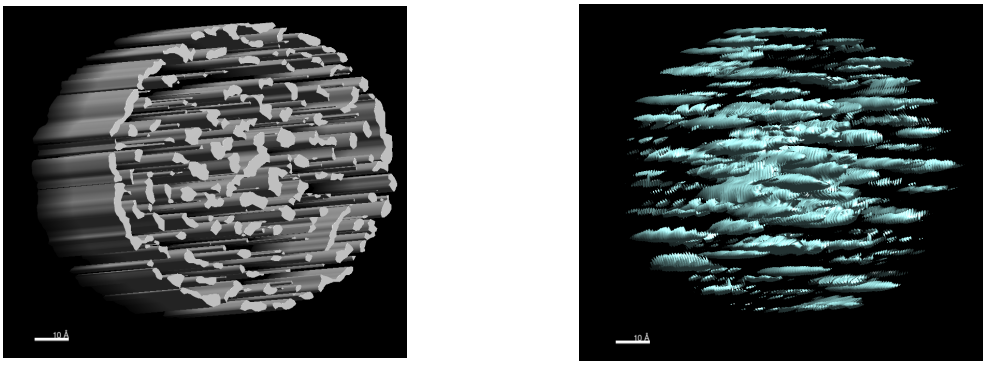
Figure 15: Projection Image using Fourier Slice Theorem. (Ghosting artifact observed is likely due to down-sampling)

6 Back Projection

Now that 2D images can be generated, back projection is conventionally the first step in reconstruction assuming that the images have been processed and are ready for reconstruction. The input to the back projection algorithm is an image and the output should be a

3D volume with the same length as the input image. The first step of back projection is to take the Fourier transform of an image I_j and shift it using `ifftshift`. Next is to generate a N by N mesh grid for scaling where the scaled image is defined as $\hat{I_j}$.

Next, we compute $\mathcal{F}\{l_j\}$, defined to be $\hat{l_j} = \mathcal{F}\{l_j\} = sinc(\omega z)$ where ωz is a linear array defined on the same frequency range as $\hat{I_j}$. Once $\hat{I_j}$ and $\hat{l_j}$ are generated, the Numpy tiling routine `tile` is used so the dimensions of the two results match and can be multiplied together to generate $\hat{b_j}$. For an unweighted back projection, we can descale $\hat{b_j}$ and apply the inverse Fourier transform to generate b_j. For weighted back projection, we divide $\hat{b_j}$ by the sum of $\hat{l_j}$, descaled, and apply the inverse Fourier transform to generate a weighted b_j.



(a) Unweighted Back Projection

(b) Weighted Back Projection

Figure 16: Weighted vs. Unweighted Back Projectsion

7 Reconstruction

Now that the main algorithms are defined, the reconstruction pipeline can be illustrated. For this simulation, reconstruction is defined to take the 3D volume data ρ and a user specified input for the number of projections to take and use to reconstruct the volume. When given these inputs, it will generate random $span\{\vec{a}, \vec{b}\}$ for every projection, run the back projection on that image, and sample the projection along its corresponding $span\{\vec{a}, \vec{b}\}$ and store it in an array. Once all the back projections are completed, it will sum up the projections and divide by the sum of all $\hat{l_j}$. Lastly, it will shift the result and apply the inverse Fourier transform. The real part of this final result will be the complete reconstruction.

8 Common Lines

In reality, the beginning of reconstruction consists only of 2D images without any information regarding their orientation in space. Hence, we use the common lines technique to solve for their orientations. For k images, we iterate through the images in triplets to solve for each corresponding $\vec{a_i}$, $\vec{b_i}$. Let ℓ_{ij} denote the common line between planes i and j *in*

the image I. For each triplet, we first find the common line pairs ℓ_{ij}, ℓ_{jk}. Then we find the angles between the common lines to solve for a, b, c where c is the angle between ℓ_{12}, ℓ_{13}, b is the angle between ℓ_{21}, ℓ_{23}, and a is the angle between ℓ_{31}, ℓ_{32}. This will give us the arc lengths of the spherical triangle described in the previous section. Next, we use the Spherical Law of Cosines to solve for α, β, and γ, which denote the angles between the images. We then choose unit vectors $\hat{v}_{12}$, $\hat{v}_{13}$ along L_{12}, L_{13}. Note that these are easily found since they lie in I_1, which we have set to be the xy-plane.

We set $F_2' = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$. We then rotate the columns of F_2' about the z-axis such that ℓ_{12}, L_{12} line up. The resulting columns are rotated about L_{12} by α, the angle between I_1 and I_2. These operations will result in the matrix corresponding to the linear transformation that sends I_2 from the xy-plane to the correct orientation with respect to I_1.

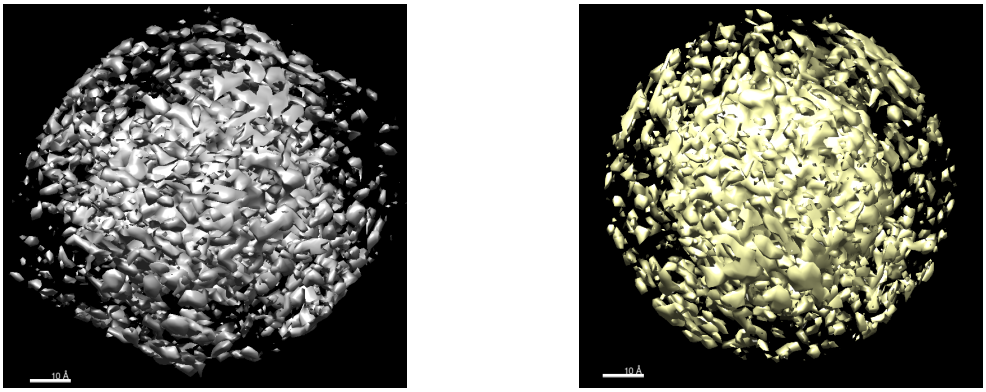
We have $\vec{a_1}$, $\vec{b_1}$, so we have L_{12} and $\vec{a_2}$, $\vec{b_2}$, which gives us L_{23}. With this, we can find $\vec{a_3}$, $\vec{b_3}$ by a matrix representing the linear transformation that sends ℓ_{13} to L_{13} and ℓ_{32} to L_{23} in space. The corresponding matrix will be

$$\begin{bmatrix} L_{13} & L_{23} \end{bmatrix} \begin{bmatrix} \ell_{13} & \ell_{32} \end{bmatrix}^{-1}$$

. The columns of this matrix will be $\vec{a_3}, \vec{b_3}$.

This procedure for I_3 may be repeated for any I_j. Thus, we can find the relative orientations for all images.

9 Results



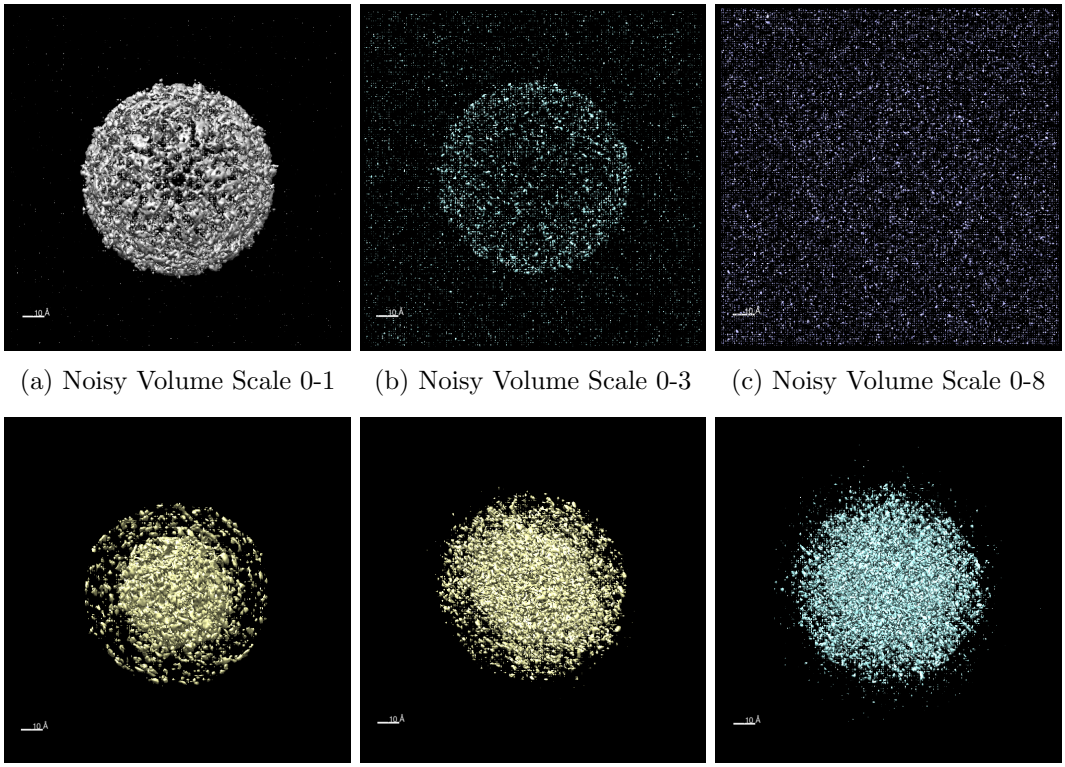
(a) Reconstruction with 5 images

(b) Reconstruction with 10 images

Figure 17: Reconstruction with different number of images. (a) An example of reconstruction with only 5 images. (b) An example reconstruction with 10 images.

The resolution of the reconstructed model depends on the number of images that are used as well as the resolution and quality of the images. From Figure 4, it can be seen

that with only 5 images, the back projection can only capture a limited number of orientations, producing a deformed sphere that has density protruding outside of the surface. In contrast, reconstruction with a higher number of images allow for a smoother and higher resolution model. Note that the number of images required for a decent reconstruction for the simulation is drastically lower than the standard scale because of the down-sampled data as well as the highly symmetrical structure of the Zika Virus. Moreover, the typical raw EM data are very noisy and often have to be processed with several layers of filters and adjustments.



(a) Noisy Volume Scale 0-1 (b) Noisy Volume Scale 0-3 (c) Noisy Volume Scale 0-8

(d) Reconstruction **(a)** (e) Reconstruction **(b)** (f) Reconstruction **(c)**

Figure 18: Reconstruction with different Signal to Noise Ratio

Here, we simulate the effects of adding various degrees of noise to our data. The noise is generated with the routine `np.random.normal`, which generates a volume of the same dimension with randomly assigned values within a specified scale. The noise is then added to the original data to generate a new volume with noise. The varying parameter in this noise simulation is the scale chosen when generating the noisy volume. The scales used in Figure 5 are: 1, 3, and 8. The value 8 was chosen by the maximum signal value in the original data. It can be seen that the reconstruction algorithm is able to de-noise most of the data through its weighted back projection. This is largely due to the sample size and

symmetry of the data. If the sample was not as symmetrical, the result would have a much higher noise.

Although this simulation makes many assumptions, it attempts to capture the core aspects of reconstruction. It covers fundamental topics such as the Fourier Slice Theorem with results that can be visually assessed. It also addresses projection and back projection which are fundamental building blocks of reconstruction. These techniques and concepts apply not only to single particle analysis, but also to nearly all the imaging techniques in the EM field.

10 Fourier Transform Exercises

1. Compute the Fourier series for $f(x) = x^2$ periodic on $[-\pi, \pi]$, and use it to derive the identity

$$\sum_{n=1}^{\infty} \frac{(-1)^n}{n^2} = -\frac{\pi^2}{12}$$

Fourier Series of $f(x) = x^2$

$$\begin{aligned} f(x) &= \frac{\pi^2}{3} + \sum_{n=1}^{\infty} \frac{4}{n^2} cos(n\pi) cos(nx) \\ &= \frac{\pi^2}{3} + \sum_{n=1}^{\infty} \frac{4}{n^2} (-1)^n cos(nx) \\ &= \frac{\pi^2}{3} + 4 \sum_{n=1}^{\infty} \frac{(-1)^n}{n^2} cos(nx) \end{aligned}$$

$$\begin{aligned} f(0) = 0^2 = 0 &= \frac{\pi^2}{3} + 4 \sum_{n=1}^{\infty} \frac{(-1)^n}{n^2} \\ -\frac{\pi^2}{3} &= 4 \sum_{n=1}^{\infty} \frac{(-1)^n}{n^2} \\ -\frac{\pi^2}{12} &= \sum_{n=1}^{\infty} \frac{(-1)^n}{n^2} \end{aligned}$$

2. Suppose f and g are two functions $\mathbb{R} \to \mathbb{R}$ with $|\int_{-\infty}^{\infty} f(x)dx| < \infty$ and $|\int_{-\infty}^{\infty} g(x)dx| < \infty$. The convolution of f and g, denoted $f \star g$, is given by

$$(f \star g)(z) = \int_{-\infty}^{\infty} f(x) g(z - x) dx$$

if $F(f)$ is the Fourier Transform of f, prove that

$$F(f \star g) = F(f) \cdot F(g)$$

Proof. Let $(f \star g)(\mathrm{z}) = h(\mathrm{z})$

$$F(h(t)) = \int_{-\infty}^{\infty} h(z)e^{-2\pi itz}dz$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x)g(z-x)dxe^{-2\pi itz}dz$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x)g(z-x)e^{-2\pi itz}dxdz$$

$|\int_{-\infty}^{\infty} f(x)dx| < \infty$ and $|\int_{-\infty}^{\infty} g(x)dx| < \infty$ so by Fubini's Theorem,

$$F(h(t)) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x)g(z-x)e^{-2\pi itz}dzdx$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(z-x)e^{-2\pi itz}dzf(x)dx$$

let $y = z - x$

$$F(h(t)) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(y)e^{-2\pi it(x+y)}dyf(x)dx$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(y)e^{-2\pi ity}dyf(x)e^{-2\pi itx}dx$$

$$= \int_{-\infty}^{\infty} g(y)e^{-2\pi ity}dy\int_{-\infty}^{\infty} f(x)e^{-2\pi itx}dx$$

$$= F(f) \cdot F(g)$$

□

3.

$$(f \star g)(z) = \int_{-\infty}^{\infty} f(x)g(z-x)dx$$

Let f, g be complex-valued functions defined on the set $S = \{-N, -N+1, ..., N-1, N\}$ where $N \in \mathbb{Z}$. The discrete convolution of f, g is

$$(f \star g)(z) = \sum_{n=-N}^{N} f(n)g(z-n)$$

For each z, the algorithm has the complexity of $O(N)$, hence for all $z \in S$ the complexity of the discrete convolution is $O(N^2)$.

$$\begin{aligned} F(f \star g) &= F(f) \cdot F(g) \\ (f \star g) &= F^{-1}(F(f \star g)) \\ &= F^{-1}(F(f)) \cdot F^{-1}(F(g)) \end{aligned}$$

We know that the Fast Fourier Transform for 1 dimension has complexity $NlogN$. Similar to the Inverse Discrete Fourier Transform, the Inverse Fast Fourier Transform has the same complexity. Hence, with two functions f, g the complexity is:

$$= O(NlogN) + O(NlogN) = O(NlogN)$$

11 Proofs Related to Tomography

The following proofs are based on the mathematics introduced in the textbook "Computational Methods for Three-Dimensional Microscopy Reconstruction" by Gabor T. Herman and Joachim Frank.

12 Proofs for Chapter 8.6.2, "Computational Methods for Three-Dimensional Microscopy Reconstruction"

These proofs regard reconstruction of soft Cryo X-ray Tomography. This is a tomography method that uses X-rays instead of electrons. Images that are created by SXT have a limited depth of field and suffer inaccuracies that need to be corrected for mathematically. The following proofs show the ability to correct for the signal to noise ration using the Image Formation Model.

Assuming the sample is completely focused, the Poisson Summation Formula can be treated independently from the z axis. Below are the proofs of Eq. 8.49 and Eq. 8.50 in Chapter 8.6.2, The EM Tomography Case: Image Formation Model with z-Constant Point Spread Function. Note that μ is the 3 dimensional distribution of the intensity absorption coefficients.

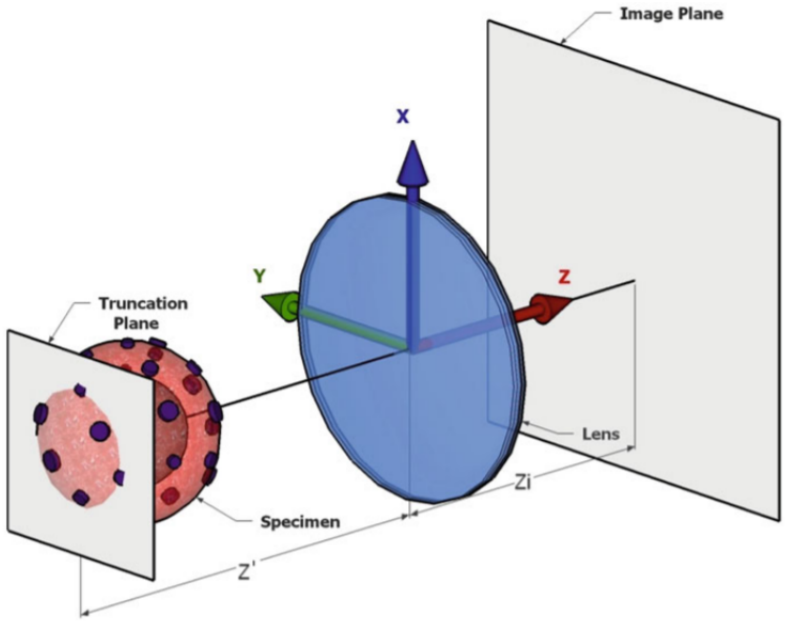


Figure 19: Visual representation of the axes of the sample and the microscope relationship and the soft X-ray tomogram

In order to recover these coefficients, we will prove the Image Formation Model with z-Dependent Point Spread Function described in section 8.6.1.

The Image Formation Model can be written as:

$$I^{z_i}(x,y,z_A) = I^{z_i}(x,y,z_B) - \int_{z_B}^{z_A} \left[\mu(x,y,z^{'}) I_g^{z_i}(x,y,z_B) e^{-\int_{z_B}^{z^{'}} \mu(x,y,\xi)d\xi}\right] \otimes h(x,y,z^{'})dz^{'}$$

This is a function to describe the illumination of the sample using the Frensel approximation, which models a z-constant model. Technically, the Point Spread Function(PSF) would slightly depend on the z-plane; however assuming it constant can still give an accurate approximation. Also, we can place $I_g^{z_i}(x,y,z_B)$ out of the integral as it is independent of z.

$$\begin{aligned}
I^{z_i}(x,y,z_A) &= I^{z_i}(x,y,z_B) - I_g^{z_i}(x,y,z_B) \int_{z_B}^{z_A} [\mu(x,y,z') e^{-\int_{z_B}^{z'} \mu(x,y,\xi)d\xi}] dz' \otimes h(x,y) \\
&= I_g^{z_i}(x,y,z_B) \otimes h(x,y) - I_g^{z_i}(x,y,z_B) \int_{z_B}^{z_A} [\mu(x,y,z') e^{-\int_{z_B}^{z'} \mu(x,y,\xi)d\xi}] dz' \otimes h(x,y) \\
&= \left[I_g^{z_i}(x,y,z_B) - I_g^{z_i}(x,y,z_B) \int_{z_B}^{z_A} [\mu(x,y,11z') e^{-\int_{z_B}^{z'} \mu(x,y,\xi)d\xi}] dz'\right] \otimes h(x,y) \\
&= I_g^{z_i}(x,y,z_B)\left[1 - \int_{z_B}^{z_A} [\mu(x,y,z') e^{-\int_{z_B}^{z'} \mu(x,y,\xi)d\xi}] dz'\right] \otimes h(x,y)
\end{aligned}$$

Let $F(x,y,x') = -\int_{z_B}^{z'} \mu(x,y,\xi)d\xi$
By second FTC, $\frac{\partial F(x,y,z')}{\partial z'} = -\mu(x,y,z')$

$$
\begin{aligned}
I^{z_i}(x,y,z_A) &= I_g^{z_i}(x,y,z_B)\left[1 + e^{-\int_{z_B}^{z'} \mu(x,y,\xi)d\xi}\right]_{z_B}^{z_A} \otimes h(x,y) \\
&= I_g^{z_i}(x,y,z_B)\left[1 + e^{-\int_{z_B}^{z_A} \mu(x,y,\xi)d\xi} - e^{-\int_{z_B}^{z_B} \mu(x,y,\xi)d\xi}\right] \otimes h(x,y) \\
&= I_g^{z_i}(x,y,z_B)\left[e^{-\int_{z_B}^{z_A} \mu(x,y,\xi)d\xi} + 1 - 1\right] \otimes h(x,y) \\
&= \left[I_g^{z_i}(x,y,z_B)e^{-\int_{z_B}^{z_A} \mu(x,y,\xi)d\xi}\right] \otimes h(x,y) \\
I^{z_i}(x,y,z_A) \otimes h^{-1}(x,y) &= \left[I_g^{z_i}(x,y,z_B)e^{-\int_{z_B}^{z_A} \mu(x,y,\xi)d\xi}\right] \otimes h(x,y) \otimes h^{-1}(x,y) \\
I^{z_i}(x,y,z_A) \otimes h^{-1}(x,y) &= \left[I^{z_i}(x,y,z_B)e^{-\int_{z_B}^{z_A} \mu(x,y,\xi)d\xi}\right] \otimes h^{-1}(x,y) \\
I^{z_i}(x,y,z_A) \otimes h^{-1}(x,y) &= \left[I^{z_i}(x,y,z_B) \otimes h^{-1}(x,y)\right] e^{-\int \lim_{z_B}^{z_A} \mu(x,y,\xi)d\xi} \\
e^{-\int_{z_B}^{z_A} \mu(x,y,\xi)d\xi} &= \frac{I^{z_i}(x,y,z_A)h^{-1}(x,y)}{I^{z_i}(x,y,z_B) \otimes h^{-1}(x,y)} \\
\int_{z_B}^{z_A} \mu(x,y,\xi)d\xi &= -ln\left(\frac{I^{z_i}(x,y,z_A) \otimes h^{-1}(x,y)}{[I^{z_i}(x,y,z_B) \otimes h^{-1}(x,y)}\right)
\end{aligned}
$$

This equation shows that the ratio of signal $I^{z_i}(x,y,z_A)$ to noise $I^{z_i}(x,y,z_B)$ can be corrected with the inverse optical transformation $h^{-1}(x,y)$.

13 Proofs for Chapter 8.6.4, "Comparison Between EM and X-Ray Tomography in Fourier Space"

Similar to X-Ray crystallography, we can find the diffraction pattern in Fourier space. In the case of electron microscopy, we can analyze how projections are related to the Fourier transform of the sample. The Contrast Transfer function can mathematically describe how aberrations of the microscope affect the image of the sample. We can describe CTP as

$$c_z = -kcos(2\pi a_s z + b_s)$$

where $a_s = \frac{1}{2\lambda s^2}$ and $b_s = -2\pi(\frac{C_s\lambda^3 s^4}{4} - \frac{z_0\lambda s^2}{2}) - cos^{-1}Q$ and $s = \sqrt{{f_x}^2 + {f_y}^2}$

For the 1D Fourier Transform of c_z, $\hat{c}(\xi)$,

$$\begin{aligned}
\hat{c}(\xi) &= \int_{-\infty}^{\infty} c_z e^{-2\pi i z\xi} dz \\
&= \int_{-\infty}^{\infty} \big\{ -k\cos(2\pi a_s z + b_s)\big\} e^{-2\pi i z\xi} dz \\
&= -k\int_{-\infty}^{\infty} \big\{ \frac{e^{i(2\pi a_s z + b_s)}}{2} + \frac{e^{-i(2\pi a_s z + b_s)}}{2} \big\} e^{-2\pi i z\xi} dz \\
&= -\frac{k}{2}\int_{-\infty}^{\infty} \big\{ e^{i(2\pi a_s z + b_s) - 2\pi i z\xi} + e^{-i(2\pi a_s z + b_s) - 2\pi i z\xi} \big\} dz \\
&= -\frac{k}{2} e^{ib_s}\int_{-\infty}^{\infty} e^{2\pi i a_s z - 2\pi i z\xi} dz - \frac{k}{2} e^{-ib_s}\int_{-\infty}^{\infty} e^{-2\pi i a_s z - 2\pi i z\xi} dz \\
&= -\frac{k}{2} e^{ib_s}\int_{-\infty}^{\infty} e^{-2\pi i z(\xi - a_s)} dz - \frac{k}{2} e^{-ib_s}\int_{-\infty}^{\infty} e^{-2\pi i z(\xi + a_s)} dz \\
&= -\frac{k}{2}\big[e^{ib_s}\delta(\xi - a_s) + e^{-ib_s}\delta(\xi + a_s) \big]
\end{aligned}$$

$$\hat{c}(f_x, f_y, f_z) = -\frac{k}{2}\big[e^{ib_s}\delta(f_z - a_s) + e^{-ib_s}\delta(f_z + a_s) \big]$$

Using the model for Fourier space, we can simplify in the case of cryo-EM:

$$\begin{aligned}
\hat{p}(f_x, f_y) &= \int \hat{v}(f_x, f_y, f_z)\hat{c}(f_x, f_y, -f_z) df_z \\
&= \int \hat{v}(f_x, f_y, f_z)\Big[-\frac{k}{2}\big[e^{ib_s}\delta(-f_z - a_s) + e^{-ib_s}\delta(-f_z + a_s) \big] \Big] df_z \\
&= -\frac{k}{2}\int \hat{v}(f_x, f_y, f_z) e^{ib_s}\delta(-f_z - a_s) df_z - \frac{k}{2}\int \hat{v}(f_x, f_y, f_z) e^{-ib_s}\delta(-f_z + a_s) \big] df_z \\
&= -\frac{k}{2} e^{ib_s}\hat{v}(f_x, f_y, -a_s) - \frac{k}{2} e^{-ib_s}\hat{v}(f_x, f_y, a_s) \\
&= -\frac{k}{2}\big[e^{ib_s}\hat{v}(f_x, f_y, -a_s) + e^{-ib_s}\hat{v}(f_x, f_y, a_s) \big]
\end{aligned}$$

$$\begin{aligned}
p'(x, y) &= I^{Z_i}(x, y, z_B) - I^{Z_i}(x, y, z_A) \\
&= \int_{z_A}^{z_B} \big[\mu(x, y, z') I_g^{z_i}(x, y, z_B) e^{-\int_{z_B}^{z'} \mu(x, y, \xi) d\xi} \big] \otimes h(x, y, z') dz' \\
v'(x, y, z) &= \mu(x, y, z') I_g^{z_i}(x, y, z_B) e^{-\int_{z_B}^{z'} \mu(x, y, \xi) d\xi} \\
p'(x, y) &= \int_{z_A}^{z_B} v'(x, y, z) \otimes h(x, y, z') dz
\end{aligned}$$

By the convolution theorem,

$$\hat{p}(f_x, f_y) = \int \hat{v_z}(f_x, f_y)\mathcal{H}(f_x, f_y)dz$$

By the 2D Fourier transform of the projection p in the plane (x,y) (Equation 8.53),

$$\hat{p}(f_x, f_y) = \int \hat{v_z}(f_x, f_y)c_z(f_x, f_y)dz$$

$$\begin{aligned}
\mathcal{H}(f_x, f_y) &= c_z(f_x, f_y) \\
\mathcal{F}_{x,y}\big\{h(x, y)\big\} &= c_z(f_x, f_y) \\
\mathcal{F}_z\Big\{\mathcal{F}_{x,y}\big\{h(x, y)\big\}\Big\} &= \hat{c_z}(f_x, f_y, f_z)
\end{aligned}$$

While x, y and z are independent,

$$\hat{c_z}(f_x, f_y, f_z) = \mathcal{F}_{x,y,z}\big\{h(x, y)\big\}$$

Reference Code

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import seaborn as sns
import mrcfile
import math
from scipy import fftpack, interpolate, linalg, ndimage
%matplotlib inline
```

In [3]:
```
def project_fst(rho, a, b):
    """
    RGI method
    """
    rho_hat = np.fft.fftshift(np.fft.fftn(rho))
    N = len(rho)

    etax, etay = (np.meshgrid(np.arange(-(N-1)/2 , (N-1)/2 + 1, dtype=int),
                              np.arange(-(N-1)/2, (N-1)/2 + 1, dtype=int),
                              indexing='ij'))
    etax = etax[..., np.newaxis]
    etay = etay[..., np.newaxis]

    grid = etax * a + etay * b #200x200x3

    wx, wy, wz = (np.meshgrid(np.arange(-(N-1)/2 , (N-1)/2 + 1, dtype=int),
                              np.arange(-(N-1)/2 , (N-1)/2 + 1, dtype=int),
                              np.arange(-(N-1)/2 , (N-1)/2 + 1, dtype=int)))

    rho_hat = np.exp(np.pi * 1j * (wx + wy + wz)) / N**3 * rho_hat

    x = np.arange(-(N-1)/2, (N-1)/2+1)
    y = np.arange(-(N-1)/2, (N-1)/2+1)
    z = np.arange(-(N-1)/2, (N-1)/2+1)


    rho_hat_f = (interpolate.RegularGridInterpolator(points=[x, y, z],
                                                     values=rho_hat,
                                                     bounds_error=False,
                                                     fill_value=0))
    result = rho_hat_f(grid)

    result = np.exp(-np.pi * 1j * (etax[..., 0] + etay[..., 0]))* N**2 * result
    image = np.fft.ifftn(np.fft.ifftshift(result))

    return image.real
```

In [4]:
```
def back_projection(I):
```

```
        # compute image_hat
        N = len(I)
        image_hat = np.fft.fftshift(np.fft.fftn(I))
        # scale
        etax, etay = (np.meshgrid(np.arange(-(N-1)/2 , (N-1)/2 + 1, dtype=int),
                                  np.arange(-(N-1)/2, (N-1)/2 + 1, dtype=int),
                                  indexing='ij'))
        image_hat = np.exp(np.pi * 1j * (etax + etay)) * N**2 * image_hat

        # compute F{l_j}
        wz = np.arange(-(N-1)/2, (N-1)/2+1)
        l_j_hat = np.sinc(wz)

        # tiling
        l_j_hat_tiled = np.tile(l_j_hat, (N, N, 1))
        I_j_hat = np.tile(image_hat[..., np.newaxis], (1, 1, N))

        b_j_hat = I_j_hat * l_j_hat
        # scale
#       w_x, w_y, w_z = np.meshgrid(np.arange(-(N-1)/2 , (N-1)/2 + 1, dtype=int),
#                                   np.arange(-(N-1)/2 , (N-1)/2 + 1, dtype=int),
#                                   np.arange(-(N-1)/2 , (N-1)/2 + 1, dtype=int))
#       b_j_hat = np.exp(-np.pi * 1j * (w_x + w_y + w_z)) / N**3 * b_j_hat

        # inverse fft
#       b_j = np.fft.ifftn(np.fft.ifftshift(b_j_hat))
        return (b_j_hat, np.sum(l_j_hat))
```

In [31]:
```
def reconstruction(rho, k_images):

    sum_b_j_hat = 0
    sum_l_j_hat = 0

    for i in np.arange(k_images):

        a = 2*np.random.rand(3)-1
        b = np.cross(a, 2*np.random.rand(3)-1)
        a = a/np.linalg.norm(a)
        b = b/np.linalg.norm(b)

        image = project_fst(rho, a, b)
        bp_result = back_projection(image)
        b_j_hat = bp_result[0]
        l_j_hat = bp_result[1]

        N = len(image)

        # rotated coordinates
```

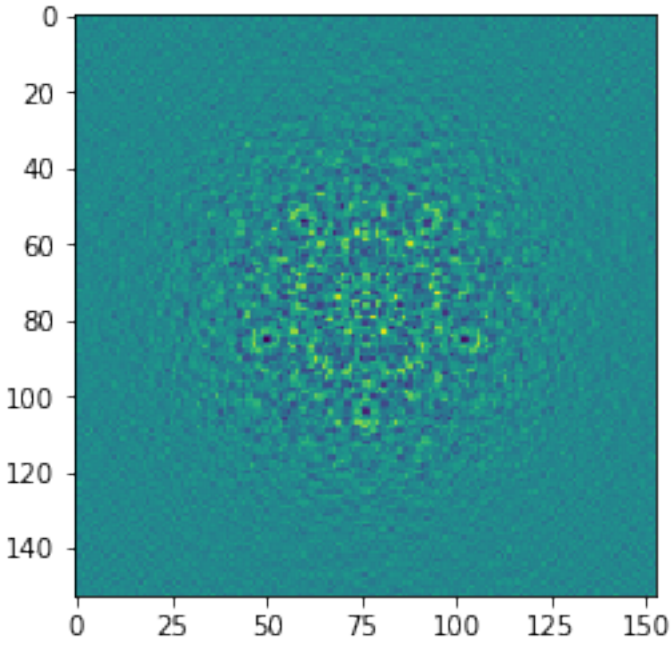
```
            rot_x, rot_y, rot_z = (np.meshgrid(
                                   np.arange(-(N-1)/2 , (N-1)/2 + 1, dtype=int),
                                   np.arange(-(N-1)/2 , (N-1)/2 + 1, dtype=int),
                                   np.arange(-(N-1)/2 , (N-1)/2 + 1, dtype=int)))

            rot_x = rot_x[..., np.newaxis] * a
            rot_y = rot_y[..., np.newaxis] * b
            rot_z = rot_z[..., np.newaxis] * np.cross(a, b)

            grid = rot_x + rot_y + rot_z

            # standard coordinates
            x = np.arange(-(N-1)/2, (N-1)/2+1)
            y = np.arange(-(N-1)/2, (N-1)/2+1)
            z = np.arange(-(N-1)/2, (N-1)/2+1)

            b_j_hat_f = (interpolate.RegularGridInterpolator(points=[x, y, z],
                                                             values=bp_result[0],
                                                             bounds_error=False,
                                                             fill_value=0))

            interpolated = b_j_hat_f(grid)
            w_x, w_y, w_z = (np.meshgrid(
                             np.arange(-(N-1)/2 , (N-1)/2 + 1, dtype=int),
                             np.arange(-(N-1)/2 , (N-1)/2 + 1, dtype=int),
                             np.arange(-(N-1)/2 , (N-1)/2 + 1, dtype=int)))
            interpolated = np.exp(-np.pi * 1j * (w_x + w_y + w_z)) / N**3 * interpolated
            sum_b_j_hat += interpolated
            sum_l_j_hat += l_j_hat

        final_recon = np.fft.ifftn(np.fft.ifftshift(sum_b_j_hat))
        return final_recon.real

In [76]: def estimate_orientations(images):

        N = len(images[0])
        pairs = [(0, 1), (0, 2), (1, 0), (1, 2), (2, 0), (2, 1)]

        for k in np.arange(2, len(images)):
            pairs = []
            for p in pairs:
                i = p[0]
                j = p[1]

                #find common line pairs
                possible_l_ij = []
                possible_l_ji = []
                for theta_i in np.linspace(0, math.pi, num=180):
```

```
                    for theta_j in np.linspace(0, math.pi, num=180):
                        R_i = (np.matrix([[np.cos(theta_i), -np.sin(theta_i)],
                                          [np.sin(theta_i), -np.cos(theta_i)]]))
                        R_j = (np.matrix([[np.cos(theta_j), -np.sin(theta_j)],
                                          [np.sin(theta_j), -np.cos(theta_j)]]))
                        # rotate unit vectors by theta_i and theta_j
                        l_ij = [0, 1] * R_i
                        l_ji = [0, 1] * R_j

                        x = np.arange(0, N)
                        y = np.arange(0, N)
                        l_ij_f = (interpolate.RegularGridInterpolator(
                                          points=[x, y], values=I[i],
                                          bounds_error=False, fill_value=0))
                        l_ji_f = (interpolate.RegularGridInterpolator(
                                          points=[x, y], values=I[j],
                                          bounds_error=False, fill_value=0))
                        sampled_l_ij = l_ij_f(l_ij)
                        sampled_l_ji = l_ji_f(l_ji)
                        possible_l_ij.append(sampled_l_ij)
                        possible_l_ji.append(sampled_l_ji)
                    l_ij = np.argmax(possible_l_ij)
                    l_ji = np.argmax(possible_l_ji)
                    pairs.append((l_ij, l_ji))
                C = ((np.dot(pairs[0][0], pairs[1][0])) /
                     (np.dot(np.linalg.norm(pairs[0][0]), np.linalg.norm(pairs[1][0]))))
                B = ((np.dot(pairs[0][1], pairs[1][1])) /
                     (np.dot(np.linalg.norm(pairs[0][1]), np.linalg.norm(pairs[1][1]))))
                A = ((np.dot(pairs[2][0], pairs[2][1])) /
                     (np.dot(np.linalg.norm(pairs[2][0]), np.linalg.norm(pairs[2][1]))))

                #spherical law of cosines, solve for alpha, beta, gamma
                alpha = (np.cos(A) - np.cos(B)*np.cos(C)) / np.sin(B)*np.sin(C)
                beta = (np.cos(B) - np.cos(A)*np.cos(C)) / np.sin(A)*np.sin(C)
                gamma = (np.cos(C) - np.cos(A)*np.cos(B)) / np.sin(A)*np.sin(B)



                return
```

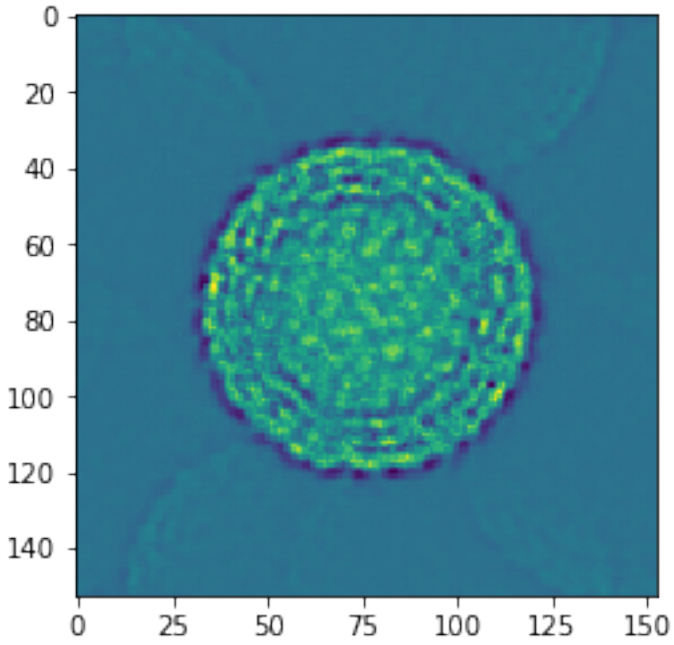
In [6]:
```
zika_small = mrcfile.open('input_data/zika_153.mrc')
small_rho = zika_small.data
```

In [7]:
```
plt.imshow(small_rho[0])
```

Out[7]:
```
<matplotlib.image.AxesImage at 0x11466fb70>
```



In [7]:
```
a = 2*np.random.rand(3)-1
b = np.cross(a, 2*np.random.rand(3)-1)
a = a/np.linalg.norm(a)
b = b/np.linalg.norm(b)
I = project_fst(small_rho, a, b)
# test.shape
plt.imshow(I)
plt.savefig('project_fst.png')
```




```
In [32]: noise = np.random.normal(loc=0.0, scale=3.0, size=small_rho.shape)
         noisy_rho = small_rho + noise

In [33]: test_noise = reconstruction(noisy_rho, 10)
         test_noise.shape

Out[33]: (153, 153, 153)

In [30]: test = reconstruction(small_rho, 10)
         test.shape

Out[30]: (153, 153, 153)

In [10]: back_projection_test = back_projection(I)

In [11]: back_projection_test.shape

Out[11]: (153, 153, 153)

In [35]: # Using mode 'r+' allows read and write access:
         mrc = mrcfile.new('unweighted_noisy_recon_s3.mrc', overwrite=True)
         mrc.set_data(test_noise.astype('float32'))
         mrc.data
         mrc.close()
```