Particle Picking and 3D Reconstruction in Cryogenic Electron Microscopy

A Dissertation
Submitted in partial fulfillment of
the requirements for the degree of
Master of Technology
by

Khursheed Ali
(Roll No. 163059009)

Supervisor:
Prof. Ajit Rajwade

Department of Computer Science and Technology
Indian Institute of Technology Bombay
Mumbai 400076 (India)

31 August 2019

Approval Sheet

This dissertation entitled "Particle Picking and 3D Reconstruction in Cryogenic Electron Microscopy" by Khursheed Ali is approved for the degree of Master of Technology.

Sharat 27/6

Prof. Sharat Chandran
Department of Computer Science and Engineering

Examiner

29/6/2019

Prof. Parag Chaudhuri
Department of Computer Science and Engineering

Examiner

Prof. Ajit Rajwade
Department of Computer Science and Engineering

Supervisor

Sharat 27/6

Prof. Sharat Chandran

Chairman

Date: 27/6/2019

Place: IIT Bombay

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources.. I declare that I have properly and accurately acknowledged all sources used in the production of this report. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Khursheed Ali
(Roll No. 163059009)

Date: 26 June 2019

Abstract

Cryogenic Electron Microscopy (Cryo-EM) is a special type of transmission electron microscopy (TEM) which is used for getting projections (tomographic projections) for very tiny biological samples like bacteria, proteins, etc. in cryogenic environments. Then, these projections are used for deriving the 3D internal structure of particles. But, the output from the electron microscope is a very large 2D image (also called as micrograph) containing projections of many copies of an identical 3D structure against a background. So, before going for 3D reconstruction, projections are to be marked out manually or automatically for reconstruction. Marking of such projections in the micrograph is called as *particle picking*.

Each micrograph contains thousands of projections, so marking manually is a very time-consuming process. Also, 3D reconstruction requires projections to be aligned on their center for robust reconstruction. Thus, marking thousands of particles manually with very high precision is very time-consuming. Because of that, there is a need for a robust automatic algorithm which can mark particles with high precision. This thesis presents a series of algorithms for this purpose.

Marked particles are denoise first. Then, these projections are used for 3D reconstruction. But, as angles and shifts for projections are unknown, so reconstruction is not possible directly. Firstly angles and shifts estimation is performed, and then reconstruction is carried out. So, we present a robust algorithm for the joint problem of 3D reconstruction of the underlying structure along with the estimation of the projection viewing parameters (2D shifts and 3D rotation parameters for each particle).

Table of Contents

List of Figures

List of Tables

Chapter 1

Background

1.1 Tomographic Reconstruction

1.1.1 Introduction

The objective of tomographic reconstruction is to obtain 3D representation of the internal structure of 3D object. For example, in computed tomography (CT) scan of human head (figure 1.1), the structure of the brain can be seen.

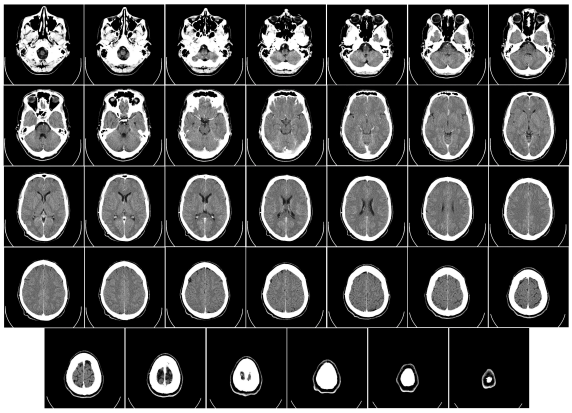


Figure 1.1: Computed tomography of human brain, slice of 3D volume from base of the skull to top. Source: wiki

Before creating 3D internal structure of the object, there are in between many steps which have to be performed. In order to make it simple, the example of CT machine can be considered. In CT machine, X-rays are used because it has ability to penetrate most of the objects. Using X-rays, many 2D projections are taken for the 3D object (in case of 2D object, 1D projections are taken) at different angles. These projections are also called as *tomographic projections*. These tomographic projections are defined as *Radon transform*.

Then, by using all these 2D tomographic projections, internal structure of 3D object is created.

1.1.2 Tomographic Projection

For better understanding, consider 2D object and its 1D projection (figure 1.2) as it is much easier to understand. But, this concept can easily be extended for 3D objects and their 2D projections. For taking projection, a parallel beam of X-rays are fired on the object at some fixed angle θ_k. The degree of absorption of X-rays by the object are recorded by the detector which is called as projection at angle θ_k. Now mathematical equation for the projection can be formulated. Let 2D object be defined as $f(x, y)$. Then Projection at angle θ_k is given by

$$R(f) = g(\rho_j, \theta_k) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y)\delta(x \cos\theta_k + y \sin\theta_k - \rho_j)dxdy \tag{1.1}$$

$R(f)$ is called as *Radon transform* of function f. $g(\rho_j, \theta_k)$ is read as projection of $f(x, y)$ at angle θ_k and ρ_j distance from origin.

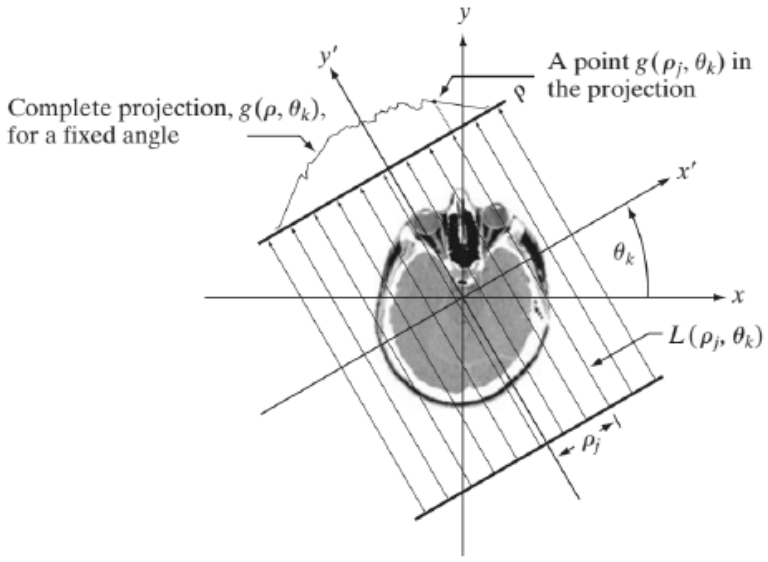


Figure 1.2: Parallel Beam (X-ray) Projection.
Source:Rafael C. Gonzalez , Richard E. Woods, Digital Image Processing

1.1.3 Fourier Slice Theorem

Fourier Slice Theorem gives us the relationship between Fourier transform and Radon transform. Radon transform of $f(x, y)$ is given by eq. (1.1). The 1D Fourier transform $g(\rho, \theta)$ w.r.t ρ with fixed θ is given by eq (1.2).

$$G(\mu,\theta) = \int_{-\infty}^{\infty} g(\rho,\theta)e^{-j2\pi\mu\rho}d\rho \tag{1.2}$$

Now expanding the $g(\rho,\theta)$ and simplifying the equation

$$G(\mu,\theta) = \int_{-\infty}^{\infty} g(\rho,\theta)e^{-j2\pi\mu\rho}d\rho \tag{1.3}$$

$$G(\mu,\theta) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\delta(x\cos\theta + y\sin\theta - \rho)e^{-j2\pi\mu\rho}dxdyd\rho \tag{1.4}$$

$$G(\mu,\theta) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\Big[\int_{-\infty}^{\infty}\delta(x\cos\theta + y\sin\theta - \rho)e^{-j2\pi\mu\rho}d\rho\Big]dxdy \tag{1.5}$$

$$G(\mu,\theta) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-j2\pi\mu(x\cos\theta + y\sin\theta)}dxdy \tag{1.6}$$

$$G(\mu,\theta) = [F(u,v)]_{u=\mu\cos\theta, v=\mu\sin\theta} \tag{1.7}$$

Here, $F(u,v)$ represents the 2D discrete Fourier transform of function $f(x,y)$. By eq (1.7) it can be concluded that, Fourier transform of projection of 2D object along some direction θ i.e $G(\mu,\theta)$ is equal to a slice of 2D Fourier transform of object along same direction θ in frequency plane, passing through origin. This is called as *Fourier Slice Theorem* or *Projection Theorem*. Similar theorem is also there for 3D object.

1.1.4 Filtered Back Projection (FBP)

From section (1.1.2) and (1.1.3), it can be known how projections (tomographic projections) are taken and it's relation with Fourier transform of object. By exploiting these facts, the internal structure of the object can be reconstructed. Let us denoted 2D object as $f(x,y)$ and it's 2D F.T (Fourier transform) is represented as $F(u,v)$.

$$f(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(u,v)e^{j2\pi(xu+yv))}dudv \tag{1.8}$$

Let u=$\mu\cos\theta$ v=$\mu\sin\theta$,

$$f(x,y) = \int_{0}^{2\pi}\int_{0}^{\infty} F(\mu\cos\theta,\mu\sin\theta)e^{j2\pi(x\mu\cos\theta + y\mu\sin\theta))}\mu d\mu d\theta \tag{1.9}$$

By using Fourier slice theorem eq (1.7) we get,

$$f(x,y) = \int_{0}^{2\pi}\int_{0}^{\infty} G(\mu,\theta)e^{j2\pi(x\mu\cos\theta + y\mu\sin\theta))}\mu d\mu d\theta \tag{1.10}$$

After simplification we get,

$$f(x,y) = \int_0^{\pi} \int_0^{\infty} G(\mu,\theta) e^{j2\pi(x\cos\theta + y\sin\theta))} |\mu| d\mu d\theta \tag{1.11}$$

So eq (1.11) says that, if there are many projections at different angles then it can recover $f(x,y)$, i.e. object internal structure from projections. But, one concern is there i.e. in eq (1.11) $|\mu|$ (ramp filter) is an *unbounded* function, therefore it is not integrable. In order to solve this problem Ram-Lak filter ($rect(\mu D)$) or Ram-Lak hamming filter is used. Updated FBP equation is given by eq (1.12) using Ram-Lak filter.

$$f(x,y) = \int_0^{\pi} \int_0^{\infty} |\mu| rect(\mu D) G(\mu,\theta) e^{j2\pi(x\cos\theta + y\sin\theta))} d\mu d\theta \tag{1.12}$$

1.1.5 Reconstruction using Compressed Sensing

In this section, another method for reconstruction using the set of projections taken at different angles is discussed. This method uses the concept of Compressed sensing (CS). CS works here because number of angles of projections are limited due to various reason like cost, energy and health considerations. This type of problem is also called as *angle starved* problem. It is known that images are sparse or compressible in standard basis like DCT. So, CS based optimization equation for tomographic reconstruction is given by eq (1.13)

$$E(\boldsymbol{\beta}) = \|\boldsymbol{y} - \boldsymbol{R}\boldsymbol{U}\boldsymbol{\beta}\|^2 + \lambda\|\boldsymbol{\beta}\|_1 \tag{1.13}$$

Here, $\boldsymbol{y}$ vector is created by concatenating 1D projections of various angles. $\boldsymbol{U}$ is the basis matrix in which image/object f(x,y) is sparse or compressible. $\boldsymbol{\beta}$ is the representation of $f(x,y)$ in $\boldsymbol{U}$ basis, i.e. $f(x,y)$ satisfy eq (1.14). $\boldsymbol{R}$ is the Radon matrix (it can also be Radon operator)

$$vec(f)) = \boldsymbol{U}\boldsymbol{\beta} \tag{1.14}$$

Eq (1.13) can easily be solved by using many optimization algorithm such as ISTA[2]. It is being observed that when number of angles are less then CS based reconstruction gives good result as compared to FBP up to certain limit.

1.1.6 Application

Tomographic reconstruction has many applications mostly in health care such as CT scan, industrial application such as fault detection in machines, observation of plant roots, remote sensing i.e. observing underground objects or phenomena and most importantly

study of protein, bacteria, Ribosome, cells and virus by biologist for creating medicine or vaccination or for any other purpose.

1.2 Tomography under unknown viewing parameters

In previous section (1.1), reconstruction from projections was considered where angle at which each projections were taken were known. But if only projections are known but not angles then, previously seen method for reconstruction will not be applicable.
There are various application where angles are not known such as patient moving during CT scanning (this mostly happens in pediatric CT), geometric errors in CT machine [12], moving insect CT scan [17], cryo-electron tomography[14] and many more [11]. In all these cases, the projection angles of the available projections are unknown. Moreover in cryo-electron microscopy the shifts of the projections are also unknown. Therefore, this is termed the problem of *tomography under unknown viewing parameters*.
This section talks about reconstruction of internal structure of object back from projection when angles are not known i.e. only with the help of tomographic projections.

1.2.1 Moment Based Reconstruction

Again, for simple understanding, lets consider 2D object and it's 1D projection. But this logic can easily be extend for 3D object and it's 2D projection.
Let say f(x,y) be the object then, moment of order (p,q) for f(x,y) is given by eq (1.15).

$$\boldsymbol{M}_{p,q} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)x^p y^q dxdy \tag{1.15}$$

Let $P_\theta(s)$ be the projection at angle θ then moment of order (n) is given by eq (1.16)

$$\boldsymbol{M}_{\theta}^{(n)} = \int_{-\infty}^{\infty} P_\theta(s)s^n ds \tag{1.16}$$

where,

$$P_\theta(s) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\delta(x\cos\theta + y\sin\theta - s)dxdy \tag{1.17}$$

Therefore, by substituting $P_\theta(s)$ in eq(1.16) by eq(1.17) and after simplification,

$$\boldsymbol{M}_{\theta}^{(n)} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)(x\cos\theta + y\sin\theta)^n dxdy \tag{1.18}$$

So, the Helgasson Ludwig Consistency Conditions (HLCC) [13] [1] [16] gives us the relation between the image moment and its projection moment which is given by eq(1.19).

$$\boldsymbol{M}_{\theta}^{(n)} = \Sigma_{l=0}^{n} f(x,y)\boldsymbol{C}_{n}^{n-l}(\cos\theta)^{n-l}(\sin\theta)^{l}\boldsymbol{M}_{n-l,l} \tag{1.19}$$

Here, projections of the f(x,y) is known. So it's moment i.e $\boldsymbol{M}_{\theta}^{(n)}$. By eq(1.19) unknown angles can be found out by iteratively solving for angles [13].Once the angles for these projections are found then any of the earlier mentioned methods for reconstruction can be applied.

Advantages of this method are simple to implement, makes direct use of consistency conditions and work well for small number of angles. But, major drawback of this method is that it is highly sensitive to noise.

1.2.2 Order Based Reconstruction

Order based [1] utilized simple heuristics like a nearest neighbour search to find angle ordering. In this, all acquired projections are sorted i.e. in order of increasing angles. Only issue is that this method requires large number of projection and it assumes that all unknown projection angles are independently sampled from uniform distribution, also angles can be from any known distribution but then accordingly algorithm will change. Assuming distribution to be uniform, then this problems reduce to matching problem where we have to match each projection to one of the angles sampled evenly from unit circle.

It is an iterative algorithm, where initial projection say P_0 and one angle say θ_0 are chosen. Then, P_0 projection is mapped with angle say θ_0. After that, find the closest projection to P_0 and then that projection is mapped with next angle from sorted distribution. This process is repeated till full mapping is completed.

Once it finds the angles for all projections then apply any of the earlier mentioned method for reconstruction.

Chapter 2

Introduction to Cryo-EM

2.1 Motivation

There are many applications where tomography concept is being used such as CT in which X-Rays are used. Suppose a biologist wants internal 3D model of a protein particle. Then, for this type of problem, CT machine cannot be used because it is neither designed for it nor a light-microscope. It is designed for viewing surface. Since late 20th century, a method called as *X-ray Crystallography* is being used for this kind of problem when sample (object under study) is very tiny[27].

Next section talks briefly about *X-ray Crystallography* and problem faced in it. Also the benefit of *Cryo-EM* over X-ray Crystallography.

2.1.1 X-ray Crystallography

As the name suggest, X-ray is being used for taking projections of tiny particle like protein or DNA. It uses the Braggs's law of *X-ray diffraction* by crystal for taking the tomographic projection [27]. Here, in figure (2.1), DNA sample is firstly crystalline and then placed in between focused X-ray beam and detector. Because of crystal, X-rays are diffracted and this diffraction is captured on the detector screen. Then, from the recorded reading model reconstruction is done.

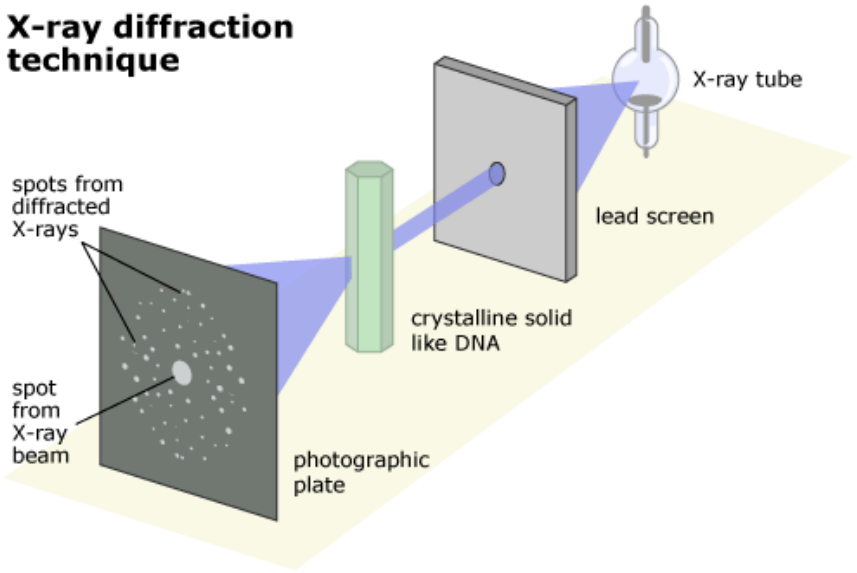


Figure 2.1: X-ray Diffraction.
Source: undsci.berkeley.edu - article-dna 04

2.1.2 Cryogenic Electron Microscopy (Cryo-EM)

Cryo-EM uses the electron beam for taking projections. Because of that, resolution of cryo-em is much better than X-ray crystallography. Depending on the type of sample under study, power of electron dose is set to get maximum possible resolution for that sample [6]. Here, the procedure for sample creation is completely different from X-ray crystallography. Firstly, a slice of sample to study is placed in vitrified water (water with some amount of ethane) and then freeze below $-150°C$ [6] [27]. After that slide is placed in electron microscopy (EM) and observations/projections are made in cryogenic conditions. As whole thing is carried out in cryogenic conditions and appliance used is Electron-Microscope (EM) that's why this is called *Cryo-EM*. See figure (2.2) for working of Cryo-EM.

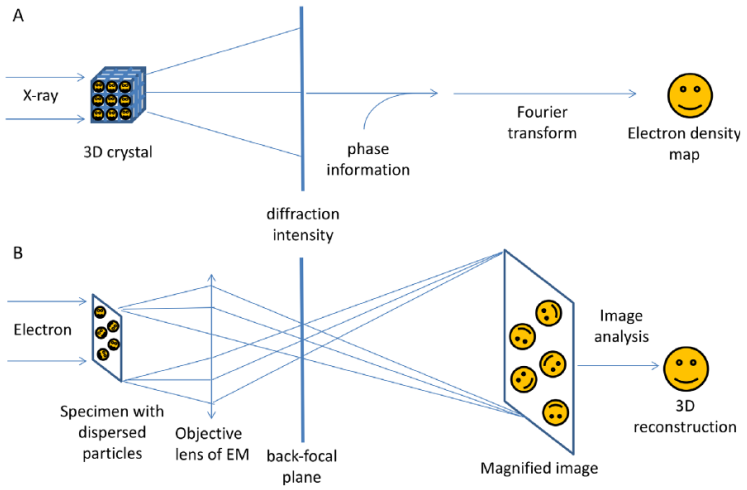


Figure 2.2: Functional difference between X-ray crystallography (A) and single particle cryo-EM (B) Source: Wang et al. [27]

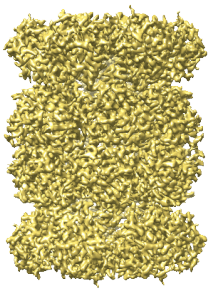
X-ray crystallography resolves structures of macro-molecules at atomic level. But, cryo-em can provide structural information at resolution from 3Å to 3nm. As cryo-em sample creation doesn't want sample to be in crystalline form so it can also examine non-crystalline structure. Also, it reveals the structure closer to its native state than X-ray crystallography [27]. See figure (2.2) for working difference of both methods.

2.2 Basic Pipeline

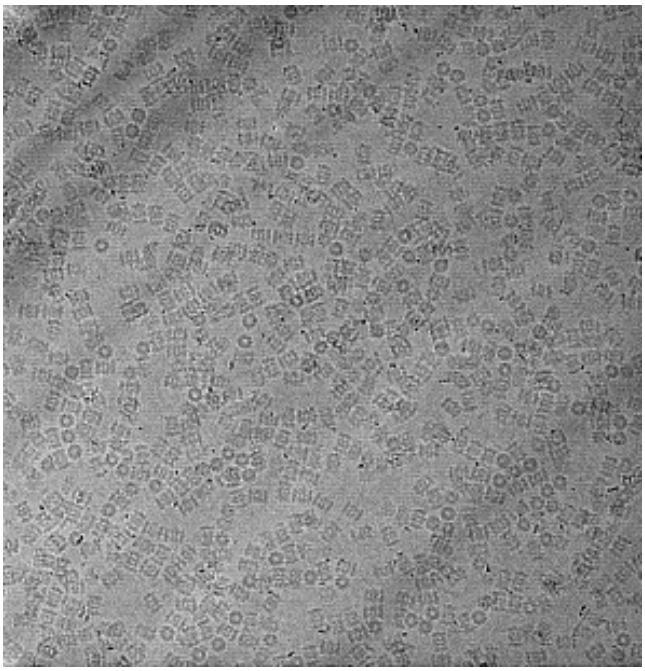
This section talks about cryo-em whole base pipeline starting from selecting the particle from micrograph (image (b) 2.3) also called as *particle picking* generated from Cryo-EM to it's 3D model reconstruction (similar to image (a) 2.3).

2.2.1 Particle Picking

Cryo-EM gives micrograph of very large size example 7000x7000. These micrographs contain lots of radon projections of the sample (around in thousands) taken by parallel beam because EM shoots electron beam in parallel. For reconstruction process, the very first steps is to manually or automatically mark out these projections by drawing bounding box around particles or just by marking its center in micrograph (2.4) and then crop out these projections. After that all the cropped out projections are collected into *stack* of projection images. As micrograph is very huge and it contains large amount of projections, so manually marking is very time consuming process. Now a days, a biologist may crowd-source this initial phase of particle picking [3]. But, even after crowd



(a) T20S Proteasome



(b) Micrograph of T20S Proteasome at 2.8 Å Resolution Source: EMPIAR-EBI data-bank

Figure 2.3: Source: EMPIAR-10025 EBI data-bank: https://bit.ly/2JrFCD5

sourcing, they have to remove wrongly marked particle by themselves which is still a very time consuming activity.

For saving time in particle picking, there are few automated and semi-automated algorithms. Example for one such automated algorithm is "difference of Gaussian" where a broad 2D Gaussian function (i.e. having large σ) convolved with micrograph is subtracted from narrow 2D Gaussian function (i.e. having smaller σ) convolved with micrograph. Result of this method gives circular white object pattern with a dark surrounding [21]. Problem with this approach is the tuning of hyperparameters like here, σ depending on the type particles and noise level in micrograph. An example of semi-automated method

[1]EBI data bank: http://www.ebi.ac.uk/pdbe/emdb/empiar/entry/10025/

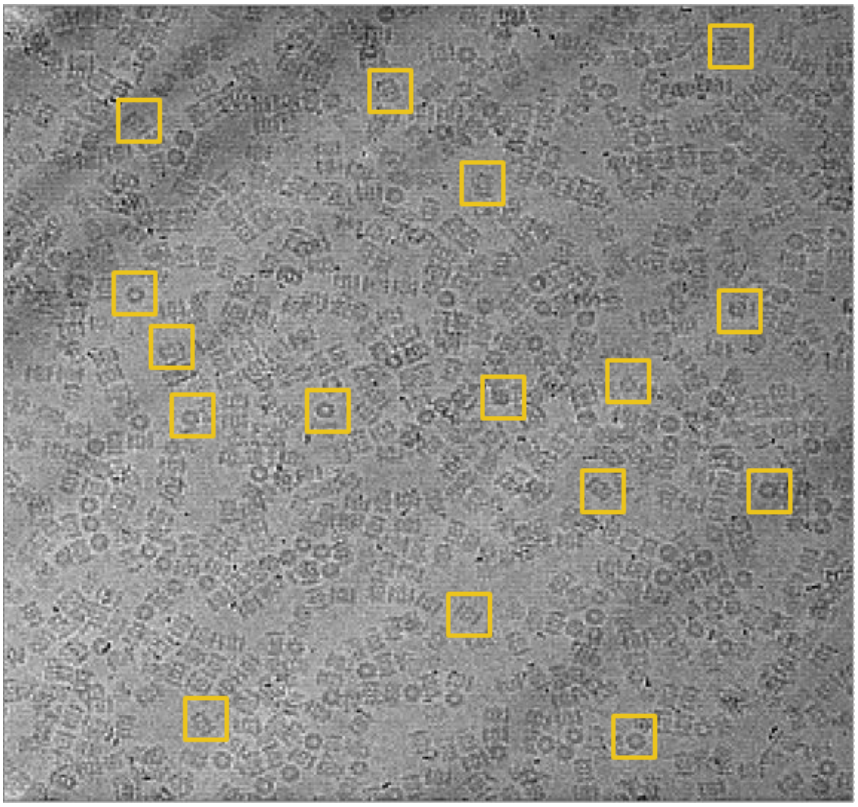


Figure 2.4: Manual particle picking from micrograph of T20S Proteasome Source: EMPIAR-10025 EBI data-bank[1]

21

is a machine learning based particle picking [4]. In case of machine learning, classifiers are trained in same noisy environment as of micrograph and tested on the same noisy environment. The next chapter talks about some semi-automated method and new detector design and it's experimental results on simulated data.

2.2.2 CTF correction

Image formation in an electron microscope for biological samples taken under cryogenic environment by freezing the sample in vitrified water can mathematically be described as Contrast Transfer Function (CTF). CTF depends on many parameters such as defocus, astigmatism, beam coherence, energy spread and tilting of the sample [6]. CTF oscillates rapidly, with some Fourier components transferred with positive contrast and other with inverted contrast. CTF is directly proportional to defocus i.e. if defocus increases then density of reversal increases which means higher contrast. So if defocus is higher, it helps in visualizing micrograph which further helps in marking of the particle and determining its orientation. But, this high defocus give rise to astigmatism i.e circular

ring in CTF turns into ellipse [21]. Because of all these issues, *CTF correction* is needed for the micrograph.

2.2.3 Clustering

The projection image from the stack of images is highly noisy i.e. it has very low SNR value since the electron beam used for taking projection is of very low power otherwise it will damage the sample. Also, distortion due to CTF effects makes it difficult to evaluate the projection. So, in order to resolve this problem, *clustering* of all projections from stack are done. After the clustering, all projections within a cluster are averaged out to get one average projection per cluster. This helps a lot in reducing the noise.

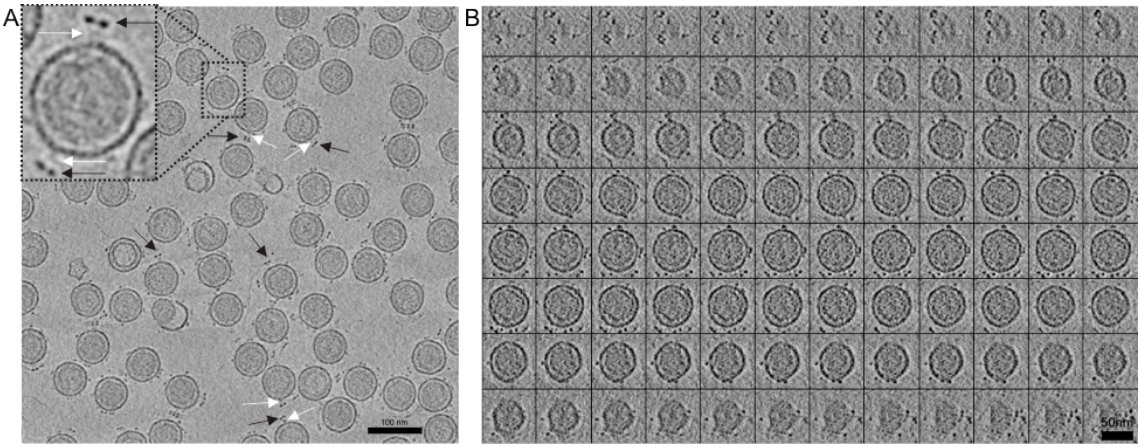


Figure 2.5: a) Phi12 bacteriophage micrograph b) Each row represent similar projection. Showing cluster of of projection before averaging.
Source: med.nyu.edu [2]

There are lots of methods for performing clustering. Algorithm used by RELION (REgularized LIkelihood OptimizatioN) uses maximum likelihood 2D classification, it is a very widely used software tool which performs all the base pipeline task [20]. In brief, this algorithm firstly select an initial set of reference projections. Then, for each projection, probability w.r.t rotation and translation with each reference projection is calculated. Then averaging is carried out for projections with higher probabilities. Averaged projections are then taken as new reference projections and the whole process is repeated few times.

[2]EMPIAR-1002 data bank: https://med.nyu.edu/skirball-lab/stokeslab/phi12.html

2.2.4 Angle Assignment

After clustering (section 2.2.3) step, projections are little noise free. Now, 3D model is reconstructed using these projections, but for 3D reconstruction, there is a need of angles at which these projections are taken if reconstruction is done using method discussed in previous section i.e. FBP (1.1.4) or CS (1.1.5). Finding angles and then reconstruction is called as tomography under unknown angle. So angles can be found out using two methods i.e. Moment-based (1.2.1) and Order-based (1.2.2).

2.2.5 3D Reconstruction

Once angles for each projection are known then 3D reconstruction is possible using FBP (1.1.4) or CS (1.1.5) based method. As always, if there are variety of projections at different angles then 3D reconstruction will be more accurate. But in *SPR (single particle reconstruction)* [21], there are situations where acquiring projections at various angle is not possible because of the particle orientation as projection angle of EM is fixed i.e. perpendicular to slide. In EM, slide is the place where sample is placed for observation or projection. So the only way to get projections at different angles is from particles orientation in frozen vitrified water sample [6]. But problem is that particles orientation in sample is not fully random. So there will always be the case in which projection can not acquired along certain direction. For solving this problem, biologist tilt the sample slide with some angle and then projection are taken. This tilting is done many times in many direction to cover large possible directions. If projections are taken this way, then effect of CTF will be different for different tilt direction and also within slide because of tilt. In this type of situation CTF correction is more challenging [6]. This method is called as *Single Particle Reconstruction*.

2.3 Challenges

2.3.1 Radiation damage

In Cryo-EM, electron beam is used. Beam electrons may start interacting with the sample electron which leads to radiation damage. In other terms, if the exposure time of vitrified biological sample is increased then this leads to breaking of covalent bonds of sample (because of electron-electron integration) and progressively it will do the structural degradation[6]. Therefore, total dose of electron that can be used is limited. This also affects the resolution i.e. magnification of cryo-em which means depending on the biological sample under study, resolution will be different.

2.3.2 Noise

Higher power electron dose usually destroys the sample. So, for handling these kind of issues, low power electron doses are used. But, this leads to increase in noise w.r.t. signal i.e. SNR for the micrograph become very low. In Cryo-EM, the distribution of noise is *Poisson* defined by eq (2.1), which makes the problem more challenging as Poisson noise is signal dependent. In case of an underlying image $I(x, y)$,, it is given by equation eq (2.2) which means removal of Poisson noise is far more challenging than signal independent noise (like i.i.d Gaussian noise)

$$P(X = i) = \frac{\lambda^i}{i!} e^{-\lambda} \tag{2.1}$$

$$Y(x, y) \sim \mathrm{P}oisson(I(x, y)) \tag{2.2}$$

Y(x,y) represent Poisson-noisy version of I(x,y).

2.3.3 Unknown angles

When projections are taken in cryo-em, angle at which beam is fired is known, but problem here is that when biological samples are frozen in vitrified water, then the orientation of each of these samples is random and unknown. So, the projection angles are different for every particle because of their orientation. As orientation of these samples are random which also means projection angles are also unknown, this information lose causes problem at the time of reconstruction from these projections.

2.3.4 Heterogeneity

Sample of same biological particles many not be identical i.e. every particle has some heterogeneity associated with it. Because of this complexity in whole process increases. At the time of selecting particle from micrograph, one has to make sure that consistency of particle has been marked otherwise this causes problem at the time of finding angles and also it might give rise to wrong 3D structure reconstruction [21]. Our current work does not deal with problem of heterogeneity.

2.3.5 Manual Particle Picking

Picking of particle is mostly time consuming and an important task. As discussed previously, wrong particle picking will lead to wrong result. Still today, many prefer

to mark these particle in micrograph with hands so that error in particle picking can be reduced.

So, by the nature of problem, there is a need of robust automatic or semi-automatic algorithm for particle picking which can handle outlier like few wrongly marked particle. Also, these algorithms should be fast at the time of finding the particle location in micrograph. The next chapter is mostly focused on first stage of the pipeline i.e. *particle picking* and will talk about few semi-automated methods and their testing on simulated data set.

2.4 Nobel Prize

In the year 2017, Jacques Dubochet, Joachim Frank and Richard Henderson shared the *Nobel Prize in Chemistry* for developing a technique to image biomolecules. There work helps researchers to see biomolecules look in 3D. In 1970, Richard Henderson, a molecular biologist was trying to determine the shape of a protein called bacteriorhodopsin but X-ray Crystallography was not working because of the issue with structure bacteriorhodopsin. Then they used electron microscope for the first time for biological particle and produced their first 3D model of the protein. During same period Joachim Frank developed image-processing software to make sense of the fuzzy pictures that are produced when an electron microscope is aimed at a protein, and to convert these two-dimensional blurs images into 3D molecular structures. Also in the early 1980s, Jacques Dubochet worked out how to prevent water-soluble biomolecules from drying out in the vacuum of an electron microscope and his team found a way to flash-freeze solutions of proteins using liquid ethane, keeping the molecules relatively still when they were analyzed under electron microscope. There research during the period 1970s-1980s has benefited researchers in analyzing biomolecules in 3D because of that in the year 2017 they got Nobel Prize in Chemistry.

Chapter 3

Particle Picking using Machine Learning

3.1 Semi-Automated Method

In semi-automated procedure, there is some need of human intervention in the procedure. This chapter talks about such type of semi-automated method for particle picking. All methods discussed here require only one time human intervention and that even at the start of method such as training of *Support Vector Machine (SVM)* classifier. Here, a human user will mark out a few a few (say around 2K, far less than the total number of say 30K projections available in the micrograph). Then these projections are given as input to a machine learning based classification algorithm which then try to distinguish between particle and background.

3.2 Training Architecture

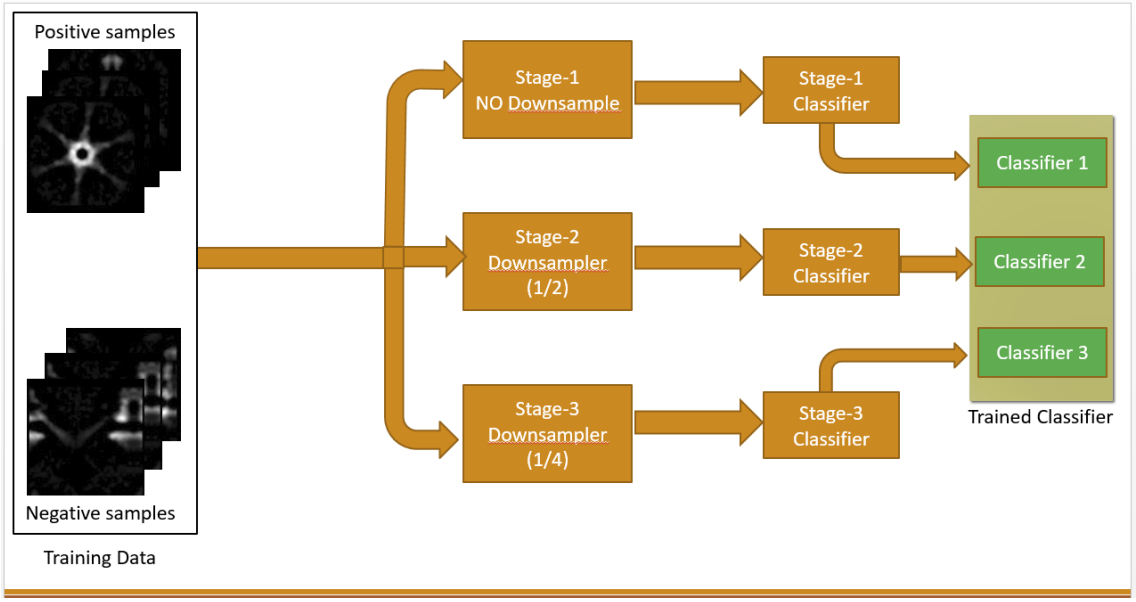


Figure 3.1: Particle Detector Training Architecture

One simple way to train machine learning classifier is to feed training set to the SVM classifier and tune the hyper-parameters which are best suited for the application. Then, use that trained classifier for detection purpose. But, this architecture of training will not work in cryo-em micrograph because actual micrograph dimensions are in the thousands example 7000x7000 or much bigger than this. In our experiment, the dimension of micrograph is 7420x7670 (56 million pixels). So, if simple architecture is used then time required for creating *probability map* (some also calls it *score map* or *heat map*) for micrograph will be huge. Probability of a patch centered at $\boldsymbol{x_i}$ (x_i, y_i) for a positive class is calculated using per class score generated by SVM (our case only classes) and softmax function. For two class, softmax is given by eq(3.1).

$$P(y = +ve|\boldsymbol{x_i}) = \frac{e^{si_{+ve}}}{e^{si_{+ve}} + e^{si_{-ve}}} \tag{3.1}$$

Here, si_{+ve} and si_{-ve} are the class scores generated by SVM model for patch centered at x_i. Scores are inversely related to the distance from the separating hyperplane.

For example, take micrograph of dimension 2000×2000 i.e. 4×10^6 pixels. Let us say, already trained classifier takes 0.001 sec for classifying one patch of dimension 333x333 centered at ith pixel and give its probabilistic score. Then, for finding score at every pixel will take around 40000 secs ($4 \times 10^6 \times 0.001$) which is approximately equals to 11hrs which means that this is a very poor particle picker as expected time should be

in seconds or up to minutes but not in hours.

In order to overcome this problem, the new architecture (by tweaking the current way of training) follows downscaling of model i.e. in case of *3-downscale* model three classifiers are trained. The first classifier is simple one i.e. no downscaling, the second model train classifier at downscale of 2 and the third model at downscale of 4. So, once the whole training process in completed, output will be *3-trained-classifiers* in case of 3-downscale model. Figure (3.1) shows the training architecture of 3-downscale model.

Depending on the size of the micrograph, the architecture code can be increased or decreased (just by changing configuration file) considering computation time and accuracy of the system.

3.3 Particle Detector Architecture

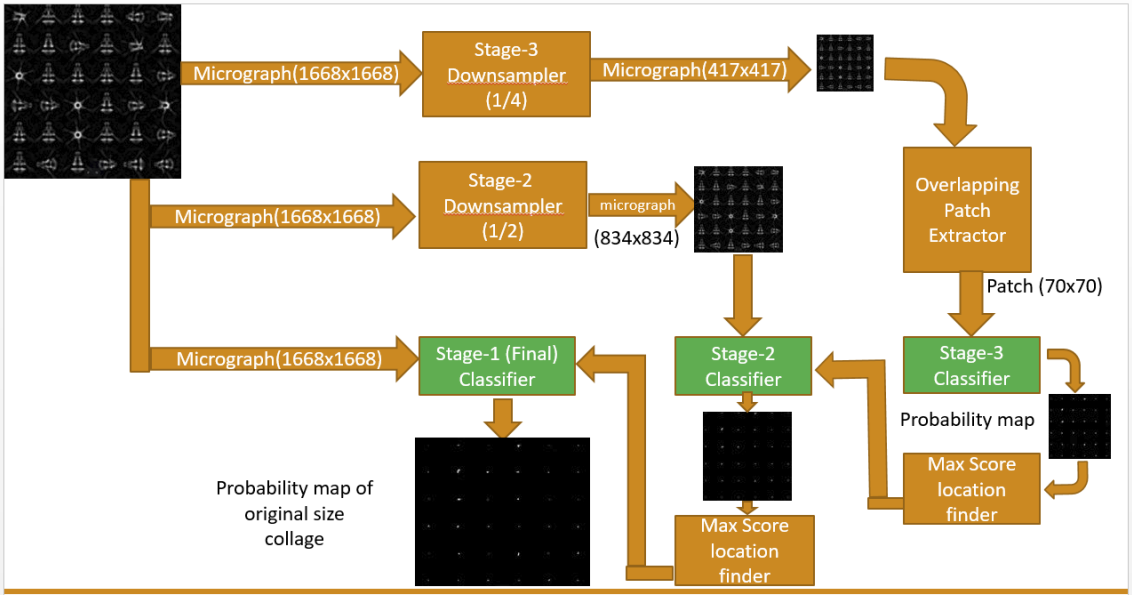


Figure 3.2: Particle Detector

Detecting particle in the micrograph and creating its probability map will take huge amount of time in case of single classifier, see section (3.2) for its computation time. In order to reduce this detection time, the new training architecture has been proposed i.e. *downscale* model. The detection using this new trained model is little complex as compared to a *new trained modelled architecture*. Figure (3.2) describes the work flow of detection process.

The detection process is a reverse training process in terms of stages. In case of 3-downscale model, firstly stage-3 is performed then stage-2 and at last stage-1 i.e. in reverse order. Stage-3 has only one input i.e. the original micrograph is downsampled by 4 and remaining i-1 stage will have 2 inputs i.e. for stage-j ($j < 3$) one input will be micrograph downsampled by 2^{j-1} and the other input will the output of (j+1)th stage passed through "Max score location finder" function. "Max score location finder" finds all the (x,y) coordinates in the probability map which are having probability values more than certain *min-threshold* (default 0.8). Min-Threshold has to be set depending on how well a classifier is being trained. Apart from nth stage (here 3rd) remaining stages do not run on every pixel. They use the location given by the previous (j+1)th stage. After up scaling these locations, their 4 point neighbour probabilistic score is calculated using input micrograph at that stage using that stage classifier.

Our code also supports three execution modes. These are single CPU, CPU Pooling and GPU. The computational table (3.1) shows time for getting probability map of a micrograph dimension of 1998x1998 three modes.

-	Single CPU	12-CPU Pool	GPU
Our Arch Time	~ 1.15 hrs	~ 9 mins	~ 4mins
Simple Arch Time	Not worth computing	16.97 hrs (61100 sec)	-

Table 3.1: Computation Table

3.4 Classifier

This section talks about three classifiers, first is *SVM* , second is *Random Forest* and last one uses deep learning which is *Faster R-CNN*. The next section will discuss about the experimental results when these classifiers are used for particle picking.

3.4.1 Support vector machine (SVM)

Support Vector machine is a machine learning algorithm used for classification. In our case, only two classes are there, one is positive class (i.e. true projection) and another is negative class. Also, as compared to deep learning, it requires far less data points to train.

SVM tries to optimize following objective function eq (3.3). Let y^i be the true label of $\boldsymbol{x}^i$ data point. In our case, as only two classes are there, so y^i will be 1 for positive image and -1 for negative image. In eq(3.2), $\boldsymbol{b}$ is bias term (unknown), $\phi(\boldsymbol{x})$ is a feature function i.e. for given input data point $\boldsymbol{x}$, it returns features vector and $\boldsymbol{w}$ is weight vector which classifier learns along with $\boldsymbol{b}$. ξ_i represent the slackness condition. In case of SVM, ξ_i is used when there is no perfect separability of the data point.

$$y^i = \boldsymbol{w}^T \phi(\boldsymbol{x^i}) + \boldsymbol{b} \tag{3.2}$$

$$\begin{aligned} (\boldsymbol{w^*}, \boldsymbol{b^*}) = \; & argmin_{w,b} \; \frac{\|w\|_2}{2} \\ & subject \;\; to, \\ & y^i(\boldsymbol{w}^T \phi(\boldsymbol{x^i}) + \boldsymbol{b}) \geq 1 - \xi_i \\ & \xi_i \geq 0 \;\; \forall i = 1, 2 \cdots, n \end{aligned} \tag{3.3}$$

For better classification, $\phi(.)$ places a vital role. $\phi(.)$ is not learned during training, it should be known before hand. In order to solve this problem, there is an alternate representation of SVM classifier i.e. using Kernel function $\boldsymbol{K(.,.)}$. Kernel representation do not use $\phi(.)$ and it is given by eq (3.4). In eq(3.4) y_i is the class label of the datapoint x_i, $\boldsymbol{K(.,.)}$ is the kernel function and C is the hyperparameter.

$$\begin{aligned} \max_{\alpha} \; & \frac{-1}{2} \sum_i^n \sum_j^n \alpha_i \alpha_j \boldsymbol{K}(\boldsymbol{x_i}, \boldsymbol{x_j}) + \sum_i^n \alpha_i \\ & subject \;\; to, \\ & \forall \alpha_i \epsilon [0, C] \\ & \sum_i^n \alpha_i y_i = 0 \end{aligned} \tag{3.4}$$

3.4.2 Random-Forest

Random-Forest is a variant of *Decision-Tree*. Random-Forest has ensemble of trees (see figure 3.3) whereas Decision tree has single tree. Each tree of Random-Forest is constructed in the same way as Decision-Tree. Only difference is that tree is constructed for *random* subset of dataset and features.

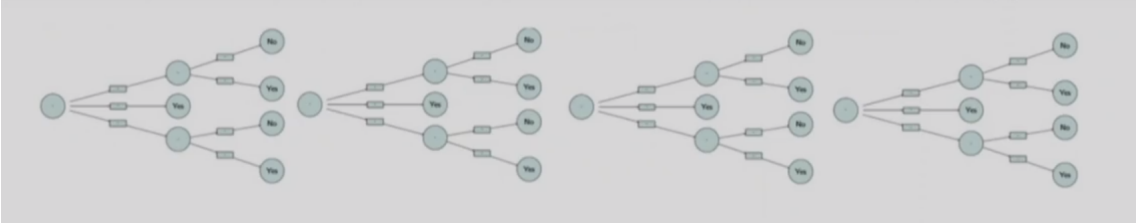


Figure 3.3: Random Forest. Circular nodes are features and rectangles are thre attributes of their respective features. Source: CS725 IIT-Bombay: Lecture-25 by Prof. Ganesh Ramakrishnan

Below is the *Bagging* algorithm for constructing Random-Forest. Bagging stands for "**B**ootstrapping **Agg**regat**ing**". Let **B** be the number of trees to be constructed, **D** denotes complete dataset, $\boldsymbol{\phi}$ denotes complete features set of dataset **D** and $\boldsymbol{T} =$ be the Forest set (initially empty). In Bagging, each tree has equal weightage i.e 1 but in Boosting, each tree as different weightage. Also, in Bagging, each trees are independent of each other because of that these can be constructed parallely.

Random-Forest Algorithm:

1. Uniformly at random (with replacements), sample subsets $\boldsymbol{D_s} \subseteq \boldsymbol{D}$ of the training data, $\boldsymbol{\phi_s} \subseteq \boldsymbol{\phi}$ of the feature set.
2. Using $\boldsymbol{D_s}$ and $\boldsymbol{\phi_s}$, construct Decision-Tree $T\,s$ using *Decision-Tree Algorithm*.
3. Repeat the step 1 and 2 for B times and keep adding the tree $T\,s$ into the forest set $\boldsymbol{T}$.

Decision-Tree Algorithm:

Let S is a sample of training examples, p_{ci} is a proportion of examples with class C_i (feature) in S and K be the number of class.

1. Find $\phi^* \leftarrow$ The **best** decision feature for next node
   (a) Entropy measures impurity of S:
   $$H(S) = \Sigma_{i=1}^{K} - p_{ci} \log_2 p_{ci} \tag{3.5}$$
   (b) For class ϕ_i, *Gain* is defined by below equation. $Gain(S, \phi_i)$ means expected Gain due to choice of ϕ_i
   $$Gain(S, \phi_i) = H(S) - \Sigma_{v \in Values(\phi_i)} \frac{|S_v|}{|S|} H(S_v) \tag{3.6}$$
   where, S_v means set of all examples having value of feature ϕ_i as v.

(c) Therefore, best ϕ^* feature for given S is define below equation.

$$\phi^* = argmin_{\phi_i} G(S, \phi_i) \tag{3.7}$$

2. Assign ϕ^* as the decision feature for the node

3. For each value (attribute) of ϕ^*, create new descendant of node

4. If training examples perfectly classified. Then STOP, Else iterate over new leaf nodes i.e. for each leaf node repeat the steps 1,2 and 3 recursively.

Each tree of the Random-Forest is a week tree having very high variance and low bias. During testing, final result is calculated by averaging the results of all the B trees (see eq. 3.8) because of this final variance of the model goes down as they get canceled out. Probabilistic score of new test point $\boldsymbol{x}$ is given by equation 3.8.

$$Pr(c|\boldsymbol{x}) = \frac{1}{|B|} \Sigma_{t=1}^{B} Pr_t(c|\boldsymbol{x}) \tag{3.8}$$

where $Pr(c|\boldsymbol{x})$ means probability of $\boldsymbol{x}$ belonging to class c and σ is over all the trees of the Random-Forest.

3.4.3 Faster R-CNN

Faster R-CNN [19] is the faster version of *Fast R-CNN* [8]. R-CNN stands for Region based Convolution Neural Network. R-CNN is the multiple object detector (localization and classification) which is the prior version of Fast R-CNN. The major difference between R-CNN [9] and Fast R-CNN [8] is that Fast R-CNN has ROI (Region of Interest) pool layer with fully connected neural network for classification and regression (localization) but, in case of R-CNN there is no concept of ROI, also SVM is used for of classification and CNN for regression. See figure 3.4 for R-CNN and figure 3.5 for Fast R-CNN.

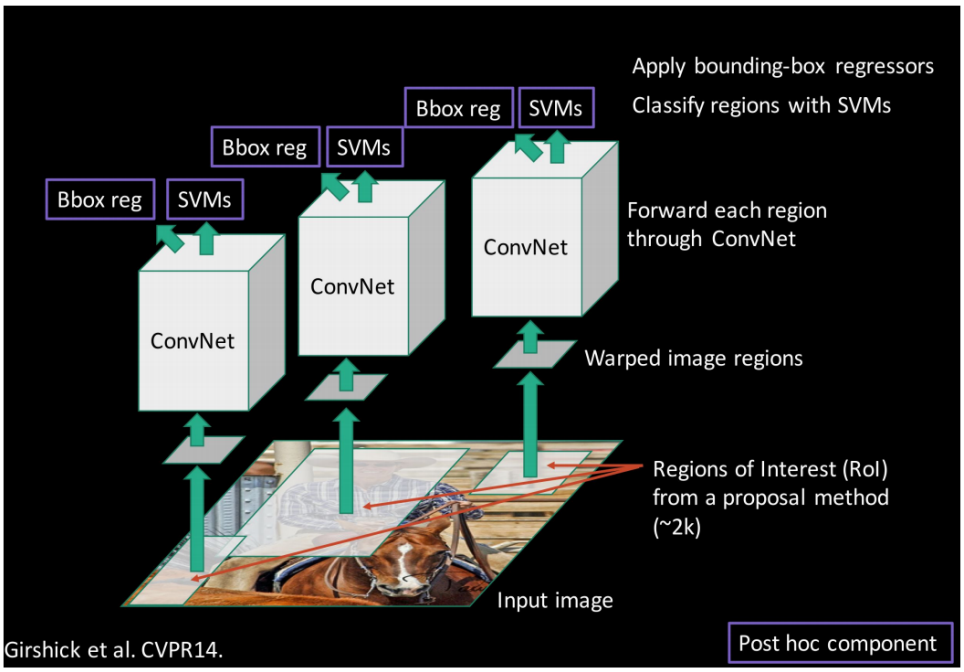


Figure 3.4: R-CNN [9] . Source: CS231-Stanford[1]

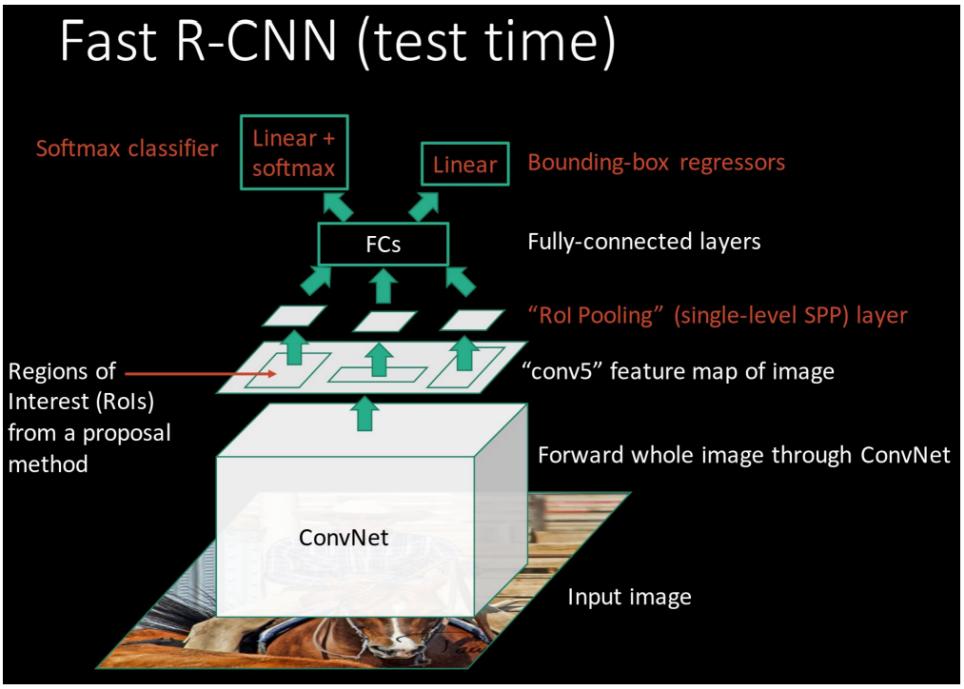


Figure 3.5: Fast R-CNN [8] . Source: CS231-Stanford[2]

As the name suggests, Faster R-CNN is much faster than Fast R-CNN (see table 3.7) w.r.t. test speed and accuracy. The big difference between Faster R-CNN (figure 3.6)

[1] Stanford:http://cs231n.stanford.edu/slides/2016/winter1516_lecture8.pdf - Slide: 53
[2] Stanford:http://cs231n.stanford.edu/slides/2016/winter1516_lecture8.pdf - Slide: 67

and Fast R-CNN (figure 3.5) is *Region Proposal Network* (RPN). In case of Fast R-CNN, RPN is external like Selective Search so it has to be trained explicitly. But in case of Faster R-CNN, RPN is a CNN network which is a part of whole object detector model (see figure 3.6).

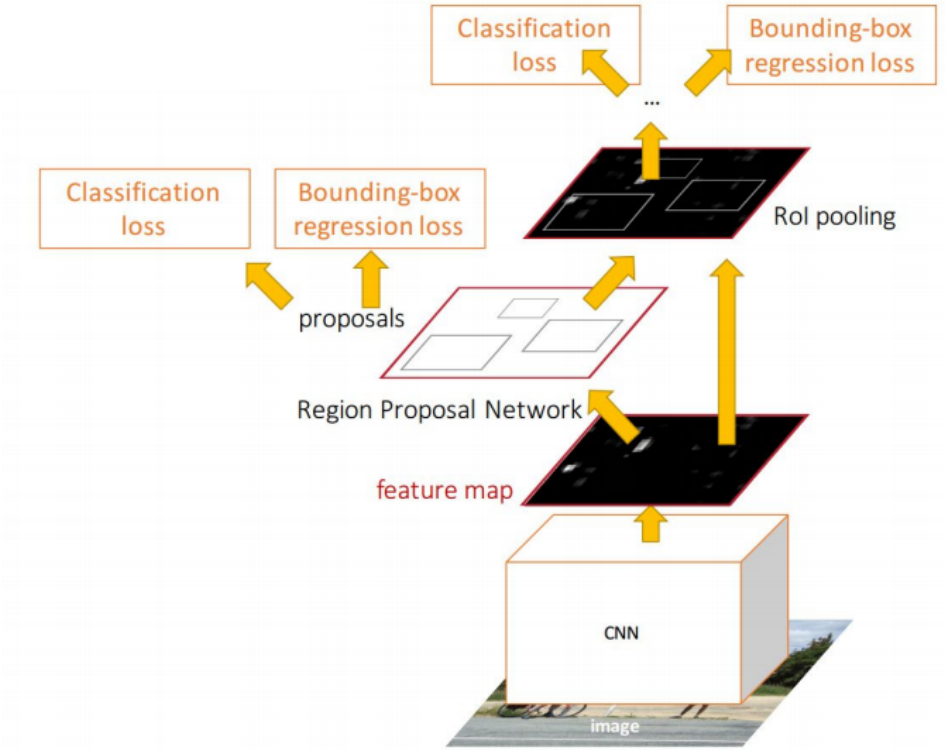


Figure 3.6: Faster R-CNN. Source: [19]

Faster R-CNN has RPN. Classification and localization are using CNN and full connected neural network. So while training Faster R-CNN, there are four loss functions that have to be minimized. Four loss functions are as follows (see figure 3.6):

- RPN classification (anchor good / bad)
- RPN regression (anchor → proposal)
- Fast R-CNN classification (over classes)
- Fast R-CNN regression (proposal → box)

Anchors helps in finding objects at different scale level [19] . So, the four losses are calculated at two levels, the first two at RPN level and the other two at output layer.

Training Faster R-CNN is not easy like other CNN networks. It has a four stage training [19] pipeline. In stage 1, RPN is trained then in stage 2 using the trained weights of RPN, "classifier and regression" network is trained. In stage 3, again RPN is trained but with full network, i.e. using the stage 2 weights of "classifier and regression" and in last stage "classifier and regression" are trained using the weights of the stage 3. Training is like an alternating fashion.

	R-CNN	**Fast R-CNN**	**Faster R-CNN**
Test time per image (with proposals)	50 seconds	2 seconds	**0.2 seconds**
(Speedup)	1x	25x	**250x**
mAP (VOC 2007)	66.0	**66.9**	**66.9**

Figure 3.7: Comparison between R-CNN, Fast R-CNN and Faster R-CNN on testing time and accuracy VOC-2007 [19] . Source: CS231-Stanford[3]

3.5 Experimental Results

In the previous experiments, the simulated datasets were used i.e. using the 3d structure of Ribosomes taken from the *EM Databank* [4] many projections were created and using these projections micrographs were generated. But to get the correct accuracy of models, it has to be tested against *Real Micrographs* of Ribosomes or Viruses.

Real Micrographs comes with many challenges with respect to *Particle Picking*. The first and the biggest challenge is marking of the particles on the micrograph. As the models are semi-automated, so to train classifiers there is a need of ~6K marked particles on micrograph. Manual marking takes time and more importantly one's need to be Biologist as identifying between heterogeneous projections and ice is hard. So for the experimental purpose and to save time, datasets taken for experiments are having markings and these marking are done by the trusted third-party. The second challenge is the size of datasets. Size of dataset varies from 500 GB to few TB (say 2 TB, our observation) in which dataset has movie files and each files may size up to 8 GB. So handling these files requires bigger size RAM and optimized algorithm.

[3] Stanford:http://cs231n.stanford.edu/slides/2016/winter1516_lecture8.pdf - Slide: 84

[4] EM-Databank: http://www.emdatabank.org/

3.5.1 Dataset

This section will talk about 4 dataset used in experiments. The dataset required for experiment need to have markings of particles in the micrograph. Otherwise these markings have to be done manually by Biologist or if someone else is doing then markings have to be verified by Biologist because wrongly marked particle may lead to poor classifier. Out of 4 datatset, three datasets are taken from *EM Databank* [5] with particles marked via *RELION* tool and remaining one dataset taken from the *IISER - Trivandrum* with marking done by *Dr. Ramanathan Natesh*[6]. Below is the detailed description about the datasets.

Dataset-1: T20S-Proteasome (T20)

This dataset is taken from the EM-Databank having EMD id as **EMD-6287** and EMPAIR id as **EMPIAR-10025**[7]. The 3D structure of T20-Proteasome is shown in figure 3.8. This 3D structure will help in finding error in 3D reconstruction.

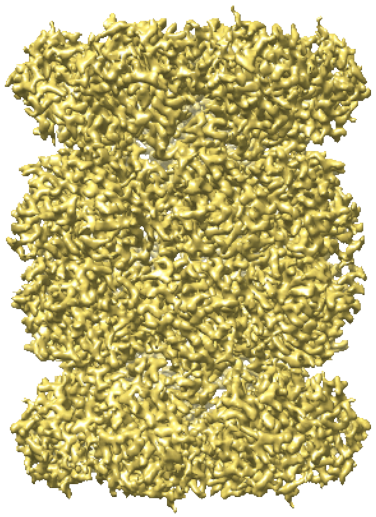


Figure 3.8: T20S Proteasome[8]2.8 Angstrom resolution

Total dataset size is 369 GB, which contains movie files each of size 3 GB in **mrc** format (mrc is the globally known standard format for saving raw images take from EM) Each having 38 frames per movie files. Dimension of each frame is 7420x7676. Markings of particles are done via a RELION [20] software but problem is that this RELION gives lot many other information, so these marking has to be parsed before it can be used. But,

[5]EM-Databank: http://challenges.emdataresource.org/?q=2015_map_challenge
[6]Dr. Ramanathan Natesh: http://faculty.iisertvm.ac.in/natesh/
[7]EMPIAR-10025:http://www.ebi.ac.uk/pdbe/emdb/empiar/entry/10025/
[8]EBI data bank: http://www.ebi.ac.uk/pdbe/emdb/empiar/entry/10025/

one issue with dataset is that not all particles are marked i.e. each micrograph has around 1500 particles or more but markings are only for approximately 250 particles.

(a) Full Scale (b) Downsampled by 12

Figure 3.9: T20S Proteasome Averaged Micrograph, In Full Scale view, it is very hard to visualize where are the particles. But after reasonable downsampling particles can visualized

Dataset-2: 80S-Ribosome (80S)

80S-Ribosome is the dataset from the EM-Databank having EMD id as **EMD-2660** and EMPAIR id as **EMPIAR-10028**[9]. Single particle 3d reconstruction have 3.2 angstroms resolution see the figure 3.10.

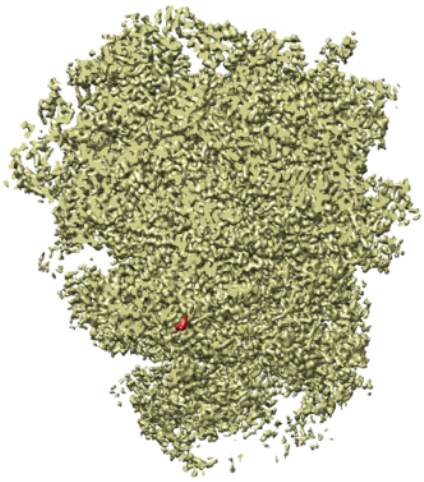


Figure 3.10: 80S Ribosome[10]3.2 Angstrom resolution

Initial dataset of 80S-Ribosome from the databank is very large, it is approximately equal to the 2 TB (Terabytes). This dataset contains lots of meta data related to each file which will not be used in particle picking but might be used at the time of reconstruction. Each file is a movie file of size 3 GB and having dimension of 4096x4096. Per movie

[9] EMPIAR-10025:http://www.ebi.ac.uk/pdbe/emdb/empiar/entry/10028/

[10]EBI data bank: http://www.ebi.ac.uk/pdbe/emdb/empiar/entry/10028/

file has 47 frames. Multiple frames will help in removing the noise by averaging of these frames. Figure 3.11 shows the averaged micrograph (average of 47 frames) of a movie file. Marking of particle is done using the RELION and each micrograph has approximately 50 to 60 particles.

(a) Full scale (b) Downsampled by 6

Figure 3.11: 80S Ribosome Averaged Micrograph. In Full Scale view, it is very hard to visualize where are the particles. But after reasonable downsampling particles can visualized

Dataset-3: β-Galactosidase (β-G)

β-Galactosidase is a dataset from the EM-Databank having EMD id as **EMD-5995** and EMPAIR id as **EMPIAR-10012/1003**[11]. Single particle 3D reconstruction at 3.2 angstroms resolution can be seen in figure 3.12.

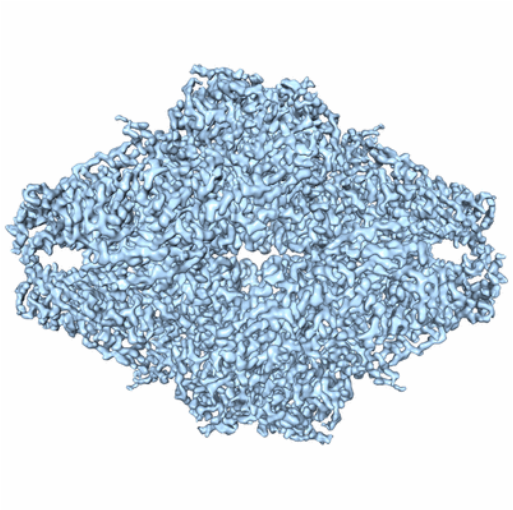


Figure 3.12: β-Galactosidase[12]3.2 Angstrom resolution

[11]EMPIAR-10025:http://www.ebi.ac.uk/pdbe/emdb/empiar/entry/10013/
[12]EBI data bank: http://www.ebi.ac.uk/pdbe/emdb/empiar/entry/10013/

Dataset is taken from the EM-databank which is approximately equal to the 108 GB. Each file is a averaged file of size 256 MB and has dimension of 7420x7676 Figure 3.11 shows the micrograph. Marking of particle is done using the RELION and each micrograph has thousands of particles but out of which approximately 50 are marked.

(a) Full Scale

(b) Downsampled by 12

Figure 3.13: β-Galactosidase Averaged Micrograph. In Full Scale view, it is very hard to visualize the particles. But after reasonable downsampling particles can visualized

The table 3.2 shows the detail about dataset in brief

Dataset	Size(GB)	Type	Number of files	Dimension
T20S Proteasome	369	Movie	196	7420x7670
80S Ribosome	2048	Movie	1081	4096x4096
β-Galactosidase	108	Averaged	509	7420x7670

Table 3.2: Dataset

3.5.2 Dataset Pre-Processing

There are many implementation challenges when these datasets are considered. As these datasets are extremely large in terms of "size of each file" so, handling these becomes tough. Below are the challenges:

File Extension

First thing in pre-processing is the file extension. Files are having **mrc** extensions. MRC stands for *Medical Research Council*. As mrc extension is not the standard format for an image or the movie. So, there is a need of parser for reading mrc extension. In mrc, each file is divided into two parts, first is the header section and next is data section. Header describes how data is presented in data section and meta data, for example NX and NY field specify the number of columns and rows in 3D data array respectively. For more

information related to header, see header table 3.14. As it is not a daily used format, so you may need to write your own parser depending upon the language.

Word	Bytes	Variable name	Description	Note
1	1-4	NX	number of columns in 3D data array (fast axis)	1
2	5-8	NY	number of rows in 3D data array (medium axis)	
3	9-12	NZ	number of sections in 3D data array (slow axis)	
4	13-16	MODE	**0** 8-bit signed integer (range -128 to 127) **1** 16-bit signed integer **2** 32-bit signed real **3** transform : complex 16-bit integers **4** transform : complex 32-bit reals **6** 16-bit unsigned integer	2
5	17-20	NXSTART	location of first column in unit cell	
6	21-24	NYSTART	location of first row in unit cell	
7	25-28	NZSTART	location of first section in unit cell	
8	29-32	MX	sampling along X axis of unit cell	
9	33-36	MY	sampling along Y axis of unit cell	
10	37-40	MZ	sampling along Z axis of unit cell	3
11-13	41-52	CELLA	cell dimensions in angstroms	
14-16	53-64	CELLB	cell angles in degrees	
17	65-68	MAPC	axis corresp to cols (1,2,3 for X,Y,Z)	4
18	69-72	MAPR	axis corresp to rows (1,2,3 for X,Y,Z)	
19	73-76	MAPS	axis corresp to sections (1,2,3 for X,Y,Z)	
20	77-80	DMIN	minimum density value	5
21	81-84	DMAX	maximum density value	
22	85-88	DMEAN	mean density value	
23	89-92	ISPG	space group number	6
24	93-96	NSYMBT	size of extended header (which follows main header) in bytes	7
25-49	97-196	EXTRA	extra space used for anything - 0 by default	
27	105	EXTTYP	code for the type of extended header	8
28	109	NVERSION	version of the MRC format	9
50-52	197-208	ORIGIN	phase origin (pixels) or origin of subvolume (A)	10
53	209-212	MAP	character string 'MAP ' to identify file type	
54	213-216	MACHST	machine stamp encoding byte ordering of data	11
55	217-220	RMS	rms deviation of map from mean density	
56	221-224	NLABL	number of labels being used	
57-256	225-1024	LABEL(20,10)	10 80-character text labels	

Figure 3.14: MRC header[13] format

Handling Movie File

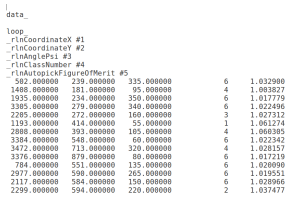
The second part in pre-processing is to convert the movie file into the average file. Averaging is done by adding up all the frames of movie file and then dividing each pixel by the number of the frames. But there is a problem, which is the size of each movie file. As each file size might go up to 8 to 9 GB for different types of datasets, so reading all the frames and then finding average will require a lot of main memory (RAM). In our case, maximum size of file for 80S-Ribosome is 3 GB (see section 3.5.1). So, to handle this problem, write mrc file parser such that it gives one frame at a time. By this, find average simultaneously while reading the movie file. Averaging will help in reducing the noise level which will further improve the classifier.

Parsing Particles Marking

The next setup in pre-processing is to parse the particle marking because each datasets has different style of representation for storing the marking. Like in case of T20S-Proteasome (3.5.1), marking is done by RELION and for all the micrographs markings are saved in

[13] MRC header: http://www.ccpem.ac.uk/mrc_format/mrc2014.php

a single file with lots of other meta data and header for the meta data. In case of 80S-Ribosome (3.5.1), marking is done through RELION and each micrograph has it's own marking file containing x and y location with meta data. For β-Galactosidase (3.5.1), each micrograph has its own marking file but marking file contains the bounding box (eman-box) information rather than coordinates of particles. Images 3.15 shows the snippet for these three molecules projections marking.



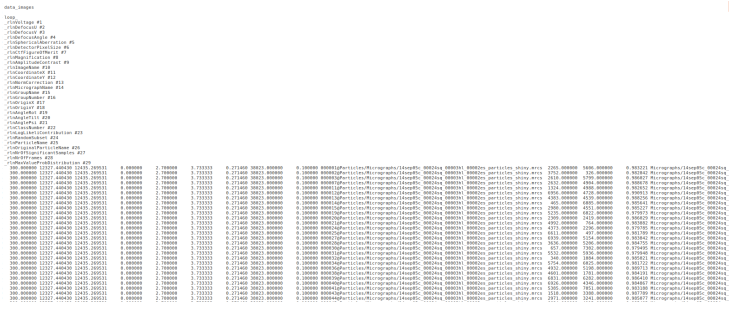
```
data_

loop_
_rlnCoordinateX #1
_rlnCoordinateY #2
_rlnAnglePsi #3
_rlnClassNumber #4
_rlnAutopickFigureOfMerit #5
 502.000000   239.000000   335.000000           6    1.032980
1408.000000   181.000000    95.000000           4    1.003827
1935.000000   234.000000   350.000000           6    1.017779
3385.000000   279.000000   340.000000           6    1.022496
2285.000000   272.000000   160.000000           3    1.027312
1193.000000   414.000000    55.000000           1    1.061274
2888.000000   393.000000   105.000000           4    1.060305
3384.000000   548.000000    60.000000           6    1.022342
3472.000000   713.000000   320.000000           4    1.028157
3376.000000   879.000000    80.000000           6    1.017219
 784.000000   551.000000   135.000000           6    1.020090
2977.000000   598.000000   265.000000           6    1.019551
2117.000000   584.000000   150.000000           6    1.020966
2299.000000   594.000000   220.000000           2    1.037477
```

(a) 80S-Ribosome

X	Y	H	W
4057	162	768	768
3456	490	768	768
1226	845	768	768
3715	854	768	768
3158	871	768	768
6371	1064	768	768
3051	1408	768	768
3257	1413	768	768
6443	2245	768	768
1745	2392	768	768
5317	2476	768	768

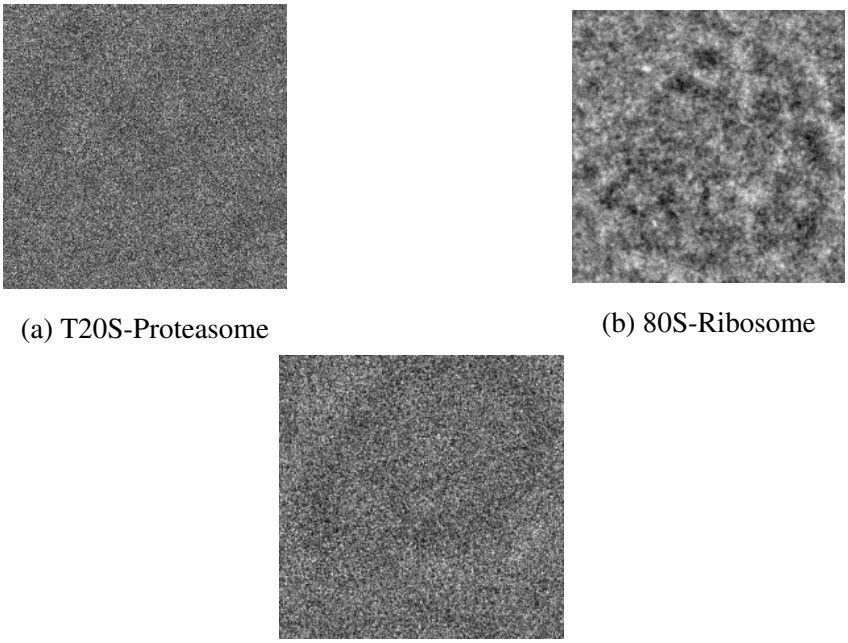
(b) β-Galactosidase



(c) T20S-Proteasome

Figure 3.15: Particle Marking Data Format. All the three datasets have different representation format. So, developing generic parser is not possible.

Training SVM and Random Forest requires training dataset and for validating the classifier, test dataset is needed. As the correct dataset is in mrc format so these cannot be directly given to model for training and on above that these are micrographs. First thing is to convert mrc micrograph to jpeg/png or directly use the image for further pre-processing. In the experiments raw micrographs are being used. Then, for generating the train and test dataset, crop the projections from micrograph by using the marking of the particles of their respective dataset. This will give *positive class* sample (see images 3.16) but for training classifier *negative class* samples such as background, ice particles, partially covered particle, multiple particles and different types of other particles are also required.



(a) T20S-Proteasome

(b) 80S-Ribosome

(c) β-Galactosidase

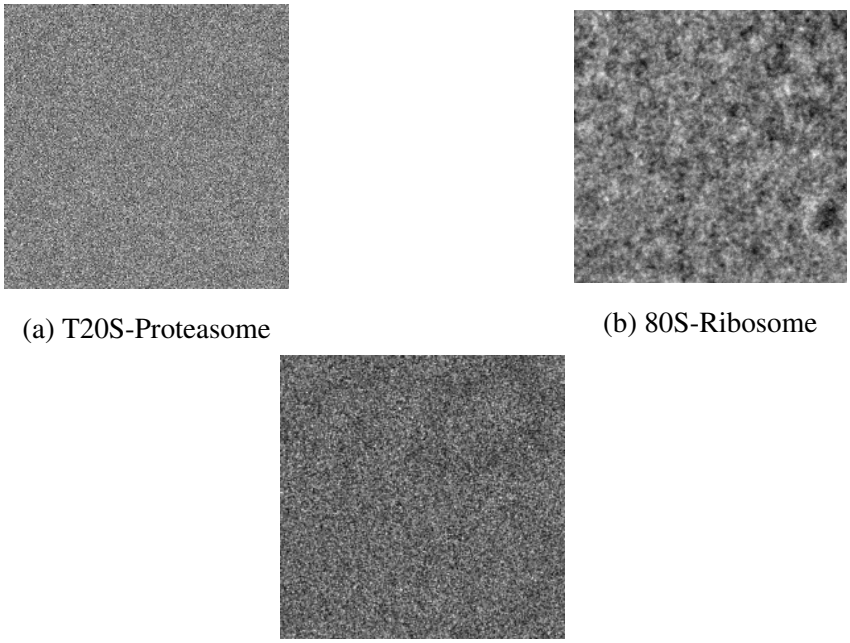
Figure 3.16: Positive Projection after cropping (Dimension 216x216)

Generating Train and Test Set - SVM and Random Forest

Negative samples (see images 3.17) plays a very vital role during classification of particle and non-particle. In the experiment, for generating negative samples [4], marking of positive particles were used. Given (x, y) location of a particle on the micrograph and maximum dimension of particle which will be known in prior (in our case it is 216x216) then two negative sample are generated by cropping at location $(x + max\_size/2, y + max\_size/2)$ and $(x - max\_size/2, y - max\_size/2)$. This will handle the case of partial particle, background and multiple particles. In our case, adding different other types of particles in negative set are not required because this trained model will only be used on image having only one type of particle.

Generating Train and Test Set - Faster R-CNN

Faster R-CNN is the variant of Convolution Neural Network (CNN). But unlike traditional CNN, it does both classification and object detection in given input image. For training Faster R-CNN, it requires images with bounding box (top-left x,top-left y,height, width) information related to the interested object rather single-single positive and negative image (216x216).



(a) T20S-Proteasome

(b) 80S-Ribosome

(c) β-Galactosidase

Figure 3.17: Negative Sample (Dimension 216x216)

In dataset, single micrograph dimension is very large and total count of these micrographs (approx. 7000x7000) are also very less. So whole dataset of micrographs are divided into train and test set. Then, each micrograph is further divided into 1000x1000 sub-micrographs. For each sub-micrograph, bounding box related to the objects present in it is also required. So these bounding boxes are calculated by finding particles coordinates which are marked and which lies in that sub-micrograph. This whole thing is also done for test set and while testing rather than passing whole big micrograph sub-micrographs (1000x1000) are passed. Image 3.18 shows the one of 1000x1000 cropped micrograph from the big micrograph and true marking of particles in the bounding box format.

Faster R-CNN also needs negative sample as that of SVM and Random Forest. But in this case there is no need of generating separate negative samples. The Faster R-CNN network will treat shifted bounding box from it's original position as the negative sample. This is very much similar to the way of generating negative sample in the case of SVM and Random Forest.

3.5.3 Train and Test Model Architecture

As seen in section 3.2 and 3.3 about the multi scale architecture. Scale of the model is the tuning parameter. Observation from the previous experiments is that if n-1 scale are

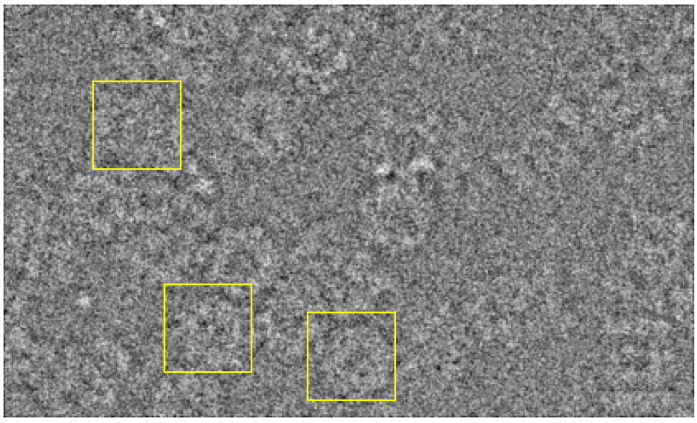


Figure 3.18: 1000x1000 cropped micrograph of T20S-Proteasome with bounding box

dropped in n downscale model then accuracy of the model will not be affected much, say for 3-downscale model [8, 4, 2]. If model is created only for downscale rate 8 then, it will give approximately same result as that of combined 3-downscale [8, 4, 2]. Reason for this effect is that classifier trained on higher resolution images gives poor result as compared to lower resolution image because in our application dataset size was very small. So in case of lower resolution image say "downscale model-8", less amount of data will required for training but in case of higher resolution image say "downscale model-4" , large amount of data will be required as compared to "downscale model-8 to get same amount of accuracy.

3.5.4 Feature Extraction

As train image dimensions are very large i.e. for 216x216 image will have 46656 pixels which is a huge number if features are considered. So to train classifier like SVM, Random forest or any other (apart from deep nets) there is a need of few feature points i.e. in few hundreds or thousands otherwise training on raw image will take many hours and resources [4].

In the experiment, PCA (Principal Component Analysis) was used for reducing the dimension of the train and test set. Number of principle components taken were 1000, so the dimension of image was reduced from 46656 to 1000 pixels. These PCA coefficients of images were then used for training the classifier. For testing test image (say 216x216), first PCA coefficients was computed or one can say that test image was projected on to the PCA space and then it was given to detector for prediction. Because classifier is trained in

PCA space, so testing has to be done in PCA space. Testing on micrograph follows (3.3) architecture, otherwise if overlapping patches are cropped on original micrograph then it will take many hours.

Dataset	Train set			Validation set			Test set (mg*)
	+ve	-ve	Total	+ve	-ve	Total	
T20S	6,000	12,000	18,000	2,072	8,288	10,360	40
80S	6,000	12,000	18,000	4,739	12,030	16,769	80
β-G	6,000	12,000	18,000	3,395	13,578	16,973	40

Table 3.3: Train and Test Dataset for SVM and Random forest (mg* stands for Micrograph)

3.5.5 Classifier - Support vector machine

For particle detector (3.3), a classifier is needed. One type of classifier is *Support vector machine* (SVM). In our experiments, SVM was used as one type of the classifier. For training the SVM, dataset was divided into train and test set. Division of the dataset is mentioned in the table 3.3. For training SVM, kernel used are linear, Radial Basis Function (RBF) and polynomial of degree 2 & 3. But accuracy given by linear kernel is much better than RBF and polynomial 2 & 3 kernel. Also, detector model used was downscale-model-12 for all the datasets, which means each image was first downsampled by 12 and then SVM was trained using the downsampled image.

Results

For training SVM with 18K images, training took couple of hours but the detection on full micrograph of dimension $\tilde{7}$000x7000 took 25-30 mins. Results for SVM with linear kernel are shown in table 3.4. Precision is very low as compared to the accuracy of the model but this is because for T20S-Proteasome, marking of particles per micrograph is very low i.e. micrograph contains thousands of particles but only few hundreds are marked.

Dataset	Recall	Precision	Translation error (pixel)			
			min.	max.	avg.	median
T20S	96%	16.96%	0	108	39	39
80S	96.6%	66.91%	0	108	25	22
β-G	94%	9.48%	0	105	24	20

Table 3.4: SVM Result

Table 3.4 shows the average result for the test micrographs (see table 3.3). The translation is very important. More the translation error, higher will be the reconstruction error if algorithm is not robust to translation error. So, the table shows the minimum, average, maximum and median translation error. Figure 3.19 shows the translation error histogram for all the particles.

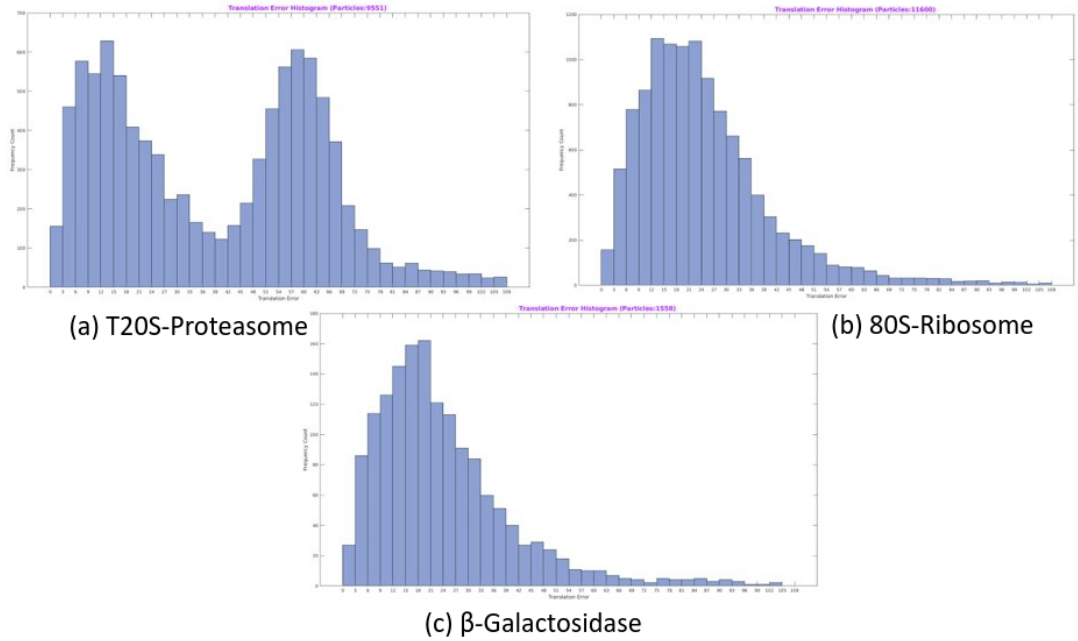


Figure 3.19: SVM-Translation Error Histogram

Micrographs 3.20 shows the particles marking on the test micrograph for all the datasets. Red mark is the predicted location by the SVM classifier and green is ground truth location. From the T20S-Proteasome and β-Galactosidase micrograph, one can observe that predicted particles (red marking) is much larger than (green marked) ground truth particles. Reason for this is that micrographs contains much more particles than the originally marked once and our classifier has marked other particles for which ground truth locations are unknown. Because of this the precision of the model is very low.

In case of 80S-Ribosome, dataset has marking for almost all the particles that's why its precision (table 3.4) is very good as compared to other dataset.

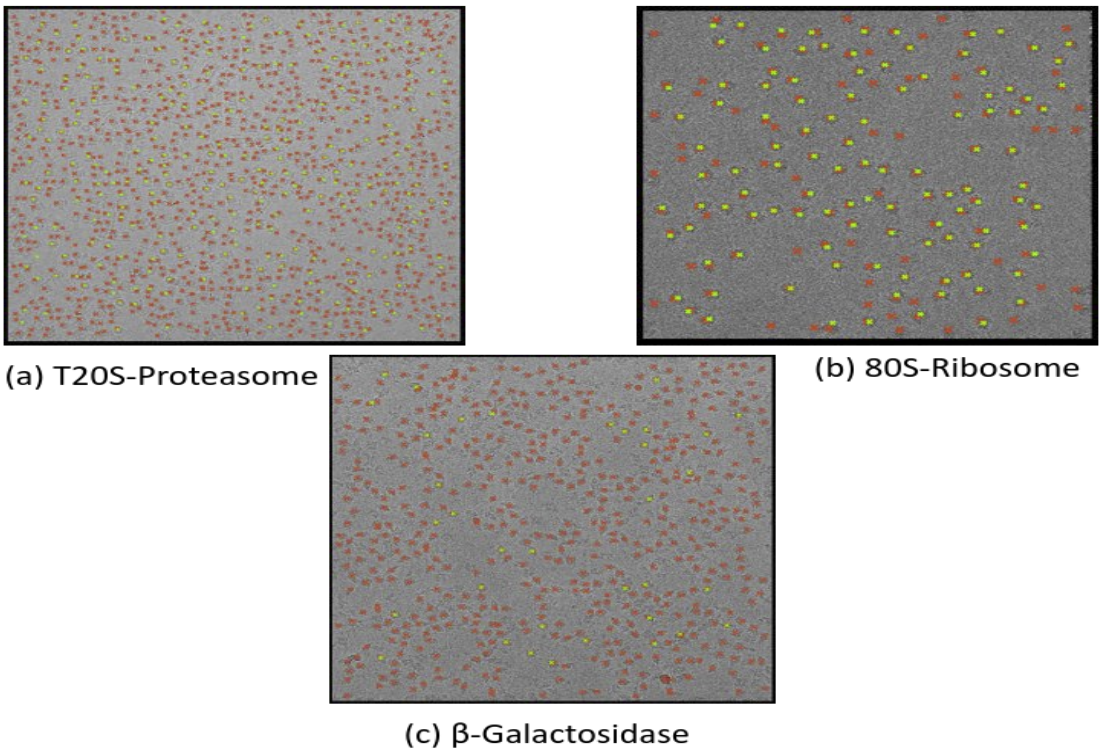


Figure 3.20: SVM- Particle marking Results. Red: Predicted location. Green: Ground Truth

3.5.6 Classifier - Random Forest

Random Forest classifier is trained on all the dataset with the same configuration. The size of train set is same for the three datasets which are T20S-Proteasome, 80S-Ribosome and β-Galactosidase (see table 3.3). Number of forest required for getting good accuracy in less amount of time are tuned. During tuning, it is found out that with 40 trees, the accuracy of classifier goes down and with 80 trees, detection time is large. Also if 50 trees are taken then accuracy of classifier was approximately same as of 80 trees and detection time is also less as compared to 80 trees. So, for all the dataset number of trees generated are 50.

Results

Time required to train Random Forest was approximately 15-20 mins whereas testing on whole micrograph took around couple of hours. The Reason for this is the large number of patches from the micrograph. As the model is downscale-model-12, so micrograph of

size 7420x7670 after downsampling by 12 will become 619x640. This test micrograph is given to the detector. Detector will then take out patches of dimension 18x18 (max size of particle is 216x216 and downsampled by 12 will give 18x18) with stride as 1, which will come out to 373822 patches. There are 50 trees and to test for each patch if it takes 2 sec (CPU based Random Forest) then total time comes out be be 8 hrs ($373822 * 2/(60 * 60 * 24)$). But, as our detector used multiple CPU for detection so with 10 core CPU overall time comes out to be 2.5-3 hrs including the meta data processing such as finding PCA of those patches before giving it to the Random Forest classifier.

Dataset	Recall	Precision	Translation error (pixel)			
			min.	max.	avg.	median
T20S	95%	23.5%	0	108	24	14
80S	96.5%	78.54%	0	108	27	24
β-G	99%	4.59%	1	103	27	23

Table 3.5: Random Forest Result

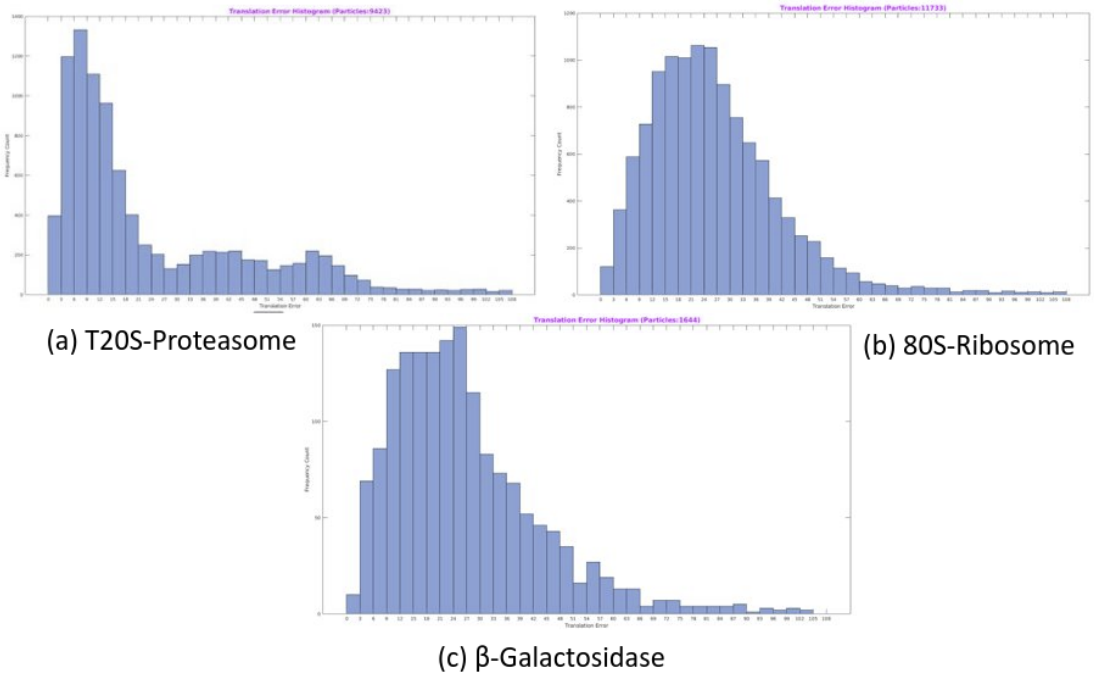


Figure 3.21: Random Forest-Translation Error Histogram

Table 3.5 shows the average result of the test micrographs. Apart from accuracy, the translation error is also very important. More the translation error higher will be the

reconstruction error. So, the table shows the minimum, average, maximum and median translation error. Figure 3.21 shows the translation error histogram of all the particles. From the histogram, one can conclude that there are only few particles having translation error more than 40 pixels. On an average, approx. 25 pixel is an average translation error for all the datasets.

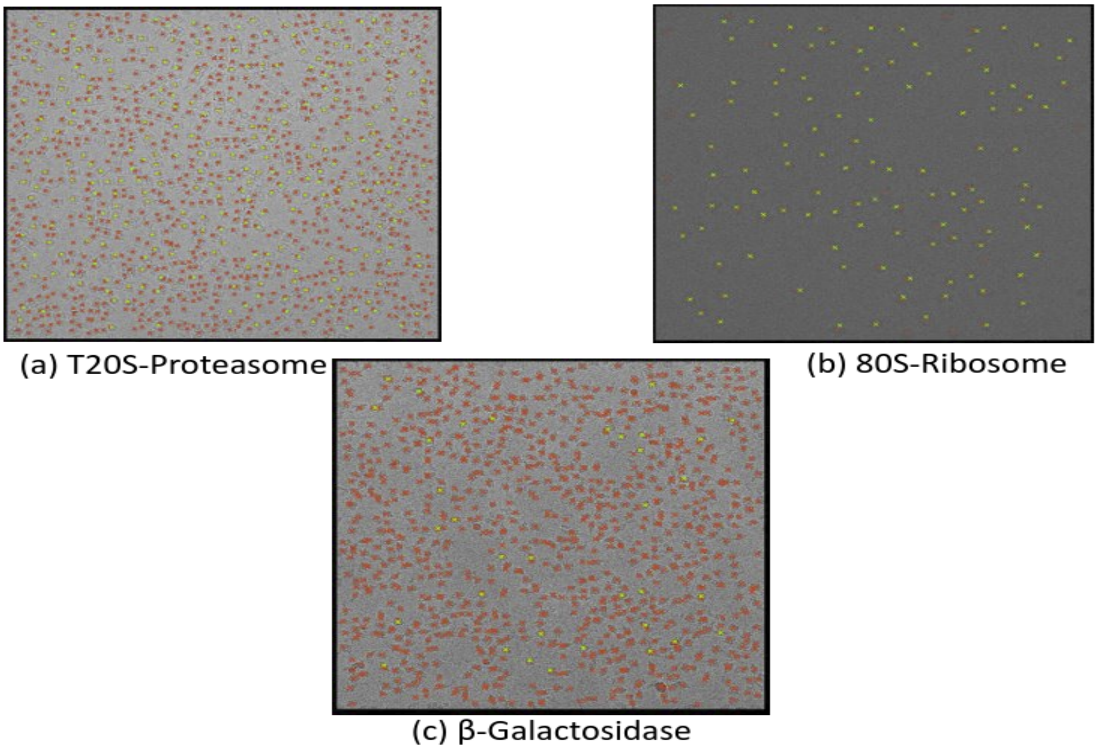


Figure 3.22: SVM- Particle marking Results. Red: Predicted location. Green: Ground Truth

Micrographs 3.22 shows the particle marking on the test micrograph for all the datasets. Red mark is the predicted location by the Random Forest classifier and green is ground truth location. One thing has be to be noted that our classifier also marked other particles for which ground truth locations were unknown. Because of this, the precision of the model is very low. But if marking for all the particles in micrograph are known then precision of the model will be good. In case of 80S-Ribosome, dataset has marking for almost all the particles that's why its precision (table 3.5) is very good as compared to other dataset.

3.5.7 Classifier - Faster-RCNN

Faster-RCNN is a variant of *Region based Convolution Neural Network*. It consists of two parts, first is the Region proposal Network (RPN) and another is the classifier and

regression network. Region proposal Network gives the probable area of where object can be present and output of this layer goes to classifier and regression network for classifying the region and finding the bounding box respectively. Unlike SVM and Random Forest, for training Faster-RCNN, the input data format is different. It requires image with bounding box (top left, top right, height, width) positions of the particle. Datasets T20S-Proteasome and 80S-Ribosome have x,y coordinate rather bounding box. See section 3.5.2 for how to generate train and test datasets.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Number of parameters (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

Figure 3.23: VGG-16: D column represent VGG-16 architecture

Faster-RCNN was trained using 4000 sub-micrographs of dimension (1000x1000) where each micrograph had approx. 3 to 4 particles bounding box marked. So, in total, number of particles were 15,831 for T20S-Proteasome dataset during training and for testing 40 micrographs (7420x7670). Firstly, a simple model with 11 layers CNN network was created and trained using downsampled sub-micrograph. Downsampled rate was 2, which means each 1000x1000 micrograph was downsampled to 500x500 before giving to network. After training this first model, it gave the accuracy of 54%. Then, second model was created by using transfer learning of AlexNet and VGG-16 [23] (D column of table in figure 3.23). Initial weights used for AlexNet and VGG-16 were the weights when these

model were trained using ImageNet. After training approximately for 5 hrs, accuracy obtained was around 2% from both the network which is extremely poor. There might be two probable reasons, firstly large amount of data is required to train the model as network is large and also it is pre-trained on ImageNet which do not have any class related to biological particle, more over these are tomographic projections. Secondly, the quality of input images i.e. input images are highly noisy and dark (see image (a) in figure 3.24).

(a) Before Pre-Processing

(b) After Pre-Processing

Figure 3.24: T20S-Proteasome: Sub-Micrograph (1000x1000). Pre-Processing includes Complement, Contrast enhancement and Noise Removal

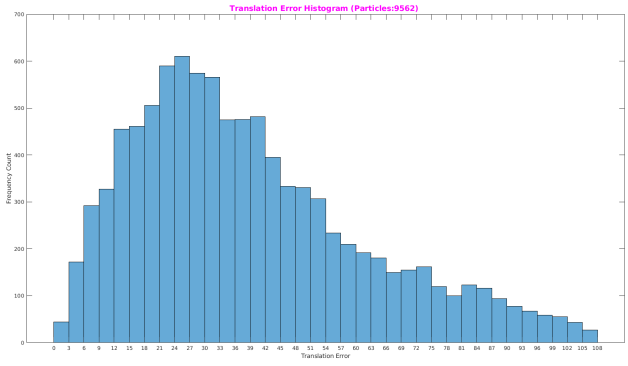


Figure 3.25: T20S-Proteasome: Translation Error Histogram

In paper [29], for training *Fast-RCNN*, contrast enhancement is done on input image before feeding to the network for reducing darkness affect. On the similar line, in our case each sub-micrograph is first passed through three level of pre-processing. First is the **complement** of the each image i.e each image is a gray scale image (i.e. one channel image) so in case of gray scale after doing the complement white will become black and vice-versa. The second step was to increase the contrast and for that **Histogram Equalization** method is used. The third and last step is to reduce the level of noise so for that **Wiener** filter of 5x5 is used using the mean of the local variance.

[13] ImageNet: http://image-net.org/

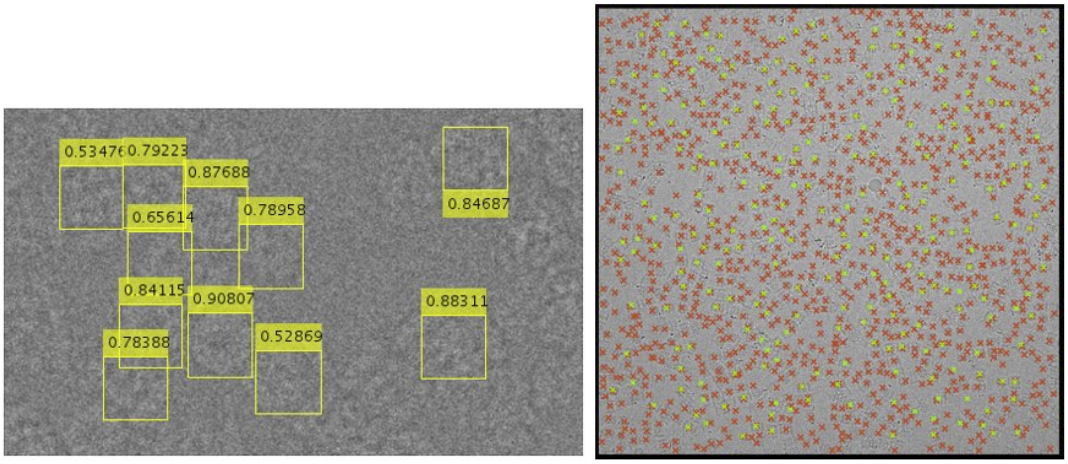
Results

Faster-RCNN was trained on one dataset which was T20S-Proteasome with 1000x1000 sub-micrographs using VGG-16 architecture. VGG-16 had 41 layers including the input and last layer. The learning parameter was set to $1 \times e^{-5}$. For anchor configuration, see table 3.6. The complete training took around 4 hrs using the pre-processed downsampled-by-4 sub-micrographs.

Dataset	Recall	Precision	Translation error (pixel)			
			min.	max.	avg.	median
T20S	96%	6%	0	108	39	34

Table 3.6: Faster-RCNN Result

Testing was performed on 40 micrograph each of dimension 7420x7670. During testing phase, when full micrograph (7420x7670) was directly passed then accuracy was poor. As the model was trained on 1000x1000 sub-micrographs which further downsampled by 4. So while testing, each test micrograph was divided into sub-micrographs of size 1000x1000 and then downsampled at 3 scale which were 4,3,2 before giving it to Faster-RCNN. Results of all the micrograph was then assembled into one and duplicated markings were removed using NMS (Non-Minimal Suppression). This gave the average accuracy of 40 micrographs as 96% and detection time was just 1.5 mins which was much better than SVM and Random Forest.



(a) Sub-Micrograph (1000x1000)

(b) Full Micrograph (7420x7670). Red: Predicted location. Green: Ground Truth

Figure 3.26: Faster-RCNN - T20S-Proteasome Result

Observation is that doing the pre-processing before giving to VGG-16 gives better result as compared to without pre-processing. When VGG-16 model was trained without pre-processing then accuracy obtained was approximately 2% but when pre-processing i.e. complement, contrast enhancement and noise reduction was performed then average result from V66-16 was 96%.

3.5.8 Conclusion

Three types of Particle-Picker are created using SVM, Random Forest and Faster-RCNN. In terms of *accuracy (Recall)*, all performed very well i.e. above 90%. With respect to *precision* on current dataset, Random Forest based model performed very well. But, judging any model based on *precision* [26] [29] for the current dataset will not be right as there are lots of particles for which markings are not present during the experiment. That means, precision is not the right parameter to judge the model. Another important parameter is *translation error* which is very important parameter w.r.t. 3D reconstruction i.e. less the translation error more right will be 3D reconstructed model. Here, Faster-RCNN (table 3.6) has average translation error of 39 pixels which is more than other two models. Between SVM (table 3.4) and Random-Forest (table 3.5), Random Forest has little less translation error. When parameter *detection time* is considered then Faster-RCNN is best because it took only 1.5 mins to test one 7420x7670 micrograph. On the other hand SVM took 25-30 mins. But CPU based Random Forest took 2-3 hrs. For the same size of micrograph which is very large time as compared to other two model.

Deciding which model is best just by one parameter will not be right. It is a mixture of accuracy, precision, translation error and detection time. An ideal model should have very high accuracy and precision. On the other hand, model should have very low translation error and detection time. Considering these datasets and these three models (SVM, Random Forest and Faster-RCNN) then, Faster-RCNN is doing very well because in terms of accuracy and detection speed, it is better than other two. Precision parameter is not considered as markings are incomplete. For translation error, approximately 15 pixel difference is there for the image of size 216×216 between Faster-RCNN and other two (SVM and Random Forest) which is not very large. But Random Forest is better than Faster R-CNN w.r.t. marking errors and translation errors. Because Faster R-CNN has marked many particles which are not actually a particle (in visual sense), also it has very large translation error. Quantitatively which model is better can only be decided when ground truth for all the particles are known.

Chapter 4

3D Reconstruction

The reason for doing particle picking is to 3D reconstruct the molecule using those markings. After the particle picking and before 3D reconstruction, there are few more steps involved such as Clustering, CTF correction, etc. (see section 2.2). 3D reconstruction is a challenging job because the angles and shifts of the projections are unknown. Without knowing the angles as well as shifts, right reconstruction is near to impossible. Tools like RELION use the Bayesian [20] approach for solving this problem and optimizes the equation in alternating fashion, i.e. by finding the rough estimate and then performing the back-projection (reconstruct the object). After that, it performs the forward projection at those angles for calculating the error, and this process is repeated.

Performing 3D parallel beam filtered back-projection becomes harder because of unavailability of libraries. For 2D parallel beam filtered back-projection, radon and iradon are available. But in case of 3D, things get more challenging because of the parallel beam, as in the field of tomography cone beam is widely used but the parallel beam is used only in few applications such as in Cryo-EM. But there is a toolbox named *ASPIRE* developed by A.Singer and group from Princeton University and *TIGRE* developed by CERN which can perform "3D parallel beam filtered back-projection". The next section will talk about the ASPIRE and TIGRE toolboxes features and how these can be used for 3D reconstruction.

4.1 ASPIRE

ASPIRE[1] stands for "**A**lgorithms for **S**ingle **P**arti**c**le **Re**construction". This toolbox is developed by then Amit Singer and his group of Princeton University for the use of

[1] ASPIRE: http://spr.math.princeton.edu/

Cryo-EM. This toolbox supports the parallel beam 3D radon reconstruction along with the common line and angle estimate for the single-particle projections and many more features.

ASPIRE is a good toolbox to start with, but it has one major drawback. For a 3D reconstruction of the object of size 64x64x64 with 500 projections, given angles, ASPIRE takes minutes (approx. 20 to 30 min or more depending on the computer). The 3D reconstruction doesn't use the GPU for faster reconstruction, which makes this library hard to used for making fast algorithms.

4.2 TIGRE

TIGRE[2] stands for "**T**omographic **I**terative **G**PU-based **Re**construction Toolbox". As the name suggests, it uses GPU (Graphical Processing Unit) for reconstruction. 3D reconstruction is computationally expensive so, if GPU is not used, then it will take many hours depending on the dimension of the object to be constructed. For reconstructing the 3D model of dimension 160x160x160 using 100 projections, TIGRE took only a few seconds, which is very good w.r.t ASPIRE. One thing that has to be considered while reconstructing is that bigger the dimension, larger will be the reconstruction time and main memory (RAM + GPU) required will be more.

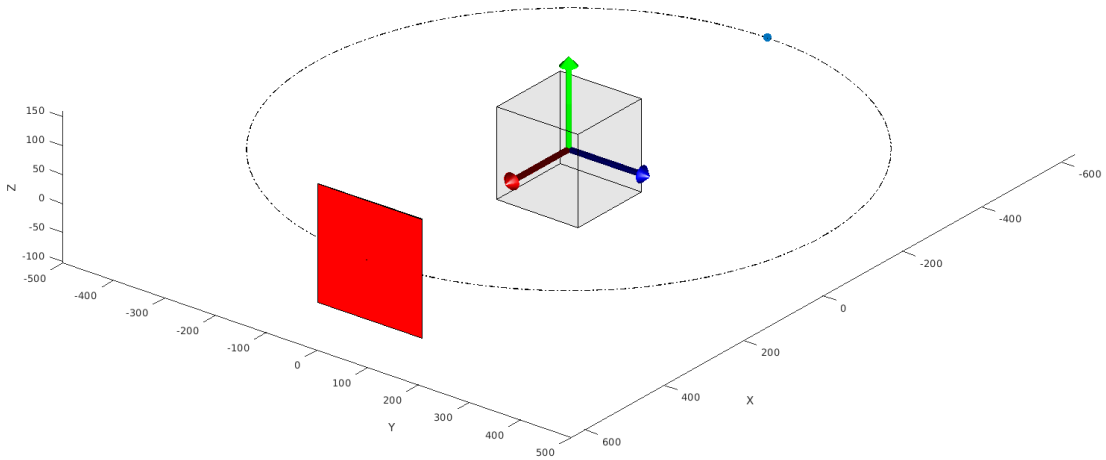


Figure 4.1: Figure represent placement of object, source and detector. Detector is at distance 1000 mm and source is at 500 mm. Blue dot represent the source and Red quadrilateral is detector.

[2]TIGRE: https://github.com/CERN/TIGRE

TIGRE toolbox has many libraries for parallel beam as well as cone beam but in our application parallel beam is required, as the electron microscope uses parallel beam. Before acquiring projections or 3D reconstruction, one needs to create "geometry". In the case of acquiring projections, "geometry" will have information related to source distance and detector distance from the object when the object is placed at origin (see image (4.1)). Geometry will also have information related to detector dimension, detector pixel size, the accuracy of interpolation, and the dimension of the object. By using library function $Ax(object, geometry, angles,' interpolated')$ of TIGRE, projections at the specific angle can be taken. For 3D reconstruction one can use library function $FBP(projections, geometry, angles)$ for filtered back-projection and BP (Back-Projection) one can use "Atb (projections, geometry, angles)".

Next section will talk about our proposed algorithm for finding angles and shift of the projections. Once these are known, then using TIGRE library function reconstruction can be done.

4.3 Proposed Algorithm

This algorithm does not uses the Bayesian approach like RELION [20]; it's correlation based optimization. Our algorithm is divided into four major parts, i.e., noise removal from projections, common line estimate, angle initialization, and correlation optimization algorithm for find

4.3.1 Noise Removal

In Cryo-EM, micrographs and projections are highly noisy. They are so noisy that without doing any pre-processing on micrographs, seeing projections perfectly with the naked eyes are near to impossible. That's why biologist does not prefer crowdsourcing of particle picking [3] much. Also, if noisy projections are used for 3D reconstruction, then the reconstructed object will also be noisy, and there is a high possibility that the object will be wrongly constructed. As for 3D reconstruction, angles and shifts for projections should be known. But here, these are unknown initially. So, before reconstruction, initial angles and shifts are computed using the projections. If projections are noisy, then estimated angles and shifts will not perfect, which further leads to the wrong reconstruction.

There are various ways this noise can be removed. One of the most popular ways is to use clustering [14]. But for clustering to work, there is a need of 30 to 50 thousands of projections. Larger the number projections larger will be the probability that more

projections will lie in one cluster. More the number of projections per clusters less noisy will be the averaged projection. Another way for removing the noise is to use the Wiener filter, BM3D method, etc.

BM3D

BM3D [5][10] is the collaborative filtering for the noise removal for 2D image. It is a two step noise reduction algorithm. It is a patch based algorithm and each stage is divided into the four modules :

1. Finding the similar patches to a given patch in the image and collaborate it in a 3D matrix or block
2. 3D Linear transformation of the 3D block
3. Shrinkage the transform coefficients also called as Thresholding or perform the Wiener filtering
4. 3D inverse transformation

The only difference between the two stages is the third module, i.e., in first step thresholding is used and in the second step, Wiener filter is used (refer the figure (4.2). Step 1 is first executed on the whole image, and then step 2 is performed. After each step, the 3D filtered patch is obtained, which the reassigned to its original location. As overlapping patches are used, each pixel has multiple values. So to handle the overlapping case, *aggregation* is performed at each pixel. BM3D works on the fact that the image has a sparse local representation in the transformation domain. Also collaborative filtering reveals the finest detail shared by the similar patches.

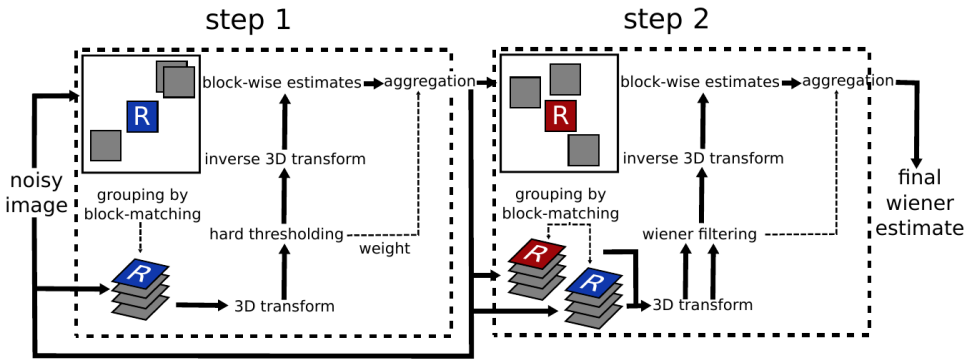


Figure 4.2: BM3D Algorithm Pipeline Source: [10]

For noise removal, in our experiments, BM3D is used for Gaussian noise removal instead of the clustering, because clustering based noise removal require a large number of projections. Also, if one is using K-Means clustering, then the image cannot be used directly because vectorised image may be very large. Using raw images in K-Means with L2 norm or cosine distance will leads to lose of spatial information. So, to overcome that, one can use *Auto-Encoder* for reducing the dimension.

4.3.2 Common Line

Projections from Cryo-EM are noisy and have unknown angles and shifts. Unknown angle problem arises due to the random orientation of Bacteria/Virus/Ribosome etc. in the sample under study. And the unknown shift problem arises due to the wrong center marking of the projections in the micrograph. Also, reconstruction required angles and shifts to be known. So for the rough initial estimate of the angles and shifts, prior information related to projections can be used.

In the case of 3D, one can do a smarter initialization by exploiting the properties of 3D geometry. Fourier slice theorem (1.1.3) says that if object is 3D and its 2D projection is taken at some angle say (α, β, γ), then Fourier transform of that 2D projection will be equal to the slice passing through the origin at angle (α, β, γ) of the Fourier transform of the 3D object. Fourier slice theorem gives the key information in relation with angle initialization. Let's consider there are two 2D projections 1 & 2 taken at different angles. Take the Fourier transform of these two projections. Now by Fourier slice theorem, a relative slice of projection 1 & 2 in the Fourier domain of the 3D object will pass through the origin. Assuming these two slices are non-coplanar then these two slice will intersect at a line which is called as *common line* [14] [25] (see figure 4.3). This logic can be extended to any number of projections, but the only constraint is that they should be non-coplanar.

Finding a common line between two projections will help in finding the rough initial angle estimate of projections. Let the common line between the projection p_i & p_j is denoted by c_{ij}. ϕ_{ij} denotes the angle made by the common line c_{ij} with the local coordinates system of the projection p_i. Similarly, ϕ_{ji} denotes the angle made by the common line c_{ij} with the local coordinates system of the projection p_j.

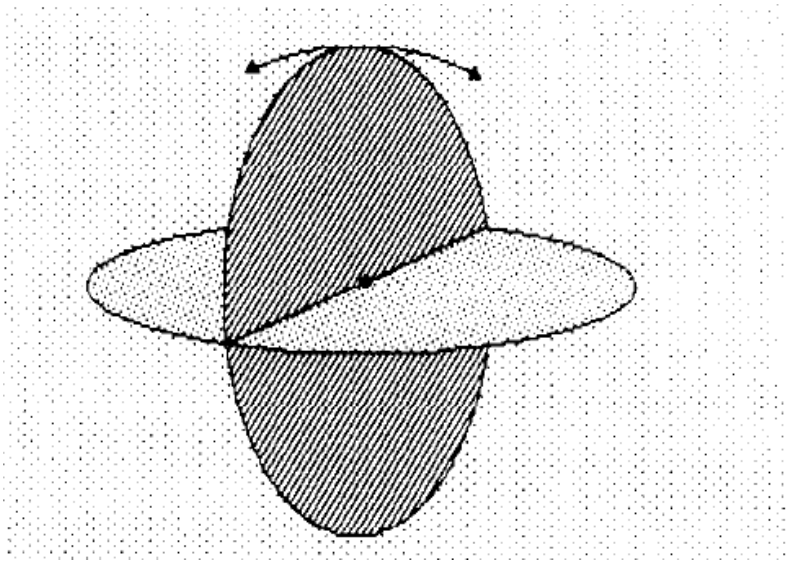


Figure 4.3: Common line: Intersection of Fourier slice of two projections in Fourier Space. Black dot represent the origin of the Fourier Space. Source: [25]

$$\Phi = \begin{bmatrix} \phi_{11} & \phi_{12} & \phi_{13} & \dots & \phi_{1n} \\ \phi_{21} & \phi_{22} & \phi_{23} & \dots & \phi_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \phi_{n1} & \phi_{n2} & \phi_{n3} & \dots & \phi_{nn} \end{bmatrix}$$

For computing ϕ_{ij}, common line between the projections p_i & p_j needs to be found out. Finding a common line can be done in the Fourier domain or in the spatial domain. For finding a common line in the spatial, below steps can be followed:

1. Define the angle search space meaning up to what accuracy level of decimal precision in ϕ_{ij} you want. For simple understanding, assume ϕ_{ij} to be integer rather than real.

2. Divide the angles from 0 to 360 by 1 degree. So we will get 360 angles, let's call it n angles.

3. Take 1D Radon transform for p_i & p_j at all these n angles. These 1D projections are following the principle of Fourier slice theorem. Working in a spatial domain helps in avoiding complex number, also for interpolation in Fourier domain one is required to use NUFFT (nonuniform fast Fourier transform) [7].

4. Create a nxn matrix, here it will be $360x360$ normalized correlation matrix. Where, each ith row denotes the ith degree starting from 0° with increment by 1 (search

space is a hyperparameter), similarly for the jth column. Each ij denotes the normalized correlation of the i^{th} 1D radon projection for p_i with the j^{th} 1D radon projection of the p_j

5. Find the ij^{th} entry for which normalized correlation value is the maximum. That ij^{th} entry, i^{th} row angle will be the ϕ_{ij} and j^{th} column angle will be the ϕ_{ji}

4.3.3 Angle Initialization

Finding projections angles using our algorithm requires some initial angles estimate for all projections at the start of the algorithm. One way to initialize angles is random initialization or use some prior knowledge about projections.

By utilizing the property of common line, initial angles estimate or rotation matrix for projections can be calculated using [14], [24] [22] or [28]. Our experiments uses the [24]. It uses the Φ (see eq. (4.3.2) for the estimation of rotation matrices instead of projections angles. But once the rotation matrix is know one can easily find out the angles. Here, in our experiments for making thing consistent at every stage *Euler-XYZ* rotation representation is used (see eq. (4.1)), where $\boldsymbol{R_x}$, $\boldsymbol{R_y}$ & $\boldsymbol{R_y}$ means rotation about rotation x-axis, y-axis and z-axis respectively.

$$\boldsymbol{R} = \boldsymbol{R_x} \times \boldsymbol{R_y} \times \boldsymbol{R_z} \tag{4.1}$$

$$\boldsymbol{c_{ij}} = \begin{bmatrix} \cos\phi_{ij} \\ \sin\phi_{ij} \\ 0 \end{bmatrix} = \begin{bmatrix} x_{ij} \\ y_{ij} \\ 0 \end{bmatrix} \tag{4.2}$$

$$\boldsymbol{R_i}.\boldsymbol{c_{ij}} = \boldsymbol{R_j}.\boldsymbol{c_{ji}} \;\; \forall\, 1 \lesssim i \lesssim j \lesssim N \tag{4.3}$$

In [22], below eq. (4.4) is optimized by exploiting the fact of equality 4.3. $\boldsymbol{R_i}$ denotes the true rotation matrix for the projection p_i. $\boldsymbol{c_{ij}}$ denotes the common line between the projections p_i & p_j w.r.t. the local coordinate system of p_i, defined as eq. (4.2), similarly for $\boldsymbol{c_{ji}}$. In eq (4.4), if $\boldsymbol{R_i}$ is a true rotation matrix then maximum sum will be N, i.e., number of projections, else it will be less than that.

$$\begin{aligned}
&\max_{R_1 \ldots R_N} \sum_{i \neq j} \boldsymbol{R_i c_{ij}} \cdot \boldsymbol{R_j c_{ji}} \\
&subject\ \ to, \\
&\boldsymbol{R_i R_j} = \boldsymbol{I} \\
&\mathbf{det}\ \boldsymbol{R_i} = 1 \\
&\forall\ \ i = 1, 2 \cdots, N
\end{aligned} \tag{4.4}$$

Optimizing eq. (4.4) directly is computationally very expensive. So, if certain things are assumed, i.e. some prior information, then some constraints can be relaxed, which further help in solving the equation. Eq. (4.4) is a quadratic in unknown rotation $\boldsymbol{R_1} \cdots \boldsymbol{R_N}$, so if constraints are relaxed properly, then solution to the optimization problem would be related to the top eigen vectors of the matrix defining quadratic form (refer [24]).

Lets define 4 $N \times N$ matrices $\boldsymbol{S^{11}}, \boldsymbol{S^{12}}, \boldsymbol{S^{21}}, \boldsymbol{S^{22}}$ define as

$$\begin{aligned}
\boldsymbol{S^{11}_{ij}} &= x_{ij}\, x_{ji} \\
\boldsymbol{S^{12}_{ij}} &= x_{ij}\, y_{ji} \\
\boldsymbol{S^{21}_{ij}} &= y_{ij}\, x_{ji} \\
\boldsymbol{S^{22}_{ij}} &= y_{ij}\, y_{ji} \\
&for,\ 1 \lneq i \neq j \lneq N, diagonal\ set\ to\ zero
\end{aligned} \tag{4.5}$$

Then, $\boldsymbol{S}$ of dimension $2N \times 2N$ is create by combing the all the four matrices $\boldsymbol{S^{11}}, \boldsymbol{S^{12}}, \boldsymbol{S^{21}}, \boldsymbol{S^{22}}$.

$$\boldsymbol{S} = \begin{bmatrix} \boldsymbol{S^{11}}, \boldsymbol{S^{12}} \\ \boldsymbol{S^{21}}, \boldsymbol{S^{22}} \end{bmatrix}_{2N \times 2N} \tag{4.6}$$

Lets denotes the column of the rotation matrix $\boldsymbol{R}$ by $\boldsymbol{r^1}, \boldsymbol{r^2}\&\boldsymbol{r^3}$ and it can be written as

$$\boldsymbol{R_i} = \left[\boldsymbol{r^1_i}, \boldsymbol{r^2_i}, \boldsymbol{r^3_i}\right]_{3\times3} = \begin{bmatrix} x^1_i, & y^1_i, & z^1_i \\ x^2_i, & y^2_i, & z^2_i \\ x^3_i, & y^3_i, & z^3_i \end{bmatrix}_{3\times3} \tag{4.7}$$

Also lets define $x, y\&z$ as

$$\boldsymbol{x} = \begin{bmatrix} x_1^1 \\ \vdots \\ x_N^1 \\ x_1^2 \\ \vdots \\ x_N^2 \end{bmatrix}_{2N\times 1} \qquad \boldsymbol{y} = \begin{bmatrix} y_1^1 \\ \vdots \\ y_N^1 \\ y_1^2 \\ \vdots \\ y_N^2 \end{bmatrix}_{2N\times 1} \qquad z = \begin{bmatrix} z_1^1 \\ \vdots \\ z_N^1 \\ z_1^2 \\ \vdots \\ z_N^2 \end{bmatrix}_{2N\times 1} \tag{4.8}$$

So, by using the equations (4.6) & (4.6) the eq. (4.4) can be rewritten as

$$\begin{aligned} &\max_{\boldsymbol{R_1 \ldots R_N}} \ \boldsymbol{x^T S x} + \boldsymbol{y^T S y} + \boldsymbol{z^T S z} \\ &sub ject \ \ to, \\ &\boldsymbol{R_i R_j} = \boldsymbol{I} \\ &\mathbf{det}\, \boldsymbol{R_i} = 1 \\ &\forall \ \ i = 1, 2 \cdots, N \end{aligned} \tag{4.9}$$

By relaxing the constraints, the optimization equation can be rewritten as

$$\max_{\|x\|=1} \ \boldsymbol{x^T S x} \tag{4.10}$$

The relaxing of the constraint is only possible if unknown rotations are sampled from the uniform distribution. Then optimizing of eq. (4.10) using the eigen value decomposition $\boldsymbol{S}$ will give the initial rotation estimate of projections, lets call this initial estimate as $\boldsymbol{R_i^{init}}$ for projection *pi*. Using this $\boldsymbol{R_i^{init}}$ initial 3D reconstruction is done, let call this initial 3D reconstruction as $\boldsymbol{G^{init}}$ and $\boldsymbol{F}$ been the true object.

4.3.4 Correlation Optimization

Our *Correlation Optimization* algorithm can be used for further correcting the rotation and translation estimate found in section (4.3.3) for *Angle Recovery Problem (ARP)* and *Shift-Angle Recovery Problem (ShARP)*. This section talks about the refinement of rotation estimate in ARP assuming no translation error and the translation and rotation estimate in ShARP. In ARP, it is assumed that there is no translation error in the projections, and they are center aligned. In ShARP, no assumptions are taken into consideration w.r.t rotation and shift.

Angle Recovery Problem

Angle recovery problem assumes that there is no translation error, so only orientations for projections will be estimated. Our algorithm tries to optimize the eq. 4.11 iteratively for

all projections till convergence. In eq. (4.11), $\boldsymbol{NCor(x, y)}$ will finds out the *normalized correlation* between the image x & y. $\boldsymbol{Radon(G, ang_{1\times3})}$ will find out the 3D radon transform of the 3D object G at the given angle $ang_{1\times3}$.

$$
\begin{aligned}
(\theta_i^1, \theta_i^2, \theta_i^3, \boldsymbol{G}) = argmax_{\theta^1,\theta^2,\theta^3,\boldsymbol{G}}\ & \boldsymbol{NCor}(p_i^{true}\ ,\ \boldsymbol{Radon}(\boldsymbol{G}, [\alpha+\theta^1, \beta+\theta^2, \gamma+\theta^3])) \\
& where, \\
& -k \lesneq \theta^1 \lesneq k \\
& -k \lesneq \theta^2 \lesneq k \\
& -k \lesneq \theta^3 \lesneq k \\
& \forall\ i = 1, 2 \cdots, N \\
& \alpha : angle\ with\ x-axis, \\
& \beta : angle\ with\ y-axis, \\
& \gamma : angle\ with\ z-axis, \\
& k : search\ area\ hyperparameter \\
& N : number\ of\ projections
\end{aligned}
\tag{4.11}
$$

Algorithm (1) tells, how to solve optimization eq. (4.11) as a closed form solution does not exist. So for solving this, iterative approach is used. The only hyperparameter is k, which will define the final angles predictions. If the initial estimates are very good, then the value of k can be less. But if the initial angle estimate is bad, then the large value for k can be set. Value of k defines the execution time of the algorithm, i.e., if the k is very large then just to complete one iteration for all N projections, it will take a lot of time. So, it's advisable to set k to 5 to 10 because even if the initial estimate is bad, the algorithm will reach its convergence but slowly.

Input: All $\boldsymbol{P}$ projections, $\boldsymbol{G^{init}}$, $\boldsymbol{R_i^{init}}$ & k

Output: Returns estimated $\boldsymbol{G^{est}}$ & $\boldsymbol{R_i^{est}}$

Initialization:

$\boldsymbol{G^{est}} = \boldsymbol{G^{init}}$

$\boldsymbol{R_i^{est}} = \boldsymbol{R_i^{init}}$

while *till convergence* **do**

  while *i: 1* $\rightarrow N$ **do**

    $\alpha, \beta, \gamma = \boldsymbol{Rot2Euler}\ (\ \boldsymbol{R_i^{est}})$

    $(\theta_i^1, \theta_i^2, \theta_i^3) = argmax_{\theta^1,\theta^2,\theta^3}$

      $\boldsymbol{NCor}(p_i^{true}\ ,\ \boldsymbol{Radon}(\boldsymbol{G^{est}}, [\alpha + \theta^1, \beta + \theta^2, \gamma + \theta^3]))$

      $where,\ -k \lesssim \theta^1, \theta^2, \theta^3 \lesssim k$

    $\boldsymbol{R_i^{est}} = \boldsymbol{Euler2Rot}\ (\ [\alpha + \theta_i^1, \beta + \theta_i^2, \gamma + \theta_i^3]\ )$

  end

  $\boldsymbol{G^{est}} =$ **invRadon** $(\ P^{true}_{1\cdots N}, \boldsymbol{R^{est}_{1\cdots N}})$

  $P_{1\cdots N} =$ **Radon** $(\ \boldsymbol{G^{est}}, \boldsymbol{R^{est}_{1\cdots N}})$

  $error =$ **NCor**$(P_{1\cdots N}, P^{true}_{1\cdots N})$ (4.12)

end

Algorithm 1: ARP

The algorithm (1) is divided into two parts, first is to use the estimated reconstructed the object for finding the approximately correct angles of projection, and second is to use the estimated projections angles for reconstructing the object back using the true projections. This process is repeated till convergence, i.e., until there is no change in the estimated projections angles or rate of change of *error* is very small. Final $\boldsymbol{G^{est}}$ is the final reconstructed object.

Shift-Angle Recovery Problem

ShARP optimization problem does not assume anything on projections angles and shifts error. Shift error is introduced at the particle picking stage irrespective of the method used. Automated, as well as the manual methods of marking projections in the micrograph, are not 100 percent perfect, i.e., projections will be shifted by some pixels from their true center. So for 3D reconstruction, the shifts should also be known along with the angles

else reconstruction will not be possible. If anyhow marked projections are perfect, i.e. without any shift error then ARP algorithm 1 can be used instead of ShARP algorithm.

ShARP algorithm tries to optimize the eq. (4.13) for projection angles and their shift simultaneously. In eq. 4.13, $\boldsymbol{NCor(x,y)}$ will finds out the *normalized correlation* between the image x & y. $\boldsymbol{Radon(G, ang_{1\times3})}$ will find out the 3D radon transform of the 3D object G at the given angle $ang_{1\times3}$. $\boldsymbol{Shift(img, [t_x, t_y])}$ will shift the image img by t_x in x-axis and t_y in y-axis from the origin. In our all experiments, origin is considered to be center of the projections.

$$
\begin{aligned}
(\theta_i^1, \theta_i^2, \theta_i^3, \boldsymbol{G}, t_x, t_y) = argmax_{\theta^1,\theta^2,\theta^3,\boldsymbol{G},t_x,t_y} \\
&\boldsymbol{NCor}(\, \boldsymbol{Shift}(p_i^{true}, [t_x, t_y])\ ,\ \boldsymbol{Radon}(\boldsymbol{G}, [\alpha+\theta^1, \beta+\theta^2, \gamma+\theta^3])) \\
&where, \\
&-k \lesssim \theta^1, \theta^2, \theta^3 \lesssim k \\
&\forall\ \ i = 1, 2 \cdots, N \\
&\alpha : angle\ with\ x-axis, \\
&\beta : angle\ with\ y-axis, \\
&\gamma : angle\ with\ z-axis, \\
&t_x : shift\ in\ x-axis, \\
&t_y : shift\ in\ y-axis, \\
&k : search\ area\ hyperparameter \\
&N : number\ of\ projections
\end{aligned}
\tag{4.13}
$$

Eq. (4.13) do not have any closed form solution, so interactive approached is used for finding the probable solution. ShARP algorithm is somewhat similar to the ARP algorithm (1) except the shift estimation. For estimating the shifts, *Cross-Power Spectrum* [18] is used. Cross-Power spectrum methods will perfectly tells the shift between images. Let say there are two images f_1 & f_2 which is given by translation eq. (4.14) then cross-power spectrum of f_1 & f_2 will give the shifts i.e t_x & t_y.

$$
f_1(x, y) = f_2(x - t_x, y - t_y) \tag{4.14}
$$

Cross-Power spectrum for two image f_1 & f_2 with Fourier Transform (FT) F_1 & F_2 is defined as,

$$
e^{i2\pi(\xi t_x + \eta t_y)} = \frac{F_1(\xi, \eta) F_2^*(\xi, \eta)}{|F_1(\xi, \eta) F_2(\xi, \eta)|} \tag{4.15}
$$

where F_2^* denotes the complex conjugate of F_2. After finding the cross-power spectrum, inverse Fourier transform will the shift offset t_x & t_y.

Input: All $\boldsymbol{P}$ projections, $\boldsymbol{G^{init}}$, $\boldsymbol{R_i^{init}}$, $\boldsymbol{S_i^{init}}$ & k
Output: Returns estimated $\boldsymbol{G^{est}}$, $\boldsymbol{R_i^{est}}$ & $\boldsymbol{S_i^{est}}$
Initialization:
$\boldsymbol{G^{est}} = \boldsymbol{G^{init}}$
$\boldsymbol{R_i^{est}} = \boldsymbol{R_i^{init}}$
$\boldsymbol{S_i^{est}} = \boldsymbol{S_i^{init}}$
$\boldsymbol{p'_i} = \boldsymbol{p_i^{true}}$
while *till convergence* **do**
  while *i: 1* $\rightarrow N$ **do**
    $\alpha, \beta, \gamma = \boldsymbol{Rot2Euler}\ (\ \boldsymbol{R_i^{est}})$

$$(\theta_i^1, \theta_i^2, \theta_i^3) = \ argmax_{\theta^1,\theta^2,\theta^3}$$
$$\{$$
$$p_{tmp} = \ \boldsymbol{Radon}(\boldsymbol{G^{est}}, [\alpha + \theta^1, \beta + \theta^2, \gamma + \theta^3])$$
$$[t'_x, t'_y] = \boldsymbol{CrossPowerSpectrum}(p'_i\ ,\ p_{tmp})$$
$$\boldsymbol{NCor}(\ \boldsymbol{Shift}(p'_i, [t'_x, t'_y])\ ,\ p_{tmp})$$
$$\}$$
$$-k \lesssim \theta^1, \theta^2, \theta^3 \lesssim k$$

    $p_{tmp} = \ \boldsymbol{Radon}(\boldsymbol{G^{est}}, [\alpha + \theta_i^1, \beta + \theta_i^2, \gamma + \theta_i^3])$
    $[t_x, t_y] = \boldsymbol{CrossPowerSpectrum}(p'_i\ ,\ p_{tmp})$
    $p'_i = \boldsymbol{Shift}(p'_i\ , [t_x, t_y])$
    $\boldsymbol{S_i^{est}} = \ \boldsymbol{S_i^{est}} + [t_x, t_y]$
    $\boldsymbol{R_i^{est}} = \boldsymbol{Euler2Rot}\ (\ [\alpha + \theta_i^1, \beta + \theta_i^2, \gamma + \theta_i^3]\ )$
  end
  $\boldsymbol{G^{est}}$ = **invRadon** ($P'_{1\cdots N}$, $\boldsymbol{R^{est}_{1\cdots N}}$)
  $P^{tmp}_{1\cdots N}$ = **Radon** ($\boldsymbol{G^{est}}$, $\boldsymbol{R^{est}_{1\cdots N}}$)
  $error$ = **NCor**($P^{tmp}_{1\cdots N}, P'_{1\cdots N}$) (4.16)
end

Algorithm 2: ShARP

In algorithm (2), $\boldsymbol{CrossPowerSpectrum(x, y)}$ denotes the inverse Fourier transform of the cross-power spectrum of x & y. Unlike ARP algorithm [1], Sharp algorithm [2] works

in three phases, i.e., finding of the projection angles, shifts, and then the 3D reconstruction. The finding of angles and shifts are done simultaneously. After that object is reconstructed using the *shifted* projections. But the algorithm initially requires the rough initial estimate of 3D reconstruction G, the initial angles, and shifts.

4.4 Experimental Results

4.4.1 Dataset: EMDB

Dataset used in our simulated experiments are taken from the "**E**lectron **M**icroscopic **D**ata **B**ank"[3]. Our proposed algorithms ARP and ShARP are tested on these two datasets EMD-8647[4] & EMD-4138[5].

EMD-8647 (figure 4.4) is a ribosome from Mycobacterium smegmatis, i.e., the human pathogen Mycobacterium tuberculosis. It's a ribosome of voxel dimension $2.5 \times 2.5 \times 2.5$ Åand map dimensions as $128 \times 128 \times 128$, but for experiments purpose, it's being downsampled by 2. So, throughout all the experiments, the map dimension will be $64 \times 64 \times 64$.

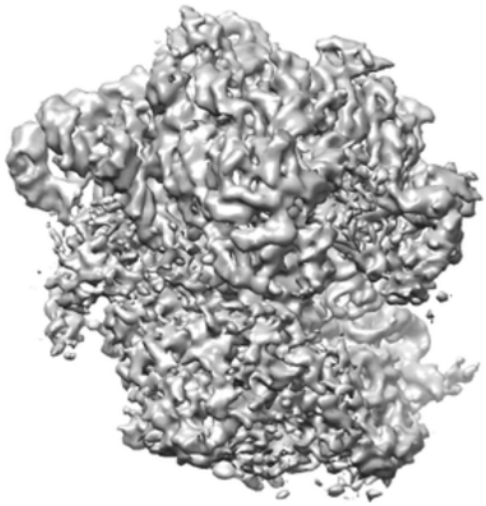


Figure 4.4: Mycobacterium Smegmatis: Resolution 2.5 Å

EMD-4138 (figure 4.5) is a **M**aedi-**V**isna **V**irus (MVV) intasome. It is having the resolution of 4.94Å and the map dimension as $300 \times 300 \times 300$. Its map has a lot of extra empty padding in 3D, so one can use it directly or remove this empty padding. In

[3]EM-Databank: http://emdatabank.org
[4]EMD-8647: https://www.emdataresource.org/EMD-8647
[5]EMD-4138: http://www.ebi.ac.uk/pdbe/entry/emdb/EMD-4138

our all experiments, cropped mapped is used rather than full map, so after cropping the dimension is $161 \times 161 \times 161$.

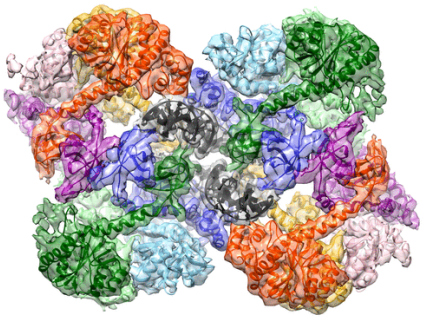
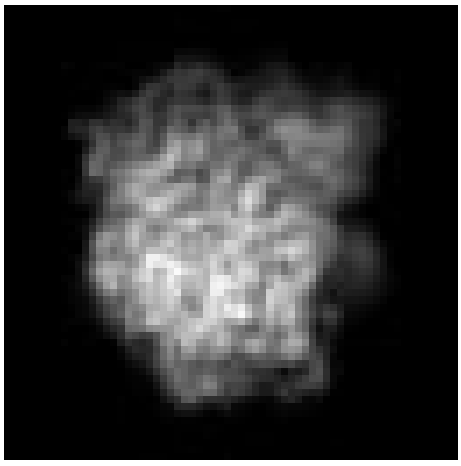


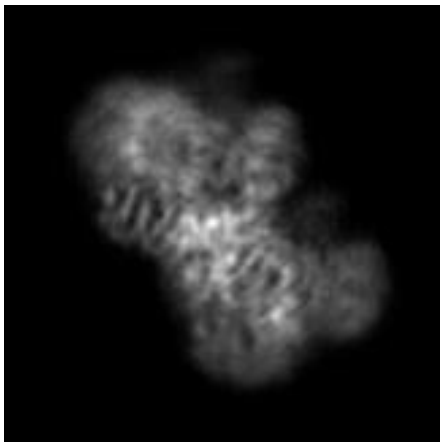
Figure 4.5: Maedi-Visna Virus (MVV) intasome: Resolution 4.94 Å

4.4.2 Dataset Generation

3D reconstruction is performed using the 2D projections taken from the EM, but as there is unavailability of EM or micrographs for & Mycobacterium Smegmatis & Maedi-Visna Virus these digital EMD-8647 & EMD-4138 have been used. For taking projections from EMD-8647 & EMD-4138 computer simulated tool TIGRE (refer section 4.2) is used. Two types of dataset is being generated, i.e., one without any shifts in the projections and one with the shifts. Figure (4.6a) & (4.6b]) shows the sample projection for EMD-8647 & EMD-4138 taken at the angles $[-35°, 62° - 115°]$ & $[-86°, 43°, 50°]$ respectively.



(a) EMD-8647: A sample projection



(b) EMD-4138: A sample projection

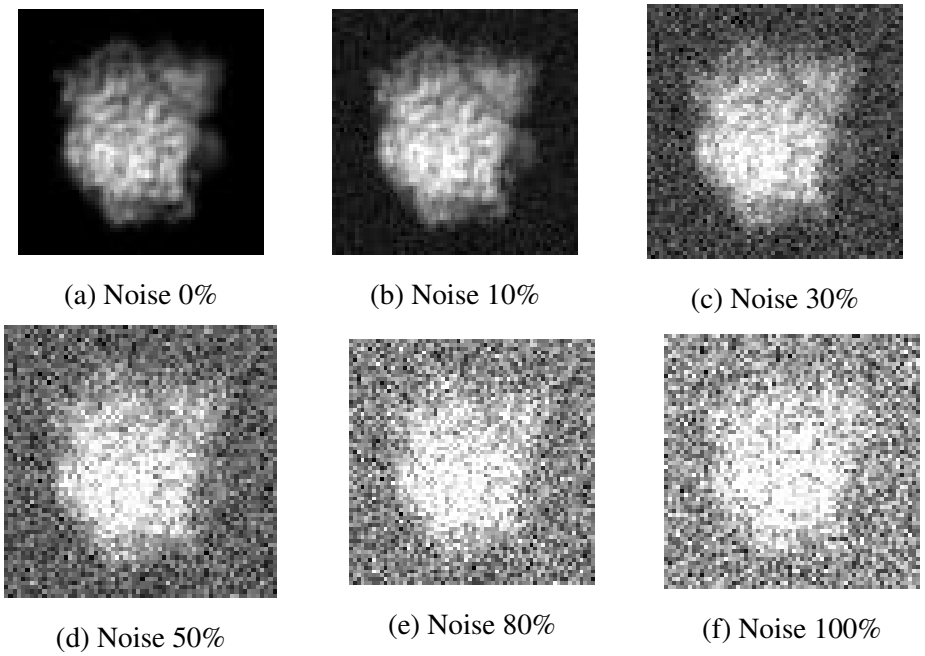
Figure 4.6: 2D Projections

The angles projections are taken from the i.i.d Gaussian distribution. So, a quternion is generated from Gaussian random generator then this quternion is converted into its angle representation. All these projections are center aligned, i.e., they are having zero

shift. But for experimenting ShARP algorithm, shifts have been added, and for ARP algorithm they are used as it is. After that, a different level of noise is being added to these projections for testing the stability algorithms.

Noise

Projections taken from the EM are highly noisy. So, our proposed algorithm must be susceptible to noise. So to make that noise has been added explicitly to these non-noisy projections. The noise which is being added is the Gaussian noise with sigma as 10%, 30%, 50%, 80% & 100% of the mean intensity of all the projections. So, a total of 6 datasets (including zero noise) have been generated per samples, and the tests have been performed on all these datasets. Figure (4.7) shows the sample projections all the 6 noise level for EMD-8647 and figure (4.8) is for EMD-4138.



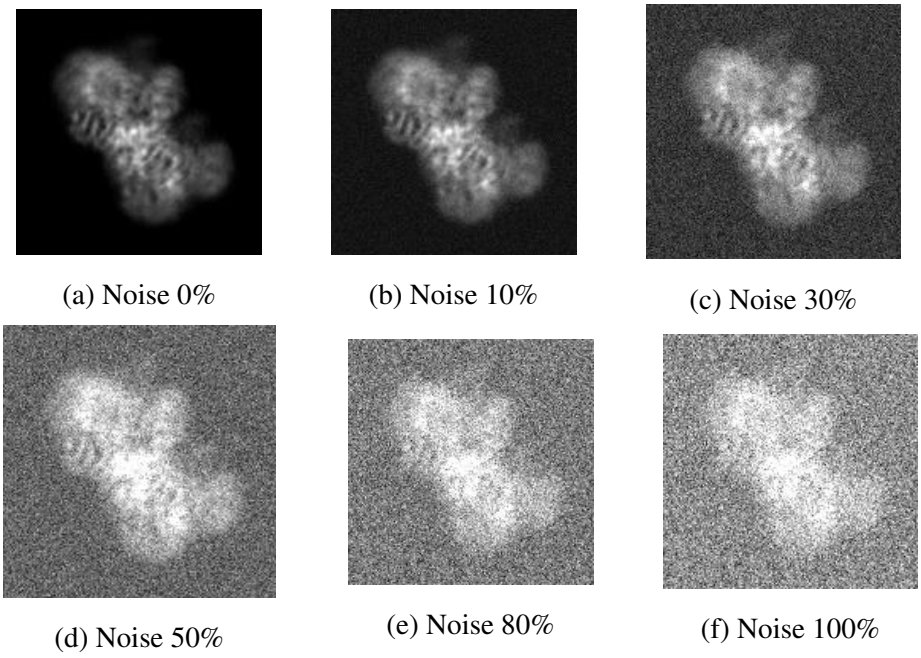
(a) Noise 0% (b) Noise 10% (c) Noise 30%

(d) Noise 50% (e) Noise 80% (f) Noise 100%

Figure 4.7: EMD-8647 Projections with no shift

Shift

Another set 6 datasets each sample in generated with the shift. First, the shift error is added to the projections, and then noise has been added. For EMD-8647 dataset, shift error added is between +/- 5 pixels in both x & y direction taking the center of the image as the origin. For EMD-4138 dataset shift error is between +/- 10 pixels in both x and y-direction.



(a) Noise 0% (b) Noise 10% (c) Noise 30%

(d) Noise 50% (e) Noise 80% (f) Noise 100%

Figure 4.8: EMD-4138 Projections with no shift

4.4.3 Algorithm Pipeline

This section talks about the complete reconstruction algorithm pipeline, i.e., combining all the modules discussed in this chapter until now. It is a serial pipeline where the output of any module will act as the input to its next module. If any of the modules are removed or replaced then final output will be affected drastically, for example replacing *Angles Initialization* with random initialization.

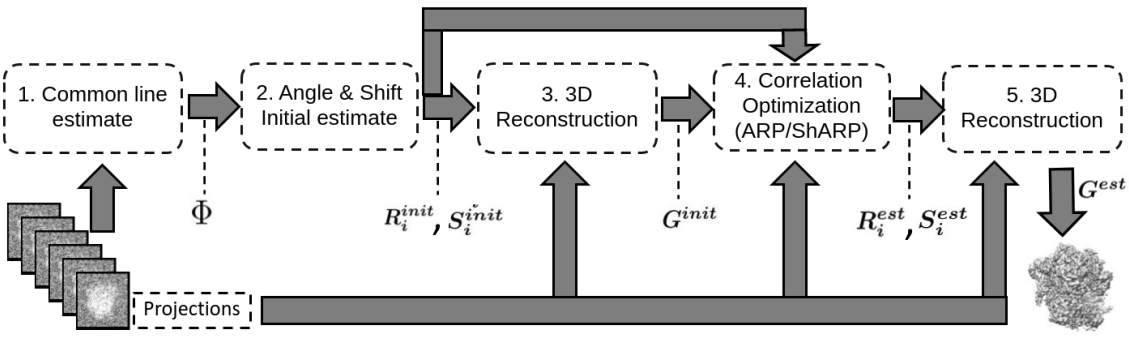


Figure 4.9: 3D-Reconstruction Pipeline

Pipeline (figure 4.9) has 5 stages, first is *Common line estimation*, second is *angle and shift initial estimation*, third is *initial 3D reconstruction*, fourth is the *Correlation Optimization* and last is the *final 3D reconstruction*. Almost all the stages take the projections as their input. The first stage takes the projections as the input to give the Φ matrix.

This Φ matrix is given input to the second stage for the initial angles and shifts estimation for the projections. These initial estimates are used for initial 3D reconstruction in the third stage. The fourth stages takes projections $\boldsymbol{R_i^{init}}$, $\boldsymbol{S_i^{init}}$ & $\boldsymbol{G^{init}}$ as the input for the optimization and the output are $\boldsymbol{R_i^{est}}$ & $\boldsymbol{S_i^{est}}$. The fifth stage will use the output of the fourth stage for doing the final reconstruction. One thing to note here is that its a forward pipeline, i.e. no loops.

4.4.4 Angle Recovery Problem: ARP

Angle recovery problem only considers the unknown angles into account and assumes that there is no shift. So the complete algorithm flow pipeline is mentioned in figure (4.10) First, the common line is estimated using the given N number of projections, which further gives $\Phi_{N\times N}$. This Φ is used for initial angles estimation using A. Singer methods (refer section 4.3.3), which returns the rotation matrix in Euler XYZ convention. Using this $\boldsymbol{R_i^{init}}$ matrices initial 3D object $\boldsymbol{G^{init}}$ is reconstructed. Then $\boldsymbol{R_i^{init}}$ & $\boldsymbol{G^{init}}$ are used for optimizing eq. (4.11), which in returns gives the final 3D reconstructed object

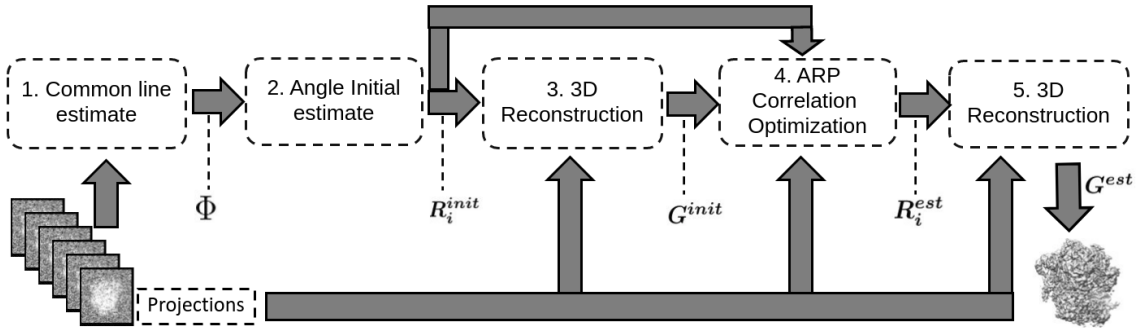


Figure 4.10: 3D-Reconstruction Pipeline - ARP

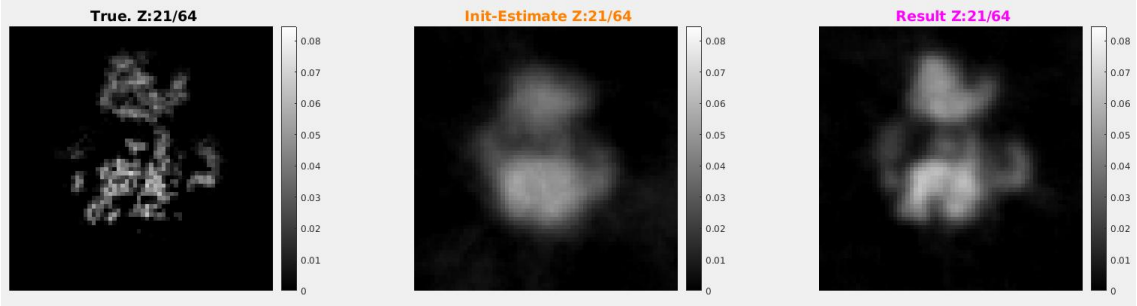
This algorithm pipeline is tested on samples, i.e., EM-8647 & EM-4138. The results for these datasets are mentioned below. These results try to compare original, initial reconstruction using initial angle estimate (see section 4.3.3) and our 3D reconstructed object.

EM-8647

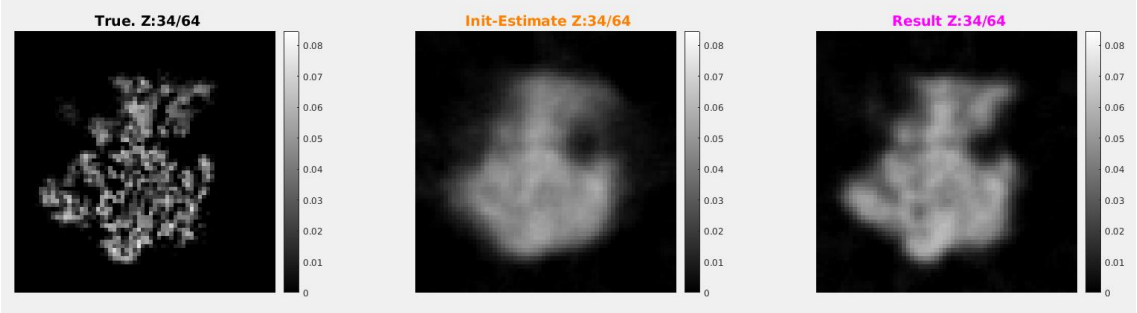
EM-8647 is downsampled to 64 × 64 × 64 before taking the projections. So, after taking the projections, the dimension of each projection is 64 × 64. Using the noiseless projections 5 more datasets are generated at different noise levels i.e. 10%, 30%, 50% 80% & 100%. Then the whole algorithm pipeline is tested on these 6 datasets. For

removing the noise instead of clustering, BM3D for Gaussian noise removal is used (see section 4.3.1).

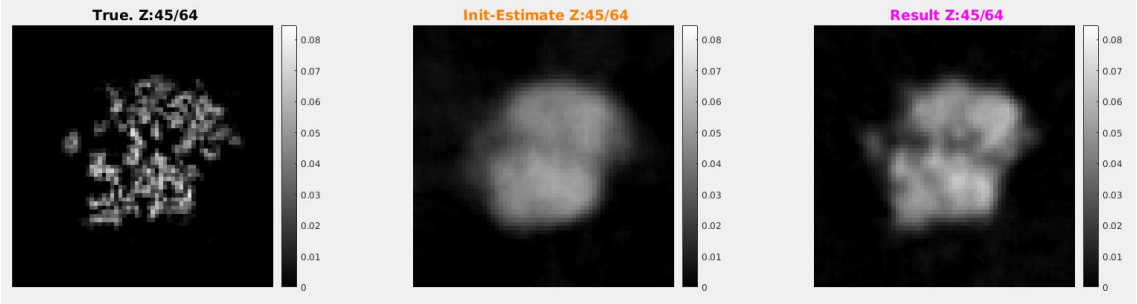
Figure (4.11) shows the 3D reconstruction using the 500 projections with 100% noisy dataset, where each projections size 64×64. Each row represents a slice of a 3D object of dimension $64 \times 64 \times 64$. Four rows are representing the four slices- 21, 34, 45 & 48 in order. Each row has three columns, where the first column shows the true object slice, the second column shows the slice of G^{init} and the third column shows the slice of G^{est}, i.e. our correlation result. Initial reconstruction & final reconstruction are some what blurry because of the noise removal method, BM3D. As BM3D is a patch base algorithm and our projections dimensions 64×64 are pretty less, this leads to blurry projections.



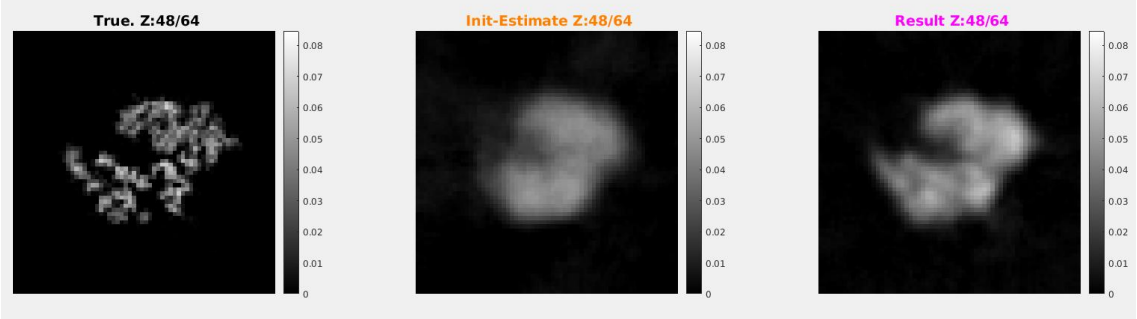
(a) Slice-21



(b) Slice-34



(c) Slice-45



(d) Slice-48

Figure 4.11: EMD-8647 reconstruction result for noise level 100% with 500 projections of dimension 64×64 (see video[6])

One thing to note here is that the initial and final rotation estimate will not be the same as true rotation because all the estimated rotation will have some come global rota-

[6]https://github.com/SeeTheC/MTP_Results/blob/master/EMD-8647_shift5_noise100/video.avi

tion which cannot be solved without knowing the true rotation. So, as these are the simulated experiments, the true rotations are known. Using these true rotations, the global rotation is calculated. Using that global rotation matrix both G^{init} & G^{est} are aligned with the true G^{true} object so that comparison can be done easily.

Convergence Plot

Plot (4.12) represents the convergence of the *Correlation Optimization* eq. (4.11) at every iteration. Y-axis represents the correlation *error* (refer eq. (4.12)) and the x-axis represent the iteration.

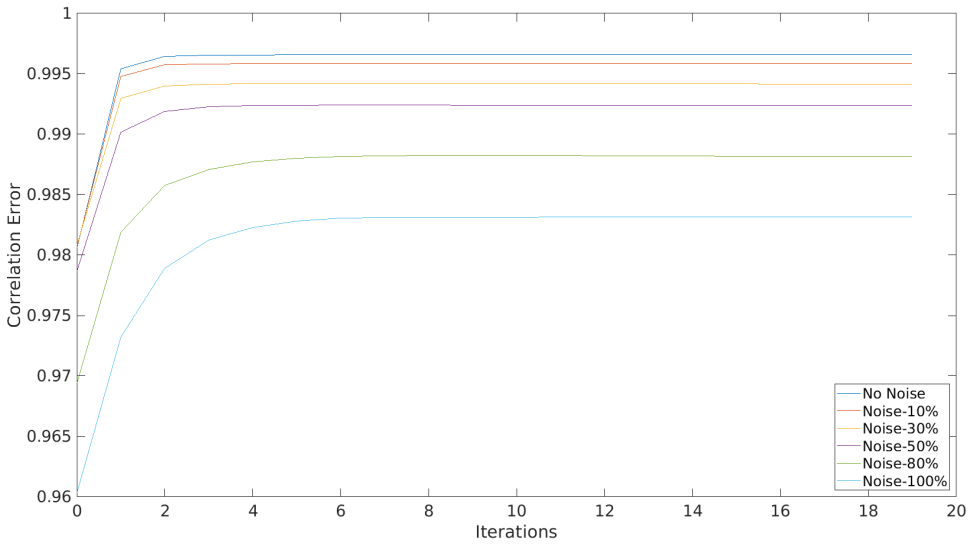


Figure 4.12: EMD-8647: Convergence plot for reconstruction using 500 projections at 100% noise level.

Table (4.1) shows the average 2D normalized correlation between the initial estimated projections and final estimated projections with the true projections. Correlation is represented by eq. (4.12).

Noise level	$NCor(P^{true}, P^{init})$	$NCor(P^{true}, P^{est})$	Iteration
0%	0.980593	0.996572	18
10%	0.980732	0.995853	20
30%	0.980877	0.994158	20
50%	0.978696	0.992373	20
80%	0.969446	0.988164	20
100%	0.960356	0.983171	20

Table 4.1: EMD-8647: 2D normalized correlation of estimated projections using 500 projections with true projections

Results

Estimated time for convergence is approximately 24 hours for the ϵ of 0.0001, ϵ represent change of error from previous iteration to current iteration. Table (4.2) represents the 3D normalized correlation (**NCor**) between initially reconstructed object $\boldsymbol{G^{int}}$ with the true object and the final reconstructed object $\boldsymbol{G^{est}}$ with true object at different noise levels.

Noise level	$NCor(G^{true}, G^{init})$	$NCor(G^{true}, G^{est})$
0%	0.870289	0.951389
10%	0.839869	0.920174
30%	0.811552	0.875694
50%	0.781301	0.841529
80%	0.741130	0.807155
100%	0.716864	0.788950

Table 4.2: EMD-8647: 3D normalized correlation of the reconstructed object using 500 projections with true object

Table (4.3) shows the final orientation error between the estimated and true orientation. In table 4.3, $r_1, r_2 \& r_3$ represents the first,second and third column vector of the rotation matrix respectively. Columns of the table shows the average initial and estimated orientation error for 500 projection. For 100% noise level initial average orientation error is approx. $17°, 18.4° \& 22.1°$ and our estimated average orientation error is $5.4°, 9.6° \& 0.01°$.

Noise	$acos(r_1^t, r_1^{init})$	$acos(r_1^t, r_1^{est})$	$acos(r_2^t, r_3^{init})$	$acos(r_2^t, r_3^{est})$
0%	5.423583	0.381765	5.208526	0.428957
10%	5.515363	0.422370	5.279616	0.464116
30%	6.218024	0.488527	5.995851	0.549958
50%	7.662524	0.683159	7.573490	0.790786
80%	12.215069	2.483265	12.662069	3.153588
100%	17.032287	5.438799	18.017217	9.617589

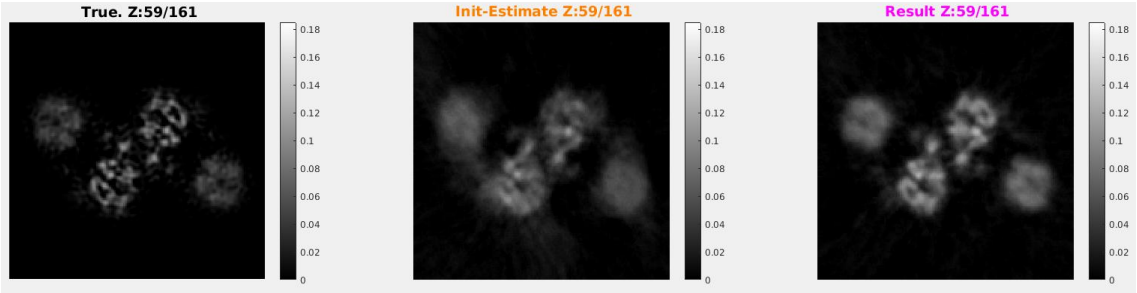
Noise	$acos(r_3^t, r_3^{init})$	$acos(r_3^t, r_3^{est})$
0%	5.981440	0.000858
10%	6.050803	0.000925
30%	6.808487	0.001096
50%	8.536821	0.001572
80%	14.948250	0.006282
100%	22.298040	0.019159

Table 4.3: EMD-8647:Final errors in degrees between estimated and correct orientation

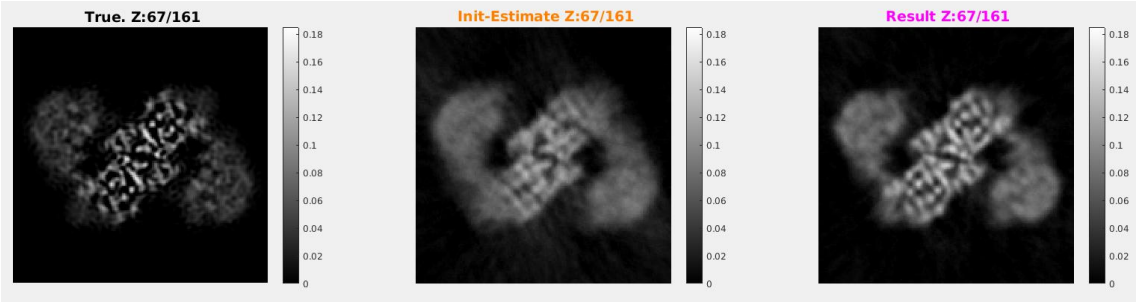
EM-4138

EM-4138 original map dimension is $300 \times 300 \times 300$ with lots of empty background space. So for experiments, it is cropped to $161 \times 161 \times 161$ before taking the projections. So, after taking the projections, the dimension of each projection is 161×161. Using the noiseless projections, 5 more datasets are generated at different noise levels i.e. 10%, 30%, 50% 80% & 100%. Then the whole algorithm pipeline is tested on these 6 datasets. For removing the noise instead of clustering, BM3D for Gaussian noise removal is used (see section 4.3.1).

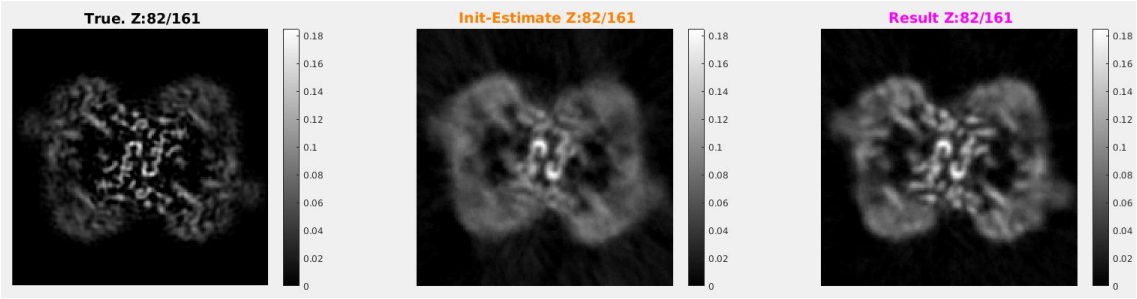
Figure (4.13) shows the 3D reconstruction using the 500 projections with 100% noisy dataset. Each row represents a slice of a 3D object of dimension $161 \times 161 \times 161$. Four rows are representing the four slices- 59, 67, 82 & 88 in order. Each row has three columns, where the first column shows the true object slice, the second column shows the slice of G^{init} and the third column shows the slice of G^{est}, i.e., our correlation result.



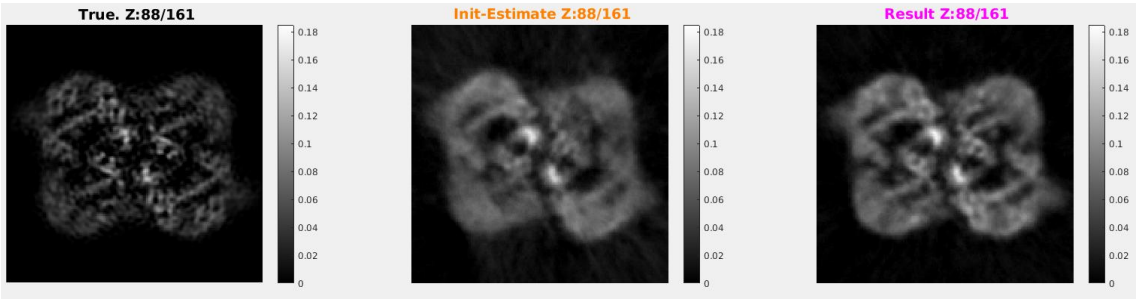
(a) Slice-59



(b) Slice-67



(c) Slice-82



(d) Slice-88

Figure 4.13: EMD-4138 reconstruction result for noise level 100% with 500 projections of dimension 161×161 (see video[7])

[7] https://github.com/SeeTheC/MTP_Results/blob/master/EMD-4138_noshift_noise100/video.avi

Convergence Plot

Plot (4.14) represents the convergence of the *Correlation Optimization* eq. 4.11 at every iteration. Y-axis represents the correlation *error* (refer eq. [4.12]) and the x-axis represents the iteration.

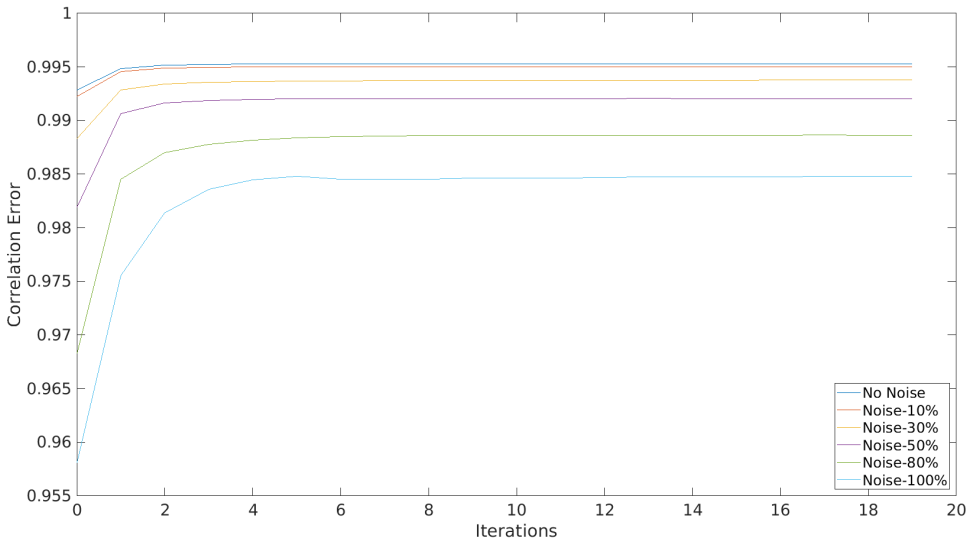


Figure 4.14: EMD-4138: Convergence plot for reconstruction using 500 projections at 100% noise level.

Table (4.4) shows the average 2D normalized correlation between the initial estimated projections and final estimated projections with the true projections. Correlation is represented by eq. (4.12).

Noise level	$NCor(P^{true}, P^{init})$	$NCor(P^{true}, P^{est})$	Iteration
0%	0.992866	0.995287	13
10%	0.992235	0.995002	13
30%	0.988352	0.993723	11
50%	0.981935	0.992033	19
80%	0.968330	0.988639	12
100%	0.958093	0.984821	14

Table 4.4: EMD-4138: 2D normalized correlation of estimated projections using 500 projections with true projections

Results

Estimated time for convergence is approximately 48 hours for the ϵ of 0.00001, ϵ represent change of error from previous iteration to current iteration. Table 4.5 represents the 3D normalized correlation (**NCor**) between initially reconstructed object $\boldsymbol{G^{int}}$ with the true object and the final reconstructed object $\boldsymbol{G^{est}}$ with true object at different noise levels.

Noise level	$NCor(G^{true}, G^{init})$	$NCor(G^{true}, G^{est})$
0%	0.948375	0.964472
10%	0.943699	0.959397
30%	0.920251	0.946016
50%	0.890837	0.932847
80%	0.840490	0.912345
100%	0.813366	0.895854

Table 4.5: EMD-4138: 3D normalized correlation of the reconstructed object using 500 projections with true object

Table (4.6) shows the final orientation error between the estimated and true orientation. In table 4.6, $r_1, r_2 \& r_3$ represents the first,second and third column vector of the rotation matrix respectively. Columns of the table shows the average initial and estimated orientation error for 500 projection. For 100% noise level initial average orientation error is approx. $8.89°, 7.6° \& 10.6°$ and our estimated average orientation error is $3.5°, 4.5° \& 0.091°$.

Noise	$acos(r_1^t, r_1^{init})$	$acos(r_1^t, r_1^{est})$	$acos(r_2^t, r_3^{init})$	$acos(r_2^t, r_3^{est})$
0%	1.457517	0.478219	1.434820	0.616686
10%	1.557083	0.519157	1.510970	0.683813
30%	2.465120	0.803880	2.435245	1.155705
50%	3.730979	0.912525	3.352269	1.113716
80%	6.805973	2.031066	6.021201	2.831163
100%	8.89796	3.562630	7.657457	4.584978

Noise	$acos(r_3^t, r_3^{init})$	$acos(r_3^t, r_3^{est})$
0%	1.988203	0.001233
10%	2.008021	0.001362
30%	3.000396	0.002302
50%	4.054885	0.002219
80%	8.076550	0.005640
100%	10.675927	0.009115

Table 4.6: EMD-4138:Final errors in degrees between estimated and correct orientation

4.4.5 Shift-Angle Recovery Problem: ShARP

ShARP do not assume anything on shifts, i.e., the shift can be there or cannot. Pipeline (4.15) is little modified version of pipeline shown in figure (4.9). As till now, in our experiments stage 1& 2 problems are not solved for shifted projections. So, S_i^{init} can't be predicted right now and this the part of the future work. So for testing our ShARP algorithm (2) the architecture 4.9 has been modified to new architecture 4.15. In this modified architecture, common line and initial angles estimate performed on projections with *no shift*, i.e., stage 1, 2 & 3. For the later stages *shifted* projections are used, i.e. stage 4 & 5. Once shift estimation problems for stage 1 & 2 are solved then architecture 4.9 can be used directly.

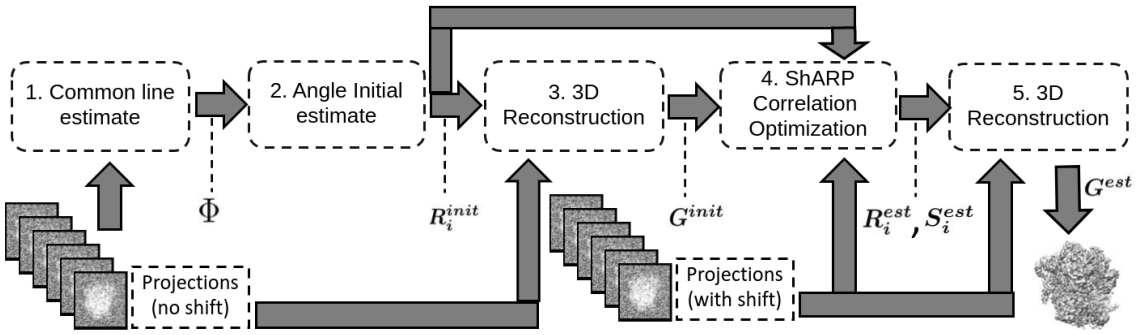
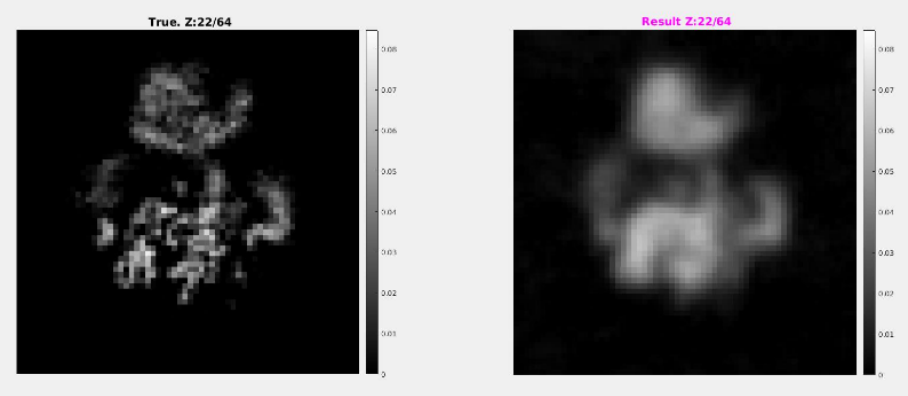


Figure 4.15: 3D-Reconstruction Pipeline

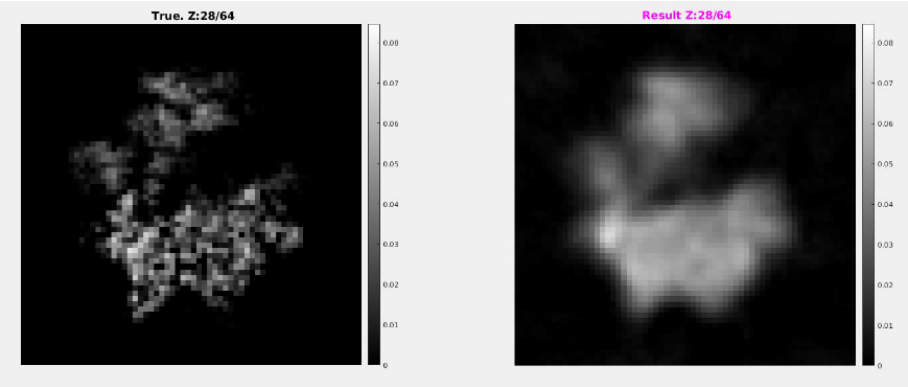
EM-8647

EM-8647 is downsampled to $64 \times 64 \times 64$ before taking the projections. So, after taking the projections, the dimension of each projection is 64×64. Then random shift between -5 to 5 pixels in both x and y directions are added onto these projections. After that, using the noiseless shifted projections, five more datasets are generated at different noise levels, i.e., 10%, 30%, 50% 80% & 100%. Then the whole algorithm pipeline is tested on these 6 datasets. For removing the noise instead of clustering, BM3D for Gaussian noise removal is used (see section 4.3.1). Rotation search space hyperparameter k is set to +/-10 degree.

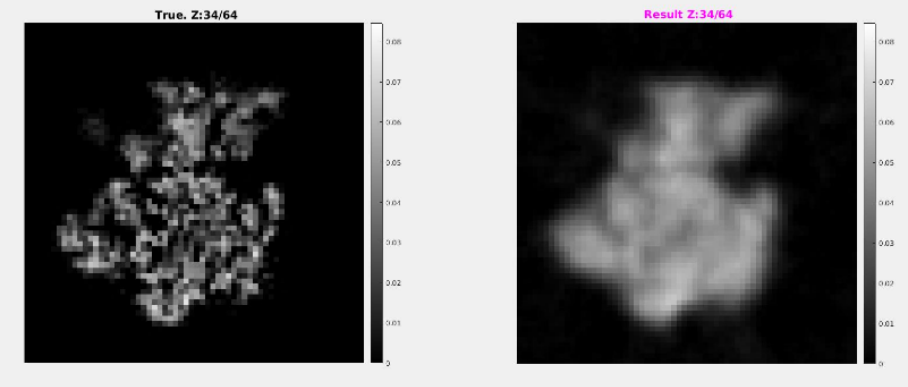
Figure (4.16) shows the 3D reconstruction using the 500 shifted projections with 100% noisy dataset. Each row represents a slice of a 3D object of dimension $64 \times 64 \times 64$. Four rows are representing the four slices- 22, 28, 34 & 49 in order. Each row has two columns, where the first column shows the true object slice, and the second column shows the slice of G^{est}, i.e., our correlation result. Initial reconstruction & final reconstruction are some what blurry because of the noise removal method, BM3D. As BM3D is a patch base algorithm and our projections dimensions 64×64 are pretty less, this leads to blurry projections.



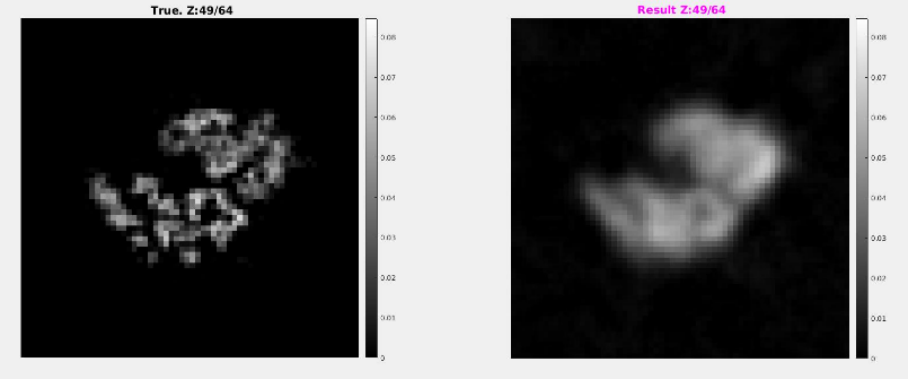
(a) Slice-22



(b) Slice-28



(c) Slice-34



(d) Slice-49

Figure 4.16: EMD-8647 reconstruction result for noise level 100% with 500 projections of dimension 64×64 having random shift between -5 to 5 pixels in both x and y direction (see video[8])

Convergence Plot

Plot (4.17) represents the convergence of the *Correlation Optimization* eq. 4.13 at every iteration. Y-axis represents the correlation *error* (refer eq. [4.16]) and the x-axis represents the iteration.

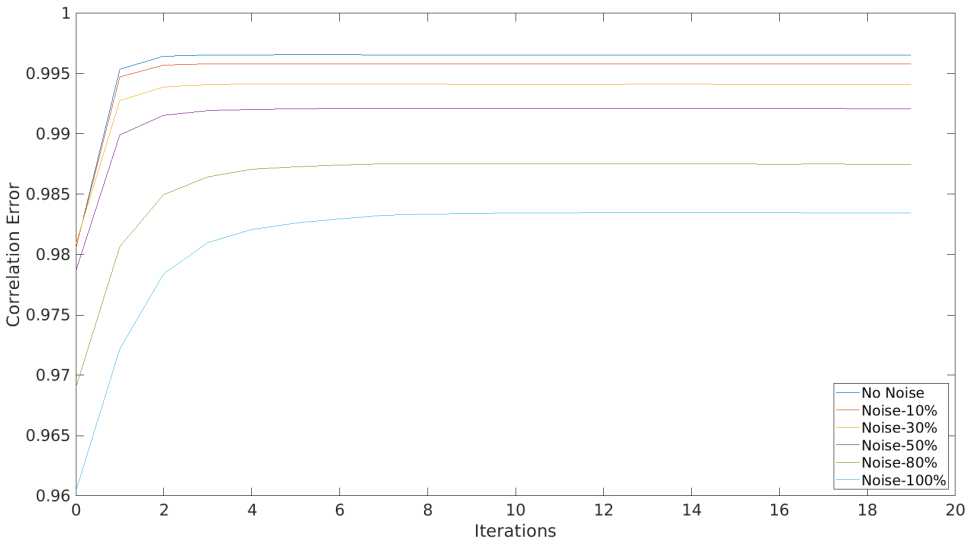


Figure 4.17: EMD-8647: Convergence plot for reconstruction using 500 projections randomly shifted from -5 to 5 pixels in both x and y direction at 100% noise level.

Table (4.7) shows the average 2D normalized correlation between the initial estimated projections and final estimated projections with the true projections. Correlation is represented by eq. [4.12].

Noise level	$NCor(P^{true}, P^{est})$	Iteration
0%	0.996562	20
10%	0.995828	10
30%	0.994116	20
50%	0.992103	20
80%	0.987488	20
100%	0.983475	20

Table 4.7: EMD-8647: 2D normalized correlation of estimated projections using 500 projections with true projections

[8]https://github.com/SeeTheC/MTP_Results/blob/master/EMD-8647_shift5_noise100/video.avi

Results

Estimated time for convergence is approximately 24 hours using one CPU and GPU for the ϵ of 0.00001, ϵ represent change of error from previous iteration to current iteration. Table 4.8 represents the 3D normalized correlation (**NCor**) between final reconstructed object $\boldsymbol{G^{int}}$ with the true object and it's *relative RMS* at different noise levels.

Noise level	$NCor(G^{true}, G^{est})$
0%	0.935679
10%	0.919888
30%	0.875297
50%	0.840329
80%	0.805708
100%	0.789092

Table 4.8: EMD-8647: 3D normalized correlation of the reconstructed object using 500 projections with true object

Table (4.9) shows the final orientation error between the estimated and true orientation. In table (4.9), $r_1, r_2 \& r_3$ represents the first,second and third column vector of the rotation matrix respectively. Columns of the table shows the average estimated orientation error for 500 projection. For 100% noise level our estimated average orientation error is $6.2°, 10.4° \& 0.02°$.

Noise	$acos(r_1^t, r_1^{est})$	$acos(r_2^t, r_2^{est})$	$acos(r_3^t, r_3^{est})$
0%	0.380730	0.425491	0.000851
10%	0.395875	0.447099	0.000891
30%	0.495726	0.584583	0.001165
50%	0.798299	0.948386	0.001889
80%	2.872667	5.175899	0.010311
100%	6.287650	10.474817	0.02086

Table 4.9: EMD-8647: Final orientation errors in degrees between estimated and correct rotation matrix

Table (4.10) shows the final shift error, i.e., by how much predicted shifts are far from the true shifts. The table shows the means shift error for all the 500 projections and also the standard deviation. As one can see that for 0%,10% & 30% noise level, the mean

shifts errors are zero, meaning the predicted shifts matches exactly with the true shifts. Here, shifts are found out using the cross-power spectrum method.

Noise	Mean		σ	
	X-axis	Y-Axis	X-axis	Y-Axis
0%	0	0	0	0
10%	0	0	0	0
30%	0	0	0	0
50%	0.001992	0.001992	0.044632	0.044632
80%	0.029880	0.013944	0.230212	0.204258
100%	0.033865	0.049801	0.276939	0.315137

Table 4.10: EMD-8647: Mean and Standard deviation of shift in pixels between estimated and true projection shifts

4.4.6 Conclusion

Experiments performed for testing ARP (see section 4.4.4) and ShARP (see section 4.4.5) algorithm and the complete pipeline shows the promising results. Considering ARP, the final 3D normalized correlation is higher than the initial one, which is computed using the Eigen value decomposition method [24]. It means running our algorithm (see [1]) after theEigen value decomposition method [24] method will give the more precise estimation.

As shown, ShARP experiments see section (4.4.5)), finding shifts using cross-power spectrum works best instead of doing the doing brute force in x and y-axis. Also, the brute force method will increase one more hyperparameter specifying the 2D search shift space, similar to the rotation hyperparameter.

Estimated execution time ARP and ShARP algorithm with one GPU is approximately 24 hrs for 500 projections of dimension $64x64$ and approximately 48 hrs for 500 projections of dimension $161x161$. Inner loop of the ARP algorithm (1) and ShARP algorithm (2) can be parallelized using multiple GPUs. As this will reduce the execution time drastically, i.e., even with two GPU cards time will reduce to half. Currently, all our experiments are performed on a single Nvidia Quadro M4000 GPU device.

Chapter 5

Future work

5.1 Particle Picking

Our experiments show that using the classifiers SVM, Random Forest, and Faster R-CNN, the translation error is between 20 to 40 pixels. Translation error denotes displacement from its true center. This translation error will create a problem at the time of 3D reconstruction. So, for testing 3D reconstruction accuracy, third-party software like Imagic, Eman, Relion, Cryo-Sparc, etc. or our proposed algorithm (section 4.3) can be used. That will help in deciding how precisely the model can mark out the particles on micrograph.

Micrograph generated for a biological sample from cryo-em has low SNR value. So, training classifier from very noisy data will give less accuracy. In our experiments, denoising of the micrographs was not handled. So one way to handle noisy data is to first denoise the train set and then train the model.

For training classifiers, SVM and Random Forest directly on raw image pixels are not advisable because 100x100 image will have 10K pixels, i.e., features and training on 10K feature will take too much time. Also, a large amount of training data is required for correct classification. So, in our experiments, PCA was used for reducing the dimension of the image, then the reduced dimensions were treated as features for training classifier. For reducing dimension, *Auto-Encoder* (deep learning based model) [15] can also be used, but Auto-Encoder also requires separate training. Also, Random Forest classifier is a CPU based algorithm because of that time of detection is very large, if Random Forest is implemented on GPU, then its detection time can be reduced drastically.

In Faster R-CNN experiment, about 15K particles were used for training. Next thing that can be done is to design new CNN architecture so that the number of marks required for training is less. Because if a model is trained with less markings and it gives good accuracy then that model will be considered best compare to other one which require a large amount of data. In Cryo-EM, getting a large amount of data means manually marking those many particles, which require a large amount of time and effort whereas our final objective is to reduce this time and efforts.

5.2 3D Reconstruction

Execution time of ARP algorithm (1) and ShARP algorithm (1) with 500 projections and rotation search space hyperparameter as $+/-10^{\circ}$ takes days for reconstruction using single GPU. So, if the algorithm is made parallelized for multiple GPU cards, then there will be a huge reduction in reconstruction time. These algorithms converge fast if the initial estimate is good, but if the initial estimate is bad, then this will also increase the convergence time. So, stage 2 for the algorithm pipeline (figure 4.9) should be good.

The ShARP algorithm pipeline (figure 4.15) is modified a little bit, as problem for stage 1 & 2 are not solved for the initial shift estimation. So, once the stage 1 & 2 shift estimation problem is solved, then the ideal algorithm pipeline (figure 4.9) can be used instead of ShARP algorithm pipeline. After then our code can be tested on the real projections taken from EM rather than simulated ones.

All the experiments are performed on simulated projections, but not on the real projections. Until algorithms (ARP & SHARP) are not tested on real projections, it cannot be used practical purpose. Also, real projections have some extra problems than need to be taken care of first before feeding it to our algorithm pipeline (figure 4.9). For example, our simulated non-noisy projections had background pixel value as zero, but this is not the case with the real projections. Real projections have ice (with noise) as there background, so this icy background has to be removed first. Next problem is less contrast, so CTF correction is needed before any use. Another major problem is the Poisson noise. Currently, in our simulated experiments, Gaussian noise is added, and for removing it, BM3D for Gaussian noise is used. But for dealing noise for the real projections, either BM3D for Poisson noise should be used or clustering. After doing all the major pre-processing of real projections, then only they have to be fed in our algorithm pipeline.

References

[1] Basu, S., and Bresler, Y., 2000, "Feasibility of tomography with unknown view angles," *IEEE Transactions on Image Processing* **9**, 1107–1122.

[2] Beck, A., and Teboulle, M., 2009, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM journal on imaging sciences* **2**, 183–202.

[3] Bruggemann, J., Lander, G. C., and Su, A. I., 2017, "Exploring applications of crowdsourcing to cryo-em," *bioRxiv*, 220145.

[4] Chen, Y., Hrabe, T., Pfeffer, S., Pauly, O., Mateus, D., Navab, N., and Förster, F., 2012, "Detection and identification of macromolecular complexes in cryo-electron tomograms using support vector machines," in *Biomedical Imaging (ISBI), 2012 9th IEEE International Symposium on* (IEEE). pp. 1373–1376.

[5] Dabov, K., Foi, A., Katkovnik, V., and Egiazarian, K., 2009, "Bm3d image denoising with shape-adaptive principal component analysis," in *SPARS'09-Signal Processing with Adaptive Sparse Structured Representations*

[6] DIEBOLDER, C., Koster, A. J., Koning, R. I., *et al.*, 2012, "Pushing the resolution limits in cryo electron tomography of biological structures," *Journal of microscopy* **248**, 1–5.

[7] Fessler, J. A., and Sutton, B. P., 2003, "Nonuniform fast fourier transforms using min-max interpolation," *IEEE transactions on signal processing* **51**, 560–574.

[8] Girshick, R., 2015, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448.

[9] Girshick, R., Donahue, J., Darrell, T., and Malik, J., 2014, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587.

[10] Lebrun, M., 2012, "An analysis and implementation of the bm3d image denoising method," *Image Processing On Line* **2**, 175–213.

[11] Levis, A., Schechner, Y. Y., and Talmon, R., 2018, "Statistical tomography of microscopic life," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6411–6420.

[12] Malhotra, E., Pandotra, H., Rajwade, A., and Gurumoorthy, K. S., 2017, "Signal recovery in perturbed fourier compressed sensing," *arXiv preprint arXiv:1708.01398*

[13] Malhotra, E., and Rajwade, A., 2016, "Tomographic reconstruction from projections with unknown view angles exploiting moment-based relationships," in *Image Processing (ICIP), 2016 IEEE International Conference on* (IEEE). pp. 1759–1763.

[14] Mallick, S. P., Agarwal, S., Kriegman, D. J., Belongie, S. J., Carragher, B., and Potter, C. S., 2006, "Structure and view estimation for tomographic reconstruction: A bayesian approach," in *null* (IEEE). pp. 2253–2260.

[15] Meng, Q., Catchpoole, D., Skillicom, D., and Kennedy, P. J., 2017, "Relational autoencoder for feature extraction," in *Neural Networks (IJCNN), 2017 International Joint Conference on* (IEEE). pp. 364–371.

[16] Natterer, F., 2001, *The mathematics of computerized tomography* (Siam).

[17] Poinapen, D., Konopka, J. K., Umoh, J. U., Norley, C. J., McNeil, J. N., and Holdsworth, D. W., 2017, "Micro-ct imaging of live insects using carbon dioxide gas-induced hypoxia as anesthetic with minimal impact on certain subsequent life history traits," *BMC Zoology* **2**, 9.

[18] Reddy, B. S., and Chatterji, B. N., 1996, "An fft-based technique for translation, rotation, and scale-invariant image registration," *IEEE transactions on image processing* **5**, 1266–1271.

[19] Ren, S., He, K., Girshick, R., and Sun, J., 2015, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, pp. 91–99.

[20] Scheres, S. H., 2012, "Relion: implementation of a bayesian approach to cryo-em structure determination," *Journal of structural biology* **180**, 519–530.

[21] Sigworth, F. J., 2016, "Principles of cryo-em single-particle image processing," *Microscopy* **65**, 57–67.

[22] Sigworth, F. J., and Singer, A., ????, "Cryo-em structure determination through eigenvectors of sparse matrices ronald. r. coifman yoel shkolnisky,"

[23] Simonyan, K., and Zisserman, A., 2014, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*

[24] Singer, A., and Shkolnisky, Y., 2009, "Three-dimensional structure determination from common lines in cryo-em by eigenvectors and semidefinite programming,"

[25] Van Heel, M., 1987, "Angular reconstitution: a posteriori assignment of projection directions for 3d reconstruction," *Ultramicroscopy* **21**, 111–123.

[26] Wang, F., Gong, H., Liu, G., Li, M., Yan, C., Xia, T., Li, X., and Zeng, J., 2016, "Deeppicker: A deep learning approach for fully automated particle picking in cryo-em," *Journal of structural biology* **195**, 325–336.

[27] Wang, H.-W., and Wang, J.-W., 2017, "How cryo-electron microscopy and x-ray crystallography complement each other," *Protein Science* **26**, 32–39.

[28] Wang, Z., and Lu, Y., 2018, "Improving initial model construction in single particle cryo-em by filtering out low quality projection images," in *International Conference on Intelligent Computing* (Springer). pp. 589–600.

[29] Xiao, Y., and Yang, G., 2017, "A fast method for particle picking in cryo-electron micrographs based on fast r-cnn," in *AIP Conference Proceedings*, Vol. 1836 (AIP Publishing). p. 020080.

Acknowledgements

I would like to thank Sunil Vichare, Technical Officer and Ninad N. Shaha, Technical Superintendent of Computer Center, IIT Bombay for there constant support with hardware and internet bandwidth. As, these datasets are in a few hundred GBs, so to download datasets large disk and bandwidth are required. CSE-Sysad, IIT Bombay has also helped us with the hardware support. More important, I would like thank VIGIL lab, IIT Bombay for providing voxel servers with GPUs. Without any GPU machine, completing this dissertation was near to impossible

Khursheed Ali
IIT Bombay
31 August 2019