Universität Heidelberg

Automatic Length and Angle Estimation of DNA on AFM Images

Dennis Aumiller, Lina Gundelwein, Philip Hausner, Philipp Jung,
Susanne Ibing, Sarah Schott, Christian Schütz, Roman Spilger,
Oskar Staufer, Martin Würtz

Projektseminar Biomedizinische Bildanalyse
Sommersemester 2016
19.04.2016 – 26.07.2016
PD Dr. Karl Rohr

Abstract

Atomic force microscopy (AFM) enables for exquisite spatial resolution providing users with topographic maps that resolve even single atoms, underlining this dynamometric probe scanning technique as an extremely valuable method to explore structural features of macromolecular complexes. Particularly the interlaced architecture of complexes involved within biological processes can be adequately probed by the AFM technology. Here, especially the parceling interactions between DNA molecules and molecular spools known as histones has intensively been studied by AFM imaging. These approaches have however been exceedingly limited by the lack of appropriate tools for automated evaluation of the molecular interplays within nucleosomes (DNA - Histone complexes), forcing users to perform time consuming manual data evaluation. Major bottlenecks challenging the development of software for effective automated evaluation were not only the discrimination between AFM typical prominent sprinkled artifact signals and nucleosome complexes but also the accurate determination of critical nucleosome features like histone binding strength, DNA entering/ exiting angles or DNA turns per histone. Therefor, we here report on our recent advances on the development of an algorithm designed for fast and automated nucleosome feature extraction (ANuFE algorithm) from AFM data. By applying adaptive low pass filtering for elimination of highly contrasted regions and non - local means denoising techniques, ANuFE allows for thoroughly deletion of artifact signals. Based on adaptive thresholding and 3 dimensional Hogh voting, ANuFE is furthermore able to discriminate between free DNA and nucleosome complexes, which allows for subsequent extraction and quantification of nucleosome features. The algorithm was designed not only to perform expeditious on frequently used TIFF image data but also to handle high variations in image quality, which are typical for AFM data. By applying ANuFE on test data displaying nucleosomes harboring either wild type or mutant histones, we were able to accurately identify significant differences in nucleosome features between the two classes, pointing to an impairment of nucleosome stability by the inserted mutations.

Contents

1 Organisation

1.1 Distribution of Tasks

In the following the task distribution is shown. If the section is mentioned, the subsections are composed by the same person, otherwise the subsections are listed separately.

Dennis Aumiller	length estimation (Intro and Method D)
Lina Gundelwein	team leader, filtering, thinning, nucleosome detection, angle measurements, assembling report Sections 3.7, 3.6, 3.9
Philip Hausner	
Philipp Jung	software architecture, performance optimization Section 3.2, 3.5.1, 3.5.2, 3.5.3, 3.5.4, 3.2, 3.10
Susanne Ibing	literature research, test data creation, validation, assembling presentation slides Section 2.5
Sarah Schott	literature research, test data creation, validation Section 2.5
Christian Schütz	OpenCV denoising, thresholding (investigation of Octave functions), length estimation (Intro and Method C), evaluation tools Section 3.3, 3.5, 3.8.2, 3.8.2.3, 3.8.2.4
Roman Spilger	literature research, test data creation, validation
Oskar Staufer	literature research, test data creation Sections 2.2, 2.3, 2.4
Martin Würtz	team leader, literature research, test data creation, validation, assembling presentation slides Section 3.1

1.2 Time Schedule

In the beginning of the project, an expected time line was generated and presented. Now after finishing the project, it is possible to compare the expected and actual needed time for each subsection of the project (see Figure 1. For most of the subsections, such as the project selection, team building, software

architecture, generation of test data, optimization of the algorithm, protocol writing and the generation of the endpresentation, the expected amount of time is similar to the actual amount of time. Those subsections are mostly not dependend upon other previous tasks. For the implementation of the algorithm however, four more weeks were necessary. Many of the images were hard to work with since the background is often very noisy and sometimes include dirt or undefined particles. Due to the fact that we did not have a limitless amount of images, we were dedicated to try to make the algorithm a very robust one. As visible in Figure 1, the filtering, denoising and thresholding of the images was very time consuming even though now very successful. For the next steps of the algorithm, a robust binary image was necessary which is why it took much longer than expected. The evaluation of the algorithms, test data and the biological results could only be performed after the successful establishing of the algorithm. Because of the delay in algorithm implementation, the evaluation took place as well four weeks later than expected. In general, Figure 1 shows that with the number of team members, effective, simultaneous working was possible during the project.

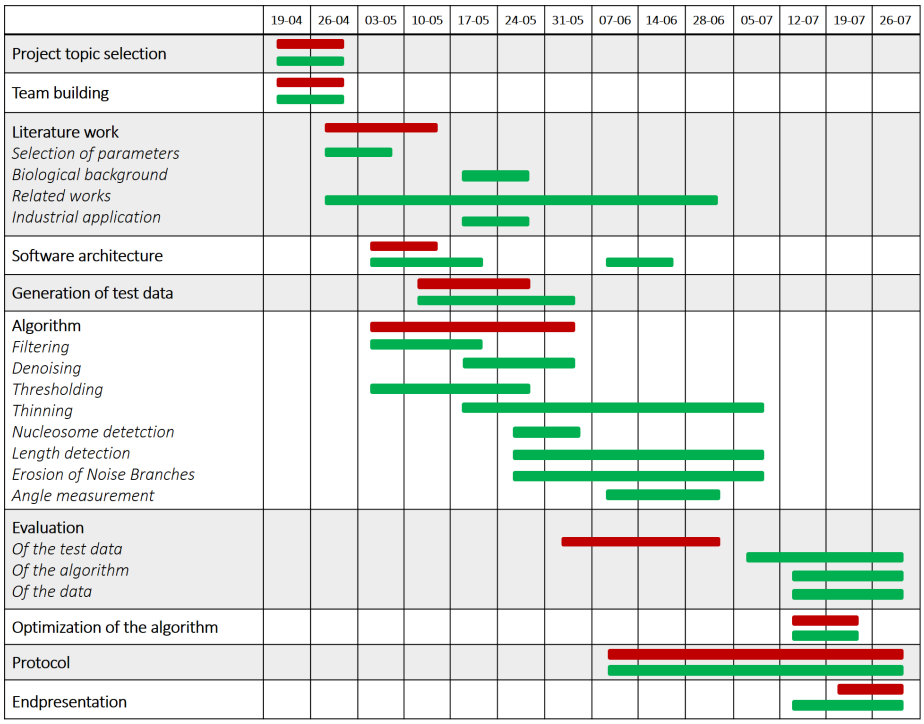


Figure 1: The expected needed amount of time for each subsection (red) and the actual needed amount of time (green).

2 Introduction

2.1 Motivation

DNA-histone interactions are of major importance for fundamental biological processes like epigenetic silencing or chromatin condensation and remodeling. However, probing nucleosome dynamics on a single molecule level has remained challenging and has until now mostly been tackled by studies relaying on in vivo FRET (Förster resonance energy transfer) or FCS (fluorescence correlations spectroscopy) [1] analysis or in vitro AFM topographic imaging, where intracellular dynamics with moderate resolution or more artificial environments on a high resolution can be measured respectively.

Here, several histones residues located within the DNA-docking domains have been identified as key players in DNA–histone interactions, as their replacement by neutral or negatively charged amino acids strongly weakens the electrostatic interaction strength perceivable in destabilization of nucleosome complexes (unpublished data).

High throughput analysis of nucleosome stability on single molecule scale could therefor provide greater insides into a variety of pathological processes involved in inflammatory diseases or cancer malignancy.

Even if AFM has been successfully applied to image DNA–histone interactions on high resolutions, tedious manual data analysis has yet hampered large scales analysis [2]. We therefor aimed to implement a fully automated algorithm to analysis nucleosome dynamics based on AFM data. Our ANuFe (Automated Nucleosome Feature Extraction) algorithm automatically extracts features like DNA length, DNA entering and exiting angles and nucleosome volume, based on single nucleosome resolved AFM data, thus enabling for nucleosome stability analysis on high throughputs.

In order to evolve the ANuFE, AFM images from the Langowski Lab of the DFKZ, which is interested in nucleosome dynamics, have been used. In detail the stability of wild type and H2A histone mutants (R81A, R88A, R81E and R88E) have been investigated. Thereby the mutations being located in the docking domain of H2A were proved to be good tools to analyze the stability and the opening behavior of nucleosomes on single molecule level. Single particle Förster resonance energy transfer (spFRET) studies already demonstrated that these mutants are significantly destabilized compared to wild-type nucleosomes (unpublished data).

For AFM investigations, the described H2A histone mutants were bound to a 660 bp DNA strand containing the 170 bp Widom 601 histone octamere positioning sequence [3] 199 bp apart from one end of the fragment. The DNA and the different nucleosomes were measured in air on a poly-L-lysine coated mica with different nm to pixel ratios.

ANuFe was designed to extract nucleosome features from AFM data of conventional quality (moderate to high background signals and varying nm to pixel ratios) and quantity (moderate to high nucleosome densities with intra-/ intermolecular DNA crossing) from frequently used TIFF data to address the

diversity of requirements in varying experimental setups.

2.2 AFM

Since its invention in 1982 by IBM scientists Gerd Binning and Heinrich Rohrer, atomic force microscopy (AFM) has found widespread applications in various fields ranging from semiconductor science to polymer physics. The simplicity of the underlying concept allows for implementation in a variety of challenging experimental setups including high-energy physics [4] and live cell biology [5]. The AFM technology has particularly been used not only to obtain high-resolution topographic images (with resolution of up to 1 nm) but also to measure extremely low forces that occur for example during molecular interactions. In this way, major achievements, like first topologies of single atoms and small molecules and their connecting electron bonds, have been made possible [6].As AFM is a non- destructive imaging technique and large fields of view can be scanned on appropriate time scales, it has found more and more application in the study of living cells and organisms (Figure 2). Here, it enables even for high resolution studies of the interactions between single cell membrane receptors and ligand drugs [7]. Moreover, as nanotechnological approaches are becoming increasingly popular, AFM has gained attention for quality control purposes in industry and academia.

However, automated evaluation of AFM images has remained challenging, especially when highly diverse biological structures, such as DNA molecules, are studied. Mostly because high background and varying morphologies hinder accurate automated evaluations. We here present an image processing algorithm, designed for automated quantification of AFM data to further study DNA histone interactions in higher throughput.

2.2.1 Concept

The heart of an AFM setup consists of a sharp silicon tip with an ending radius of curvature of up to 1 nm attached to a flexible micro cantilever (Figure 3).

The cantilever is mounted onto a Piezo controlled z-stage for precise lifting and dipping. With this, the cantilever tip is brought into (close) contact to the surface to be examined. The tip is then further scanned over the surface with constant z-stage deflection and thereby bended by topographic heterogeneities of the sample according to the mechanics described by Hook for simple springs. To precisely measure and amplify these minimal deflections, a focused laser beam is projected onto the cantilevers reflecting top. Redirected onto a four photodiode detector, x- and y- deflection of the elastic cantilever can be recorded as potential differences between pairing photodiodes. Small bendings are thereby amplified by the laser deflection and can be used to compute topographic images. Although being extremely simple, high resolutions can be achieved that are mostly limited by the tips radius and shape (Figure 4).

However, to preserve the integrity of the examined specimens, which is of particular importance when observing living systems where harsh perturbations

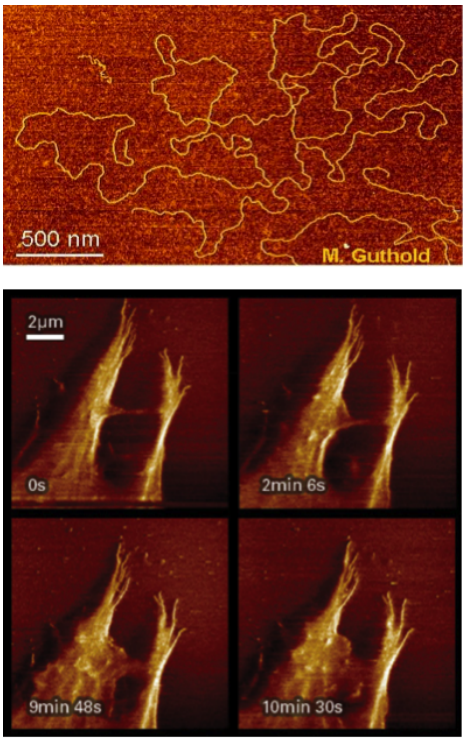


Figure 2: (Top) AFM image of a complete λ phage genome with single strand resolution. Colors represent cantilever tip deflection [8]. (Bottom) AFM image of a living cell showing filopodia rearrangement. Colors represent cantilever tip deflection [9].

are likely to cause artifacts, the AFM is run in a so called tapping mode [12]. Here, an alternating current is applied to the supporting Piezo element thus oscillating the cantilever over the surface. By this, the tip-surface interaction is minimized and sample can be scanned with less interference. The core setup has been further modified and extended to even measure intermolecular forces [13]. For this, the cantilever tip is slowly approached to a surface. At a specific distance, electrostatic forces will lead to an attraction of the tip and thus a bending of the cantilever. The kinetics of this bending are characteristic for differing molecular interactions [14]. By attaching molecules of interest to the AFM tip, specific interaction between these and molecules located at a surface can be measured. Using this concept, the interplay between drug molecules and G-protein-coupled receptors on cell surfaces could be quantified [15]. Furthermore, by using conductive tips, the electric properties of materials were studied, an approach that has found wide applications in semiconductor and microprocessor science [16].

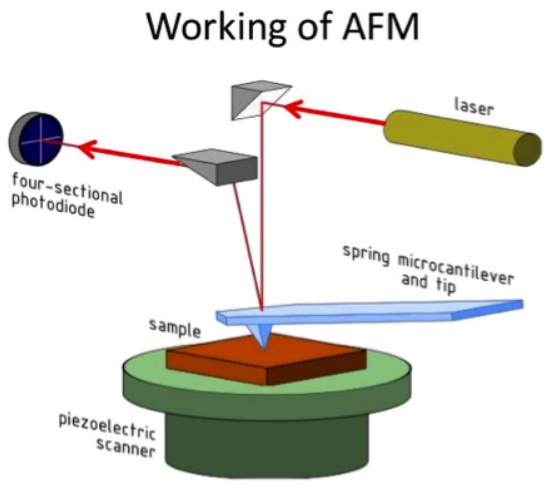


Figure 3: Schematic representation of an AFM setup with optical path of the laser beam in red and xy-controllable Piezo scanner [10].

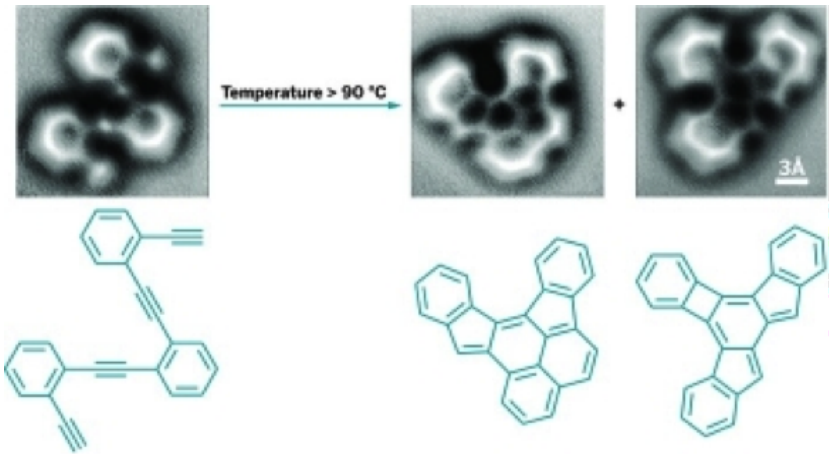


Figure 4: Single atom resolution AFM image and corresponding structural formula of an enediyne compound before and after induction of cyclisation by heating [11].

2.3 Epigenetics and Histones

Genetic information is mostly encoded within the DNA sequence and its modular components known as genes. However, during the last decades epigenetic mechanisms that regulate genetic information processing, have been recognized as key players in gene regulation. To date, several mechanisms of epigenetic regulation have been found in eukaryotic cells [17]. Two of the most prominent examples are DNA methylation and chromatin rearrangement. While methyla-

tion is a direct chemical modification of DNA that leads to impaired recognition by DNA interacting proteins [18], chromatin remodeling mechanisms are versatile [19, 20, 21]. Here, the accessibility of specific sequences which are crucial for DNA processing are altered. The central elements of chromatin structures are large heteromeric protein complexes known as histones [22]. These highly alkaline proteins are exclusive to eukaryotic cells and some archaea, where they act as spools around which DNA can bind. In this way, histones do not only alter DNA accessibility but also condense the genetic information within the nucleus, a critical steps especially during cell division.

2.3.1 Chromatin and Histone Structure

Histone complexes are formed between five major components, the histones proteins H1/H5, H2B, H2A, H3 and H4 [22]. The histone core is formed between H2A, H2B, H3 and H4 while H1/5 is known to serve as a linking element. The core histones exist as homodimers; all possessing a histone fold domain that is crucial for the interaction between the different dimers. This domain is comprised of three alpha helices that interact as handshake motifs with corresponding domains on the dimer partner. Thus, the histone complex is an octameric aggregate with an approximate diameter of 63 Å where 147 DNA base pairs can wrap around in 1,65-left handed turns. The histone protein H1 binds to the entering and exiting DNA strand thereby stabilizing the DNA histone complex. H1 is also crucial for the formation of higher order chromatin complexes as it mediates the arrangement of histone fibers in which several histone-DNA complexes pair to highly condensed chromatin (Figure 5).

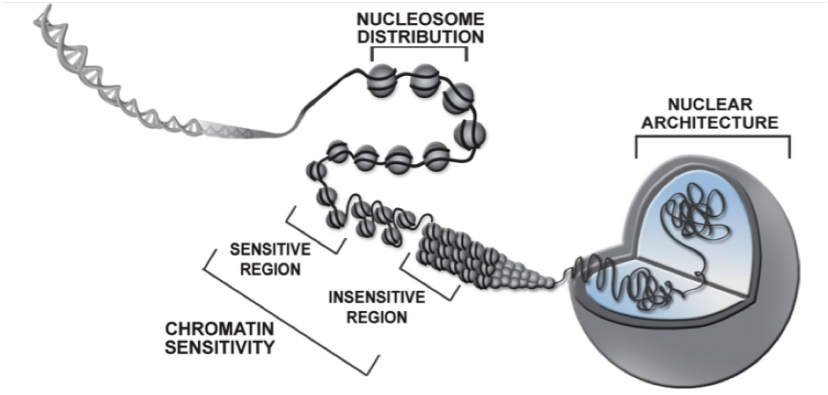


Figure 5: Chromatin structure from a single DNA double helix (left) to the fully evolved nuclear DNA architecture (right). DNA bound single histones are depicted in their modification sensitive conformation, where epigenetic mechanisms regulate gene transcription, and their insensitive condensed form [11].

2.3.2 Histone Modifications

Together with DNA methylation, chemical modifications of histone complexes are a key process in epigenetic regulation as with this, interactions between DNA and nuclear proteins such as transcription factors, polymerases or other regulatory elements can be altered [23]. As histones represent complex macromolecular structures, the possible chemical modifications are diverse and yet not fully understood. However, some key players have been identified. Most modifications occur at the tails of histone proteins H3 and H4, which protrude from the DNA-histone complex [24] (Figure 6).

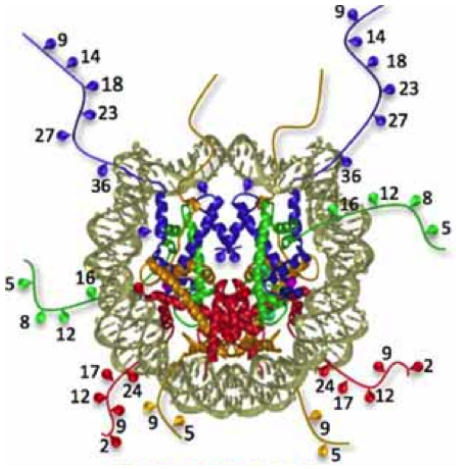


Figure 6: Crystal structure of a histone complex with a modeled double helical DNA strand wrapped around. Colors represent histone dimers (H2A yellow, H2B red, H3 blue, H4 green). Numbers correspond to amino acid sequence position and show frequent modification sites [25].

Here, acetylations, methylations, phosphorylations, ubiquitinations and citrullination have been reported. Additionally, the histone core can also be modified leading to highly complex modification patterns as several modification combinations can occur. By this, either the direct interaction between the DNA and a histone is sterically hampered or the modification acts as a recognition element for other regulatory proteins. For example, lysine acetylation in histone complexes is responsible for the loss of one positive charge and therefore reduces the electrostatic interaction strength to the negatively charged DNA backbone [26]. Therefore, acetylated histones are thought to form less condensed and thus more accessible chromatin structures. On the other hand, gene promoter regions that are bound to non-acetylated histones are known to be responsible for gene down regulation. Another prominent example for histone modification is lysine methylation. Here, up to three methyl groups can be added to a lysine residue at the histone tail. Although this does not diminish the positive lysine charge and thus no direct inhibition of the DNA-histone complex formation is

observed (even if the small methyl groups slightly sterically alter this interaction), it serves as a recognition motive for other nuclear responsive elements with Tudor or PHD domains [27]. The underlying mechanisms appear to be extremely sensitive, as opposite effect between mono- and demethylation have been reported [28]. Some well-studied histone modifications that promote gene transcription are triple methylation of H3 lysine 4, which mostly occurs in the promoter region of highly transcribed genes [29], and triple methylation of H3 lysine 36, which is frequently observed in the gene bodies of upregulated genes as it recruits histone deacetylase thereby ensuring proper gene transcription [30]. Prominent examples for modifications that repress gene transcription are trimethylation at lysine 27 of H3 [31], which induced histone acetylation, and trimethylation of H4 lysine 20 [27]. Hence, histone modification is a generally accepted epigenetic mechanism to regulate gene transcription, where the interaction strength between histones and DNA is of major importance. Mutations within histone complexes that alter this interaction have been associated with various diseases like cancer and chronic inflammations [32].

2.4 Histone Complexes by AFM

As DNA-histone configuration is of major importance for epigenetic regulation, a profound understanding of the regulatory mechanisms and crucial structural features that conquer this interaction is highly desirable and of special interest for drug design. Here, the AFM technology is preferentially suited to study the molecular configuration with sufficient resolution. AFM is not only able to easily resolve single DNA strands but also to image the volume of single DNA-histone complexes (and thereby the number of DNA turns per histone) and the entering/ exiting angle of bound DNA, which gives insides into the bonding strength between DNA and histone. An automated evaluation of DNA structure is not only of interest to study DNA-protein interaction but also to evaluate cancerogenic compounds that alter the DNA architecture. Here, correlations between the cancerogenicity and the geometrical deformation of the compound bound DNA have been observed [33]. Therefore, quantitative screening of pharmaceutical compounds for their ability to influence DNA structure with AFM resolution could be highly interesting for toxicity studies. Moreover, many modern technologies used within the life science sector are based on DNA molecules. For example DNA bound antibodies have been used for immunohistochemical stainings where a DNA bound fluorophore enhances the fluorescence signal [34]. Other applications include DNA micro arrays to study and compare gene expression [35]. Here, DNA molecules with specific gene complementary sequences are spot printed onto glass cover slips to quantify the amount of specific cDNA present in probes. This widely used technique can monitor slight changes in gene expression. However, quality control of the microarrays themselves has remained challenging. Specialized ventures already test for DNA microarray quality by AFM imaging but are yet not able to provide single DNA strand resolution as automated evaluation of this has been elusive [36]. With the here presented algorithm we tackle current restrictions in DNA-protein interaction

analysis and DNA morphology quantification by AFM.

2.5 Related Work

The analysis of AFM images can be categorized into manual, semi-automatic and automatic approaches. Since manual image analysis were human operators draw manually the backbone of DNA filaments is very time consuming and error-prone, we focused on a fully automatic analysis without any supervision. In semi-automatic image analysis approaches, the threshold had been set manually or the initial and/or final point of a DNA filament was set by a human operator [37], [38], [39].

Automatic image analysis predominantly focussed on the determination of the contour length of DNA filaments [40], [41], [42], [43]. The steps of the algorithms are very similar and will be explained below. The contour length is defined as the polymer's length at maximal physical extension [44]. Some research groups also considered the curvature or the spatial orientation of DNA filaments [45], [46]. Doyen et al. used an automated approach for nucleosome recognition according area and height criteria and free DNA length determination of the nucleosomes.

The main steps of DNA contour length determination are first the generation of a binary image, then the skeletonization of traced DNA fragments, and then the length measurement. Many papers are referring to additional optional steps which are not necessary depending on which algorithms were implemented for the thinning step. In the following sections the single steps of the algorithms are explained.

2.5.1 Filtering

In order to reduce the noise, one apply filters before generating the binary image. The filters are either 3x3 mean filters [47] or 3x3 median filters [46], [48] or a Gaussian filter and an adaptive filter [45].

2.5.2 Segmentation

The generation of a binary image is based on a thresholding algorithm which is either determining the threshold globally or locally, depending on the method. Global thresholds are only considering the individual grey value of a pixel, whereas in local thresholding, the neighbourhood of a pixel is critical as well [49]. A neighbourhood of a pixel consists of four or eight pixels. They can be included in the thresholding process by calculating the mean or median of the grey values. Since local thresholding methods are very CPU-intensive, most studies use global thresholding algorithms. Their thresholds mostly depend on the pixels' intensity values, whereas in one case a global threshold of 0.2 nm was used [41]. Finding the optimal threshold, where only valid fragments and no background are considered, is a challenging task. Ficarra et al. [45], [46], [48] use the Ridler method [50], a method based on a grey value histogram which

iteratively determines the optimal threshold. Spisz et al. [40] use two different thresholding techniques. The first technique is based on two Gaussian distributions fitted to the fore- and background grey values and finds the optimal threshold in between those distributions [51]. The algorithm cannot be applied to all AFM images, therefore they implemented a second method which finds the minimum threshold where the number of recognized fragments (blobs) does not change [52]. Setting a too high threshold leads to the fragmentation of DNA filaments (see Figure 7).

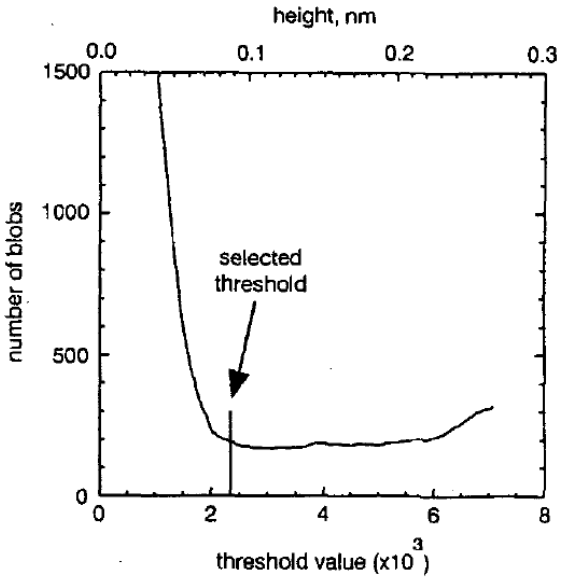


Figure 7: Determination of an optimal threshold. For each threshold the resulting number of blobs were calculated. The optimal threshold is the minimal threshold value at which the number of blobs does not decrease any further. [52]

Another option is to consider a pixel's neighbouring intensity values instead of the pixel's intensity value [38]. By doing so, filtering and segmentation are combined into one step.

2.5.3 Thinning

The thinning step is used in order to erode the fragments to skeletons with the width of one pixel which is necessary for the length determination. The fast parallel thinning algorithm by Zhang and Suen iteratively removes pixels from each fragment if they possess all the conditions of removal [45], [48], [46], [40], [53], [43]. This algorithm does not remove corner pixels and removes valid end pixels with a high probability. The algorithm by Brugal and Chassery [54], [41] iteratively removes connected pixels in a specific order. The end pixels

are not affected by the algorithm, therefore no end pixel restoring is necessary. Sundstrom et al. do not describe their thinning algorithm but only emphasize that it is necessary to convert the fragments into skeletons with a width of one pixel [42].

2.5.4 Removal of Corner Pixel

During thinning invalid corner pixels can be included into the DNA skeleton which results in a longer distance in corner areas. Those pixels are not real compartments of the DNA structure and hence need to be removed [41], [48], [40].

2.5.5 Removal of Objects Across the Image Boundary

For fragments at the image boundary a complete analysis is not possible. Therefore, such fragments have to be excluded from further steps. Ficarra et al. used a 8-pixel neighbourhood for the detection of such objects. If the connectivity in the neighbourhood was not interrupted at the image boundary the object has been removed [45], [46].

2.5.6 Pruning

After thinning some short branches often remain at the skeletons. The reasons are impurities in the sample or noise close to the DNA fragment. Sundstrom et al. [42] referred to a master thesis by Silvio Cirrone CHECK who transformed the skeleton into a graph. Thus the problem was formulated as a graph optimization problem to cope with short branches. Otherwise, the removal of the branch pixels is necessary. Ficarra et al. [48], [45] created a mask to distinguish between unbranched and completely detected cases, branches, critical cases, and corners. Spurious branches have the characteristic of being much shorter than the fragment. This feature is used to identify such branches and to delete them recursively.

2.5.7 Removal of Invalid Fragments

In this step, critical molecules are removed before analysing their length. Spisz et al. [40] carry out this step before thinning the fragments. Fragments are defined as invalid when they are overlapping with other fragments or with themselves, when they are in a closed circle conformation, when the endpoints are not distinguishable, when more than two endpoints are detected and when they exceed a user-defined size [40], [45], [48], [46]. Ficarra et al. used for the removal of such invalid fragments different masks. For overlapping molecules the masks shown in Figure 8 have been used [45].

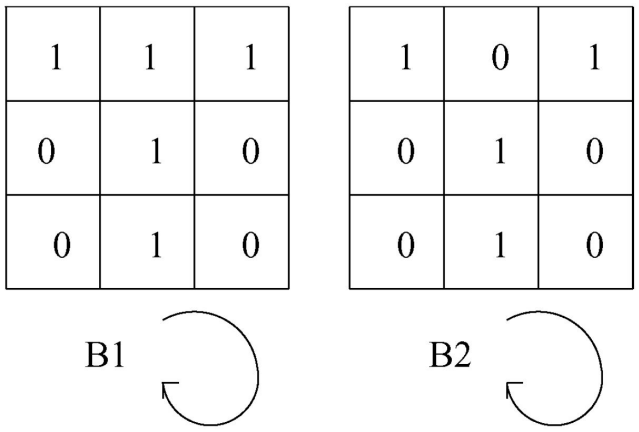


Figure 8: Masks for identifying overlapping molecules. [48]

2.5.8 Pixel Restoring

Pixel at the skeleton ends only need to be restored if they were erroneously deleted during thinning of the fragment. The end pixels of each backbone are virtually extended in the direction of the last two pixels. If the new possible end point was part of the fragment before, the pixel is restored [40], [45], [48], [46].

2.5.9 Length Determination

The determination of the length of the DNA skeletons is achieved by estimating the contour length which is defined as the polymer's length at maximal physical extension [44]. During digitization, the exact contour is lost. Anyhow, the accurate determination of the contour length is for many applications crucial. The Freeman estimator is the most commonly used method to determine the length [40], [43]. When the fragment is reduced to a skeleton with the width of one pixel, the connection between one and another pixel can be represented by eight directions (Figure 9A). The Freeman estimator is adding the distance between the connected pixels from one endpoint to another. Even connections thus with vertical or horizontal direction are counting as 1, odd connections with diagonal connections are multiplied by 1.414.

$$L_F = n_e + \sqrt{2}n_o = 1.000n_e + 1.414n_o$$

L_F: DNA contour length determined by the Freeman estimator
n_e: number of even connections
n_o: number of odd connections

Even though the Freeman estimator is often used, it overestimates the DNA length by approximately eight percent [41]. Rivetti and Codeluppi [44] analysed in 2001 six different algorithms to determine the contour length, one of them being the Freeman estimator. Secondly, they tested the most probable origin estimator, which similar to the Freeman estimator computes an (n_e, n_o) characterization. This algorithm calculates the contour length as a function of the number of egde pixels, number of pixels on diagonal edges and of the area [55], [56].

$$L_{MPO} = \sqrt{(n_e + n_o)^2 + n_e^2}$$

L_{MPO}: DNA contour length determined by the most probable origin estimator

Much better results were obtained for the Kulpa estimator which is derived from the Freeman estimator. The coefficients of even and odd pixels minimize the error when measuring the length of long fragments.

$$L_K = 0.948n_e + 1.343n_o$$

L_K: DNA contour length determined by the Kulpa estimator

The corner count estimator considers additionally to the number of even and odd connections the number of different code pairs, so called corners (Figure 9B) [57]. The coefficients were found by least-square fitting [58].

$$L_C = 0.980n_e + 1.406n_o - 0.091n_c$$

n_c: number of corners

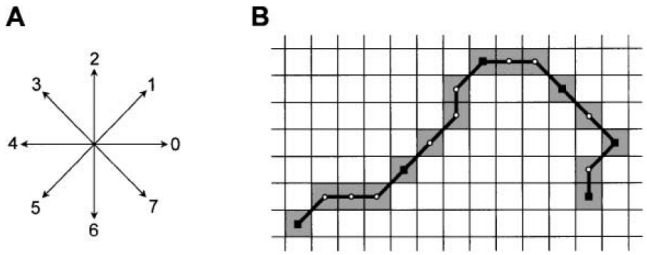


Figure 9: (A) Scheme of the eight connected chain code. (B) Example of a fragment skeleton containing 16 pixels (grey grid elements). The chain is connected by 15 codes, from the left to the right chain the codes can be written as 100111210077756. There are six even codes (n_e) and nine odd codes (n_o). The total number of corners (n_c) is six.

As fourth estimator Rivetti and Codeluppi tested an estimator where the

backbone was primarily smoothed by using polynomial fitting. The coordinates of each pixel where thereby adjusted. The polynomial degree was three and the moving window consisted of five points since the estimation with this combination were very close to the real length.

$$L_{PF} = \sum_{i=1}^{n-1} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}$$

L_{PF}: DNA contour length determined with polynomial smoothing

The last algorithm included in their analysis was the edge chain algorithm which was originally implemented to measure the length of the roots of plants. It can be applied for other objects with relatively constant width and draws chords along the object edge to determine its perimeter.

$$L_{ECA} = \frac{P + \sqrt{P^2 - 16A}}{4}$$

L_{ECA}: DNA contour length determined by the edge chain algorithm
P: perimeter
A: area

Rivetti and Codeluppi generated synthetic data similar to their AFM images with DNA fragments of different length and tested the estimators while knowing the exact length of the synthetic filaments. The estimations with the Kulpa estimator, the corner count estimator, and with polynomial smoothing were significantly closer to the real length of the filaments than the estimates generated by the other three estimators. The overestimation of the Freeman estimator was confirmed [44].

Ficarra et al. [48], [45] calculated the molecule length using the Euclidean distance which was integrated over all consecutive pixels of the molecule. For calculation the pixel coordinates were recalculated as weighed average by using a weight factor k.

$$X_P = k(x_{p-1} - x_p) + x_p + k(x_{p+1} - x_p)$$
$$L_{P+1,p} = \sqrt{(X_{p+1} - X_p)^2 + (Y_{p+1} - Y_p)^2}$$

X_P: weighed average x coordinate
k: single weight factor
$L_{p+1,p}$: modified distance between the points with coordinates x and y

3 Methods

3.1 Manual Analysis

To annotate and evaluate the results of the algorithm the open source program ImageJ (2.0.0-rc-46/1.50g) was used. Therefore, the features from a sample of objects or from whole images, were analyzed manually with the tools of ImageJ. For the objects, which were single DNA-strands and mono nucleosomes, the location and the couture length were determined. For nucleosomes, also the angle between the two DNA-strands entering the nucleosome core, the approximate area of the nucleosome core and the mean gray value of the nucleosome core were measured. To annotate the manual data from one object to the results of the algorithm the centre of mass was used. Therefore a binary image was created by thresholding. Then the "Analyze Particle Tool" was used to read out the x-y coordinates of the centre of mass from the different objects. All other parameters were determined on the original images. To measure the different lengths, the "Segmented Line Tool" was used. One end of a DNA strand was set as starting point, from which the line-points were drawn centrally through the fragment towards the other end of the fragment. This so called couture length was also measured for mono nucleosomes, by drawing centrally through the DNA and through the approximate center of the nucleosome core (figure). For mono nucleosomes, however two lengths were determined. The second length was the distance between the approximate center of the nucleosome core and the end of the shorter of the two entering DNA-strands. The entry/exit DNA-strand angles of mono nucleosomes were measured by using the three point selection "Angle Tool" of ImageJ. Figure illustrates that the entry/exit DNA-strand angles can be determined in two different ways. In that context, α_1 and α_2 were measured on different mono nucleosomes for the test data evaluation. α_1 was measured by connecting the two entry/exit points of the nucleosome with the approximate center of the nucleosome core. For the determination of α_2, the two DNA-axes at the nucleosome core were traced. The angle between the two DNA-axes was then measured at their intersection point. The approximate area and mean gray value of nucleosomes were determined by manually fitting an elliptical structure to the nucleosome cores with the "Oval Selection Tool" of ImageJ. Besides single DNA strands and mono nucleosomes, the locations of special objects were recorded. The special objects were intersecting DNA-fragments/nucleosomes, self-intersecting DNA-fragments/nucleosomes, objects on the edges of an image and undefined objects. To increase objectivity, test data were measured from four different persons.

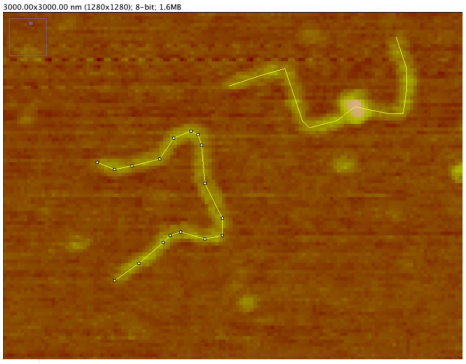


Figure 10: Couture length measurements of DNA and nucleosomes The contour length of DNA and nucleosomes were measured with the "Segmented line tool" of ImageJ. The white bars indicate the selected points. The lines have been traced centrally through the DNA fragments from one end point to the other end point.

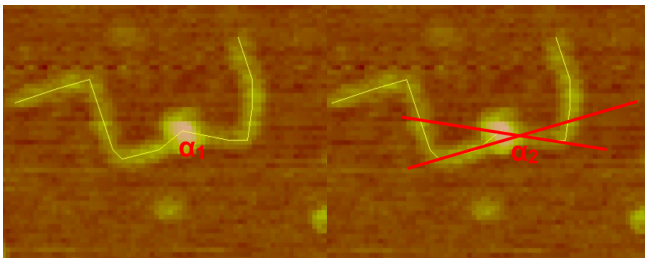


Figure 11: Angle measurement of mono nucleosomes The figure shows two possible angles α_1, α_2 which can be measured between the two entering DNA strands of an nucleosome. α_1 is the angle between the entry/exit points of the DNA with respect to the approximate center of the nucleosome core. α_2 is the angle between the DNA-axes (red lines), which are crossing the entry/exit sites of the nucleosome core, with respect to their intersecting point.

3.2 Software Architecture

3.2.1 Processing Pipeline

All images of the dataset are processed in a sequential way by the following nine steps:

1. Denoise
2. Filter
3. Create binary image
4. Detect nucleosomes
5. Detect DNA strands

6. Calculate length of DNA strands
7. Calculate angle between arms of a DNA strand if it possesses exactly one nucleus
8. Save Results as CSV file
9. Write processed image to disk

At first every image is denoised by the OpenCV function fastNlMeansDenoising [1] (see Section 3.3) and passed on to the MATLAB main pipeline(see figure 13). Inside the MATLAB pipeline an object-oriented approach is used. This has the advantage that fields can be accessed easily like this:

imageObject.property

Also the data is stored in a structured way which not only provides accessibility and modularity but also ensures maintainability and expandability.

Each image is instantiated as a object of the image class (figure 12) which can hold essential properties such as a dnaList for detected DNA strands and numerous other information about the detected objects. All data gathered during the process, such as the filtered image, the binary image, the thinned DNA strands, is stored as an instance variable. Therefore all processing steps can be reviewed separately later on. For example the denoising and filtering steps where improved by examining the intensity histograms of the pre-processed images. Also different thresholds can be tested and assessed directly. After being filtered by a median filter(see section 3.4) the lowPassFilter(see section 3.4.2) function is applied to each image. To reduce noise even further the background is subtracted from the original image by the MATLAB imOpen [2] method. The threshold distribution over the entire dataset is required in step three. Therefore all images have to be pre-processed by step one denoising and step two filtering before step three can proceed further. To create the binary image the adaptive thresholding function(see section 3.5.1) is used. After that every image is scanned for nuclei by the findNuklei(see section 3.6)function which also stores them as centers and radii as fields of the current image instance. Remaining objects are classified as DNA strands and therefore instances of the DNA class are created and saved in the DnaList field. Each DNA strand is either stored as DNAFree if no attached nucleus was detected or as DNABound if at least one attached nucleus was detected. Since all objects which persist after the three initial steps are classified as DNA strands, polluted image regions with a similar size are misclassified as DNA. To solve this problem DNA strands are marked as invalid if they do not meet certain criteria such as number of pixels, length and number of attached nucleus. Only DNA objects with zero or one nuclei are permitted. Step six, calculating the length of the DNA strands also marks DNA objects as invalid based on their length (see section 3.8). In order

[1] http://docs.opencv.org/2.4/modules/photo/doc/denoising.html
[2] http://www.mathworks.com/help/images/ref/imopen.html

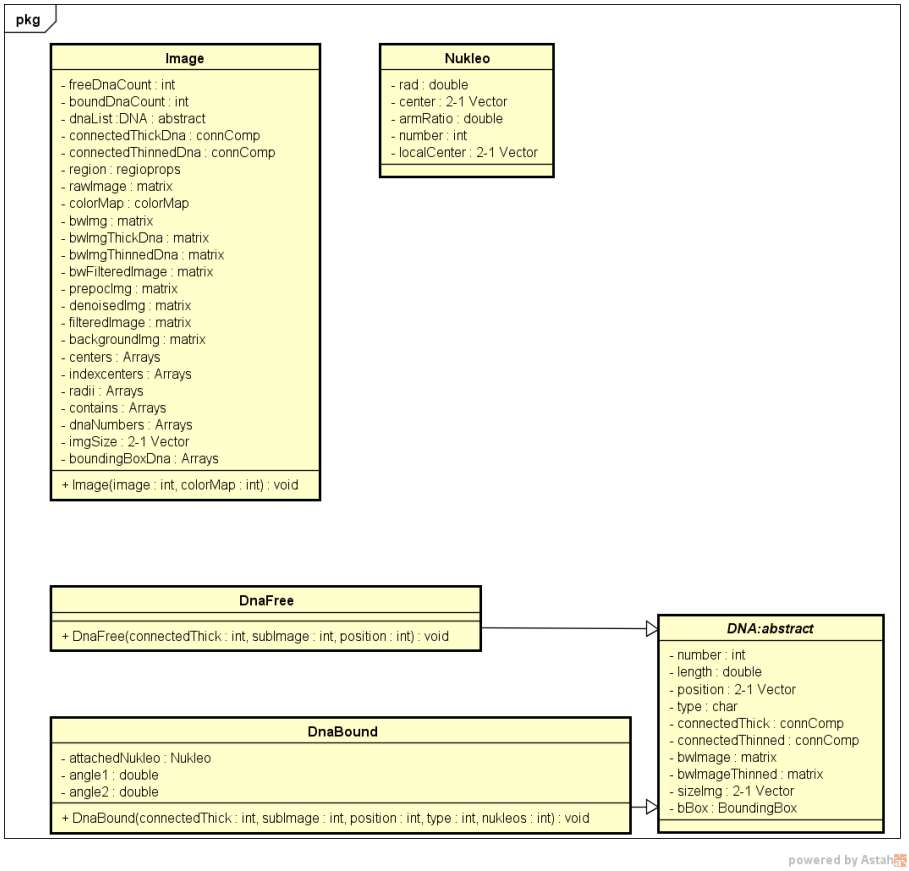


Figure 12: class diagram of the pipeline of the program

to calculate the length correctly it is necessary to thin the DNA strands first and remove the branches which arise during this thinning. Step seven(Section 3.9) calculates the angle between the two arms of the DNA strand, this only applies to DNA strands with an attached nucleus which are stored as DNABound in the dnaList variable. DNA strands with more than one attached nucleus are marked as invalid. The features length, angle, position on the image, position of the attached nucleus, radius of attached nucleus and the isValid flag are exported as a CSV file with filename,"meaning the name of the file" [59], [numberOfImage]-test.CSV.

By exporting the isValid flag invalid DNA strands can be ignored during the evaluation but still persist in order to analyze difficulties, increase detection rate and improve performance. At last all images created during this process, such as the binary image, are written to the disk.

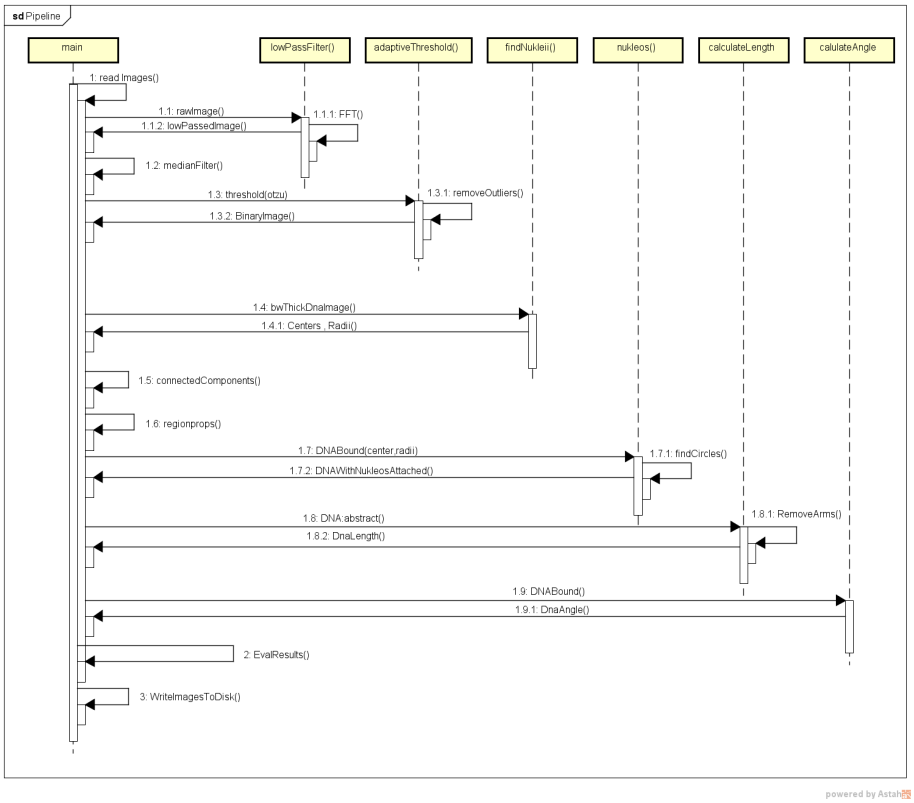


Figure 13: sequence diagram of the pipeline of the program

3.3 Denoising

The first steps in our MATLAB routine deal with preparing all images in order to binarize them with the most success, i.e. with the best DNA recognition rate as well as most accurate length estimation. This means that the routine removes as much noise and as many different kinds of noise so that it automatically and, before all else, correctly can classify any image pixel as either belonging to a DNA fragment / nucleus or as background. For this to achieve, we found that the initial step needs to be a noise reduction method called *Non-local Means Denoising*. It is very well suited to remove white noise from images, which is a noise type that is very typical for AFM images. White noise is defined as a random signal that has a constant power spectral density. Although the image processing library of MATLAB is very extensive, no function implementing *Non-local Means Denoising* could be found. Those denoising functions that are provided by MATLAB, such as *wdencmp* in conjunction with *ddencmp*, or *wiener2*, proved not to be as suitable for denoising the here analysed images, due to their very specific noise types (which are also discussed in greater detail

in Sections 3.4 and 3.5). However, the method is provided by the open source computer vision and machine learning library *OpenCV* ([60] [61]).

3.3.1 Non-local Means Denoising Algorithm

The algorithm of *Non-local Means Denoising* is provided by the OpenCV library function *fastNlMeansDenoising* that requires the following parameters:

- *filter strength* (float): The larger this parameter, the better noise is removed and the more details are lost. In our case we chose the value 2 since this appeared to provide a good balance between noise removal and loss of DNA details.
- *template window size* (int): Window size in pixels of the patch that is used to compute weights. The recommended value is 7, which we found to be sufficient.
- *search window size* (int): Window size in pixels of the patch that is used to compute the denoised value of the currently examined pixel from. Since this value affects performance linearly, we chose this value to be slightly smaller than the recommended value of 21, namely 17. This proved to be sufficient for good denoising results.

The heuristics to find these parameters were a) to achieve fastest possible runtime while b) providing satisfactory levels of denoising as measured by final DNA detection rate and length estimation accuracy. Increasing the search window size from 17 to up to 31 greatly increased the removal of background in the resulting preprocessed images. However, in conjunction with the subsequent steps of Filtering (3.4) and Thresholding (3.5), this gain did not result in a notably improved detection rate of DNA objects or increased accuracy of length calculation (data not shown). Since it also had a drastic negative impact on the overall runtime, we decided to keep this parameter as low as possible without loosing the benefit of denoising. Comparable observations were made when investigating the required template window size. Increasing it resulted in increased runtime without any distinct gains in image denoising.

The great strength of this method is that for each pixel p of an image it searches for similar pixels - not in a close distance around that pixel, like many other noise reduction approaches do, but over a large portion of the image. The average value of these pixels is then calculated and set as the new, denoised value of the currently examined pixel p. In order to further increase robustness, similarity between two pixels p and q furthermore is computed by calculating a weighted Euclidean distance between square patches around these pixels, and not between only those two pixels. So, the denoised value $\hat{u}(p)$ of pixel p is calculated as follows:

$$\hat{u}(p) = \frac{1}{C(p)} \sum_{q \in B(p,r)} u(q) w(p,q)$$

with

- u(q) being the current value of pixel q
- B(p,r) being the neighborhood of pixel p in a (2r+1) x (2r+1) large window centered at p
- C(p) being a normalisation factor:

$$C(p) = \sum_{q \in B(p,r)} w(p,q)$$

- w(p,q) being a weight function with an exponential kernel for the determination of similarity between pixels p and q:

$$w(p,q) = e^{-\frac{max(d^2 - 2\sigma^2, 0.0)}{h^2}}$$

- d^2 being the square Euclidean distance of the $(2f+1)x(2f+1)$ large color patches centered at p and q, respectively:

$$d^2(B(p,f), B(q,f)) = \frac{1}{3(2f+1)^2} \sum_{j \in B(0,f)} (u(p+j) - u(q+j))^2$$

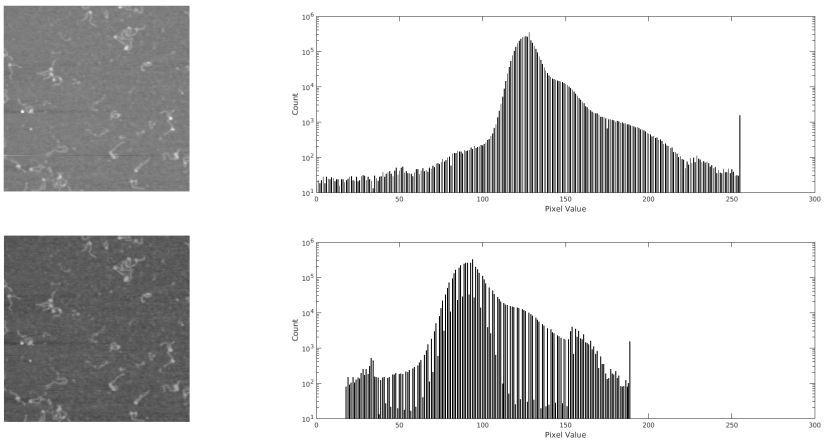
- σ being the standard deviation of the noise
- h being a filtering parameter depending on the noise level σ

A much more detailed description of the algorithm can be found at [62], which we recommend to the interested reader.

3.3.2 Benefits of using Non-local Means Denoising

As can be seen in Figure 14, the denoising procedure results in a much sharper separation of DNA objects from their background than without. This is especially apparent when comparing the histogram of an yet unprocessed image with the histogram of the same image after it has been denoised (Figure 14b, upper row and lower row, respectively). Before denoising, all transitions between any at least slightly bright object and the background were rather smooth as is indicated by the overall brightness of the original image and the smooth surface of its histogram. However, after denoising, these transitions have become much steeper, which is represented by the gaps in the histogram of the denoised image and which is reflected in an increase of image contrast. Although not too strong in its distinctiveness, this outcome is the primary basis for the efficacy of all subsequent noise reduction measures.

Furthermore, some artifacts already could be removed. AFM images often show long but only few pixels wide horizontal lines that stem from the cantilever skipping while scanning the image. One such example is visibile in the upper image of Figure 14a in the lower left corner. This artifact is not visible anymore in the denoised image (lower image of Figure 14a) - it was replaced by background during denoising.



(a) **(b)**

Figure 14: Effect of denoising procedure on grey scale images and their histograms. The upper row shows an unprocessed image before denoising (a) as well as its histogram (b), while the lower row shows the same image (a) and histogram (b) after denoising. Histograms show a logarithmic y-axis. It is apparent that denoising results in a sharper separation of DNA from background, thereby increasing the image's contrast.

3.3.3 Technical considerations / limitations

As mentioned in Section 3.2, our program is written using MATLAB. However, OpenCV is a C++ library, so *fastNlMeansDenoising* is a C++ function. Consequently, we had to find a way to combine MATLAB with C++ code.
For such cases, there exists a C++ interface library that is provided by MATLAB (see [63]). With its help, C++ code can be compiled into so-called mex files, which themselves call OpenCV functions and which can be called like regular MATLAB functions. We succeeded in integrating such specifically compiled mex files into our main program and, thusly, in denoising all images. Unfortunately, we had to discover that the resulting, denoised images were considerably different from those that had been denoised by the original C++ function. Since we were unable to discern the reason for this unexpected behavior, we had to find other measures to use OpenCV functionality from within our MATLAB routine. Therefore, our routine now makes a system call in order to run a C++ executable file. This file is a standard executable compiled from C++ code containing the denoising instructions; it will read all images, denoise them, and temporarily create and write new, denoised images. These images then are read by our MATLAB routine as input for the rest of the program.

3.4 Filtering

3.4.1 Necessity of Filtering

At first, one needs to ask why it is necessary to use filtering techniques in the first place, as it is stated in various other papers (compare [45], [45]). Since the goal of this work was to find a most efficient and fast solution, a heuristic approach was proposed which only took object size of connected components into account. For this the MATLAB function bwconncomp was applied to a black and white image of the thick DNA strands.
Although this eliminated most of the smaller fragments, and some larger ones, obviously some objects remained that were dirt or impulsive noise.
Like Ficarra et al. [45] pointed out, a median filter with a 3x3 mask helps eliminating these components quite well, yet there is still room for improvement. Regarding the aspect of very limited available data, it was another aim to maximize the number of sampled strings. Therefore, other, less common techniques besides the already tested filters were investigated.
As it was rather soon apparent that regions with impulsive noise had a rather high frequency, a lowpass filter approach seemed suitable.

3.4.2 The Concept of Adaptive Low Pass Filtering

Firstly, the most prominent example for a lowpass filter is the Gaussian filter. It essentially smoothes the data, but in a relatively predictive way, since it has a predefined filtering matrix. But, since the signal-to-noise ratio is not always known , a simple Gaussian filter might only blur the edges of the outer DNA regions, instead of effectively reducing the amount of white noise. Rather than simply smoothing the image with a Gaussian filter, the idea was to eliminate regions with high contrast.
Removing such high-contrast regions is best achieved in frequency space. The freuquency domain, simply put, transforms the original image data into a decomposition of sinus waves. For the transformation between spatial domain and frequency domain the MATLAB implementation of the Fast Fourier Transformation was used. Without going into further detail, the amplitude of such a frequency representation gives a comparatively abstract, but nonetheless intuitive, visualisation of such high contrasts as shown in figure 15a.
As one can easily observe, there is a denser circular area in the image center. This central area represents the lower frequencies which corresponds to a constant image region. This can be seen on every picture of the used dataset but is to be expected in every other dataset as well. The only difference lies in the clear distinction and brightness of the center. Since a classic lowpass filter requires a clipping region as a fixed parameter, it would not work perfectly for every picture because one would inevitably overfit for a certain set of images. Standard circular detection via circular Hough transform has not been able to sufficiently detect this area.
It should be mentioned that for non-quadratic images an ellipsoid instead of a circle can be observed. In this case the algorithm differs only in one step, so for

consistency and simplicity only the circular case is presented.

3.4.3 Description of Proposed Method

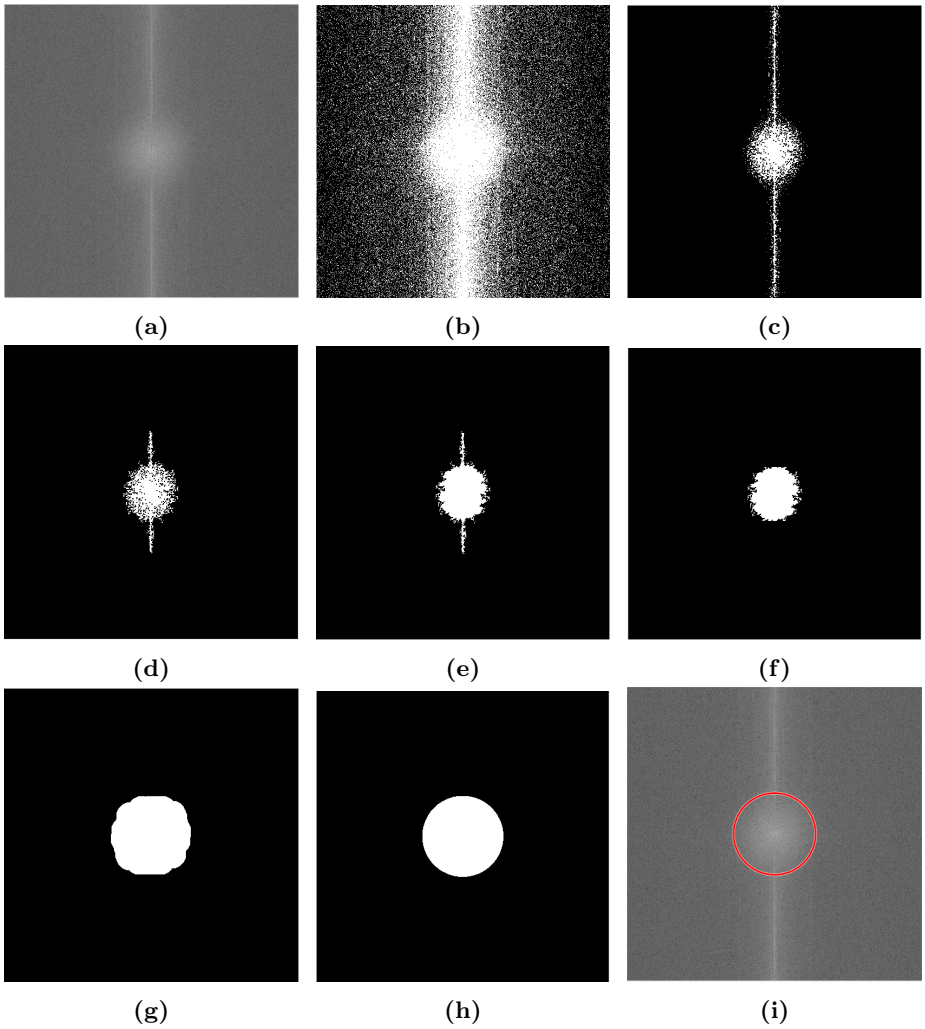


Figure 15: Circle detection (a) Amplitude representation of the initial FFT transform (b) Binary image (c) Eroded image (d) largest connected component (e) After hole filling for (f) Removal of spurious edges (g) Dilated image (h) Final result, and (i) comparison between initial and final image

As input, the method takes an arbitrary gray-value image which then is transformed to the frequency domain with MATLAB fft2 function. For a represen-

tation similar to figure 15a a shifting of the result has to be done as well. A detailed description of the frequency functions and their specific usage can be found on Mathworks' homepage[3].
The complex result has then to be transformed into a gray-value amplitude image in which the imaginary part is discarded and the real part is visualized as a logarithmic absolute value. Now the result is generally the same as above.
For the approximation of the circle it is also necessary to convert the image to a binary image. Since only a rough threshold is necessary a simple and fast algorithm for thresholding is used (built-in Otsu-Thresholding as described in Section 3.5). This rough approximation can be excused due to the clear lack of a distinct circular contour. This also explains why the implemented circular Hough transform (MATLAB imfindcircles) was not able to deliver satisfying results.

In the used dataset sometimes a distinct lack of very low frequencies - equaling a sharp (yet only one pixel in width) line in the center - could be noticed. To rule out similar cases in other datasets as well, a 20x20 pixel white rectangle was inserted in the center (see result section for further discussion). Afterwards standard erosion with a disk shape is performed. With erosion, noise that persists in the binary image is eliminated by fitting these disks around each black pixel and coloring all pixels within this shape black as well. A detailed explanation on how to define structuring elements such as the used disk and the MATLAB imerode function can be found on Mathwork[4].
Out of the remaining connected objects the circle now represents the largest one. Hence it is filtered out with MATLAB

```
bwareafilt(.., 1, 'largest')
```

which extracts this largest region and the resulting image has only this central part colored in white. To assure a compact center another operation is performed which fills in enclosed black areas within the circular region

```
imfill(...,'holes')
```

The remaining spikes at the top and bottom of the cirlce are removed by deleting every row where there are only a certain number of white pixels. To reverse the erosion process and get a representative circle the MATLAB imdilate operation is performed with a disk structure.
MATLAB's regionprops function delivers the centroid as well as the minor and major axis length of the object which then is used to calculate an approximation of the actual radius of the circle. Notice that it is also possible to extract an elliptical structure from non-quadratic images which will not be described in detail here since it does not differ from the steps for a quadratic image.
For the radius approximation the following formula is used: (insert formula here)

[3] http://www.mathworks.com/
[4] http://www.mathworks.com/

Since the removal of the spikes also flattens the circle a bit, it is important to slightly weight the major axis length a bit heavier. For the distance to the center a 2D-grid is produced by MATLAB ndgrid, using one grid for the distance in x- and one for the distance in y-direction. Afterwards, the distance to the center point is then calculated by the ordinary circular formula to determine whether a point is within the circle (another formula here). Evaluating this formula for each point allows to select the circular region which is then stored as a binary matrix.
Finally, the circular region is "cut out" of the original image by multiplying it with the initial fourier transformed image. Before retransforming the image, an inverse shift has to be done with ifftshift. After using ifft2 to recover a spatial form of the image it is also important to transform it back into a uint8 image, using only the real-valued part of the transform. A uint8 image is an image consisting of unsigned 8-bit integer values in the range of 0 to 255.
This can all be put together in one code line, resulting in

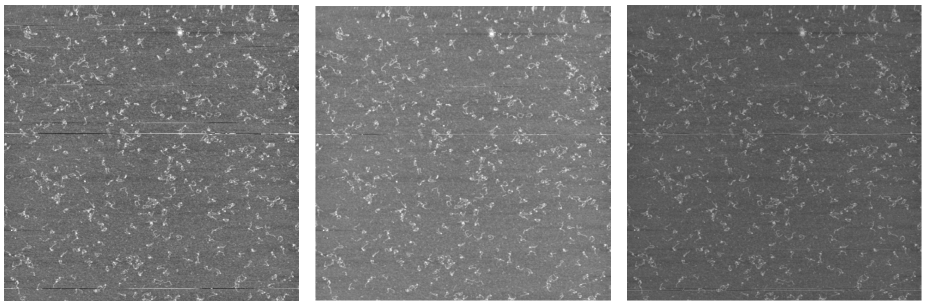
```
image = uint8(real(ifft2(ifftshift(image))));
```

3.4.4 Advantages over Other Filtering Techniques

Before comparing the results of the proposed method it should also be discussed why no other filter was used instead. The biggest issue with filtering techniques is that it is hugely dependent on the available dataset. While others (insert papers with AFM image data here) usually have a pretty distinct noise level and try to optimize for this specific set of images, the goal of this work was to really adapt to various datasets. For different image qualities of AFM pictures, and especially for different resolutions and cutout sizes, results may vary although done by the same instrument or person.
Ficarra et al. [45] use three distinct filtering techniques for the preprocessing step: A 3x3 median filter, an adaptive Wiener filter and a high-pass filter. As for the median filter, the proposed method was also used in this work (compare chapter Median Filter). As a side note it should be mentioned that the median filter has good results for filtering out sudden peaks of noise but is rather inefficient with larger dabs of noise in an image.
The mentioned Wiener filter was not pursued any further for the following reasons. First, it works best for a constant noise level which isn't given in all datasets what ultimately let to the design of an own method. Also it is very suitable to clean up the inital data which isn't as much a requirement to this algorithm since it has a fully devoted step of image denoising before filtering out anything. Hence another denoising step would only disturb the image more than it would help remove further artifacts.
The problem of the high-pass filter was a bit more subtle. Although it is helpful in providing sharper edges for the later detection of DNA strings, it also attenuates the intensity border between dirt and actual DNA strings (compare section on Histograms). As for the high pass filter itself, it does the exact opposite of a low pass filter: Namely sorting out all of the lower frequency in Fourier

space. The aim of Ficarra et al. [45] was to eliminate noise which was rather low-freuqent but yet again this is a step usually eliminated with denoising in this algorithm.
Generally the methods by Ficarra et al. would lead to a much lower rate of actual DNA in the resulting objects and therefore requiring a more subtle decision in whether or not it actually is a DNA string which is currently viewed. By sacrificing a distinct border between the DNA and background, a much lower percentage of noise will be classified as a false positive in the end.

3.4.5 Results and Comparison

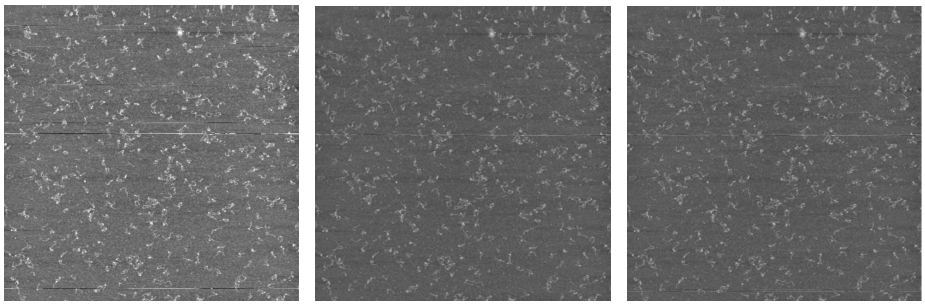


(a) **(b)** **(c)**

Figure 16: Comparison between different filtering techniques
(a) original image (b) median filter (c) low pass filter

In this part the described low pass filter will be compared to a standard median filter. The used median filter was implemented by Mathworks and the algorithm used is medfilt2 [5]. In the original image one can see very clearly many noticeable horizontal lines. The occuring of this lines itself makes it obvious again why a good filtering method is needed. And indeed the used median filter removes most of the lines. Only one prominent line in the middle remains.
On the third image one can see the effect of the implemented low pass filter. Immediately it is obvious that this image is much darker than the other two images, especially the median filtered. This is due to the fact that the average color of the original image is rather dark. Since the low pass filter averages over the data the result is darker than the original. On AFM images this is nearly always the case. Secondly, in the low pass filterd image one can see more than one horizontal line. Each of these lines is not as prominent as the one extracted by the median filter though.
On the other hand both methods hugely decrease the background noise compared to the original image. However on the low pass filterd image the background is even more uniform.

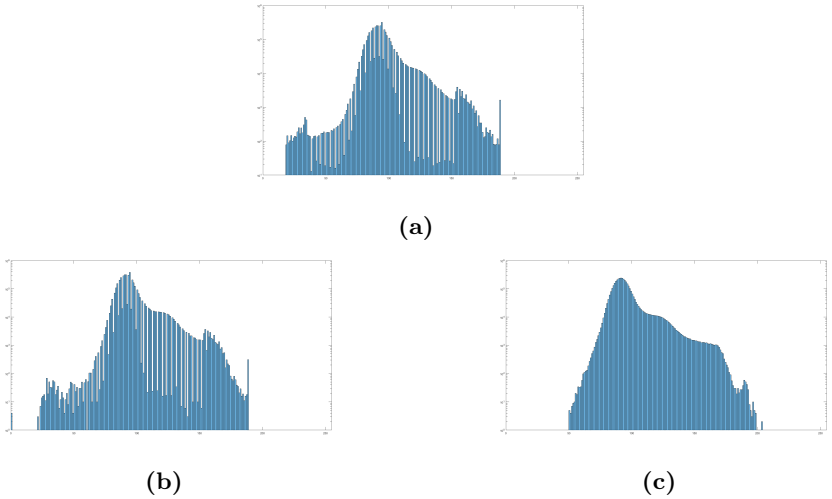
[5] http://de.mathworks.com/help/images/ref/medfilt2.html



(a) **(b)** **(c)**

Figure 17: Comparison between different filtering techniques
(a) original image (b) median then low pass filter
(c) low pass then median filter

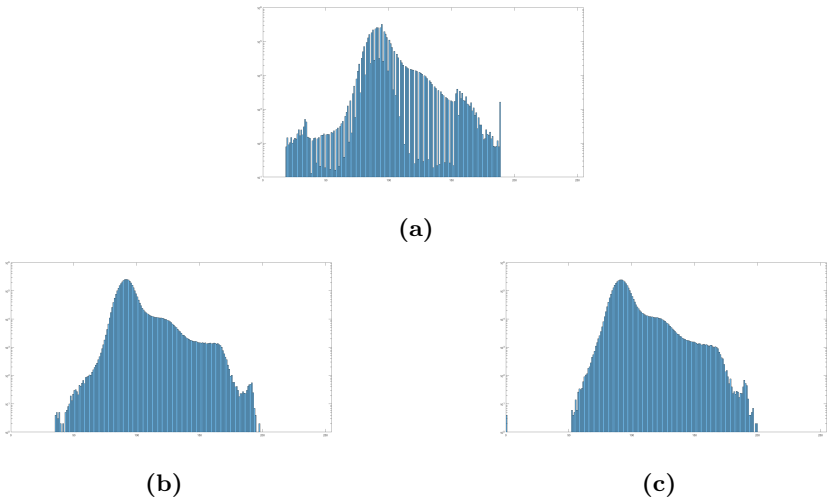
Since both filtering methods did not achieve an ideal outcome yet the idea came up to combine both methods. In figure 17 one can see the original image and a combination of the two filters. On both filtered images a similar effect to above can be observed. If the median filter is applied first the number of horizontal lines are reduced greatly but the remaining line is really prominent. When the low pass filter is used before the median filter the number of lines is higher but the lines itself are smaller and do not stand out as much as in the other case.
However, both methods share the advantage that the background is as uniform as the background of an only low pass filtered image.
Both of these approaches were implemented and analyzed in regard to their respective results of later processing steps. It could be shown that more useful DNA strands could be extracted by using method 17b. This is in accordance with what one would expect looking at the results above. As a result in this project always the median filter was applied before the low pass filter.



(a) **(b)** **(c)**

Figure 18: Histograms of different filtering techniques with logarithmic y-axis scaling. (a) original image (b) median filter (c) low pass filter

In figure 18 three histograms of the original image and after the application of the two filtering techniques can be seen. As one can see in 18a the original image is lacking an explicit distribution. Many peaks can be observed, especially a striking one at the right end of the spectrum. Obviously there is some shadowing in the distribution as well which makes the distinction between noisy and DNA regions quite hard because a patch consisting of only a few pixels of the same value might still be in range of the values the DNA consists of.
In comparison the median filtered image does look very similar at first glance. However, a better distinction can be observed at lower values while at higher values the distribution does not differ as much and there is a very striking peak at 0 which is irritating but negligible. A big advantage can be seen regarding the shadowing of the values. The shadowed peak is much narrower than in the non-filtered image and the low points in medium range are not as low as before. On the other hand the slope of the peak is much higher which is useful for differentiating between background and DNA objects.
The histogram of the low pass filtered image 18c is eliminating the shadowing problem. The distribution of the values is really smooth after this processing step. One can observe that low values below 50 are cancelled out which leads to the impression that the slope is steeper than before. In truth only the distribution of the values has changed. On the other side of the spectrum the high peak is smoothed out too which leads to higher values than in the original image. This shift to higher values leads to the herein before mentioned effect of darkening the picture. On the other hand the general distribution of the values remains the same and the peak is not narrowed as it is by using the median filter. Similar results can be observed in all images.



(a) **(b)** **(c)**

Figure 19: Histograms of different filtering techniques with logarithmic y-axis scaling. (a) original image (b) median then low pass filter (c) low pass then median filter

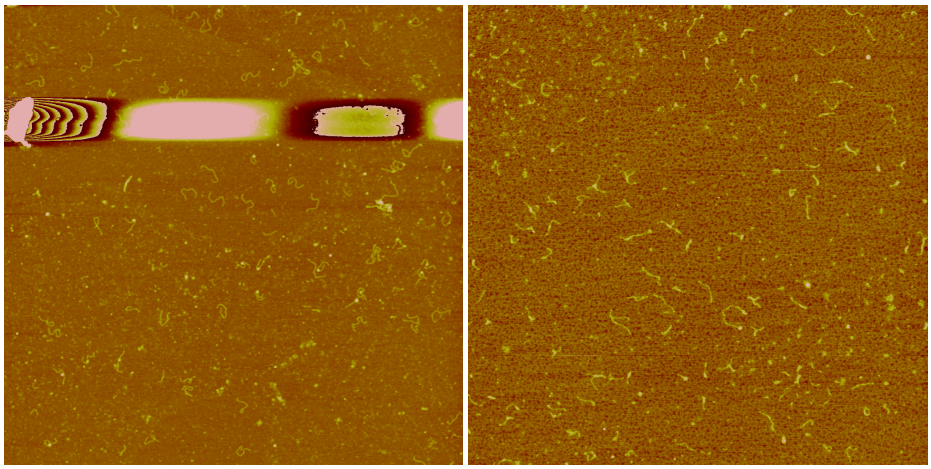
In figure 19 the result of the combination of the two filtering techniques can be observed. Both sequences share the trait of the low pass filter that there is no shadow anymore and that the values are higher overall. The high peak at the end of the spectrum has vanished in both cases too.
When the low pass filter is applied before the median filter one can see a peak at 0 again. If the algorithms are executed in the reverse order this effect vanishes and pixel values below 50 exist which does not matter much but should be considered nonetheless. In case 19b the slope between values of 50 and about 90 is a bit steeper than in case 19c and the values are more equally distributed at pixel values betwenn 130 and 170.
The steeper slope explains the results that could be observed earlier. By applying the median filter first the DNA and the background can be distinguished better. Another useful property is that the upper bounds are clearly discernible through a steep slope as well. Although this wasn't particularly helpful in the used dataset, it might come in handy for normalized datasets. The reason behind that is that a normalized dataset (with uniform gray-values for DNA strands) would allow to reject noise peaks as well. As mentioned before, in the used dataset no such normalization was performed which led to further spread values for DNA and would not allow a discernible region for noise.
One should keep in mind that the peak is not the DNA but the background. So a clear peak at values of around 100 extracts the background perfectly. The results seen in 19b confirm the impression given earlier in figure 17 that applying the median filter and then the low pass filter deliver the best results.

3.5 Thresholding

After denoising and the subsequent application of our filter combination, the next logical step is to determine a threshold for the binarization of each image. Binarization means that each pixel of an image can be classified either as background or as belonging to a potential DNA strand / nucleosome. It is done by computing a specific gray value, the threshold, each pixel's gray value is compared to in order to classify it.
In a first approach, we tried several different thresholding algorithms, such as Intermeans-, Moments-, Intermodes-, Maxentropy-, Minerror-, Percentile- and Otsu-Thresholding. Except for the last, all of these were adapted from GNU Octave ([64]), since MATLAB only provides the Otsu-Thresholding method. GNU Octave is an open source high-level numerical computation language comparable to MATLAB. Some of them determine a global threshold, some a local. However, after several tests especially in conjunction with the rest of our Thresholding algorithm we found that Otsu-Thresholding provided us with the best results (data not shown).
The greatest problem was that many images still contained noise types that have not yet been removed but that had a strong influence during the computation of these thresholds. If not removed, the corresponding threshold will be so much distorted that meaningful / useful binarization will not be possible.
Therefore, an adaptive thresholding algorithm (see 3.5.1) was designed that allowed us, on any image, to create a much more uniform background. This step is one of the main reasons for the general robustness of our routine towards different noise sources.

3.5.1 Adaptive Thresholding

The TIFF images often vary in intensity, distribution and scaling of the height profile of the DNA samples. This presents a couple of problems for the thresholding algorithms, such as pollution of some image regions which distort the threshold. As discussed above, these issues lead to a false classification of the DNA strands or even to a complete loss of image information. Two sorts of noise persist after the initial pipelining steps Opencv Denoising, LowPassFilter and Median Filter (see Sections 3.3, 3.4).
The first type of noise as seen in Figure 20a are large distorted regions in the upper value range at an approximate height of 190. If not handled these regions lift the threshold to a level where DNA-strands are missed entirely. The second sort of noise seen in Figure 20b consists of small imperfections scattered over the whole image. Unfortunately the height profile of these imperfections is very similar to the one of the DNA-strands and it is therefore very hard to automatically distinguish between them. The overwhelming majority of disturbance on the images observed is part of the first type. The aim of this part of the pipeline is to pre-process the data by three additional steps which treat the outliers in the upper value range and if possible improve detection and removal of scattered noise.



(a) **(b)**

Figure 20: (a)Noise Example 1 (b)Noise Example 2

In order to tackle these issues the following three step method is proposed:

1. Homogenize the background to provide a consistent and distinguishable layer on which the DNA-strands can be recognized easily.
2. Identify and remove polluted image regions and outliers in the upper intensity values.
3. Limit the threshold to an appropriate range.

3.5.2 Level Background

Figure 21 shows a histogram of the height values of a representative original image from the dataset. In the observed dataset DNA-strands can only be found in an intensity range from about 110 to 160, while nuclei can be found between 160 and 200.

The accumulation point at a value of 95 is distinctive for background noise. Therefore, smaller values are set to this boundary. This approach creates a more consistent background for all images and limits the variation due to noise. It is important to mention that this method can only be used after the lowPassFilter which smooths the height distribution and thereby guarantees that only noise is removed. If applied to the raw unfiltered image this method will also remove outliers within the DNA-strands which results in perforated DNA objects. The resulting image shown in Figure 22 yields a more consistent threshold and with that a higher error resistance.

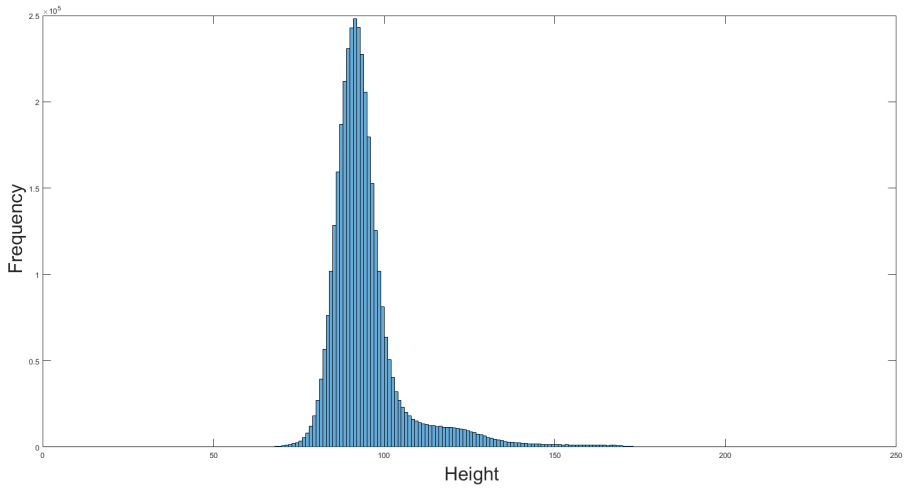
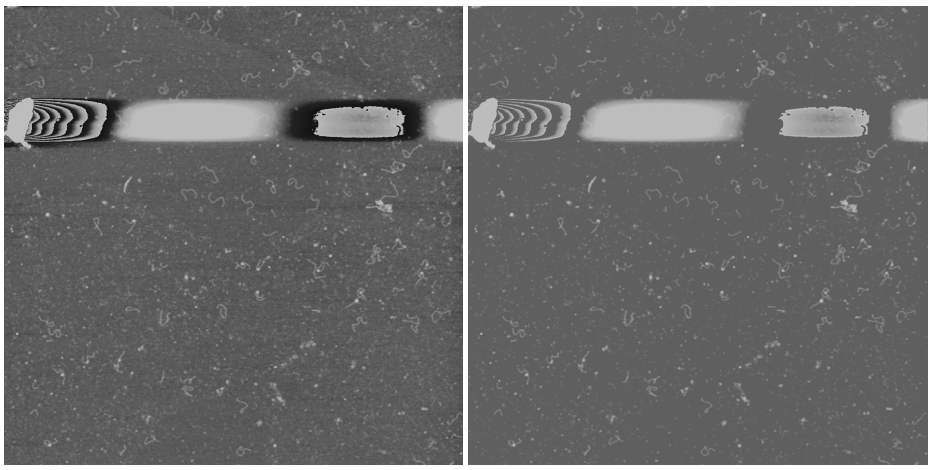


Figure 21: Histogram of TIFF image



(a) **(b)**

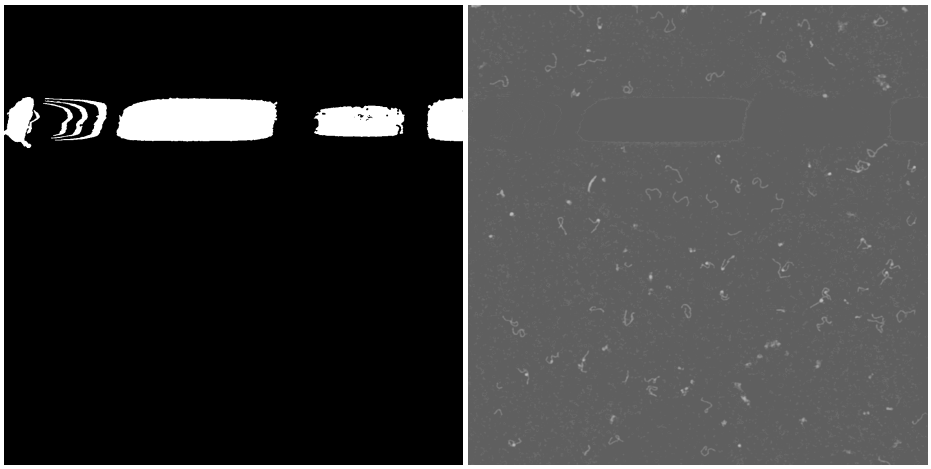
Figure 22: (a)Before leveling (b)After leveling

3.5.3 Identify and remove outliers

At first a global threshold algorithm like the Otsu method (see section 3.5) is used to create a binary image. All objects that could be a DNA object are removed based on size. Now only noise and other non-DNA objects are left.

This image is then used as a mask (compare Figure 23a) which removes all non-DNA objects on the original image from step one.

This means subtract the detected regions on the mask from the original



(a) **(b)**

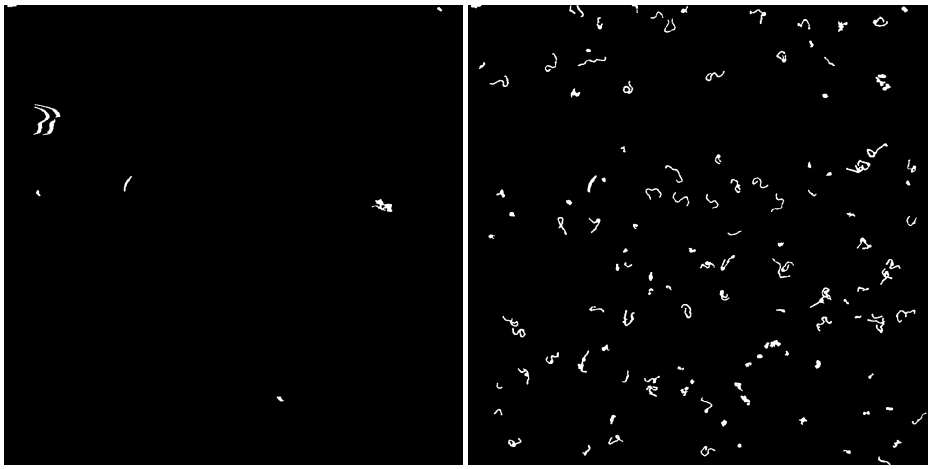
Figure 23: (a)Mask from binary image (b)Mask applied to original image

image and set them to the previous determined background level. The result Figure 23b shows how the large outliers are removed and will therefore no longer contribute to the threshold in a negative way. After this process a second threshold algorithm, in this case the moments threshold (see 3.5), is applied to the original image. Duo to the previous steps the new calculated threshold is no longer influenced by background noise or polluted image regions.

In this case the new threshold is now at a intensity value of 111, compared to a intensity value of 135. Figure 24 shows the improvement in terms of recognized DNA objects on the sample image for type one noise as seen in Figure 20a. By using this approach it is now possible to analyze pictures with the first type of noise while they had to be discarded entirely before.

3.5.4 Limit Threshold

The first two steps focused on keeping the noise and distribution of the height values in a tolerable range on image to image basis. The third and final step of this method aims at limiting the thresholds over the entire dataset. For this technique a larger dataset of at least 50 images is required. The histogram of all 105 test images shown in Figure 25 shows that step one and two already improve the thresholds to a smaller range. Also, only thresholds, which were too high, have been limited by these steps since only outliers in the upper value range have been treated, while lower values have been set to a fixed background. The newly calculated thresholds are in a range between 0.4 and 0.46 which corresponds to pixel values of 102 and 117. Looking at the images with the highest and lowest thresholds, it turns out that they are still prioritizing lower respectively higher pixel values. This results in too small DNA-strands or false positive detections. In order to limit the probability of these cases the permissible range for thresh-



(a) Before **(b)** After

Figure 24: Comparison with previous method, noise type 1

olds is reduced even further. This is achieved by only allowing thresholds in the range of 1.5 times the standard deviation around the mean of the distribution. The range is thereby limited to:

$$[\bar{X} - 1.5 * \sigma, \bar{X} + 1.5 * \sigma]$$

$$= [0.4345 - 1.5 * 0.0129, 0.4345 + 1.5 * 0.0129] = [0.4152, 0.4538]$$

All values outside of this interval are set to the nearest boundary.

As seen in Figure 24 this method effectively detects and removes polluted image regions and ensures a high DNA-strand detection on images with noise type one. Due to the nature and height field of noise of type two this approach can only limit false detections to a certain degree (see Figure 26). Although it was not expected to work on this kind of imperfections at all. While almost all DNA-strands are detected correctly some false positive still exist. In conclusion, the proposed approach improves image quality and detection rate on very noisy images greatly while leaving already well processed images the way they are.

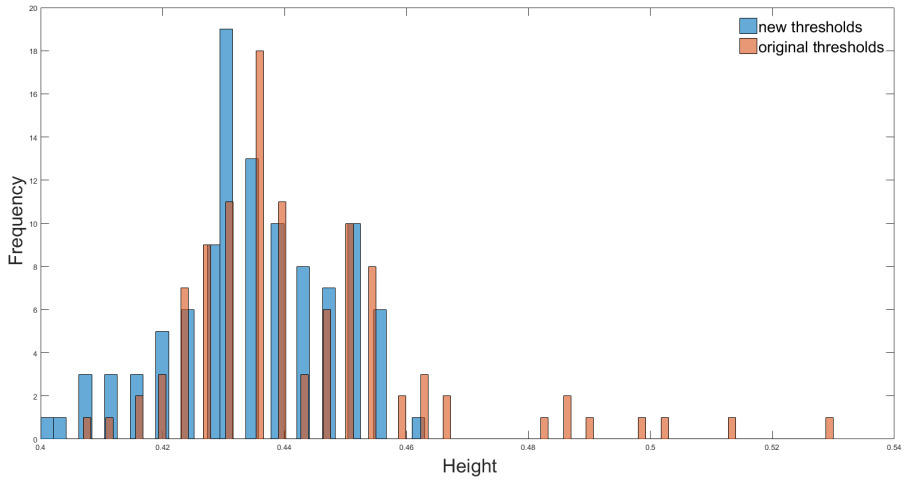
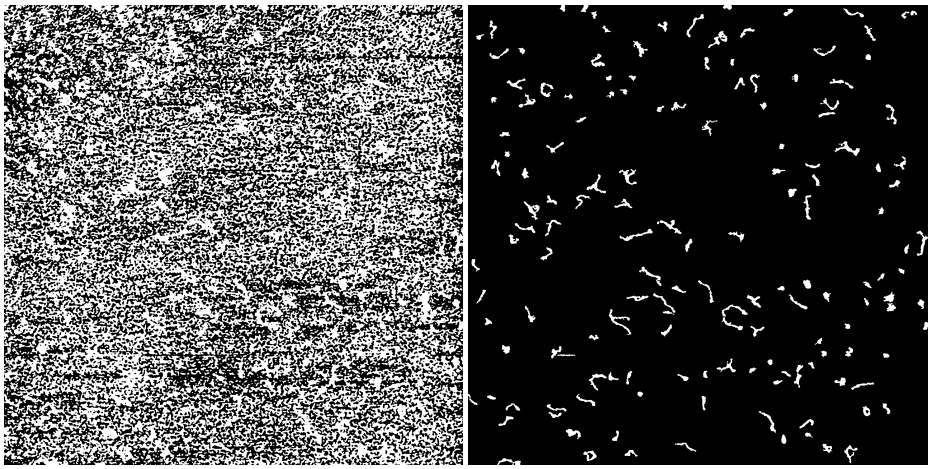


Figure 25: Threshold Distribution over the entire dataset (orange)before and (blue)after step one and two were applied



(a) Before **(b)** After

Figure 26: Comparison with previous method, noise type 2

3.6 Nucleosome Detection

To determine whether the DNA contains a nucleosome and to measure the angle of the two branches wrapped around the nucleosome, an algorithm for nucleosome detection has been implemented. Due to the nucleosomes's appearance, an algorithm to detect bright circles with center (i, j) and radius r, defined by

$$(x - i)^2 + (y - j)^2 = r^2 \tag{1}$$

has been chosen, the MATLAB imfindcircles function. It is based on a Circular Hough Transform which provides robustness in the presence of noise, occlusion and varying illumination. Votes are cast in the Hough space which is made up of a cell for each pixel. Initially each cell is set to 0.

The algorithm considers foreground pixels that have a high image gradient value as candidates for the edges of a circle. For the image data in this project the foreground pixels are bright pixels. As the nucleosomes in the AFM images are blurry and of low contrast in relation to the background, their edges are weak and thus the image gradient threshold value has to be set relatively low. Each edge candidate pixel $p_{edge} = (x, y)$ is then the center point to a voting circle of a radius r, meaning that each edge candiate pixel casts votes in the Hough space to the bins $b = (i, j)$ which according to equation 1 could be the center of the circle and therewith the center of the nucleosome.

The point where locally most voting circles coincide, which equals the bin in Hough space where votes accumulate, is marked as the center point of the actual circle, see Figure 27.

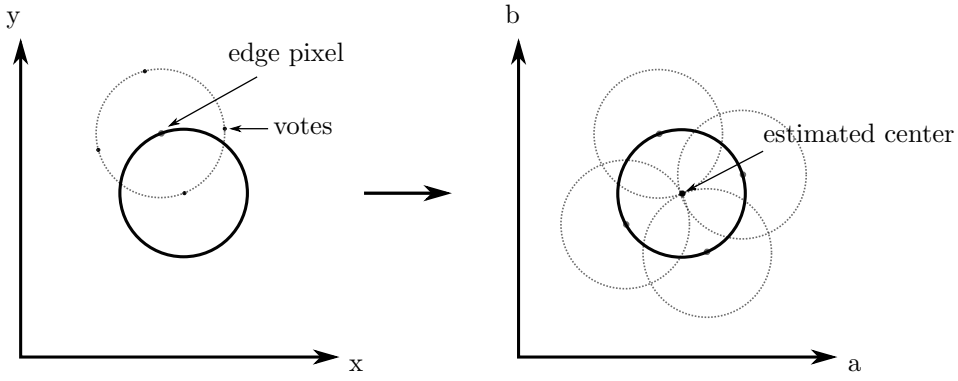


Figure 27: Left: A voting circle (dashed) has its center point at a pixel of high gradient, e.g. on the actual circle's edge. Right: The actual circle that is to be detected (solid) is centered around the point where most voting circles meet.

However, the nucleosomes have different radii ranging between 4 and 6 pixels, such that the voting has to be done for several different radii leading to not just intersecting circles, but intersecting cones, see Figure 28.

As the image gradient direction is known, the problem is slimmed down to intersecting lines in the Hough space instead of the intersecting cones, see Figure 29. This is possible because the gradient direction tells the exact direction in which the center of the circle has to lie and subsequently one does not have to vote for a whole circle of possible center locations but just a single point or respectively for unknown radii, a line. The radius and circle center location is then given by the bin with the most votes, that is where most of the lines intersect.

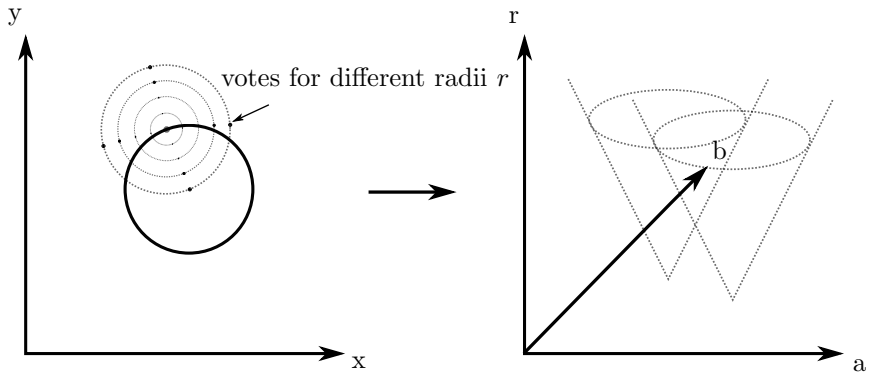


Figure 28: Left: If the radius is unknown, votes will be cast not just to one circle but to many circles of radii r in a given range. Right: Voting circles of different radii add up to voting cones in Hough space.

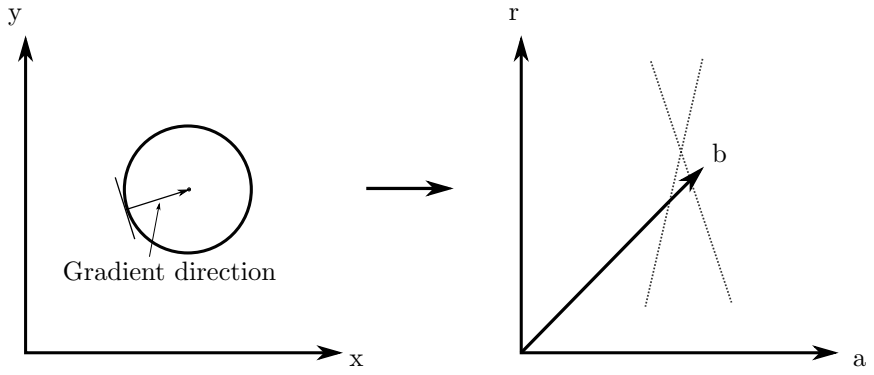


Figure 29: Left: The image gradient gives information as to where the center of the actual circle is placed, hence votes are cast to just a point not to a circle. Right: Voting points for different radii add up to a voting line in Hough space.

Applying this method, the nucleosomes as well as noise of circular shape was detected. As only nucleosomes that are located on a DNA strand are of interest, the other detected circles can be dropped. This is done checking the pixel intensity value of the center points of all detected circles in the binary images. In those images noise is already deleted, such that the circles that were fitted to noise will have a pixel intensity value of zero at their center point. Thus, only circles with a non-zero pixel intensity value are considered as nucleosomes, see Figure 30.

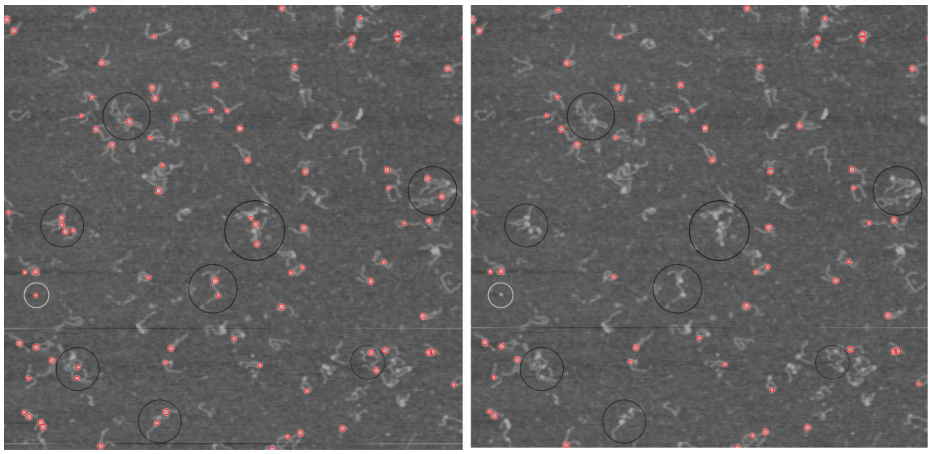


Figure 30: Left: All detected nucleosomes are shown as red circles. Right: The detected nucleosomes after dropping those on overly big DNA strands (marked with black circles) and noise (marked with a white circle).

3.7 Thinning

To obtain an accurate length measurement, the center line of the DNA is extracted. This is done by thinning the segmented DNA down to a single pixel width line corresponding to the skeleton of the DNA which can then be further processed for length and angle measurement. Two thinning algorithms have been tested in the scope of the project.

3.7.1 Hilditch's Sequential Thinning

The first method is based on Hilditch's sequential thinning algorithm [65] and is implemented via MATLAB's `bwmorph` function. For each DNA pixel p the 8-neighborhood $N(p)$ is considered when deciding whether to disregard the pixel or keep it as part of the resulting skeleton. The neighborhood $N(p)$ includes the points $x_i, i = 1, \dots 8$ starting from the point east of pixel p and counting upwards counterclockwise, compare Figure 31. Their values correspond to the pixel intensity, such that

$$x_i = \begin{cases} 1 & \text{if } x_i \text{ is a DNA pixel} \\ 0 & \text{if } x_i \text{ is a background pixel} \end{cases}.$$

The pixels p are scanned from left to right and top to bottom and are considered for deletion if they satisfy the following properties:

1. p is a white pixel.
2. p is not an isolated or end point
3. p is a contour pixel, i.e., p has at least one black 4-neighbor.

x_4	x_3	x_2
x_5	p	x_1
x_6	x_7	x_8

Figure 31: 8-Neighborhood of pixel p.

For all those pixels, the algorithm performs two sub-iterations. In the sub-iterations pixel p is deleted if the following conditions hold:

- **Condition 1** There is exactly one white connected component including white 4-neighbor pixels in $N(p)$:

  $$X_H(p) = 1,$$

  where the crossing number $X_H(p)$ is the number of white connected components in $N(p)$:

  $$X_H(p) = \sum_{i=1}^{4} b_i,$$
  $$\text{where } b_i = \begin{cases} 1 & \text{if } x_{2i-1} \wedge (\overline{x}_{2i} \vee \overline{x}_{2i+1}) \\ 0 & \text{else} \end{cases}$$

  and where x_i are the pixels in the neighborhood $N(p)$ as defined above and the pixel values are treated as boolean values such that $\overline{x}_i$ is the negation of x_i.

- **Condition 2** Pixel p is not an endpoint, branch point or isolated pixel and has to be a boundary pixel:

  $$2 \leq \min\{n_1(p), n_2(p)\} \leq 3,$$

  where

  $$n_1(p) = \sum_{k=1}^{4} \overline{x}_{2k-1} \vee \overline{x}_{2k}$$
  $$n_2(p) = \sum_{k=1}^{4} \overline{x}_{2k} \vee \overline{x}_{2k+1}$$

- **Condition 3** One pixel wide vertical and horizontal lines are not eroded by deletion of pixel p.

– First sub-iteration: $(x_2 \wedge x_3 \wedge \overline{x}_8) \vee x_1$
– Second sub-iteration: $(x_6 \wedge x_7 \wedge \overline{x}_4) \vee x_5$

The two sub-iterations are repeated until the image stops changing. The result are one-pixel-wide ridges representing the center lines of the DNA strands, see Figure 32. It can be observed that noise as well as the nucleosomes result in little branches on the center line. The branches are not needed for the calculation of the DNA's length such that they have to be removed, see Section 3.8.2.2.

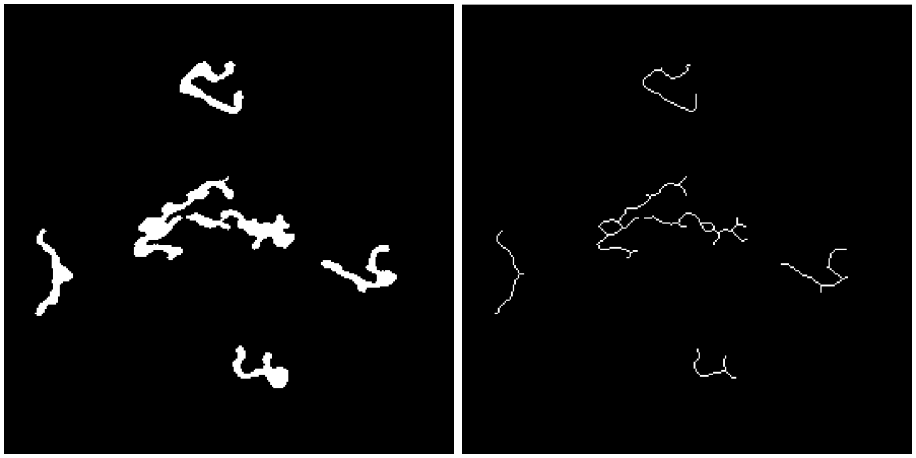


Figure 32: Section of an AFM image after segmentation and binarization (left) and after thinning (right).

3.7.2 Zhang Suen Parallel Thinning

The second method is the Zhang Suen thinning algorithm [53] following the implementation given in [66]. It can be used to attenuate various types of digital patterns [67]. Since it is a parallel method, the new value for any pixel can be computed using only the 8-neighborhood $N(p)$ values known from the previous iteration. The contour pixels are deleted in two sub-iterations until no more changes occur in the image and a one-pixel wide 8-connected skeleton is obtained. A DNA pixel is deleted by setting its value to zero, the label of the background pixels. Pixel p is deleted, if the following conditions hold:

- **Condition 1** There is exactly one white connected component in $N(p)$:

  $$A(p) = 1\,,$$

  where $A(p)$ is the number of white to black crossings in the ordered set $x_1, x_2, \ldots, x_8$ of $N(p)$.

- **Condition 2** Pixel p is not an endpoint or isolated pixel and has to be a boundary pixel:

  $$2 < n(p) < 6,$$

where $n(p)$ is the number of nonzero neighbors of p:

$$n(p) = \sum_{i=1}^{8} x_i$$

- **Condition 3**
  - First sub-iteration: Pixel p is either a south-east boundary point or a north-west corner point.

    $$\overline{x}_1 \vee \overline{x}_7 \vee (\overline{x}_3 \wedge \overline{x}_5)$$

  - Second sub-iteration: Pixel p is either a north-west boundary point or a south-east corner point.

    $$\overline{x}_3 \vee \overline{x}_5 \vee (\overline{x}_1 \wedge \overline{x}_7)$$

The result of this algorithm differs from the result of Hilditch's algorithm as can be seen in Figure 33. The center lines extracted with Hilditch's algorithm are smoother than the Zhang Suen algorithm. Further differences are discussed in Section 4.

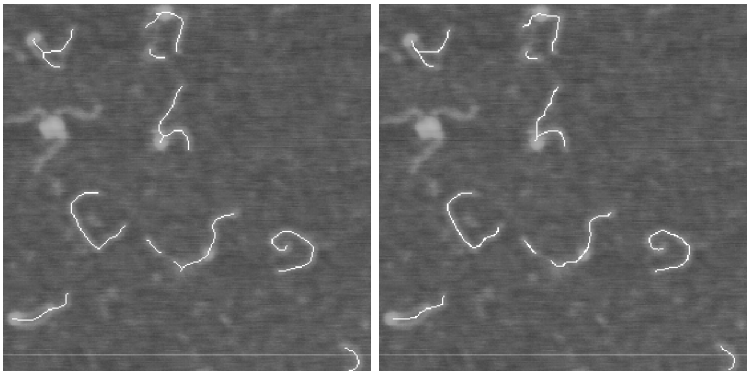


Figure 33: An AFM image section overlayed with the thinned DNA strands in white. Result of Hilditch's Sequential Thinning algorithm (left) and result of the Zhang Suen algorithm (right).

3.8 Length Estimation

At this stage of the program, each DNA object already has been isolated into a small black-and-white image that otherwise only contains this one DNA object, and both thinning algorithms have produced thinned DNA-strands. However, due to the nature of their operations, in most cases the resulting strands not only have two end points, but contain branches and multiple end points. Several examples can be seen in the right panel of Figure 32 above. In order to correctly determine DNA length, it is necessary to erode these branches so that afterwards only DNA strands without any branches and with only two ends are left. These will be called DNA backbones. Consequently, the next step in our routine is the erosion of these noise branches and the extraction of DNA backbones.

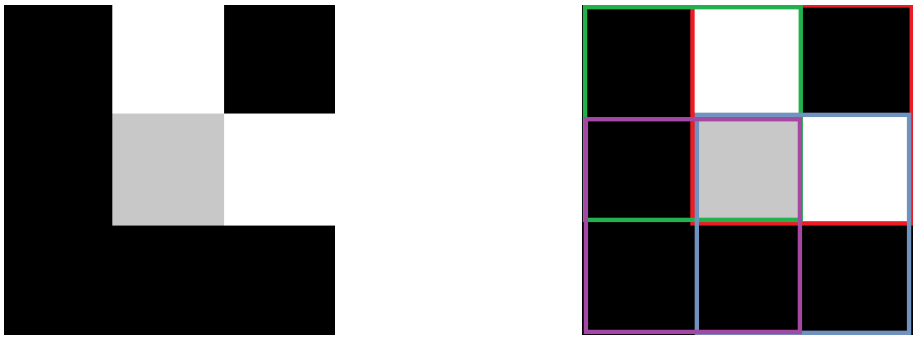
For Estimating the length of DNA fragments, several approaches seemed plausible. Firstly, backbone approximation by spline fitting was one prominent idea. For spline fitting, the DNA strand has to be subdivided into a few piecewise strands that add up into the original strand. Out of every piecewise strand some points are taken and these points are use to interpolate with a so called spline which is a polynom of n-th degree (n equals the number of supporting points minus 1). Normally these splines have to satisfy some more constraints to improve the accuracy and the smoothness of the interpolation. By interpolation of every substring it is possible to calculate the overall length of the object. The challenge of this method is to choose the best substrings and supporting points, and to perform these costly calculations without increasing the overall runtime too much. For these reasons, we decided not to follow this line.

Another approach is to alter the DNA in a way that only one single strand remains and all side strands, which are most probably dirt or noise anyway, and some special points that are explained below are removed. After that the length of this single strand can be determined by simple length estimation algorithms as described by Rivetti [44]. The most effort for this algorithm is altering the DNA string while the actual length estimation afterwards is rather easy. The huge advantage of this method is that by altering the DNA cases can be discovered in which the DNA forms a ring or overlaps with another DNA. Other length estimation algorithms have to take care about this separately. This is one of the methods used in our routine, and it is described in detail in Section 3.8.1.

An entirely different approach is described in Section 3.8.2, however, which represents the other method used in our routine. Here, each thinned DNA fragment is converted into a graph object. On the basis of this, breadth-first-search and shortest path algorithms are used in order to extract the DNA backbone. After subsequent recovery of those pixels that were lost during the thinning process, length calculation is performed with help of the afore mentioned Kulpa Estimator.

3.8.1 Algorithm D: Iterative length estimation

3.8.1.1 Problems of the data



(a) **(b)**

Figure 34: (a) one variant of a so called "L-point"
(b) the four different submasks

In the dataset used for this algorithm often it occurs that the DNA strand is not one single strand but at some points sidebranches can be detected. This makes the length estimation much harder. Also L-points as shown in figure 34a are a common problem since the algorithm has no distinct way to get from one of the outer pixels to the other outer one. In the proposed algorithm this can lead to problems where the parts of the DNA string are deleted that are not intended to or the object can be divided into multiple objects. Both effects are not desirable and are avoided by removing this critical L-points.
Another problem occurs when a branch is deleted. In some cases it can arise that a former branch point becomes an L-point after the deletion of the branch. In that case it has to be deleted, too, so the distinct way through the object is not destroyed again. So for every deleted branch there has also to be a routine that detects newly generated L-points.

(a) **(b)**

Figure 35: (a) is a simple example for two overlapping strands
(b) DNA forming a ring

In figure 35 two special cases are shown. Case 35a shows a simple example for two overlapping DNA strands. Even though this is only an easy example in

most cases where two DNAs overlap some kind of cross can be found. In case 35b a DNA forming some kind of ring can be seen. This is mostly an error in the segmentation process or the DNA is intersecting with itself. All these cases can not be analyzed and are ignored in further processing steps.
For dealing with this two issues a special constraint is introduced. To better understand this constraint it is helpful to realize that a distinct DNA string has only two endpoints and no branchpoints (otherwise there would be a branch and therefore no distinct way through the object). Adding one additional branchpoint now automatically implies that there also has to be another endpoint. To be precise there has to be exactly one more endpoint because otherwise there would be a cross like in example 35a or a ring like in 35b when the branchpoints produces no new endpoint. Secondly, it is important to notice that a ring like in figure 35b has only one endpoint and one branchpoint. Creating a new endpoint would require another branchpoint too.
So what are the implications of these thoughts? Ultimately one will come to the conclusion that a good constraint to check for regular DNA strands is $\#endpoints - \#branchpoints = 2$.
This idea takes in regard that having a lower difference than 2 would imply to have more branchpoints and so some kind of ring in the DNA and having a higher difference would imply some kind of cross.
Of course this constraint will lead to the problem that some cases where noise is forming a cross or the DNA is a ring that could be analyzed anyhow will be thrown out. These cases are mostly DNAs that are in very noisy areas though and the accuracy of the length estimation for these strands would be low anyway. On the other hand it is a huge advantage to recognize molecules which are faulty and to make no false assumptions of their length. In that way statistics like the mean over all DNA strings are not influenced incorrectly.
In the course of the implementation another problem arised. What happens if the cross- and ring structure combine in a way that adds up in a constraint of 2 nonetheless (imagine a ring and two additional intersections)? This problem was eliminated by only looking if a cross occurs. Further explanation can be found later. The case of a DNA forming a ring is still treated by checking with the constraint.
Eventually this method has found a good middle ground between evaluating as many molecules as possible and throwing out DNA one can not analyze in a reliable way.

3.8.1.2 Description of the proposed method

Firstly, the algorithm asserts that the image is not empty. This is just a step of precaution but should be done in every good code nonetheless. If the image is empty a length of 0 is returned and the algorithms immediately terminates.
To avoid border cases of points lying at the edge of the image the image is padded with a single row of zeros. Afterwards essential information is extracted out of the image like width, height and the connected components of the object. Reextracting the connected components is necessary since by padding the pixel

indices are shifted.
The next important step is to create an adjacency matrix of the pixels of the connected component. An adjacency matrix essentially describes for each pixel which other pixels can be reached from this one. In image processing this refers most often to neighbouring pixels in a 8-neighbourhood. If the reader is not familiar with adjacency matrices reading of further papers is strongly recommended. It should be mentioned that it is sufficient to create the matrix by using only the pixels that are part of the object which are stored in a pixel index list and it is not needed to create a adjacency matrix using all image pixels. A speciality of the proposed method is that the adjacency matrix does not only represent the neighbourhood but also the direction in which the neighbour lies (see fig 36d).
The next step is to remove all L-points. L-points are described earlier in this section. The removal is performed by iterating over all object pixels that have exactly two neighbourgs. Points with less neighbours are endpoints and pixels with more neighbours branchpoints. Therefore a 3x3 clipping around the current pixel is cut out and the clipping again subdivided into 4 quadrants (see fig 34b). A pixel is now identified as an L-point if the point is considered an L-point in one of the quadrants (see red quadrant in fig 34b). L-points are deleted from the picture as well as the adjacency matrix.

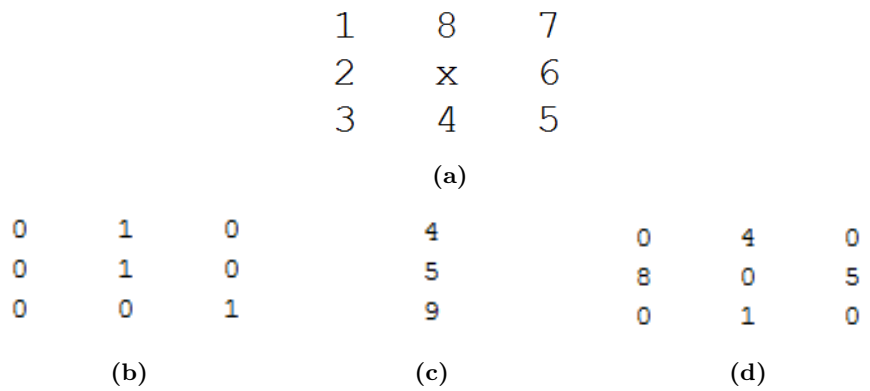


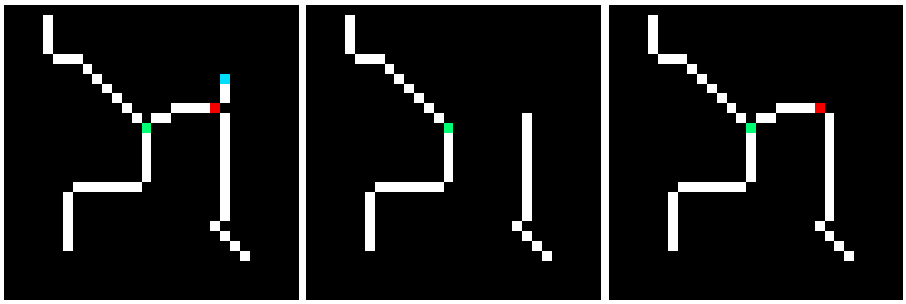
Figure 36: (a) numeration of the neighborhood (b) simple example of thinned strand (c) index list of the image (d) resulting adjacency list
Note that the adjacency list is symmetric only with opposing cases

At this point the branchpoints and endpoints are computed with

```
bwmorph(picture, 'branchpoints')
bwmorph(picture, 'endpoints')
```

respectively. As described before, the segmented strand is tested on its validity by the constraint of $\#endpoints - \#branchpoints = 2$. If a branchpoint now

has exactly 4 neighbours, it is assumed to be a self-intersection which is reliable for the sequential Hilditch-Thinning but not for the algorithm by Zhang Suen. Further explanation and discussion of the results can be found in Sections 3.7.1 and 3.7.2.



(a) **(b)** **(c)**

Figure 37: (a) DNA string with the selected branchpoint (green), another branchpoint (red) and the nearest endpoint (blue) to the green point (b) incorrect deletion of a substring (c) correct deletion

For every valid object the algorithm now iterates over the list of branchpoints until it is empty. With each loop an arbitrary single branch point will be extracted from the mask provided by bwmorph. Using bwdistgeodesic with this singular branchpoint mask, a distance estimation towards each endpoint is calculated. The function is only used in this case since it has no option to calculate the length with more precise estimators (like the Kulpa estimator proposed by [44]) but is reliable enough to give an idea which endpoint is nearest to the current branchpoint that is critical in this step. Removing all pixels of this shortest branch should in theory be rather easy but may lead to unwanted problems. For example a strand with another branchpoint between the used branchpoint and the closest endpoint would result in two separate objects. In figure 37 a correct and incorrect removal of a substring can be seen.
To avoid this, only the pixels from the endpoint to the *closest* branchpoint are removed. For this the adjacency matrix is of great help since it provides the neighbouring pixels. After deleting a point from the image, it has to be deleted from the image as well as the adjacency matrix.
Another, yet unlikely, case would happen if two endpoints are of exactly the same distance from the branchpoints. Should bwdistgeodesic indeed return two valid endpoints, only the first element of the array is selected to avoid critical errors.
At the end of each iteration step the branchpoint and endpoint matrices are updated as well. It is crucial that in the branchpoint mask the deleted branchpoint is altered and not necessarily the one which was initially selected by the algorithm. Also note that the branchpoint is *not* removed from the image but only from the branchpoint mask. The initial branch will still be deleted in a

later iteration step either way, so the order doesn't impact it in any negative way but can prevent critical cases as shown.

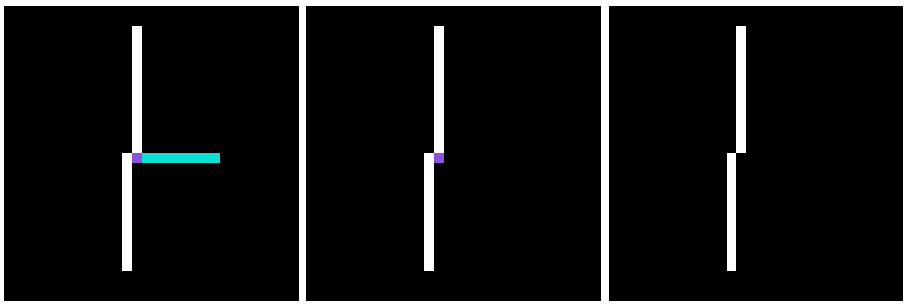


Figure 38: (a) original DNA string (b) without L-point removal (c) with L-point removal

Another problem that showed up during initial testing was that the algorithm sometimes didn't fully remove a certain branch, or so it seemed. This was the result of the branchpoint, after removing the shortest branch, could transform into an L-point. To avoid such behaviour, and assert a singular and unique path from end to end, it has to be checked whether the branchpoint fulfills the L-point criteria mentioned earlier in this chapter. In figure 38 the problem is visualized. Note how the purple point in 38b would lead to a non-unique path. With this step the processing of the strand has been almost completed. If there have been no problems up to this point, the strand seems to be a valid object and is therefore treated as such. In the case that one of the earlier mentioned abortion criteria terminated the algorithm, the dnaObject is then marked as invalid by setting the isValid flag of the object class to 0.
The following steps are similiar to the algorithm described in Sections 3.8.2.3 and 3.8.2.4, and have only been slightly altered. The method is only described here for sake of completeness.
To improve the quality of the final estimate it should also be considered to reestimate the border regions of the strand. With the thinning algorithms, especially the region around the endpoints is usually eroded from each direction which leads to an overall shorter strand in thinned form. Various approaches have already been proposed as already described in Section 2.5.9. Based on the endpoints and their respective neighbouring pixels, a reestimation direction is established by taking a vector in this direction (see figure). By overlaying the original fragment with the thinned one it can now be determined whether there is another pixel in the estimation direction that belongs to the fragment. The process is repeated until it meets a pixel which is no longer part of the fragment or the image boundary is reached.
The last step in the algorithm is then to calculate the partial length of each

fragment, so long as they do have exactly one attached nucleosome. For this a boolean function parameter called dnaHasNucleos was introduced which lets the algorithm decide whether there are any nucleosomes attached at all. The exact number of attached nucleosomes can be referenced by the object's value 'attachedNucleo'.
For these valid objects the length of each partial strand (divided by the nucleosome) is stored which results in an array of length two.
For fragments without any nucleosomes, or more than one attached, the fragment length is stored as one single value. The final length calculation is done by using the Kulpa estimator which provides a more accurate estimate than the classic Freeman estimator [44] for certain DNA lengths.
For both formulas the neighborhood of the pixel is considered: Either the pixels are evenly aligned or are adjoining diagonally. By multiplying the number of each cases with a respective factor a final length can be calculated.

Freeman estimator: $1 * \#EvenPixels + \sqrt{2} * \#OddPixels$
Kulpa estimator: $0.948 * \#EvenPixels + 1.343 * \#OddPixels$

For this purpse the following subroutine was introduced to calculate the final length:

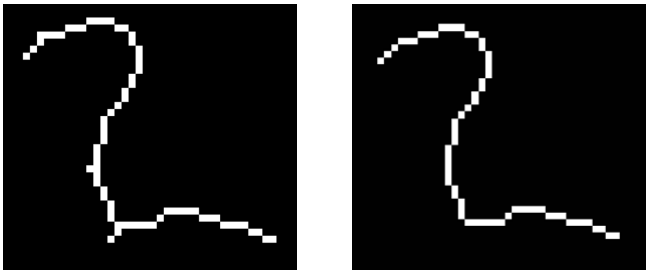
```
calcKulpaLength(IndexList, ThinnedImage)
```

For calculating subsegment-length, this function is simply called twice with the respective partial strand.
The matrix indices of the pixels are split into row and column indices. Following the introduced neighborhood model, a evenly aligned pixel-pair would either have the same row or column indices. By simply substracting the difference of neighboring index-pairs it is possible to accumulate the total number of evenly aligned pixels. The number of odd pixels is easily calculated by substracting the evenly number from the total amount of pixels belonging to the strand. As a last step the length is determined by the aforementioned formula. An example is shown in figure 39.

(a) **(b)**

Figure 39: An example strand with a nucleosome

3.8.1.3 Results



(a) **(b)**

Figure 40: (a) DNA string before processing (b) DNA strand after processing

In figure 40 one can see the strand before and after the before mentioned algorithm processed it. One can clearly see that the L-Point in the upper left corner is removed as well as the two branches in the middle of the string. These results can be reproduced with any valid string of DNA, even if it is more complex. One can conclude that the first goal of the algorithm, which was essentially the pruning of the strand and the removal of the L-points, is working as intended.

The length estimation itself returns reasonable lengths and the general calculated range of the analysed DNA strands is accurate. However, one problem of the Kulpa estimator is that it was originally developed to analyze DNA strands with a length of more than 600 base pairs [44]. The DNA in the used data set has a length of about 660 base pairs. While the Kulpa should be rather accurate for this cases it seems that the length of the DNA is a bit too short for this method and as a result the Kulpa estimator deviates from the original lengths. An exact analysis of this can be found in chapter (reference).
Overall the Kulpa estimator proved to be more stable than the Freeman estimator and for regular cases (no outliers) the deviation was significantly smaller. To be exact the deviation of the Freeman estimator was at least four percent while the Kulpa estimator only deviated by about one percent.
Another goal of the method was to sort out as many invalid DNA strings as possible. With the introduced constraint and the sorting out of DNA strands that have an intersection this aim was achieved and the amount of false positives could be greatly reduced. A disadvantage of this is that some valid DNA is discarded. The advantage of having the security that the result is comparable in its entirety is favourable though. Otherwise deductions like the mean would be influenced by false positives and, therefore, less reliable.
A disadvantage of the algorithm is that the subroutine for determining the partial lengths was originally designed for algorithm C (see 3.8.2) and the embedding in this code was harder than it seemed at first. Since the implementation did not return satisfying results a comparison is not possible at this point.

3.8.2 Algorithm C: Erosion using Breadth-First-Search and Shortest Path Algorithms

Basis of this method are the conversion of DNA objects into graphs and subsequent usage of graph algorithms for branch removal. It consists of the following steps:

1. Exclusion of invalid DNA objects
2. Conversion of pixel lists to graph representations
3. Determination of the two most distant end points
4. Determination of shortest path between those end points

Invalid DNA objects are those that meet any of the following criteria:

- Self-intersecting: The length of an DNA strand that intersects itself cannot be meaningfully determined.
- Wrong size: Since the size of all DNA strands that have been used in the preparation of the AFM probe is known, any detected DNA object can be expected to be in size range of 40 to 150 pixel. If it contains less pixels, it either has been truncated during preparation of the AFM probe or it is no DNA at all. If it is made up from more pixels it either is an artifact that has not been deleted during filtering and thresholding, or it is made up from several overlapping DNA strands.

In all of these cases, length determination does not make sense. Such objects are marked as being invalid and are excluded from further processing.

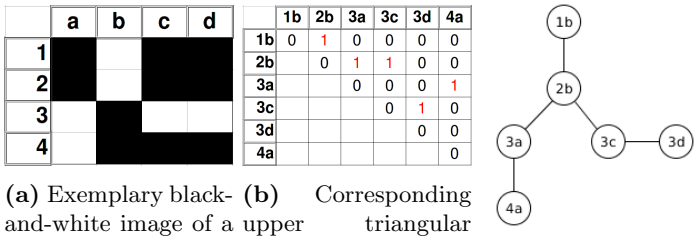
Figure 41: Self-intersecting DNA object. The Euler number of this black-and-white image is 0, indicating that there is at least one object with one hole in the image.

Self-intersection of an DNA object means that the DNA is looping back on itself and then touches or crosses itself (compare Figure 41). A consequence of this is that the DNA object can be considered to be an object with a hole. This, in turn, means that the topology of this object is different compared to an object that does not have a hole, which is reflected in the Euler number of its image. Therefore, for each DNA object the Euler number is calculated using the MATLAB function *bweuler*. It returns the number of objects in an image minus the number of holes. Since for each DNA object an image exists that only contains

this one object, this function efficiently can be used to distinguish between self-intersecting and non-intersecting DNA objects. If the Euler number is 0, then an object with an hole has been detected, and the DNA object is flagged as being invalid. Only if the Euler number is 1, the DNA object will be further processed.

3.8.2.1 Conversion of DNA fragments into graph objects

The next step is the conversion of DNA fragments to graph objects. This approach relies on the fact that, at this stage of the entire process, the so far detected and thinned DNA objects internally are represented as chains of white pixels that belong to exactly one object. Since thinning already has taken place, this approach assumes that any pixel in such a chain is only neighbor to other pixels that represent other links of the chain.



	1b	2b	3a	3c	3d	4a
1b	0	1	0	0	0	0
2b		0	1	1	0	0
3a			0	0	0	1
3c				0	1	0
3d					0	0
4a						0

(a) Exemplary black-and-white image of a small DNA object

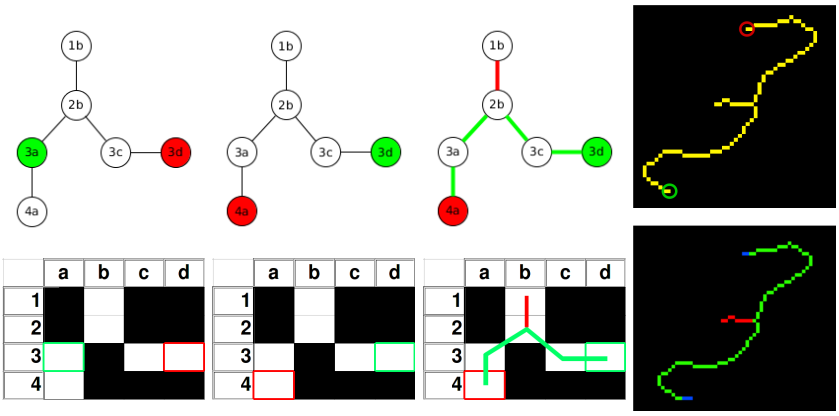
(b) Corresponding upper triangular adjacency matrix

(c) Undirected graph

Figure 42: Simple example for the conversion of DNA objects to undirected graphs. (a) From a small image containing exactly one DNA object, (b) an upper triangular adjacency matrix is created where all ones (red) represent edges between directly neighboring pixels. Only pixels containing information, i.e. white pixels, need to be considered and here, no pixel is neighbor to itself. (c) Such a matrix represents an undirected graph.

Therefore, each DNA object can be transformed into a graph G (Figure 42). The nodes of this graph are those pixels representing the DNA object, while the graph's edges describe the direct 8-neighborhood between pixels (Figure 42b). Since in this case neighborhood between pixels is a bijection, the resulting graph G is undirected, and it suffices to create G from an upper triangular adjacency matrix.

Having created a graph representation for each DNA object, another simple measure allows for the detection of further invalid DNA objects. At this stage of the program, a valid DNA object is expected to be a single DNA strand without any intersections. Its graph will be a tree, i.e. a graph in which each pair of nodes is connected via exactly one, and only one path. However, if smaller DNA strands intersect each other without fully forming any circle, they appear to be one large object that still can be within the size limits. Although a graph of such



(a) First bfs **(b)** Second bfs **(c)** Shortest path **(d)** Example DNA

Figure 43: Determining DNA backbones via two breadth-first searches (bfs) and a shortest path search. (a) On the graph G representing a DNA object, one iteration of breadth-first search (bfs) with a randomly chosen start node (green) will return one node (red) of the pair of nodes that is connected via the longest path within G. (b) Using this node as starting node (green) in a second iteration of bfs will return the other node (red) of this pair. (c) Calculating the shortest path between these two nodes returns the DNA backbone (green edges) while excluding noise branches (red edge). The upper row of panels (a) to (c) depicts results on the graph object introduced in Figure 42, the lower row shows the same results in the respective DNA object representation. (d) A thinned DNA object as detected by the program is shown. The upper image indicates the pair of nodes furthest apart from each other, the lower image highlights the shortest path between them (green), a noise branch (red) and pixel that were lost during the thinning process (blue).

an object also will be a tree, this tree will have a high branching factor, which means that the difference between the number of edges and the number of nodes is larger compared to a valid DNA object as described above. Therefore, any DNA object whose graph representation does not fulfill the following criterion is also excluded from further processing and flagged as being invalid.

$$number\ of\ edges\ -\ number\ of\ nodes \leq 2$$

3.8.2.2 Erosion of Noise Branches

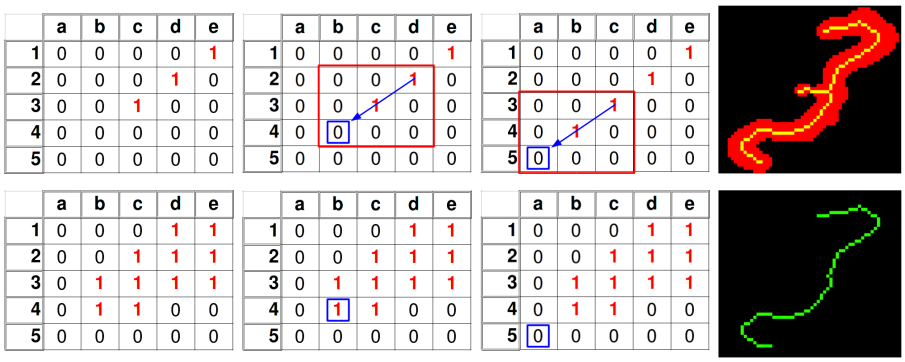
Due to their graph representations, it is now possible to determine exactly those end points of the DNA objects that are the furthest apart (see Figure 43). This is equivalent to determining each graph's diameter, i.e. the largest distance between any two connected nodes of a graph. It can be achieved by successively performing two breadth-first searches (bfs) on G. MATLAB already provides such a functionality (*bfsearch*) that is optimised for MATLAB internal graph objects. Therefore, we specifically converted DNA objects into MATLAB graph objects. For the first bfs (compare Figure 43a), a start node is chosen randomly.

Due to the nature of this algorithm, the last node to be discovered can only be a node that has the largest distance to the starting node. If there are several nodes with the same largest distance, which of these is discovered last depends on the implementation of the bfs algorithm. Because all of them are equally far apart from the start node, any of these nodes is considered a correct result.
This node, however, can then be used as start node in the second run of bfs (compare Figure 43b). Since bfs will again return the node with the largest distance to the start node last, we now know the exact pair of nodes that are furthest apart from each other in G.
This information next is used to find the shortest path between these two nodes in G (see Figure 43c). Also here an optimized implementation provided by MATLAB was used, *graphshortestpath*. Since all graphs are undirected with uniform edge weights of 1, a shortest path algorithm specifically designed for this type of graph was used that has an overall time complexity of O(N+E), with N being the number of nodes and E being the number of edges. The resulting paths represent DNA backbones excluding noise branches, since for a shortest path between two nodes it holds that it connects its two extremal nodes (start and target node) in G via a path that cannot be shortened without severing the connection between both nodes.

3.8.2.3 Recovery of lost pixels

Having determined the true DNA backbones, it next is necessary to restore pixels that have been lost from the DNA fragments during the thinning process and that are essential for length determination. Such pixels are those that lay at the end of an DNA fragment, as can be seen in the upper image of Figure 44d. This loss cannot be avoided since both tested thinning algorithms cannot differenciate between pixels that are part of the end of a fragment and pixels that constitute some other part of the DNA fragment.

An useful and easy to implement approach already has been indicated in Section 2.5.8. Each DNA backbone only has two ends by definition. So, for each of a DNA backbone's end, consider the last two pixels as depicted in the upper panel of Figure 44b. The relation of those two pixels towards each other within the 8-neighborhood of the end pixel of the two identifies an unambiguous position at which a pixel possibly could have been lost. The information whether a pixel has been lost or not can easily be retrieved by comparing the pixel at this position with the pixel at the same position in the image of the not-yet-thinned DNA fragment (see Figure 44b, lower panel). If the pixel at that position is part of the DNA fragment, it has to be restored, and can now be considered to be the new end point of this DNA fragment end. This process is repeated until no pixel can be restored anymore. At this point, the DNA backbone is complete, and its length can be computed.



(a) Initial state **(b)** First iteration **(c)** Second iteration **(d)** Result

Figure 44: Recovery procedure for pixels lost during DNA thinning.
The upper row shows a thinned DNA fragment's end, the lower row the respective DNA's state before thinning. (a) After determination of the DNA backbone, those pixels that were lost from the fragment's ends during the thinning process need to be recovered. One of those ends is schematically depicted. (b) In the first iteration, a direction is derived from the last two pixels of one end. The thick DNA image is used as reference to find lost pixels. (c) In the second iteration, the newly discovered pixel is used to determine whether another pixel needs to be recovered. (d) The upper image shows an overlay of thin and thick DNA, clearly indicating that pixels were lost. The lower image shows the same DNA strand after pixel recovery.

3.8.2.4 Final Length Estimation

Length computation generally is done as has already been described in Section 2.5.9. Because Rivetti and Codeluppi [44] showed that the Kulpa Estimator has higher accuracy than the Freeman Estimator, is easy to implement and has only low impact on the overall runtime, we decided to use this measure for DNA fragment length calculation, as well.

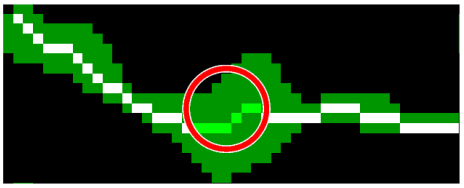


Figure 45: Intersection between a DNA backbone and the nucleosome of its DNA fragment.
Dark green pixels represent the not-yet-thinned DNA fragment with its bound nucleosome, and its DNA backbone is represented by white and light-green pixels. The nucleosome, as it was detected by the program, is shown in red.

Length estimation has to be done differently for DNA fragments with and without nucleosome, respectively. For a DNA fragment without nucleosome,

simply its entire DNA backbone needs to be considered. In the case of an DNA fragment with nucleosome, however, the DNA backbone has to be divided into exactly those two parts that point away from the nucleosome, i.e. the nucleosome's both arms. This can be achieved by using the same method that is used during angle measurement (compare Section 3.9). Intersections of the DNA backbone with the circle (red) which is defined by the radius of the nucleosome is calculated as visualized in Figure 45. Then, those parts of the backbone that are inside this circle are temporarily deleted (light green pixels), which results in only two arms remaining (white pixels). For each of these arms, now, the length is calculated similarly as it is done for the entire backbone in case of free DNA fragments. In the unlikely case that an intersection between the detected nucleus and the DNA backbone cannot be computed, no length will be determined.

3.9 Angle Measurement

For the DNA strands with a nucleosome, the angle between the two DNA branches is measured. DNA strands with more than one nucleosome will not be assigned an angle measurement neither will DNA strands without nucleosome. Furthermore, if the nucleosome is positioned at the end of the DNA strand as depicted in Figure 46, the angle cannot be measured either.

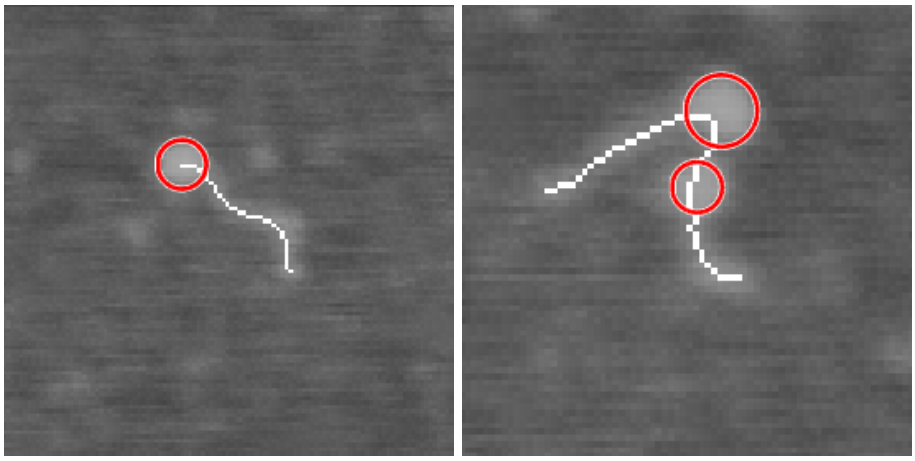


Figure 46: DNA strand with the nucleosome on the strand end (left) and more than one detected nucleosome (right). These cases are not considered for angle measurement.

To discard those cases a mask is set up which is zero at all pixels belonging to the nucleosome and one elsewhere. Multiplication of the mask with the image of the DNA's center line, that is the thinned DNA after erosion of all branches, leads to an image of the DNA branches. After applying a connected component analysis each branch is represented by one connected component. All samples

with a number of connected components unequal to two will be dropped and only the remaining samples with two connected components, respectively branches, are considered for the angle measurement. Two ways to measure the angle between the two branches of the DNA have been implemented in the scope of this project.

3.9.1 Measurement over Nucleosome Center

The first method for the angle measurement is to determine the intersection of the DNA branches and the circle fitted to the nucleosome, draw one line from each intersecting point to the nucleosome center and then calculate the angle between those two lines. This approach has been described in [68]. To determine the intersection of the DNA branches with the circle fitted to the nucleosome, for both branches the Euclidean distance of each of its pixels to the DNA's nucleosome center c is measured. The pixel with the smallest distance d equals the intersecting pixel points p of this branch. The angle θ between the two lines connecting the intersecting points p_1 and p_2 with the nucleosome center c is calculated by

$$\theta = \arccos \frac{a \cdot b}{||a||||b||}\,, \qquad (2)$$
$$\text{where } a = c - p_1 \text{ and } b = c - p_2\,.$$

However, it turned out that the resulting angles are in many cases not representative for the actual branch angles, see Figure 47.

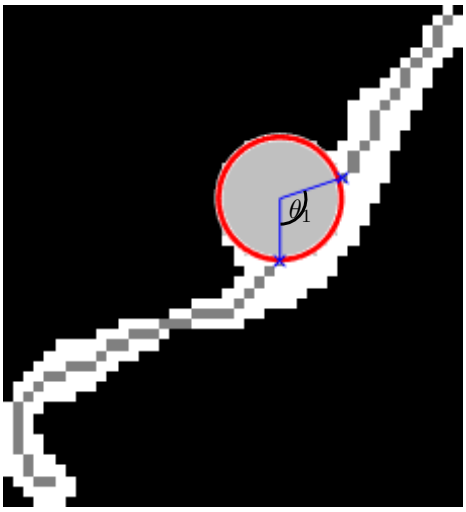


Figure 47: The angle $\theta = \theta_1 = 108^\circ$ between the two DNA branches measured with method 1.

3.9.2 Measurement over Fitted Lines

The second method for the angle measurement follows [69] fitting a line to the first part of a few micrometers of each branch and calculates the angle between those two fitted lines. The first step is to select the pixels belonging to the first part of the two branches.

Let

$$r_{max} = r + s\,,$$

where r is the nucleosome radius, estimated in Section 3.6 and s is the value determining the size of the first part. A mask which is zero outside of a circle of radius r_{max} around the nucleosome center is applied to the image of the DNA branches leading to the pruned DNA branch stumps.

Again a connected component analysis is performed on the pixel set, returning two pixel sets, each representing one of the two branch stumps. The next step is to fit one line to each branch. This is done by computing a least squares polynomial of order one for the given branch points, returning a slope m and an offset t for each branch line. Subsequently the angle between the two lines is calculated according to Equation (2), which requires the intersecting point $c = [c_x\, c_y]^\top$ of the two fitted lines given by:

$$\begin{aligned} c_x &= -\frac{t_1 - t_2}{m_1 - m_2} \\ c_y &= m_1 c_x + t_1 = m_2 c_x + t_2\,. \end{aligned}$$

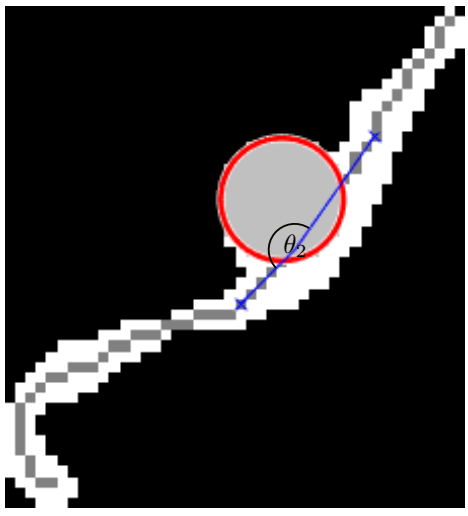


Figure 48: The angle $\theta = \theta_2 = 170^\circ$ between the two DNA branches measured with method 2.

3.10 Runtime Optimization

The main pipeline of the program (see section 3.2) takes up a total amount of 405 seconds for 105 images with a resolution of 1920x1920 on a standard workstation. That means each image needs about 3.9 seconds to be processed from a raw image to a fully evaluated binary image with detected features as described in section 3.2. 3.9 seconds is already a short time period but the use cases for this kind of software usually contains processing thousands of these images. Therefore "the faster the better" applies as long as the quality of the results is not reduced. In order to optimize the runtime three steps were taken:

1. optimize the use of MATLAB commands and indexing
2. parallelize the execution of the pipeline and use multiple CPU cores
3. use GPU for time consuming function calls

To locate time consuming factors the MATLAB profiler [6] was used.
At first inefficient array and matrix indexing like:

$$list(find(list > value))$$

was optimized by using logical indexing instead of the find function. Also some for-loops could be accelerated by using vector or matrix operations which MATLAB handles very well. Since five different developers worked on the program some values, such as the connected Components were calculated more than ones which could be improved by storing the values as an instance variable. This process already improved the executing time from 405 seconds down to 350 seconds over the entire dataset(figure 49) or from 3,9 seconds to 3,4 seconds per image.

As a second step the pipeline was parallelized such that multiple images could be processed at once. Using the the parfor [7] command from the MATLAB parallel computing toolbox [8] multiple workers/threads are started where each worker can handle one loop pass at a time. The number of workers is limited by the number of CPU Cores available on the workstation used which were six in this case. It is important to ensure that all workers only access disjoint parts of the data they work on. Ideally the processing time is divided by six when not accounting for the multithreading overhead. Practically the processing time is quartered which means gong down from 350 seconds to 87 seconds over all and from 3.4 seconds to 0.83 seconds per image (figure 49). Such an improvement is only achievable when using large datasets of more than 100 images.

The MATLAB parallel computing toolbox also provides GPU implementations for some functions such as bwmorph [9] which is used during the thinning

[6] http://www.mathworks.com/help/matlab/matlab_prog/profiling-for-improving-performance.html
[7] http://www.mathworks.com/help/distcomp/parfor.html
[8] http://www.mathworks.com/products/parallel-computing/
[9] http://www.mathworks.com/help/images/ref/bwmorph.html

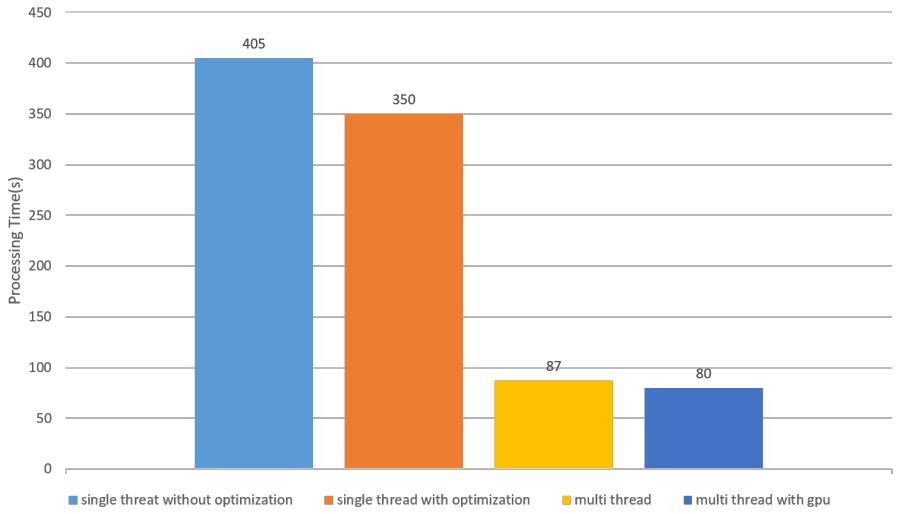


Figure 49: Processing time over 105 images on a workstation (6 CPU Cores)

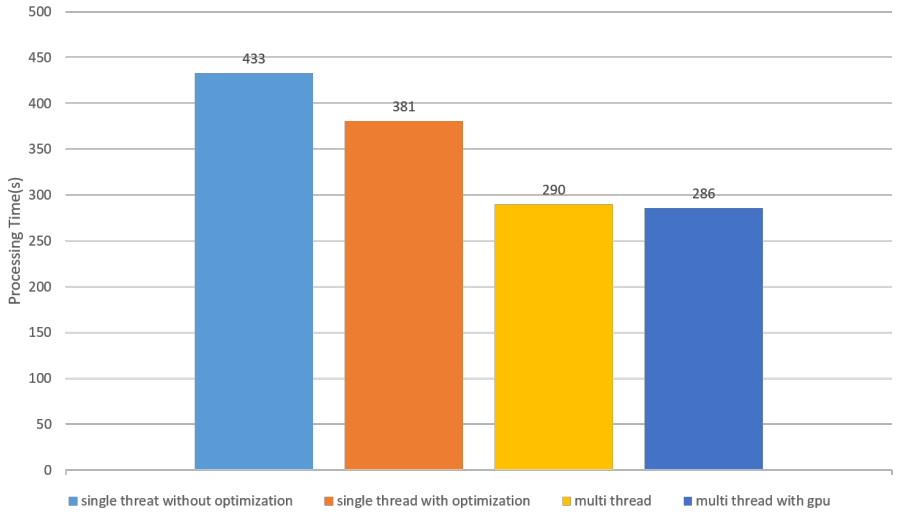


Figure 50: Processing time over 105 images on a consumer laptop (2CPU Cores)

of the DNA strands (see section 3.7). This approach reduced computation time from 87 seconds to 80 seconds respectively from 0.83 seconds to 0.76 seconds per picture.

All steps combined reduce execution time by 80.2 % on a standard workstation. Using a consumer grade laptop with two CPU cores holds far less potential for performance optimization. Still, computing time can be reduced by 34% over the entire dataset which corresponds to a performance gain of 1.4 seconds per

image or an improvement from 433 seconds to 286 seconds for 105 images(figure 50). These values improves by the size of the dataset because the multithread and GPU overhead is independent of the size of the dataset.

4 Results

4.1 Evaluation

To evaluate the performance of our four algorithms we use a test data set of six images generated as described in chapter 2.1. These images (figure 51) are randomly selected of data that are of a sufficiently high quality to be relevant for drawing a biological conclusion by a manual evaluation. AFM images with a much too high DNA density are not considered. An example of this is shown in figure 52.

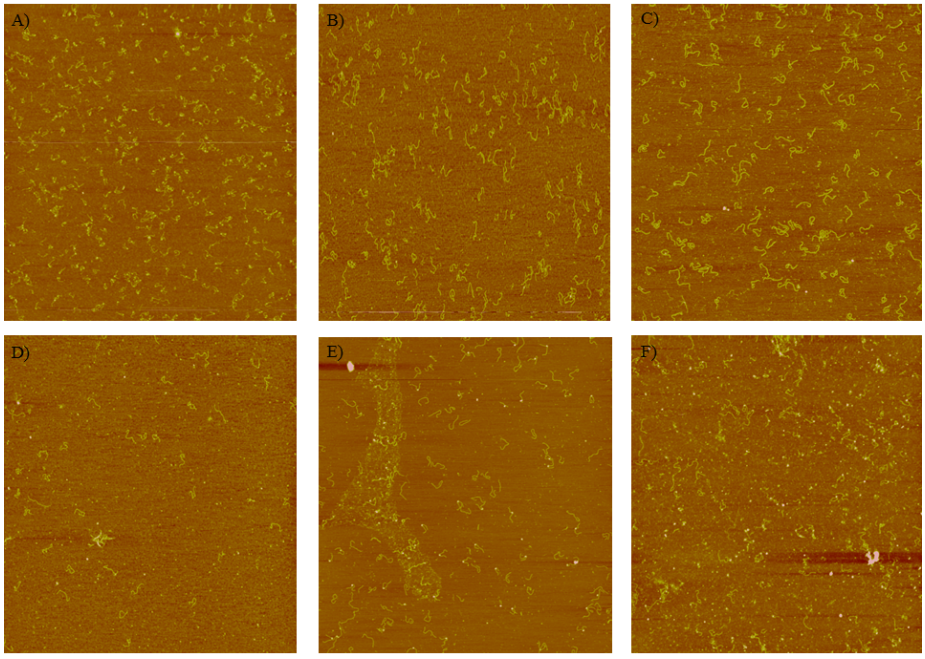


Figure 51: Images used for evaluation of the algorithms. Images are selected of data that are of a sufficiently high quality to be relevant for drawing a biological conclusion.

In the following the two algorithms using breadth first search for length determination are referred to as algorithm C and the two using Kulpa-estimator for length determination are referred to algorithm D. The additional description fast or slow gives information on the applied thinning method. Fast means Hilditch‘s sequential thinning and slow means Zhang Suen‘s thinning. The differences of our algorithms are listed in table 1.

Each of the three persons generating test data determines three times the length and the angle of the same ten objects for examination the reproducibility of the manual measurements. The results are shown in table 2. The variance of the manual length determination is relatively low suggesting a good precision of

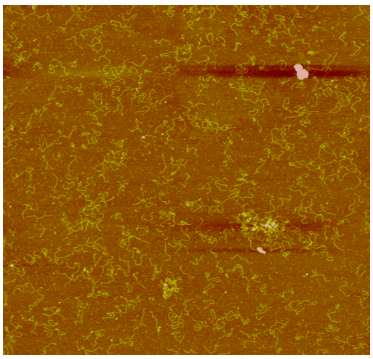


Figure 52: Example for an image not used for algorithm evaluation. An image like this was not considered to the evaluation of our algorithms due to a very poor quality. The DNA density is much too high and therefore such an image has no biological relevance.

Table 1: Overview of the algorithms. The four algorithms differ in their implemented thinning or length determination method.

Abbreviation	Thinning method	Length determination method
Algorithm C fast	Hilditch	Graph-based approach
Algorithm C slow	Zhang Suen	Graph-based approach
Algorithm D fast	Hilditch	Iterative length estimation
Algorithm D slow	Zhang Suen	Iterative length estimation

the method. However, the manual angle measurement seems to be much more inexact due to a very high standard deviation.

Each output of our four algorithms is compared to the test data set. Both for manual and algorithmic measurements, objects with a determined length ≤ 130 nm are not considered as DNA. Figure 53 shows the amount of correct detected free DNA molecules, objects falsely identified as DNA and nucleosome bound DNA wrongly classified as free DNA. Algorithm D slow detects no free and no bound DNA. Therefore, this algorithm is not taken into further considerations within this evaluation. 54.2 % (algorithm C slow), 49.6 % (algorithm C fast) and 49.0 % (algorithm D fast) of 349 free DNA molecules within the test data are correct identified. Meanwhile, 29.5 % (algorithm C slow), 32.7 % (algorithm C fast) and 30.5 % (algorithm D fast) of the identified objects are falsely detected as free DNA.

Figure 54 shows the same statistics for nucleosome bound DNA. 32.7 % (algorithm C slow), 30.6 % (algorithm C fast) and 28.6 % (algorithm D fast) of 196 bound DNA molecules within the test data are correct identified. Meanwhile, 36.0 % (algorithm C slow), 28.4 % (algorithm C fast) and 35.3 % (algorithm D fast) of the identified objects are falsely detected as bound DNA.

For each algorithm, the computed length of the correct identified DNA

Table 2: Reproducibility of the manual measurements. The length of ten and the angle of seven objects are measured three times by each of the three persons generating test data.

Object	Average of the length [pixel]	Standard deviation of the length [pixel]	Average of the angle [°]	Standard deviation of the angel [°]
1	95.1	2.5	-	-
2	95.1	3.5	-	-
3	81.3	3.3	75	19.5
4	79.9	3.3	56.3	26.3
5	95.7	1.7	-	-
6	79	2.5	69	21.5
7	78.9	2.1	56.7	39.4
8	79.4	2.9	68	20.6
9	79.4	2.9	68	20.6
10	76.9	3.1	113.1	20.8

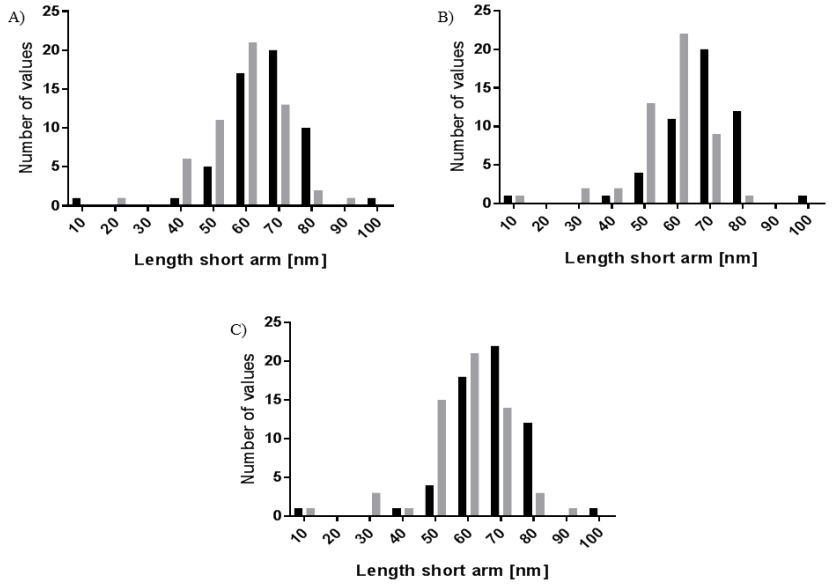


Figure 53: Classification of the objects identified by the algorithms as free DNA. Green bars indicate correct detected free DNA molecules, red bars indicate objects falsely identified as DNA and red hatched bars indicate nucleosome bound DNA wrongly classified as free DNA. The black line marks the actual amount of free DNA in the test data set.

molecules is plotted against the corresponding manual determined length. Free DNA and nucleosome bound DNA is considered separately. The scatter plots

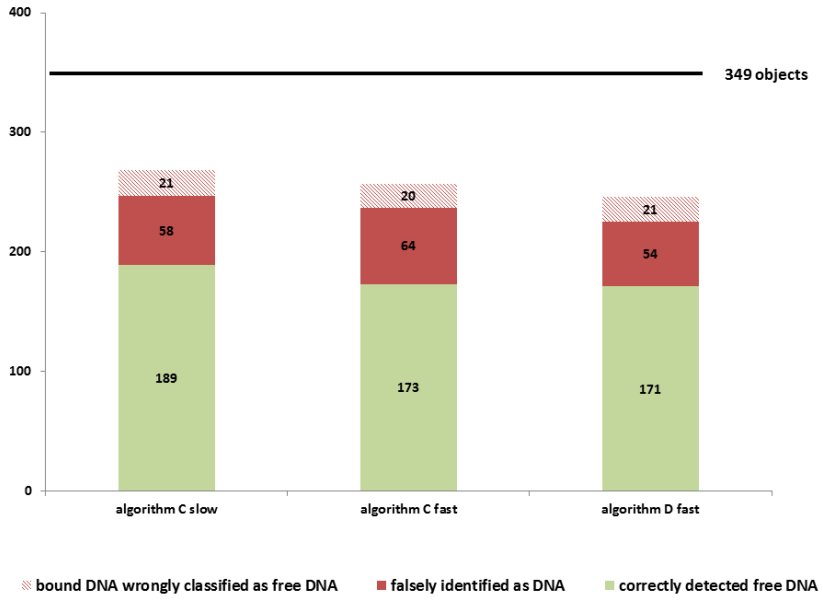


Figure 54: Classification of the objects identified by the algorithms as nucleosome bound DNA. Green bars indicate correct detected bound DNA molecules, red bars indicate objects falsely identified as DNA and red hatched bars indicate free DNA wrongly classified as bound DNA. The black line marks the actual amount of bound DNA in the test data set.

for automatic against manual measurements are shown in figure 55 and 56.

The scatter plots and the according R2 indicate that using breadth first search for length determination combined with Hilditch's sequential thinning is best to measure more exact lengths of DNA molecules. For each algorithm the angle distribution of all correct identified nucleosome bound DNA molecules is shown on figure 57. Also the corresponding angle distributions of the manual evaluation are visualized.

The distribution of the angles determined by the algorithms seems to be relatively equal between 20 ° and 120 °. Lower and higher angles are relatively rare. Manual measured angles have a peak between 50 ° and 90 ° but also do not really conform to a normal distribution. These observations make a meaningful evaluation impossible. Synthetic test data are therefore needed. Furthermore, for each algorithm the distribution of the length of the short arms is shown on figure 58. Again, just correct identified nucleosome bound DNA molecules are considered. The corresponding short arm length distributions of the manual measurements are also visualized.

The distributions of manual measured short arm lengths and short arm lengths determined by the algorithms are very similar. All conform to a normal distribution with a low standard deviation and a mean between 60 and 70

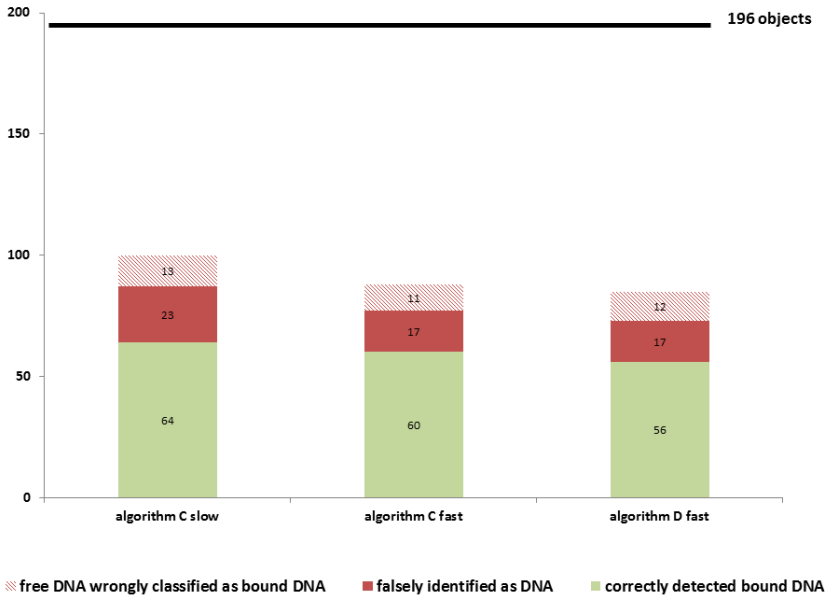


Figure 55: Scatter plots for automatic against manual length determination of free DNA. The lengths of free DNA molecules determined by manual measurements are plotted against the corresponding length calculated by algorithm A) C fast B) D fast C) C slow. Different colors of the points indicate the six different images in our test data set. The black line is the line through the origin and the red line is the linear fit of the scatter plot. Just DNA molecules identified correct by the algorithms are considered in this part of the evaluation.

nm. These observations suggest that our method to measure the length of short arms works very well and precise. Table 3 and 4 give a final overview of all evaluation results for free DNA and bound DNA. Formula 3, 4 and 5 describe the calculation of the recovery rate, true positive rate and false positive rate.

$$recovery\ rate = \frac{objects\ correct\ identified\ by\ the\ algorithm}{objects\ identified\ by\ manual\ measurements} \tag{3}$$

$$true\ positive\ rate = \frac{objects\ correct\ identified\ by\ the\ algorithm}{objects\ identified\ by\ algorithm} \tag{4}$$

$$false\ positive\ rate = \frac{objects\ false\ identifies\ by\ the\ algorithm}{objects\ identified\ by\ algorithm} \tag{5}$$

The results are very similar for all three algorithms and in most cases they differ only minimally. It seems to be that breadth first search for length determination and Zhang Suen‘s thinning causes an increased object detection. The combination of both methods (algorithm C slow) has the best recovery rate. However, the true positive rate and false positive rate seems to be unaffected.

Table 3: Overview of the evaluation results for free DNA molecules. Algorithm D slow detects no free DNA. Therefore, this algorithm is not taken into considerations of the evaluation. Manual measurements identified 349 free DNA molecules with a mean length of 220.0 ± 37.1 nm in our test data set.

	Algorithm C fast	**Algorithm D fast**	**Algorithm C slow**
Identified objects	257	246	268
True positive objects	173	171	189
Recovery rate	0.5	0.49	0.54
True positive rate	0.67	0.7	0.71
False positive rate	0.33	0.3	0.29
Correlation coefficient with manual length determination (R)	0.79	0.72	0.63
Mean of true positive objects	213.7 ± 28.1 nm	203.1 ± 25.4 nm	205.8 ± 31.1 nm
Mean of all identified objects	201.9 ± 34.4 nm	193.5 ± 31.4 nm	197.8 ± 34.4 nm

Table 4: Overview of the evaluation results for nucleosome bound DNA. Algorithm D slow detects no bound DNA. Therefore, this algorithm is not taken into considerations of the evaluation. Manual measurements identified 196 nucleosome bound DNA molecules with a mean length of 178.4 ± 23.8 nm in our test data set.

	Algorithm C fast	**Algorithm D fast**	**Algorithm C slow**
Identified objects	88	85	100
True positive objects	60	56	64
Recovery rate	0.31	0.29	0.33
True positive rate	0.68	0.66	0.64
False positive rate	0.32	0.34	0.36
Correlation coefficient with manual length determination (R)	0.75	0.7	0.75
Mean of true positive objects	159.4 ± 42.4 nm	160.0 ± 44.3 nm	160.4 ± 41.1 nm
Mean of all identified objects	161.9 ± 39.1 nm	166.8 ± 48.9 nm	166.4 ± 45.6 nm

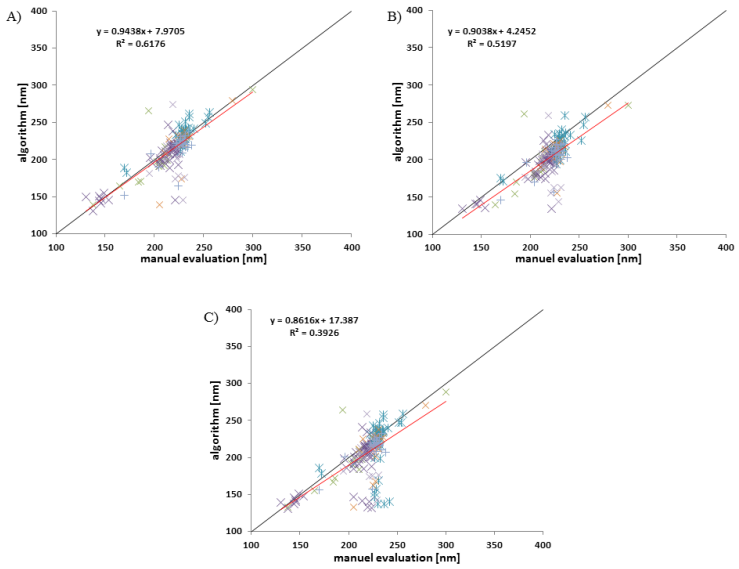


Figure 56: Scatter plots for automatic against manual length determination of nucleosome bound DNA. The lengths of nucleosome bound DNA molecules determined by manual measurements are plotted against the corresponding length calculated by algorithm A) C fast B) D fast C) C slow. Different colors of the points indicate the six different images in our test data set. The black line is the line through the origin and the red line is the linear fit of the scatter plot. Just nucleosome bound DNA molecules identified correct by the algorithms are considered in this part of the evaluation.

For free DNA molecules the true positive rate decreases from algorithm C slow via algorithm D fast to algorithm C fast. But for nucleosome bound DNA it is the reverse. The best correlation with manual length determination is reached by algorithm C fast (Hilditch's sequential thinning + breadth first search) for both object types. Algorithm C slow has for free DNA a significantly worse correlation than the other algorithm. However, for nucleosome bound DNA is this not the case. The mean length of all DNA molecules determined by the algorithms is for both object types always around 15 nm shorter than the manual determined mean value.

Due to problems during execution of different steps of the algorithm and unsatisfactory results, we only evaluated the Hilditch thinning method combined with the length determination method C and D. For the further comparison of the two algorithms with the manually measured results, we calculated independent mean values in reasonable ranges. Only free DNA molecules between 130 and 240 nm were considered in order to exclude fragments that are not important for our biological evaluation. This range has enough space for fluctuation around the expected value between 190 nm and 225 nm. For histone bound DNA, a range between 100 and 223 nm was chosen because the mean

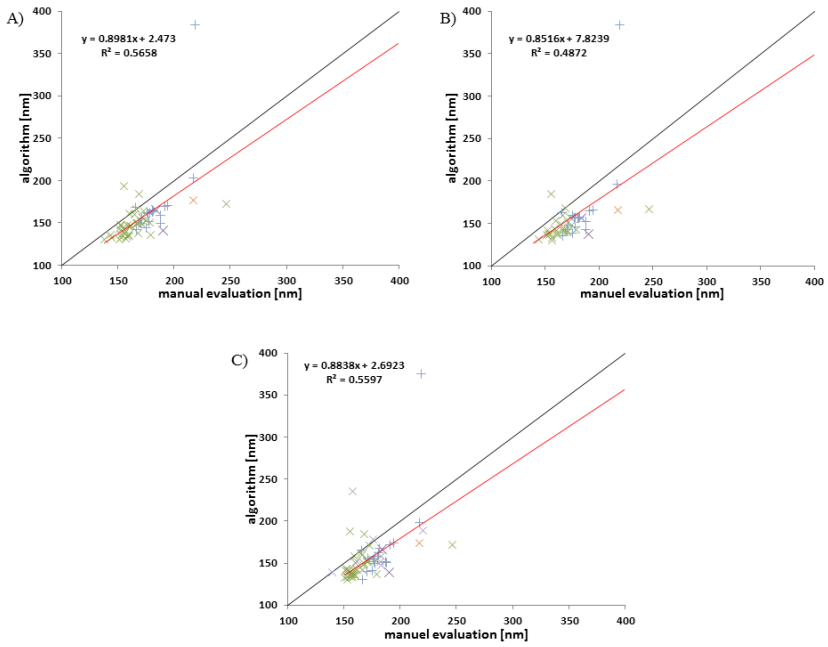


Figure 57: Angle distribution of correct identified nucleosome bound DNA. The number of values is plotted against the angle between the short and long arm of nucleosome bound DNA molecules determined by algorithm A) C fast B) D fast C) C slow (gray bars). The distribution of the corresponding manual measured angles is also shown (black bars) in the diagrams.

value of bound DNA should be shifted to shorter distances. All valid objects of the eight test data images in the described nm ranges were used for the independent mean values (see table 5). The mean values for free DNA, histone bound DNA and the short DNA arm are similar to the calculated mean values of the validation where only correlating values were considered. The length of free DNA, bound DNA and the short DNA arm are shorter in both automatic length determination methods. For this comparison, manually determined values are not corrected. The table also shows the mean values of the radii of the nucleosome core measured automatically and manually. They are lying close together around a value of 11 nm. This results in nearly twice the theoretical diameter of the nucleosome core particle with which the length of the manual evaluation was corrected so far. These results show that the correction for the manual data set was too short. Considering the actual distance on the images, the lengths of free and bound DNA of the manually and automatically generated data are comparable.

To get a better idea of the false positively (FP) detected nucleosomes of the data set with our algorithm without manual review, we had a look on six images containing only free DNA. Table 6 shows the results of the algorithm

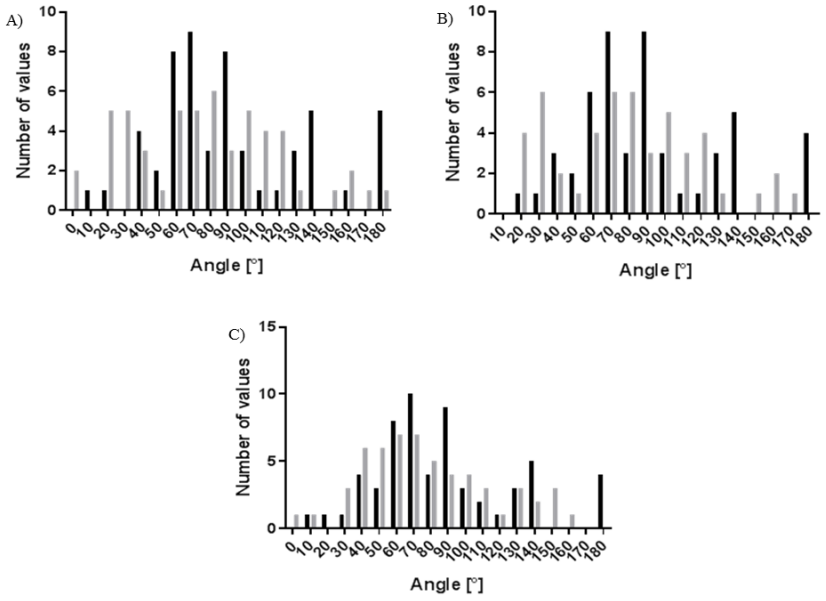


Figure 58: Short arm length distribution of correct identified nucleosome bound DNA molecules. The number of values is plotted against the short arm length of nucleosome bound DNA molecules determined by algorithm A) C fast B) D fast C) C slow (gray bars). The distribution of the corresponding manual measured short arm length is also shown (black bars) in the diagrams.

Table 5: Mean values of the length of the free DNA fragments, bound DNA fragments, the short DNA arm of bound DNA and the radius of the nucleosome core particle in the range from 130 nm - 240 nm for free DNA and 100 nm – 223 nm for bound DNA. The values were determined manually for eight test data images and compared with the automatically measured length with algorithm C and algorithm D. Both algorithms work with the Hilditch thinning method include different length determination methods.

	Manual evaluation [nm]	**Method C [nm]**	**Method D [nm]**
Free DNA	218	196	189
Bound DNA	176	147	145
Short DNA arm	69	49	51
Radius Nucleosome	10.4	11.7	11.6

in the proposed length ranges. The numbers of wrongly detected nucleosomes were around 30 objects compared to 630 overall detected objects on the images. Similar to the comparison of the manual and automatically data, the algorithm detected nucleosomes on these images where no nucleosomes should be detected.

The here measured false positive rate shows to be around 10 % if we consider the correct detected free DNA objects in the corresponding range. The length determination method D has a smaller error rate. Anyhow, this error cannot be set as fixed value of the algorithm. The image data set is a very heterogeneous and we have not examined how the used just free DNA containing images represent the full data set. However, this error has to be considered by looking at the following results, especially because the examined image data set has different nm/ pixel ratios and a lot of noise and dirt. The mean length of free DNA in a range of 130 to 223 nm averaged over both length determination methods is 196.5 nm (Method C 197.5 nm ± 27.9; Method D 195.5 nm ± 23.1 nm)

Table 6: Six images containing just free DNA fragments were analysed by algorithm C and D, respectively, in order to determine the number and length of detected DNA fragments. All detected bound DNA fragments in the 130 nm -223 nm range can be counted as false positive nucleosomes.

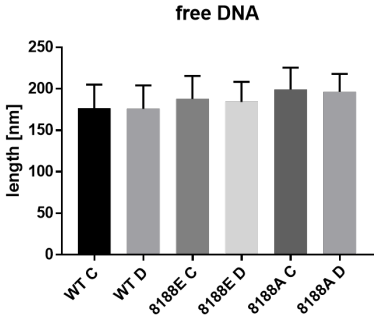
Valid Objects	**Algorithm C**	**Algorithm D**
Detected as bound DNA (130 nm - 223 nm long)	28	26
Detected as free DNA (130 nm - 223 nm long)	211	287
Total Objects	620	658

4.2 Biological Relevance

Our initial biological question was to compare the DNA binding characteristics of the wild type and mutated histones. In order to examine whether there is a difference in the number of detected nucleosomes, the bound DNA length, the positon of the nucleosome core and the distribution of the angles of the entering DNA arms, we evaluated the data set of in total 76 images with the Hilditch thinning algorithm and with the two length determination methods (algorithm C fast and algorithm D). The images were containing wild type, H2A R81A R88A and H2A R81E R88E nucleosomes. All nucleosomes were reconstituted on a 660 bp long DNA fragment containing the Widom 601 positioning sequence. This evaluation was done to proof that the results of the algorithm applied on that heterogeneous set of images lead to reasonable values for the investigated DNA and nucleosomes, without checking each objects for correctness. Since we did not control the images manually, only objects in the length regions mentioned before were considered. We focused on the intersect of the two ranges which lead to a range from 130 nm - 223 nm. The results of the length determination were converted into nanometres. Afterwards, the free DNA objects and the nucleosome objects of each histone type for each length determination method were sorted in eight groups. Inside this group the mean values and the standard deviations of the parameter of interest were calculated and the distribution of the lengths and angles were plotted.

4.3 Free DNA

The mean values of the length of free DNA varies between the different nucleosome types more than between the two length determination algorithms C and D. But mainly the number of analysed DNA objects on the H2A R81A R88A images indicates that the two methods can detect a clearly distinct number of objects in that wide nm range (see figure 59). This wide range leads also to a standard deviation from 24 nm to 29 nm and to a shift of the mean value to shorter distances. The lengths distribution of the free DNA in the investigated range shows very clear that both methods have a defined peak in the expected region from 190 nm to 223 nm. For all three types of nucleosome images the algorithm detected around 100 free DNA objects in the range between 223 nm and 240 nm which were not considered in our evaluation. Besides the differences of the image qualities, the exact experimental conditions during the AFM experiment of the different types of nucleosomes might have differed which has a high, not really estimable impact on the here shown results. However, the shift of the mean value of free DNA on wild type nucleosome images is also influenced by an accumulation of lengths around 150 nm, which is not that present on the images of the two mutant nucleosomes (see figure 60).



	WT C	WT D	8188E C	8188E D	8188A C	8188A D
Mean	176.35	175.94	187.84	184.16	199.18	196.36
Standard deviation	28.76	28.12	27.77	24.38	26.33	21.64
# of analysed objects	697	724	762	784	659	878

Figure 59: The mean values in nm of the length of the detected free DNA on wild type nucleosome (WT), H2A R81A R88A nucleosome (8188A) and H2A R81E R88E nucleosome (8188E) images for the both length determination methods C and D are plotted. The error bars represent the standard deviation. The exact values, as well as the number of analysed objects are listed below. For each method of the algorithm all detected free DNA objects in a nm range between 130 nm and 223 nm were used for this evaluation without any further controlling step. 70 images were included in this analysis.

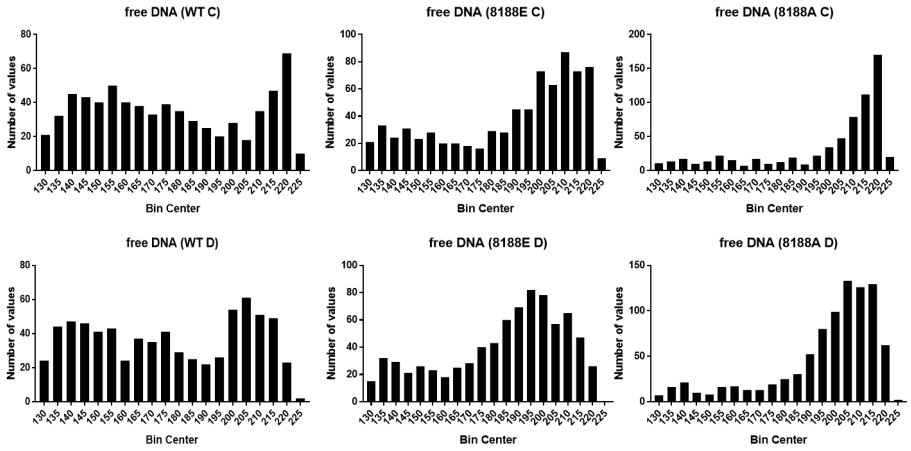
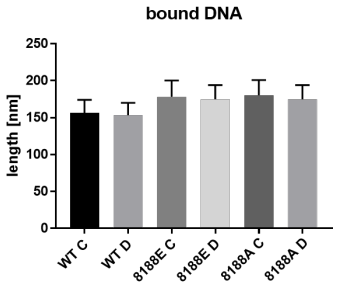


Figure 60: The values of lengths of the detected free DNA objects on wild type nucleosome (WT), H2A R81A R88A nucleosome (8188A) and H2A R81E R88E nucleosome (8188E) images for the both length determination methods C and D are plotted in 5 nm steps All detected nucleosome objects of 76 images in a nm range between 130 nm and 223 nm were used for this evaluation without any further controlling step for algorithm C and D, respectively.

4.4 Bound DNA

The results of the length measurements of histone bound DNA are shown in figures 61 and 62. The number of detected wild type nucleosomes is for both algorithm C and D dramatically higher than the number of the two mutant nucleosomes. Furthermore, the bound DNA length of wild type nucleosomes is significantly shorter than the bound DNA length of the two mutant nucleosomes. The bound DNA lengths of wild type nucleosomes are normally distributed around the mean value with only a low amount of longer lengths (over 180 nm). The left end of the distribution, which are bound DNAs shorter than 130 nm, is not considered here (around 257 DNA fragments between 100 nm and 130 nm). The distributions of mutant nucleosomes result in higher mean value around 175 nm. The standard deviations in that range for bound DNA is around 20 nm and smaller compared to the free DNA length. The two algorithms show the same tendencies as the results of the free DNA measurement. Algorithm C measures longer lengths then algorithm D. The distribution, especially from wild type nucleosomes indicate clearly an accumulation of values in a distinct region. This is also the case for mutant nucleosomes, but the total number of detected objects is far lower.

The automatically measured lengths of the short DNA arm of the bound DNA also shows differences between wild type and the mutants (see figure 63). The mean value of the length of the short DNA arm on wild type images is approximately 60 nm, whereas the short DNA arm of the detected objects with



	WT C	WT D	8188E C	8188E D	8188A C	8188A D
Mean	156.08	153.30	178.30	174.92	180.22	174.97
Standard deviation	17.99	16.81	22.00	19.07	20.74	19.16
# of analysed objects	642	567	94	77	114	103

Figure 61: The mean values in nm of the length of the detected bound DNA on wild type nucleosome (WT), H2A R81A R88A nucleosome (8188A) and H2A R81E R88E nucleosome (8188E) images for the both length determination methods C and D are plotted. The error bars represent the standard deviation. The exact values, as well as the number of analysed objects are listed below. For each method of the algorithm all detected nucleosome objects in a nm range between 130 nm and 223 nm were used for this evaluation without any further controlling step. 70 images were included in this analysis.

mutated nucleosomes is around 40 nm. The two methods lead to almost the same mean values for wild type and H2A R81A R88A nucleosomes. For the H2A R81E R88E mutated nucleosome images, algorithm C results in a mean value that is 5 nm shorter than the mean value of algorithm D. The standard deviations indicate that the values of the mutant nucleosomes fluctuated more than the values of wild type nucleosomes.

4.5 Radius

The expected values for the nucleosome core radii from our manual evaluation are displayed in the automatically generated results (see figure 64). The radius of nucleosomes core is around 11 nm for wild type and H2A R81A R88A nucleosomes and for H2A R81E R88E almost 10 nm. Both algorithm C and algorithm D measure similar radii for each nucleosome type with low fluctuations.

4.6 Angle

The distribution of angles between the two DNA arms, entering the nucleosome core, measured in with the two methods is displayed in figure 65. Only the histogram of the automatic detection via algorithm C is shown. The histograms

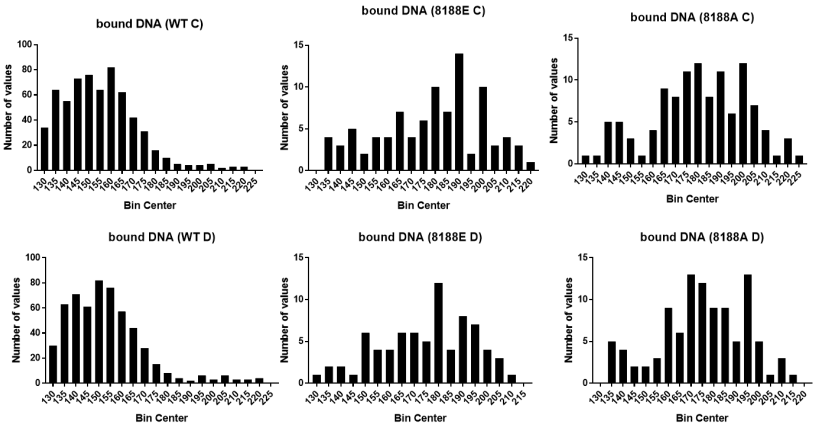


Figure 62: The values of the lengths of the detected bound DNA on wild type nucleosome (WT), H2A R81A R88A nucleosome (8188A) and H2A R81E R88E nucleosome (8188E) images for the both length determination methods C and D are plotted in 5 nm steps. All detected nucleosome objects of 76 images in a nm range between 130 nm and 223 nm were used for this evaluation without any further controlling step for algorithm C and D, respectively.

for method 1 confirm the during manual evaluation observed accumulation of angle over 100 ° and only a low amount of acute angles. The histograms for angle method 2 shows a more dominant appearance of acute angles. For wild type nucleosomes there is a peak around 80 °. The histogram of the two mutants are based on a lower amount of angles due to fewer detected nucleosomes. The distribution for algorithm C and D are comparable for both methods of angle determination.

5 Discussion

5.1 Discussion of the Algorithm

During test data analysis, the contour length was measured through the centre of the nucleosome core. Since the algorithm gives us the values for the DNA arms without the nucleosome core, the length results of the manual measurements should be in longer than the ones measured automatically. The diameter of the nucleosome core particle is in that context an important correction factor for the manually measured DNA counter length as well as for the length of the short DNA arm. Therefore, the test data results were also corrected by the factor of 11 nm which is the known distance of the nucleosome core particle [70], [71] before comparing it with the contour length of the algorithm.

Due to the heterogeneity of the investigated AFM images and the different nanometer to pixel ratios, we focused our evaluation on valid objects of the test

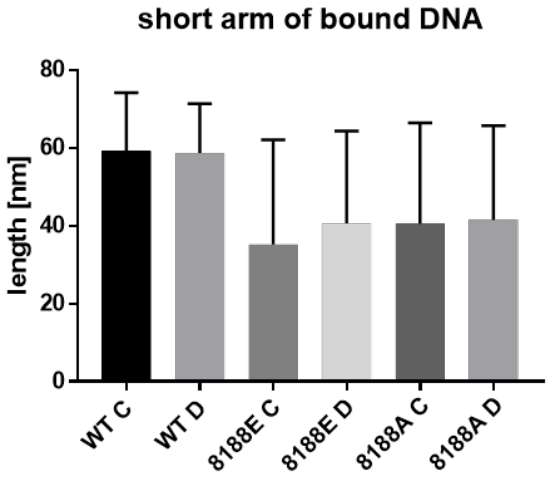


Figure 63: The mean values in nm of the length of the short DNA arm of the detected bound DNA on wild type nucleosome (WT), H2A R81A R88A nucleosome (8188A) and H2A R81E R88E nucleosome (8188E) images for the both length determination methods C and D are plotted. The error bars represent the standard deviation. The exact values, as well as the number of analysed objects are listed below. For each method of the algorithm all detected nucleosome objects in a nm range between 130 nm and 223 nm were used for this evaluation without any further controlling step. 70 images were included in this analysis.

data images. The free DNA on the investigated images should have a length of 660 bp. The length of nucleosome objects should be up to 150 bp shorter than the free DNA. Since the known distance for one base pair of canonical DNA is 0.34 nm, the length of the free DNA should be in a range of 223 nm. Since the images of our project are vitiated by an experimental error of the AFM measurements, the measured distances can differ from theoretical distances.

We showed mean values of the length parameters from single images to answer the biological question with the results from the algorithm and manual analysis. Therefore, we compared the mean values of the contour length of free DNA molecules, the contour lengths of nucleosomes and the positioning of nucleosomes with the short arm of the free DNA in nucleosome objects. For the mean values we excluded objects differing too much from the expected DNA. Our work shows that the results of our algorithm clearly depends on the quality of the images. The images are in Tagged Image File Format (TIFF), which leads to the loss of most of the height information. Therefore, it was very hard to handle dirt, which is in the range of our fixed DNA parameters. A confirmation by an operator controlling the found objects is a possibility to reduce FP and FN. The algorithm produces a new image with the found objects, which allows a relatively easy controlling. The most critical comparison of our training data and the automatically measured data are the angle values.

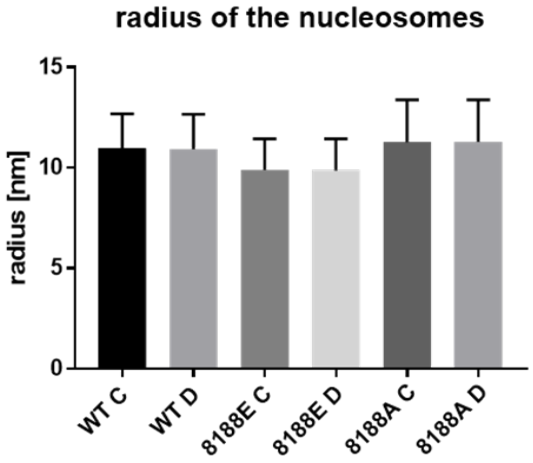


Figure 64: The mean values in nm of the length of the nucleosome cores on wild type nucleosome (WT), H2A R81A R88A nucleosome (8188A) and H2A R81E R88E nucleosome (8188E) images for the both length determination methods C and D are plotted. The error bars represent the standard deviation. The exact values, as well as the number of analysed objects are listed below. For each method of the algorithm all detected nucleosome objects in a nm range between 130 nm and 223 nm were used for this evaluation without any further controlling step. 70 images were included in this analysis.

The angle measuring tool in ImageJ is not a very precise tool, small changes of the initial points result in huge changes of the final angle. Therefore, we tested two different methods of measuring the angles. Anyhow, the manually measured values, mainly the ones of method 1, differ too much among each other. For this purpose, synthetic data would have been a useful tool to evaluate the algorithm. The generation of synthetic DNA skeletons has previously been described by Rivetti and Codeluppi. They simulated two-dimensional polymers with defined length as a connected chaincode and then converted the binary image into a AFM-like image by grating it hypothetically with an AFM tip [44]. Extending this algorithm with a feature that occasionally positions a circular object (nucleosome) at defined sharp corners of the polymer and defined distances to the endpoints could generate synthetic data that is suitable to evaluate the algorithm.

5.2 Discussion of the biological Data

The results of the evaluation of the complete image data set with our algorithm will be discussed in the following. The exact biological evaluation with wrapping length and calculation of exact base pair values are strongly influenced from the experimental conditions. Therefore, we focused on the basic order of the results.

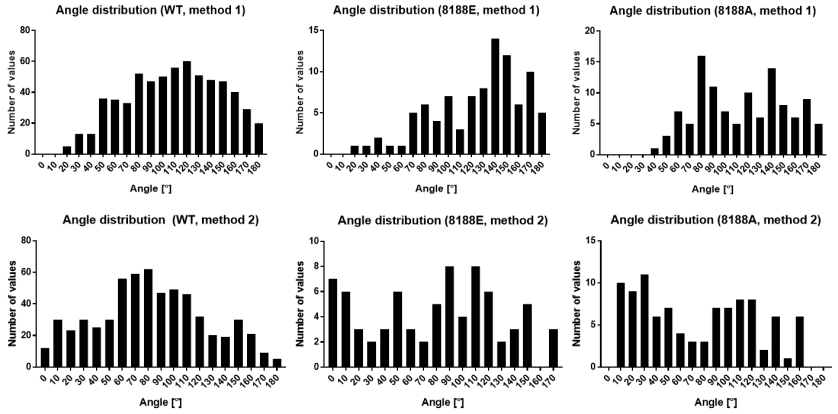


Figure 65: The distribution of angles of the DNA arms at the nucleosome core from the detected bound DNA objects on wild type nucleosome (WT), H2A R81A R88A nucleosome (8188A) and H2A R81E R88E nucleosome (8188E) images for the both length determination methods C and D is plotted in 5 ° steps. All detected nucleosome objects of 76 images in a nm range between 130 nm and 223 nm were used for this evaluation without any further controlling step for algorithm C and D, respectively.

The experimental conditions, for instance the exact condition of the coating of the surface, should be constant for all image acquisitions. Anyhow, the different qualities of the images and the different types of noise and artefacts indicate that they might not have been exactly the same. Additionally, the different scan-size settings during the recording of the different images could have influenced the apparent length parameters on the images. However, we considered all relevant images, converted the lengths into nm and analysed the results of the algorithm for the three types of nucleosomes. Our data set consists of images with very heterogeneous quality, a lot of background noise and artefacts. The aim was to generate an algorithm with the capacity to, despite the circumstances, was robust enough to analyse those images. For the biological data analysis, we did not control every measured object but worked with all detected objects within the expected range. First of all, we needed to decide, which ranges were appropriate for free and bound DNA. The objects under or above this range were not considered. These drastic limitations were necessary due to the complicated data set with a high amount of dirt and artefacts, and short DNA-fragments that would have a significant evaluation impossible. Still, the chosen range was wide enough to cover possible fluctuations around the expected length value. For a more detailed analysis one should also consider the nm range from 223 nm – 230 nm for free DNA, and the range from around 110 nm – 130 nm for nucleosomes. The here shown approximate evaluation with mean values and qualitative description of the distributions is sufficient for the investigated range and order of results. The rate of falsely detected nucleosomes of roughly 10 % on images containing only free DNA cannot represent an accurate error

of the algorithm, again due to the heterogeneity of the images. It is more a combination of the real false positive rate of the algorithm and the errors from the quality of the images used. Indeed, not every detected nucleosome in the investigated nm range is a valid object, but the error rate strongly correlates with the quality of the images. So this 10 % error is somehow relevant for this data set but not for the algorithm itself. Nevertheless, the error of false detection is an important factor of such an algorithm and has to be further improved. Although the data set is complicated and one have to calculate with an error of false detected objects, the algorithm leads to reasonable results. The order of the results of the different parameters of interest are in the expected range which clearly indicates that the algorithm can analyse such a complicated data set.

5.3 Free DNA

The distributions of the length of free DNA demonstrate that the algorithm is able to correctly measure the lengths by using both method C and method D. The expected length of free DNA is between 190 nm and 223 nm. Here, the length of free DNA might have been influenced trough interactions histone proteins. Meaning that not octamer/hexamer histone protein could somehow interact with DNA, which could influence the length. Inside the 130 nm to 223 nm range, we checked also the distributions of the different length values. This shows that mainly the resulted mean value of the free DNA is shifted, because of that wide range. Mainly the free DNA length of wild-type nucleosomes is influenced from an accumulation of lengths in the range of 150 nm. This length lies in the range of bound DNA and probably derive from bound DNA objects which were not detected correctly. Anyway, with a smaller range around 190 nm – 223 nm this influence would be reduced. The calculated mean value of the length of free DNA on images only containing DNA is around 200 nm. This mean value could be more representative for the actual free DNA length.

5.4 Bound DNA

The length of bound DNA was measured automatically by adding the length of the two DNA arms beside the nucleosome core, as previously described [72]. Due to different radii of different nucleosomes, measuring through the nucleosome core would lead to more error-prone results. For our purpose, the results of the length measurement were satisfactory. However, for different applications with images of higher resolution, measuring the length through the nucleosome core might be an interesting or necessary feature. Our algorithm is already able to produce those results as well. As for free DNA, the measured parameter in the 130 nm - 223 nm range for bound DNA matches with the expected values. The bound DNA length of wild-type nucleosomes results in a distribution around the mean of a reasonable shortening through wrapping DNA around the octamer. As mentioned before one can also consider the left side of the distribution below 130 nm. And because of the heterogeneous data set we calculate not exact values

for wrapping length. The distribution for wild-type nucleosomes and the peak around 150 nm is similar to the results found in literature under comparable experiment conditions, such as the AFM instrument and the theoretical free DNA length [69]. The same is for the distribution of the angles of the DNA arms at the nucleosome core determined with method 2. The accumulation of angles from 70°- 100° was also observed for wild-type nucleosomes [69]. The positioning of histones on the 601 positioning sequence can be determined by measuring the length of the short DNA arm of bound fragments. Here, for wild type nucleosomes it is in the expected range. Different studies demonstrated that the strong positioning of nucleosomes on that positioning sequence is also detectable via AFM [73], [74]. The positioning sequence is 199 bp – 369 bp apart from one end of the DNA, which would correspond to a short arm of the bound DNA starting at 68 nm. The measured value of around 60 nm matches very good with the theoretical value. One has to consider that not all nucleosomes are positioned at that sequence. Together with the shortening of the measured lengths during the AFM measurements the results of the short arm of wild type nucleosomes together with the length of bound DNA clearly indicates that the algorithm actually measures real nucleosome objects in a precise manner.

5.5 Differences between wild Type and mutated Nucleosomes

The most valid result of the comparison between wild type nucleosomes images and the two mutant nucleosome images, during the AFM experiments, was the different rate free DNA to the bound DNA. By checking the images by eye, one can clearly see that inside the investigated nm range, the number of nucleosomes is significant lower on the images with mutant nucleosomes than the number of free DNA objects. On wild type images the rate of bound DNA to free DNA is clearly higher. The tendency of this observation was also confirmed by our algorithm. For images of the mutants, the detected number of bound DNA is clearly lower than the detected number of free DNA. On the contrary, on wild type images the numbers of bound DNA is comparable to the number of free DNA. Although that rates are strongly influenced by the false detected object, as discussed before, the order of results of the algorithm completely agrees with the expected results. Also the other parameters of the mutant nucleosomes indicate that there could be differences to wild type nucleosomes. Since these mutants were not investigated in AFM experiments before, it is hard to classify the results of the algorithm. Previous experiments with these mutants let suggest that there could be a difference in the length of bound DNA (unpublished data). AFM-Experiments of Montel.et indicate that nucleosomes with the H2A.Bbd variant wrapped less base pairs around the histone octamer compared to wild-type nucleosome [72] which underlies the hypothesis. Our results from the algorithm for the images with mutant histones indicate that there is a difference compared to wild-type nucleosomes. The mean value of the length of bound DNA of the two mutants is clearly higher than the mean value of wild-type nucleosomes. Also distribution of the bound DNA length indicate that

longer lengths are present. This agrees with the hypothesis that the wrapping length of the mutants is smaller than by wild-type nucleosomes. The radii of the nucleosome core were also in the expected range from the manual evaluation. They show that the nucleosome cores of the image data set are broaden compared to the theoretical value, due to the resolution of the measurements as described before [68]. The radii of H2A R81E R88E seems to be smaller than those of the wild-type and the H2A R81A R88A nucleosomes. This could be also interesting observation, which one should consider for further experiments. Also the results of the short DNA arm of bound DNA show differences to wild-type nucleosomes. On mutant nucleosome objects the fluctuation of the short DNA arm is more present compared to wild type nucleosome. Also the mean value of the length is clearly apart from the theoretical value of positioning on the Widom 601 sequence. These differences in the results of wild type and mutant nucleosomes of the algorithm indicate a significant difference. But since the results were from a difficult image data set, which also lead to false detected objects, one has to consider them with caution. Because it is the first time investigating these mutants on AFM one has to further control the data and experimental conditions, for reliable results. That biological question need a more detailed look on the data. Furthermore, there have to be more experiments with more comparable conditions and a better quality to proof this results, which is not part of this project. But the results of the mutant nucleosome images demonstrate that one can get also for the mutant nucleosomes information out of the investigated data set. Further that it is possible that one can find out precise and valid differences on the basis of a better image data set. Anyway the fact that the results for mutant nucleosome images are different to the results wild type data, demonstrate the power of the algorithm to analyse AFM images of wild type nucleosomes. The aim of our project, which was to implement an algorithm which can analyse a difficult image data set, that one can compare wild type and mutant nucleosomes, is fully accomplished in that context.

5.6 Length Determination Method C and D

Since the algorithm can determine the length of DNA and nucleosome objects in two different methods (C and D) we have two results for every object. Until this point we discussed the general biological results, because the results of the two length determination methods are that close together. Due to our evaluation, where we consider every object for every method which is in the nm range the mean values do not consist of completely identical objects. This different numbers of objects lead to the fact that we focus on the overall results for each of the two methods independent form the other method. The differences between the two methods of length determination is also influenced by evaluated nm range. The detected objects from method D result in shorter lengths compared to method C. This is indicated by the mean values as well as the shift in the distributions. In all three types of nucleosome images, method D has more free DNA objects in the investigated nm range. For free DNA objects of H2A R81A

R88A images the mean value of method C is lower. The distribution shows that method D detected more objects in the 190- 223 nm region. But in the results of method C the highest number of objects is in the 220 nm range, whereas the peak of the distribution in the results of method D is at lower lengths. This results lead to the assumption that method D measures shorter length of free DNA than method C. In that context, the clear different number of object on the H2A R81A R88A images have to be investigated further. But the distribution of both methods are still very similar. Also the evaluation of bound DNA lead to comparable results with slightly shorter lengths measured by method D. Here, method C detects more bound DNA objects on all types of nucleosome images. This switch in the detection compared to free DNA indicates a different precision between the two methods, which has to be further investigated and improved. Also the difference in the length of the short DNA arm of H2A R81E R88E nucleosomes indicate that the different detection and probably the number of the detected objects may be the reason for bigger fluctuation between the two methods. The angle distribution was only shown for method C. The angle distributions of angle determination method 1 and 2 showed to be in the expected ranges. However, method 1 has not the biological impact as method 2, because this angle does not represent the actual entering DNA arms. One could think of further evaluations, combining the approximate region of the angle with the wrapping length for each nucleosome, to better analyse the exact state of the nucleosomes [74].. All in all, two different methods of length determination is a big power of our algorithm. Since both the results from method C and D are in a reliable region and close together, it is hard to tell if one method is better than the other. Moreover, a combination of both methods could bring the best results. One can also think of only consider the intersect of both methods which was not done in this evaluation. If there is a significant difference declaring objects as valid, which the numbers of detected objects indicate, the consideration of objects which both method detect could be the best solution which also could reduce the error. Therefore, it is necessary to further evaluate the difference between the two algorithm. The reason why the two algorithms detected different numbers of valid objects in the investigated nm range is still unclear and has to be further analysed. The switch in the higher number of detected objects between the two methods and free to bound DNA is also an interesting observation which could further lead to an optimization of the error rate.

6 Conclusion and Outlook

The evaluation of the whole image data set proofed that our algorithm can analyse full automatically the images. Without any controlling of the detected objects, the user can get reasonable values for the investigated DNA or nucleosome objects. Here it is important to mention that on images with better quality also the errors rates of the algorithm are decreased. Furthermore, the algorithm gives the user, with the overlay image with the numbered detected

objects as output additionally to the table with the results, the possibility to control the results. Since our algorithm was implemented for images converted into TIFF format, exact height and dilatation criteria, which are also possible parameter for the evaluation of AFM experiments, were not considered to analyse nucleosome objects with our algorithm [69], [72], [73]. Therefore, users of AFM imaging of DNA/nucleosomes can get very fast and easy information about their objects on the image, independent from the setup they use. Furthermore, it is thinkable that the algorithm can analyse other objects like intermediate filaments or DNA/nucleosomes on EM images.

References

[1] Shantanu Sharma and Nikolay V Dokholyan. Dna sequence mediates nucleosome structure and stability. *Biophysical journal*, 94(1):1–3, 2008.

[2] Masa H Sato, Kiyoe Ura, Ken I Hohmura, Fuyuki Tokumasu, Shige H Yoshimura, Fumio Hanaoka, and Kunio Takeyasu. Atomic force microscopy sees nucleosome positioning and histone h1-induced compaction in reconstituted chromatin. *FEBS letters*, 452(3):267–271, 1999.

[3] PT Lowary and Jonathan Widom. New dna sequence rules for high affinity binding to histone octamer and sequence-directed nucleosome positioning. *Journal of molecular biology*, 276(1):19–42, 1998.

[4] E Fischbach, DE Krause, VM Mostepanenko, and M Novello. New constraints on ultrashort-ranged yukawa interactions from atomic force microscopy. *Physical Review D*, 64(7):075010, 2001.

[5] EA Evans and DA Calderwood. Forces and bond dynamics in cell adhesion. *Science*, 316(5828):1148–1153, 2007.

[6] PM Hoffmann, A Oral, RA Grimble, S Jeffery, JB Pethica, et al. Direct measurement of interatomic force gradients using an ultra-low-amplitude atomic force microscope. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, volume 457, pages 1161–1174. The Royal Society, 2001.

[7] OH Willemsen, MME Snel, A Cambi, J Greve, BG De Grooth, and CG Figdor. Biomolecular interactions measured by atomic force microscopy. *Biophysical Journal*, 79(6):3267–3281, 2000.

[8] Guthold M. Afm image of lambda dna. `http://users.wfu.edu/gutholdm/AFM%20imaging.html`. Accessed: 2016-07-05.

[9] Bruker corporation: Bio dynamics with atomic force microscopes. `https://www.bruker.com/products/surface-and-dimensional-analysis/atomic-force-microscopes/campaigns/bioafms-and-biological-applications/bio-dynamics-with-afm.html`. Accessed: 2016-07-05.

[10] J Battacharjee. Atomic force microscope: Fundamental principles. `http://de.slideshare.net/joybiitk/atomic-force-microscope-fundamental-principles`. Accessed: 2016-07-05.

[11] The florida state university, dennis lab. `http://chromatin.bio.fsu.edu/`. Accessed: 2016-07-05.

[12] G Binnig, CF Quate, and C Gerber. Atomic force microscope. *Physical Review Letters*, 56(9):930, 1986.

[13] P Hinterdorfer and YF Dufrêne. Detection and localization of single molecular recognition events using atomic force microscopy. *Nature Methods*, 3(5):347–355, 2006.

[14] B Cappella and G Dietler. Force-distance curves by atomic force microscopy. *Surface Science Reports*, 34(1):1–104, 1999.

[15] M Radmacher. Measuring the elastic properties of biological samples with the afm. *IEEE Engineering in Medicine and Biology Magazine*, 16(2):47–57, 1997.

[16] KM Lang, DA Hite, RW Simmonds, R McDermott, DP Pappas, and John M Martinis. Conducting atomic force microscopy for nanoscale tunnel barrier characterization. *Review of Scientific Instruments*, 75(8):2726–2731, 2004.

[17] A Bird. Perceptions of epigenetics. *Nature*, 447(7143):396–398, 2007.

[18] C Cuozzo, A Porcellini, T Angrisano, A Morano, B Lee, A Di Pardo, S Messina, R Iuliano, A Fusco, MR Santillo, et al. Dna damage, homology-directed repair, and dna methylation. *PLoS Genet*, 3(7):e110, 2007.

[19] T Jenuwein, G Laible, R Dorn, and G Reuter. Set domain proteins modulate chromatin domains in eu-and heterochromatin. *Cellular and Molecular Life Sciences CMLS*, 54(1):80–93, 1998.

[20] AJ Gottschalk, G Timinszky, SE Kong, J Jin, Y Cai, SK Swanson, MP Washburn, L Florens, AG Ladurner, JW Conaway, et al. Poly (adp-ribosyl) ation directs recruitment and activation of an atp-dependent chromatin remodeler. *Proceedings of the National Academy of Sciences*, 106(33):13770–13774, 2009.

[21] JC Lin, S Jeong, G Liang, D Takai, M Fatemi, YC Tsai, G Egger, EN Gal-Yam, and PA Jones. Role of nucleosomal occupancy in the epigenetic silencing of the mlh1 cpg island. *Cancer Cell*, 12(5):432–444, 2007.

[22] L Mariño-Ramírez, MG Kann, BA Shoemaker, and D Landsman. Histone structure and nucleosome stability. *Expert Review of Proteomics*, 2(5):719–729, 2005.

[23] EL Mersfelder and MR Parthun. The tale beyond the tail: histone core domain modifications and the regulation of chromatin structure. *Nucleic Acids Research*, 34(9):2653–2662, 2006.

[24] Y Lorch, JW LaPointe, and RD Kornberg. Nucleosomes inhibit the initiation of transcription but allow chain elongation with the displacement of histones. *Cell*, 49(2):203–210, 1987.

[25] JC Hansen, JK Nyborg, K Luger, and LA Stargell. Histone chaperones, histone acetylation, and the fluidity of the chromogenome. *Journal of Cellular Physiology*, 224(2):289–299, 2010.

[26] A Ozdemir, S Spicuglia, E Lasonder, M Vermeulen, C Campsteijn, HG Stunnenberg, and C Logie. Characterization of lysine 56 of histone h3 as an acetylation site in saccharomyces cerevisiae. *Journal of Biological Chemistry*, 2005.

[27] G Schotta, M Lachner, K Sarma, A Ebert, R Sengupta, G Reuter, D Reinberg, and T Jenuwein. A silencing pathway to induce h3-k9 and h4-k20 trimethylation at constitutive heterochromatin. *Genes & Development*, 18(11):1251–1262, 2004.

[28] N Kourmouli, P Jeppesen, S Mahadevhaiah, P Burgoyne, R Wu, DM Gilbert, S Bongiorni, G Prantera, L Fanti, S Pimpinelli, et al. Heterochromatin and tri-methylated lysine 20 of histone h4 in animals. *Journal of Cell Science*, 117(12):2491–2501, 2004.

[29] NJ Krogan, J Dover, A Wood, J Schneider, J Heidt, MA Boateng, K Dean, OW Ryan, A Golshani, M Johnston, et al. The paf1 complex is required for histone h3 methylation by compass and dot1p: linking transcriptional elongation to histone methylation. *Molecular Cell*, 11(3):721–729, 2003.

[30] BD Strahl, PA Grant, SD Briggs, Z Sun, JR Bone, JA Caldwell, S Mollah, RG Cook, J Shabanowitz, DF Hunt, et al. Set2 is a nucleosomal histone h3-selective methyltransferase that mediates transcriptional repression. *Molecular and Cellular Biology*, 22(5):1298–1306, 2002.

[31] R Cao, L Wang, H Wang, L Xia, H Erdjument-Bromage, P Tempst, RS Jones, and Y Zhang. Role of histone h3 lysine 27 methylation in polycomb-group silencing. *Science*, 298(5595):1039–1043, 2002.

[32] C Sawan and Z Herceg. Histone modifications and cancer. *Adv Genet*, 70:57–85, 2010.

[33] A Japaridze, A Benke, S Renevey, C Benadiba, and G Dietler. Influence of dna binding dyes on bare dna structure studied with atomic force microscopy. *Macromolecules*, 48(6):1860–1865, 2015.

[34] F Chen, PW Tillberg, and ES Boyden. Expansion microscopy. *Science*, 347(6221):543–548, 2015.

[35] A Adomas, G Heller, Å Olson, J Osborne, M Karlsson, J Nahalkova, L Van Zyl, R Sederoff, J Stenlid, R Finlay, et al. Comparative analysis of transcript abundance in pinus sylvestris after challenge with a saprotrophic, pathogenic or mutualistic fungus. *Tree Physiology*, 28(6):885–897, 2008.

[36] ME Dokukin, NV Guz, and I Sokolov. Towards nano-physiology of insects with atomic force microscopy. *Journal of Insect Physiology*, 57(2):260–264, 2011.

[37] PA Wiggins, T Van Der Heijden, F Moreno-Herrero, A Spakowitz, R Phillips, J Widom, C Dekker, and PC Nelson. High flexibility of dna on short length scales probed by atomic force microscopy. *Nature Nanotechnology*, 1(2):137–141, 2006.

[38] J Marek, E Demjénová, Z Tomori, J Janáček, I Zolotová, F Valle, M Favre, and G Dietler. Interactive measurement and characterization of dna molecules by analysis of afm images. *Cytometry Part A*, 63(2):87–93, 2005.

[39] V Cassina, Manoel Manghi, D Salerno, A Tempestini, V Iadarola, L Nardo, S Brioschi, and F Mantegazza. Effects of cytosine methylation on dna morphology: An atomic force microscopy study. *Biochimica et Biophysica Acta (BBA)-General Subjects*, 1860(1):1–7, 2016.

[40] MTS Spisz, Y Fang, RH Reeves, CK Seymour, IN Bankman, and JH Hoh. Automated sizing of dna fragments in atomic force microscope images. *Medical and Biological Engineering and Computing*, 36(6):667–672, 1998.

[41] A Sanchez-Sevilla, J Thimonier, M Marilley, J Rocca-Serra, and J Barbet. Accuracy of afm measurements of the contour length of dna fragments adsorbed on mica in air and in aqueous buffer. *Ultramicroscopy*, 92(3):151–158, 2002.

[42] A Sundstrom, S Cirrone, S Paxia, C Hsueh, R Kjolby, JK Gimzewski, J Reed, and B Mishra. Image analysis and length estimation of biomolecules using afm. *IEEE Transactions on Information Technology in Biomedicine*, 16(6):1200–1207, 2012.

[43] LS Marturelli, CA Achete, and GAG Cidade. Automated length measurement based on the snake model applied to nanoscience and nanotechnology. In *Proceedings of the XXXI. Congresso Nacional de Matemática Aplicada e Computacional*, pages 170–175. Sociedade Brasileira de Matemáatica Aplicada e Computacional, 2008.

[44] C Rivetti and S Codeluppi. Accurate length determination of dna molecules visualized by atomic force microscopy: evidence for a partial b-to a-form transition on mica. *Ultramicroscopy*, 87(1):55–66, 2001.

[45] E Ficarra, L Benini, E Macii, and G Zuccheri. Automated dna fragments recognition and sizing through afm image processing. *Information Technology in Biomedicine, IEEE Transactions on*, 9(4):508–517, 2005.

[46] E Ficarra, D Masotti, E Macii, L Benini, G Zuccheri, and B Samorì. Automatic intrinsic dna curvature computation from afm images. *IEEE Transactions on Biomedical Engineering*, 52(12):2074–2086, 2005.

[47] DJ Rigotti, B Kokona, T Horne, EK Acton, CD Lederman, KA Johnson, RS Manning, SA Kane, WF Smith, and R Fairman. Quantitative atomic force microscopy image analysis of unusual filaments formed by the

acanthamoeba castellanii myosin ii rod domain. *Analytical Biochemistry*, 346(2):189–200, 2005.

[48] E Ficarra, D Masotti, L Benini, M Milano, and A Bergia. Automated dna curvature profile reconstruction in atomic force microscope images. *AI* IA Notizie*, 4:64–68, 2002.

[49] JS Weszka. A survey of threshold selection techniques. *Computer Graphics and Image Processing*, 7(2):259–265, 1978.

[50] TW Ridler and S Calvard. Picture thresholding using an iterative selection method. *IEEE Trans Syst Man Cybern*, 8(8):630–632, 1978.

[51] CR Gonzales. Wintz p.: Digital image processing. *Addison-Wesley Publishing Company*, 1:987, 1987.

[52] JC Russ. The image processing handbook, 1992. *Boca Raton, FL: CRC*, 1992.

[53] TY Zhang and Ching Y Suen. A fast parallel algorithm for thinning digital patterns. *Communications of the ACM*, 27(3):236–239, 1984.

[54] G Brugal and JM Chassery. [a new image-processing system designed for densitometry and pattern analysis of microscopic specimen. application to the automated recognition and counting of cells in the various phases of the mitotic cycle (author's transl)]. *Histochemistry*, 52(3):241–258, 1977.

[55] WL Pan and RP Bolton. Root quantification by edge discrimination using a desktop scanner. *Agronomy Journal*, 83(6):1047–1052, 1991.

[56] L Dorst and AWM Smeulders. Length estimators for digitized contours. *Computer Vision, Graphics, and Image Processing*, 40(3):311–333, 1987.

[57] AM Vossepoel and AWM Smeulders. Vector code probability and metrication error in the representation of straight lines of finite length. *Computer Graphics and Image Processing*, 20(4):347–364, 1982.

[58] L Yang, F Albregtsen, T Lønnestad, and P Grøttum. Methods to estimate areas and perimeteres of blob-like objects: a comparison. 1994.

[59] Jochen Ludewig and Horst Lichter. *Software Engineering*. dpunkt-Verl., Heidelberg, 3., korr. aufl. edition, 2013.

[60] Itseez. *The OpenCV Reference Manual*, 2.4.13.0 edition, May 2016.

[61] Itseez. Open Source Computer Vision Library. `https://github.com/itseez/opencv`, 2016.

[62] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. Non-Local Means Denoising. *Image Processing On Line*, 1, 2011.

[63] OpenCV Interface Support. `http://de.mathworks.com/help/vision/opencv-interface-support-package.html`, 2016.

[64] GNU Octave. `https://www.gnu.org/software/octave/`, 2016.

[65] L Lam, SW Lee, and CY Suen. Thinning methodologies-a comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(9):869–885, 1992.

[66] J Linbo. Implementation of zhang-suen thinning algorithm for skeletonization. `https://github.com/linbojin/Skeletonization-by-Zhang-Suen-Thinning-Algorithm`, 2014.

[67] AR Widiarti. Comparing hilditch, rosenfeld, zhang-suen, and nagendraprasad-wang-gupta thinning. *World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering*, 5(6):563–567, 2011.

[68] M Bussiek, K Toth, N Brun, and J Langowski. Dna-loop formation on nucleosomes shown by in situ scanning force microscopy of supercoiled dna. *Journal of Molecular Biology*, 345(4):695–706, 2005.

[69] JF Kepert, KF Tóth, M Caudron, N Mücke, J Langowski, and K Rippe. Conformation of reconstituted mononucleosomes and effect of linker histone h1 binding studied by scanning force microscopy. *Biophysical Journal*, 85(6):4012–4022, 2003.

[70] Malte Bussiek, Gabriele Müller, Waldemar Waldeck, Stephan Diekmann, and Jörg Langowski. Organisation of nucleosomal arrays reconstituted with repetitive african green monkey α-satellite dna as analysed by atomic force microscopy. *European Biophysics Journal*, 37(1):81–93, 2007.

[71] Curt A Davey, David F Sargent, Karolin Luger, Armin W Maeder, and Timothy J Richmond. Solvent mediated interactions in the structure of the nucleosome core particle at 1.9 å resolution. *Journal of molecular biology*, 319(5):1097–1113, 2002.

[72] F Montel, E Fontaine, P St-Jean, M Castelnovo, and C Faivre-Moskalenko. Atomic force microscopy imaging of swi/snf action: mapping the nucleosome remodeling and sliding. *Biophysical Journal*, 93(2):566–578, 2007.

[73] I Nazarov, I Chekliarova, G Rychkov, AV Ilatovskiy, C Crane-Robinson, and A Tomilin. Afm studies in diverse ionic environments of nucleosomes reconstituted on the 601 positioning sequence. *Biochimie*, 121:5–12, 2016.

[74] J Nishikawa and T Ohyama. Selective association between nucleosomes with identical dna sequences. *Nucleic Acids Research*, page gks1269, 2012.