caesar
center of advanced
european studies
and research

UNIVERSITÄT
KOBLENZ · LANDAU

Nanotemplates for the combined structural and functional analysis of membrane-associated proteins

vorgelegt von

Nikolai Krupp

geboren am 08.12.1986 in Bonn

Angenommene Dissertation zur Erlangung des akademischen
Grades eines Doktors der Naturwissenschaften (Dr. rer. nat.)

Fachbereich 3: Mathematik/Naturwissenschaften
Universität Koblenz · Landau

Gutachterinnen und Gutachter:
Prof. Dr. Barbara Hahn
Prof. Dr. Elmar Behrmann

Prüfungskommission:
Prof. Dr. Barbara Hahn
Prof. Dr. Elmar Behrmann
Prof. Dr. Eberhard Fischer

Tag der Disputation: 08. Februar 2019

Abstract

Plasma membranes are essential for life because they give cells an identity. Plasma membranes are almost impermeable to fluids and substances. Still, transport between inside and outside needs to be possible. An important transport way is endocytosis. This mechanism relies on membrane-associated proteins that sense and induce curvature to the plasma membrane. However, the physics and structural dynamics behind proteins acting on membranes is not well understood. There is a standard method *in vitro* to investigate membrane-associated proteins sensing spherical geometries: They are incubated on unilamellar vesicles. This procedure allows to analyze these proteins in their bound state. This approach is inappropriate for GRAF1 (*GTPase Regulator Associated with Focal Adhesion Kinase-1*), a key player in endocytosis because it senses tubular geometries instead. However, GRAF1 extrudes lipid tubes from vesicles that can be analyzed. Still, this is a limited method because these tubes suffer from inhomogeneity and they do not enable the observation of intermediate and lower concentration binding states. To overcome this issue they can be incubated on pre-tubular structures called nanotemplates. There have been studies using carbon nanotubes and Galactosylceramide lipid tubes as nanotemplates. These approaches require complex chemical modifications or expensive components and they are not necessarily flexible. In this work we present a simple and easy new approach to prepare nanotemplates using Folch lipid mixture. We show on the basis of BPG, a truncate of GRAF1, that our nanotemplates are suitable for Cryo-EM and that it is possible to use IHRSR (*Iterative Helical Real Space Reconstruction*) to analyze the structure of BPG in its bound state. Moreover, the qualification for Cryo-EM allows to use plunge freezing to interrupt the incubation on our nanotemplates abruptly. This enables the analysis of intermediate binding states to understand the binding process.

Zusammenfassung

Die Inkubation von unilamellaren Lipid-Vesikeln mit einem als *BPG* bezeichneten Abschnitt des Proteins *GRAF1* führte zur Entstehung von röhrenförmigen Strukturen. Eine Untersuchung mittels negativkontrastierter Elektronenmikroskopie zeigte eine körnige Beschichtung auf der Röhrenoberfläche. Durch die höhere Auflösung der Elektronentomographie war es möglich, helikale Oligomere auf den Röhren zu erkennen. Des Weiteren zeigte sich, dass die Röhren im Durchmesser inhomogen waren und Biegungen aufwiesen. Das hatte ebenfalls Unregelmäßigkeiten in der Ordnung der helikalen Struktur zur Folge. Das Herstellungsverfahren von unilamellaren Vesikeln mittels Extruder wurde so modifiziert, dass es anstatt von Vesikeln Nanoröhren entstehen ließ. Im Gegensatz zu den Röhren, die das Protein formte, wiesen diese eine lineare Struktur mit konstantem Durchmesser auf. Trotz der Inkubation mit BPG behielten diese ihre homogene Form und zeigten nun eine geordnete helikale Beschichtung auf ihrer Oberfläche. Allerdings waren sie nicht hoch genug konzentriert für die Kryoelektronenmikroskopie, und eine reine Erhöhung der Lipidkonzentration führte nicht zu einer höhren Anzahl von Röhren, sondern zu Lipidaggregation. Weitere Experimente zeigten, dass die Chelatliganden EDTA und EGTA in der Pufferlösung zur Entstehung unilamellarer Vesikel führten. Da EDTA und EGTA die Metallionen Mg^{2+} und Ca^{2+} absorbieren, wurden weitere Tests mit neuen Pufferlösungen durchgeführt. Zur Absorption bereits vorhandener Metallionen wurden beide Chelatliganden hinzugegeben und eine Variante mit zusätzlichem Magnesiumchlorid und eine weitere mit zusätzlichem Calciumchlorid hergestellt. Es wurde gezeigt, dass Magnesiumchlorid unilamellare Vesikel zur Folge hatte und Calciumchlorid zu einer höhren Konzentration der Nanoröhren führte. Daraus wurde geschlossen, dass Calciumionen die Enstehung von Nanoröhren fördern. Zur weiteren Optimierung der Konzentration wurde eine Varianzanalyse (CCD) der einflussreichsten Variablen im Extrusionsprozess durchgeführt:

- Eine höhere Anzahl von Frier-Tau-Zyklen zur Unterdrückung multilamellarer Vesikel hatte einen positiven Einfluss.
- Es konnte keine Korrelation zwischen der Extrudertemperatur und der Konzen-

tration von Nanoröhren nachgewiesen werden.

- Mehr Extrusionszyklen hatten eine höhere Nanoröhrenkonzentration zur Folge.

Durch das optimierte Protokoll war die Konzentration hoch genug für die Kryoelektronenmikroskopie und eine helikale Rekonstruktion. Dennoch war eine 2D-Klassifizierung nicht erfolgreich, da die Nanoröhren im aufgenommenen Datensatz nicht homogen genug im Durchmesser waren und zudem eine unterschiedliche Anzahl an multilamellaren Rändern aufwiesen. Aus diesem Grund wurde ein Skript entwickelt, das die Röhren nach ihrem Durchmesser sortiert. Mit einer Untermenge des Datensatzes, dessen Röhren einen ähnlichen Durchmesser hatten, wurde eine weitere 2D-Klassifizierung durchgeführt. Daraus resultierten schließlich 2D-Klassen der helikalen Oberflächenstruktur.

Conclusion

The incubation of the *GRAF1* truncate *BPG* on unilamellar vesicles resulted in protein-extruded lipid tubes. Negative stain electron microscopy showed lipid tubes and vesicles with a grainy coat on their surface. The superior resolution of electron tomography enabled to see helical oligomers on the surface of the tubes. However, the experiments revealed curved shapes and inhomogeneities in the diameter of these tubes. Hence, the helical oligomers are irregular. A protocol derived from the preparation of unilamellar vesicles using a lipid extruder yielded *LNTs* (**L**ipid **N**ano**T**ubes) instead of vesicles. They feature a linear shape and a constant diameter. When incubated with BPG they kept their shape and still showed a regular helical coat on their surface. The concentration of LNTs was insufficient for Cryo-EM. Increasing the lipid concentration yielded aggregates instead of higher LNT concentrations. Experiments showed that the presence of the chelating agents EDTA and EGTA in the buffer resulted in unilamellar vesicles instead of LNTs. More experiments were performed because EDTA and EGTA absorb Mg^{2+} and Ca^{2+} ions. Tests with new buffers containing both chelating agents to bind already existent metal ions and additionally magnesium chloride or calcium chloride were performed. While the presence of magnesium chloride resulted in vesicles, calcium chloride yielded higher LNT concentrations. This showed that calcium ions favor the tube formation. In the next step a Central Composite Design was performed to optimize the tube yield by reviewing the most influential variables in the extrusion process:

- More freeze-thaw-cycles to disrupt multilamellar vesicles showed a positive influence.
- The extruder temperature did not show any reliable correlations.
- More extrusion cycles resulted in higher tube concentrations.

Using the optimized protocol the concentration was high enough for Cryo-EM and also for IHRSR. However, the 2D classification failed because of inhomogeneous tube diameters and a variable number of multilamellar tube borders in the dataset. A script was developed to perform an automated tube diameter classification.

Another 2D classification for IHRSR was performed with a subset of tubes with similar diameters and truncated tube borders. Finally, 2D classes of the helical protein coat on LNTs could be obtained.

Contents

Abstract i

Zusammenfassung iii

Conclusion v

1 Introduction 2

1.1 Curvature Energy . . . 3

1.2 Analysis of Membrane Curvature-Sensing Proteins . . . 5

1.3 GRAF1 and BPG . . . 7

1.4 Nanotemplates . . . 8

1.4.1 Carbon Nanotubes . . . 9

1.4.2 Lipid Tubes . . . 10

1.5 3D Electron Microscopy . . . 10

1.5.1 Electron Tomography 12

1.5.2 Sub-Tomogram Averaging 15

1.5.3 Helical Reconstruction 16

1.6 Central Composite Design 19

2 Aim of this work 22

3 Materials and Methods 24

3.1 Chemicals and Consumables 24

3.1.1 Buffer Solutions 25

3.2 Devices and Software 26

3.2.1 Electron Microscopes 26

3.3 Methods 27

3.3.1 Lipid Extruder and Unilamellar Vesicles 27

3.3.2 Lipid Nanotubes 28

3.3.3 Optimization of Nanotemplates 30

3.3.4 BPG Tubulation Experiment 31

3.3.5 Glow Discharge 32

3.3.6 Negative Staining 32

3.3.7 Fiducial Markers for Tomography 32

3.3.8 Graphene Oxide and Poly-L-Lysine 33

3.3.9 Plunge Freezing 33

3.3.10 Tube Counting 34

3.3.11 Manual Tube Measurement 36

3.3.12 Negative Stain Tomography 36

3.3.13 Cryo Data Acquisition 37

3.3.14 Particle Selection 39

3.3.15 Automated Tube Analysis 40

3.3.16 Helical Reconstruction 49

4 Results **50**

4.1 Unilamellar Vesicles 50

4.1.1 ULVs with BPG 51

4.1.2 Helical Parameters 53

4.2 Lipid Nanotubes 54

4.2.1 BPG on LNTs 55

4.2.2 Concentration 56

4.3 Optimization 57
4.3.1 Dependency on Buffer Components 57
4.3.2 Central Composite Design 60
4.3.3 Dependency on Extrusion Temperature 63
4.3.4 Stability of LNTs 65
4.4 Dataset Acquisition 67
4.5 Automated LNT Analysis 68
4.6 Helical Reconstruction 72
4.6.1 Diameter Class Selection 72
4.6.2 Diameter Group Selection 74
4.6.3 Classification by RELION 75

5 Discussion **77**

5.1 Energy Distribution 77
5.2 Nanotemplates 78
5.3 BPG on LNTs 80
5.4 Optimization of LNTs 81
5.5 Helical Reconstruction 88
5.6 Outlook 92

Bibliography **93**

List of Figures **110**

List of Tables **112**

A Central Composite Design **113**

B imod-prepare-stack **116**

C LineAdd.py **120**

Danksagungen **125**

Chapter 1

Introduction

The plasma membrane defines biological cells as it separates inside from outside. While it is almost impermeable to fluids and substances its function is not to prevent all exchanges. Still, there need to be ways to transport amino acids, specific ions, sugars and vitamins through the membrane[1]. It consists of a phospholipid bilayer, robust enough to protect the cell. The membrane is also dynamic enough to bend, fold and flex. Phospholipids, in case of membranes phosphoglycerides, are amphipathic molecules that feature hydrophobic tails and hydrophilic head groups. The non-polar tails of two lipid molecules connect to each other by van der Waals interactions and form the hydrophobic core of the membrane bilayer. The polar head groups are pointing away from the hydrophobic core and stabilize the membrane by ionic and hydrogen bonds to their neighboring head groups. The thickness of the membrane corresponds to the size of two phospholipids. There are three major ways to transport substances through the plasma membrane:

- **Simple diffusion:** Small hydrophobic molecules can pass the membrane.
- **Ion channels:** Transmembrane proteins are embedded in the lipid bilayer. They facilitate active or passive transport of substances across the membrane. The passive transport allows ions to diffuse through an opened channel. For the active transport ions are physically pumped against chemical

gradients. There are also carrier proteins that diffuse through the membrane while they bind specific molecules.

- **Endocytosis** and **Exocytosis:** For this transport way proteins deform the membrane and constrict vesicles that enclose fluids and macromolecules[2][3]. There are two forms of endocytosis: One is mediated by the protein clathrin[4] and the other is mediated independently from clathrin by other proteins[5][6].

In the relaxed state, the membrane adopts the state with the lowest curvature energy. Changing curvature requires mechanical energy that is provided by proteins[7]. This process is not well understood. Membrane proteins are complex macromolecules subdivided into different domains in dynamic arrangements. To understand the energy distribution of these proteins it is needed to analyze different molecular states at a high resolution. The structure in unbound states can be studied using X-ray crystallography. However, this does not allow to analyze bound and dynamic states. One conventional method to study proteins in their bound state is to analyze protein-coated lipid vesicles[8]. This method can only be considered for proteins sensing spherical geometries. In this work a method is shown that allows to obtain atomic resolution models of tubular geometry sensing proteins in a bound state. Moreover, it allows to observe intermediate states of the binding reaction to a membrane. For this method nanotemplates are used as tubular shaped binding targets for the proteins. Snapshots of the binding reaction on nanotemplates can be analyzed using Cryo Electron Microscopy. This allows to obtain molecular models of dynamic protein states and to draw conclusions about the energy distribution.

1.1 Curvature Energy

During endocytosis proteins change the curvature energy of the membrane while acting on them. An approach to calculate total energy of a curved lipid bilayer is the *Area-Difference Elasticity Model*[9]. It is based on the assumption that the cell or vesicle is a spherical volume V with the radius R. Its total curvature energy G can be calculated by equation 1.1:

$$G = \frac{\kappa}{2} \oint (C_1 + C_2)^2 \, \mathrm{d}A. \tag{1.1}$$

The volume is surrounded by its curved surface area A and depends on the two principal curvatures C_1 and C_2 of the surfaces of the bilayer and the local bending rigidity κ. The curvatures can be calculated using the *Riemann Curvature Tensor* Ω that depends on the Cartesian coordinates x, y, z:

$$\Omega = \begin{pmatrix} \frac{\partial^2 z}{\partial x^2} & \frac{\partial^2 z}{\partial x \partial y} \\ \frac{\partial^2 z}{\partial y \partial x} & \frac{\partial^2 z}{\partial y^2} \end{pmatrix} \tag{1.2}$$

To calculate the corresponding radii R_1 and R_2 to the curvatures only its main diagonal elements k_1 and k_2 are needed:

$$R_1 = \frac{1}{k_1},\ R_2 = \frac{1}{k_2},\ C_1 + C_2 = \mathrm{tr}(\Omega) \tag{1.3}$$

The surface area can be calculated with the help of the volume radius and also by the number of lipid molecules N_1, N_2 in the bilayer and the equilibrium density ϕ_0:

$$A = 4\pi R^2 = \frac{N_1 + N_2}{2\phi_0} \tag{1.4}$$

In figure 1.1 N_1 and N_2 and their distance to the center are shown:

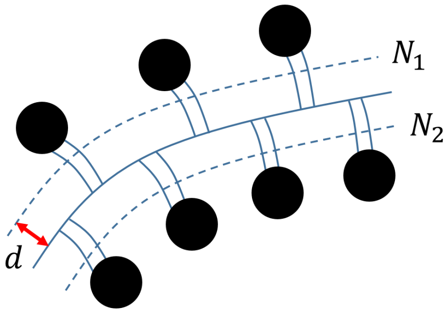


Figure 1.1: Two layers of lipid molecules form a curved membrane with the number of lipid molecules N_1 and N_2 and the distance d from the center.

Finally, there is a more general form of the total curvature energy named F calculated by equation 1.5:

$$F = \frac{\kappa' \pi}{2Ad^2}(\Delta A - \Delta A_0)^2 + \frac{\kappa}{2} \oint (C_1 - C_2 + C_0)^2 \, \mathrm{d}A \tag{1.5}$$

It depends on the spontaneous curvature C_0, the distance d (in figure 1.1) of the molecules from the center of the bilayer and the non-local bending rigidity κ'. The equation also depends on the difference ΔA from an unstreched equilibrium value ΔA_0. The full derivation can be found in [10].

1.2 Analysis of Membrane Curvature-Sensing Proteins

One conventional method to study curvature-sensing proteins is to analyze *ULVs* (**Uni****L**amellar **V**esicles) coated by proteins[8]. To mimic a close to native environment for the proteins *in vitro* the experimental setup has to fulfill some requirements[11][12]:

- **Buffer solution:** The ionic concentrations and the pH value have to match the protein's environment in nature.
- **Lipid composition:** The vesicle's bilayer needs to consist of the right lipid mixture. The length of hydrophobic tails and the charge of the head groups are important.
- **Vesicle geometry:** The size of the vesicles defines their surface curvature. The proteins can only sense and bind to membranes with a specific curvature.
- **Ligands and energy suppliers:** Many proteins are only active in the presence of ligands such as ions, molecules and other proteins. Some proteins also require an energy supply from nucleotides for the binding process. These need to be present in the right concentration as well.

If the experiment meets all requirements the protein can bind on the vesicles and these can be analyzed. Besides proteins that sense spherical geometries there are also proteins that sense tubular geometries[13]. Some of these proteins can also extrude lipid tubes from vesicles by oligomerization *in vitro*, e.g. some BAR domain based[14], Epsin[15], EHD family[16]. In figure 1.2 a self-extruded tube with helical protein coat is illustrated.

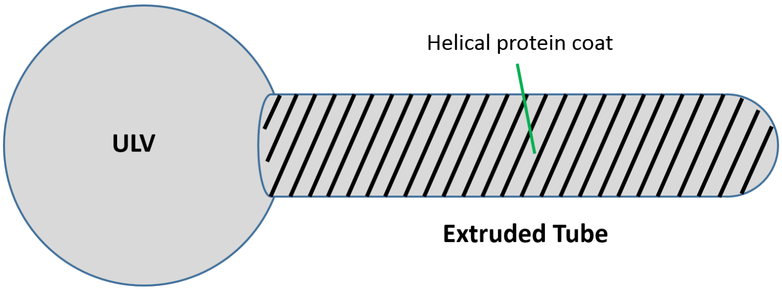
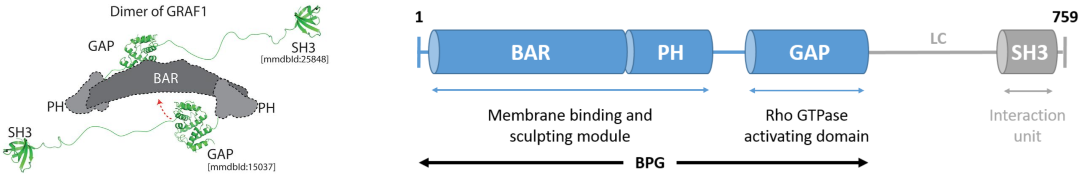


Figure 1.2: From ULV extruded lipid tube. The stripes on the tube illustrate a helical oligomeric coat of protein molecules.

Studying tubular sensing proteins in a state bound on vesicles does not reveal them in their natural geometry and may lead to wrong reconstructions. Thus, a tubular shaped support is a better approach. It is possible to analyze the lipid tubes self-extruded by the protein itself. However, this method has disadvantages: First of all, these tubes are not regular. They are bent, and the diameter is not constant. This makes a molecular reconstruction more difficult because the organization of the molecules in an oligomer is deranged (see chapter 4.1.1). Secondly, self-extruded tubes only show the reaction in its final state. There is no possibility to observe oligomers formed at lower concentrations with less molecules. Moreover, there is no possibility to see intermediate binding states that reveal the (intra-) molecular dynamics of the protein. To overcome these problems a pre-tubular shaped support is needed. This enables to see fewer molecules binding on the support and the binding reaction can be stopped in intermediate states. This is possible by rapidly freezing aqueous samples to vitrified ice layers and analyze them in *Cryo-EM* (Cryo **E**lectron **M**icroscopy)[17][18]. At atomic resolution only Cryo-EM allows to analyze non-crystallized aqueous samples (compared to X-ray crystallography).

1.3 GRAF1 and BPG

To study membrane bending the protein *GRAF1* (**G**TPase **R**egulator **A**ssociated with **F**ocal Adhesion Kinase-**1**) was used. GRAF1 is involved in endocytosis. It remodels membranes into tubulovesicular *CLICs* (**Cl**athrin-**I**ndependent **C**arriers) that mediates lipid-anchored receptor endocytosis[19][20]. The GRAF1 monomer consists of N-BAR, PH, GAP and SH3 domains. *In vivo*, GRAF1 always acts as a dimer as shown in figure 1.3a. In absence of proper ligands the SH3 domains prevent the N-BAR domain from binding on membranes. In presence of an appropriate ligand the SH3 domains are folded away by intramolecular interactions. To cause the maximum binding activity, a truncate of GRAF1 lacking the SH3 domains was used. It only contains the N-BAR, PH and GAP domains and is called *BPG* (**B**AR-**P**H-**G**AP)[21]. In figure 1.3b the truncate BPG lacking SH3 domains is shown. BPG's N-BAR domain is always active and can bind on membranes without the control of ligands or signals[22].



(a) GRAF1 protein dimer[19].

(b) BPG fragment.

Figure 1.3: **(a)** shows a dimeric domain model of GRAF1. It is composed of N-BAR, PH, rhoGAP and SH3 domains. The SH3 domains interact with ligands to unblock the membrane binding module. **(b)** BPG is a fragment of GRAF1 lacking the SH3 domains. BPG is always active

BAR (**B**in **A**mphiphysin **R**vs) domain proteins are an important superfamily of curvature-sensing proteins. Each character of the name represents a protein family including BAR domains. The name also comes from the concave-, bar- or banana-shaped geometry of BAR dimers. BAR proteins regulate the membrane shape of organella and membrane fusion and fission[23]. Via its positive charge BAR dimers bind and induce curvature in negative charged lipid membranes[24][25]. In

addition, *N-BAR* domains feature an N-terminal amphipathic helix that inserts into the membrane and induces curvature[26].
BPG senses tubular-shaped geometries. Cylindrical nanotemplates can be used to bind BPG and study its molecular structure *in vitro* and close to nature.

1.4 Nanotemplates

To analyze proteins sensing tubular geometries it is possible to use cylindrical structures instead of ULVs as protein binding targets. We call them (cylindrical) NANOTEMPLATES. To successfully bind proteins to nanotemplates they must fulfill multiple strict requirements: Nanotemplates need a biomimetic shell consisting of a lipid bilayer mimicking the biological cell membrane that the protein is targeting at. Thus, the nanotemplate's surface must match the cell's lipid composition and charge in the same way as ULVs in chapter 1.2. Secondly, the diameter of the nanotemplates should be close to the diameter of self-extruded lipid tubes to match the membrane curvature the protein binds to. When the deviation in diameter is too much the protein's helical oligomer will not be able to coat the surface of the nanotemplate. Finally, nanotemplates must be suitable for Cryo-EM. They must not decay during plunge freezing and not disturb the vitreous ice layer on EM grids. Figure 1.4 illustrates the requirements on the nanotemplates.

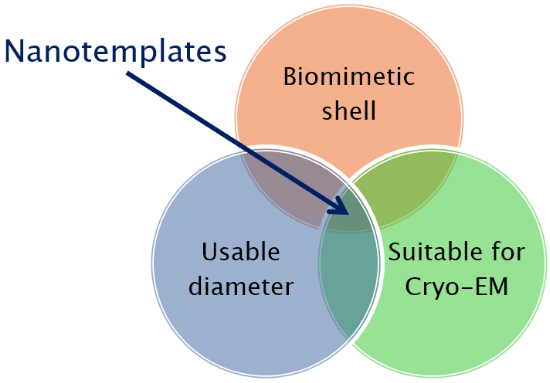


Figure 1.4: Nanotemplates must meet all requirements at once.

If the nanotemplates are mechanically rigid structures they also enable protein analysis by helical reconstructions. These methods require linear tubular structures with a constant diameter. They enable reconstructions of 3D models of protein molecules at atomic resolution. This is described in detail in chapter 1.5.3. There are different ways to build nanotemplates: There have been approaches for cylindrical nanotemplates that can be divided mainly into two different classes: carbon nanotubes and lipid tubes. These two approaches are most reasonable to provide mechanically rigid structures for Cryo-EM that can support protein oligomers.

1.4.1 Carbon Nanotubes

CNTs (**C**arbon **N**ano**t**ubes) only consists of carbon. The simplest CNT is basically a rolled sheet of graphene with a cylindrical shape. When they only consist of a single layer of graphene they are called *SWNTs* (**S**ingle-**W**alled C**NT**s). There are also CNTs consisting of graphene multilayers called *MWNTs* (**M**ulti-**W**alled C**NT**s). Their diameter depends on the number of layers. Each layer adds 3.35 Å to the diameter[27]. Thus, a diameter can in theory be selected by using CNTs with the related count of layers. CNTs feature high mechanical strength because each carbon atom in the lattice is three times covalently bound. MWNTs can be purchased with different diameter classes, so that we can select an appropriate diameter for a specific protein. CNTs cannot be used as nanotemplates in their raw condition because they are hydrophobic and not soluble in water and also not linear shaped. In addition, the protein needs a biomimetic structure with lipid bilayers. Therefore, CNTs need to be modified by chemically anchoring a lipid bilayer to their surface and to be straightened mechanically. There are different approaches to apply these modifications: CNTs can be functionalized with covalently bound linkers[28] or by π-π-stacking[29][30]. Figure 1.5 shows an illustration of a carbon nanotube. To straighten CNTs they can be cut into smaller pieces using ultrasound.

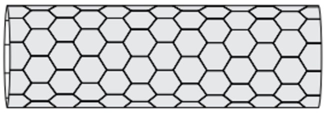


Figure 1.5: Illustration of a carbon nanotube provided by *Elmar Behrmann*. Each vertex of each hexagon represents a carbon atom with three covalent bindings to other carbon atoms in the lattice.

1.4.2 Lipid Tubes

Hydrated dispersions of lipid mixtures including *GalCer* (**Gal**actosyl**Cer**amide) can self-assemble into bilayer cylinders[31]. The preparation protocol is similar to the one of ULVs[32] (see chapter 3.3.1) but does not require to use extruders because GalCer lipids form tubes without any interventions. GalCer tubes are linear shaped and have a length of 250 to 400 nm and diameters of 25 to 30 nm constant over the whole tube. The lipid mixture must consist of at least 20 % GalCer. GalCer is only responsible for the tube extrusion. At lower concentrations there are not enough GalCer molecules present to form tubes. The other components of the lipid mixture are responsible for protein binding. They need to be selected depending on the intended use in terms of molecular properties like charge and the length of the hydrophobic tail in order to mimic a cell membrane for the protein. As a result, GalCer tubes that consist of a suitable lipid mixture can be used as nanotemplates for specific proteins[33].

1.5 3D Electron Microscopy

The structure of proteins on nanotemplates can be studied using *Cryo TEM* (**T**ransmission **E**lectron **M**icroscopy). This method provides us with high-resolution 2D images of the protein of interest. A major pitfall with biological samples in EM is, that they are radiation sensitive. Imaging biological samples in EM is always a compromise between a strong signal on the detector and beam damage of the sample. Recording high resolution micrographs allows only low electron doses before radiation damage destroys the structure of the protein. Using only a non-harmful dose results in a low *SNR* (**S**ignal to **N**oise **R**atio). While today's DEDs (**D**irect **E**lectron **D**etectors) already yield much stronger signals at low doses the SNR is still not good enough to obtain detailed structures of small particles without advanced methods that average multiple micrographs and sum up the containing information[34][35]. Averaging methods get more complex with the resolution they are able to gain. For the protein-extruded tubes and nanotemplates of this work, only methods can be used that provide the analysis of tubular structures. In figure 1.6 is a resolution chart of today's most common appropriate methods to create 3D volumes from 2D micrographs in EM.

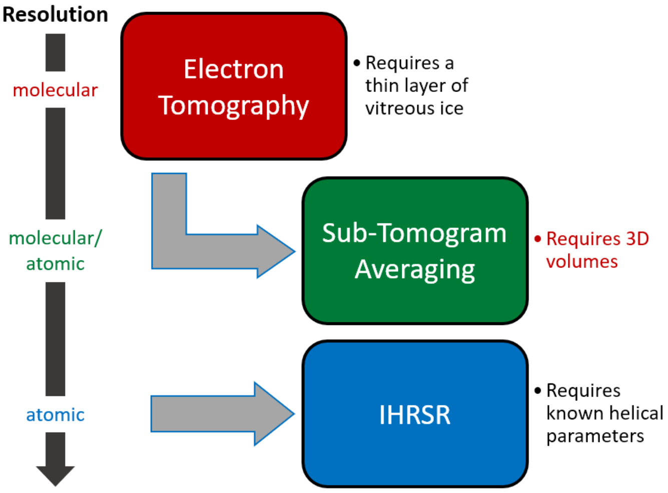


Figure 1.6: Requirements and resolution chart of 3D EM methods suitable for the analysis of tubular structures. While tomography only requires Cryo samples, Sub-Tomogram Averaging can combine the information of multiple sub-volumes of the reconstructed tomogram. IHRSR (**I**terative **H**elical **R**eal **S**pace **R**econstruction) is the most complex analysis method gaining the best resolution for filaments. But it requires known helical parameters guessed or obtained from the other methods.

STA (**S**ub-**T**omogram **A**veraging) can be used to obtain a high resolution structure and requires reconstructed tomograms and not a high concentration. IHRSR builds up on helical parameters. These can be obtained from ET or initially guessed. It is used because it is the most suitable high resolution method for helical structures at high enough concentration.

1.5.1 Electron Tomography

Electron Tomography reconstructs a 3D Volume from 2D micrographs recorded from many different viewing angles. The sample is tilted stepwise between the selected minimum and maximum angles and at every step a micrograph is recorded. The resulting stack of micrographs is aligned via cross correlations. To ensure a precise alignment *fiducial markers* are added to the sample. They consist of gold crystals and add high contrast position markers to Cryo samples. Finally, the different views are back projected to a 3D volume. There are different algorithms to perform this task[36]. The most common ones are *Filtered Back Projection* and *SIRT* (**S**imultaneous **I**terative **R**econstruction **T**echnique). The more angles the tomogram covers the more detailed is the final volume because of the *Fourier Slice Theorem*[37]. It states that a projection in real space provides information as a slice perpendicular to the projection in Fourier space. The process of ET is visualized in figure 1.7.

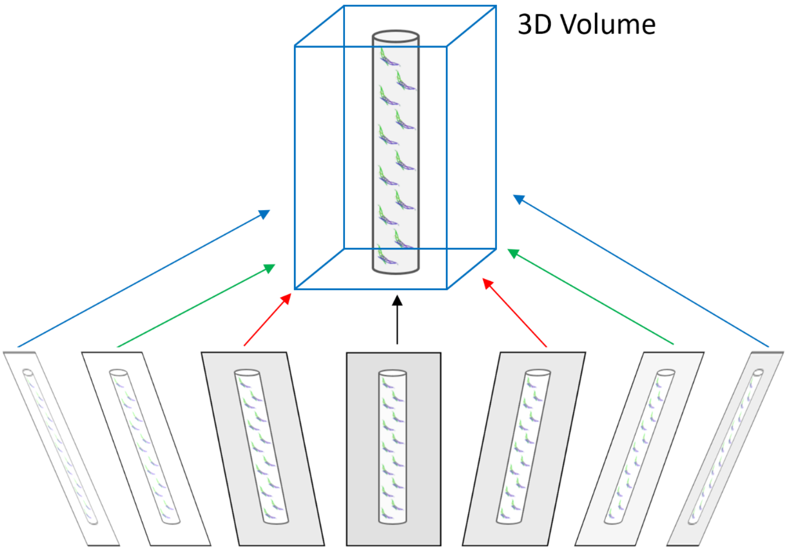


Figure 1.7: Visualization of Electron Tomography: Micrographs from various tilt angles are backprojected to a 3D volume. BAR domain illustration taken from [26].

ET is used to create volumes from unique samples. There are no requirements on the concentration of the sample and symmetry of particles. However, ET has some major disadvantages[38]:

- **Electron Dose:** The sample is exposed to a high electron dose because all micrographs are recorded from the same position of the sample. This allows only low-dosed micrographs and results in a low SNR.
- **Missing Wedge:** The mechanical construction of the goniometers of today's TEMs does not allow to tilt the sample up to 180 °. In case of transmission images, 180 ° covers the complete volume. As a result of limited viewing angles there are missing parts in the 3D volume because no micrographs were recorded at these angles. This is called the *Missing Wedge*. This does not only result in missing views at specific angles. Instead, it results in missing information in the whole volume in Fourier space because of the Fourier Slice Theorem.
- **Alignment mismatches:** ET is significantly more sensitive to mechanical inaccuracies of the microscope stage and beam drift compared to other 3D electron microscopy methods. In case of ET, the volume is reconstructed from micrographs of different viewing angles. Stage or beam misalignments between the micrographs can be compensated only partially and lower the resolution. Other 3D EM methods build up on micrographs recorded at the same angle (perpendicular to the beam).

A tomogram contains a lot more information than a single micrograph. Its only requirements to samples are an appropriate ice thickness and a grid position that does not affect tilts. Although, compared to averaging methods it is not possible to refine the volume with additional information. This problem is solved by Sub-Tomogram Averaging.

Filtered Back Projection

The filtered back projection algorithm is based on the inversed *Radon Transform* $\mathcal{R}^{-1}$[37]. The projection $p(z)$ is convoluted with the high pass filter kernel $g(z)$ in real space. Integration over the polar projection angle ϕ allows to restore the real image $f(x,y)$:

$$f(x,y) = \int\limits_0^\pi \left[\int\limits_{-\infty}^{\infty} p(z') \cdot g(z-z') \, \mathrm{d}z' \right] \mathrm{d}\phi \tag{1.6}$$

$f(x,y)$ depends on the Cartesian coordinates $x = \sin(\phi)$ and $y = \cos(\phi)$. Using the *Fourier Transform* $\mathcal{F}$ allows to simplify the convolution to a multiplication.

$$f(x,y) = \int\limits_0^\pi \mathcal{F}^{-1} \big\{ \mathcal{F}\{p\}(k) \cdot G(k) \cdot W(k) \big\} \, \mathrm{d}\phi \tag{1.7}$$

$G(k) = \mathcal{F}\big\{g(z)\big\} = |k|$ is the filter kernel in Fourier space with the spatial angular frequency k and $W(k) = \mathrm{e}^{-\beta|k|}$ is an additional window function to reduce spectral leakage[39] with the damping factor β.

Simultaneous Iterative Reconstruction Technique

SIRT is an iterative algorithm based on a linear equation system[36]. The vector $\vec{x} \in \mathbb{R}^n$ contains the gray values for each pixel and the vector $\vec{b} \in \mathbb{R}^m$ the projection data. Both are linked via the matrix $A \in \mathbb{R}^{m \times n}$.

$$A\vec{x} = \vec{b} \tag{1.8}$$

The algorithm computes the approximate solution of the projection error $||A\vec{x} - \vec{b}||$ of the equation system. With each iteration k the error is minimized.

$$\vec{x}^{(k+1)} = \vec{x}^{(k)} - \lambda C A^T D\big(A\vec{x}^{(k)} - \vec{b}\big) \tag{1.9}$$

$\lambda \in \mathbb{R}$ is a selectable relaxation parameter and $C \in \mathbb{R}^{n\times n}$ and $D \in \mathbb{R}^{m\times m}$ are diagonal transformation matrices depending on A's main diagonal elements.

$$\begin{aligned} C &= \operatorname{diag}\big(1/c_1, \dots, 1/c_n\big),\ c_j = \sum_{i=1}^{m} a_{ij} \\ D &= \operatorname{diag}\big(1/d_1, \dots, 1/d_m\big),\ d_i = \sum_{j=1}^{n} a_{ij} \end{aligned}$$

The SIRT algorithm is computationally more expensive than back projection but more robust to noise.

1.5.2 Sub-Tomogram Averaging

Sub-Tomogram averaging builds up on ET and requires completely reconstructed 3D volumes from ET. Basically, STA is a combination of SPR (**S**ingle **P**article **R**econstruction) and ET because it averages particles from ET's 3D volume. The usage of this method is only an option if the sample contains periodic patterns that can be averaged up to a 3D model. Figure 1.8 illustrates the process of STA[40].

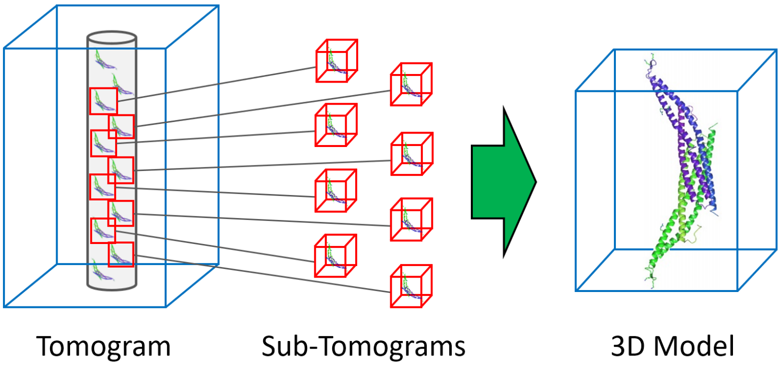


Figure 1.8: Visualization of Sub-Tomogram Averaging: Sub-volumes of the tomogram in figure 1.7 are averaged and refined to a molecular 3D model.

Sub-volumes are picked by the user from reconstructed tomograms. These particles are summed up and sorted into 2D classes. After selecting suitable classes, a 3D model is assembled. STA also needs an initial model of the structure as an input for the refinement. The algorithm behind STA consists of four major steps[41][42]:

1. **Alignment:** Calculate suboptimal rigid transformations of all sub-tomograms to restore their original positions in the volume.
2. **Integration:** Approximate integration over all suboptimal rigid transformations to calculate a first average (via summation). An accurate calculation of the integral would be computationally infeasible.
3. **Averaging:** Use a maximum-likelihood based algorithm to update the subtomogram averages. This refinement process is used instead of an accurate integration from the last step. It is a less expensive computation.
4. **Classification and modeling:** The averages are classified and assembled to a 3D model or refining an initial model.

Electron tomograms lack information in the missing wedge (see chapter 1.5.1). To reconstruct a complete molecular structure using STA, the missing wedge of the tomograms needs to be compensated. Therefore, it is needed to record and reconstruct multiple tomograms that cover different sets of viewing angles. Finally, every viewing angle needs to be included. Compared to other SPR-based reconstruction methods (e.g. IHRSR) this needs to be considered.

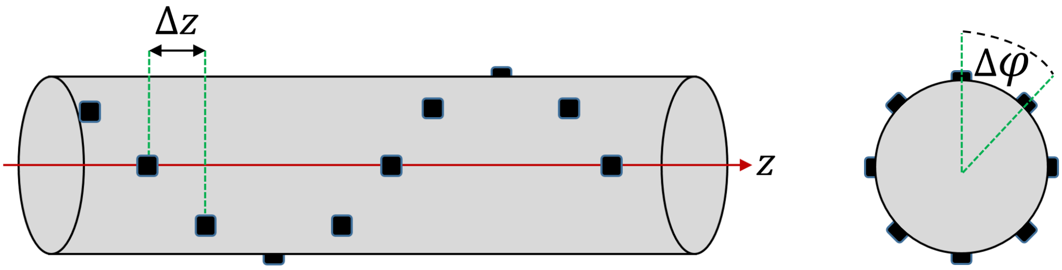
1.5.3 Helical Reconstruction

The most common helical reconstruction method is *IHRSR* (**I**terative **H**elical **R**eal **S**pace **R**econstruction) and can obtain atomic resolution models of helical structures. Compared to its predecessor, *Fourier-Bessel*, it is a real space reconstruction method and based on SPR. There have already been reconstructions of helical structures in real space before[43][44] but IHRSR is the most robust and flexible approach[45]. IHRSR averages up periodic repeats in helical protein assemblies around filaments.

To extract particles from the helices it needs two parameters to be known: The azimuthal rotation per subunit $\Delta\varphi$ and the axial subunit translation Δz. These *helical parameters* describe the basic geometry and repeats of the helix using a mathematical function $f(r,\varphi,z)$[46]:

$$f(r,\varphi,z) = f\big(r,\varphi + n\Delta\varphi, z + n\Delta z\big),\ n = \pm1, \pm2, \dots \tag{1.10}$$

Δz is shown on the illustrated helix in figure 1.9a and $\Delta\varphi$ in the helix top view in 1.9b.



(a) Side view. **(b)** Top view.

Figure 1.9: Helical parameters shown in a virtual tube. Each black dot represents a unit of the helix[46]. Δz is the axial subunit translation and $\Delta\varphi$ the azimuthal rotation per subunit.

Helical particles are extracted from filaments picked by the user in dataset micrographs. To provide more information about the helical symmetry most efficiently, two succeeding extracted particles need an overlap[45]. Extracted particles are back projected to a 3D volume (see chapter 1.5.1). To impose helical symmetry to the 3D volume a density array ρ_{cart} between the voxels is calculated and then converted from Cartesian coordinates to ρ in cylindrical coordinates:

$$\rho_{\text{cart}}(x,y,z) \rightarrow \rho(r,\varphi,z) \tag{1.11}$$

The helical axis z is set along the radius r at $r = 0$ (*red* line in figure 1.9a). The density function then follows the same symmetry as equation 1.10:

$$\rho\big(r, \varphi + n\Delta\varphi, z + n\Delta z\big) \tag{1.12}$$

The helical symmetry is now determined and applied to the volume by minimizing 2^{nd} degree polynomials for each of the two parameters $\Delta\varphi$ and Δz using the mean-squared density $\langle\rho^2\rangle$:

$$\langle\rho^2\rangle = a_1 + b_1\Delta\varphi + c_1(\Delta\varphi)^2 \;\rightarrow\; \Delta\varphi = \frac{-b_1}{2c_1}$$
$$\langle\rho^2\rangle = a_2 + b_2\Delta z + c_2(\Delta z)^2 \;\rightarrow\; \Delta z = \frac{-b_2}{2c_2}$$

a_i, b_i and c_i are coefficients of terms of each degree in the polynomials. The minimization of the equations is done iteratively. The helical parameters can be obtained from ET and STA as shown in figure 1.6 or by initially *guessing* the parameters and refining them in successive reconstructions. With (approximately) known helical parameters the process of IHRSR is shown in figure 1.10. Due to its derivation from SPR, IHRSR needs tubular fragments to be picked from micrographs of datasets and an initial model.

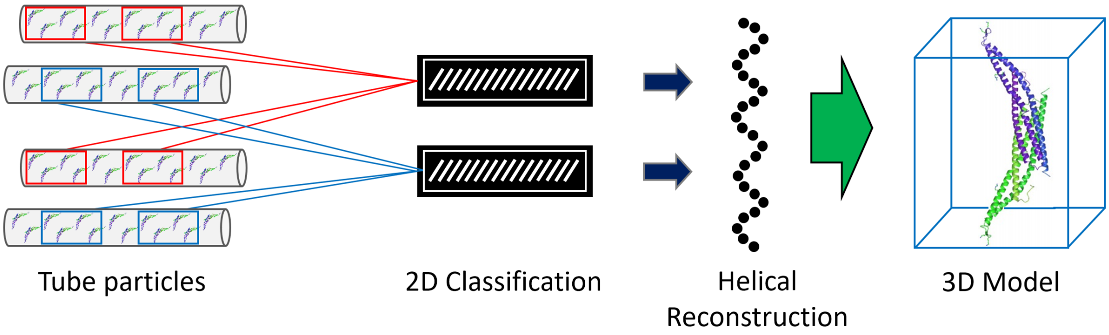


Figure 1.10: Visualization of IHRSR: 2D projections from helical particles are classified and averaged to reconstruct the helix. From the helix an atomic 3D model of the protein can be obtained.

The detailed process of IHRSR is described based on *RELION*[47]:

1. **Particle Extraction:** With help of the helical parameters particles are extracted from picked tube fragments.
2. **2D Classification:** The extracted particles are averaged up and classified iteratively.
3. **3D Classification:** Helical 3D classes are assembled from selected 2D classes.
4. **Molecular Model:** A refined helical 3D model of the helix based on selected 3D classes can be used to obtain a molecular model of the protein forming the assembly.

For the extraction of a preferably homogeneous set of particles it is important that the helical assembly features a constant diameter and the absence of curvatures. Otherwise the helical parameters do not fit anymore and particle views may be distorted by wrong geometries. For the required initial model, IHRSR is robust enough to use a featureless cylinder as an initial 3D template.

1.6 Central Composite Design

The *CCD* (**C**entral **C**omposite **D**esign) method is used to study the influence of more than one variable on the experiment's results. It requires less experiments in total compared to the analysis of each variable separately. Moreover, it can also show quadratic dependencies. The number of required experiments for a CCD can be calculated by equation 1.13. It contains a quadratic term and a linear term and the central point[48].

$$N = 2^k + 2k + C_p \tag{1.13}$$

In case of this work the CCD was customized to optimize the lipid tube concentration in chapter 3.3.2. In this case we are not interested in the quadratic interaction term of the CCD. Thus, its equation simplifies to 1.14[49]:

$$N = 2k + C_p \tag{1.14}$$

The CCD was applied on three important variables of the tube preparation process: The number of freeze-thaw-cycles n_{FT}, the extruder temperature T_{Ex} and the number of extrusion cycles n_{Ex}. It is assumed that the tube count correlates linear with the concentration c_{Lipid}. Thus, the concentration is kept constant and the other influencing variables are being optimized. With $k = 3$ variables the CCD needs nine experiments. The variables span a spatial orthonormal basis in a coordinate system. The axis assignment is selectable and was set as the following:

$$\vec{x}_1 = n_{\mathrm{FT}}\,\vec{e}_{x_1},\ \vec{x}_2 = T_{\mathrm{Ex}}\,\vec{e}_{x_2},\ \vec{x}_3 = n_{\mathrm{Ex}}\,\vec{e}_{x_3}$$

Figure 1.11 shows the spatial coordinate system with the plotted CCD points.

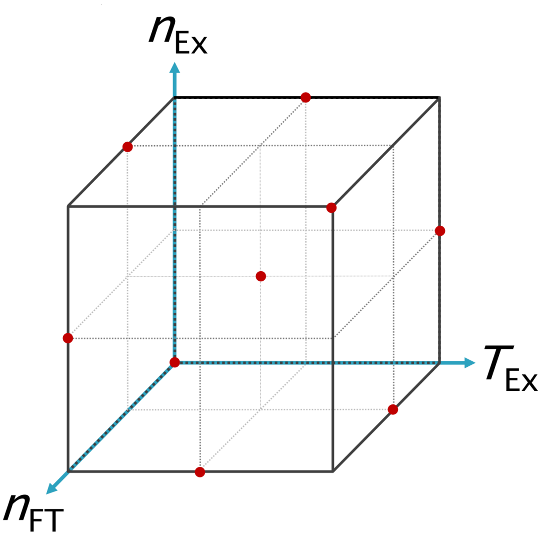


Figure 1.11: Orthonormal basis of CCD: Each red dot represents a coordinate point of the CCD.

In case of the tubes the concentration is denoted as the count of the tubes in micrographs of a defined area. Thus, lipid nanotube extrusion is described as a function of the variables $n_{FT}, T_{Ex}, n_{Ex} \in \mathbb{R} \geq 0$ that maps on the tube count N:

$$N = f\left(n_{\mathrm{FT}}, T_{\mathrm{Ex}}, n_{\mathrm{Ex}}\right) \in \mathbb{R} \tag{1.15}$$

The CCD model interprets the variables as coordinates. The function f is a scalar field. The gradient of the function f points in the direction of the maximum change $\vec{r}$ in a spatial vector base:

$$\vec{r} = \nabla f\big(n_{\mathrm{FT}}, T_{\mathrm{Ex}}, n_{\mathrm{Ex}}\big),\ \ f : \mathbb{R} \to \mathbb{R}^3 \tag{1.16}$$

An example for the gradient is illustrated in figure 1.12 in advance of the results shown in chapter 4.3.2 and discussed in chapter 5.4.

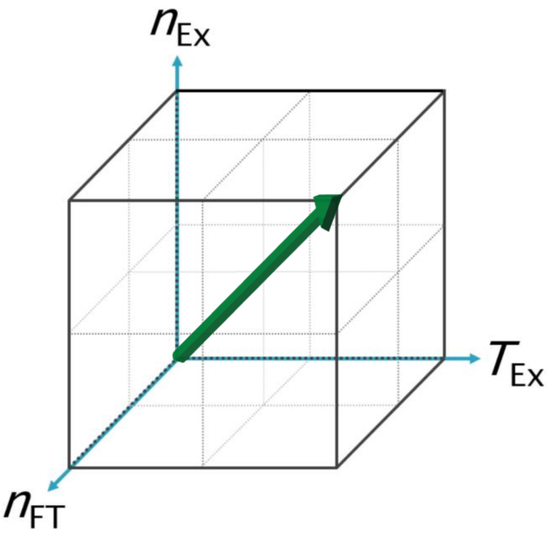


Figure 1.12: Estimated gradient of CCD in chapter 1.6.

For the optimization of the lipid tube concentration the gradient of the CCD is the most important result because it shows in which composition of the variables the highest concentration is achieved and which variables are the most responsible ones.

Chapter 2

Aim of this work

Some classes of tubular geometry sensing proteins extrude lipid tubes from unilamellar vesicles *in vitro*. A good approach to analyze structural and functional properties of the protein is the analysis of these lipid tubes. In this work we want to develop a method that enables to observe the oligomerization and understand how these proteins act on a membrane and exert forces to bend membranes. Understanding the dynamics of intramolecular interactions between protein domains is not convenient with methods like X-ray crystallography that require crystallization and is far away from meaningful environments. To analyze different snapshots of the functional states of a protein we intend to use Cryo-EM. This technique allows to shock freeze aqueous *in vitro* reactions in intermediate states.

We want to use the truncate BPG of the protein GRAF1 as a representative of the tubular geometry sensing protein class that form molecular oligomers on lipid tubes. In the first step we want to review the analysis of protein-extruded lipid tubes in Cryo-EM and characterize their oligomeric organization.
However, protein-extruded tubes do not enable the analysis of protein binding in lower concentrations. This is needed to observe intermediate binding states. Therefore, we want to use nanotemplates as pre-tubular rigid structures to bind the protein and stop the binding reaction in different states via plunge freezing

for analysis in Cryo-EM. Cylindrical nanotemplates suitable for binding protein oligomers are not a novel idea but most of them suffer from complexity and expensive components. In this work we want to develop an easy and robust approach for nanotemplates and compare them to the protein-extruded tubes.

Finally, we intend to use sub-tomogram averaging or IHRSR to prove that our approach for nanotemplates satisfies our expectations in a serious protein analysis.

Chapter 3

Materials and Methods

3.1 Chemicals and Consumables

Table 3.1: Substances used for lab experiments.

Substance	**Formular**	**Grade**	**Source**
BPG protein	N/A	81.3 µM in Buffer solution	M.K. Francis
Calcium Chloride	$CaCl_2$	dehydrated powder, $\geq 98\,\%$	Carl Roth
Chloroform	$CHCl_3$	anhydrous, $\geq 99\,\%$	Sigma Aldrich
Dithiothreitol (DTT)	$C_4H_{10}O_2S_2$	$\geq 97\,\%$	Sigma Aldrich
EDTA	$C_{10}H_{16}N_2O_8$	$\geq 99\,\%$	Sigma Aldrich
EGTA	$C_{14}H_{24}N_2O_{10}$	$\geq 99\,\%$	Sigma Aldrich
Graphene Oxide	$C_xO_yH_z$	dispersion in H_2O, $2\,\frac{mg}{m\ell}$	Sigma Aldrich
HEPES	$C_8H_{18}N_2O_4S$	$\geq 99.5\,\%$	Carl Roth
Magnesium Chloride	$MgCl_2$	anhydrous, $\geq 98.5\,\%$	Carl Roth
Octyl Maltoside, Fluorinated	$C_{20}H_{25}F_{13}O_{11}$	*Anagrade*	Anatrace
Poly-L-Lysine Hydrobromide	$C_6H_{15}BrN_2O_2$	solid, mol wt 150-300 MDa	Sigma Aldrich
Sodium Chloride	NaCl	$> 99.8\,\%$	Carl Roth
Uranyl Acetate Dihydrate	$C_4H_6O_6U \cdot 2H_2O$	2 % in H_2O	Science Services
Distilled Water	H_2O	Barnstead MicroPure UF/UV	Thermo Fisher Scientific

Table 3.2: Used lipids.

Lipid	**Name**	**Source**
Brain Extract from bovine brain Type I, *Folch* Fraction I	Folch	Sigma Aldrich

Table 3.3: Consumables used for lab experiments.

Consumable	**Source**
Autogrids + C-Clips	Thermo Fisher Scientific
Carbon Support Films Au 200 mesh	Quantifoil
Eppendorf Safe-Lock tubes 1.5 $m\ell$	Eppendorf, *0030120086*
Fiducial Gold 10 nm	UMC Utrecht
Filter Supports 10 mm	Avanti Polar Lipids, *610014*
Syringe needles	Hamilton blunt tips
Holey Carbon Films R2/1 Au 200 mesh	Quantifoil
Holey Carbon Films R2/1 Au 200 mesh + 5 nm C	Quantifoil
Pipette tips	Eppendorf epT.I.P.S.
Polycarbonate Membranes 0.8 µm 19 mm	Avanti Polar Lipids, *610009*
Sample vials 2 $m\ell$ Rotilabo	Carl Roth, *E159.1*
Screw caps with bore hole	Carl Roth, *KE36.1*
UranyLess EM Stain	Electron Microscopy Sciences, *22409*
Whatman filter paper, Grade 2 50 mm	Sigma Aldrich, *WHA1004050*
Whatman filter paper, Grade 4 50 mm	Sigma Aldrich, *WHA1004050*

3.1.1 Buffer Solutions

- **BPG B**: 1 mM EDTA, 20 mM HEPES, 75 mM NaCl, pH 7.4, *adjusted by* NaOH.
- **BPG B$^{\mathbf{EGTA}}$**: 1 mM EGTA, 20 mM HEPES, 75 mM NaCl, pH 7.4, *adjusted by* NaOH.
- **BPG B2**: 20 mM HEPES, 75 mM NaCl, pH 7.4, *adjusted by* NaOH.
- **BPG B3$^{\mathbf{MgCl_2}}$**: 1 mM EDTA, 1 mM EGTA, 20 mM HEPES, 5 mM $MgCl_2$, 75 mM NaCl, pH 7.4, *adjusted by* NaOH.
- **BPG B3$^{\mathbf{CaCl_2}}$**: 5 mM $CaCl_2$, 1 mM EDTA, 1 mM EGTA, 20 mM HEPES, 75 mM NaCl, pH 7.4, *adjusted by* NaOH.
- **EHD2 E**: 30 mM ATP, 5 mM $CaCl_2$, 2.5 mM DTT, 1 mM EDTA, 1 mM EGTA, 20 mM HEPES, 20 mM $MgCl_2$, 150 mM NaCl, pH 7.5, *adjusted by* NaOH.

3.2 Devices and Software

Table 3.4: Instruments used for lab experiments.

Instrument	Manufacturer	Name
Centrifuge	VWR	Micro Star 17R
Cryo Chamber	Leica	EM UC7 / EM FC7
Glow Discharger	Balzers	MED 010
Syringes	Hamilton Gastight	
Incubation Shaker	INFORS HT	Ecotron
Lipid Extruder	Avanti	Mini Extruder
Magnetic stirrer/Heat plate	IKA	C-MAG HS 7
pH Meter	Mettler Toledo	FE20 FiveEasy
Pipettes	Eppendorf	Research Plus
Plunge Freezer	Thermo Fisher	Vitrobot Mark IV
Plunge Freezer	Leica	EM-GP
Precision Scale	Sartorius	BP 211 D
Thermomixer	Eppendorf	Thermomixer C
Thermomixer	Eppendorf	Thermomicer Compact
Ultrasonic bath	Bandelin	Sonorex
Vortexer	Scientific Industries	Vortex-Genie 2

Table 3.5: Software used for data processing.

Name	Version
EMAN2 / sparx	2.21
Gctf	0.50
imod / etomo	4.9.2
Leginon	3.1
MotionCor2	1.2.0
RELION	3.0
SerialEM	3.6

3.2.1 Electron Microscopes

For this work, two transmission electron microscopes were used. For all negative stained grids and testing of Cryo conditions the *JEOL JEM-2200FS* was used (shown in figure 3.1a). It is equipped with a 200 kV field emission gun and a *TVIPS*

F416 scintillator CMOS camera. For all Cryo data acquisitions the *Thermo Fisher Titan Krios* was used (shown in figure 3.1b). It uses a 300 kV field emission gun, a CS-corrector, and the *Thermo Fisher Falcon II* direct electron detector.

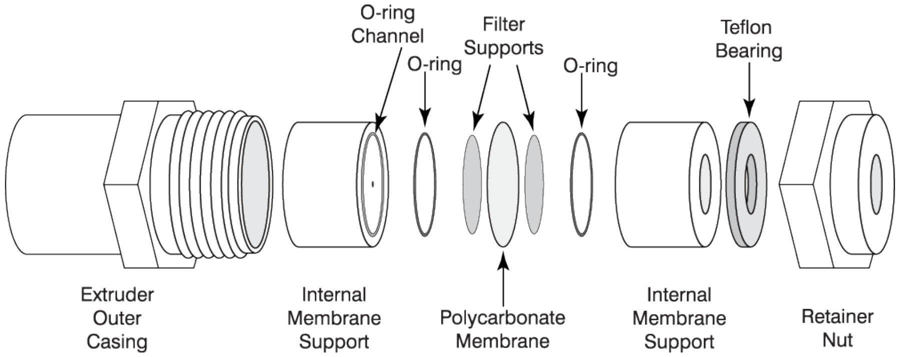
(a) JEOL JEM-2200FS. **(b)** Thermo Fisher Titan Krios.

Figure 3.1: Transmission electron microscopes used for this work.

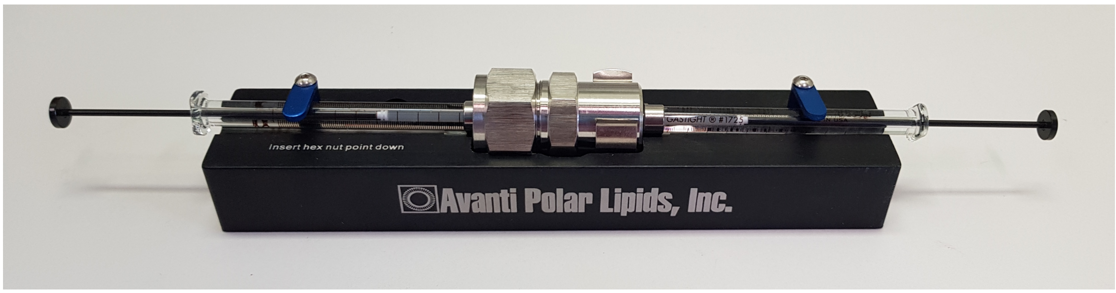
3.3 Methods

3.3.1 Lipid Extruder and Unilamellar Vesicles

For the preparation of *ULVs* (**Uni****L**amellar **V**esicles) 4 mg of in chloroform dissolved Folch *Type I* lipids were dried in a 2 $m\ell$ glass vial in Argon stream. Residual solvent was evaporated in the vacuum desiccator overnight. The lipids were rehydrated in 250 $\mu\ell$ buffer solution B ($4\,\frac{mg}{m\ell}$) for 20 min. To disrupt multilamellar vesicles the solution was frozen on dry ice and thawed at 25 °C for 5 min in *Thermomixer* for five times. The Lipid extruder was assembled with filter supports and 800 nm polycarbonate membrane. Figure 3.2a shows a scheme of the extruder. It was flushed five times with buffer to avoid air gaps in the syringes. Lipid solution was taken using the right syringe and extruded 21 times through the membrane[50]. Figure 3.2b shows an image of the extruder in use with the two syringes on the left and on the right side. Lipid vesicles were taken from the left side of the extruder to avoid lipid aggregates and multilamellar vesicles. Unilamellar vesicles were stored for a maximum of 3 days at 4 °C in *Eppendorf tubes* and used at the earliest one hour after preparation.



(a) Scheme of the Avanti Lipid Extruder[51]. The central part is a polycarbonate membrane between two filter supports.



(b) Image of the lipid extruder in use.

Figure 3.2: Lipid extrusion method.

3.3.2 Lipid Nanotubes

The preparation of Lipid Nanotubes was derived from 3.3.1. In chloroform dissolved Folch lipids were dried in 2 $m\ell$ glass vials using an Argon stream and kept in vacuum overnight. They were rehydrated in the selected buffer solution (table 3.1.1) for 20 min. The solution was sonicated until all lipids were resuspended. They were frozen and thawed and briefly vortexed for a defined number of times. Liquid nitrogen was used for cooling instead of dry ice in the ULV protocol. The solution was frozen in aliquots of 200 $\mu\ell$ and stored at -80 °C until use. The preparation procedure is illustrated in figure 3.3. One aliquot was thawed and vortexed and extruded using a polycarbonate membrane for an odd number of times. Using an odd number ensures that only lipid solution is saved that passed the extruder membrane. The lipid tube solution was stored at room temperature and used for a maximum of 5 days.

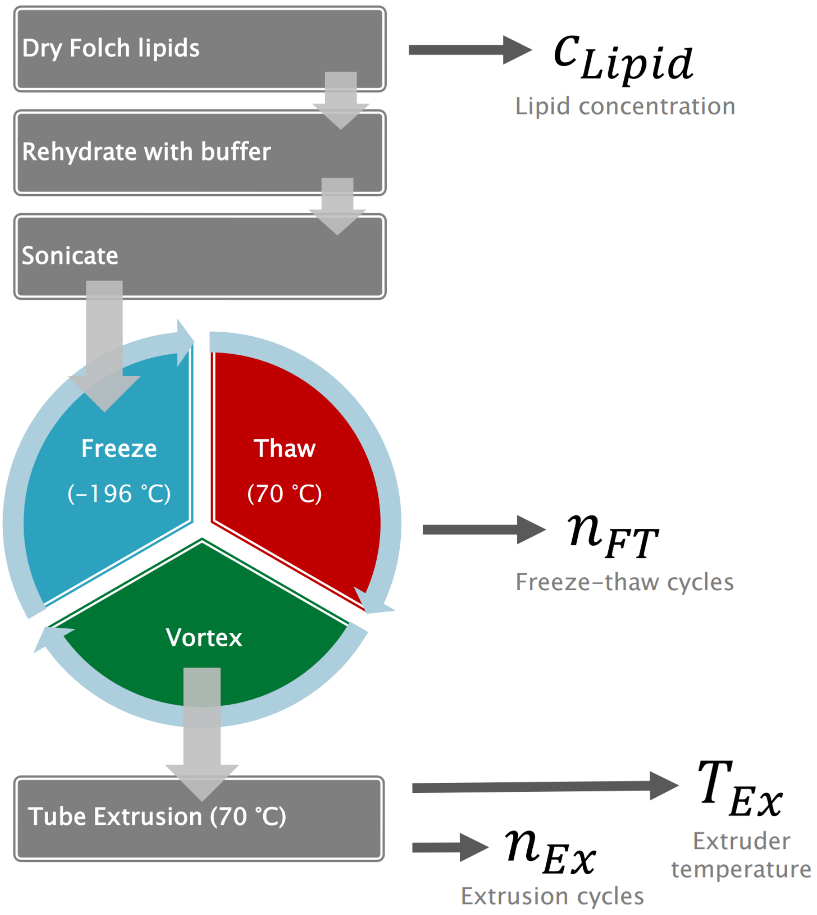


Figure 3.3: Preparation process of Folch Lipid Tubes and its variables.

For further optimization of the preparation process the used variables and constants are listed below:

Variables in the tube extrusion process in figure 3.3:

- $\mathbf{c}_{\text{Lipid}}$: Lipid concentration dissolved in buffer solution.
- $\mathbf{n}_{\text{FT}}$: Count of freeze-thaw-cycles after rehydrating dried lipids in buffer solution.
- $\mathbf{T}_{\text{Ex}}$: The extruder is precooled or preheated to a defined temperature before extruding lipids.
- $\mathbf{n}_{\text{Ex}}$: Count of extrusion cycles. Each cycle corresponds to pressing the lipids in one direction from one syringe to the other.

Constants in the tube extrusion process in figure 3.3:

- **Freezing and thawing temperatures:** Liquid nitrogen was used to cool the lipid solution down to -196 °C. The solution was thawed in the Thermomixer for 2:45 min at 70 °C.
- **Solution temperature at extrusion process:** The last thawing step is followed by the extrusion. Thus the temperature of the solution is 70 °C.
- **Extrusion Volume:** All extrusions were performed with a constant volume of 250 $\mu\ell$. The concentration was only varied with the amount of lipids c_{Lipid}.

The lipid nanotube experiment was performed using the parameters in table 3.6:

Table 3.6: Lipid nanotube preparation.

Buffer	$\mathbf{c}_{\text{Lipid}}$	$\mathbf{n}_{\text{FT}}$	$\mathbf{T}_{\text{Ex}}$	$\mathbf{n}_{\text{Ex}}$
B	$4\,\frac{mg}{m\ell}$	5 ×	25 °C	21 ×

3.3.3 Optimization of Nanotemplates

The first step to optimize the yield of lipid nanotubes was to increase the concentration c_{Lipid} of the lipids from $4\,\frac{mg}{m\ell}$ to $8\,\frac{mg}{m\ell}$ and $12\,\frac{mg}{m\ell}$. At higher concentrations undissolved lipid aggregates remained in the solution.

Dependency on Buffer Components

To investigate the dependency of the tube formation on buffer components different concentrations and components were used:

Buffers (see chapter 3.1.1) used for tube formation:

1. **B2** (omitting EDTA): In the first step EDTA was removed from the original buffer.
2. $\mathbf{B}^{\mathbf{EGTA}}$ (substitute with EGTA): As a negative control to buffer B, 1 mM EGTA instead of EDTA was used.
3. $\mathbf{B3}^{\mathbf{MgCl_2}}$ (adding EGTA and $MgCl_2$): 1 mM EGTA and 5 mM magnesium chloride were added to the original buffer.
4. $\mathbf{B3}^{\mathbf{CaCl_2}}$ (adding EGTA and $CaCl_2$): 1 mM EGTA and 5 mM calcium chloride was added.

All other parameters were adopted from table 3.6.

3.3.4 BPG Tubulation Experiment

For the tubulation experiments on ULVs and LNTs the fragment BAR-PH-GAP of the protein GRAF1 (see chapter 1.3) was used[21]. It was purified and provided by *Monika Francis* from Umeå University on 5th January 2016 with a concentration of 81.3 µM in a buffer solution containing 25 mM HEPES and 150 mM NaCl. The protein stock was first diluted with buffer to the half of the concentration and then mixed with ULVs or LNTs. In table 3.7 the detailed experiment parameters are specified.

Table 3.7: BPG Experiment.

Template	Buffer	Conc.	BPG conc.	Mixer	Temp.	Incubation
ULV	B	$2\,\frac{mg}{m\ell}$	6.38 µM	300 rpm	25 °C	10 min
LNT	$B3^{\mathrm{CaCl_2}}$	$6\,\frac{mg}{m\ell}$	6.38 µM	0 rpm	25 °C	20 min

After the tubulation experiment the solution was used immediately for negative staining or Cryo-EM.

3.3.5 Glow Discharge

By default, the carbon film on EM grids is hydrophobic. To incubate it with an aqueous sample solution it must be hydrophilic. There are chemical methods and glow discharging. The advantage of glow discharging grids is a more homogeneous result and a less complex process. For this work, the grids were glow discharged for 2 min using *Balzers MED 010* with an Argon plasma. The chamber was at first pumped to a vacuum of 10^{-7} mbar. Secondly, a balance between the vacuum pumps and Argon pressure was adjusted to 3.5 mbar. The voltage between the electrodes was constant at 4.25 kV and the plasma current at 40 to 50 mA. The grids were used within a maximum of one hour after glow discharging.

3.3.6 Negative Staining

Due to the low contrast of light elements in biological samples grids were negative stained with a heavy metal coat. 3.5 $\mu\ell$ of liquid sample was pipetted on the Carbon film side of a glow-discharged TEM grid and incubated for 1 min and then blotted away with *Whatman 2* paper. The grid was washed two times with buffer solution and once with staining solution. Uranyl acetate or *UranyLess* was used as staining solution and applied for 35 s or 10 s, respectively. The staining solution was blotted away and dried by airflow. In case of using UranyLess the grids were additionally dried in vacuum for at least 1 h to evaporate remaining water in the stain coat preventing bubbles when exposed by the electron beam.

3.3.7 Fiducial Markers for Tomography

For an optimized tracking of the sample position in different viewing angles the samples for tomography were mixed with fiducial markers. These are gold crystals with about 10 nm diameter. They were mixed with the sample immediately before usage in negative staining or plunge freezing. They were added in a ratio of 1 : 10 to the sample volume.

3.3.8 Graphene Oxide and Poly-L-Lysine

Glow discharged holey carbon grids for Cryo-EM were coated with graphene oxide and Poly-L-Lysine[52][53]. The graphene oxide solution was diluted from 2 $\frac{mg}{m\ell}$ to 0.2 $\frac{mg}{m\ell}$ with H_2O and centrifuged at 300 g for 30 s. 20 $\mu\ell$ was added to a glow discharged grid. After 2 min the grid was blotted using Whatman **2** filter papers and washed three times on 30 $\mu\ell$ H_2O (two times on the front and once on the back side). The grid was put on 20 $\mu\ell$ of Poly-L-Lysine solution at 1 $\frac{mg}{m\ell}$ for 1 min and again washed two times on 30 $\mu\ell$ H_2O and dried in air. There was no more need to glow-discharge the grids before using them because of the hydrophilic Poly-L-Lysine coating.

3.3.9 Plunge Freezing

For plunge freezing *Thermo Fisher Vitrobot Mark IV* or *Leica EM-GP* were used. Individual parameters are listed in table 3.8. Holey carbon grids coated with graphene oxide from chapter 3.3.8 were used for plunge freezing. The desired humidity and temperature for the chamber of the used blotter were set. The sample was mixed with surfactant solution. When the blotter was ready to use with the previously set parameters the blotting papers were wetted (in case of Vitrobot) and inserted. Gaseous ethane was condensed in a container cooled down by liquid nitrogen. Blotting was started when the liquid ethane reached a temperature of a maximum of -170 °C. 4 $\mu\ell$ of the sample was pipetted on the carbon side of the grid. After the blotting procedure the grid was left in liquid ethane for 1 min to avoid devitrification and stored in liquid nitrogen.

Freezing Parameters

Table 3.8: Plunge freezing parameters.

	Vitrobot Mk IV	EM GP
Blotting paper	**wet** *Whatman 4*	**dry** *Whatman 2*
Blotting time	3.5 s	4 s
Cryogen temperature	-170 °C	-180 °C
Humidity	100 %	80 %
Temperature	23 °C	20 °C

Wetting Blot Papers

For the Vitrobot 3 *Whatman 4* papers were put in a petry dish and homogeneously wetted with 1 $m\ell$ of buffer solution. The top paper was used as blot paper for the front side of the grid and the middle paper was used as blot paper for the back side of the grid. The bottom paper was discarded.

Surfactant Preparation

For a more homogeneous ice layer on the grid a surfactant was added to the sample before plunge freezing. As surfactant fluorinated Octyl Maltoside dissolved in H_2O with a concentration of $0.1\,\frac{mg}{m\ell}$ was used. The sample solution was diluted by 10 % of its volume with the surfactant solution.

3.3.10 Tube Counting

A tube was only counted if it could be identified as a (linear shaped) tube and if its length and diameter was at least equal to the minimum length and diameter of the smallest BPG-coated tubes in chapter 4.2.1. Aggregates were not counted. An object was accepted as linear shaped when it was possible to draw a line through the whole object parallel to its borders and the line did not touch the outside. If one end of the tube was invisible, or the tube left the image it was not counted because of falsifying the result in case that the major part of the tube is outside of the image. An illustration of the tube counting method can be found in figure 3.4.

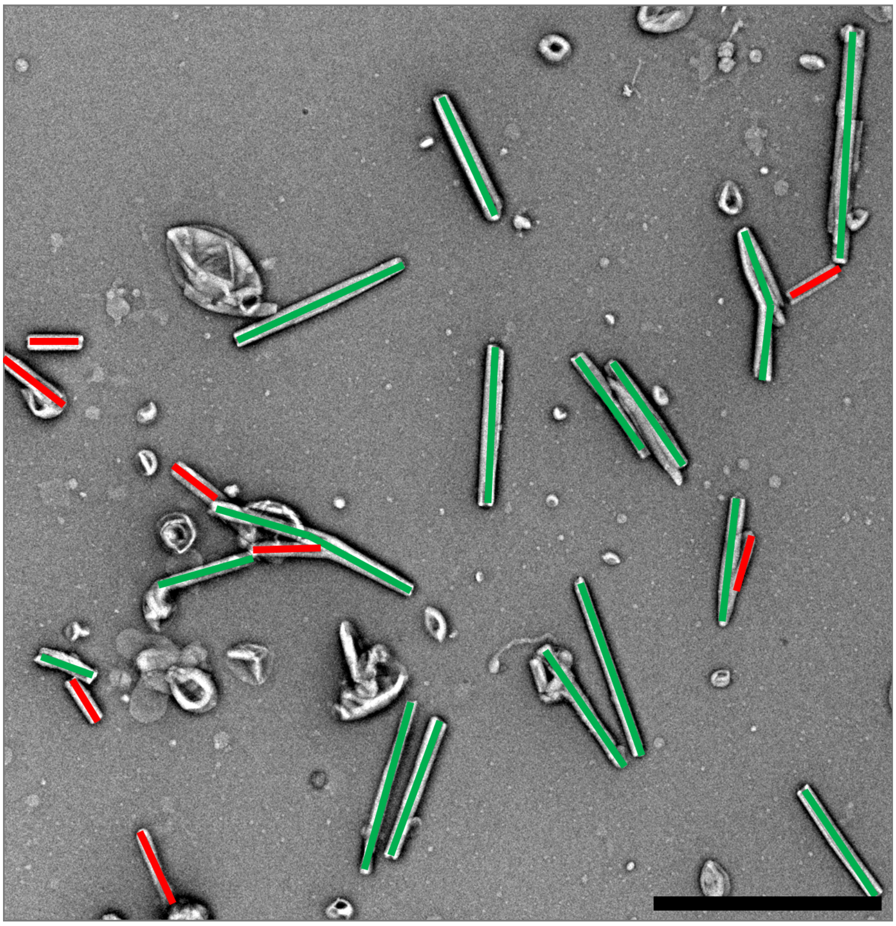


Figure 3.4: Illustration of tube counting method. Scale bar: 2 µm. The green colored tubes were counted. The red ones were ignored because of their size or one invisible end.

All tubes were counted manually with the following settings:

Settings for counting tubes

- JEOL JEM-2200FS microscope
- Quantifoil Au 200 mesh Carbon Support Films stained with *Uranyless* solution for 10 s
- 8000 × magnification. Pixel size: 1.96 $\frac{nm}{px}$. Image size: $(4096\,\mathrm{px})^2$
- Area: $(6.028\,\mu\mathrm{m})^2$
- Tube count of images averaged

3.3.11 Manual Tube Measurement

Lipid nano tube sizes were measured manually using the straight-line tool in *FIJI*. The length and diameter were measured in the micrographs in pixels and then multiplied by the pixel size of the microscope. The sizes of a minimum of 50 tubes were averaged. The micrographs were obtained with the following settings:

Settings for measuring tubes

- Thermo Fisher Titan Krios microscope
- Leica EM-GP (see chapter 3.8)
- 5000 × magnification. Pixel size: 1.28 $\frac{nm}{px}$. Image size: $(4096\,\mathrm{px})^2$
- Area: $(5.242\,\mu\mathrm{m})^2$

3.3.12 Negative Stain Tomography

Electron tomograms were recorded using the software *SerialEM* [54]. After a suitable target for a tilt series was found on the grid, the goniometer was tilted to the highest possible angle in both directions without being affected by grid bars. The tilt-series for the tomograms were acquired automatically in tilt-steps of 2 °.

Tomogram reconstruction was done via *etomo* script[55] that accesses the software *imod*[56][57]. The used reconstruction algorithm was *Back Projection*[58]. For a precise alignment of the micrographs a fiducial seed model was used.

3.3.13 Cryo Data Acquisition

Data in Cryo-EM were acquired using the *Thermo Fisher Titan Krios* microscope equipped with a *Thermo Fisher Falcon II DED* (**D**irect **E**lectron **D**etector). The DED records 17 frames per second. In the original setup these are added up and saved as a micrograph. In the context of a master thesis written by *Daniel Rudolph*[59] a frame grabber computer was installed. It captures the frames from the DED via an optical fiber and saves them as separate files. This allows to correct the beam induced motion by cross correlation. For this task, the software MotionCor2[60] was used. Besides, the software can dose-filter the micrographs by omitting high dose frames. This is important because these frames lack high frequency information due to beam damage and disrupt the sample signal and notably the *CTF* (**C**ontrast **T**ransfer **F**unction) estimation. In case of this work only motion-corrected and dose-filtered frame sums were used. For electron tomography and single particle acquisition the grids were screened manually to evaluate the ice quality and sample concentration. General parameters for all Cryo acquisition methods are listed in table 3.9.

Table 3.9: Cryo-EM data acquisition parameters.

Acceleration voltage	**Objective aperture**	**Spherical aberration**
300 kV	100 µm	$c_s = 0.0001\,\text{mm}$

The detailed parameters for each dataset can be found in table 3.10.

Cryo Electron Tomography

Cryo electron tomogram acquisition and reconstruction is similar to negative stain tomography described in chapter 3.3.12. The main difference is the use of a dose-symmetric tilt-scheme[61]. Usually, the stage is tilted continuously and micrographs for all angles are recorded in the following order:

$$-60\,^\circ, -58\,^\circ, -56\,^\circ, -54\,^\circ, \ldots, 54\,^\circ, 56\,^\circ, 58\,^\circ, 60\,^\circ$$

For the dose-symmetric tilt-scheme the angles around $0\,^\circ$ are recorded first:

$$0\,^\circ, 2\,^\circ, -2\,^\circ, -4\,^\circ, \ldots, -56\,^\circ, -58\,^\circ, 58\,^\circ, 60\,^\circ$$

The advantage is that low angles are recorded at low doses. These micrographs contain the most information in contrast to the ones recorded at high angles. This tilt-scheme ensures to keep the radiation damage as low as possible for low-angle micrographs in the tomogram. Unfortunately, imod does not support reconstruction methods for dose-symmetric tilt-schemes by default. The micrographs of the tomogram must be sorted by their angles before building tomograms. For this purpose, a program called imod-prepare-stack was written in C++ (Appendix B). It takes the information about the micrographs from SerialEM's .mdoc file and sorts the related micrographs by angle. Finally, a new .mdoc file is created that can be used in imod. In figure 3.5, a scheme of imod-prepare-stack is shown.

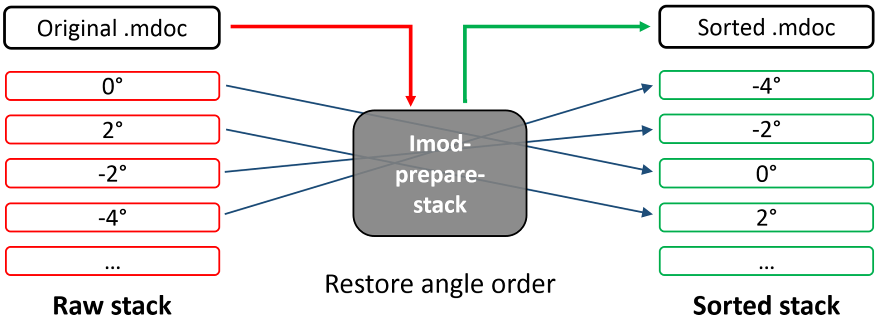


Figure 3.5: imod-prepare-stack reads the .mdoc file and uses it to restore the correct angle order of the frame stack. It also writes a new, sorted .mdoc file.

The final reconstruction does not differ from negative stain tomogram reconstruction.

Dataset Acquisition

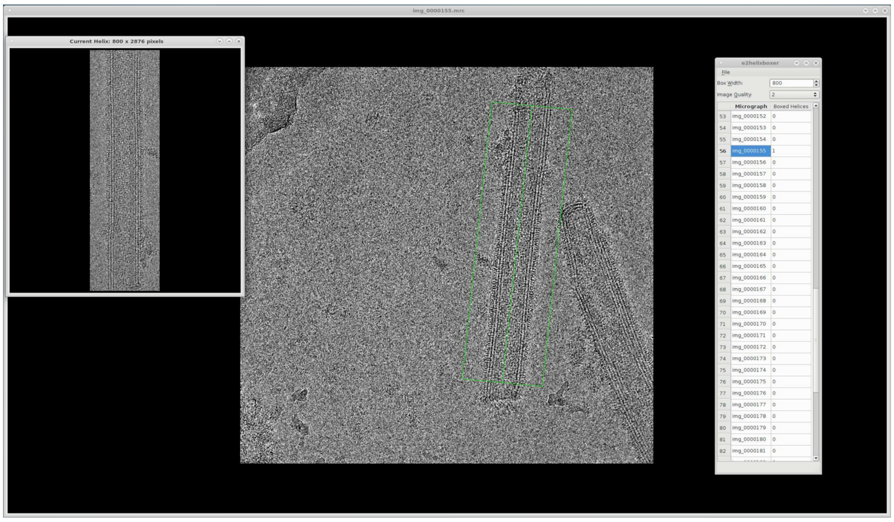
Single particle datasets were recorded using the software *Leginon*[62]. First, a low-resolution map of the grid was recorded using the *Gatan Orius SC1000* (scintillator CCD camera). Suitable squares and holes on the grid were preselected by ice quality. Each hole was previewed at lower magnification to identify tubes and select them for the acquisition. Immediately before acquiring all selected tubes, the CS-corrector was adjusted. In table 3.10 the detailed acquisition conditions for the used datasets are listed.

Table 3.10: Single particle acquisition parameters.

Dataset	Illumination area	Pixel size	Electron Dose	Exposure time
May 2018	150 µm	1.07 Å	$39\,\frac{e^-}{\text{Å}^2}$	3 s

3.3.14 Particle Selection

Not all of the micrographs can be used for classification and helical reconstruction. Good tube micrographs were selected using *e2helixboxer.py* that is part of the software package *EMAN2*[63]. Figure 3.6a shows a screenshot of the helix selection process. Tubes were only selected when they had a linear shape, a homogeneous color inside the tube and no ice crystals or dirt on top of the tube. When a tube was only partially dirty or shaped inhomogeneously it was divided into usable subparticles. Figure 3.6b is an example for a tube usable for helical reconstruction. The ice crystal on the right edge is not touching the inner part of the tube. In contrast, 3.6c shows a non-linear shaped and a dirty inner tube surface. The helical box size was selected to be the same for all particles and wide enough to fit the largest tube found in the dataset.

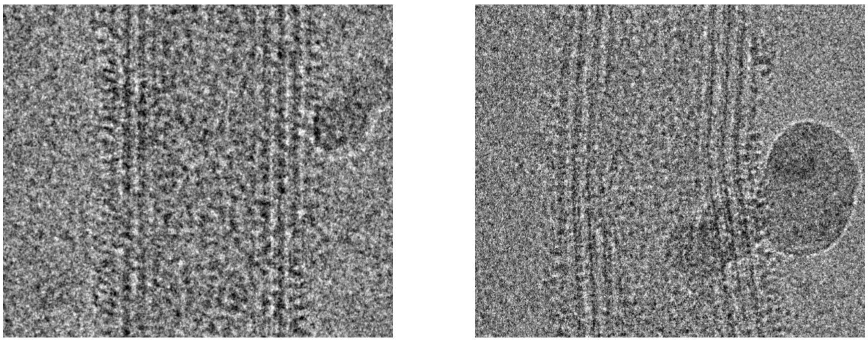


Box Width: 800

Image Quality: 2

	Micrograph	Boxed Helices
53	img_0000152	0
54	img_0000153	0
55	img_0000154	0
56	img_0000155	1
57	img_0000156	0
58	img_0000157	0
59	img_0000158	0
60	img_0000159	0
61	img_0000160	0
62	img_0000161	0
63	img_0000162	0
64	img_0000163	0
65	img_0000164	0
66	img_0000165	0
67	img_0000166	0
68	img_0000167	0
69	img_0000168	0
70	img_0000169	0
71	img_0000170	0
72	img_0000171	0
73	img_0000172	0
74	img_0000173	0
75	img_0000174	0
76	img_0000175	0
77	img_0000176	0
78	img_0000177	0
79	img_0000178	0
80	img_0000179	0
81	img_0000180	0
82	img_0000181	0

(a) GUI of *e2helixboxer.py*.



(b) Good tube. **(c)** Bad tube.

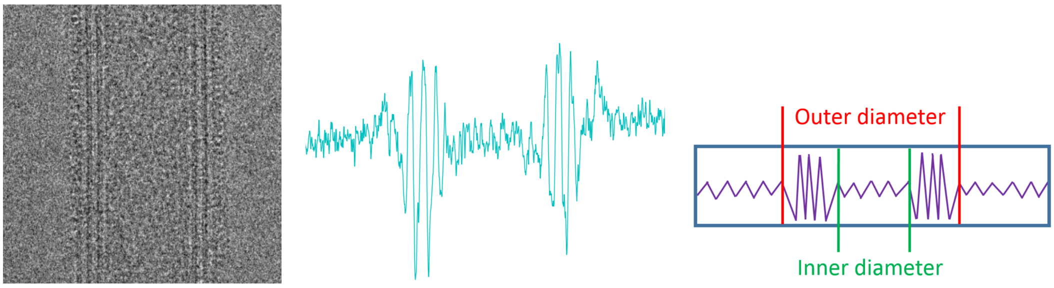
Figure 3.6: Tubes were selected in the GUI **(a)** of *e2helixboxer.py*. While **(b)** shows a usable tube **(c)** was discarded.

Fragments of micrographs saved by *e2helixboxer.py* were used for further tube analysis and helical reconstruction.

3.3.15 Automated Tube Analysis

For an automatic lipid tube analysis a script in Python named LineAdd.py was written for this work (Appendix C). It accesses libraries from *SPARX* [64] and generates 1D density graphs of micrographs of the selected tube particles from chapter

3.3.14. It approximates the position of the ends of each tube and calculates the inner and outer tube diameters. After the tube measurement, the script classifies the micrographs by their inner tube diameter and moves them to corresponding class folders for helical reconstruction. Figure 3.7a shows a tube micrograph and 3.7b its corresponding density graph. In figure 3.7c the inner (*green*) and outer (*red*) diameters are highlighted in an illustration of the density graph.



(a) Tube micrograph. **(b)** 1D density graph. **(c)** Analysis of the graph.

Figure 3.7: A tube micrograph **(a)** is averaged to a 1D signal **(b)** and the tube ends detected in the graph. **(c)** shows a graph illustration with identified inner and outer diameters.

In the following, the functionality of the tube detection algorithm is shown in detail.

Density Graph Calculation

The first task of the script is to calculate a 1D density signal by averaging all horizontal lines of the micrograph matrix $M[x, y]$ containing $m \times n$ pixels. The density signal is a single line $l[x]$ containing m pixels as used in [65]:

$$f \colon \mathbb{R}^{m\times n} \rightarrow \mathbb{R}^{m}$$

$$l[x] = \frac{1}{n} \sum_{y=0}^{n} M[x, y] \tag{3.1}$$

All functions are discrete and thus it has to be valid that $x, y \in \mathbb{N}$. Figure 3.8 shows a visualization of the density graph calculation by line averaging of the micrograph:

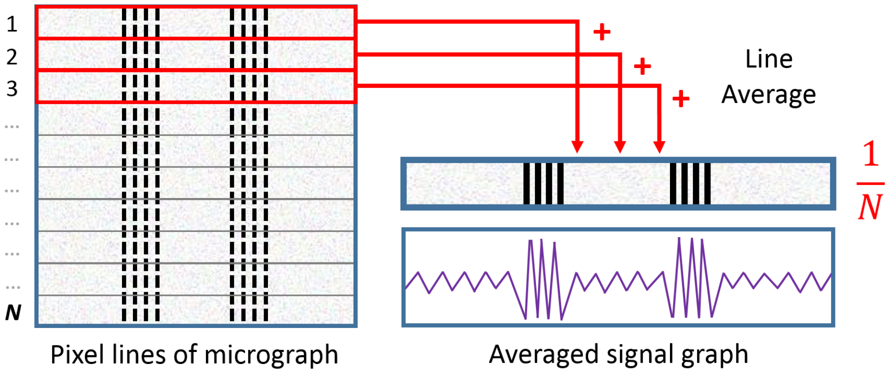
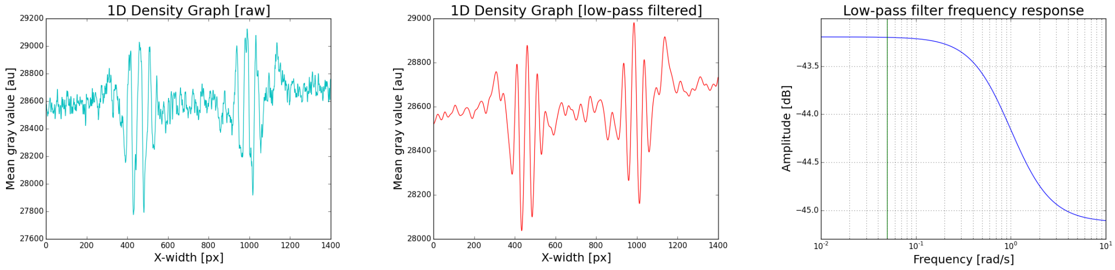


Figure 3.8: LINEADD.PY averages all the pixel lines of the micrograph and plots the 1D density graph.

Low-Pass Filter

The tube end detection algorithm analyzes the local contrast between two pixels. Noise can disrupt the identification of a tube as it contains high frequencies that increase the local contrasts in the signal and washes out the contrast between the maxima of the tube edges and the background signal. High frequency signal noise can be filtered using a low-pass filter. It cuts off all information of the signal at frequencies higher than the cut-off at ω_c. Frequencies lower than the cut-off pass the filter unaffected. Figure 3.9b shows the low-pass filtered raw signal shown in 3.9a from figure 3.8. Figure 3.9c shows the low-pass filter function with a *green* line showing the cut-off frequency.



(a) Raw averaged signal $l(x)$.

(b) Filtered signal $f(x)$.

(c) Bode diagram of the low-pass filter gain of $H(s)$.

Figure 3.9: Illustration of the low-pass filter processing in LINEADD.PY. The raw signal $l[x]$ **(a)** is low-pass filtered to $f[x]$ **(b)** using the filter $H(s)$ in **(c)**.

The filtered signal function $f[x]$ is the result of a discrete convolution between the raw density signal function $l[x]$ and the filter impulse response kernel $h[x]$, where $f, h, l \in \mathbb{R}$ and $x \in \mathbb{N}$:

$$f[x] = \big(h * l\big)[x] = \sum_{k=-m}^{m} h[k]l[x-k] \tag{3.2}$$

For LINEADD.PY, a 2$^{\text{nd}}$ order *Butterworth* low-pass filter[66] was used. Signal filters are described in the frequency domain which is linked to the time domain via Laplace transform $\mathcal{L}$. In the frequency domain $s = i\omega$ denotes the complex frequency ($s \in \mathbb{C}$). Moreover, the Laplace transform allows to simplify the math of the convolution to a multiplication[67].

$$F(s) = H(s) \cdot L(s) = \mathcal{L}\big\{(h * l)[x]\big\} \tag{3.3}$$

$F(s)$, $H(s)$ and $L(s)$ are the transformed counterparts of $f[x]$, $h[x]$ and $l[x]$. A Butterworth filter can be described by the gain at the zero frequency $G_0 = G(\omega = 0)$ and the cut-off frequency ω_c:

$$H(s) = \frac{G_0}{B_n(a)}, \ a = \frac{s}{\omega_c} \tag{3.4}$$

$B_n(a)$ is linked to the normalized Butterworth polynomial $B_n(s)$ in equation 3.5. In case of $\mathbf{2^{nd}}$ order filters the Butterworth polynomial for an **even** order number is used:

$$B_n(s) = \prod_{k=1}^{\frac{n}{2}} \left[s^2 - 2s\cos\left(\frac{2k+n-1}{2n}\pi\right) + 1 \right] \tag{3.5}$$

After applying the filter on the density function $L(s)$ in the frequency domain, the low-pass filtered signal $F(s)$ is inverse Laplace-transformed:

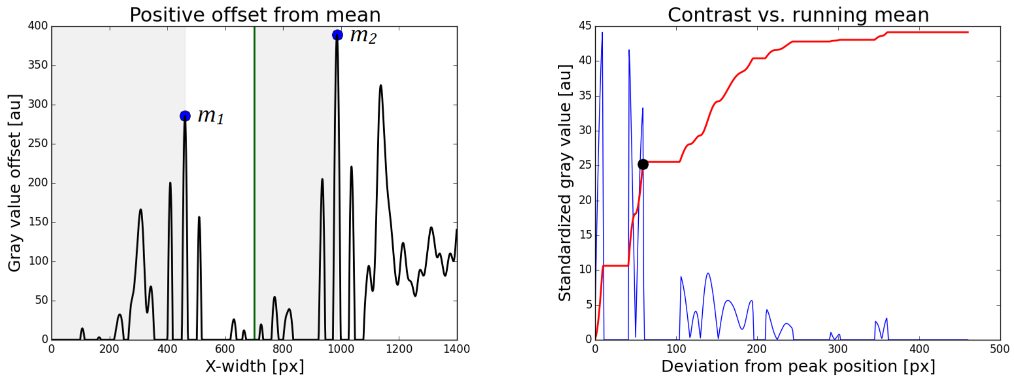
$$f(x) = \mathcal{L}^{-1}\big\{F(s)\big\} \tag{3.6}$$

For this work the Butterworth filter from *SciPy* package[68] with a cut-off frequency of $\omega_c = 0.5\,\frac{\text{rad}}{s}$ was used.

Tube End Approximation

Behind the low-pass filter only values higher than the mean $\bar{f}$ of the filtered signal $f[x]$ are considered. Crooked cut tube particle micrographs can contain black areas leading to global minima in the density graph. These can disrupt the tube approximation algorithm. $f_+[x]$ is the sliced positive offset of the signal from its mean:

$$f_+[x] = \begin{cases} f[x] - \bar{f} & f[x] - \bar{f} > 0 \\ 0 & f[x] - \bar{f} \leq 0. \end{cases}$$



(a) $f_+[x]$: Offset from mean value. **(b)** Graph showing the detection method.

Figure 3.10: (a) is the positive offset from the mean of the filtered function. The *blue* dots are at the maxima m_1 and m_2 and the *green* colored line is the center of the graph. In **(b)** the *blue* graph is the variance function $c[x]$ and the *red* graph is its normed running mean function $r_s[x]$. The highest value for x where both graphs meet is marked with the *black* dot.

$f_+[x]$ is graphed in figure 3.10a. The rough locations of the tube ends can be found at the two local maxima m_1 and m_2 (*blue* dots) left and right to the center of the graph. The graph is divided into four intervals (alternating *gray* and *white* background) separated by the two maxima and the center of the graph (*green* colored line). The script runs the same algorithm for each interval separately: From the maxima towards the outer edges to find the outer ends and from the maxima towards the center to find the inner ends. The algorithm in the following is described based on the left outer end. The first step is to calculate the local contrast variance $c[x]$ between each two pixels starting from the left maximum towards the left edge of the density graph.

$$c[x] = \left[\sqrt{(f_+[m_1 - k] - f_+[m_1 - k - 1])^2}\right]_{k=0}^{m_1} \tag{3.7}$$

In the second step the running mean function $r[x]$ of the local contrast variance $c[x]$ is calculated:

$$r[x] = \frac{1}{x}\sum_{k=0}^{x} c[m_1 - k] + c[m_1 - k - 1] \tag{3.8}$$

When comparing both functions it has to be considered that $\bar{r} \gg \bar{c}$. $r[x]$ needs to be normed to the values of $c[x]$. The normed function is denoted by $r_s[x]$:

$$r_s[x] = \frac{\max c[x]}{\max r[x]} r[x] \tag{3.9}$$

The last step is to find the values for x where the graphs of $r_s[x]$ and $c[x]$ meet. There can be multiple positions but only the last common point between the graphs is an accurate approximation for the tube end position. The found position is not an absolute location. It is an offset from the maximum m_1. Thus, the most accurate approximation for the left outer end position o_1 can be calculated by:

$$o_1 = m_1 - \max_x \Big\{ x \in \mathbb{N}, c[x] - r_s[x] = 0 \Big\} \tag{3.10}$$

The inner end positions are i_1 and i_2, respectively. In figure 3.10b the functions $c[x]$ (*blue*) and $r_s[x]$ (*red*) are graphed. The *black* dot marks the position of the tube end o_1. After applying the algorithm to all four intervals in 3.10a the resulting graph is displayed in figure 3.11. The *red* dots mark the outer ends of the tube and the *green* dots mark the inner ends of the tube.

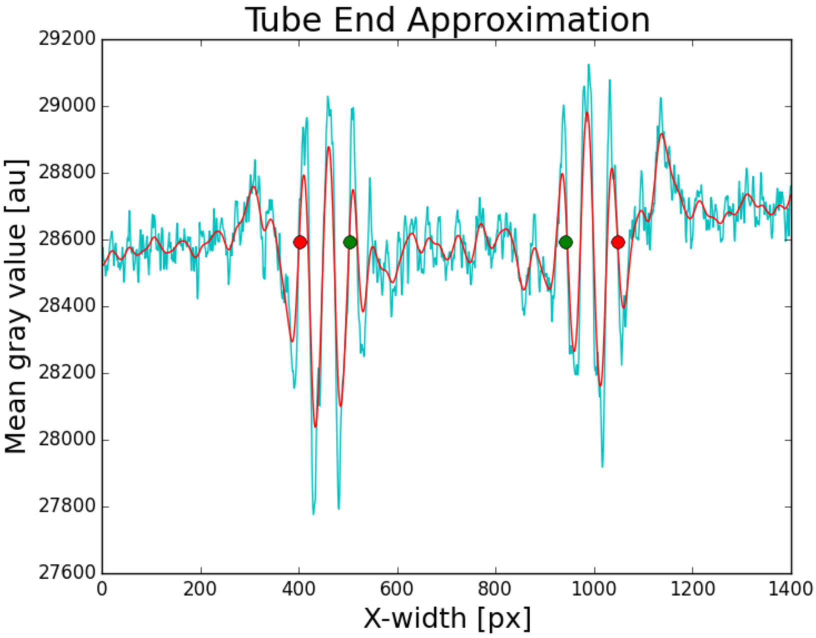


Figure 3.11: Tube ends detected in the 1D density graph (*cyan*: raw, red: low-pass filtered): *Red* dots are the approximated outer ends of the tube and *green* dots are the approximated inner ends of the tube.

The calculation of tube ends enables the determination of the outer diameter d_{out} and the inner diameter d_{in}:

$$d_{\text{out}} = o_2 - o_1, \;\; d_{\text{in}} = i_2 - i_1 \tag{3.11}$$

The thickness of a single tube border d_{border} can be calculated from the difference of outer and inner diameter:

$$d_{\text{border}} = \frac{d_{\text{out}} - d_{\text{in}}}{2} \tag{3.12}$$

The algorithm calculates only approximations of the tube end positions and these can deviate from the exact tube end positions. Though, a small offset is still accurate enough to classify the tubes by their diameter. Nevertheless, there are micrographs with completely wrong approximated tube ends. For this case a failure detection is needed.

Approximation Failure

LineAdd.py tries to determine if the approximation of the tube end is inaccurate. Failure checks are performed for each inner and outer tube end separately. In case of the first outer tube end, the most trivial criterion for an inaccurate approximation is when its location and the location of the first maximum are equal ($o_1 = m_1$). In this case the algorithm did not detect any difference to the maximum at all. The extended criterion is defined over the distance ratio around the tube end limited by the left edge of the micrograph and the maximum m_1:

$$o_1 < \frac{1}{2}(m_1 - o_1) \tag{3.13}$$

By default, the script excludes tubes with poor approximated inner diameters. It can also be configured to exclude inner diameters from the statistics if the corresponding outer diameter approximation failed.

Tube Classification

The automatic determination of diameters allows a classification of tubes by their properties. LineAdd.py sorts the tubes by inner diameter and calculates their minimum d_{min} and maximum d_{max}. After specifying the pixel size and diameter classes d_k with the width Δd in units of [Å] all tubes are sorted into the fitting interval limited by $[a_k, b_k]$:

$$[a,b] := \Big\{(d_1,\ldots,d_n) \in \mathbb{N}^n \mid a_1 \leq d_1 \leq b_1,\ldots,a_n \leq d_n \leq b_n\Big\}$$

$$a_1 := d_{\text{min}},\, b_n := a_1 + n \cdot \Delta d,\, n = \frac{d_{\text{max}} - d_{\text{min}}}{\Delta d} + 1$$

For each class a directory is created. All tube micrographs are moved to the class directory they belong to. Micrographs with a failed tube end detection are moved to a separate folder. As a plus, the script generates statistics about the tubes and plots diagrams with the mean diameters and the standard deviations. The classification is plotted with an automatic Gaussian fit. Detailed statistics are shown in chapter 4.5.

3.3.16 Helical Reconstruction

For IHRSR *RELION* was used[69][70]. It was originally developed for single particle analysis, but helical reconstruction support was added in 2017[47]. The .MRC stacks of the dose-filtered micrographs (see chapter 3.3.13) corresponding to the picked particles (see chapter 3.3.14) were imported in RELION. The coordinates of the picked particles were renamed from .TXT to .BOX and also imported. CTF estimation was performed using *Gctf* [71]. As box size B [px] the largest outer diameter of the particles was used. Particles were extracted with an overlap of 90 % (the following particles start at a repeat of 10 % of the box size). The helical parameters (helical rise R [Å] and the number of asymmetrical units U) for the particle extraction were first *guessed* by the following relation:

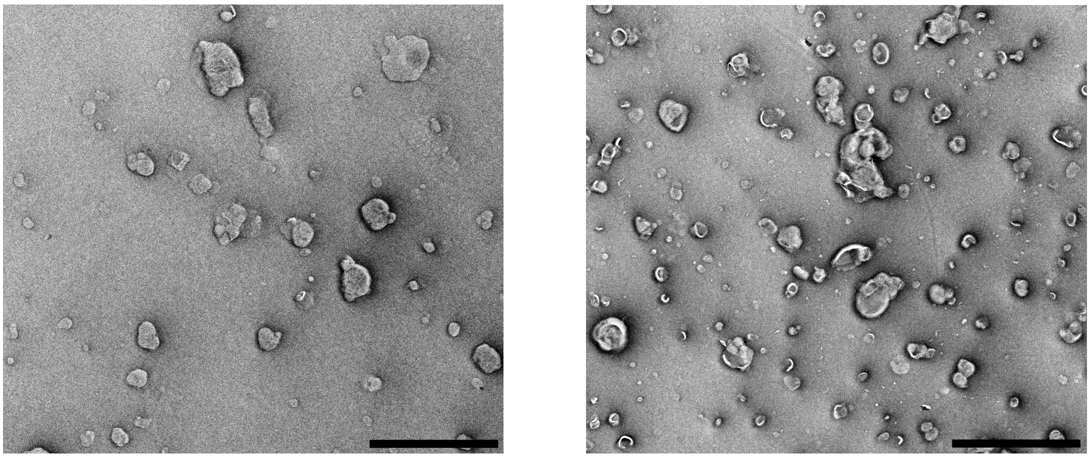
$$R \cdot U \approx \frac{B}{10} \tag{3.14}$$

Chapter 4

Results

4.1 Unilamellar Vesicles

ULVs (**Uni****L**amellar **V**esicles) were prepared using the protocol in chapter 3.3.1.



(a) Folch vesicles, 4 $\frac{mg}{m\ell}$. Count: 38.

(b) Folch vesicles, 8 $\frac{mg}{m\ell}$. Count: 69.

Figure 4.1: Folch unilamellar vesicles prepared using the lipid extruder in Buffer *B*. All the objects with curved shapes in the micrographs are dried vesicles. The doubled lipid concentration in **(b)** resulted in 1.8 × higher vesicle concentration compared to **(a)**. Scale bars: 2 µm.

Folch concentrations of 4 $\frac{mg}{m\ell}$ and 8 $\frac{mg}{m\ell}$ are compared in figure 4.1a and 4.1b in images of negative stained samples. Negative staining includes drying of the sample, that is why the vesicles look crumpled. The doubled concentration resulted in 1.8 times higher vesicle concentration.

4.1.1 ULVs with BPG

As described in chapter 3.3.4 the ULVs were incubated with BPG. BPG extrudes lipid tubes from ULVs. From the one of the vesicles in the *green* boxes a tube (marked in *red*) was extruded in figure 4.2. It features a length of 3.7 µm. Its diameter varies between 49 nm and 77 nm. A grainy coat can be found on the tubes extruded from the vesicle in the *blue* close-up. The resolution of the micrograph is too low to identify the grainy coat as helical protein oligomers.

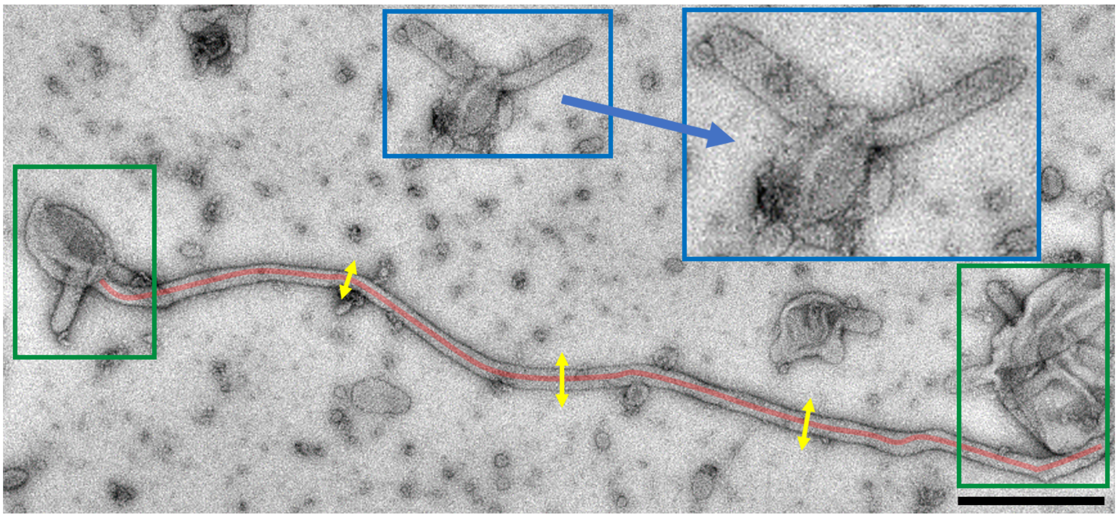
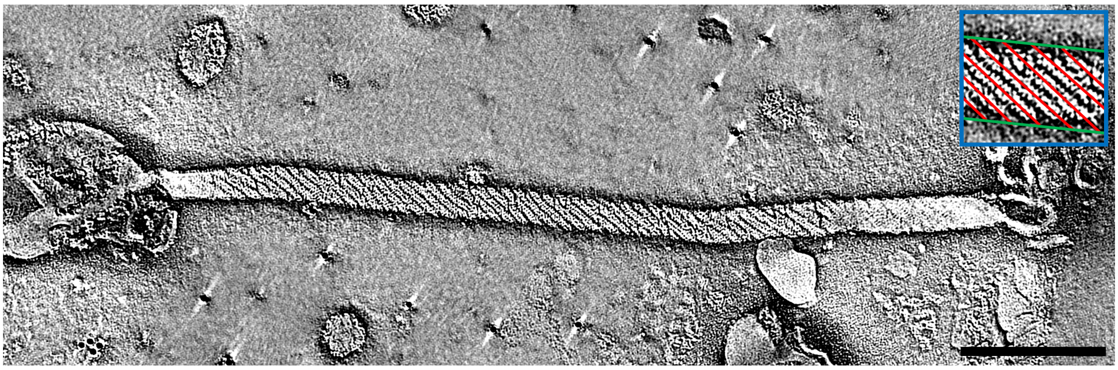


Figure 4.2: From one of the vesicles in the *green* boxes a 3.7 µm long tube (*red*) was extruded. Inside the *blue* close-up box, a circular vesicle with two extruded tubes to the left and to the right can be seen. Notably on these tubes a grainy coat is visible. At the *yellow* markers the diameter was measured: 1. 49 nm, 2. 77 nm, 3. 64 nm. Scale bar: 500 nm.

To increase the resolution, a negative stain and a Cryo tomogram of BPG-extruded tubes were recorded, and images were stacked up to a slice. In the negative stain slice in figure 4.3a the helical protein coat is visible. The stripes have an angle of 36 ° compared to 58 ° in the Cryo image in 4.3b. However, the Cryo slice features far less contrast and it is not possible to see the more than stripes on the tube surface. In the *blue* boxes the stripes are highlighted in *red*. The angle was measured to the *green* tube axis.



(a) Slice of a negative stain tomogram (50 images stacked). The lipid tube was extruded by BPG from the large vesicle on the left. On the surface of the tube a helical BPG coat is visible. It is emphasized in the *blue* box with *red* highlighted stripes. They are tilted 36 ° to the *green* tube axis. The tube has a length of 1.2 µm and a diameter of 53 nm. Scale bar: 200 nm.

(b) Slice of a Cryo tomogram (28 images stacked). The tube has a diameter of 75 nm and a length of 400 nm. The stripes on the tube are tilted 58 ° to the tube axis. The contrast is too weak to see the same details as in **(a)**. Only lines are visible. Scale bar: 100 nm.

Figure 4.3: Slices of tomograms of tubes extruded by BPG from ULVs in negative stain **(a)** and Cryo **(b)**. The high contrast of the negative stain image allows to see *white* dots in a helical formation.

4.1.2 Helical Parameters

Helical parameters $\Delta\varphi$ and Δz were obtained from the slice of the negative stain tomogram in figure 4.3a of a BPG-extruded lipid tube using the following equation:

$$\begin{aligned}\Delta\varphi = \frac{2\pi}{U}\Delta y = \frac{360\,^\circ}{2\pi r}\Delta y = \frac{360\,^\circ}{2\pi \cdot 25\,\mathrm{nm}} \cdot 2.84\,\mathrm{nm} = 6.51\,^\circ \\ \Delta z = 3.56\,\mathrm{nm}\end{aligned} \tag{4.1}$$

The figures 1.9a and 1.9b in chapter 1.5.3 show the helical parameters on the basis of an illustrated tube. The distances in y- and z-direction were measured manually and averaged over five molecules in figure 4.4.

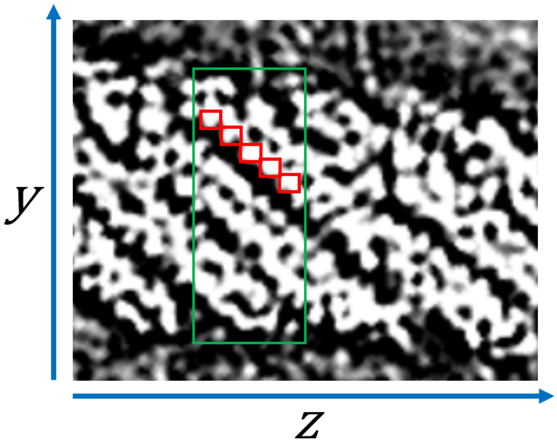


Figure 4.4: Helical parameters were obtained manually from a tomogram of a negative stained tube (section of figure 4.3a). The distances between the molecules in the *red* squares were measured in y- and z-direction and averaged. The *green* rectangle shows the diameter of the tube at the measured location.

4.2 Lipid Nanotubes

When applying the modified ULV protocol from chapter 3.3.2 linear shaped LNTs (**L**ipid **N**ano **T**ubes) are formed as displayed in figure 4.5. They have a length of 0.9 ± 0.29 µm. The shortest length of 50 measured tubes was 0.5 µm and the longest 2.0 µm. The diameter is homogeneous over the tubes and was measured at 96.0 ± 19.1 nm. Dimensions of LNTs were measured using the method described in chapter 3.3.11.

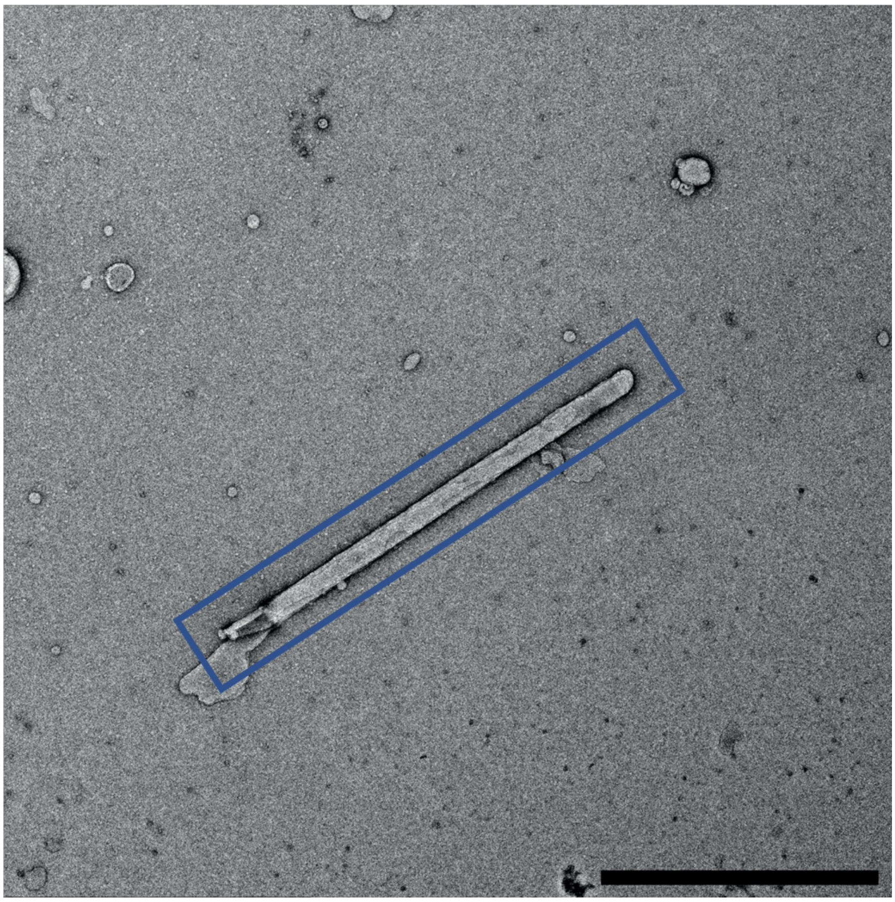


Figure 4.5: A single Folch Lipid Nanotube (in the *blue* box) with a length of 1.6 µm and a diameter of 90 nm. The box emphasizes the linear shape of the tube. Scale bar: 1 µm.

4.2.1 BPG on LNTs

After incubating LNTs with BPG the tubes retain their linear shape with a constant diameter while they feature the same stripes as the BPG-extruded tube in figure 4.3b. The tube in figure 4.6 shows 40 stacked images of a Cryo tomogram of a BPG-coated LNT. The tube has a diameter of 75 nm and a length of more than 930 nm. The tube in the tomogram is cut off at its ends. The stripes have an angle of 67 ° to the tube axis.

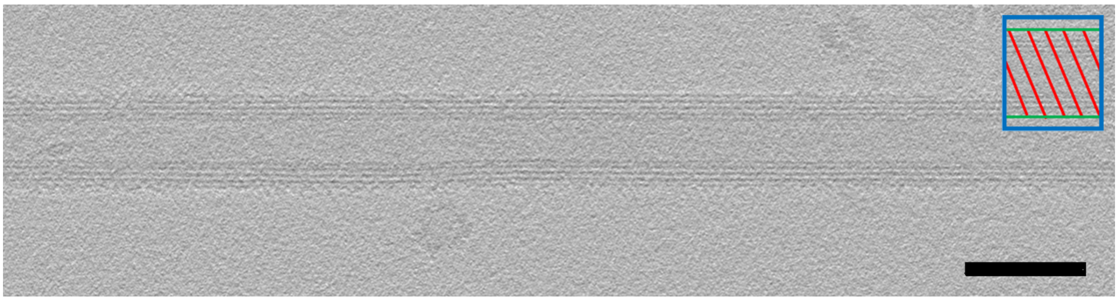


Figure 4.6: Slice of a Cryo tomogram (40 images stacked) of BPG-coated LNT. Scale bar: 100 nm. The tube has a diameter of 75 nm and a length of more than 93 nm. The *red* stripes in the *blue* box have an angle of 67 ° to the *green* tube axis.

For the experiments in this work the diameter is more important than the length of the tubes because the tube shape must meet the curvature sensed by BPG. In table 4.1 the diameter d of 50 BPG-coated tubes was measured in Cryo and compared to the raw LNTs. $\overline{d}$ is the mean diameter and σ_d is the standard deviation. $d_{\min}$ and $d_{\max}$ are the minimum and maximum of measured tube diameters.

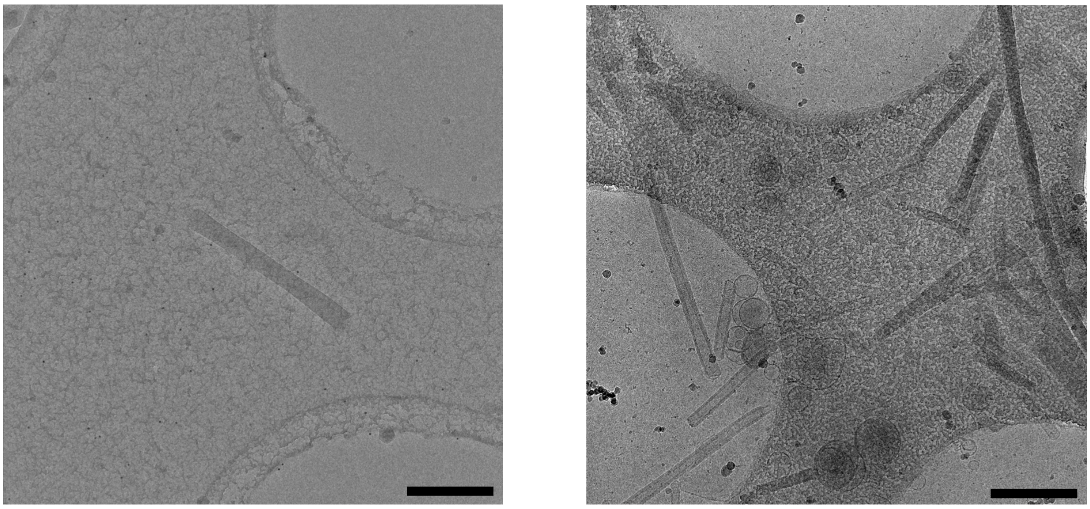
Table 4.1: Diameter measurements of raw and BPG-coated LNTs.

Type	$\overline{\mathbf{d}}$ **[nm]**	$\sigma_{\mathbf{d}}$ **[nm]**	$\mathbf{d}_{\mathbf{min}}$ **[nm]**	$\mathbf{d}_{\mathbf{max}}$ **[nm]**
raw	96.08	19.10	71.31	149.10
BPG-coated	88.48	15.59	57.60	125.52

The tubes coated with BPG oligomers feature a 12 % smaller diameter than the original raw lipid tubes.

4.2.2 Concentration

The sample concentration used to prepare LNTs in Cryo-EM was insufficient for single particle acquisition because the tube density was too low. In figure 4.7a the micrograph of a sample, prepared with the original LNT protocol, is shown. For helical reconstruction, several hundreds of tubes are needed. Therefore, it is desirable to have at least one particle per hole. To reach this, each preparation step was optimized to increase the particle density on the grid. 4.7b shows a sample of raw LNTs prepared with the original conditions and the final conditions after all optimization steps. Counting the tubes in both micrographs, as described in chapter 3.3.10, results in a 16 times higher tube concentration after the optimization process.



(a) Before optimization: 1 tube. **(b) After** optimization: 16 tubes.

Figure 4.7: Comparison of the concentrations in Cryo with 12 $\frac{mg}{m\ell}$ in the same area. **(a)** is the original LNT protocol and **(b)** is the protocol including all optimizations. The concentration was 16 × higher in **(b)** after the optimization. Scale bars: 500 nm.

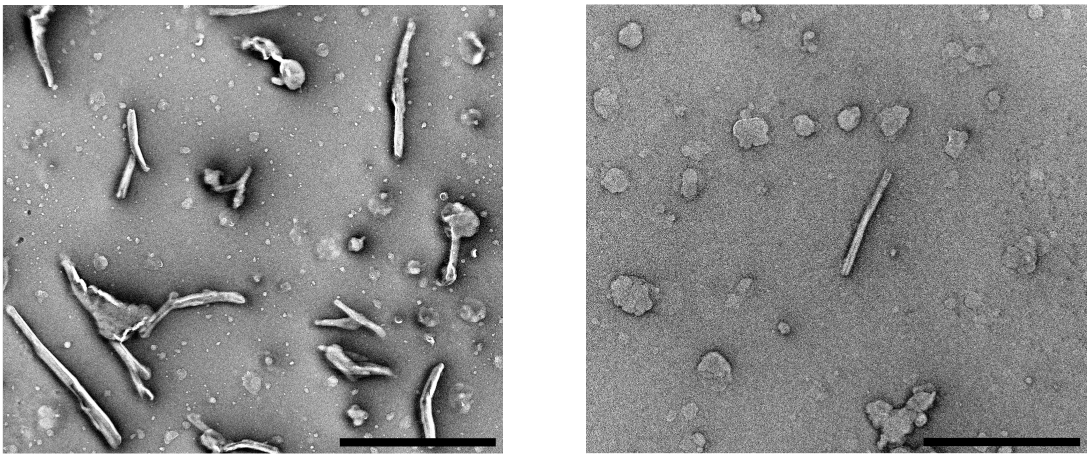
4.3 Optimization

As shown in chapter 4.2.2 the concentration of the LNTs was not enough for Cryo-EM. All previous LNTs were prepared at a Folch concentration of $4\,\frac{mg}{m\ell}$. The most trivial way to increase the concentration is to dissolve more lipids in the same buffer volume. A new experiment with a Folch concentration of $12\,\frac{mg}{m\ell}$ was performed and compared to the standard concentration. Moreover, the following parameters of the protocol were optimized:

- The buffer components were optimized by determination of ions responsible for the tube extrusion.
- A Central Composite Design was performed to find relations between the number of freeze-thaw-cycles, the number of extrusion cycles and the extrusion temperature.
- The stability of the LNTs was reviewed over the storage time to determine the optimal point in time to use them for experiments.

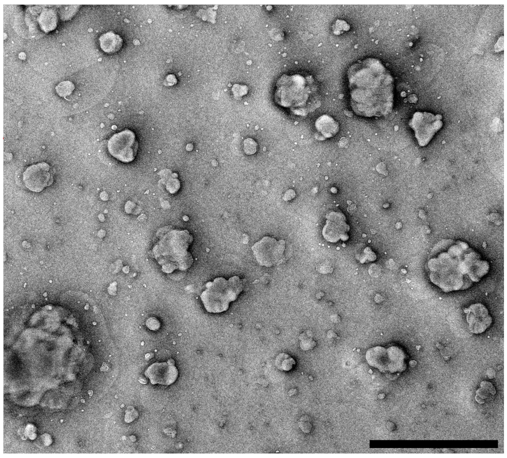
4.3.1 Dependency on Buffer Components

The original buffer from unilamellar vesicle preparation protocol (chapter 3.3.1) B included 1 mM EDTA. As shown in figure 4.8a a new buffer solution $B2$ without EDTA yields a seven times higher tube concentration compared to the original buffer in 4.8b but also in 1.5 times more vesicles. The sample in figure 4.8c contains 1 mM EGTA instead of EDTA. This results in no LNTs but three times more vesicles compared to the sample without any chelating agents. The size of the vesicles also increased visually with their count. Thus, a measurement was performed: While the mean diameter of the vesicles remained constant at 229 ± 24 nm the standard deviation was increased from 51 nm up to 173 nm with more vesicles. This shows that the size distribution was extended and there are more of larger and more of smaller vesicles in combination with less tubes.



(a) LNTs without EDTA (buffer $B2$): 7 tubes and 19 vesicles. Diameter of vesicles: $229 \pm 51\,\text{nm}$.

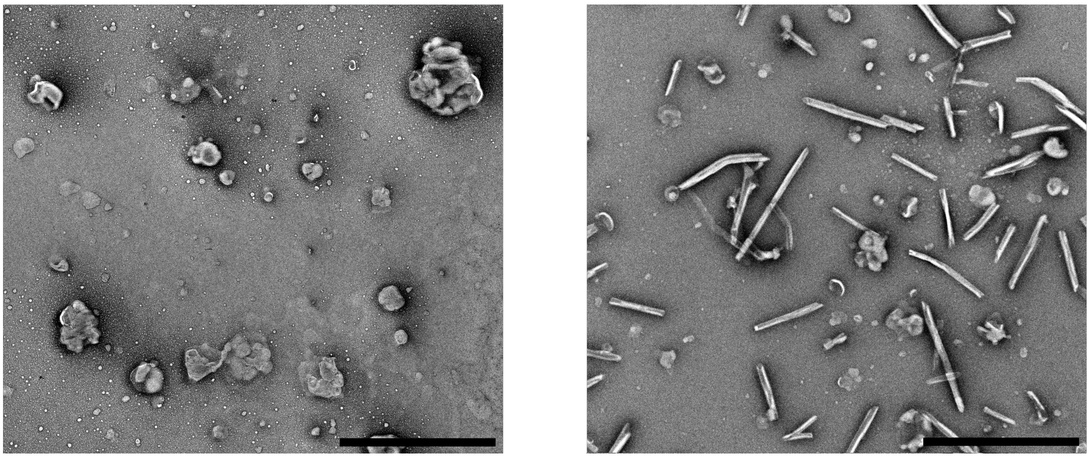
(b) LNTs with 1 mM EDTA (buffer B): 1 tube and 29 vesicles. Diameter of vesicles: $253 \pm 91\,\text{nm}$.



(c) LNTs with 1 mM EGTA (buffer B^{EGTA}): 0 tubes and 59 vesicles. Diameter of vesicles: $205 \pm 173\,\text{nm}$.

Figure 4.8: Comparison between buffers without chelating agents and buffers containing 1 mM EDTA and 1 mM EGTA. Omitting EDTA in **(a)** resulted in 7 × more tubes and 1.5 × more vesicles compared to the original buffer in **(b)**. The sample in **(c)** containing EGTA yielded no tubes but 3 × more vesicles. The mean diameter stayed constant at $229 \pm 24\,\text{nm}$ but the standard deviation was increased from 51 nm to 173 nm with more vesicles. Scale bars: 2 µm.

These experiments show that both chelating agents yield less tubes but higher amounts of vesicles. Besides a direct dependency, the concentration of the chelating agents can possibly show a dependency on Mg^{2+} and Ca^{2+} ions because the chelating agents absorb these ions. Thus, another experiment was performed: The buffer $B3$ contained 1 mM of EDTA and 1 mM of EGTA and in one case 5 mM $MgCl_2$ and in the other case $CaCl_2$ to see the effect of each type of metal ions. The results are shown in figure 4.9a and 4.9b:



(a) LNTs with 5 mM $MgCl_2$ (buffer $B3^{MgCl_2}$): no tubes and 19 vesicles.

(b) LNTs with 5 mM $CaCl_2$ (buffer $B3^{CaCl_2}$): 24 tubes and 14 vesicles.

Figure 4.9: Comparison between LNTs with buffers containing $MgCl_2$ and $CaCl_2$. **(a)** $MgCl_2$ in the sample results in no tubes but 19 vesicles. **(b)** In contrast, the sample with $CaCl_2$ contains only 14 vesicles but 24 LNTs. This shows that $CaCl_2$ favors tube extrusion. Scale bars: 2 µm.

The experiment with 5 mM $MgCl_2$ did not contain any tubes. It yielded 19 vesicles. In contrast, $CaCl_2$ yielded a high tube concentration (24 LNTs) and less vesicle concentration (14 vesicles).

Finally, the highest tube concentration was achieved by adding both chelating agents in the same concentration of 1 mM and 5 mM $CaCl_2$.

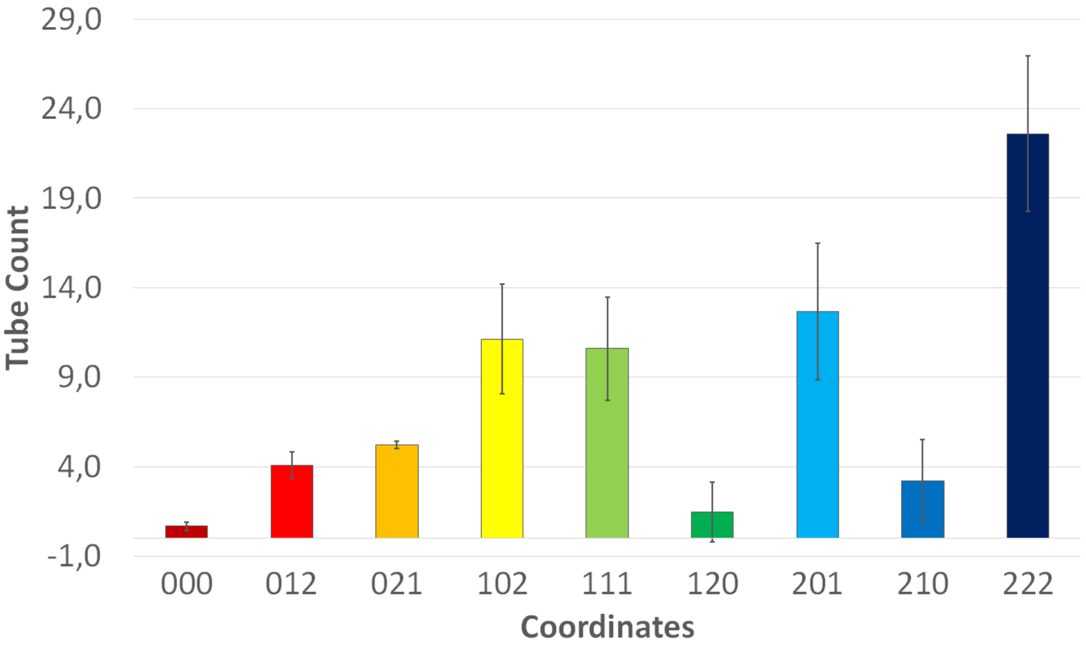
4.3.2 Central Composite Design

The formation of LNTs was characterized by the number of freeze-thaw-cycles n_{FT}, the extrusion temperature T_{Ex} and the number of extrusion cycles n_{Ex}. The lipid concentration c_{Lipid} was kept constant at 12 $\frac{mg}{m\ell}$ (see chapter 3.3.2 for details). To find the experimental correlation between them a CCD (**C**entral **C**omposite **D**esign) consisting of nine experiments in total (described in chapter 1.6) was performed. For each experiment a negative stain test was performed and the tubes on the EM grid were counted (chapter 3.3.10). The complete CCD was performed with $N = 3$ and the averaged results are shown in table 4.2. The full results including the tube count for every single image can be found in appendix A in table A.1. For a better imagination of the CCD method every experiment was labeled with a specific color and its coordinates in the spatial diagram.

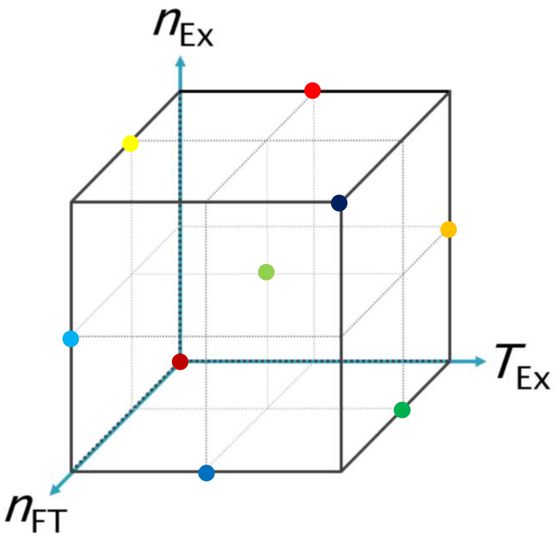
Table 4.2: Results for LNT Central Composite Design.

Coords.	**CCD Color**	Freeze-thaw-cycl. $\mathbf{n}_{\mathrm{FT}}$	Extrusion Temp. $\mathbf{T}_{\mathrm{Ex}}$	Extrusion cycl. $\mathbf{n}_{\mathrm{Ex}}$	Tube count $\mathbf{N}_{\mathrm{Tubes}}$	Standard deviation $\boldsymbol{\sigma}_{\mathbf{Tubes}}$
000	●	0 ×	5 °C	0 ×	0.7	0.2
012	●	0 ×	25 °C	61 ×	4.1	0.8
021	●	0 ×	45 °C	21 ×	5.2	0.2
102	●	5 ×	5 °C	61 ×	11.1	3.1
111	●	5 ×	25 °C	21 ×	10.6	2.9
120	●	5 ×	45 °C	0 ×	1.5	1.7
201	●	10 ×	5 °C	21 ×	12.7	3.8
210	●	10 ×	25 °C	0 ×	3.2	2.3
222	●	10 ×	45 °C	61 ×	22.6	4.3

Figure 4.10a shows bar graphs of the averaged results of the CCD. Figure 4.10b shows the position of the points in the spatial coordinate system.



(a) Bar graph of the averaged tube counts in table 4.2. The tube count at the coordinates 222 is the highest. At this position are the maximum values for all three variables ($n_{\mathrm{FT}} = 10\times, T_{\mathrm{Ex}} = 45\,^{\circ}C, n_{\mathrm{Ex}} = 61\times$).



(b) CCD basis of chapter 1.6 labeled with the result flags of the LNT CCD. Each colored dot is a measured point with three coordinates.

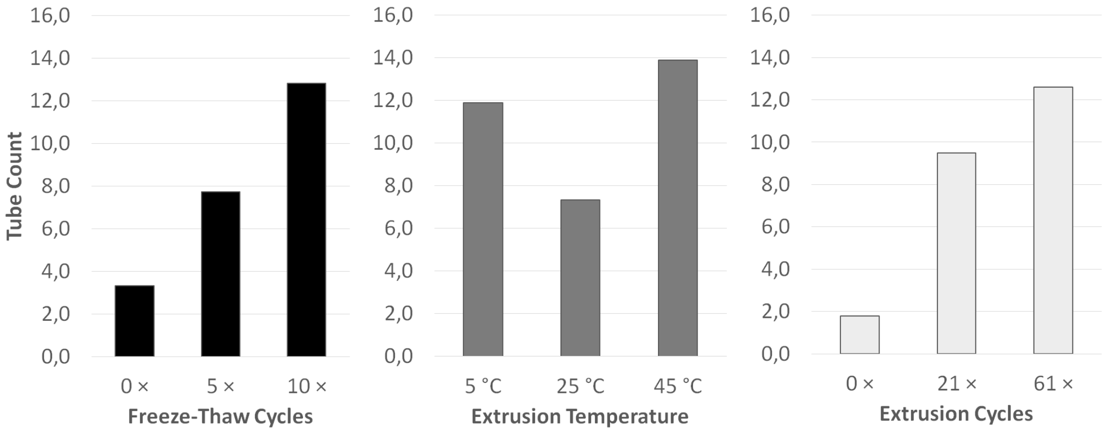
Figure 4.10: Visual illustration of LNT optimization CCD.

In table 4.3 the tube counts of each CCD variable and its tested values are analyzed separately. Each value of each variable appears $\sqrt{9} = 3\times$. The results of these values were averaged and compared to the other values of the belonging variable to see a trend in the tube count. The 1^{st} and 2^{nd} columns contain the variables and each value they can accept. The 3^{rd} column lists all corresponding CCD points that contain the value of the 2^{nd} column. In the 4^{th} column are the assigned diagram colors belonging to the averaged points. For the visualization in figure 4.11 a new color was assigned to the average of each CCD variable. In case of $n_{\text{Ex}} = 0\times$ the extruder was not used and the temperature T_{Ex} is inapplicable. Thus, the averages of the extruder temperature only contain two results in contrast to the other CCD variables.

Table 4.3 shows the averages of the variables for each tested value.

Table 4.3: Correlation between variables and tube count.

CCD Variable	**Value**	**Avg. CCD points**	**Avg. CCD colors**	$\overline{\mathbf{N}}_{\text{Tubes}}$
$\mathbf{n}_{\text{FT}}$ ●	$0\times$	000, 012, 021	● ● ●	3.3
	$5\times$	102, 111, 120	● ● ●	7.7
	$10\times$	201, 210, 222	● ● ●	12.8
$\mathbf{T}_{\text{Ex}}$ ●	5 °C	102, 201	● ●	11.9
	25 °C	012, 111	● ●	7.3
	45 °C	021, 222	● ●	13.9
$\mathbf{n}_{\text{Ex}}$ ●	$0\times$	000, 120, 210	● ● ●	1.8
	$21\times$	021, 111, 201	● ● ●	9.5
	$61\times$	012, 102, 222	● ● ●	12.6



(a) Dependency on $\mathbf{n}_{\mathrm{FT}}$. **(b)** Dependency on $\mathbf{T}_{\mathrm{Ex}}$. **(c)** Dependency on $\mathbf{n}_{\mathrm{Ex}}$.

Figure 4.11: Correlation between variables and tube count. In case of **(a)** n_{FT} and **(c)** n_{Ex} the tube count increases with higher values. Only in case of **(b)** (T_{Ex}) there is a decreased tube count for 25 °C.

In contrast to n_{FT} and n_{Ex} a higher extruder temperature T_{Ex} did not result in an increased tube count. With the data shown at this point it is not possible to show any temperature correlation.

4.3.3 Dependency on Extrusion Temperature

The results of the CCD from chapter 4.3.2 were not enough to determine a correlation between the extruder temperature T_{Ex} and the tube count. For more precise results another experiment was performed. The highest tube count could be observed when all CCD variables were at their highest value. These parameters were adopted and the extruder temperature varied from 5 °C to 45 °C in steps of 10 °C. Each experiment was performed three times. The resulting tube counts can be found in table 4.4. The full data for this experiment can also be found in appendix A in table A.2.

Table 4.4: Detailed dependency on extrusion temperature.

$\mathbf{T}_{\text{Ex}}$	**Coords.**	$\mathbf{N}_{\text{Tubes}}$	$\boldsymbol{\sigma}_{\textbf{Tubes}}$
5 °C	202	16.6	6,0
15 °C	$2\frac{1}{2}2$	16.4	10.6
25 °C	212	24.9	10.5
35 °C	$2\frac{3}{2}2$	15.1	4.0
45 °C	222	17.1	5.0

Although the experiment is not part of the CCD itself the 2nd column includes the CCD coordinates to classify the results in comparison to the CCD. In figure 4.12 the averaged results have been plotted with the related error bars:

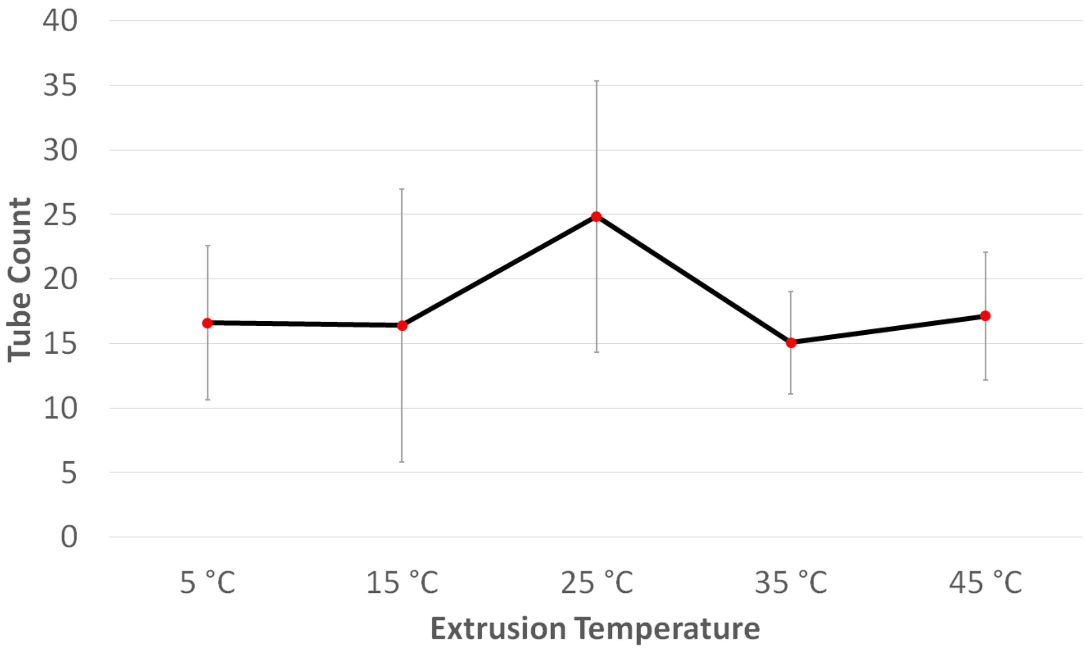
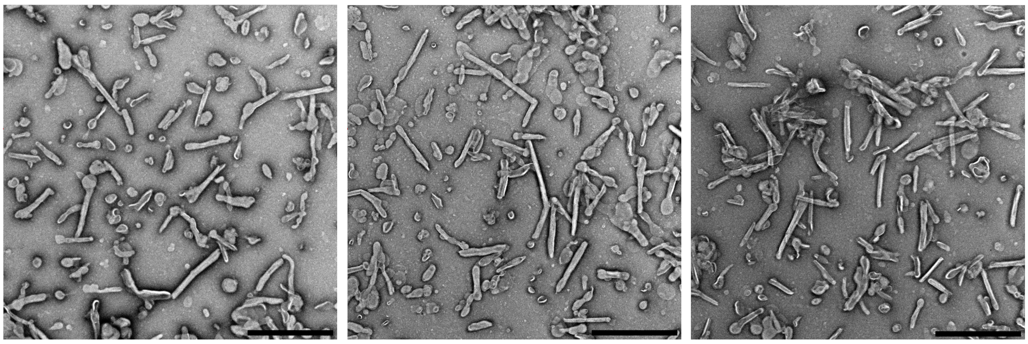


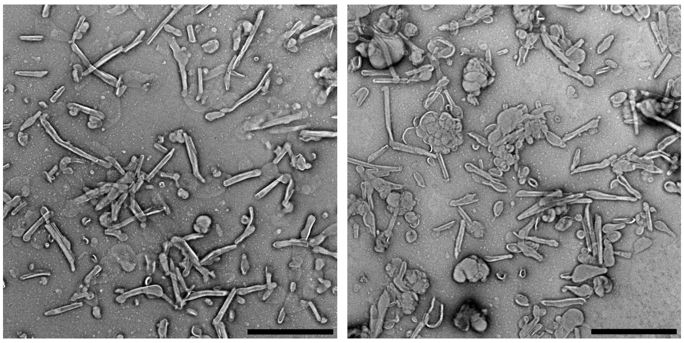
Figure 4.12: Detailed dependency on extrusion temperature: There is a slight increase at 25 °C but the error bars show big uncertainty factors.

4.3.4 Stability of LNTs

To test the stability of lipid nanotubes, the optimized preparation protocol at a lipid concentration of $12\,\frac{mg}{m\ell}$ was used. The first grid was stained 1 h after preparation. The Eppendorf tube was kept at room temperature for the next days without being moved. The following grids were stained on the 3rd, 5th and 9th days. On the last day also the difference between tube solution taken from the top and the bottom of the Eppendorf tube was compared. Micrographs of all grids are shown in figure 4.13. The tubes were counted as described in chapter 3.3.10.



(a) 1st day. 13 : 51. **(b)** 3rd day. 14 : 38. **(c)** 5th day. 18 : 20.



(d) 9th day (top). 20 : 20. **(e)** 9th day (bottom). 9 : 34.

Figure 4.13: (a)-(e): [Tubes : Non-tubular objects]. Stability of LNTs at room temperature over a maximum of eight days. The number of tubes increases (2/3 ×) with time and the number of non-tubular objects decreases to (0.4 ×). The only exception is the sample taken from the bottom at the 9th day. Scale bars: 2 µm.

The sample from the 1st day contains four times more non-tubular objects than tubes. On the 3rd day there were 2.7 times more non-tubular objects than tubes. On the 5th day non-tubular objects and tubes were equally distributed. At the 9th day the solution taken from the top was still equally distributed. The solution taken from the bottom of the Eppendorf tube contained 3.7 times more non-tubular objects than tubes. The tube concentration increased to 1.5 times during the storage time. The comparison between the top and the bottom sample on the 9th day shows that the concentration of usable tubes was two times higher on the top and there were 0.4 times less non-tubular objects. The counts of non-tubular objects and lipid tubes are visualized in figure 4.14.

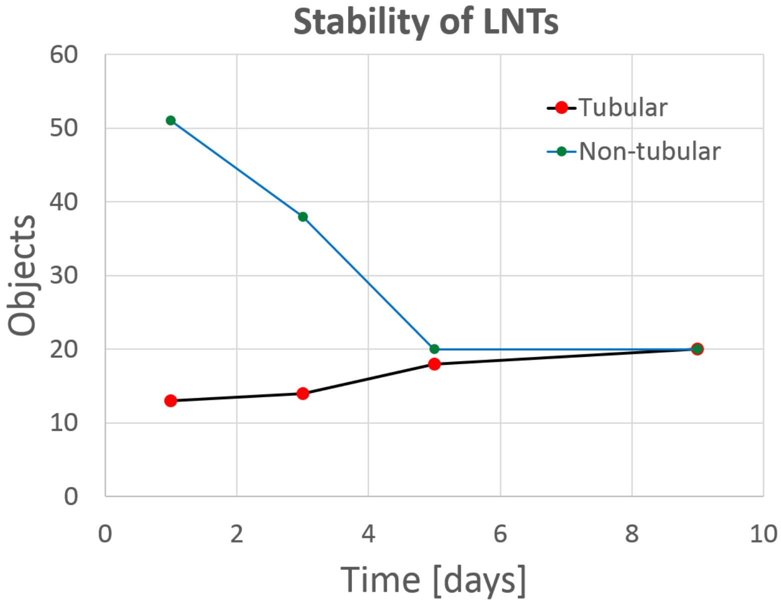
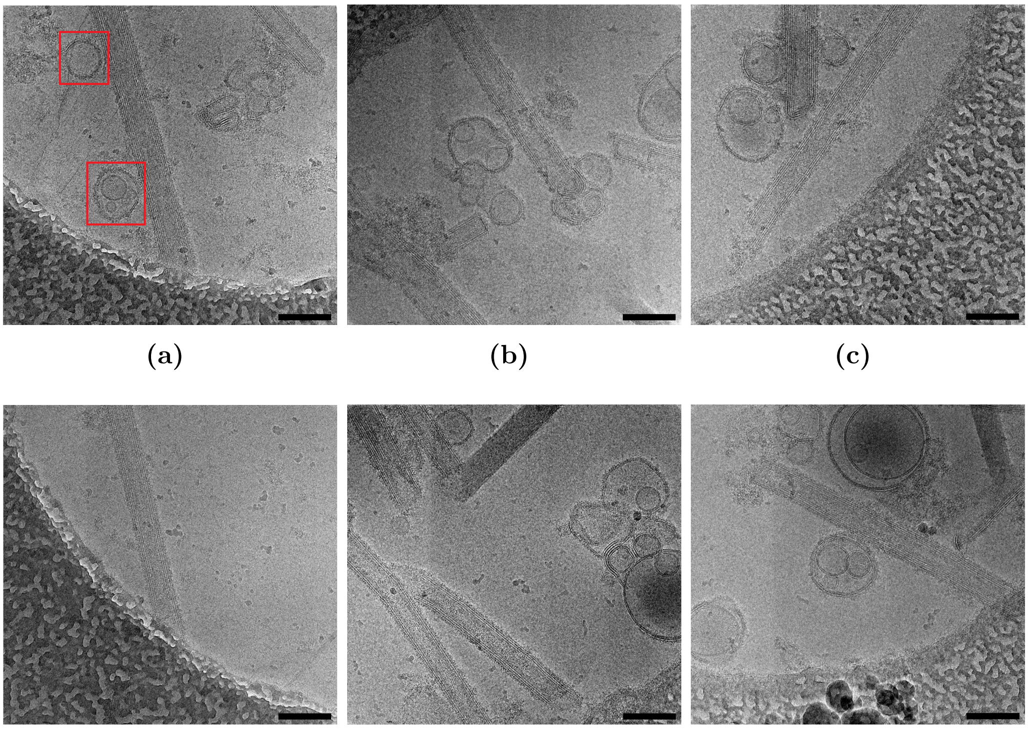


Figure 4.14: Stability graph of LNTs. The number of non-tubular objects decreases with time while the tube count increases. The difference between the 5th day and the 9th day is negligible. The sample taken from the bottom of the Eppendorf tube on the 9th day was not included in the graph.

4.4 Dataset Acquisition

After the optimization of the tube yield for Cryo-EM it was finally possible to acquire datasets for single particle analysis. The micrographs in figure 4.15 were acquired with a pixel size of 1.07 Å per pixel and a dose of 39 $\frac{e^-}{Å^2}$. Besides the tubes, there were also vesicles in almost all cases, see figure (a). The presence of vesicles implies a decreased tube concentration because the lipids did not form tubes in this case. On some of the tubes the helical coat is visible, notably in the close-up view in 4.16b.


(a) (b) (c)

(d) (e) (f)

Figure 4.15: Tubes recorded as part of a dataset. Scale bars: 40 nm. In **(a)** vesicles are marked in *red* squares.

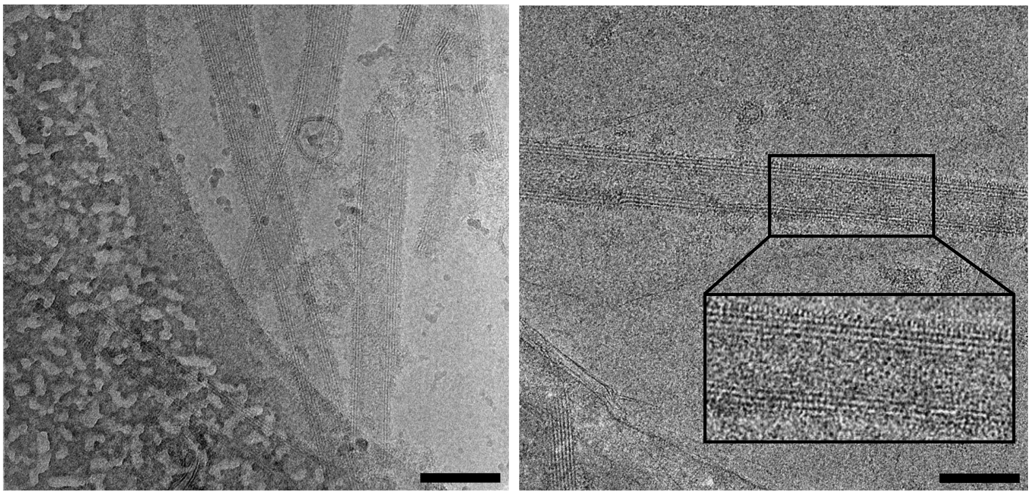
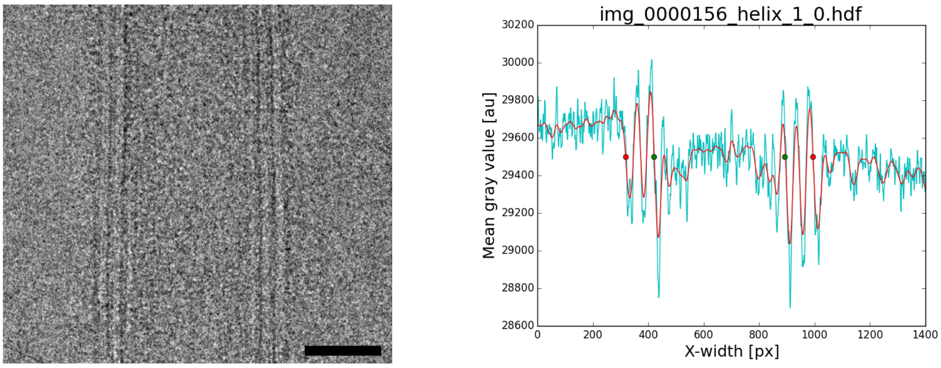

(a) (b)

Figure 4.16: In the *black* marked close-up view in **(b)** the helical coat is visible. Scale bars: 40 nm.

After the acquisition particles for helical reconstruction could be selected from the micrographs as described in chapter 3.3.14.

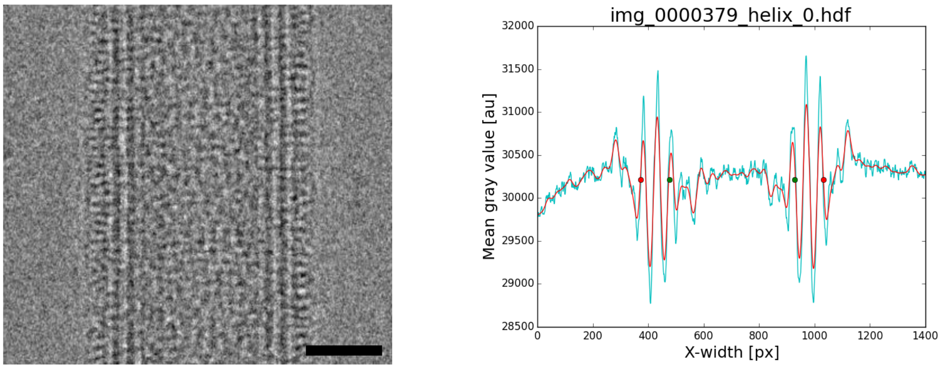
4.5 Automated LNT Analysis

After selecting particles from the acquired micrographs an automatic tube analysis was performed using the script LineAdd.py described in chapter 3.3.15. Figure 4.17c shows two examples for tubes with correctly detected tube ends. While (a) and (c) are raw micrographs, (b) and (d) are the plotted density graphs (*cyan*) including the low-pass filtered ones (*red*). High frequency content is visible in the density graph of figure 4.17a and underlines the difference between the unfiltered graph and the low-pass filtered graph.



(a) Raw micrograph of a helical particle.

(b) The high frequency information is clearly visible and cut off by the low-pass filter.

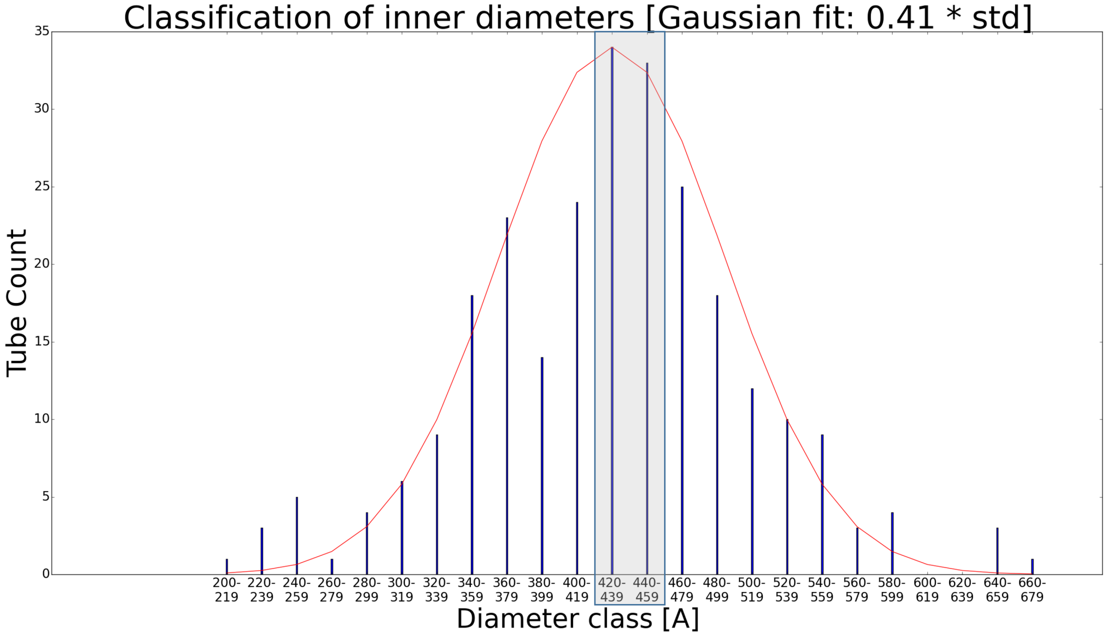


(c) Raw micrograph of helical particle with higher defocus.

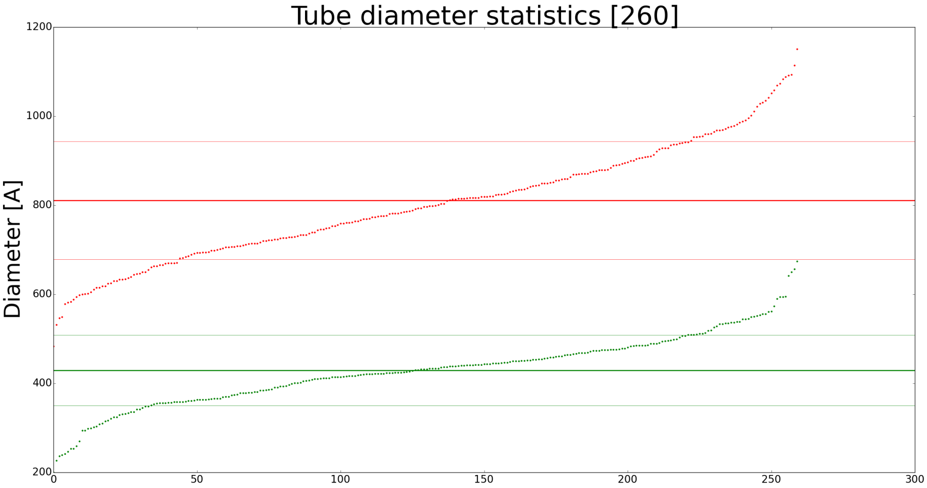
(d) The raw density graph contains less high frequency information compared to **(b)**.

Figure 4.17: Raw micrographs and the output of LineAdd.py. The raw density graphs are *cyan*-colored. In contrast, the (*red*) low-pass filtered graphs are cleaner because they lack high frequency information that can distort the tube end detection algorithm. The difference in defocus between both raw micrographs is visible in the high frequency content of the raw density graphs. Scale bars: 20 nm.

For the tube class histogram the script was configured to consider only inner tube diameters when the corresponding outer ends were detected flawlessly. The class diameter was set to 20 Å. Figure 4.18a shows the histogram of the dataset recorded in May 2018. 4.18b shows a plot of all calculated diameters. The inner (*green*) and outer (*red*) diameters were sorted independently. The centered lines mark the mean values (d_i, d_o) and the thinner lines around them mark the positive and negative standard deviations ($\pm\sigma_i, \pm\sigma_o$).



(a) Tube class histogram in 20 Å steps with a *red* colored best fit Gaussian distribution.



(b) Tube statistics: Each *green* dot represents a measured inner diameter and each *red* dot an outer diameter. The thick lines are the mean diameters and the thin lines are the standard deviations ($\pm\sigma$).

Figure 4.18: Statistics generated for the dataset recorded in May 2018. The *gray* highlighted classes in **(a)** were used for helical reconstruction. The measured inner and outer diameters in **(b)** were sorted separately.

Table 4.5: Detailed Dataset statistics.

Dataset	**Micrographs**	**Excluded**	$\bar{d}_i$	σ_i	$\bar{d}_o$	σ_o
May 2018	280	20	429.3 Å	79.2 Å	811.0 Å	132.7 Å

In figure 4.19 the border thickness of a single border is plotted against the outer diameter of the tube. Overall the border diameter increases with the outer diameter. However, the result is far away from being a linear function. The highlighted interval from 100 to 150 Å was selected for helical reconstruction.

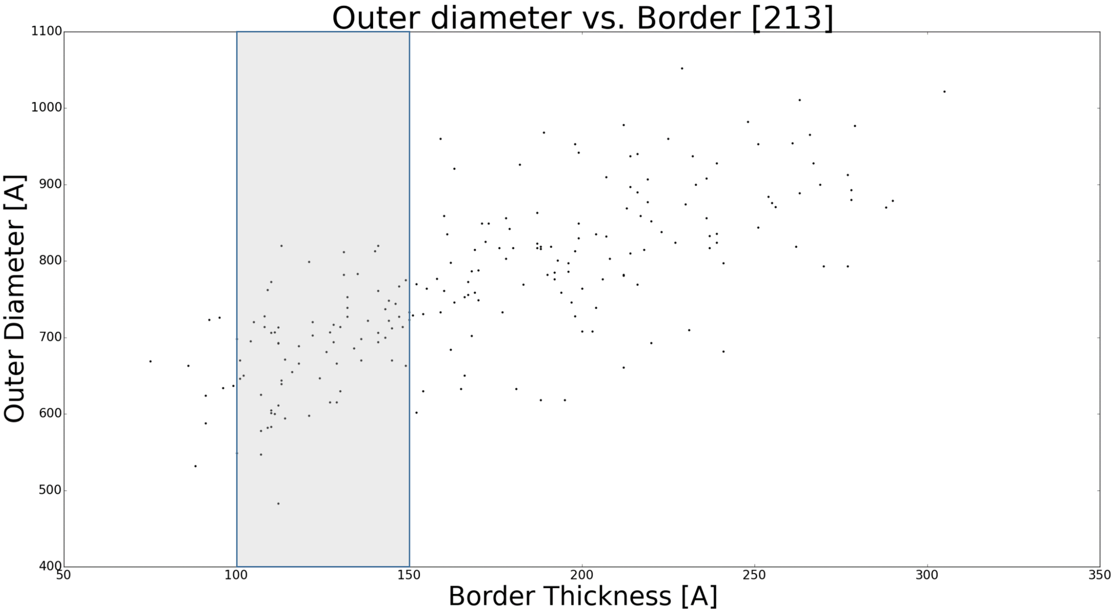


Figure 4.19: Border thickness plotted against the outer tube diameter. This graph visualizes the distribution of the number of multilayers compared to the total diameter of the tube. It helps to find an outer diameter interval with similar multilayer numbers. A dense interval from 100 to 150 Å was selected.

4.6 Helical Reconstruction

For IHRSR in *RELION* particles were picked using *e2display.py* as described in chapter 3.3.14. At first, a 2D classification of the full dataset of May 2018 was performed. In this run, RELION 2D-classified tubes by their number of multilamellar borders and ignored the helical coat between them. Due to this reason three different methods to sort the tubes by diameter were performed to improve the results of the 2D classification.

4.6.1 Diameter Class Selection

In a second run, the tubes were first classified by diameter using LineAdd.py. From its output only the two most occupied diameter classes in figure 4.5 were used: 420 to 439 Å and 440 to 459 Å. This yielded 67 tubes. Three different particle extractions and three different 2D classifications were run. The extraction parameters are listed in table 4.6. In each run 1986 particles were extracted.

Table 4.6: Particle extraction parameters.

	Box size	**Tube diameter**	**Bimodal priors**	**Asymmetrical units**	**Helical rise**
#1	1100 px	410 Å	yes	2	20 Å
#2	1100 px	1000 Å	yes	2	20 Å
#3	1100 px	1000 Å	no	2	20 Å

The tubes inside the selected diameter classes can still vary in 40 Å. At a resolution of 10 Å a count of four different 2D classes can be expected because different diameters are sorted in different classes. To be on the safe side, the number of classes was doubled to eight and rounded up to ten. Each *line* of table 4.6 is linked to a *column* of 2D classification parameters in table 4.7:

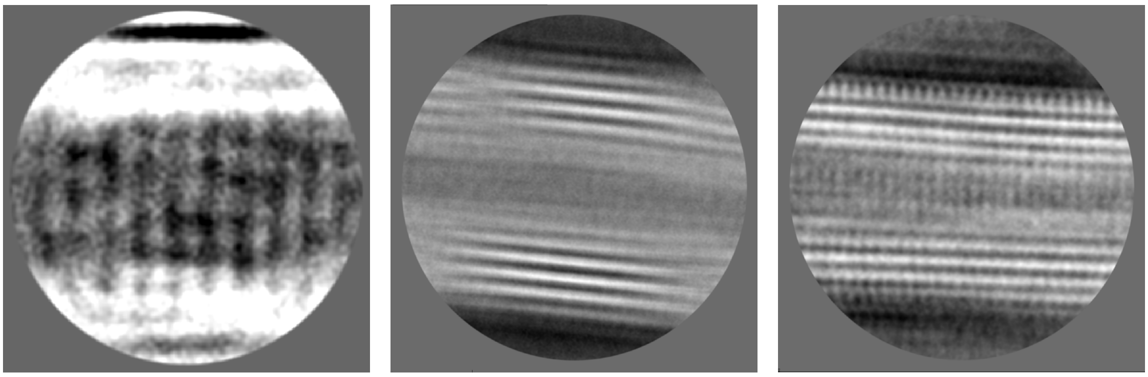
Table 4.7: 2D classification parameters.

	#1	**#2**	**#3**
Number of Classes	8/10	3/10	9/10
Mask diameter [Å]	410	1100	1100
Limit resolution [Å]	-1	5	10
In-plane angular sampling	6	6	1
Offset search range [px]	5	5	10
Offset search step [px]	1	1	5
Tube diameter [Å]	410	1000	1000
Bimodal searches	yes	yes	no
Angular search range [deg]	2	6	6

For the first extraction and 2D classification the parameters of the 1st line and column were used as a test to classify the helical coat on the LNTs. For this purpose, the mask diameter was set to 410 Å to cut off the strong signal of the multilamellar borders. This yielded eight classes and the densest one is shown in figure 4.20a. The vertical stripes of the helical coat are clearly visible. However, for 3D model building the tube borders must be considered as well because they are part of the underlying geometry.

In the set of the 2nd line and column the mask diameter was set to 1100 Å to include the complete tube. This resulted in an overfitted and blurred class because tubes with different border numbers were averaged. In this case seven of ten classes collapsed. The densest one is shown in figure 4.20b.

After some variations in the 2D classification parameters the best result was found without bimodal angular searches and decreased sampling values to avoid overfitting in the 3rd column. For this run also a new particle extraction had to be performed because the bimodal options must be in accordance (3rd line). The other extraction parameters were left unchanged. As a result, the classes were not blurred or washed out but only the least dense class in figure 4.20c showed the stripes of the helical coat. Only one of the classes collapsed.

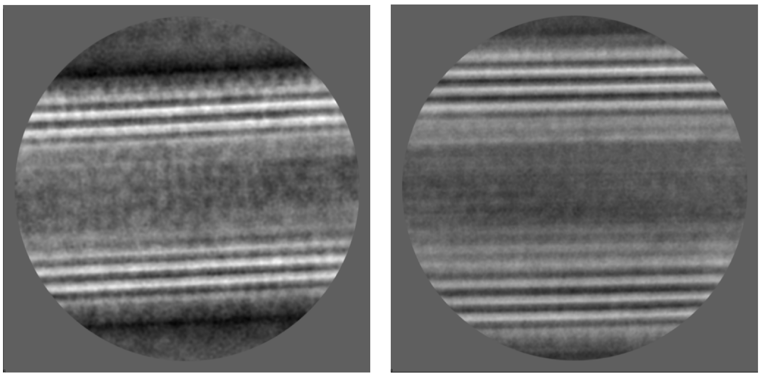


(a) 410 Å, 107 particles. **(b) 1100 Å**, 379 particles. **(c) 1100 Å**, 38 particles.

Figure 4.20: Examples of different 2D classification runs of IHRSR of 1986 extracted particles after 25 iterations with tubes sorted by LineAdd.py highlighted in figure 4.6.1. In **(a)** the mask diameter was set to 410 Å. This covers only the inner tube diameter and cuts off most the borders. The stripes of the helical oligomer are visible. **(b)** is the result of 1100 Å mask diameter and covers more than the complete tube diameter. The class view is blurred because of overfitting. Particles with a different border thickness were averaged. The helical coat is washed out. In **(c)** the sampling was optimized. This class was the best one obtained after various sampling experiments. However, this is only the least dense class.

4.6.2 Diameter Group Selection

To overcome the problem of different border numbers tubes with similar border thickness and similar outer diameters were selected using LineAdd.py. The dense area between 100 Å and 150 Å highlighted in figure 4.19 was selected as new input for RELION and contained 77 tubes. However, the differences in the diameter up to 206 Å between the classes worsened the problems of blurred classes and diameter-sorted classification as shown in figure 4.21a and 4.21b. The particle extraction and 2D classification parameters were identical to the ones from the 3$^{\text{rd}}$ line and column of the tables 4.6 and 4.7.



(a) 1100 Å, 125 particles. **(b) 1100 Å**, 81 particles.

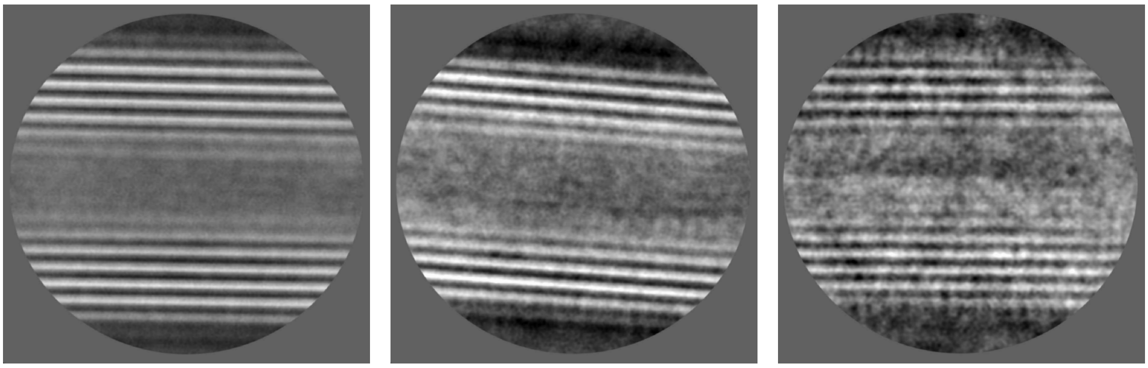
Figure 4.21: Two of ten classes of the slice marked in figure 4.19 after 2D classification. Notably in the shown classes the diameter between **(a)** (747 Å) and **(b)** (953 Å) is too much for a suitable 2D classification because tubes were again classified by their number of multilamellar borders instead of their helical coat. Moreover, with big differences in diameter the geometry between the tubes is also not comparable due to different curvatures.

4.6.3 Classification by RELION

Another approach to overcome the problem of different diameters is to obtain 2D classes of the complete dataset containing 1986 particles. The results of the previous tests showed that RELION classifies tubes by their borders. Instead of using LINEADD.PY to sort tubes by diameter RELION's 2D classification was used. Its Bayesian-based dynamic algorithm can be more precise than the static tube end detection algorithm because it takes more data into account. To perform the 2D classification the parameters from the 3$^{\text{rd}}$ run in table 4.7 were adopted and the number of classes was set to 100 because of the diameter difference between the thinnest and the thickest tube in the histogram of figure 4.18a is 580 Å. For a safer classification 100 instead of 58 classes were used. This resulted in 75 classes and 25 collapsed. The densest class contained 519 of 1986 particles. This is more than one third and should be enough to show if the approach works.

With the selected particles another 2D classification was performed with the previous parameters and ten classes. This run yielded eight classes. The densest class

(996 Å diameter) is shown in figure 4.22a and did not solve the previous problems. The class in figure 4.22b contains the thinnest tubes (931 Å diameter). The difference in the class diameter is 65 Å. The class in figure 4.22c is less blurred compared to the densest class but it contains only 17 particles of 519.



(a) 1100 Å, 327 particles. **(b) 1100 Å**, 65 particles. **(c) 1100 Å**, 17 particles.

Figure 4.22: Three of eight classes after using RELION for a diameter classification. **(a)** is the densest class and blurred by overfitting with a diameter of 996 Å. **(b)** contains thinner tubes and results in a 931 Å class diameter. **(c)** is less blurred than the other classes but contains with 17 particles far less than the other classes (327 and 65). It contains not enough particles to make the stripes of the helical coat visible.

Chapter 5

Discussion

5.1 Energy Distribution

In vitro, BPG extrudes lipid tubes from vesicles. The total energy difference ΔE between the energy of the extruded tube E_{Tube} and the energy of the original vesicle E_{Vesicle}[72] has to be supplied by the protein:

$$E_{\mathrm{Tube}} = E_{\mathrm{Vesicle}} + \Delta E$$
$$\Delta E = E_{\mathrm{Tube}} - E_{\mathrm{Vesicle}}$$

It can be calculated using the curvature energies for the vesicle and the tube calculated by equation 1.1 in chapter 1.1. This requires the shell area of a cylinder ($A_{\mathrm{cyl}} = 2\pi r_{\mathrm{T}} h$) and the surface area of a sphere ($A_{\mathrm{sph}} = 4\pi r_{\mathrm{V}}^2$). The total energy distribution is then provided by equation 5.1. In case of the cylinder only the shell is considered because the curvature of top and bottom base is small in comparison.

$$\Delta E = \frac{\kappa\pi}{4}\left[r_{\mathrm{T}} h_{\mathrm{T}}\left(C_{1,\mathrm{T}} + C_{2,\mathrm{T}} - C_0\right)^2 - 2r_{\mathrm{V}}^2\left(C_{1,\mathrm{V}} + C_{2,\mathrm{V}} - C_0\right)^2\right] \tag{5.1}$$

With known helical parameters $\Delta\varphi$ and Δz (figures 1.9a and 1.9b) and the tube length l the *approximate* number of units per tube can be obtained. Finally, the molecular work W_{mol} per unit can be estimated by equation 5.2:

$$W_{\mathrm{mol}} = \frac{\Delta\varphi\Delta z}{l \cdot 360\,^{\circ}}\Delta E \tag{5.2}$$

It would be infeasible to prepare a single vesicle and add BPG to extrude a tube from it and measure the curvature before and after incubation. However, a statistical comparison between the sizes of vesicles and tubes after BPG incubation can approximate a binding energy per molecule.

5.2 Nanotemplates

There are many different types of nanotemplates but not many are suitable for binding helical protein oligomers. They need a curvature that is sensed by the protein and the right diameter to be coated with a protein oligomer. Moreover, they should be linear and constant in diameter. For this case and diameter class there are notably two approaches: *CNT*-based (**C**arbon **N**ano **T**ube) and lipid-based. They differ in key features as preparation effort, flexibility and complexity. The least effort and flexibility feature GalCer-based lipid tubes. They extrude their selves when the lipid mixture contains more than 20 % GalCer. Their shape is linear and constant in diameter. However, according to literature, the diameter is fixed at 25 to 30 nm due to molecular properties of GalCer[31][33]. For small helical protein oligomers or binding single molecules (e.g. streptavidin on biotinylated lipids) GalCer tubes are an option. As second component of the lipid mixture a lipid type suitable for protein binding must be selected. The flexibility can be increased by selecting lipid cocktails like Folch containing differently charged lipids with different tail lengths and head groups. The major disadvantage besides the fixed diameter is the fact that GalCer is needed. This lipid is expensive and does not belong to most protein's native environments.

In contrast to GalCer tubes, CNT-based nanotemplates offer a lot of flexibility: CNTs are constant in diameter due to their structure and the diameter can in theory be selected by the number of graphene layers. The most common production method for multi-walled CNTs is *Chemical Vapour Deposition*. In this technique, CNTs are grown on catalyst ions. The growing point diameter controls the diameter of the CNT[73]. This is not very precise, but it allows to produce CNTs in diameter classes (about ±10 nm). Thus, CNTs can be purchased in the desired diameter groups. Moreover, long CNTs can be easily cut into smaller pieces with the help of ultrasound. There are also electroconductive CNTs[74]. This feature can enable experiments that require measurements of electrical properties. The main disadvantage of CNTs is the requirement of complex preparation methods. CNTs need to be modified chemically because of their non-biomimetic and hydrophobic properties. They need to be equipped with a lipid bilayer featuring a suitable curvature that can be sensed by the protein. Covalently bound lipid layers require disengaged bindings between carbon atoms in the surface lattice of the tube. This can be done using chemical radicals but there is no possibility to control the location and amount of split atom bindings. As a result, the lipid layers are not attached homogeneously. Non-covalently bound lipid layers require expensive synthetic lipids with pyrene groups to be attached to the CNT's surface by π-π-stacking. An alternative method is the usage of linker molecules between CNT (non-covalent side) and lipid head group (covalent side). The additional length of the linkers must be taken into account when selecting CNTs with a suitable diameter. Another disadvantage is that CNTs are not linear-shaped and in result not usable for helical reconstruction directly. Cutting CNTs into smaller fragments is one solution to straighten them. In addition, protein oligomerization can force CNTs to linear-shaped tubes. In conclusion, CNTs are an interesting approach for nanotemplates but it is very complex and requires many problems to be solved.

The Folch lipid tubes presented in this work are like GalCer tubes in their properties. They also feature the linear shape and constant diameter. Compared to GalCer tubes, the average diameter is larger at 96 ± 18 nm (table 4.1) and features a larger variance. This adds a bit of flexibility for protein oligomers but

the extrema are less likely to appear (figure 4.18a). The effort for the preparation using the lipid extruder is also low. The main advantage of Folch lipid tubes is that they already consist of a cocktail of differently charged lipids and also lipids with different tail lengths[75]. This allows them to be used without modification for many different proteins that are in the same environment of Folch *Type I* (extracted from bovine brain). Moreover, Folch lipids are at a price of $^1/_{28}$ (November 2018) far less expensive compared to GalCer. This calculation does not include the additional lipids needed to prepare GalCer lipid tubes.
Folch lipid tubes are multilamellar and hollow inside. This allows the protein to shrink them to the best fitting diameter for the protein oligomer and adds another flexibility compared to both, GalCer lipid tubes and CNTs.

5.3 BPG on LNTs

Mixing LNTs with BPG results in Cryo in the same helical stripes on the tube surface as BPG's self-extruded lipid tubes (figure 4.3b compared to 4.6). LNTs feature already tubular shapes, so there is no need for BPG to extrude its own tube and it binds directly. Except a small shrinking of the diameter (12 %) the LNTs keep their shape upon BPG binding, as shown in table 4.1. When BPG oligomerizes on the LNT it compresses the tube to its preferred diameter[76] constantly over the complete length of the tube. After shrinking the diameter of the BPG-coated LNTs is closer the diameter of BPG-extruded tubes (88 nm compared to 75 nm). The new diameter is the result of BPG's molecular organization. However, there is still a variance (±16 nm) in the diameter of LNTs (with and without BPG) because they feature different counts of multilamellar borders. Figure 5.1 visualizes the oligomerization of BAR domain proteins on tubes.

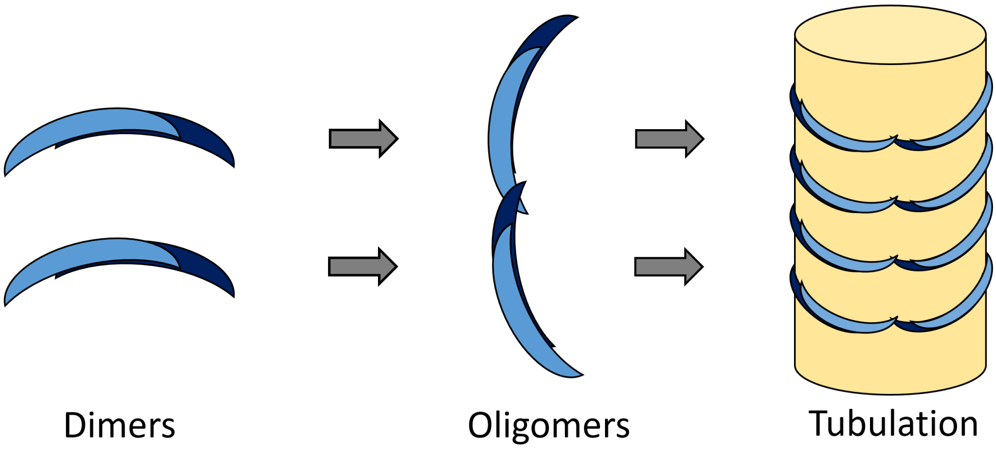


Figure 5.1: Dimers of the BAR domain protein form oligomers and extrude tubes from unilamellar vesicles. Phosphorylation at the tips of the dimers inhibit oligomerization to ensure that the oligomers get their characteristic shape[76].

LNTs keep their linear shape and the constant diameter when they are coated by BPG. In figure 4.6 the angle of the helical stripes is constantly at 67 ° over the whole tube. In comparison to the LNT the helical stripes on BPG's self-extruded tube in figure 4.3b are at an angle of only 58 °. Both tubes have a diameter of 75 nm. This is not a big difference, but it can possibly be a hint that the helical order on LNTs is different. Moreover, the BPG-extruded tube in Cryo and the one in negative stain in figure 4.3a show not a constant angle of the stripes over the whole tube. Behind curvatures in the tube the angle changes. This proves that an analysis of the self-extruded tubes would not yield reliable results in a helical reconstruction and distort the build of molecular models. Hence, LNTs solve the curvature issues and variable diameter of BPG's self-extruded tubes and enable the use of helical reconstruction to obtain the structure of BPG.

5.4 Optimization of LNTs

The original protocol for LNTs (chapter 3.3.2), derived from the ULV protocol (chapter 3.3.1), had to be optimized to yield a higher tube concentration. The first try to increase the concentration was to use a higher lipid concentration in the same buffer volume. Doubling and triplicating the concentration did only result in more aggregation instead of more tubes. Hence, a more sophisticated approach was needed. The buffer B adopted from the ULV protocol contains three components: NaCl, HEPES and EDTA. While HEPES keeps the pH value at 7.4 NaCl provides a meaningful concentration of Na^+ and Cl^-. These are present in similar concentrations inside eukaryotic cells (50 mM for Na^+ and 150 mM for Cl^-)[1]. EDTA is a chelating agent that binds Mg^{2+} and Ca^{2+}. These metal ions can affect lipid formation and aggregation. Omitting EDTA in buffer $B2$ results in a seven times higher amount of LNTs as shown in figure 4.8a compared to 4.8b. EDTA has a higher binding affinity for magnesium ions than for calcium ions. There is also the chelating agent EGTA that has a higher binding affinity for calcium ions compared to the affinity for magnesium ions[77]. To see the result of a lower calcium concentration a second test with EGTA instead of EDTA was performed. A comparison between the buffer solution B and the new one containing 1 mM EGTA (B^{EGTA}) instead of EDTA shows that EGTA exclusively results in unilamellar vesicles (4.8c). It seemed that magnesium supports the formation of vesicles and calcium in contrast the formation of tubes. Both chelating agents can also bind other metal ions. Therefore, it was needed to prove if magnesium or calcium are the ions that are responsible for the different results. A buffer solution $B3$ was mixed. It contains 1 mM of both chelating agents to ensure that magnesium and calcium are not present in the buffer solution in a significant concentration. In the next step either 5 mM of $MgCl_2$ ($B3^{\mathrm{MgCl_2}}$) or 5 mM $CaCl_2$ ($B3^{\mathrm{CaCl_2}}$) were added. The result in figure 4.9a proves that magnesium ions yield vesicles and 4.9b that calcium ions yield LNTs. Calcium is known to be rigidifying and ordering lipid layers by changing head groups[78], ordering acylic chains[79] and dehydrating lipids[80]. It also compresses bilayers[81]. These properties favor the formation of rigid tubes with ordered multilayers. In contrast to calcium, magnesium is a fusion catalyst without the rigidifying and ordering properties[82][83]. Thus, vesicles and aggregates are formed in presence of magnesium ions. In conclusion, the buffer $B3^{\mathrm{CaCl_2}}$ containing both chelating agents and 5 mM $CaCl_2$ yields the highest tube concentration.

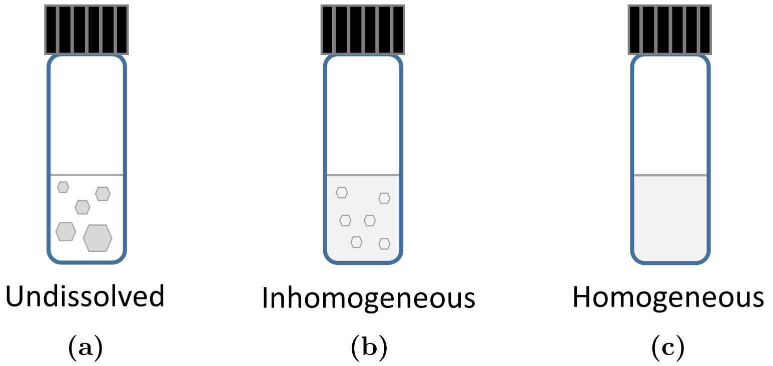
There are more variables in the tube preparation protocol: The number of freeze-thaw-cycles n_{FT}, the extrusion temperature T_{Ex} and the number of extrusion cycles n_{Ex}. These parameters had to be optimized to increase the tube yield. These variables cannot be tested separately because they can be correlated with each other. For this reason the CCD method for all three variables was used and each variable was tested for three different values. The results are shown in chapter 4.3.2 particularly in figure 4.10a. The results for each value of each variable were averaged to display the influence of the variables separately in figure 4.11.
At first, there is the number of freeze-thaw-cycles n_{FT} before using the lipid extruder to finish the protocol. Originally, this step should disrupt multilamellar vesicles to improve the yield of unilamellar vesicles during extrusion. It was not known if LNTs profit from freeze-thaw-cycles in the same way as ULVs do. In Cryo-EM images LNTs show multilamellar borders and they may profit from skipping this step to preserve multilamellar lipid vesicles. To investigate the influence of freeze-thaw-cycles, tests with more or less than five cycles were performed. Accordingly, n_{FT} was tested at $0\times$, $5\times$ and $10\times$. The results show that the concentration almost scales linearly with number of freeze-thaw-cycles n_{FT}. This shows that disrupting multilamellar vesicles is important for the formation of tubes. Although the tubes seem to be multilamellar as well, they form from unilamellar vesicles. During the extrusion non-uniform sized vesicles are pressed through the membrane and constrict new vesicles of a size like the membrane's pore size. Multilamellar vesicles are more rigid than unilamellar vesicles because more molecular layers must be deformed. Due to this reason it is less likely for a multilamellar vesicle to form a tube behind the membrane. In presence of calcium as a fusion agent for lipid molecules that also orders lipid layers new tubular multilamellar structures can be formed.

LNTs are formed by pressing lipids through the extrusion membrane. Increasing the number of extrusion cycles n_{Ex} should result in higher concentrations. Originally, 21 extrusion cycles were performed. When comparing the results for $0\times$, $21\times$ and $61\times$ it is clearly visible that almost no tubes form when the lipid-buffer-solution was not extruded $(0\times)$. This proves that the primary origin of tubes is the extrusion process. However, some tubes are also formed without extrusion. It is less likely that they form but it is not impossible because their origin is the same

as the origin of vesicles. At 61× there is only a small increase over 21 ×. At some point the tube formation reaches a limit because there are no lipids left to form tubes or too high tube concentration disrupts the formation of new tubes because it blocks the extruder membrane.
In literature was shown that more extrusion cycles reduce the multilamellarity of vesicles[84] and that the number of freeze-thaw-cycles increases the trap volume of vesicles[85]. Both variables show an increased tube yield. This shows that unilamellarity favors the tube extrusion and freeze-thaw-cycles may increase the number of large vesicles that can form tubes in presence of Ca^{2+}.

The extrusion was originally performed at a temperature T_{Ex} of 25 °C and at 21 × for the number of extrusion cycles n_{Ex}. Both variables can influence the tube yield and thus they have to be taken into account in the optimization process as well. T_{Ex} influences the viscosity of the lipid-buffer-solution and Folch is a complex mixture of multiple lipid types. The temperature can influence molecular transitions between them. For T_{Ex}, values of 5 °C, 25 °C and 45 °C were tested. While extrusion temperatures (T_{Ex}) of 5 °C and 45 °C yielded similar tube concentrations, 25 °C first showed a clearly (0.5 times) lower concentration. To see if there is a reason for the disadvantage at this temperature more values for T_{Ex} were added while keeping the n_{FT} and n_{Ex} constant. The results for the more detailed temperature correlation in figure 4.12 show similar results for 5 °C, 15 °C, 35 °C and 45 °C. In contrast to the previous CCD experiments the tube concentration was higher for 25 °C and not lower than the other temperatures. This is contradictory. However, when considering the error bars the results are not reliable enough to see a temperature correlation at all (notably the ones for 15 °C and 25 °C). This was also shown in literature[84]. The large error bars show that there must be a different reason behind the variation between experiments that are identical in all defined variables.

First of all, the lipid concentration can vary between experiments because some lipids can remain undissolved. After rehydrating with buffer solution for 20 min the there were only aggregates in the mixture (figure 5.2a). Normally, after sonication the solution was homogeneously non-transparent because all lipids were dissolved (5.2c). The sonication time was varying between experiments.



(a) (b) (c)

Figure 5.2: Glass vials showing the steps in lipid rehydration. **(a)** is a transparent buffer solution containing undissolved lipid aggregates. In **(b)** most lipids are dissolved but there remain some smaller almost invisible aggregates and the solution is already non-transparent. **(c)** does not contain an appreciable amount of undissolved lipids and is homogeneously non-transparently colored. Sonication of **(a)** finally results in **(c)**.

However, if the sonication time is not long enough the solution still contains small barely visible aggregates, as illustrated in figure 5.2b, and these do not contribute to the extruded solution and lower the tube concentration. Moreover, a sedimentation of lipid aggregates on the walls of the glass vials could be observed during freeze-thaw-cycles. This was notably visible around the water surface area.

Secondly, during the experiments was noticed that the volumes inside the glass vials were different between aliquots after the freeze-thaw-cycles although they were filled with a constant amount of buffer solution in the beginning. This could be observed when using a pipette set to a defined volume to divide the solution into smaller aliquots to be stored after the freeze-thaw-cycles. The fluid volume can be less because of less dissolved lipids. Moreover, it is possible that the glass vials were not tight enough that water could evaporate and escape as steam during heating up to 70 °C. In addition, lipid aggregates on the wall of the vial can also retain water between multiple lipid layers[72]. This amount of water is not present inside the volume of the solution anymore. However, the volume itself cannot be responsible for differences in the tube concentration because the pressure p is constant due to the constant area A of the extrusion membrane: $p = \frac{F}{A}$. However, the force F can change the pressure. In literature it was shown that the extrusion pressure is influencing the vesicle size and trapped volume[85]. All extrusions were done by hand using the extruder shown in figure 3.2b. With larger volumes inside the syringes the deflection of the fingers increases. With less deflection more force can be exerted on the piston of the syringe and the pressure can be increased. Thus, the pressure differences dependent on the volume are due to manual inaccuracies. When using an automatic extruder with a fixed pressure volume-dependencies should be negligible.

Finally, the gradient of the CCD is used to visualize the results of the whole optimization process: It points towards the maximum change in the spatial diagram (see chapter 1.6). It was first approximated without considering the additional temperature experiments. According to these experiments the tube concentration has its maximum where all three variables have their highest value at the coordinates 222 ($n_{\mathrm{FT}} = 10\,\times$, $T_{\mathrm{Ex}} = 45\,°\mathrm{C}$, $n_{\mathrm{Ex}} = 61\,\times$). In figure 5.3 the green arrow is the gradient of the original CCD experiment.

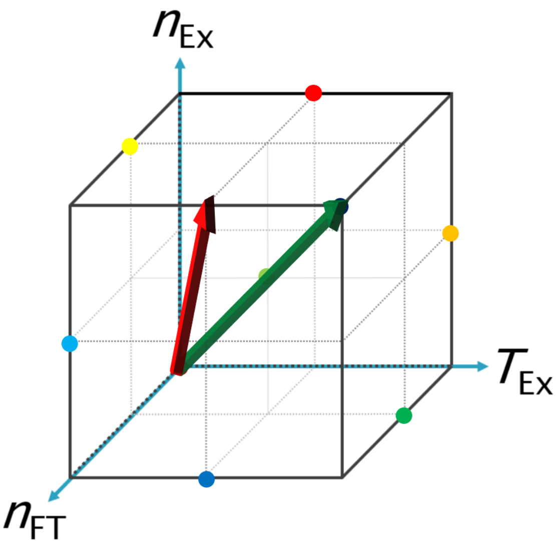


Figure 5.3: The green arrow is the gradient of the original CCD experiment. The red arrow is a more likely result for the gradient.

Assuming that the tube count at 222 is similar to the points 102, 111 and 201 a better approximation for the gradient is the red arrow in figure 5.3 because 201 and 102 yield high concentrations with T_{Ex} lower than the maximum at 45 °C. The additional temperature correlation experiment shows a maximum at 25 °C. Taking this result into account when estimating the gradient, the red arrow is even more likely pointing towards 212 ($n_{\mathrm{FT}} = 10\,\times$, $T_{\mathrm{Ex}} = 25\,^{\circ}\mathrm{C}$, $n_{\mathrm{Ex}} = 61\,\times$). This seems to be the best approximation of the gradient but due to the high error bars in figure 4.12 no temperature correlation could be verified, and a more precise gradient cannot be established with the present data.

Another important aspect to achieve an optimal LNT concentration in samples is to consider their stability and usage. ULVs should be stored for a maximum of three days at 4 °C and must not be vortexed. Otherwise they collide and aggregate. LNTs are more robust due to their rigid multilamellar shape. They must not be vortexed as well but they can be stored at room temperature and for a longer time than three days. Although, after some days they show differences in shape and aggregation in the same aliquot. For example, a higher tube concentration could be observed when using a stock that was stored for one or two days compared to a fresh one. Other samples suffered from aggregation like tubes that have been vortexed. The graph of the experiment in figure 4.14 shows that the count of non-tubular objects in the aliquot decreases while the tube concentration increases within days converging to a maximum after five days. This can be a proof that non-tubular objects are straightened. During the extrusion process vesicles are pressed through the extruder membrane. Due to the extrusion force vesicles with a new shape and size are formed and translational energy is converted into curvature. However, flexible strained surfaces tend to relax to their equilibrium and lose curvature[10]. When losing curvature in presence of Ca^{2+} ions this could yield new tubes. In the CCD was shown that tubes can also form without any extrusion cycles. Their low concentration shows that this is not very likely. For long tube-like vesicles it is more likely to straighten into LNTs over the time because they are closer to their final shape. Lipid structures stored in an aqueous solution aggregate over time due to gravity and thermal induced movements. In presence of fusion agents' smaller vesicles can aggregate and also lose curvature. In combination with Ca^{2+} ions ordering and dehydrating lipid structures this could be a second way to form tubes over the storage time.

5.5 Helical Reconstruction

The helical parameters obtained from the negative stain tomograms in chapter 4.1.2 do not have to match the parameters obtained from Cryo-EM because the tubes might be deformed when the sample is dried on the grid for negative staining. However, obtaining helical parameters from Cryo-EM samples is much more complex because of the low contrast compared to stained samples. From the Cryo tomogram in figure 4.6 the helical parameters cannot be obtained manually in a reliable way. Originally it was attempted to use sub-tomogram averaging to acquire helical parameters of Cryo samples and to study BPG because the first LNT preparations did not yield concentrations high enough for helical reconstruction. However, the successful optimization of the LNT preparation enables IHRSR. In case of high enough concentrations of tubular shaped particles this is an easier approach because it does not require reconstructed tomograms and a compensation of the missing wedge.

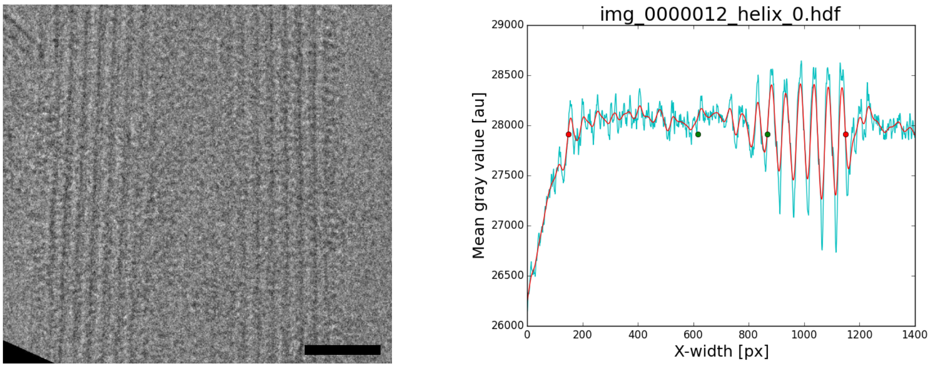
In a first attempt of helical reconstruction the distribution of different diameters between the tubes led to problems in the 2D classification. Instead of classifying the helical structure in the center of the tube, RELION classified tubes by the number of multilamellar lipid layers visible in the tube borders. Setting the inner mask diameter to fit in between the borders of the smallest tube would be a trivial solution but the cylindrical shape of the surface area on the tubes must be considered as well. The different curvatures of small and large tubes at locations with the same radius result in incomparable classes. The only solution to overcome this problem is presorting the tubes by their diameter. Instead of measuring the tubes manually the script LINEADD.PY (described in chapter 3.3.15) was written. Its tube detection algorithm sorts the tubes by their inner diameter in predefined classes. The histogram plot of all diameter classes is plotted in figure 4.18a and looks like a Gaussian distribution. The Gaussian graph with the best fitting standard deviation was automatically calculated and plotted. However, there is not enough data to examine the correct underlying statistical distribution. There is a density peak in the center of the histogram.

For a second try of IHRSR only presorted tubes of the two central classes (420 to 439 Å and 440 to 459 Å) highlighted in figure 4.18a were selected. The mask diameter of 410 Å covers only the inner diameter of the tube. This proved that it is indeed possible to see the vertical stripes of the helical BPG-coat on the LNTs as shown in the 2D class in figure 4.20a. However, this was only a test and is not suitable for obtaining a 3D model because a mask diameter covering only the inner part does not include the full tubular geometry and can in theory only reconstruct a partial model of the helix.
When using a mask diameter covering more than the whole tube (1.5 times is recommended[86]) with a presorted tube subset the tubes are not only classified by their number of multilamellar borders anymore. Certainly, there are still tubes of different diameters and border numbers in the subset as it covers an inner diameter interval of 40 Å in total (highlighted in figure 4.18a). These border diameter variations in the selected subset disrupt the classification because now tubes with small diameter differences are averaged to one class. Moreover, there is almost no correlation between the outer diameter and the border diameter as shown in figure 4.19. This also means that there is almost no correlation between the inner and outer diameter. Accordingly, tubes presorted by their inner diameter feature different outer diameters randomly distributed.
In case of the presorted classification seven of ten classes collapsed. This blurred out the classes and added up the strong signals of the tube borders (compared to the helical coat) as shown in figure 4.20b.
Testing different sets of particle extraction and 2D classification parameters revealed that smaller sampling ranges and steps lead to less blurred classes. Small parameters would help to align the tubes in their axis perpendicular to the tube. Though these were too small to align the stripes of the helical coat in the tube

axis. Unfortunately, RELION only supports one parameter for both axes[86]. Furthermore, testing parameter sets revealed that the option for *bimodal angular searches* is inappropriate for BPG-coated LNTs. This shows that the helical segments lack polarity and indicate a dihedral $\mathbf{D_n}$ symmetry[86]. Switching off this option does not solve the problem of averaging up tubes of different diameters but makes the helical coat on the tubes visible in figure 4.20c. This is the least dense class of this run and contains only 38 of 1986 particles. This is not sufficient for a high-resolution 3D reconstruction. Most of the other particles were averaged up in blurred classes again.

To overcome the problem of the different tube diameters and different border numbers a new subset of the dataset was used. The diagram in figure 4.19 was used to find a subset with similar outer diameters and similar border diameters. In theory, the perfect dataset would consist of tubes with equal diameters and borders. Selecting a subset containing *similar* tubes is a better approximation than using the complete dataset. For the subset a crowded area in the diagram with an interval of 50 Å for the border diameters was used. This is close to previous 2D classifications using the two main peaks of the inner diameter with an interval of 40 Å and resulted in a similar tube count (77 compared to 67 tubes). However, the outer diameters of the selected interval also vary around more than 200 Å. This explains the diameter difference of 206 Å between the class of the thinnest tubes in figure 4.21a and the class of the thickest tubes in 4.21b. This disturbs the classification and results in more blurred classes compared to the previous classification. Moreover, it was not successful to use RELION's 2D classification to classify the tubes diameter. This resulted in a less precise classification than LINEADD.PY shown in figure 4.18a (65 Åcompared to 40 Å difference in diameter). 327 particles were put in the densest class (4.22a) and led to a blurred not usable class.

In conclusion, the two main problems of the 2D classification are differences in the diameter and the strong signal of the multilamellar borders compared to the helical coat in between. While the diameter difference is a physical problem it can only be improved using a better diameter classification algorithm in the script. Although a better diameter sorting always implies less particles for a 2D classification. They can refine an existing model in later runs but for building an initial model they are not available. There are also wrongly classified tubes. These can as well disturb the classification because particles are missing in the right class and instead disturbing another class. The failure detection described in chapter 3.3.15 is only based on a static mathematical criterion that does not recognize all possible failures. As an example, figure 5.4 shows a tube micrograph and the corresponding density graph.



(a) Wrong detected particle. Scale bar: 20 nm.

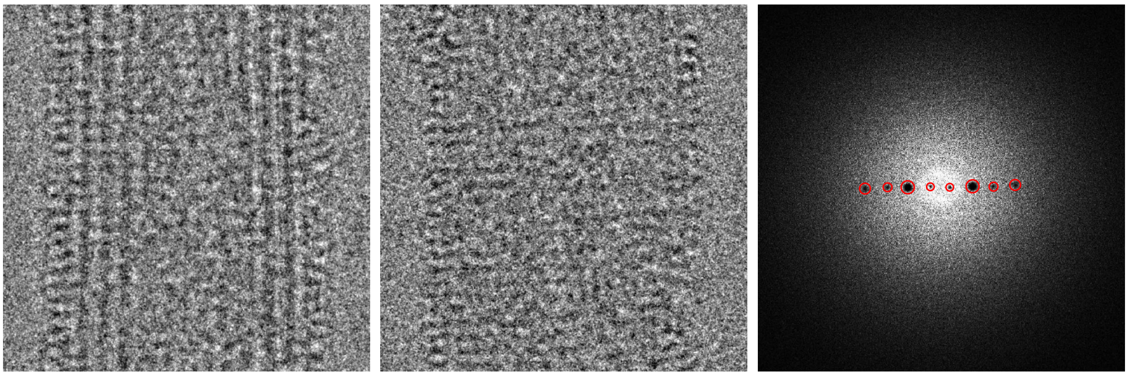
(b) Output of LINEADD.PY. The left borders are not visible.

Figure 5.4: An example for a poor tube end detection. The low average contrast of the left border results in a wrong inner diameter approximation.

By eye the whole tube can be identified. In contrast, the script does not use the complete tube. It first averages all horizontal lines over the tube. Parts with strong contrast are mixed up with parts with weak contrast. If most of the parts are weak the tube average will show weak contrast in the density graph as well. For a more robust detection algorithm all lines must be detected separately and averaged to a weighted result. This application could benefit from a machine learning algorithm as well as other single particle based detections[87][88]. Moreover, skew picked tubes, as shown in figure 3.6c, resulted in a washed-out border signal and imprecise and unreliable tube detections. These tubes can also disrupt IHRSR because the columns in the micrograph containing similar views are shifted and the helical parameters are not suitable anymore for that particle.

5.6 Outlook

The remaining problem of the difference in the signal between borders and inner tube can be solved using a selective filter that removes the tube border signal in Fourier space. However, this filter should not just cut the frequencies responsible for the borders because in this case any other information at the same frequencies is also cut. A better approach is to remove only selective reflexes in Fourier space considering their spatial location. In figure 5.5 a tube micrograph is shown before and after filtering and also its Fourier transform after filtering with cut reflexes of the tube borders.



(a) Tube micrograph $A[x,y]$.

(b) Filtered micrograph $A_{\text{filt}}[x,y]$.

(c) Normed Fourier transform $\left|\mathcal{F}\{A_{\text{filt}}[x,y]\}\right|$.

Figure 5.5: The raw tube micrograph in **(a)** is filtered in Fourier space and results in **(b)**. The tube borders of **(a)** are not present in **(b)** anymore because the responsible frequencies were cut in Fourier space in **(c)**. To show the amplitude it was normed ($f : \mathbb{C}^{n \times n} \to \mathbb{R}^{n \times n}$) as shown in equation 5.4. The centered black dots (highlighted in *red* for better view) are zeroed and contain no more information.

Equation 5.3 describes the filter process of the quadratic and 2^k-padded real space image $A[x,y] \in \mathbb{R}^{n \times n}$ with the filter kernel $F[u,v] \in \mathbb{C}^{n \times n}$ and a window function $W[u,v] \in \mathbb{C}^{n \times n}$[37].

$$A_{\text{filt}}[x,y] = \mathcal{F}^{-1}\big\{\mathcal{F}\{A\}[u,v] \cdot W[u,v] \cdot F[u,v]\big\} \tag{5.3}$$

$$\big|\mathcal{F}\{A_{\text{filt}}[x,y]\}\big| = \Big(\Re\big(\mathcal{F}\{A_{\text{filt}}[x,y]\}\big)^2 + \Im\big(\mathcal{F}\{A_{\text{filt}}[x,y]\}\big)^2\Big)^{\frac{1}{2}} \tag{5.4}$$

As a filter kernel it should be sufficient to draw an image containing zeroes (*black* color) located at the coordinates of the reflexes to be removed and ones (*white* color) for all other pixel gray scale values. A multiplication in Fourier space removes only the selected reflexes while preserving the other information. The filter does not need to remove reflexes completely. It should be enough to attenuate them to be at a similar amplitude as the frequencies of the helical coat. The filter can be applied to all micrographs using a script and enable a 2D classification yielding classes of the helical coat instead of tube borders.

Although not completely proven in this work, this is a sophisticated approach to solve the remaining problem. The 2D classes will be the basis for a 3D classification and 3D auto-refinement. Finally, when a working approach with the given data is established there is the possibility to improve the resolution and refine the first 3D structure with more datasets.

From the current point of view different states of the BPG binding process can be analyzed with IHRSR. This can also enable the analysis of other tubular geometry sensing proteins. For example, *EHD2* (**E**ps **H**omology **D**omain Containing **2**) extrudes lipid tubes from Folch vesicles as well[89]. There are good chances that EHD2 is usable on our Folch-based nanotemplates as well.

Bibliography

[1] Lodish, H. *et al.* *Molecular Cell Biology 6th Edition*, vol. 6 (W.H. Freeman, 2007).

[2] Sandvig, K., Pust, S., Skotland, T. & van Deurs, B. Clathrin-independent endocytosis: mechanisms and function. *Current Opinion in Cell Biology* **23**, 413–420 (2011).

[3] Johannes, L., Wunder, C. & Bassereau, P. Bending "on the rocks"–a cocktail of biophysical modules to build endocytic pathways. *Cold Spring Harbor perspectives in biology* **6** (2014).

[4] Kirchhausen, T., Owen, D. & Harrison, S. C. Molecular structure, function, and dynamics of clathrin-mediated membrane traffic. *Cold Spring Harbor perspectives in biology* **6**, a016725 (2014).

[5] Doherty, G. J. & Lundmark, R. GRAF1-dependent endocytosis. *Biochemical Society transactions* **37**, 1061–1065 (2009).

[6] Mayor, S., Parton, R. G. & Donaldson, J. G. Clathrin-independent pathways of endocytosis. *Cold Spring Harbor perspectives in biology* **6** (2014).

[7] Lacy, M. M., Ma, R., Ravindra, N. G. & Berro, J. Molecular mechanisms of force production in clathrin-mediated endocytosis. *FEBS Letters* (2018).

[8] Shen, H.-H., Lithgow, T. & Martin, L. Reconstitution of membrane proteins into model membranes: seeking better ways to retain protein activities. *International journal of molecular sciences* **14**, 1589–1607 (2013). Original DateCompleted: 20130125.

[9] Seifert, U., Miao, L., Döbereiner, H.-G. & Wortis, M. Budding transitions of fluid-bilayer vesicles: The effect of area-difference elasticity. *Physical review. E, Statistical physics, plasmas, fluids, and related interdisciplinary topics* **49**, 5389–5407 (1994).

[10] Seifert, U. Fluide Membranen. *Inistitut für Festkörperforschung, Forschungszentrum Jülich, Theorie II* .

[11] Fenz, S. F. & Sengupta, K. Giant vesicles as cell models. *Integrative biology : quantitative biosciences from nano to macro* **4**, 982–995 (2012).

[12] Sezgin, E. & Schwille, P. Model membrane platforms to study protein-membrane interactions. *Molecular membrane biology* **29**, 144–154 (2012).

[13] Voeltz, G. K., Prinz, W. A., Shibata, Y., Rist, J. M. & Rapoport, T. A. A class of membrane proteins shaping the tubular endoplasmic reticulum. *Cell* **124**, 573–586 (2006).

[14] Simunovic, M. *et al.* Friction Mediates Scission of Tubular Membranes Scaffolded by BAR Proteins. *Cell* **170**, 172–184.e11 (2017).

[15] Ford, M. G. J. *et al.* Curvature of clathrin-coated pits driven by epsin. *Nature* **419**, 361–366 (2002).

[16] Melo, A. A. *et al.* Structural insights into the activation mechanism of dynamin-like EHD ATPases. *Proceedings of the National Academy of Sciences of the United States of America* **114**, 5629–5634 (2017).

[17] McDowall, A. W. *et al.* Electron microscopy of frozen hydrated sections of vitreous ice and vitrified biological samples. *Journal of microscopy* **131**, 1–9 (1983).

[18] Handley, D. A., Alexander, J. T. & Chien, S. The design and use of a simple device for rapid quench-freezing of biological samples. *Journal of microscopy* **121**, 273–282 (1981).

[19] Doherty, G. J. *et al.* The endocytic protein GRAF1 is directed to cell-matrix adhesion sites and regulates cell spreading. *Molecular biology of the cell* **22**, 4380–4389 (2011).

[20] Lundmark, R. *et al.* The GTPase-activating protein GRAF1 regulates the CLIC/GEEC endocytic pathway. *Current biology : CB* **18**, 1802–1808 (2008).

[21] Francis, M. K. *et al.* Endocytic membrane turnover at the leading edge is driven by a transient interaction between Cdc42 and GRAF1. *Journal of cell science* **128**, 4183–4195 (2015).

[22] Eberth, A. *et al.* A BAR domain-mediated autoinhibitory mechanism for RhoGAPs of the GRAF family. *The Biochemical journal* **417**, 371–377 (2009).

[23] Noguchi, H. Membrane tubule formation by banana-shaped proteins with or without transient network structure. *Scientific reports* **6**, 20935 (2016).

[24] Zimmerberg, J. & McLaughlin, S. Membrane Curvature: How BAR Domains Bend Bilayers. *Current Biology* **14**, R250–R252 (2004).

[25] Stanishneva-Konovalova, T. B., Derkacheva, N. I., Polevova, S. V. & Sokolova, O. S. The Role of BAR Domain Proteins in the Regulation of Membrane Dynamics. *Acta naturae* **8**, 60–69 (2016).

[26] Peter, B. J. *et al.* BAR domains as sensors of membrane curvature: the amphiphysin BAR structure. *Science (New York, N.Y.)* **303**, 495–499 (2004).

[27] Marsh, H. & Rodríguez-Reinoso, F. *Activated Carbon* (Elsevier Science, 2006).

[28] Dayani, Y. & Malmstadt, N. Lipid bilayers covalently anchored to carbon nanotubes. *Langmuir : the ACS journal of surfaces and colloids* **28**, 8174–8182 (2012).

[29] Liu, J. *et al.* Stable non-covalent functionalisation of multi-walled carbon nanotubes by pyrene–polyethylene glycol through π–π–stacking. *New J. Chem.* **33**, 1017–1024 (2009).

[30] Ehli, C. *et al.* Interactions in Single Wall Carbon Nanotubes/Pyrene/Porphyrin Nanohybrids. *Journal of the American Chemical Society* **128**, 11222–11231 (2006).

[31] Kulkarni, V. S., Anderson, W. H. & Brown, R. E. Bilayer nanotubes and helical ribbons formed by hydrated galactosylceramides: acyl chain and headgroup effects. *Biophysical journal* **69**, 1976–1986 (1995).

[32] Lai, C.-L. *et al.* Membrane binding and self-association of the epsin N-terminal homology domain. *Journal of molecular biology* **423**, 800–817 (2012).

[33] Wilson-Kubalek, E. M., Brown, R. E., Celia, H. & Milligan, R. A. Lipid nanotubes as substrates for helical crystallization of macromolecules. *Proceedings of the National Academy of Sciences of the United States of America* **95**, 8040–8045 (1998).

[34] Grigorieff, N. Direct detection pays off for electron cryo-microscopy. *eLife* **2**, e00573 (2013). Original DateCompleted: 20130222, Original DateCompleted: 20140206.

[35] Allegretti, M., Mills, D. J., McMullan, G., Kühlbrandt, W. & Vonck, J. Atomic model of the F420-reducing [NiFe] hydrogenase by electron cryo-microscopy using a direct electron detector. *eLife* **3**, e01963 (2014). Original DateCompleted: 20140226.

[36] Alpers, A. *et al.* Geometric reconstruction methods for electron tomography. *Ultramicroscopy* **128**, 42–54 (2013).

[37] Buzug, T. M. *Einführung in die Computertomographie* (Springer-Verlag GmbH, 2005).

[38] Turoňová, B., Marsalek, L. & Slusallek, P. On geometric artifacts in cryo electron tomography. *Ultramicroscopy* **163**, 48–61 (2016).

[39] Harris, F. On the use of windows for harmonic analysis with the discrete fourier transform. *Proceedings of the IEEE* **66**, 51–83 (1978).

[40] Bharat, T. A. M. & Scheres, S. H. W. Resolving macromolecular structures from electron cryo-tomography data using subtomogram averaging in RELION. *Nature protocols* **11**, 2054–2065 (2016).

[41] Zhao, Y., Zeng, X., Guo, Q. & Xu, M. An integration of fast alignment and maximum-likelihood methods for electron subtomogram averaging and classification. *Bioinformatics (Oxford, England)* **34**, i227–i236 (2018).

[42] Scheres, S. H. W., Melero, R., Valle, M. & Carazo, J.-M. Averaging of electron subtomograms and random conical tilt reconstructions through likelihood optimization. *Structure (London, England : 1993)* **17**, 1563–1572 (2009).

[43] Bluemke, D. A., Carragher, B. & Josephs, R. The reconstruction of helical particles with variable pitch. *Ultramicroscopy* **26**, 255–270 (1988).

[44] Sosa, H., Hoenger, A. & Milligan, R. A. Three different approaches for calculating the three-dimensional structure of microtubules decorated with kinesin motor domains. *Journal of structural biology* **118**, 149–158 (1997).

[45] Egelman, E. H. A robust algorithm for the reconstruction of helical filaments using single-particle methods. *Ultramicroscopy* **85**, 225–234 (2000).

[46] Behrmann, E. *et al.* Real-space processing of helical filaments in SPARX. *Journal of structural biology* **177**, 302–313 (2012).

[47] He, S. & Scheres, S. H. W. Helical reconstruction in RELION. *Journal of structural biology* **198**, 163–176 (2017).

[48] Witek-Krowiak, A., Chojnacka, K., Podstawczyk, D., Dawiec, A. & Pokomeda, K. Application of response surface methodology and artificial neural network methods in modelling and optimization of biosorption process. *Bioresource technology* **160**, 150–160 (2014).

[49] Myers, R. H., Montgomery, D. C. & Anderson-Cook, C. M. *Response Surface Methodology* (John Wiley & Sons Inc, 2016).

[50] Zhu, T. F., Budin, I. & Szostak, J. W. Vesicle extrusion through polycarbonate track-etched membranes using a hand-held mini-extruder. *Methods in enzymology* **533**, 275–282 (2013).

[51] Avanti Polar Lipids, Inc. *Lipid Extruder.*

[52] Bokori-Brown, M. *et al.* Cryo-EM structure of lysenin pore elucidates membrane insertion by an aerolysin family protein. *Nature communications* **7**, 11293 (2016).

[53] Yuan, B., Zhang, Z. & Zhou, K. Graphene oxide monolayers as supporting films for high resolution transmission electron microscopy. *Applied Surface Science* **257**, 5754–5758 (2011).

[54] Mastronarde, D. N. Automated electron microscope tomography using robust prediction of specimen movements. *Journal of structural biology* **152**, 36–51 (2005).

[55] Mastronarde, D. N. & Held, S. R. Automated tilt series alignment and tomographic reconstruction in IMOD. *Journal of structural biology* **197**, 102–113 (2017).

[56] Kremer, J. R., Mastronarde, D. N. & McIntosh, J. R. Computer visualization of three-dimensional image data using IMOD. *Journal of structural biology* **116**, 71–76 (1996).

[57] Mastronarde, D. N. Correction for non-perpendicularity of beam and tilt axis in tomographic reconstructions with the IMOD package. *Journal of microscopy* **230**, 212–217 (2008).

[58] Frank, J. *Electron Tomography: Three-Dimensional Imaging With the Transmission Electron Microscope* (1992).

[59] Rudolph, D. *Charakterisierung des Falcon-Direktelektronendetektors und Anwendungen in der Kryo-Transmissionselektronenmikroskopie.* Master's thesis (2014).

[60] Zheng, S. Q. *et al.* MotionCor2: anisotropic correction of beam-induced motion for improved cryo-electron microscopy. *Nature methods* **14**, 331–332 (2017).

[61] Hagen, W. J. H., Wan, W. & Briggs, J. A. G. Implementation of a cryo-electron tomography tilt-scheme optimized for high resolution subtomogram averaging. *Journal of structural biology* **197**, 191–198 (2017).

[62] Carragher, B. *et al.* Leginon: an automated system for acquisition of images from vitreous ice specimens. *Journal of structural biology* **132**, 33–45 (2000).

[63] Tang, G. *et al.* EMAN2: an extensible image processing suite for electron microscopy. *Journal of structural biology* **157**, 38–46 (2007).

[64] Hohn, M. *et al.* SPARX, a new environment for Cryo-EM image processing. *Journal of structural biology* **157**, 47–55 (2007).

[65] Behrmann, E. *Structure of the Actin/Tropomyosin/Myosin Rigor Complex as Revealed by Cryo-Electron Microscopy*. Ph.D. thesis (2012).

[66] Butterworth, S. On the Theory of Filter Amplifiers. *Experimental wireless & the wireless engineer* **7**, 536–541 (1930).

[67] Bronstein, I. N., Mühlig, H., Musiol, G. & Semendjajew, K. A. *Taschenbuch der Mathematik*, vol. 7 (2008).

[68] The Scipy community. Butterworth digital and analog filter design. URL `https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.signal.butter.html`.

[69] Scheres, S. H. W. A Bayesian view on cryo-EM structure determination. *Journal of molecular biology* **415**, 406–418 (2012).

[70] Scheres, S. H. W. RELION: implementation of a Bayesian approach to cryo-EM structure determination. *Journal of structural biology* **180**, 519–530 (2012).

[71] Zhang, K. Gctf: Real-time CTF determination and correction. *Journal of structural biology* **193**, 1–12 (2016).

[72] Lipowsky, R. & Sackmann, E. *Structure and Dynamics of Membranes: I. from Cells to Vesicles / II. Generic and Specific Interactions* (ELSEVIER SCIENCE & TECHNOLOGY, 1995).

[73] Purohit, R., Purohit, K., Rana, S., Rana, R. & Patel, V. Carbon nanotubes and their growth methods. *Procedia Materials Science* **6**, 716–728 (2014).

[74] Patole, S. P., Arif, M. F., Susantyoko, R. A., Almheiri, S. & Kumar, S. A wet-filtration-zipping approach for fabricating highly electroconductive and auxetic graphene/carbon nanotube hybrid buckypaper. *Scientific reports* **8**, 12188 (2018).

[75] FOLCH, J., LEES, M. & SLOANE STANLEY, G. H. A simple method for the isolation and purification of total lipides from animal tissues. *The Journal of biological chemistry* **226**, 497–509 (1957).

[76] Salzer, U., Kostan, J. & Djinović-Carugo, K. Deciphering the BAR code of membrane modulators. *Cellular and molecular life sciences : CMLS* **74**, 2413–2438 (2017).

[77] Metzler, D. E. *Biochemistry: Chemical Reactions of Living Cells* (Academic Pr, 1977).

[78] Boettcher, J. M. *et al.* Atomic view of calcium-induced clustering of phosphatidylserine in mixed lipid bilayers. *Biochemistry* **50**, 2264–2273 (2011).

[79] Garidel, P., Blume, A. & Hübner, W. A fourier transform infrared spectroscopic study of the interaction of alkaline earth cations with the negatively charged phospholipid 1, 2-dimyristoyl-sn-glycero-3-phosphoglycerol. *Biochimica et biophysica acta* **1466**, 245–259 (2000).

[80] Porasso, R. D. & López Cascales, J. J. Study of the effect of Na^+ and Ca^{2+} ion concentration on the structure of an asymmetric DPPC/DPPC + DPPS

lipid bilayer by molecular dynamics simulation. *Colloids and surfaces. B, Biointerfaces* **73**, 42–50 (2009).

[81] Melcrová, A. *et al.* The complex nature of calcium cation interactions with phospholipid bilayers. *Scientific reports* **6**, 38035 (2016).

[82] Lind, T. K. *et al.* Formation and characterization of supported lipid bilayers composed of hydrogenated and deuterated escherichia coli lipids. *PloS one* **10**, e0144671 (2015).

[83] Schultz, Z. D., Pazos, I. M., McNeil-Watson, F. K., Lewis, E. N. & Levin, I. W. Magnesium-induced lipid bilayer microdomain reorganizations: implications for membrane fusion. *The journal of physical chemistry. B* **113**, 9932–9941 (2009).

[84] Hunter, D. & Frisken, B. Effect of Extrusion Pressure and Lipid Properties on the Size and Polydispersity of Lipid Vesicles. *Biophysical Journal* **74**, 2996–3002 (1998).

[85] Hope, M. J., Bally, M. B., Webb, G. & Cullis, P. R. Production of large unilamellar vesicles by a rapid extrusion procedure: characterization of size distribution, trapped volume and ability to maintain a membrane potential. *Biochimica et biophysica acta* **812**, 55–65 (1985).

[86] He, S. & Scheres, S. H. W. Helical processing (2017). URL `https://www2.mrc-lmb.cam.ac.uk/relion/index.php?title=Helical_processing`.

[87] Wagner, T. *et al.* SPHIRE-crYOLO: A fast and well-centering automated particle picker for cryo-EM (2018).

[88] Heimowitz, A., Andén, J. & Singer, A. APPLE picker: Automatic particle picking, a low-effort cryo-EM framework. *Journal of Structural Biology* **204**, 215–227 (2018).

[89] Shah, C. *et al.* Structural insights into membrane interaction and caveolar targeting of dynamin-like EHD2. *Structure* **22**, 409–420 (2014).

List of Figures

1.1 Two layers of lipid molecules form a curved membrane with the number of lipid molecules N_1 and N_2 and the distance d from the center. 4

1.2 From ULV extruded lipid tube. The stripes on the tube illustrate a helical oligomeric coat of protein molecules. 6

1.3 **(a)** shows a dimeric domain model of GRAF1. It is composed of N-BAR, PH, rhoGAP and SH3 domains. The SH3 domains interact with ligands to unblock the membrane binding module. **(b)** BPG is a fragment of GRAF1 lacking the SH3 domains. BPG is always active 7

1.4 Nanotemplates must meet all requirements at once. 8

1.5 Illustration of a carbon nanotube provided by *Elmar Behrmann.* Each vertex of each hexagon represents a carbon atom with three covalent bindings to other carbon atoms in the lattice. 10

1.6 Requirements and resolution chart of 3D EM methods suitable for the analysis of tubular structures. While tomography only requires Cryo samples, Sub-Tomogram Averaging can combine the information of multiple sub-volumes of the reconstructed tomogram. IHRSR (**I**terative **H**elical **R**eal **S**pace **R**econstruction) is the most complex analysis method gaining the best resolution for filaments. But it requires known helical parameters guessed or obtained from the other methods. 11

1.7 Visualization of Electron Tomography: Micrographs from various tilt angles are backprojected to a 3D volume. BAR domain illustration taken from [26]. 12

1.8 Visualization of Sub-Tomogram Averaging: Sub-volumes of the tomogram in figure 1.7 are averaged and refined to a molecular 3D model. 15

1.9 Helical parameters shown in a virtual tube. Each black dot represents a unit of the helix[46]. Δz is the axial subunit translation and $\Delta\varphi$ the azimuthal rotation per subunit. 17

1.10 Visualization of IHRSR: 2D projections from helical particles are classified and averaged to reconstruct the helix. From the helix an atomic 3D model of the protein can be obtained. 18

1.11 Orthonormal basis of CCD: Each red dot represents a coordinate point of the CCD. 20

1.12 Estimated gradient of CCD in chapter 1.6. 21

3.1 Transmission electron microscopes used for this work. 27

3.2 Lipid extrusion method. 28

3.3 Preparation process of Folch Lipid Tubes and its variables. 29

3.4 Illustration of tube counting method. Scale bar: 2 µm. The green colored tubes were counted. The red ones were ignored because of their size or one invisible end. 35

3.5 IMOD-PREPARE-STACK reads the .MDOC file and uses it to restore the correct angle order of the frame stack. It also writes a new, sorted .MDOC file. 38

3.6 Tubes were selected in the GUI **(a)** of *e2helixboxer.py*. While **(b)** shows a usable tube **(c)** was discarded. 40

3.7 A tube micrograph **(a)** is averaged to a 1D signal **(b)** and the tube ends detected in the graph. **(c)** shows a graph illustration with identified inner and outer diameters. 41

3.8 LineAdd.py averages all the pixel lines of the micrograph and plots the 1D density graph. 42

3.9 Illustration of the low-pass filter processing in LineAdd.py. The raw signal $l[x]$ **(a)** is low-pass filtered to $f[x]$ **(b)** using the filter $H(s)$ in **(c)**. 43

3.10 **(a)** is the positive offset from the mean of the filtered function. The *blue* dots are at the maxima m_1 and m_2 and the *green* colored line is the center of the graph. In **(b)** the *blue* graph is the variance function $c[x]$ and the *red* graph is its normed running mean function $r_s[x]$. The highest value for x where both graphs meet is marked with the *black* dot. 45

3.11 Tube ends detected in the 1D density graph (*cyan*: raw, red: low-pass filtered): *Red* dots are the approximated outer ends of the tube and *green* dots are the approximated inner ends of the tube. 47

4.1 Folch unilamellar vesicles prepared using the lipid extruder in Buffer *B*. All the objects with curved shapes in the micrographs are dried vesicles. The doubled lipid concentration in **(b)** resulted in 1.8 × higher vesicle concentration compared to **(a)**. Scale bars: 2 µm. . . 50

4.2 From one of the vesicles in the *green* boxes a 3.7 µm long tube (*red*) was extruded. Inside the *blue* close-up box, a circular vesicle with two extruded tubes to the left and to the right can be seen. Notably on these tubes a grainy coat is visible. At the *yellow* markers the diameter was measured: 1. 49 nm, 2. 77 nm, 3. 64 nm. Scale bar: 500 nm. 51

4.3 Slices of tomograms of tubes extruded by BPG from ULVs in negative stain **(a)** and Cryo **(b)**. The high contrast of the negative stain image allows to see *white* dots in a helical formation. 52

4.4 Helical parameters were obtained manually from a tomogram of a negative stained tube (section of figure 4.3a). The distances between the molecules in the *red* squares were measured in y- and z-direction and averaged. The *green* rectangle shows the diameter of the tube at the measured location. 53

4.5 A single Folch Lipid Nanotube (in the *blue* box) with a length of 1.6 µm and a diameter of 90 nm. The box emphasizes the linear shape of the tube. Scale bar: 1 µm. 54

4.6 Slice of a Cryo tomogram (40 images stacked) of BPG-coated LNT. Scale bar: 100 nm. The tube has a diameter of 75 nm and a length of more than 93 nm. The *red* stripes in the *blue* box have an angle of 67 ° to the *green* tube axis. 55

4.7 Comparison of the concentrations in Cryo with 12 $\frac{mg}{m\ell}$ in the same area. **(a)** is the original LNT protocol and **(b)** is the protocol including all optimizations. The concentration was 16 × higher in **(b)** after the optimization. Scale bars: 500 nm. 56

4.8 Comparison between buffers without chelating agents and buffers containing 1 mM EDTA and 1 mM EGTA. Omitting EDTA in **(a)** resulted in 7 × more tubes and 1.5 × more vesicles compared to the original buffer in **(b)**. The sample in **(c)** containing EGTA yielded no tubes but 3 × more vesicles. The mean diameter stayed constant at 229 ± 24 nm but the standard deviation was increased from 51 nm to 173 nm with more vesicles. Scale bars: 2 µm. 58

4.9 Comparison between LNTs with buffers containing $MgCl_2$ and $CaCl_2$. **(a)** $MgCl_2$ in the sample results in no tubes but 19 vesicles. **(b)** In contrast, the sample with $CaCl_2$ contains only 14 vesicles but 24 LNTs. This shows that $CaCl_2$ favors tube extrusion. Scale bars: 2 µm. 59

4.10 Visual illustration of LNT optimization CCD. 61

4.11 Correlation between variables and tube count. In case of **(a)** n_{FT} and **(c)** n_{Ex} the tube count increases with higher values. Only in case of **(b)** (T_{Ex}) there is a decreased tube count for 25 °C. 63

4.12 Detailed dependency on extrusion temperature: There is a slight increase at 25 °C but the error bars show big uncertainty factors. . . 64

4.13 **(a)-(e):** [TUBES : NON-TUBULAR OBJECTS]. Stability of LNTs at room temperature over a maximum of eight days. The number of tubes increases ($^2/_3$ ×) with time and the number of non-tubular objects decreases to (0.4 ×). The only exception is the sample taken from the bottom at the 9th day. Scale bars: 2 µm. 65

4.14 Stability graph of LNTs. The number of non-tubular objects decreases with time while the tube count increases. The difference between the 5th day and the 9th day is negligible. The sample taken from the bottom of the Eppendorf tube on the 9th day was not included in the graph. 66

4.15 Tubes recorded as part of a dataset. Scale bars: 40 nm. In **(a)** vesicles are marked in *red* squares. 67

4.16 In the *black* marked close-up view in **(b)** the helical coat is visible. Scale bars: 40 nm. 68

4.17 Raw micrographs and the output of LineAdd.py. The raw density graphs are *cyan*-colored. In contrast, the (*red*) low-pass filtered graphs are cleaner because they lack high frequency information that can distort the tube end detection algorithm. The difference in defocus between both raw micrographs is visible in the high frequency content of the raw density graphs. Scale bars: 20 nm. 69

4.18 Statistics generated for the dataset recorded in May 2018. The *gray* highlighted classes in **(a)** were used for helical reconstruction. The measured inner and outer diameters in **(b)** were sorted separately. . 70

4.19 Border thickness plotted against the outer tube diameter. This graph visualizes the distribution of the number of multilayers compared to the total diameter of the tube. It helps to find an outer diameter interval with similar multilayer numbers. A dense interval from 100 to 150 Å was selected. 71

4.20 Examples of different 2D classification runs of IHRSR of 1986 extracted particles after 25 iterations with tubes sorted by LineAdd.py highlighted in figure 4.6.1. In **(a)** the mask diameter was set to 410 Å. This covers only the inner tube diameter and cuts off most the borders. The stripes of the helical oligomer are visible. **(b)** is the result of 1100 Å mask diameter and covers more than the complete tube diameter. The class view is blurred because of overfitting. Particles with a different border thickness were averaged. The helical coat is washed out. In **(c)** the sampling was optimized. This class was the best one obtained after various sampling experiments. However, this is only the least dense class. 74

4.21 Two of ten classes of the slice marked in figure 4.19 after 2D classification. Notably in the shown classes the diameter between **(a)** (747 Å) and **(b)** (953 Å) is too much for a suitable 2D classification because tubes were again classified by their number of multilamellar borders instead of their helical coat. Moreover, with big differences in diameter the geometry between the tubes is also not comparable due to different curvatures. 75

4.22 Three of eight classes after using RELION for a diameter classification. **(a)** is the densest class and blurred by overfitting with a diameter of 996 Å. **(b)** contains thinner tubes and results in a 931 Å class diameter. **(c)** is less blurred than the other classes but contains with 17 particles far less than the other classes (327 and 65). It contains not enough particles to make the stripes of the helical coat visible. 76

5.1 Dimers of the BAR domain protein form oligomers and extrude tubes from unilamellar vesicles. Phosphorylation at the tips of the dimers inhibit oligomerization to ensure that the oligomers get their characteristic shape[76]. 81

5.2 Glass vials showing the steps in lipid rehydration. **(a)** is a transparent buffer solution containing undissolved lipid aggregates. In **(b)** most lipids are dissolved but there remain some smaller almost invisible aggregates and the solution is already non-transparent. **(c)** does not contain an appreciable amount of undissolved lipids and is homogeneously non-transparently colored. Sonication of **(a)** finally results in **(c)**. 85

5.3 The green arrow is the gradient of the original CCD experiment. The red arrow is a more likely result for the gradient. 86

5.4 An example for a poor tube end detection. The low average contrast of the left border results in a wrong inner diameter approximation. 91

5.5 The raw tube micrograph in **(a)** is filtered in Fourier space and results in **(b)**. The tube borders of **(a)** are not present in **(b)** anymore because the responsible frequencies were cut in Fourier space in **(c)**. To show the amplitude it was normed ($f : \mathbb{C}^{n \times n} \rightarrow \mathbb{R}^{n \times n}$) as shown in equation 5.4. The centered black dots (highlighted in *red* for better view) are zeroed and contain no more information. . . 92

List of Tables

3.1 Substances used for lab experiments. 24

3.2 Used lipids. 24

3.3 Consumables used for lab experiments. 25

3.4 Instruments used for lab experiments. 26

3.5 Software used for data processing. 26

3.6 Lipid nanotube preparation. 30

3.7 BPG Experiment. 31

3.8 Plunge freezing parameters. 34

3.9 Cryo-EM data acquisition parameters. 37

3.10 Single particle acquisition parameters. 39

4.1 Diameter measurements of raw and BPG-coated LNTs. 55

4.2 Results for LNT Central Composite Design. 60

4.3 Correlation between variables and tube count. 62

4.4 Detailed dependency on extrusion temperature. 64

4.5 Detailed Dataset statistics. 71

4.6 Particle extraction parameters. 72

4.7 2D classification parameters. 73

A.1 Detailed CCD results. 114

A.2 Detailed CCD temperature results. 115

Appendix A

Central Composite Design

Table A.1: Detailed CCD results.

		#1	**#2**	**#3**	**#4**	**#5**	$\bar{\mathbf{n}}$	σ
$N = 1$	000	3	1	0	0	0	0.8	1.3
	012	4	4	2	7	6	4.6	1.9
	021	8	7	1	6	4	5.2	2.8
	102	11	14	23	11	6	13.0	6.3
	111	15	12	9	22	7	13.0	5.9
	120	6	2	8	0	1	3.4	3.4
	201	9	12	10	13	22	13.2	5.2
	210	1	1	0	2	3	1.4	1.1
	222	12	17	25	21	13	17.6	5.5
$N = 2$	000	0	0	1	2	1	0.8	0.8
	012	3	4	5	4	6	4.4	1.1
	021	5	7	6	5	2	5	1.9
	102	18	16	9	10	11	12.8	4.0
	111	9	23	9	9	7	11.4	6.5
	120	0	1	0	2	0	0.6	0.9
	201	7	11	9	9	7	8.6	1.7
	210	6	2	1	2	1	2.4	2.1
	222	25	25	25	27	25	25.4	0.9
$N = 3$	000	1	0	0	0	1	0.4	0.5
	012	2	6	3	4	1	3.2	1.9
	021	5	7	6	5	4	5.4	1.1
	102	6	9	10	5	8	7.6	2.1
	111	4	5	8	10	10	7.4	2.8
	120	0	0	1	1	0	0.4	0.5
	201	27	11	12	12	19	16.2	6.8
	210	16	8	0	4	1	5.8	6.5
	222	26	21	25	28	24	24.8	2.6

Table A.2: Detailed CCD temperature results.

			#1	**#2**	**#3**	**#4**	**#5**	$\bar{\mathbf{n}}$	σ
$N = 1$	202	5 °C	18	13	13	10	7	12.2	4.1
	$2\frac{1}{2}2$	15 °C	4	8	0	2	7	4.2	3.3
	212	25 °C	24	19	13	12	17	17.0	4.8
	$2\frac{3}{2}2$	35 °C	10	6	12	11	15	10.8	3.3
	222	45 °C	12	13	16	10	8	11.8	3.0
$N = 2$	202	5 °C	27	24	22	21	23	23.4	2.3
	$2\frac{1}{2}2$	15 °C	26	20	21	23	20	22	2.5
	212	25 °C	42	39	29	41	33	36.8	5.6
	$2\frac{3}{2}2$	35 °C	14	18	21	21	19	18.6	2.9
	222	45 °C	24	19	24	24	17	21.6	3.4
$N = 3$	202	5 °C	14	13	16	14	14	14.2	1.1
	$2\frac{1}{2}2$	15 °C	24	21	22	24	24	23.0	1.4
	212	25 °C	24	26	18	22	14	20.8	4.8
	$2\frac{3}{2}2$	35 °C	16	18	20	12	13	15.8	3.3
	222	45 °C	23	12	20	19	18	18.4	4.0

Appendix B

imod-prepare-stack

```cpp
/*
    imod_prepare_stack.cpp V1.0.2, 2018/02/06
    by Nikolai Krupp, E-Mail: mailto:nikolai.krupp@caesar.de
*/

#include <vector>
#include <string>
#include <cstdlib>
#include <fstream>
#include <iostream>
#include <dirent.h>
#include <cstring>
#include <cstdio>
#include <array>
#include <memory>
#include <stdexcept>

using namespace std;

struct TILTINFO {
    int num;
    string File;
    double Angle;
    vector<string> *MDOC_section;
};

const char ST[10] = ".st";
const char REV[10] = "--r";
const char MRC[10] = ".mrc";
const char TXT[10] = ".txt";
const char TLT[10] = ".tlt";
const char OLD[10] = "_OLD";
const char MDOC[10] = ".mdoc";
const char ZVALUE[10] = "ZValue = ";
const char NEWSTACK[23] = "newstack --fileinlist ";
const char TILTANGLE[10] = "TiltAngle";
const char NEWSTACKTILT[10] = " --tilt ";
const char IMAGEFILE[13] = "ImageFile";
const char EXTRACTTILTS[14] = "extracttilts ";

string Trim(string);
bool exec(string, string&);
void bubblesort(TILTINFO *, int);
void bubblesort(vector<string> *);
inline bool IsNumeric(string,int,void *);
std::vector<string> *Split(string,string,int&);

int main(int argc, char** argv) {
    cout << "---------------------------------------------------------------------------" << endl;
    cout << ".::          imod_prepare_stack V1.0.2 02/2018 Nikolai Krupp             ::." << endl;
```

```cpp
    cout << "---------------------------------------------------------------------------" << endl;
    cout << "=> Input MRC file name order has to be equal to the chronological order." << endl;
    cout << "=> Optional argument: ./imod_prepare_stack FILENAME.mrc OR --r (see manual)." << endl;
    cout << "=> Please make sure that the module 'imod' is loaded." << endl << endl;
    DIR *dir;
    dir = opendir("./");

    // Mode: imod_fix_xmas
    bool bMRC = false;
    bool bRev = false;
    string inputfile;
    if (argc==2) {
        inputfile = (string) argv[1];
        if (inputfile==REV) {
            bRev = true;
            cout << "INFO: --r: Reversion of MRC file order switched on." << endl << endl;
        }
        else bMRC = true;
    }

    // Search all files in this folder and keep only MRCs and one MDOC:
    int p;
    struct dirent *entry;
    string file, mdocfile = "";
    vector<string> *files = new vector<string>;
    while ((entry=readdir(dir))!=NULL) {
        file = entry->d_name; p = file.size();
        if (p<5) continue;
        if (file.substr(p-strlen(MRC), strlen(MRC))==MRC) files->push_back(file);
        else if (file.substr(p-strlen(MDOC), strlen(MDOC))==MDOC) mdocfile = file;
    }

    int i, n, sn;
    TILTINFO *tilt;
    vector<string> *s;
    string cmd, output, line;

    bool bNoData = false;
    ifstream mrc;
    if (!bMRC) { // Using files as input.
        n = files->size();
        if (n==0) {
            cout << "ERROR: No MRC files found in this folder!" << endl;
            return 0;
        }
```

```cpp
        else cout << n << " MRC files found in this folder." << endl;

        // Sort MRC files by name to restore their chronological order:
        tilt = new TILTINFO[n];
        if (n>1) bubblesort(files);
        if (!bRev) for (i=0;i<n;i++) tilt[i].File = files->at(i);
        else for (i=0;i<n;i++) tilt[i].File = files->at(n-i-1);
        files->clear();
    }
    else { // Using MRC stack as input.
        s = Split(inputfile, ".", sn);
        mrc.open(inputfile, std::ios::in);
        if (sn<2||!mrc.good()) {
            cout << "ERROR: '" << inputfile << "' is not a valid MRC stack!" << endl;
            mrc.close();
            return 0;
        }
        mrc.close();

        string tltfile_in = s->at(0) + TLT;
        cmd = EXTRACTTILTS + inputfile + " " + tltfile_in;
        cout << "Running extracttilts ( " << cmd << " )..." << endl;
        cout << "---------------------------------------------------------------------------" << endl;
        if (exec(cmd, output)) {
            cout << "ERROR: Failed to execute shell command!" << endl;
            return 0;
        }
        cout << output;
        cout << "---------------------------------------------------------------------------" << endl << endl;
        output = "";

        // Read TLT file:
        ifstream tlt_in;
        tlt_in.open(tltfile_in, std::ios::in);
        if (!tlt_in.good()) {
            cout << "WARNING: Could not extract angles from '" << inputfile << "' to '" << tltfile_in << "'! Is module 'imod' loaded?" << endl;
            if (mdocfile=="") {
                cout << "ERROR: No (MDOC) tilt angles available!" << endl;
                return 0;
            }
            bNoData = true;
        }

        if (!bNoData) {
            double value;
            vector<double> *angles = new vector<double>;
            while (getline(tlt_in, line)) {
```

```cpp
                if (!IsNumeric(Trim(line),1,&value)) {
                    cout << "WARNING (tilt angle is invalid): " << line << endl;
                    continue;
                }
                angles->push_back(value);
            }
            tlt_in.close();

            n = angles->size();
            tilt = new TILTINFO[n];
            for (i=0;i<n;i++) tilt[i].Angle = angles->at(i);
            angles->clear();
        }
    }

    vector<TILTINFO> *vectilt = new vector<TILTINFO>;
    vector<string> *MDOC_header = new vector<string>;
    if (mdocfile!="") {
        // Try to open .mdoc:
        ifstream mdoc;
        mdoc.open(mdocfile, std::ios::in);
        if (!mdoc.good()) {
            mdoc.close();
            cout << "ERROR: Could not access '" << mdocfile << "'!" << endl;
            return 0;
        }
        else cout << "Using file '" << mdocfile << "'..." << endl << endl;

        // Read MDOC file and extract tilt angles:
        bool bHeader = true;
        string TiltAngle = TILTANGLE;

        TILTINFO *Buffer;
        while (getline(mdoc,line)) {
            if (line.find(ZVALUE)!=std::string::npos) { // [ZValue = x]
                if (!bHeader) vectilt->push_back(*Buffer);

                bHeader = false;
                Buffer = new TILTINFO;
                Buffer->MDOC_section = new vector<string>;
                Buffer->MDOC_section->push_back(line);
            }
            else {
                if (bHeader) {
                    MDOC_header->push_back(line);
                    continue;
                }

                Buffer->MDOC_section->push_back(line);
                s = Split(line, "=", sn);
                if (sn>=2 && Trim(s->at(0))==TiltAngle) {
                    if (!IsNumeric(s->at(1),1,&((*Buffer).Angle))) {
                        Buffer->Angle = -361.;
                        cout << "WARNING (tilt angle is invalid): " << line << endl;
                    }
                }
            }
        }

        if (!bHeader) vectilt->push_back(*Buffer);
        else {
            cout << "ERROR: '" << inputfile << "' did not contain any usable sections!" << endl;
            return 0;
        }
        mdoc.close();

        // Combine data from MDOC and MRC file(s):
        if (bNoData) {
            n = vectilt->size();
            tilt = new TILTINFO[n];
            for (i=0;i<n;i++) tilt[i] = vectilt->at(i);
        }
        else {
            int cnt = vectilt->size();
            if (cnt>n) {
                if (!bMRC) cout << "WARNING: '" << mdocfile << "' contains more angles (" << cnt << ") than files found in this folder (" << n << ")! Excess angles will be ignored." << endl << endl;
                else cout << "WARNING: '" << mdocfile << "' contains more angles (" << cnt << ") than sections found in '" << inputfile << "' (" << n << ")! Excess angles will be ignored." << endl << endl;
            }
            else if (cnt<n) {
                if (!bMRC) cout << "WARNING: This directory contains more MRC files (" << n << ") than sections (" << cnt << ") in '" << mdocfile << "'. Excess files will be ignored." << endl << endl;
                else cout << "WARNING: '" << inputfile << "' contains more sections (" << n << ") than '" << mdocfile << "' (" << cnt << "). Excess sections will be ignored." << endl << endl;
                n = cnt;
            }
            else n = cnt;

            for (i=0;i<n;i++) {
                tilt[i].Angle = vectilt->at(i).Angle;
                tilt[i].MDOC_section = vectilt->at(i).MDOC_section;
            }
        }
        vectilt->clear();
    }

    // Sort sections by tilt angle:
```

```
    for (i=0;i<n;i++) tilt[i].num = i;
    bubblesort(&tilt[0], n);
    for (i=0;i<n;i++) {
        cout << tilt[i].Angle;
        if (!bMRC) cout << " => " << tilt[i].File;
        cout << endl;
    }

    // Prepare file names and NEWSTACK command:
    ofstream stack;
    s = Split((mdocfile!="") ? mdocfile : inputfile, ".", sn);
    string tltfile_out = s->at(0) + TLT;
    string stackfile = s->at(0) + TXT;
    string outputfile = s->at(0) + ST;
    string newstack = NEWSTACK + stackfile + " " + outputfile;
    string newmdocfile = s->at(0) + ST + MDOC;

    // Write sorted MDOC file:
    if (mdocfile!="") {
        // Rename original file:
        cmd = "mv " + mdocfile + " " + s->at(0) + OLD + MDOC;
        cout << endl << "=> " << cmd << endl;
        exec(cmd, output);
        cout << output;

        // Write new file:
        ofstream newmdoc;
        cout << "Writing new MDOC '" << newmdocfile << "' file with sorted angles..." << endl;
        newmdoc.open(newmdocfile, std::ios::out);
        if (!newmdoc.good()) {
            newmdoc.close();
            cout << "ERROR: Could not create new MDOC '" << newmdocfile << "' file!" << endl;
            return 0;
        }

        // Update filename of stack:
        int j;
        string imgFile = IMAGEFILE;
        for (i=0;i<MDOC_header->size();i++) {
            s = Split(MDOC_header->at(i), "=", sn);
            if (sn>=2 && Trim(s->at(0))==imgFile) {
                MDOC_header->at(i) = imgFile + " = " + outputfile;
                break;
            }
        }

        for (i=0;i<MDOC_header->size();i++) newmdoc << MDOC_header->at(i) << endl;
        for (i=0;i<n;i++) {
            newmdoc << "[" << ZVALUE << i << "]" << endl;
            for (j=1;j<tilt[i].MDOC_section->size();j++) newmdoc << tilt
```

```
              [i].MDOC_section->at(j) << endl;
        }
        MDOC_header->clear();
        newmdoc.close();
    }

    // Write new TLT file:
    ofstream tlt_out;
    tlt_out.open(tltfile_out, std::ios::out);
    if (!tlt_out.good()) {
        tlt_out.close();
        cout << "ERROR: Could not create TLT '" << tltfile_out << "' file!" << endl;
        return 0;
    }
    for (i=0;i<n;i++) tlt_out << tilt[i].Angle << endl;
    tlt_out.close();

    // Write TXT file as NEWSTACK input:
    stack.open(stackfile, std::ios::out);
    if (!stack.good()) {
        stack.close();
        cout << "ERROR: Couldn't create stack file '" << stackfile << "' for newstack!" << endl;
        return 0;
    }

    if (!bMRC) {
        // Write a file containing the found MRC files:
        stack << n << endl;
        for (i=0;i<n;i++) {
            stack << tilt[i].File << endl << "0" << endl;
            tilt[i].MDOC_section->clear();
        }
        newstack += NEWSTACKTILT + tltfile_out;
    }
    else {
        // Rename original file:
        s = Split(inputfile, ".", sn);
        string inputfile_old = s->at(0) + OLD + "." + s->at(1);
        cmd = "mv " + inputfile + " " + inputfile_old;
        cout << endl << "=> " << cmd << endl;
        exec(cmd, output);
        cout << output;

        // Write a file with sorted angles only:
        stack << "1" << endl << inputfile_old << endl;
        for (i=0;i<n;i++) {
            stack << tilt[i].num;
            if (i==n-1) break;
            stack << ",";
        }
        stack << endl;
```

```
    }
    stack.close();
    delete[] tilt;
    cout << endl << "Finished: Stack file '" << stackfile << "' was written sucessfully." << endl << endl;

    // Run NEWSTACK:
    cout << "Running newstack ( " << newstack << " )..." << endl;
    cout << "---------------------------------------------------------------------------" << endl;
    if (exec(newstack, output)) {
        cout << "ERROR: Failed to execute shell command!" << endl;
        return 0;
    }
    cout << output;
    cout << "---------------------------------------------------------------------------" << endl << endl;

    mrc.open(outputfile, std::ios::in);
    if (!mrc.good()) cout << "ERROR: Writing of output file '" << outputfile << "' failed! Is module 'imod' loaded?" << endl;
    else cout << "Finished. Output file '" << outputfile << "' was written sucessfully." << endl;
    mrc.close();

    return 0;
}

void bubblesort(vector<string> *args) {
    int i,j;
    string temp;
    for (i=1;i<args->size();i++)
    {
        for (j=0;j<(args->size()-1);j++) {
            if(args->at(j)>args->at(j+1)) {
                temp = args->at(j);
                args->at(j) = args->at(j+1);
                args->at(j+1) = temp;
            }
        }
    }
}

void bubblesort(TILTINFO *arr, int n) {
    int i, j;
    TILTINFO Buffer;
    for (i=0;i<n;i++) {
        for (j=0;j<n-1;j++) {
            if (arr[j].Angle>arr[j+1].Angle) {
                Buffer = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = Buffer;
            }
        }
    }
}

// Split a string by a delimiter:
std::vector<string> *Split(string Str, string Delimiter, int& n) {
    std::vector<string> *v = new std::vector<string>;
    int j, i=0;
    while (true) {
        j = Str.find(Delimiter, i);
        if (j==-1) break;
        v->push_back(Str.substr(i,j-i));
        i = j+Delimiter.size();
    }
    int l = Str.size()-i;
    if (l>0) v->push_back(Str.substr(Str.size()-l, l));
    n = v->size();
    return v;
}

// Check if a string contains a valid number:
inline bool IsNumeric(string num, int FP, void *n) {
    try {
        if (FP==0) *((int *)n) = atoi(Trim(num).c_str());
        else *((double *)n) = atof(Trim(num).c_str());
    }
    catch (...) {
        return false;
    }
    return true;
}

// Trim all spaces around a string:
string Trim(string Str) {
    int i,j;
    for (i=0;i<Str.length();i++) if (Str[i]!=' ') break;
    for (j=Str.length()-1;j>=0;j--) if (Str[j]!=' ') break;
    return (i>0||++j<Str.length()) ? Str.substr(i,j-i) : Str;
}

// Execute shell commands in Linux:
bool exec(string cmd, string& output) {
    array<char, 128> Buffer;
    shared_ptr<FILE> pipe(popen(cmd.c_str(), "r"), pclose);
    if (!pipe) return true;
    while (!feof(pipe.get())) {
        if (fgets(Buffer.data(), 128, pipe.get()) != nullptr) output +=
          Buffer.data();
    }
    return false;
}
```

Appendix C

LineAdd.py

```python
# Written by Nikolai Krupp, 26.11.2018
# LineAdd.py V2.4

import os
import sys
import glob
from sparx import *
from math import ceil
from shutil import move
from shutil import copyfile
from scipy.stats import norm

import numpy as np
import scipy.stats as stats
import scipy.signal as signal
import matplotlib.pyplot as plt

#                       .:: PREFERENCES ::.
#####################################################################
bMRC = True                     # Copy original MRC as well?        #
bCopy = True                    # Copy files?                       #
bGauss = True                   # Gaussian fit?                     #
bExport = True                  # Export files of interval?         #
bSafeOnly = True                # Consider wrong outer diameters?   #
d_interval = 20                 # Classification interval (A), int. #
bOuterSort = False              # Sort outer diameter?              #
bMarkDiameters = True           # Plot diameters?                   #
bClassifyDiagrams = True        # Move diagrams to /Data?           #
N = 2                           # Filter order                      #
lowpass_freq = 0.05             # Filter cutoff frequency           #
Data_folder = "Data"            # Classification Destination        #
LineAdd_folder = "./LineAdd/"   # Output folder                     #
Pixel_Size = 1.07               # Calculate real distances  (A/px)  #
Gauss_std_min = 0.01            # Gaussian fit min std              #
Gauss_std_max = 1.0             # Gaussian fit max std              #
Gauss_std_step = 0.01           # Gaussian fit std step             #
Export_min = 100                # Min. border thickness             #
Export_max = 150                # Max. border thickness             #
Export_folder = "/Export/"      # Folder for export files           #
#####################################################################

cnt = 0
img = EMData()
print "Welcome to LineAdd.py Version V2.4, by NK"
print "Low pass filter cutoff frequency:", lowpass_freq
print "Pixel size is: " + str(Pixel_Size) + " A/px"
print
  "========================================================================="

try:
    os.makedirs(LineAdd_folder)
except OSError:
    if not os.path.isdir(LineAdd_folder):
```

```python
        print "ERROR: Could not create folder LineAdd. Aborting!"
            sys.exit(0)

fobj_out = open(LineAdd_folder + "Tube_statistics.txt","w")
B, A = signal.butter(N, lowpass_freq, output='ba')

# Plot filter:
w, h = signal.freqs(B, A)
plt.plot(w, 20 * np.log10(abs(h)))
plt.xscale('log')
plt.title('Low-pass filter frequency response')
plt.xlabel('Frequency [rad/s]')
plt.ylabel('Amplitude [dB]')
plt.rcParams.update({'font.size': 18})
plt.tick_params(axis='both', which='major', labelsize=12)
plt.tick_params(axis='both', which='minor', labelsize=10)
plt.margins(0, 0.1)
plt.grid(which='both', axis='both')
plt.axvline(lowpass_freq, color='green')
plt.savefig(LineAdd_folder + "Filter.png")
plt.clf()

d_wrong = []        # Wrong (inner) diameters
d_inside = []       # Inner diameters
d_outside = []      # Outer diameters
Filenames = []      # Filenames
Diagrams = []       # Diagram filenames

Data_folder = LineAdd_folder + Data_folder

for file in glob.glob("*.hdf"):
    print file

    # Prepare buffers:
    img.read_image(file, 0)
    nx = img.get_xsize()
    ny = img.get_ysize()
    targetimg = model_blank(nx,1,1,0.0)

    # Sum up all lines to a single averaged line:
    for i in xrange(ny):
        tmp = Util.window(img,nx,1,1,0,i-ny//2,0)
        targetimg = targetimg + tmp
    fintarget = targetimg/float(ny)

    filename = file[:len(file)-4] + "_Line"
    newfile = LineAdd_folder + filename + ".hdf"
    print newfile
    fintarget.write_image(newfile)

    # Copy the averaged gray values to a normal array:
    img_vector = []
    for i in xrange(nx):
```

```python
        img_vector.append(fintarget.get_value_at(i, 0, 0))

    # Low pass filter:
    npvector = np.array(img_vector)
    npvector = npvector.astype(np.float)
    img_filt = signal.filtfilt(B, A, npvector)

    # Plot unfiltered and filtered gray values:
    plt.ylabel('Mean gray value [au]')
    plt.xlabel('X-width [px]')
    plt.title(file)
    plt.plot(npvector, 'c')
    plt.plot(img_filt, 'r')
    plt.rcParams.update({'font.size': 18})
    plt.tick_params(axis='both', which='major', labelsize=12)
    plt.tick_params(axis='both', which='minor', labelsize=10)
    del img_vector[:]
    np.delete(npvector, 0)

    # Calculate the global mean of the filtered array:
    gl_mean = 0.0
    for i in xrange(nx):
        gl_mean += img_filt[i]
    gl_mean /= nx
    print "Global average:", gl_mean

    a1 = 0
    a2 = 0
    b1 = nx//2-1
    b2 = nx//2-1
    min1 = img_filt[0]
    min2 = img_filt[nx//2-1]
    max1 = min1
    max2 = min2

    # Calculate left and right minima and maxima:
    for i in xrange(nx//2):
        val1 = img_filt[i]
        if val1 < min1:
            min1 = val1
            a1 = i
        if val1 > max1:
            max1 = val1
            a2 = i
        val2 = img_filt[nx-i-1]
        if val2 < min2:
            min2 = val2
            b1 = nx-i-1
        if val2 > max2:
            max2 = val2
            b2 = nx-i-1

    print "1st maximum:", a2
```

```python
    print "2nd maximum:", b2

    # Consider only maxima, minima should be zero:
    hw_image = []
    for i in xrange(nx):
        v_img = img_filt[i]-gl_mean
        if v_img>0:
            hw_image.append(v_img)
        else:
            hw_image.append(0.0)

    # Outer diameter:
    ################################################################

    # front:
    # --------------------------------------------------------------

    mean = 0.0
    var_arr = []
    mean_arr = []
    var_max = ((hw_image[a2]-hw_image[a2-1])**2)**0.5
    mean_max = var_max

    for i in xrange(a2):
        v_img = ((hw_image[a2-i]-hw_image[a2-i-1])**2)**0.5
        var_arr.append(v_img)
        if v_img > var_max:
            var_max = v_img
        mean += v_img
        if mean > mean_max:
            mean_max = mean
        mean_arr.append(mean)

    factor = var_max / mean_max
    for i in xrange(a2):
        mean_arr[i] *= factor

    o_end1 = -1
    for i in xrange(a2):
        if mean_arr[a2-i-1] <= var_arr[a2-i-1]:
            o_end1 = a2-i-1
            break

    # back:
    # --------------------------------------------------------------

    mean = 0.0
    var_max = ((hw_image[b2-1]-hw_image[b2])**2)**0.5
    mean_max = var_max
    del var_arr[:], mean_arr[:]

    for i in xrange(nx-b2):
        v_img = ((hw_image[b2+i-1]-hw_image[b2+i])**2)**0.5
        var_arr.append(v_img)
        if v_img > var_max:
            var_max = v_img
        mean += v_img
        if mean > mean_max:
            mean_max = mean
        mean_arr.append(mean)

    factor = var_max / mean_max
    for i in xrange(nx-b2):
        mean_arr[i] *= factor

    o_end2 = -1
    for i in xrange(nx-b2):
        if mean_arr[nx-b2-i-1] <= var_arr[nx-b2-i-1]:
            o_end2 = nx-b2-i-1
            break

    # --------------------------------------------------------------
    # Outer failure detection:
    oWarn = False
    outer_end1 = a2
    outer_end2 = b2

    if o_end1 != -1:
        outer_end1 = a2 - o_end1
    else:
        oWarn = True

    if o_end2 != -1:
        outer_end2 = b2 + o_end2
    else:
        oWarn = True

    if (outer_end1 < 0.5*(a2-outer_end1)) and not oWarn:
        oWarn = True

    if (nx-outer_end2 < 0.5*(outer_end2-b2)) and not oWarn:
        oWarn = True

    if oWarn:
        print "Warning: Outer tube detection failed (" + str(o_end1) + ", " +
          str(o_end2) + ")!"

    print "1st outer End:", outer_end1
    print "2nd outer End:", outer_end2

    # Inner diameter:
    ################################################################

    # front:
    # --------------------------------------------------------------
```

```
    mean = 0.0
    var_max = ((hw_image[a2]-hw_image[a2+1])**2)**0.5
    mean_max = var_max
    del var_arr[:], mean_arr[:]

    n = nx//2-a2

    for i in xrange(n):
        v_img = ((hw_image[a2+i]-hw_image[a2+i+1])**2)**0.5
        var_arr.append(v_img)
        if v_img > var_max:
            var_max = v_img
        mean += v_img
        if mean > mean_max:
            mean_max = mean
        mean_arr.append(mean)

    factor = var_max / mean_max
    for i in xrange(n):
        mean_arr[i] *= factor

    i_end1 = -1
    for i in xrange(n):
        if mean_arr[n-i-1] <= var_arr[n-i-1]:
            i_end1 = n-i-1
            break

    # back:
    # --------------------------------------------------------------

    mean = 0.0
    var_max = ((hw_image[b2]-hw_image[b2-1])**2)**0.5
    mean_max = var_max
    del var_arr[:], mean_arr[:]

    n = b2-nx//2

    for i in xrange(n):
        v_img = ((hw_image[b2-i]-hw_image[b2-i-1])**2)**0.5
        var_arr.append(v_img)
        if v_img > var_max:
            var_max = v_img
        mean += v_img
        if mean > mean_max:
            mean_max = mean
        mean_arr.append(mean)

    factor = var_max / mean_max
    for i in xrange(n):
        mean_arr[i] *= factor

    i_end2 = -1
    for i in xrange(n):
```

```
        if mean_arr[n-i-1] <= var_arr[n-i-1]:
            i_end2 = n-i-1
            break

    # --------------------------------------------------------------
    del var_arr[:], mean_arr[:]
    # Inner failure detection:
    iWarn = False
    inner_end1 = a2
    inner_end2 = b2

    if i_end1 != -1:
        inner_end1 = a2 + i_end1
    else:
        iWarn = True

    if i_end2 != -1:
        inner_end2 = b2 - i_end2
    else:
        iWarn = True

    if (nx//2-inner_end1 < 0.5*(inner_end1-a2)) and not iWarn:
        iWarn = True

    if (inner_end2-nx//2 < 0.5*(b2-inner_end2)) and not iWarn:
        iWarn = True

    if iWarn:
        print "Warning: Inner tube detection failed (" + str(i_end1) + ", " +
          str(i_end2) + ")!"

    print "1st inner End:", inner_end1
    print "2nd inner End:", inner_end2

    ################################################################
    hw_image[:]
    np.delete(img_filt[:],0)

    if bMarkDiameters:
        plt.plot(outer_end1, gl_mean, 'ro')
        plt.plot(outer_end2, gl_mean, 'ro')
        plt.plot(inner_end1, gl_mean, 'go')
        plt.plot(inner_end2, gl_mean, 'go')

    Diagrams.append(filename + ".png")
    plt.savefig(LineAdd_folder + Diagrams[cnt])
    plt.clf()

    outer_diameter = (outer_end2-outer_end1) * Pixel_Size
    inner_diameter = (inner_end2-inner_end1) * Pixel_Size

    print "Diameter: " + str(outer_diameter) + " A, Inner diameter: " + str
      (inner_diameter) + " A"
```

```
    print
      "-------------------------------------------------------------------------
      "

    Filenames.append(file)
    d_outside.append(int(ceil(outer_diameter)))
    d_inside.append(int(ceil(inner_diameter)))

    if bSafeOnly and oWarn:
        d_wrong.append(True)
    else:
        d_wrong.append(iWarn)

    fobj_out.write(str(cnt) + "    " + str(outer_diameter) + "    " + str
      (inner_diameter) + "    " + filename + "    " + str(outer_end1) + "    " +
       str(outer_end2) + "    " + str(inner_end1) + "    " + str(inner_end2))

    if oWarn:
        fobj_out.write("    *")
    if iWarn:
        fobj_out.write("    *")

    fobj_out.write("\n")

    cnt+=1
    #if cnt == 10:
    #   break

if cnt == 0:
    print "No files found."
    sys.exit(0)

# Statistics:
c_good = 0
d_in_avg = 0.0
d_in_std = 0.0
d_out_avg = 0.0
d_out_std = 0.0

for i in xrange(cnt):
    if not d_wrong[i]:
        c_good += 1
        d_in_avg += d_inside[i]
    else:
        if bSafeOnly:
            continue
    d_out_avg += d_outside[i]

d_in_avg /= c_good
if bSafeOnly:
    d_out_avg /= c_good
else:
    d_out_avg /= cnt
```

```
for i in xrange(cnt):
    if not d_wrong[i]:
        d_in_std += (d_inside[i] - d_in_avg)**2
    else:
        if bSafeOnly:
            continue
    d_out_std += (d_outside[i] - d_out_avg)**2

d_in_std /= c_good
if bSafeOnly:
    d_out_std /= c_good
else:
    d_out_std /= cnt

d_in_std = d_in_std**0.5
d_out_std = d_out_std**0.5

print "Mean inner diameter: " + str(d_in_avg) + " (+/- " + str(d_in_std)
print "Mean outer diameter: " + str(d_out_avg) + " (+/- " + str(d_out_std)

# Sort ALL arrays by inner diameter:
iBuffer = 0
fBuffer = 0.0
sBuffer = ""
for i in xrange(cnt):
    for j in xrange(cnt-1):
        if d_inside[j]>d_inside[j+1]:
            fBuffer = d_inside[j]
            d_inside[j] = d_inside[j+1]
            d_inside[j+1] = fBuffer
            fBuffer = d_outside[j]
            d_outside[j] = d_outside[j+1]
            d_outside[j+1] = fBuffer
            iBuffer = d_wrong[j]
            d_wrong[j] = d_wrong[j+1]
            d_wrong[j+1] = iBuffer
            sBuffer = Diagrams[j]
            Diagrams[j] = Diagrams[j+1]
            Diagrams[j+1] = sBuffer
            sBuffer = Filenames[j]
            Filenames[j] = Filenames[j+1]
            Filenames[j+1] = sBuffer

mrc_file = []
box_file = []
box_name = []
if bMRC:
    # Find belonging MRCs and boxes:
    for i in xrange(cnt):
        Split = Filenames[i].split('_')
        mrc_file.append(Split[0] + "_" + Split[1] + ".mrc")
        box_file.append(Split[0] + "_" + Split[1] + "_boxes.txt")
```

```
        box_name.append(Split[0] + "_" + Split[1] + ".box")

# Clean array from wrong detected diameters:
c = 0
d_in = []
d_out = []
d_diff = []
d_legacy = []
for i in xrange(cnt):
    if d_wrong[i]:
        continue
    d_legacy.append(i)
    d_in.append(d_inside[i])
    d_out.append(d_outside[i])
    d_diff.append((d_outside[i]-d_inside[i])/2)
    c += 1

# Sort outer diameter by outer diameter:
if bOuterSort:
    for i in xrange(c):
        for j in xrange(c-1):
            if d_out[j]>d_out[j+1]:
                fBuffer = d_out[j]
                d_out[j] = d_out[j+1]
                d_out[j+1] = fBuffer

# Plot sorted and cleaned diameter statistics:
plt.ylabel('Diameter [A]')
plt.title('Tube diameter statistics [' + str(c) + ']')
plt.axhline(d_in_avg, color='green', linewidth=2.0)
plt.axhline(d_in_avg+d_in_std, color='green', linewidth=0.5)
plt.axhline(d_in_avg-d_in_std, color='green', linewidth=0.5)
plt.axhline(d_out_avg, color='red', linewidth=2.0)
plt.axhline(d_out_avg+d_out_std, color='red', linewidth=0.5)
plt.axhline(d_out_avg-d_out_std, color='red', linewidth=0.5)
plt.plot(d_out, 'r.')
plt.plot(d_in, 'g.')
plt.gcf().set_size_inches(30, 15)
plt.rcParams.update({'font.size': 42})
plt.tick_params(axis='both', which='major', labelsize=20)
plt.tick_params(axis='both', which='minor', labelsize=18)
plt.savefig(LineAdd_folder + "Tube_statistics.png")
plt.clf()

try:
    os.makedirs(Data_folder)
except OSError:
    if not os.path.isdir(Data_folder):
        print "ERROR: Could not create folder " + Data_folder + ". Aborting!"
        sys.exit(0)

if bSafeOnly and not bOuterSort:
    export = Data_folder + Export_folder
```

```
    if bExport:
        print "Exporting files..."
        try:
            os.makedirs(export)
        except OSError:
            if not os.path.isdir(export):
                print "ERROR: Could not create folder " + export + ". Aborting!" ↩

                sys.exit(0)

    for i in xrange(c):
        if bExport:
            if d_diff[i] <= Export_max and d_diff[i] >= Export_min:
                print "Copying file " + Filenames[j] + " to " + export
                copyfile(Filenames[d_legacy[i]], export + "/" + Filenames       ↩
                  [d_legacy[i]])
                copyfile(LineAdd_folder + Diagrams[d_legacy[i]], export + "/" + ↩
                  Diagrams[d_legacy[i]])
                if bMRC:
                    copyfile(mrc_file[d_legacy[i]], export + "/" + mrc_file     ↩
                    [d_legacy[i]])
                    copyfile(box_file[d_legacy[i]], export + "/" + box_name     ↩
                    [d_legacy[i]])

    plt.xlabel('Border Thickness [A]')
    plt.ylabel('Outer Diameter [A]')
    plt.title('Outer diameter vs. Border [' + str(c) + ']')
    plt.plot(d_diff, d_out, 'k.')
    plt.gcf().set_size_inches(30, 15)
    plt.rcParams.update({'font.size': 42})
    plt.tick_params(axis='both', which='major', labelsize=20)
    plt.tick_params(axis='both', which='minor', labelsize=18)
    plt.savefig(LineAdd_folder + "Tube_statistics2.png")
    plt.clf()

del d_in[:], d_out[:], d_diff[:]

print "Diameter classification interval [A]:", d_interval

# Pick the first and the last correct inner diameters:
a = -1
b = -1
for i in xrange(cnt):
    if a==-1 and not d_wrong[i]:
        a = i
    if b==-1 and not d_wrong[cnt-i-1]:
        b = cnt-i-1
    if a!=-1 and b!=-1:
        break

if a==-1 or b==-1:
    print "No usable tubes detected. Aborting!"
```

```
    sys.exit(0)

# Calculate the number of classes:
d_range = d_inside[b]-d_inside[a]
d_count = d_range // d_interval + 1

# Fill classes and copy to class folders:
bSave = bCopy
d_class = []
str_d_class = []
Classification = []
for i in xrange(d_count):
    bSave = bCopy
    d_class.append(d_inside[a]+i*d_interval)
    str_d_class.append(str(d_class[i]) + "-\n" + str(d_class[i]+d_interval-1))
    class_folder = Data_folder + "/" + str(d_class[i])

    try:
        if bCopy:
            os.makedirs(class_folder)
    except OSError:
        if not os.path.isdir(class_folder):
            print "Error creating folder " + class_folder + "."
            bSave = False

    n = 0
    for j in xrange(cnt):
        if d_wrong[j]:
            continue
        if d_inside[j]>=d_class[i] and d_inside[j]<(d_inside[a]+(i+1)            ↩
          *d_interval):
            n+=1

            if bSave:
                print "Copying file " + Filenames[j] + " to " + class_folder
                copyfile(Filenames[j], class_folder + "/" + Filenames[j])
                if bClassifyDiagrams:
                    move(LineAdd_folder + Diagrams[j], class_folder + "/" +      ↩
                    Diagrams[j])
                if bMRC:
                    copyfile(mrc_file[i], class_folder + "/" + mrc_file[i])
                    copyfile(box_file[i], class_folder + "/" + box_name[i])

    Classification.append(n)

print                                                                            ↩
  "--------------------------------------------------------------------------"
del d_inside[:], d_outside[:]

# Now take care of all wrong detected diameters:
bSave = bCopy
err_folder = Data_folder + "/Incorrect"
try:
        if bCopy:
        os.makedirs(err_folder)
except OSError:
        if not os.path.isdir(err_folder):
        print "Error creating folder " + err_folder + "."
            bSave = False

for i in xrange(cnt):
    if d_wrong[i] and bSave:
        print "Copying file " + Filenames[i] + " to " + err_folder
        copyfile(Filenames[i], err_folder + "/" + Filenames[i])
        if bClassifyDiagrams:
            move(LineAdd_folder + Diagrams[i], err_folder + "/" + Diagrams[i])
        if bMRC:
            copyfile(mrc_file[i], err_folder + "/" + mrc_file[i])
            copyfile(box_file[i], err_folder + "/" + box_name[i])

if bCopy:
    print "    Done."

# Plot the histogram of tube classes:
plt.ylabel('Tube Count')
plt.xlabel('Diameter class [A]')
if not bGauss:
    plt.title('Classification of inner diameters')
plt.bar(d_class, Classification, align='center')

# Add a gaussian distribution and shift it to the class with the highest
  density:
if bGauss:
    c_max = 0
    c_pos = 0
    d_class_ext = []

    for i in xrange(d_count):
        if Classification[i]>c_max:
            c_max = Classification[i]
            c_pos = i

    if d_count-c_pos>c_pos:
        h = (d_count-c_pos)
    else:
        h = c_pos

    c_ext = 2*h

    for i in xrange(h):
        d_class_ext.append(d_class[c_pos]-(h-i)*d_interval)

    c_mid = h+1
    d_class_ext.append(d_class[c_pos])
    for i in xrange(h):
        d_class_ext.append(d_class[c_pos]+(i+1)*d_interval)
```

```
    n = int((Gauss_std_max-Gauss_std_min) / Gauss_std_step) + 1
    d_offset = c_mid-c_pos

    c_mu, c_std = norm.fit(Classification)
    print "Classification mean: " + str(c_mu) + " std: " + str(c_std)
    mu, std = norm.fit(d_class_ext)
    print "Mean value of ext. vector: " + str(mu) + " std: " + str(std)

    fit = []
    std_factor = []
    mean_deviation = []

    for i in xrange(n):
        std_factor.append(Gauss_std_min+i*Gauss_std_step)
        fit.append(stats.norm.pdf(d_class_ext, mu, std_factor[i]*std))

        # Norm gauss to Classification vector:
        f_max = fit[i][0]
        for j in xrange(c_ext):
            if fit[i][j]>f_max:
                f_max = fit[i][j]
        factor = c_max / f_max

        # Calculate the mean deviation of the fit:
        mdev = 0.0
        for j in xrange(d_count):
            fit[i][d_offset+j-1] *= factor
            mdev += ((Classification[j] - fit[i][d_offset+j-1])**2)**0.5
        mean_deviation.append(mdev / d_count)
        print "Mean deviation at std. factor " + str(std_factor[i]) + ": " + str
          (mean_deviation[i])

    # Search for the fit with the smallest mean deviation:
    best_fit = 0
    min_dev = mean_deviation[0]
    for i in xrange(n):
        if mean_deviation[i] < min_dev:
            min_dev = mean_deviation[i]
            best_fit = i

    # Cut gaussian distribution vector to fit in diagram:
    fit_small = []
    for i in xrange(d_count):
        fit_small.append(fit[best_fit][d_offset+i-1])

    plt.title("Classification of inner diameters [" + "Gaussian fit: " + str
      (std_factor[best_fit]) + " * std]")
    plt.plot(d_class, fit_small, 'r')
    del d_class_ext[:]
    np.delete(fit[:][:],0)

plt.xticks(d_class, str_d_class)
```

```
plt.gcf().set_size_inches(30, 15)
plt.rcParams.update({'font.size': 42})
plt.tick_params(axis='both', which='major', labelsize=20)
plt.tick_params(axis='both', which='minor', labelsize=18)
plt.savefig("./LineAdd/Tube_histogram.png")
plt.show()

del Filenames[:], d_wrong[:], d_class[:], Classification[:], str_d_class[:],
  mean_deviation[:], std_factor[:]
fobj_out.close()
print
  "========================================================================"

# module load hwloc/1.11.3
# module load openmpi/gcc/64/3.0.1_no_dlopen
# module load EMAN2/2.21
# /cm/shared/apps/EMAN2-versions/EMAN2-dist-2.21/bin/python LineAdd.py
```

Danksagungen

Als erstes möchte ich mich bei meinem Doktorvater **Elmar Behrmann** bedanken, der das gesamte Projekt überhaupt ermöglicht hat. Er hat mich in jeder Situation gut betreut und mein Projekt in die richtige Richtung gelenkt. Er wurde nicht müde, mir sämtliche Fragen sowohl fachlicher als auch organisatorischer Natur zu beantworten.

Außerdem bedanke ich mich inbesondere bei **Barbara Hahn**. Sie hat mein Projekt hervorragend betreut und für mich viele organisatorische Fragen zur gesamten Promotion geklärt.

Besonderer Dank gilt **Stephan Irsen**, der mir bei der Aufnahme meiner Datensätze geholfen hat und mir bei vielen Fragen und Problemen, insbesondere zur Elektronenmikroskopie, beratend zur Seite stand. Er stand außerdem jederzeit zur Verfügung bei Problemen mit Geräten oder Laboren.

Elmar Behrmann, **Monika Gunkel** und **Ingrid Krupp** danke ich für das Korrekturlesen dieser Dissertation.

Bei der Planung und Durchführung vieler chemischer Experimente hat mich **Andreas Rennhack** unterstützt.

Meine Büro- und Gruppenkollegen **Alexandra Schneider, Antje Baumgartner, Christoph Klatt, Gayathri Jeyasankar, Magdalena Schacherl, Monika Gunkel, Patrick Meckelburg, Zhuoyan Anthony Chen** haben mich in jeder Situation unterstützt und mir geholfen. Sowohl fachlich als auch persönlich.

Inga Hochheiser möchte ich für die Hilfe beim Cryo-Screening und das Bereitstellen von Eis zur Lagerung von Proben und Stiften zur Beschriftung im Labor danken.

Ich möchte mich bei der Arbeitsgruppe EMA bedanken, die mich bei vielen Experimenten unterstützt und ihre Labore zur Verfügung gestellt hat. Insbesondere **Kristina Barragán Sanz** und **Carola Tröger** haben mir jederzeit im Labor und bei der Nutzung der Elektronenmikroskope geholfen.

Anja Günther und **Olivier Le Bihan** danke ich für die hilfreichen wissenschaftlichen Diskussionen.

Mathematische Fragen wurden mir von **Pedro Goncalves** beantwortet.

Zum Schluss möchte ich außerdem meiner Familie und meinen Freunden danken, die mich aufgemuntert und unterstützt haben.