FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

ASSIGNMENT OF MASTER’S THESIS

Title:	Protein particles detection and analysis in images from optical microscopy
Student:	Bc. Petr Wudi
Supervisor:	Ing. Jakub Novák
Study Programme:	Informatics
Study Branch:	Knowledge Engineering
Department:	Department of Applied Mathematics
Validity:	Until the end of summer semester 2019/20

Instructions

Get familiar with optical microscopy techniques called structured-illumination microscopy and the resulting images. Create an algorithm that will be able to detect single particles (or particle clusters) using methods of image processing and to analyze their distribution.

Goals:
1) Perform a search in the field of structured-illumination microscopy and usable image processing methods.
2) Specify methods of image processing and design preprocessing algorithms that will lead to the detection of single particles.
3) Choose a few methods for evaluating the distribution of particles in the image.
4) Implement algorithms using appropriate programming language.
5) Test the designed algorithms on real data.
6) Evaluate the results of a few algorithms and choose the best solution for the task.
7) Discuss the results.

References

Gustafsson M.G.L. Nonlinear structured-illumination microscopy: wide-field fluorescence imaging with theoretically unlimited resolution.
Gustafsson M.G.L. Surpassing the lateral resolution limit by a factor of two using structured illumination microscopy.
Gustafsson N. Fast live-cell conventional fluorophore nanoscopy with ImageJ through super-resolution radial fluctuations.
Harke B. Resolution scaling in STED microscopy.
Heilemann M. Subdiffraction-resolution fluorescence imaging with conventional fluorescent probes.
Hofmann M. Breaking the diffraction barrier in fluorescence microscopy at low light intensities by using reversibly photoswitchable proteins.
Khan A.O. CRISPR-Cas9 mediated labelling allows for single molecule imaging and resolution.
Klar T.a, Hell S.W. Subdiffraction resolution in far-field fluorescence microscopy.

Ing. Karel Klouda, Ph.D.	doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Head of Department	Dean

Prague February 17, 2019

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Master’s thesis

Protein particles detection and analysis in images from optical microscopy

Bc. Petr Wudi

Department of Applied Mathematics
Supervisor: Ing. Jakub Novák

May 28, 2020

Acknowledgements

I’d like to thank Ing. Jakub Novák for his helpful advices and Ayoub Stelate, M.Sc., for introduction to his research and providing the sample images. I would also like to thank both of them for their time and patience.

Declaration

Citation of this thesis

Wudi, Petr. *Protein particles detection and analysis in images from optical microscopy.* Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2020.

Abstrakt

Tato práce se zabývá automatizovanou detekcí částic proteinů na snímku z mikroskopu, jejichž rozložení je následně analyzováno.

Práce obsahuje analýzu exisujících řešení zabývajících se podobnými problémy a podrobnější popis vybraných metod zpracování obrazu.

Tyto metody byly implementovány v jazyce Java a použity pro návrh algoritmu schopného detekovat na snímku jednotlivé částice.

Různé kombinace metod byly testovány na reálných datech a porovnány s manuálně anotovanými daty.

Pozice částic nalezené nejlepšími algoritmy sloužily jako vstup pro vybrané metody prostorové analýzy.

Klíčová slova detekce, lokalizace molekul, protein, částice, rozložení, zpracování obrazu

Abstract

This thesis focuses on automated detection of protein particles on microscope images. Distribution of the detected particles is analyzed.

The thesis contains an analysis of existing solutions to similar problems and description of selected image processing methods.

These methods have been implemented in Java and used in the design of a particle detection algorithm.

Several method combinations have been tested on real data and compared to manually annotated data.

Particle positions detected by the best algorithms have been processed by selected spatial analysis techniques.

Keywords detection, single-molecule localization, protein, particle, distribution, image processing

Contents

List of Figures

Introduction

Research of plant proteins is a very important field of biology with a significant impact on several other fields like pharmacy, agriculture and many others.

Proteins often form “particles” – small clusters whose size often doesn’t exceed tens of µm.

This thesis was created to facilitate research of Ayoub Stelate, M.Sc., from the Department of Experimental Plant Biology of Charles University, who studies behaviour of proteins.

One of the pieces of information useful in the research is distribution of the particles. Knowing where does each particle lie would allow application of several analytical methods.

However, getting exact information about the particle locations is difficult. Counting the particles manually would be very exhausting as there are hundreds or even thousands of them in a sample.

The most suitable response to this task is to use a program that automatically detects the particles. This thesis focuses on the creation of such a program.

There are several challenges the program has to deal with.

The images have been created using a regular light microscope, which allows examining the samples *in vivo* – living samples. The light microscopes have several limitations coming from the nature of light, which lowers the image quality.

The images are blurred and contain a lot of noise. It makes the protein particles not easily separable from each other.

A program able to process these images is designed and implemented in Java. Result of the program is compared to manually annotated data.

The resulting program is implemented as a plugin to an image processing program ImageJ. This program is often used by biologists and therefore integration to it would make detecting the particle positions more convenient.

Another part of the thesis is a basic analysis of the distribution of the particles.

Chapter 1

Research

This chapter focuses on methods and approaches other authors use to solve similar tasks.

Andersson et al [1] find centers of fluorescent particles and then tracks them. Potential fluorescent points are placed in local maximums. Each potential point should have width about the same as the diffraction limit.

Then center of the points are located by fitting Gaussian using least squares method [1].

Single fluorophore detection algorithm (SFDA) [2] detects positions of fluorophores on an image series (video) obtained using TIRFM.

The first step of this algorithm is filtering out the noise by spatial and temporal filtering [2].

SFDA relies on the assumption that emmitation of fluorophores is a temporary action with an abrupt end [2]. Therefore the algorithm seeks significant value change of large image areas across subsequent frames. This value change is detected by an algorithm similar to edge detection with Laplace/Prewitt filter – with the only exception that the algorithm does not find difference of adjacent pixels in one image but in pixels with the same location on neighbour frames [2].

A mask is created from areas with the value change. The part of the images inside a mask probably depicts a fluorophore. The fluorophore first starts emitting the light (big positive value change) and after some time it immediately darkens out (big negative value change). The potential fluorophores with very short duration are probably false alarms and are filtered out [2].

The last step of SFDA is finding centers of the fluorophores using Gaussian fitting [2].

SFDA is designed to be followed by a tracking algorithm, which assigns detected points on subsequent images to each other using nearest-neighbour approach [2].

An algorithm called fluoroBancroft [3] detects centers of protein particles. It is inspired by the Bancroft method used to approximate the position of a GPS user.

The algorithm takes into effect two kinds of noises: background noise and shot noise. The background noise is caused by phosphorescence of the sample and unwanted excitation (illumination) of samples outside the region of interest. The shot noise is caused by photons falling at different parts of a camera inequally.

Distribution of PSF of a protein particle can be described by the Airy function. Andersson [3] simplifies it as a Gaussian and describes the distribution of the two kinds of noises using another two Gauss functions. The function of a pixel value can be estimated as sum of these three functions. In this function, the pixel value depends on its distance from the particle center and several constant variables of the system. This dependence is used to present a function, which finds estimate of position of the center (and therefore also distance from any pixel to it) using the pixel values.

The previously mentioned function is the cornerstone of the fluoroBancroft.

Computation of the function (and fluoroBancroft itself) has linear complexity depending on the image size [3].

FluoroBancroft proved to reach almost the same precision as the Gaussian fitting on simulated CCD[1] images while being multiple times faster [3].

Parthasarathy [4] assumes that all protein particles are "radially symmetric" in the image. Radial symmetry (also known as rotational symmetry) is a feature of an object which means that the object looks the same after rotation.

Therefore, locations of centers of particles lie in places with local maximum of radial symmetry [4]. Such an approach is about 100 times faster than fitting 2D Gaussian functions to the captured image [4].

Yoshida [5] detects stars on astronomical images. The stars look like small circular objects [5], similar to proteins particles analyzed in this thesis.

The image is split into a dark background and foreground containing the stars by thresholding [5]. The images, however, have a brighter center than the periphery [5] (probably vignetting caused by the camera). Global thresholding[2] can't be used due to this limitation so the threshold must be different for each image position.

Yoshida computes the threshold using a quadratic flatfield function.

First of all, parameters A–F of the flatfield function are computed. Then the function is subtracted from the image and the standard deviation σ of the pixel values is computed. Every pixel having value above 2σ is considered part of a star, other pixels are suppressed [5]. Adjoining sets of pixels are grouped together to form stars [5].

Hroch [6] also detects stars. The algorithm has to deal with several imperfections of the image caused either by the sensor (noise, hot pixels) or by

[1]Charge-Coupled Device – technology used in cameras to capture the images

[2]Thresholding algorithm using only one threshold for the whole image

presence of another astronomical object in the area of view (naebulæ, cosmic ray events).

Stars can be distinguished from hot pixels by looking at their profile [6]. Intensities of star pixels have approximately Gaussian profile while hot pixels are small dots with a sharp edge [6]. Hroch, therefore, introduces a new parameter called *sharp*. This parameter is defined as I_0/G_0, where I_0 is the maximum pixel value of the object (with background subtracted) and G_0 is estimated maximum of the object using neighbourhood values, if the pixel values had Gaussian distribution.

Cosmics and other objects often have elliptical shape, compared to almost perfect circular stars [6]. This difference is captured in *shape* parameter, which is used to filter out the non-star objects. *Shape* is defined as length of the line, where centers of isophotes[3] of the object lie [6].

Zheng et al. [7] detect astronomical objects of circular shape, probably also stars.

The algorithm presented in their paper focuses on detection of both bright and faint objects [7]. Especially the detection of a faint object lying next to a very bright one is a very challenging task.

This task is achieved using two steps: "global" processing of the whole image and "local" processing of irregular subregions.

The global part consists of smoothing using Gaussian filter, background subtraction, histogram equalization and detection of the objects using the Otsu method.

The local step begins with splitting the image into irregular regions using Watershed. Each region contains at least one bright star. Then the image is modified using various transforms to increase contrast and smoothed to remove the noise.

In the preprocessed image, objects are found using "layered object detection". This algorithm detects stars in iterations – each iteration contains preprocessing described above and then segmentation using the Otsu method. Objects detected using the segmentation are saved outside the image and then deleted from the image. Next iteration, therefore, can focus on fainter objects without being distracted by the bright ones.

The last step of the "local processing" is deblending – splitting of accidentally merged objects and merging outlying objects to their nearest neighbours.

Schöfer et al. detect cellular compartments using an ellectron microscope [8]. The compartments are represented by small point labels. The greatest challenge [8] faces is finding borders of the compartments. A compartment is represented by a set of points with no obvious border. There is also some background noise which makes the task even more difficult.

The approach [8] uses to address this problem is finding areas with a high density of particles.

[3]Isophote is a line (a loop) in the image where each pixel has the same value.

Those areas are located using an approximation of the expected intensity function. The expected intensity function is estimated by blurring a gray-level image, where the labels have maximum possible value and the background lowest possible.

The intensity function is thresholded to filter out the background and segment the foreground cellular compartments.

Glasbey and Roberts [9] analyze spatial distribution of immunogold-labelled particles. For each particle, distance to the nearest neighbour is computed. Two cummulative distribution functions are computed – CDF of the expected distance to the nearest particle from a randomly selected particles and CDF of the expected distance to the nearest distance from a randomly selected point (not necessarily particle) on the image.

The CDFs were compared to CDF of a Poisson process (the same number of particles scattered on the image with an uniform distribution of x and y) to find out whether the spatial distribution is random.

Chapter 2

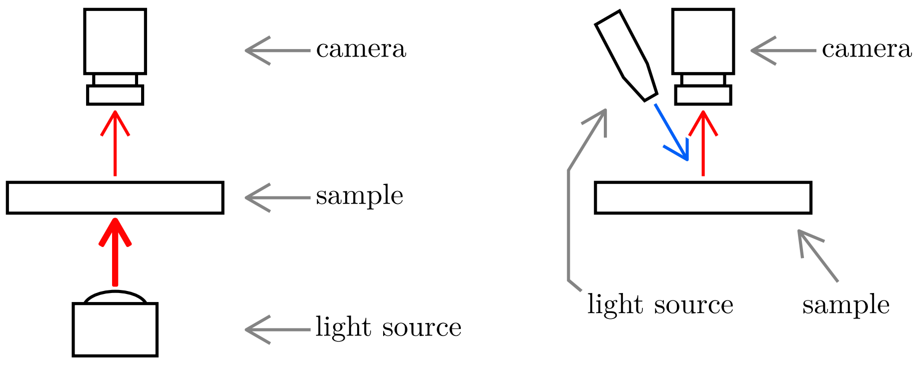
Terms and concepts

This chapter contains explanation of terms and concepts common in the area of image microscope processing, which are used in this thesis.

2.1 Fluorescence microscopy

Fluorescense microscopy uses features of some objects called fluorescence or phosphorescence to achieve images [10]. Both terms fluorescence and phosphorescence name ability of an object to absorb energy in for of light and heat and then emit it [10, 11]. The emitted light is then captured by a camera in the microscope.

Fluorescent objects emit light for a very short time, lower than 1 µm, while phosphorescent objects glow longer [10].



(a) Regular light microscope (b) Fluorescent microscope

Figure 2.1: Microscope type comparison

When capturing an image, it is necessary to send a light ray to the object of interest (“excite” the object) [10]. The object then starts to emit the light, which is captured by the microscope’s camera. Note that no external light is needed on the exact moment when the image was taken (only before it). This means, that the area of interest might remain completely dark, except for the fluorescent object [11].

Fluorescence microscopy uses special fluorescent molecules called "probes", which are attached to other molecules. Probes mark positions of the other type of molecules [11]. Therefore molecules of interest don't have to be fluorescent in order to be captured using fluorescent microscopy. There only must be an appropriate molecule type to be bound to them.

2.2 Structured Illumination Microscopy

Input images used by this thesis were created by Structured Illumination Microscopy (SIM). This technique is able to capture objects smaller than regular microscopy.

Regular light microscopes[4] are "diffraction limited". The diffraction limit makes them unable to capture objects lower than half of wavelength of lights they are using. Regular microscopes use human-visible light (wavelength about 400–800 nm [12]) so objects smaller than 200–300 nm are invisible to them [13].

SIM can bypass this limitation and photograph objects smaller than 50 nm [14].

Other advantages of SIM are the possibility to capture live cells (in contrary to e.g. electron microscope, where the object must be dead), speed, price and good contrast of the result [14].

2.3 Point spread function

Point spread function (PSF) is a function which maps a (infinitely small) point from the object plane[5] to the image plane.

In the ideal world, PSF would be another infinitely small spot [15]. Such an ideal PSF would result in a perfectly sharp image.

Due to many features of microscope design and physical limitations, the real PSF is never perfectly small.

Since the point is projected to an area, PSF of close points may overlap, which makes retrieving of the original point challenging.

PSF of a perfect optical system is Airy's function [15]. Airy's function is very similar to simpler Gaussian function. Therefore, the Gaussian function can be used to approximate the PSF.

[4]microscopes which depicts an object by capturing light that the object reflected or emitted

[5]real space that is being captured

Chapter 3

Input images

This chapter introduces the input microscope images.

Each sample is depicted in series of images. All of the images are grayscale with a dark background and light protein particles.

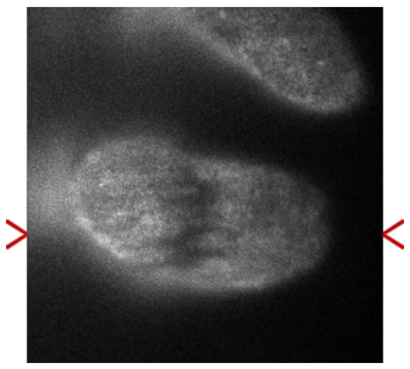


Figure 3.1: Example of an input image. The red arrows signalize the line visualized on figures 3.3 and 3.4.

The histogram of the pixel values (figure 3.2) signalizes that there is no clear border between the background and foreground pixels.

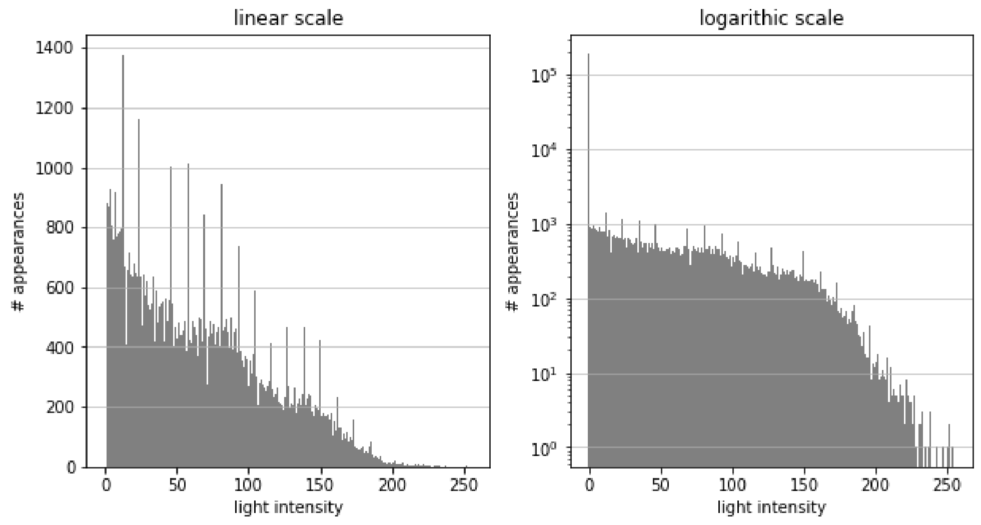


Figure 3.2: Histogram of the input image. Zero value has benn trimmed in the linear scale histogram for the sake of readability

The figure 3.3 displays values of the image on the line, that is marked by the red arrows on the figure 3.1. The figure 3.4 displays a detail of the same row.

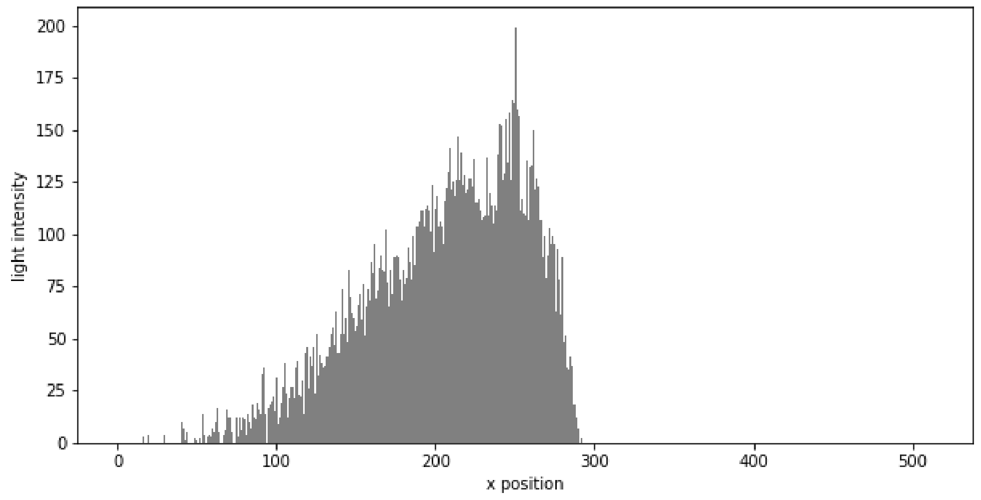


Figure 3.3: Values of pixels in one row of the input image

The histogram crearly shows that the gradients are very abrupt and that it is hard to distinguish a particle from noise based merely on the object's size.

The image also shows that presumable protein particles have different value across the image. Particles on the edges of the sample have lower value than the background on the center of the sample.

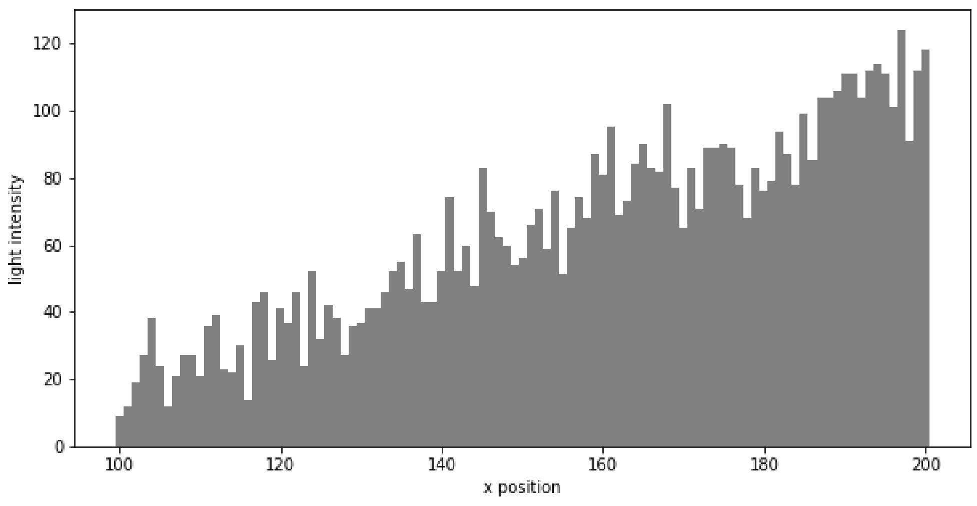


Figure 3.4: Values of pixels in one row of the input image – selection

As the image 3.5 shows, it is not a trivial task to find the protein particles on the image. The particles are blurred and the image contains a high amount of noise.

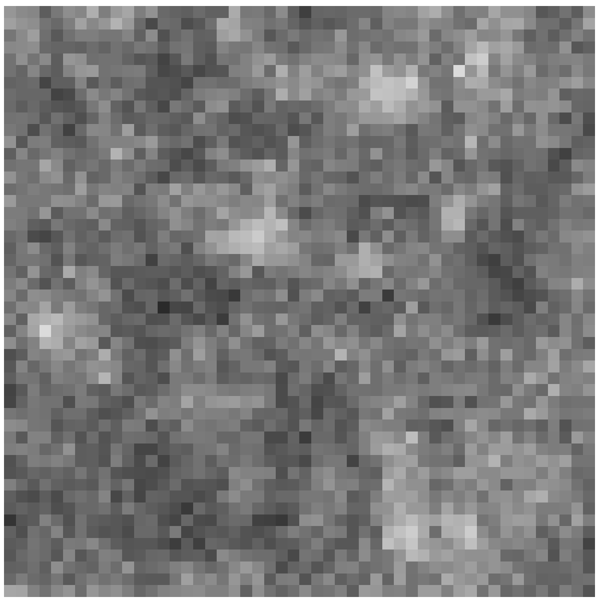


Figure 3.5: Selection of an input image

Chapter 4

Chosen methods

This section describes already existing methods chosen to be implemented, evaluated and used in the resulting algorithm.

4.1 Filters

A filter is an operation that aims to suppress low or high frequencies in the image [16].

A filter is often a convolution of the original function with some filter function. Such filters are called linear.

Equation 4.1 contains filtering of the original function f and the filter function h resulting in the output function g. Operator $*$ marks convolution.

$$g(x,y) = f(x,y) * h(x,y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(t,u)\, h(x-t, y-u)\, dt\, du \tag{4.1}$$

Computing the convolution of two discrete functions is very time-consuming. There are two approaches solving this issue. One of them is transforming the image to the frequency domain, multiplying both filters there and then transforming it back to the spatial domain [16]. Multiplying of signals in the frequency domain equals convolution in the spatial domain [16].

The other approach is estimating the filter function by a small matrix called kernel [16].

Equation 4.1 defines computation of a pixel $g(i,j)$ in a discrete linear filter [16]. Size of the kernel h is $M \times M$, where M is an odd number.

$$g(x,y) = \sum_{t=-\lfloor \frac{M}{2} \rfloor}^{\lfloor \frac{M}{2} \rfloor} \sum_{u=-\lfloor \frac{M}{2} \rfloor}^{\lfloor \frac{M}{2} \rfloor} f(x-t, y-u) \cdot h(t,u) \tag{4.2}$$

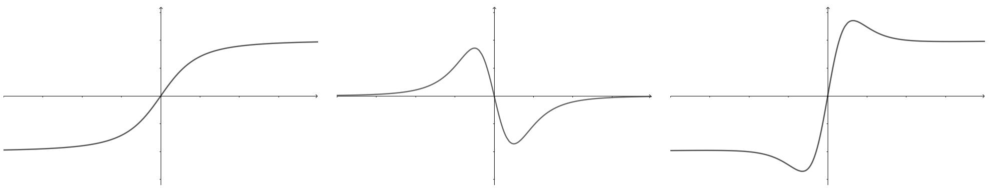
4.1.1 Laplacian filter

Laplacian filter approximates the second derivation of the image [17].

Discrete Laplacian filter uses eg. this 3×3 matrix as kernel [17]:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

A sharpened image can be obtained by subtracting the second derivation from the original image [17]. Edges in the sharpened image are sharpened because the second derivation suppresses the onset of the edge and elevates the finish of the edge (see fig 4.1).



(a) Original edge (b) Second derivation (c) Sharpened edge

Figure 4.1: Sharpening image using Laplace filter

4.1.2 Prewitt and Sobel operator

Both Prewitt operator and Sobel operator find estimate gradient in each pixel of the input image [18]. They use gradient to find edges in the image [18].

There are two variants of both operators, one detects horizontal gradients and the other detects vertical gradients.

Both variants of both filters consist of a 3×3 matrix (kernel) which is convolved with the image. Kernel for the horizontal filter is called H_x, the vertical one is H_y.

Figure 4.2 displays kernels for Prewitt and 4.3 for Sobel operator.

$$H_x = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \quad H_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

Figure 4.2: Kernels of Prewitt filter

A single image of gradient magnitude is computed as Euclidean distance of both convolved images: $|g(x, y)| = \sqrt{(g_x(x, y))^2 + g_y(x, y))^2}$ [18].

This equation is often simplified to $|g(x, y)| = g_x(x, y) + g_y(x, y)$ for sake of performace [19].

$$H_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad H_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Figure 4.3: Kernels of Sobel filter

Prewitt and Sobel perform both smoothing and gradient estimation [18]. The gradient estimation is thus less influenced by noise.

Their only difference lies in the smoothing kernel. Prewitt uses simple box smoothing $\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$. Sobel's kernel keeps higher weight on the center image: $\begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$.

Those kernels are convolved with a gradient estimation kernel $\begin{bmatrix} 1 & 0 & -1 \end{bmatrix}^\mathsf{T}$ (or functionally similar $\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}^\mathsf{T}$ for Sobel), which produces the kernels H_x above [18]. Vertical kernels H_y are computed similarly.

4.1.3 Wiener deconvolution

Wiener filter removes noise and de-blurs the image.

Each photograph or microscope image was created as a convolution of the original object with point spread function (PSF). Photographies also tend to have noise in them.

Wiener filter relies on the assumption that creation of the image g can be written as [20]:

$$g(x, y) = f(x, y) * h(x, y) + w(x, y) \tag{4.3}$$

where f is the original object, operator $*$ represents convolution, h is the point spread function and w is level of the nose.

The goal of deconvolution is to find an estimate of the original object $\hat{f}$ using a function r such as:

$$\hat{f}(x, y) = g(x, y) * r(x, y). \tag{4.4}$$

Convolution in the spatial domain is equivalent to multiplication in the frequency domain. Element-wise multiplication is much less complex operation than convolution. Therefore, all matrices are converted to the frequency domain using Fourier transform. Equation 4.5 is the frequency equivalent of the equation 4.4.

$$\hat{F}(u, v) = G(u, v)R(u, v) \tag{4.5}$$

To solve this equation and equation 4.4, it is necessary to find the function r or its frequency equivalent R.

The function $R(u, v)$ in the Wiener filter is defined using the power (absolute value) of the original signal $P_{ff}(u, v)$ and power of the noise $P_{ww}(u, v)$ [20].

$$R(u, v) = \frac{H^*(u, v)P_{ff}(u, v)}{|H(u, v)|^2 P_{ff}(u, v) + P_{ww}(u, v)} \tag{4.6}$$

The function $H(u, v)$ is the frequency equivalent of the PSF.

The equation 4.6 contains the power P_{ff} of the original signal, which is also subject of the computation. This issue can be partly addressed by dividing both numerator and denominator by $P_{ff}(u, v)$, which results in the equation 4.7.

$$R(u, v) = \frac{H^*(u, v)}{|H(u, v)|^2 + \frac{P_{ww}(u,v)}{P_{ff}(u,v)}} \tag{4.7}$$

The only unknown part of the right side of the equation remains $\frac{P_{ww}(u,v)}{P_{ff}(u,v)}$. This expression is division of the noise power by the power of the original signal.

Let's assume that the value of the expression doesn't change too much across the image – the ratio of noise to signal is always about the same.

The noise to signal ratio can be replaced by a constant $\alpha \in [0, 1]$.

The filter function is described in the equation 4.8.

$$R(u, v) = \frac{H^*(u, v)}{|H(u, v)|^2 + \alpha} \tag{4.8}$$

And finally, the estimation of the image in the frequency domain is:

$$\hat{F}(u, v) = \frac{H^*(u, v)G(u, v)}{|H(u, v)|^2 + \alpha}. \tag{4.9}$$

4.1.4 Gaussian filter

Gaussian filter is used to blur images [21]. It suppresses random noise but also blends details (unlike bilateral filter in section 4.1.5). Gaussian filtering is effective in removing Gaussian noise but not less effective in removing salt and pepper noise [21].

The kernel of the Gaussian filter is computed by the 2D Gaussian function:

$$G(x, y) = \frac{1}{2\pi\sigma^2} exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right), \tag{4.10}$$

where σ^2 is the variance of the Gaussian function (assuming that there is the same variance for the horizontal and vertical direction) and $exp(x)$ is the exponential function e^x.

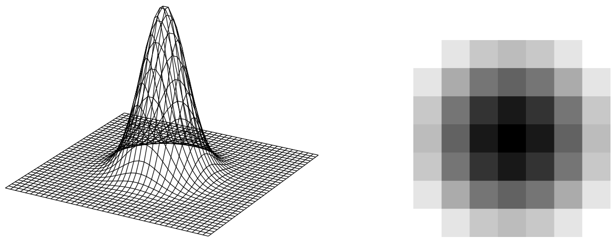


Figure 4.4: Visualization of a 2D Gaussian function and its discrete version

Value of each pixel in the blurred image is weighted average of other values in the image [22]. The central pixels in the Gaussian function have a higher value than the pixels on the periphery [21]. Therefore, value of each pixel in the blurred image is mostly influenced by its direct neighbourhood.

The Gaussian function must be discretized to be used as a kernel in discrete filter according to equation 4.2. The Gaussian function is never zero and thus the kernel function would be infinite. Therefore, the peripheral area of the function with low values has to be cut off [22].

It is also possible to speed up the calculation of the convolution by computing the horizontal and vertical components independently [22]. First, the image is convolved with a 1D Gaussian function in one direction and then the result is convolved with a 1D Gaussian in the other direction.

The 1D Gaussian is computed as:

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} exp\left(-\frac{x^2}{2\sigma^2}\right). \tag{4.11}$$

4.1.5 Bilateral filter

The bilateral filter is the only non-linear filter described in this chapter. It means that the filter can't be described as a convolution of some kernel with the image because the kernel is unique for each pixel [16].

Bilateral filter smooths images while preserving edges [23].

Blurring filters often compute the value of a pixel from its neighbours. Such filters rely on the assumption that the pixels around the currently computed pixel are similar because they depict the same object.

It s often true but this assumption fails if there is a sharp edge in the image. Gauss filter and other similar filters blur the edges.

Bilateral filter introduces a different definition of *similarity* of the pixels. It uses a combination of "closeness similarity" and "range similarity" [23] – a pixel is similar to another pixel if they are located close to each other and if they have similar values.

The most often way to compute closeness similarity in the bilateral filter is the Gaussian function but it is possible to use other functions [23].

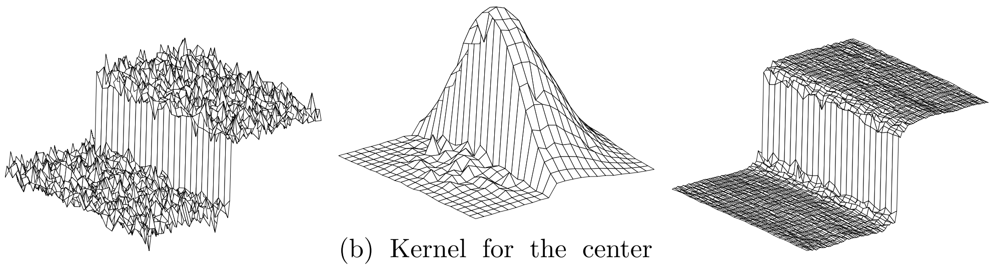

(a) Input image (b) Kernel for the center pixel (c) Filtered image

Figure 4.5: Bilateral filter. Image source: [23].

Kernel of the bilateral filter is computed for each pixel independently as multiplication of the Gaussian (or another) function and difference of the center pixel to the other pixels (see equation 4.12). Figure 4.5b shows an example of such kernel.

Equation 4.12 describes computation of discrete kernel h_b for pixel (x, y) using the closeness similarity function h_c and the range similarity h_r.

$$h_b(t,u) = \frac{h_c(t-x, u-y) \cdot h_r\left(f(x,y) - f(t,u)\right)}{\sum_{r=1}^{width} \sum_{s=1}^{height} h_c(r-x, s-y) \cdot h_r\left(f(x,y) - f(r,s)\right)} \tag{4.12}$$

A common closeness similairity function is the Gaussian function [24]. The range similarity often is the absolute value [23]. In such case, the kernel would look like this [24]:

$$h_b(t,u) = \frac{G(t-x, u-y) \cdot |f(x,y) - f(t,u)|}{\sum_{r=1}^{width} \sum_{s=1}^{height} G(r-x, s-y) \cdot |f(x,y) - f(r,s)|}. \tag{4.13}$$

4.2 Thresholding

Thresholding is a segmentation technique where pixels with value higher or equal specified parameter τ are considered to be part of the foreground and pixels with the value lower than τ are part of the background [25]. The input image is segmented into several foreground blobs divided from each other by the background.

There are three common types of thresholding which treat the parameter τ differently – global, adaptive and local.

Global thresholding uses the same value of τ across the whole image [25]. Adaptive thresholding computes the value of τ using the position of the pixel [25]. In the local thresholding, τ depends on the neighbourhood of the pixel [25].

4.3 Flat-field correction

Flat-field correction fixes effects of non-uniform illumination of a photograph [26].

Pixels in the center of a photograph tend to be lighter than those on the margin even when capturing the same object. This phenomenon is called vignetting and is caused by physical limitations of the lens and the aperture [27].

There is plenty of imperfections in camera sensor illumination beside vignetting. A large variety of approaches were created to eliminate them.

As Kask et al [26] point out, those approaches assume that there is a function with the pixel location as input, that influences the value of the pixel in the processed image. This function can be called shading function [26] or flat-field function [5].

Generally, there are two types of the flat-field function – additive functions whose values are added to the original image (added background) and multiplicative whose value is multiplied with the pixel values (vignetting or another illumination imperfection) [26].

Construction of the distorted image I using the true object image U is described in the function 4.14. S_M and S_A are multiplicative and additive flat-field functions.

$$I(x,y) = U(x,y) \cdot S_M(x,y) + S_A(x,y) \tag{4.14}$$

The true function can be estimated using estimations of the functions $\hat{S}_M$ and $\hat{S}_A$:

$$\hat{U}(x,y) = \frac{I(x,y) - \hat{S}_A(x,y)}{S_M(x,y)}. \tag{4.15}$$

A flat-field correction technique can use both of the flat-field functions or just one of them [26].

Yoshida [5], mentioned in the Research chapter, uses the additive flat-field function in equation 4.16 to estimate the background of an image depicting stars.

$$\hat{S}_A(x,y) = a + bx + cy + dx^2 + ey^2 + fxy \tag{4.16}$$

4.4 Region growing

Region growing is a segmentation technique that finds regions that satisfy some predefined similarity criterion [28].

The algorithm needs a set of starting pixels (seeds) which are then expanded.

The algorithm consists of three simple steps:

1. Choose one seed pixel. Insert the seed pixel into an empty set called "region".
2. Find all pixels neighbouring to any pixels from the region. Add them into the region if they are similar to the seed pixel. The similarity of the pixels is checked using the similarity criterion.
3. Repeat step 2 until there are no pixels to add.
4. Repeat step 1 with another pixel until there are no unprocessed seed pixels.

4.5 SRRF

Super-Resolution Radial Fluctuations (SRRF) is a purely analytical approach for increasing image resolution [29].

"Purely analytical" means that it can increase the resolution of already existing microscope images and doesn't require a specific process during obtaining the image – unlike methods like PALM, STORM and STED to which it is often compared [13, 29, 30].

The input of the algorithm is a series of images of point sources, not necessarily fluorescing protein [29]. The image recorded by a microscope resulted as convolution of two functions: original point sources and point spread function [29] (PSF, function describing how does the microscope record point source). The goal of the algorithm is to obtain an image as close as possible to the original point sources.

The algorithm relies on the assumption that the image of a particle convolved with the PSF is radially symmetric with center in the original position of the particles.

The word symmetry in image processing means invariance of the image to some transformation. Radial symmetry is invariance to rotation around the center.

Common approximations of the PSF like Airy's function or Gaussian function are invariant to rotation.

SRRF breaks every pixel in the image into subpixels.

A special transform called "radiality" is applied to each subpixel in the image. Radiality is computed inside a fixed-size window around the subpixel. This transform highlights subpixels with high radial symmetry inside the window [30].

To suppress the influence of noise, SRRF multiplies the radiality by the input intensity [29]. Noise in the image might be radially symmetric but it often has a low intensity.

If there are more input images in the input series, one single radiality image is created from them, which also decreases the level of noise [29, 30].

4.6 Morphological operators

Morphological operators is a set of image analysis techniques [31]. They extract image components such as boundaries, skeletons, convex hulls [31]. Those components can be used to analyze the shape of objects in the image [32].

Morphological operators are designed to process binary images but they were generalized to be used for grayscale or colour images too [31].

The image is analyzed using a matrix called “structuring element”. The structuring element has only binary values (can be also perceived as a set of points).

Most of the structuring elements are squares or circles. One point, mostly the center, of the structuring element is called the origin.

Each pixel of the output image is computed using the original input image and the structuring element with origin placed on the currently computed point. Pixels of the input image that are below positive pixels of the structuring element serve as an input of some operation – the operations are different for different morphological operators.

4.6.1 Basic operators

The operator called *dilation* expands the original object in the image.

Value of a pixel in dilated image is 1 if any of the input values (below the structuring element) are 1. The pixel value is 0 only if all of the values are also 0.

The operator called *erosion* reduces dimensions of the object in the image.

Value of a pixel in an eroded image is 1 if all the values below the structuring element are also 1.

Those two operators are combined to create another, more complex morphological operators.

4.6.2 Opening and closing

Operations called opening and closing are composed using dilation and erosion. Assuming I is the input image, they are defined as:

$$\text{opening}(I) = \text{dilation}(\text{erosion}(I)), \tag{4.17}$$

$$\text{closing}(I) = \text{erosion}(\text{dilation}(I)). \tag{4.18}$$

4.6.3 Top hat transform

Top hat transform suppresses slow trends in the image, while it lefts abrupt changes of values untouched [33]. For example, it deletes gradual changes of the background caused by vignetting. Objects smaller than the structuring element won't be affected by the transform [33].

Top hat enhances image contrast [33].

The top hat transform is obtained as [33]:

$$\mathrm{tophat}(I) = I - \mathrm{opening}(I). \tag{4.19}$$

A similar operation called bottom hat [33] is described by this formula:

$$\mathrm{tophat}^*(I) = \mathrm{closing}(I) - I. \tag{4.20}$$

While top hat retrieves light objects on a dark background, the bottom hat detects dark objects on a light background [34].

4.6.4 Reconstruction by dilation

The reconstruction by dilation operator removes objects smaller than the structuring element but doesn't (significantly) affect the bigger objects [35].

This operator doesn't use a structuring element but rather another image called a mask with equal size to the input image.

Morphological reconstruction can be understood as repeating of some operation (in this case dilation) until the output does not change.

The resulting image must "fit under" the mask. No pixel of the output image can have the value higher than the corresponding pixel in the mask. If it does, its value is lowered during every step to satisfy this constraint.

4.6.5 H-maxima

H-maxima transform suppresses all "domes", whose height is lower or equal a threshold h [35]. It also lowers the value of all pixels by h.

It is defined in equation 4.21, where $R_I^\delta(f)$ is reconstruction by dilation of f using the mask I.

$$HMAX_h(I) = R_I^\delta(I - h) \tag{4.21}$$

Chapter 5

Design

This chapter presents the algorithm used to detect and analyze the protein particles. The chapter describes the approach to the problem and how several types of algorithms were used to solve it. It also describes the measures used to describe distribution of the particles.

There are two possible design approaches: detecting particles on the image to be used as an input for further analysis or to estimate number of particles or other statistics measure using the raw image. Example of the measure using the raw image is estimation the number of particles using number of pixels above some threshold.

This thesis focuses on the protein particle detection approach.

The particle position data can be used to easily compute serveral statistical measures. Estimation using the raw image can provide only those measures, that have been previously implemented. For example, if the number of particles have been estimated, there is no easy way to use it to tell how close the particles tend to be to each other.

There are several downsides of the chosen approach too. It is more difficult to implement compared to estimation of several simple measures. There are also several features of the image that make the task challenging – like very close particles that are hard to distinguish from each other in the image.

5.1 Detection algorithm

The detection algorithms detects the particles in the image. The particles are represented by one pixel signalling their location and also by the areas they covers.

The algorithm for particle detection presented in this thesis consists of three phases: Preprocessing, Detection and Validation.

The algorithm uses simple image processing methods rather than methods of machine learning. Although machine learning algorithms (like neural

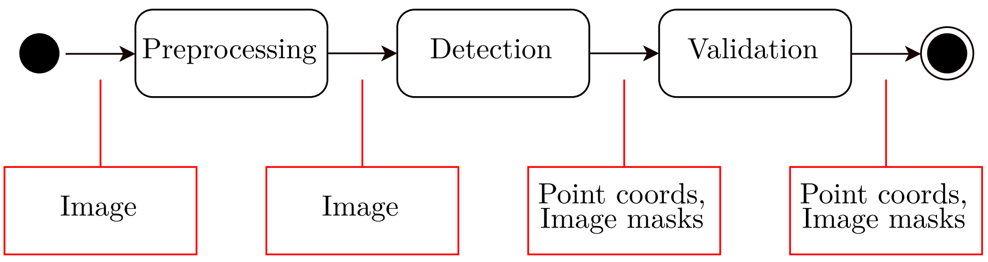


Figure 5.1: Components and dataflow of the particle detection algorithm

network) might be better for the task, they usually need a big data set to be evaluated.

The input dataset contains a limited set of images with thousands of particles in them. Using a machine learning technique would probably result in overfitting and inability to process other images of a different type of protein (maybe even different image of the same protein types). Furthermore, annotation of all the particles on those images would be a very challenging issue.

5.2 Preprocessing phase

The preprocessing phase modifies the image to increase the detection performance. This phase is optional.

Both the input and output of the preprocessing phase is an image.

There can be more than one algorithm in the preprocessing phase. The output image of one algorithm is the input of another.

These preprocessing methods were used:

- SRRF
- Correction using flatfield function
- Background suppression using flatfield function
- Laplacian filter
- Gaussian filter
- Bilateral filter
- Wiener deconvolution
- Top hat
- Image gradient detection
    - Prewitt operator
    - Sobel operator

5.2.1 SRRF

SRRF (section 4.5) is an image upsampling technique. It highlights the local maximums and deepens ridges between them.

Therefore, this approach can be combined with all detection algorithms mentioned in section 5.3.

SRRF also improves the performance of the preprocessing methods.

SRRF can be added before other algorithms or after them. Putting it before is generally a better approach because the other preprocessing algorithms might use the upsampled image (while SRRF wouldn't utilize preprocessed image too much).

SRRF also has several drawbacks.

An expected particle size (window radius) must be set before the algorithm runs. Choosing a wrong parameter decreases performance of the whole particle-detection process.

SRRF is also more time-consuming than other algorithms, although the parallel computation of radiality makes this disadvantage less significant.

5.2.2 Flat-field correction

In this method, the background of the image is detected and subtracted using the flatfield function.

This method is inspired by Yoshida [5], who uses the flatfield function in this form:

$$S_A(x, y) = a + bx + cy + dx^2 + ey^2 + fxy, \tag{5.1}$$

where x and y are coordinates of the pixel in the image and a–f are coefficients computed before the first run of the algorithm.

There are two possible usages of the flatfield function.

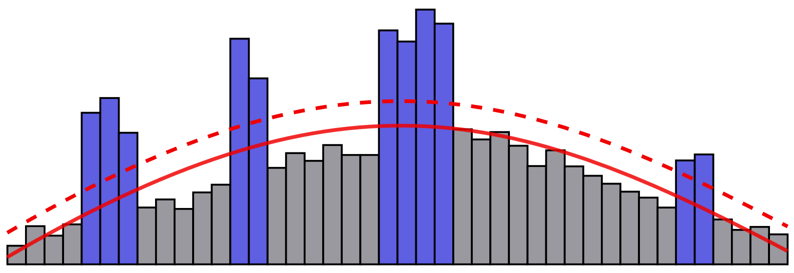
Yoshida erases every value lower than $S_A(x, y) + k\sigma$.

Constant k must be set manually (Yoshida uses $k = 2$) and σ is the standard deviation of pixel values. The standard deviation is approximated as the average of $|I(x, y) - S_A(x, y)|$.

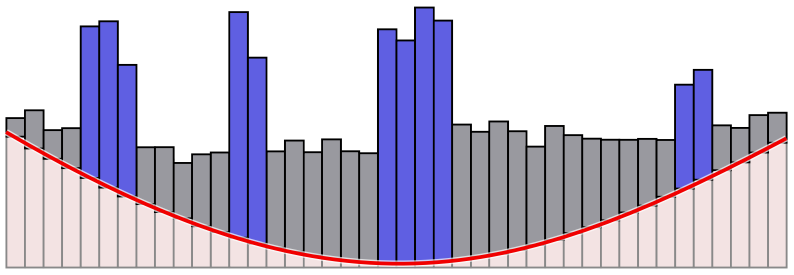
The second possible usage is to subtract the background estimation $S_A(x, y)$ from the image.

Both approaches were tested in this thesis but none of them noticeably improved the detection.

Yoshida doesn't explicitly mention how the coefficients a–f are computed. Therefore, an approach for their computation was created using the least squares method.



(a) Thresholding of the image using flatfield function. The solid line visualizes the flatfield function, red dotted line is $Flat(x, y) + c\sigma$. Pixels below the dotted line are suppressed.



(b) Normalization using flatfield function. The red line visualizes flatfield function multiplied by -1. The flatfield function is subtracted from the image (red line is added).

Figure 5.2: Thresholding and normalization using flatfield function. The blue pixels visualizes protein particles, gray pixels are background.

5.2.2.1 Computation of the flatfield function

The flatfield function contains coefficients a–f, which have to set before program run. They can be set either manually by the user or calculated from the image. The automatical calculation allows processing of very diverse images and doesn't confuse the user with various parameters to be set.

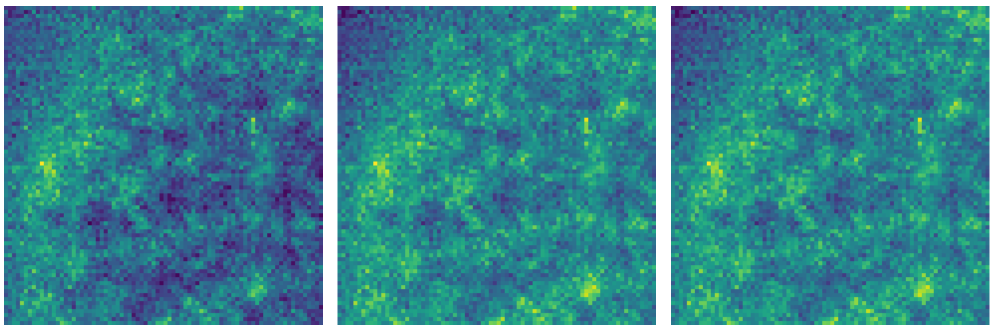
The coefficients are computed using a reference image. If there is just one input image, it is also used as the reference image. This approach is called retrospective flat-field correction.

If the input is a series of images, a "median image" is computed – see section 5.2.2.2. This series of images is specified before the algorithm runs. Such correction is called prospective.

Such coefficients are chosen, that minimize difference (error E) between the flat-field function and actual numbers of a reference image:

$$E = \sum_{x=1}^{width} \sum_{y=1}^{height} (I(x, y) - S_A(x, y))^2. \tag{5.2}$$

The lowest possible error is computed using the least squares method. The



(a) Original image (b) Retrospective flat-field correction (c) Prospective flat-field correction

Figure 5.3: Images processed by the flat-field correction. The image has been zommed in to the top left edge of the image so the difference is more visible. Nevertheless, the correction is very subtle.

aim is to find the minimum of this function:

$$E = \sum_{x=1}^{width} \sum_{y=1}^{height} (I(x,y) - (a + bx + cy + dx^2 + ey^2 + fxy))^2. \tag{5.3}$$

Gradient of a function should equal zero vector in the minimum. It means that partial derivation of the function by each coefficient should equal 0. Therefore partial derivation for each coefficient is created, like this one for C:

$$\frac{\partial E}{\partial c} = \frac{\partial \sum_{x=1}^{width} \sum_{y=1}^{height} I(x,y)^2 - 2I(x,y) + S_A(x,y)}{\partial c} \tag{5.4}$$

$$\frac{\partial E}{\partial c} = \sum_{x=1}^{width} \sum_{y=1}^{height} -2yI(x,y) + 2cy + 2y(a + bx + dx^2 + ey^2 + fxy) \tag{5.5}$$

Setting the partial derivation expression to 0 and moving the image-related member to the another side of the equation will result in:

$$\sum_{x=1}^{width} \sum_{y=1}^{height} ay + bxy + cy + dx^2y + ey^3 + fxy^2 = \sum_{x=1}^{width} \sum_{y=1}^{height} yI(x,y) \tag{5.6}$$

The coefficients can be distributed outside the sum, which will lead in a equation of 6 variables.

$$a\sum y + b\sum xy + c\sum y + d\sum x^2y + e\sum y^3 + f\sum xy^2 = \sum yI(x,y) \tag{5.7}$$

The previous equation contains simple sums instead of $\sum_{x=1}^{width}\sum_{y=1}^{height}$ for the sake of readability.

Similar equations are computed for remaining coefficient resulting in 6 equations of 6 variables, which can be wrote in matrix form and solved using Gaussian elimination method.

5.2.2.2 Computing reference image

A reference image is computed if there are more input images specified.

Every pixel in the reference image is computed as median (or similar measure, see below) of the particular pixel across all the images.

Using the median helps to reduce the influence of the proteins on the flat-field function so it only describes the background. Mean (or another similar measure) of the pixels might be influenced by very high pixels of the foreground.

However, using the median might fail if the pixel very often contains a protein particle. Median is a value that splits a sorted array of values into two halves, each containing 50% of values. If the appearance of a protein is very often (more than 50% of the frames), then the median will be influenced by the foreground, not background.

A new measure was introduced (called “generalized median”), which splits the sorted array into first half containing k% of values and the second one containing the rest. The parameter k can be any real number between 0 and 1 (including).

The purpose of the generalized median was to find a spot where the pixel is influenced only by the background but not by outlier values caused by the noise.

The tests have shown that the performance of the algorithm doesn't depend on the value of k. It means that the generalized reference image didn't bring any advantage over the regular median.

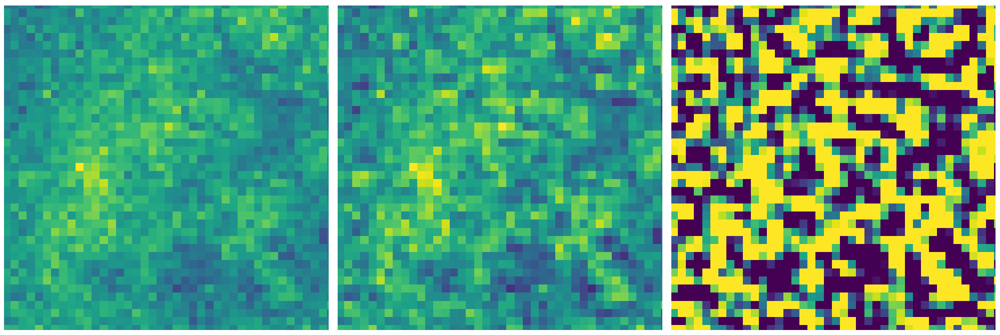
5.2.3 Laplacian filter

Laplacian filter is described in section 4.1.1.

The Laplacian filter is approximation of the second derivation of the image. The approximated second derivation is multiplied by a parameter k and then subtracted from the original image.

The parameter $k \in R$ controls intensity of sharpening. Too big k causes the image to be noisy. Too low k makes the image blurry. A common value for k lies between 0 (equals to the original image) and 1 but it is possible to have k higher than 1 or even negative. A negative k blurs the image instead of sharpening.

Another parameter of the algorithm is the size of the kernel. The default Laplacian filter has kernel of size 3.



(a) Original image

(b) Image after applying Laplacian filter, $k = 0.02$, kernel size=5

(c) Image after applying Laplacian filter, $k = 0.3$, kernel size=5

Figure 5.4: Images after application of the Laplacian filter

Laplacian filter was intended to be used together with region growing algorithm and with combination of local maximum and h-maximum. The sharper edges and a bit deeper “valley” between the protein particles were supposed to stop the growing algorithm or to prevent h-maximum to accidentally merge two particles. It may (in theory) help the thresholding detector for the same reason.

The greatest disadvantage of the algorithm is that it intensifies noise in the image. It showed up that this disadvantage surpassed its possible advantages. Alorithms containing Laplacian filter achieved worse results than those without it.

5.2.4 Gaussian filter

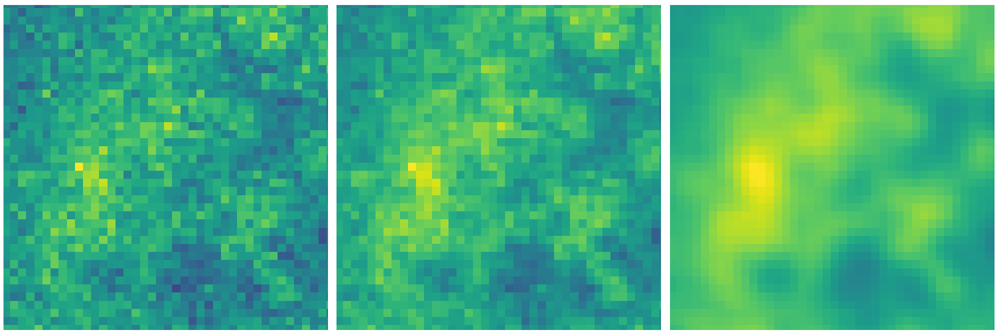
Gaussian filter smoothens the image, which prevents false detections for algorithms based on local maximums (region growing, local maximums) and preprocessing methods that search for gradients (Prewitt operator, Sobel operator, Top hat).

Its disadvantage is that it also deletes some features. It might erase ridges between protein particles and even blend the particles together.

5.2.5 Bilateral filter

Bilateral filter smoothens the image and therefore it has the same advantages as Gaussian filter described above.

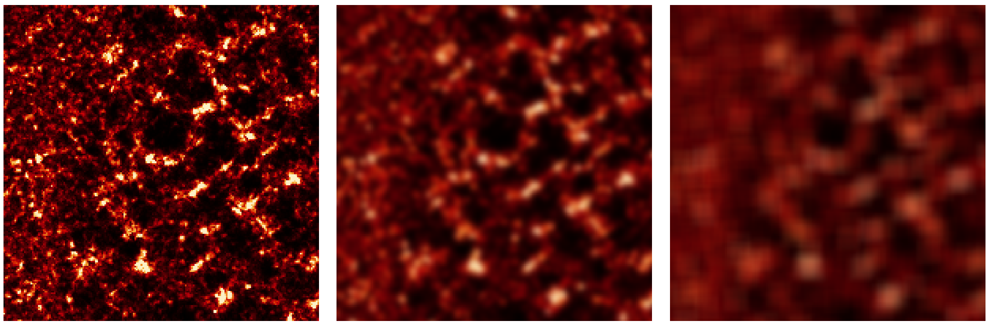
When comparing to the Gaussian filter, the main advantage of the bilateral filter is its ability to preserve sharp edges. This means that the ridges between the particles have a bigger chance to remain in the image.



(a) Original image

(b) Image after applying Gaussian filter, $\sigma = 0.5$, kernel size=5

(c) Image after applying Gaussian filter, $\sigma = 2$, kernel size=5

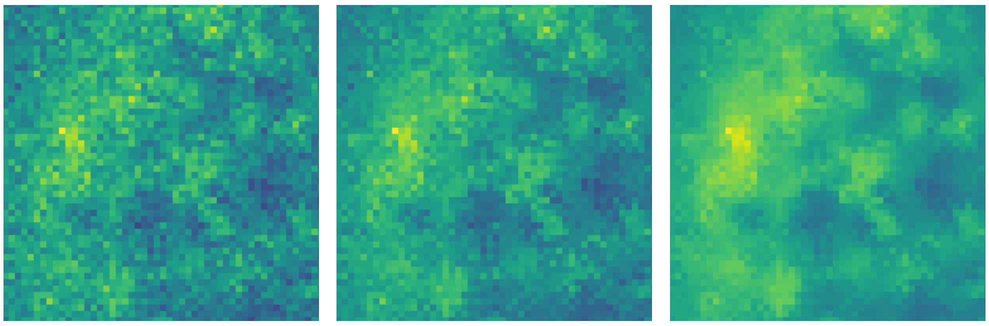


(d) Original SRRF preprocessed image

(e) SRRF preprocessed image after applying Gaussian filter, $\sigma = 2$, kernel size=15

(f) SRRF preprocessed image after applying Gaussian filter, $\sigma = 10$, kernel size=15

Figure 5.5: Images after application of the Gaussian filter

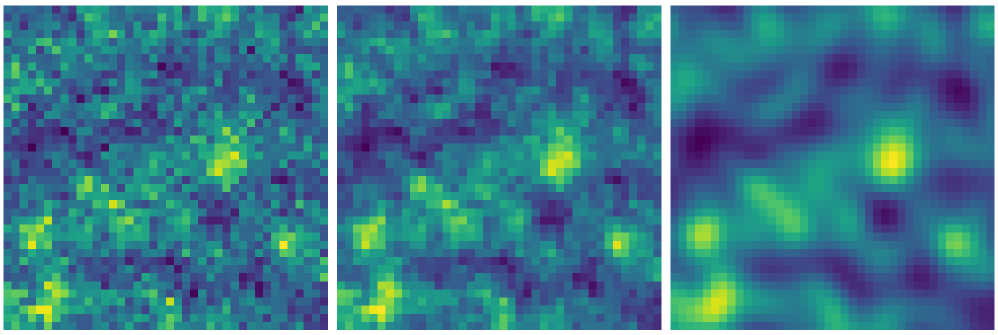


(a) Original image

(b) $\sigma_{space} = 100$, $\sigma_{color} = 50$

(c) $\sigma_{space} = 2$, $\sigma_{color} = 100$

Figure 5.6: Images after application of the Bilateral filter



(a) Original image

(b) Image after applying Wiener filter, $\sigma^2 = 0.1$

(c) Image after applying Wiener filter, $\sigma^2 = 10$

Figure 5.7: Images after application of the Wiener filter, $\alpha = 0.95$

5.2.6 Wiener filter

The Wiener filter (described in section 4.1.3) removes noise and estimates the original image before application of the PSF.

The Wiener filter was intended to sharpen the image and remove noise.

Local maximum detection algorithms often struggle with noise which is falsely detected as a local maximum. Problem of the thresholding algorithm is insufficient segmentation due to very shallow ridges between the particles. This issue is caussed by convolution of the original perfectly segmented particles with the PSF, which blurred the image.

Gausian function was asumed to be the PSF. The Gaussian funciton has a parameter σ (see equation 4.10), which has to be estimated.

Another parameter of the algorithm is the noise to signal ratio, noted α in equation 4.9.

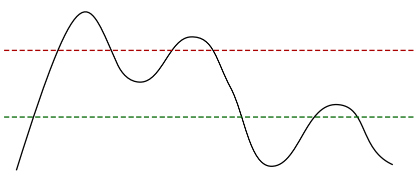
Testing of the Wiener filter revealed that it doesn't enhance performance of the local maximum algorithm. It also didn't enlarge the ridges between particles.

Resolution of the input image is probably too low to capture the particles in more than few pixels. When transformed to the frequency domain, the frequencies describing the particles are very high ones – as well as the frequencies describing the noise. The algorithm, therefore, can't differentiate noise from the particles.

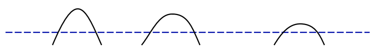
5.2.7 Top hat

The top hat transform (section 4.6.3) removes low frequencies and keeps only objects smaller than the structuring element. Therefore, it is able to remove gradual background change.

The structuring element of top hat is a circle with the diameter of the biggest expected particle size. Top hat with this structuring element removes

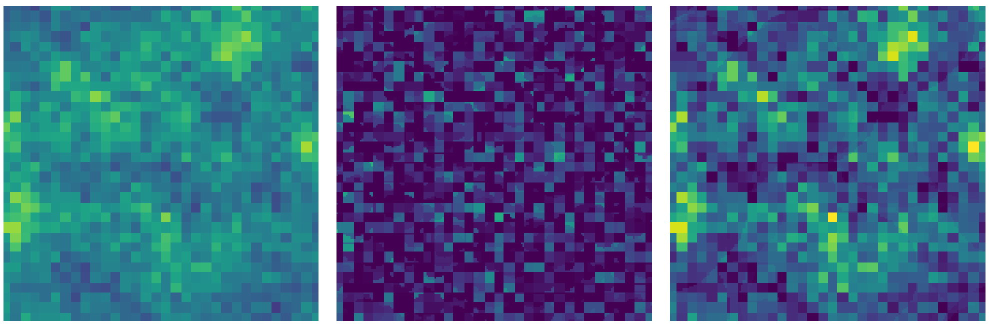


(a) Original image, none of the two threshold segments the image perfectly



(b) One threshold detects all the particles after applying top hat

Figure 5.8: Thresholding with top hat transform and without it



(a) Original (b) Kernel width = 11 px (c) Kernel width = 61 px

Figure 5.9: Result of top hat transform with different kernel sizes.

everything except for the protein particles.

Top hat also eliminates one problem of thresholding, which is uneven brightness of the particles and clusters of very close particles that might partially overlap (see Figure 5.8). This feature of top hat allows one threshold to segment the whole image.

The greatest disadvantage of top hat is its inability to process clusters of very close particles. Such a clusters might be completely erased.

If there are not enough ridges between the particles, top hat detects them as one big object. This object doesn't fit inside the structuring element so it is removed.

5.2.8 Prewitt and Sobel operators

Both Prewitt and Sobel operators (section 4.1.2) estimate gradient magnitude in the image and thus may find edges of the protein particles.

Three modifications of both operators were created:

1. pure Prewitt/Sobel operator estimating the first derivation of the image,

2. double Prewitt Sobel operator using the second derivation of the image (repeated operator two times).

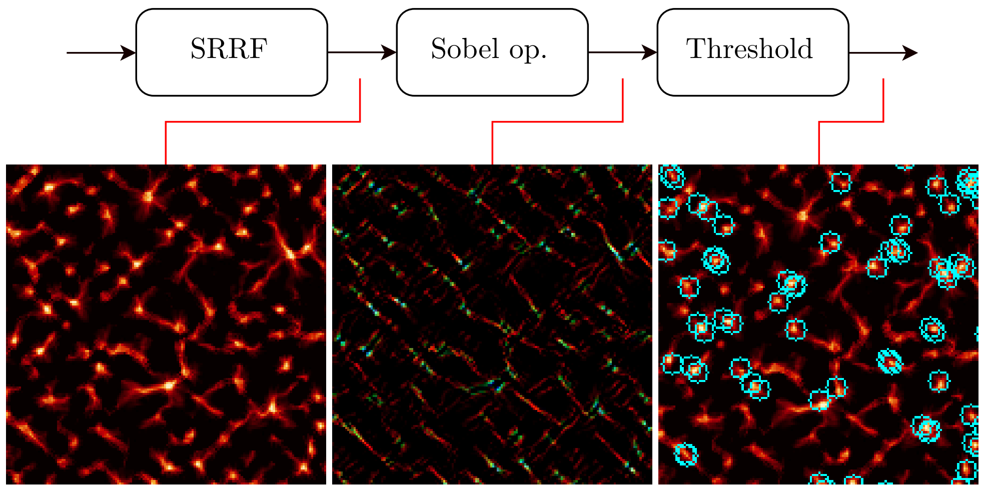


Figure 5.10: Detector using gradient magnitude created by the Sobel operator

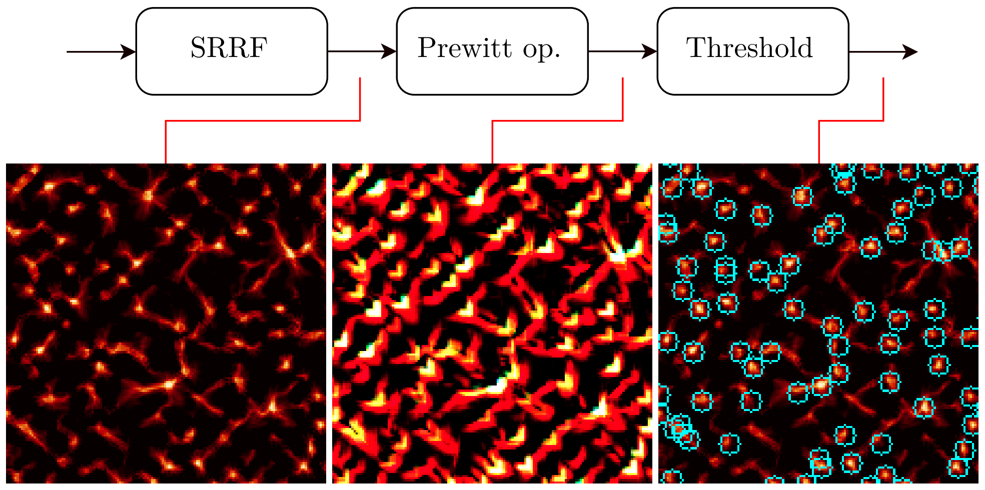


Figure 5.11: Detector using gradient magnitude created by the Prewitt operator

3. Original image with the first derivation subtracted from it.

   This method helps to segment pixels because it highlights the “ridges” between protein particles.

Kernels of the operators were modified to detect slowly graduing edges. The kernel size was increased to $n \times n$ where n is any odd number greater or equal 3. For H_x, every first and n-th column contains -1 for Prewitt operator and 1-D gaussian for Sobel operator.

The first derivation of both operators was intended to be used with thresholding as a single detector.

5.3 Detection phase

During the detection phase, particles are located in the image. A particle could be represented either by its position (one point) or as a set of adjacent pixels represented by an image mask. This phase consists of only one detection algorithm.

The input of the detection phase is an image and the output is a list of points and/or list of image masks.

These detection methods were used:

- Local maximums (non-maximum suppression)
- Region growing
- Thresholding

5.3.1 Local maximums

The local maximum algorithm detects points that are local maximums in their neighbourhood. The detection algorithm is also supposed to be able to segment the image for further processing, therefore the basic non-maximum suppression algorithm was modified to accomplish it.

Two slightly different approaches were tested.

The first approach suppresses all values in the image that are not maximums in their 4-neighbourhood[6]. Each group of adjacent pixels (in 4-neighbourhood) covers one detected particle.

The second approach uses a circle mask of diameter r, where r is an odd number specified as a parameter. The mask is moved across the image. The pixel in the center of the image is considered local maximum if there is no other pixel covered by the circle mask with the value greater than the center pixel.

If the pixel is a local maximum, it is marked as the center of a particle. All pixels inside the circle are considered to be part of the particles and are removed from teh image.

The second approach is able to segment big clusters of particles. Particles detected by the second approach have greater areas – more points are detected as local maximums and appended to a particle because the actual adjacent local maximums have been erased.

The first approach focuses only on the actual local maximums and the rest of the image is left unsegmented. See Figure 5.12 where the Plateau approach detects only “plateaus” of local maximums while the deep algorithm also segments non-maximum pixels.

[6]direct left, right, bottom and top neighbours of the pixel

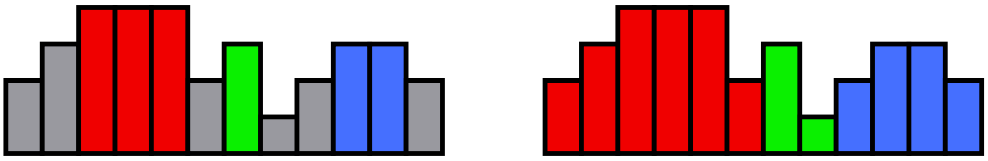

(a) 1st approach (b) 2nd approach

Figure 5.12: Segmented 1D arrays using different types of local maximum algorithm. Colour of the pixel denotes particle to which the pixel belongs. Height of the bar signalizes pixel value.

The red pixel on the red-green border in the image might also belong to the green particle and its colour depends only on the order of processing. Same applies to the green pixel on the green-blue border.

5.3.2 Region growing

Region growing algorithm, described in section 4.4, finds regions that satisfy some similarity criterion.

The region growing algorithm doesn't specifically describe the similarity criterion or the seed selection approach. Those parts of the algorithm depend on its application.

The protein particles are local maximums. Therefore, a very simple local maximum algorithm was created to find the seeds: a pixel is selected if it is a local maximum in its 4-neighbourhood. The seeds also must have value higher than a constant τ_1. This local maximum algorithm is simpler than the two local maximum algorithms introduced in the section 5.3.1 but it is sufficient.

The similarity criterion involves both spatial and color similarity.

Two pixels are similar if their Eucleidian distance is not higher than the expected maximum particle size and the difference of their values is not higher than a constant τ_2.

All overlapping regions are merged together because they probably contain the same particle.

Lowering the constant τ_2 makes the algorithm more sensible to local maximums but there's a risk that one point will be detected twice. Higher τ_2 might result in an accidental merge of two particles.

5.3.3 Thresholding

This algorithm applies global thresholding, described in section 4.2, on the image and groups together adjacent pixels above the threshold. Each group of pixels covers one protein particle.

Only the global thresholding was implemented. It means that the thresholding parameter τ is the same for the whole image. This approach is simpler than the others but it is possible to simulate more advanced types of thresholding using preprocessing. For example, flatffield correction applied before thresholding is similar to thresholding with the variable parameter τ depending on the image position.

Experimenting with the flatfield function showed that the input images don't suffer from significant vignetting and thus global thresholding is sufficient.

There are several downsides to the thresholding algorithm. If the threshold is too high, some valid points are not included in the result. If the threshold is low, adjacent proteins might not be segmented (the rigde between them has too big values).

This issue was addressed by various preprocessing and validation techniques:

- Top hat transform extracts small objects from the image. It suppresses ridges between the particles, which allows to segment the particles. The top hat transform also reduces the effect of non-uniform illumination [34].

  On the other hand, it removes big objects from the image. If there is a big cluster of particles with no significant ridges between them, top hat transform completely clears it.

- Laplacian filter and top hat – Laplacian filter highlights edges but might also bring more noise to the image. Laplacian filter was supposed to magnify ridges in big clusters, which is the biggest weakness of the top hat transform.

- Flatfield correction suppresses vignetting and transforms the global thresholding to de facto adaptive thresholding.

Otsu method for estimation of the parameter τ was also tested. It was supposed to find the spot between the foreground and background but it didn't perform well.

5.4 Validation phase

The validation phase filters out false detections and merges particles that were accidentally split by the detector. Like preprocessing, the validation phase is optional and it could consist of a series of validation algorithms.

Both input and output of the validation phase is a list of points and/or list of image masks.

These validation methods were used:

- H-maximum

- Protein size validator

5.4.1 H-maximum

H-maximum validation method is inpired by the h-maxima morphology operator (section 4.6.5). H-maxima removes all "domes" (objects that are local maximums) with height lower than a threshold h.

The morphological operator might be used in the preprocessing phase but it would be time-consuming. Therefore, a simple validation approach with similar function was created.

The validation algorithm removes objects that are not local maximums or are not delimited by a deep ridge from the rest of image.

It also serves as a segmentation algorithm.

It is defined by the following steps:

1. Select a particle $p = (x, y)$ from the set of particles detected by the Detection phase.
2. Remove the particle if $I(x, y) < h$.
3. Let S be set of pixels that have value higher than $I(x, y) - h$. All pixels in S form one segment of adjacent pixels. (It can be visualized using the green/red areas in figure 5.13.)
4. Check whether there are any another particles detected inside S.
   a) If not, continue to 5.
   b) If yes and p is the highest of them, removed those particles and continue.
   c) If yes and there is a particle whose value is higher than p, removed p.

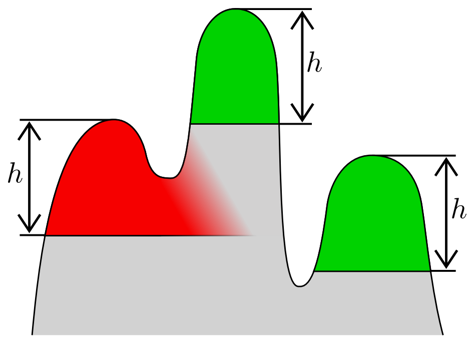


Figure 5.13: Example of H-maximum validation on a 1D image. Two objects identified by local maximums were detected in the previous step. The green objects are higher than h. The red object is not separated from the center object by a ridge of depth at least h. The green objects are perserved, the red one will be filtered out (merged to the center object). Image from [36].

5. Proceed to 1 and select another particle until there are no unprocessed particles.

6. All particles that weren't removed are the result of the algorithm.

The h-maximum algorithm doesn't only filter out the particles but it also segments the image. All pixels in set S belong to the particle p.

5.4.2 Protein size validator

Validator of protein size counts all pixels that have been assigned to the particles in the detection step.

The validator aims to delete noisy pixels that have been accidentally detected as proteins. If the detected area is too small, there is a big chance that the detected object is not protein particle.

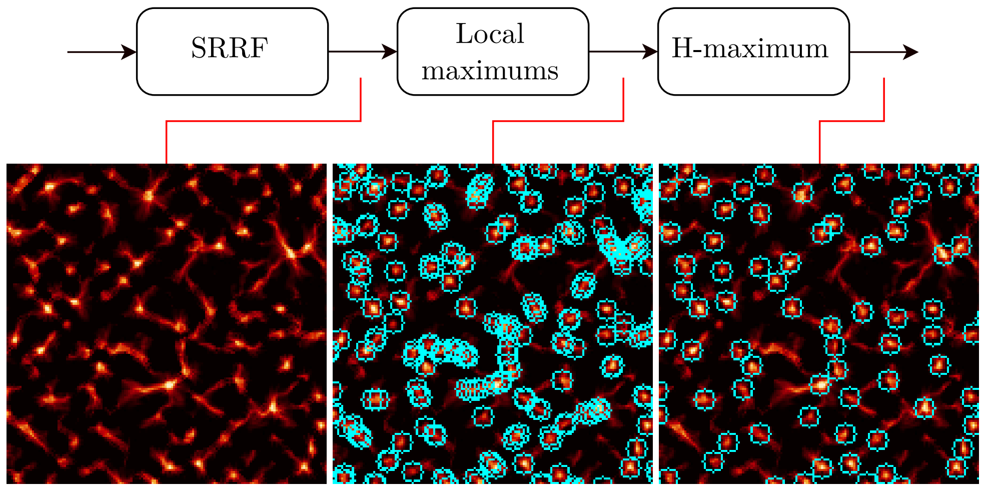
The size validator relies on the quality of the segmentation algorithm. This turned out to be the greatest weakness of the algorithm.

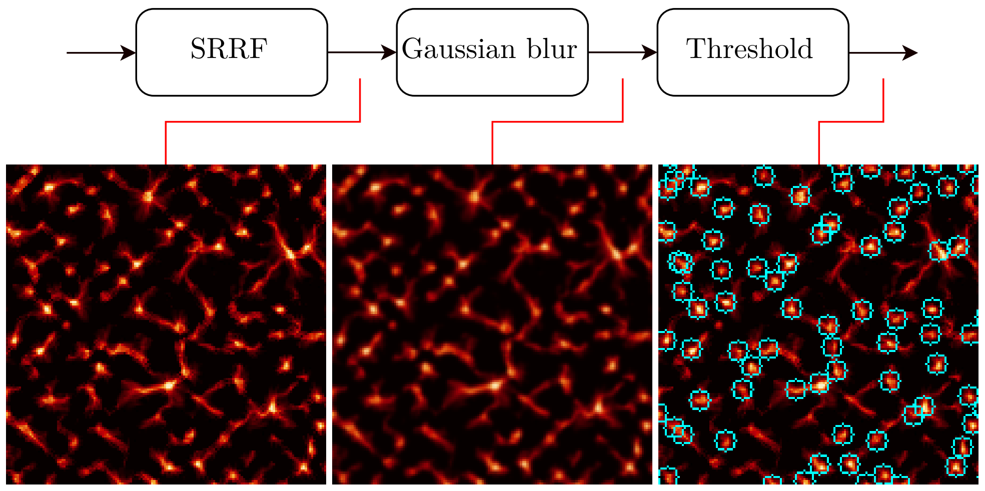
The size validator didn't bring additional performance even on sole detection algorithms without preprocessing. Region growing and threshold algorithms were tested because the local maximum algorithm finds particle centers only.

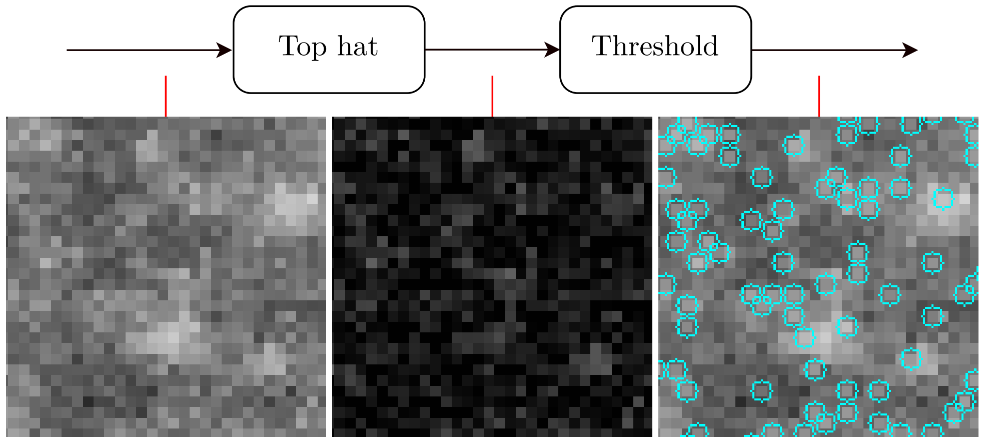
5.5 Selected combinations

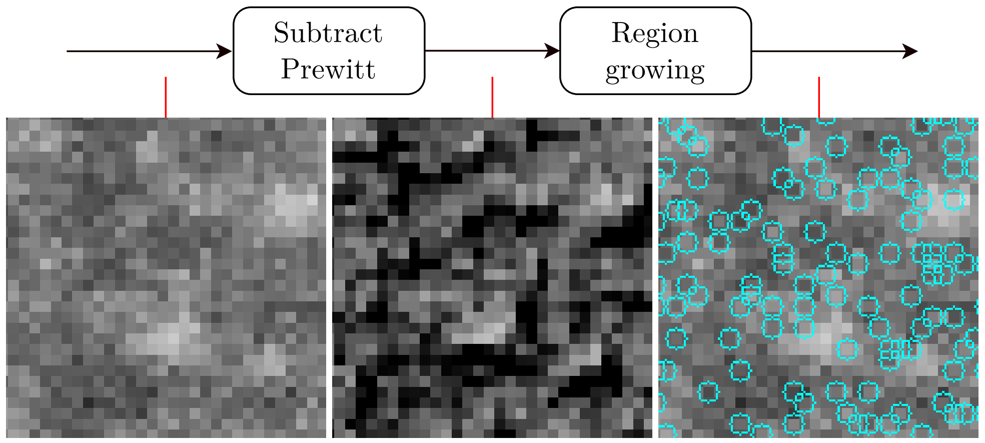
These sequences of image processing methods proved to be succesful for protein particle detection. Two combinations with SRRF and two without it were chosen. Their results can be found in chapter 7.

The first algorithm performed best during testing and therefore it was chosen as the definite result of the detector design task.




SRRF
Gaussian blur
Threshold


Top hat
Threshold


Subtract Prewitt
Region growing

5.6 Distribution analysis

The goal of the thesis is to support biologists in finding differences in images of (presumably) two different types of proteins.

This thesis focuses only on very generic techniques. It is not possible to know which features of the distribution to focus on without good knowledge of proteins.

Therefore, a very important part of the distribution analysis is creation of a visual representation of the distribution. A person trained in the field can use the representation to find some important features in it.

Another part of the distribution analysis is answering these three questions:

1. How many particles are in 1 mm^2 of the sample?
2. Do the particles tend to form clusters or are they spaced evenly?
3. Is the density of the particles evenly distributed across the image?

5.6.1 Particle count

Answering the first question, although simple, might be very useful. For this reason, number of the particles in each image is counted.

The images often have a different size so the number of particles must be divided by number of pixels in the image and then multiplied by the pixel-to-mm conversion ratio.

5.6.2 Expected intensity function

Estimation of the expected intensity function visualizes the density of particles in the image and highlights the areas, where the density is high.

This method is inspired by Schöfer et al. [8], who use the intensity function to find borders between cellular compartments.

The visualisation algorithm consists of the following steps:

1. Positions of the protein particles are drawn to a blank image. Each particle is a small white circle.
2. The image is blurred. Schöfer et al. [8] use a cone as the blur filter kernel. In this thesis, the Gaussian function is used.
3. Areas with high density are highlighted using the thresholding algorithm (the threshold obtained by the Otsu method). The edge between background and foreground is marked in the image.

Unlike Schöcher et al. [8], the thresholding in this thesis only highlights the densest areas and doesn't demarcate any object.

5.6.3 Nearest neighbour analysis

The nearest neighbour analysis provides a numerical value that describes whether the particles are clustered or uniformly spaced.

The nearest distance uses distances between each point and its closest neighbour [37]. The mean of those distances is computed to obtain a single value [37]. This value is supposed to be compared with the expected mean distances [37].

It is also possible to use the data differently than just computing the mean value. Glasbey and Roberts [9] use the cumulative distribution function of the distances.

In this thesis, the distances were visualized in a box plot. The box plot contains more information than a single value and offer a wider possibility of comparison – including the common mean, which is marked on the box plot.

The plots are compared to similar plots of a randomly generated image with the same size and the same number of particles (similarly to Glasbey and Roberts [9]).

If the particles tend to form clusters, the distances to the closest particle will be lower than distances between more evenly distributed particles [9].

5.6.4 Ripley's K–function

Ripley's K-function is a tool for analyzing samples of spatial random processes [38].

If $K(r)$ is the K-function in a Poisson process with density λ, then $\lambda K(r)$ refers to the expected number of points in a ball with radius r around a randomly chosen point of the point process [39]. The center point itself is not counted [39].

The particles located in the image can be interpreted as a random process. The value of K-function of a particle is the number of particles with distance up to r from the currently processed particle.

This metric can be used to answer both questions 2 (clustering) and 3 (density).

If the particles tend to create clusters, the number of the neighbouring particles is high. If the particles are distributed unevenly across the image, variance of the particle counts is high.

Unlike the nearest neighbour analysis, Ripley's K-function can describe characteristics of particles at several different scales [38]. However, this advantage also brings the necessity to choose a value of the radius r.

The value of r was chosen to fit (on average) 5 particles in it according to equation 5.8.

Of course, this value is different for images with a different density. Radius in very images with big particle density will be smaller than radius in images where the particles are sparse. This feature eliminates the necessity to divide the number of particles by the density λ.

Let $|I|$ be number of pixels in the image, $|T|$ number of detected particles and k the desired mean number of particles in the circle (set to be 5, as mentioned previously). Then, the radius is computed as follows:

$$r = \sqrt{\frac{k|I|}{\pi|T|}}. \tag{5.8}$$

This equation is based on the fact that there are πr^2 pixels in each circle so there are approximately $\lambda\pi r^2$ particles in the circle.

However, the radius is unknown while number of the particles in the circle k has been established before. Therefore, the expression has to be modified:

$$r^2 = \frac{k}{\lambda\pi}. \tag{5.9}$$

Replacing density λ by $|T|/|I|$ and finding the square root of the expression results in the equation 5.8.

Box plots of the Ripley's K-function have been created to allow visual comparison of the distribution of particles in different images.

The data for the box plots were computed by this algorithm:

1. Compute radius r for the image using equation 5.8.
2. Select a particle p.
3. Create a circle with the radius r around the particle p.
4. Count the particles inside the circle (excluding the particle p).
5. If there are any unprocessed particles left, proceed to 2 and select another particle.
6. Result of the algorithm is a list of particle counts, which can serve as the input of a histogram or a box plot.

Chapter 6

Implementation

This chapter describes all programs that were created to achieve the goal of this thesis.

The most important of them is the particle detection program that detects locations, areas and distribution of particles.

Another program is a plugin to image processing program ImageJ[7]. The plugin uses outputs of the detection program and makes its usage more user friendly.

Two support programs were created: one for assignment of the particles during testing and one for creating the reference data.

6.1 Particle detection program

The particle detection program contains all methods used for detection. It also contains methods for computation of the spatial analytic measures.

It was written in Java using the OpenCV framework. It can be used as a standalone program with a command line interface or as a library containing one class for each preprocessing, detection or validation method described in the chapter Design.

The only method not bundled inside the program is SRRF, which is implemented in an ImageJ plugin called NanoJ[8].

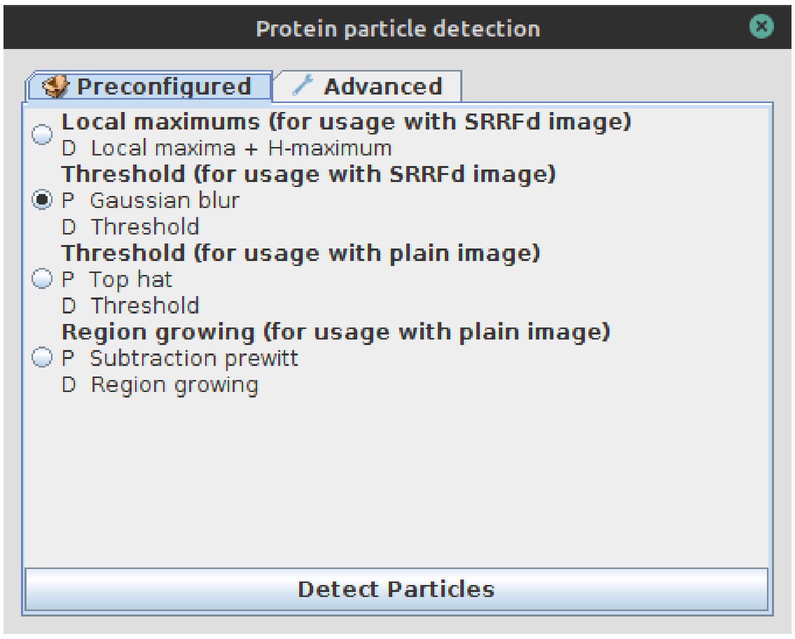
6.2 ImageJ plugin

ImageJ is an image processing program, which can be extended using plugins.

The plugin created in this thesis enhances ImageJ by a user interface for controlling the library described in the previous section. It allows the user to

[7]`https://imagej.net`

[8]`https://github.com/HenriquesLab/NanoJ-SRRF`



Protein particle detection

Preconfigured | Advanced

( ) Local maximums (for usage with SRRFd image)
D Local maxima + H-maximum

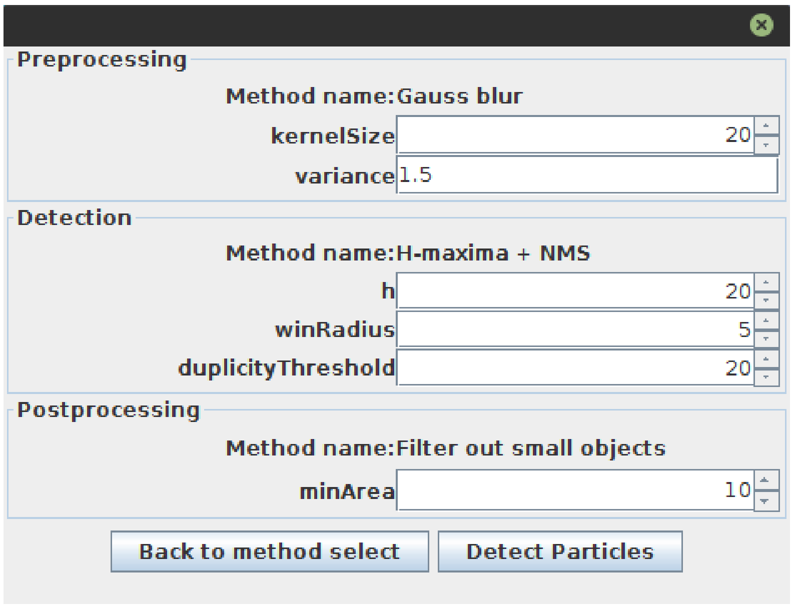
(●) Threshold (for usage with SRRFd image)
P Gaussian blur
D Threshold

( ) Threshold (for usage with plain image)
P Top hat
D Threshold

( ) Region growing (for usage with plain image)
P Subtraction prewitt
D Region growing

Detect Particles

(a) Method selection screen



Preprocessing

Method name: Gauss blur

kernelSize	20
variance	1.5

Detection

Method name: H-maxima + NMS

h	20
winRadius	5
duplicityThreshold	20

Postprocessing

Method name: Filter out small objects

minArea	10

Back to method select | Detect Particles

(b) Parameter selection screen

Figure 6.1: ImageJ plugin

choose a detection method, tweak its parameters, run the configured algorithm and visualize its result.

The plugin contains the four preconfigured combinations of methods introduced in section 5.5. It is also possible to create own combination of methods and set their parameters manually.

6.3 Evaluation program

The evaluation program pairs detected points in an image with reference points manually annotated by a human. The points are then counted to evaluate the performance of the detection algorithm.

The program is written in Java. It is controlled by a command line interface.

Design of the program is described in section 8.1. It uses Hungarian algorithm [40] to map detected points to the manually annotated reference points. All unmapped points are marked as a false positive or a false negative.

It turned out that the Hungarian algorithm was very slow and consumed a lot of memory for big data sets. It allocates matrix of size[9] $|F| \times |T|$ in the memory, which might even cause running out of heap space.

On the other hand, number of the reference particles is always reasonable (if the image isn't too big) and the only problematic part is the number of testing points. There were several algorithms that (with a bad set of parameters) annotated a very high number of pixels. To evaluate even those algorithms, a modified testing program had to be created.

A simple preprocessing algorithm runs before creation of the assignment matrix:

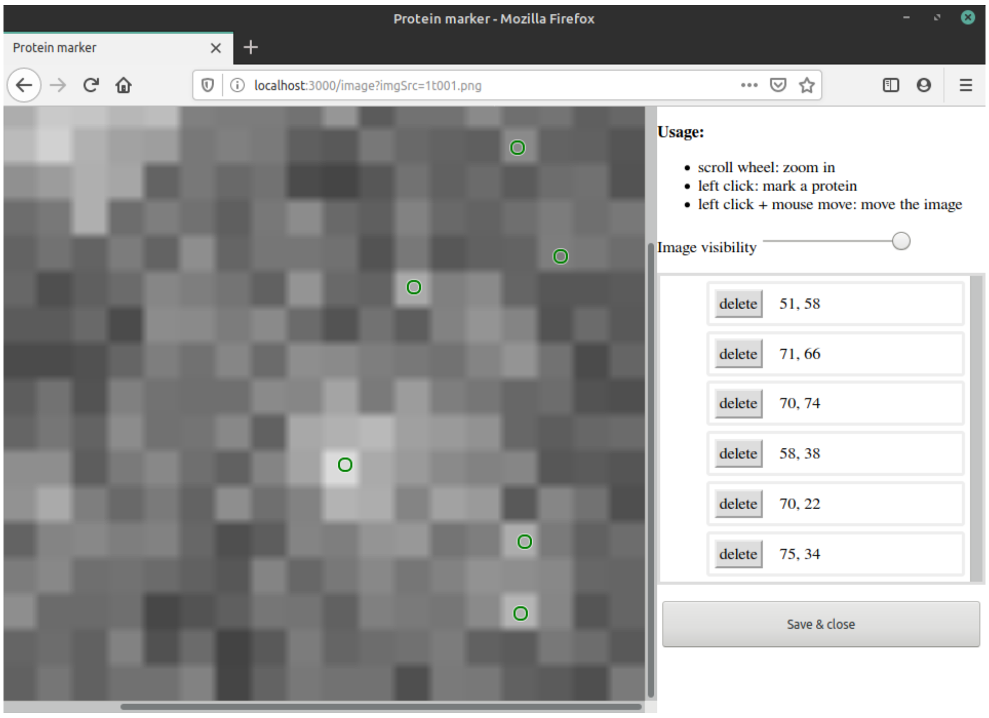
1. A testing point is deleted if there is no reference point in distance of d_{max} or closer (and vice versa).

2. A testing point T_i is mapped to a reference point F_j if their distance is lower than d_{max} and there is no another reference point in distance of d_{max} from T_i (and vice versa). This step might change the resulting mapping of the points and increase the total cost because F_j could be mapped to some closer point T_k. But it can't change number of total particles mapped – existence of the mapping between F_j and T_k would inevitably result in abandoning F_i.

All points satisfying the rule 1 are deleted, then all points are mapped according to the rule 2 and then also deleted. This process is repeated until there were no deleted or mapped points in the last step.

The simple preprocessing algorithm significantly sped up evaluation of very large testing data sets.

[9] $|F|$ = number of reference points, $|T|$ = number of testing points


Protein marker - Mozilla Firefox
Protein marker
localhost:3000/image?imgSrc=1t001.png
Save & close

Usage:

- scroll wheel: zoom in
- left click: mark a protein
- left click + mouse move: move the image

Image visibility

delete	51, 58
delete	71, 66
delete	70, 74
delete	58, 38
delete	70, 22
delete	75, 34

Figure 6.2: Particle marking program

The implementation of Hungarian algorithm was copied from Kevin Stern's Github repository [41], who published it under the MIT license.

6.4 Particle marking program

A web application was made to ease the dataset creation.

The application can display an image, allows the user to select positions of particles and save them to the server. The particles can be then re-loaded from the server again to continue marking the particles or to download the particle positions in JSON format.

The client side of the application, written in JavaScript, is responsible for visualization of the particle positions and handling selection of the particles by the user.

The server side saves the particle positions on a server in the JSON format. It was also written in JavaScript for Node.js environment. The application uses Express[10] web application framework to handle HTTP requests.

The user must enter a valid password as a URL parameter. It serves as very basic prevention against uninvited visitors.

[10]https://expressjs.com

Chapter **7**

Results

This chapter displays results of the detection and density estimation methods.

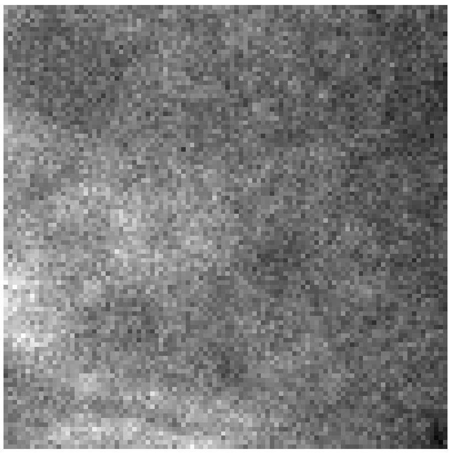
There are several input images with two types of samples. Each image contains only one type of the sample.

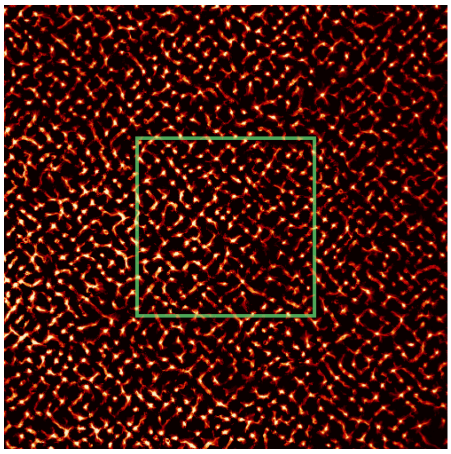
Regions of interest were selected in the input images. All the regions of interest contain only dense parts of the image and not edges of the sample.

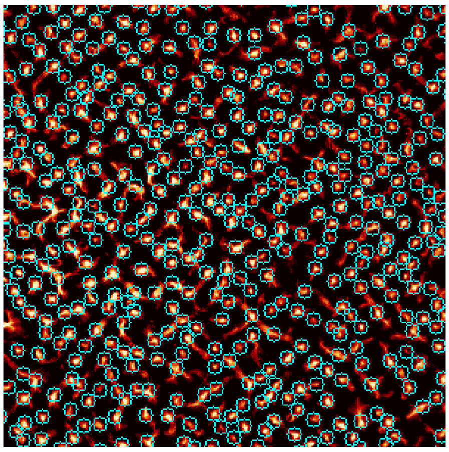
There are 10 input (ROI) images, each of them belongs to one sample type (marked A and B). Size of the images is 100×100 px or lower if the sample is not big enough.

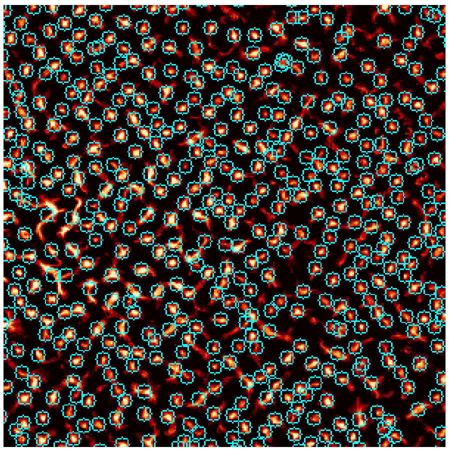
7.1 Particle detection

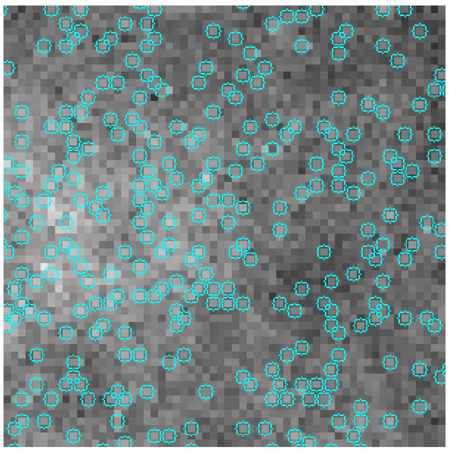
The following images contain positions of particled detected by the algorithms. The original input image (image 7.1a) was cropped to make the results more visible. The rectangle in the image 7.1b shows where the image was cropped.

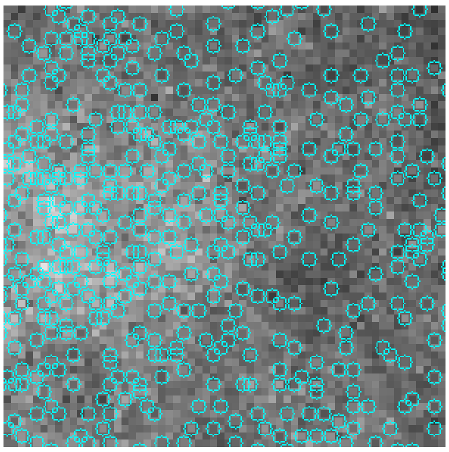

(a) Image B3, ROI 1
Original image


(b) Image B3, ROI 1
SRRF aplied


(c) Image B3, ROI 1
SRRF, Local max., H-maxima


(d) Image B3, ROI 1
SRRF, Gauss b., Threshold


(e) Image B3, ROI 1
Top hat, Threshold


(f) Image B3, ROI 1
Subtract prewitt, Region growing

7.2 Distribution analysis

This section displays an example of the distribution analysis techniques applied to the ROI images mentined above.

7.2.1 Number of particles

The following methods were applied to the particles:

1. SRRF + Local maximum + H-maxima,
2. SRRF + Gaussian blur + Threshold,
3. Top hat + Threhsold,
4. Subtract Prewitt + Region growing.

Table 7.1: Number of particles in the image

Image	Method 1	Method 2	Method 3	Method 4	Image size [px]
A1-1	1511	1466	771	1293	100×100
A2-1	1187	1247	924	699	100×100
A2-2	335	343	250	75	60×60
A7-1	1185	1244	1078	1610	100×100
A7-2	666	635	516	420	80×80
B1-1	647	684	178	569	100×100
B1-2	731	850	179	778	100×100
B3-1	1207	1260	771	1198	100×100
B3-2	735	804	434	607	80×80
B4-1	431	469	30	96	60×60

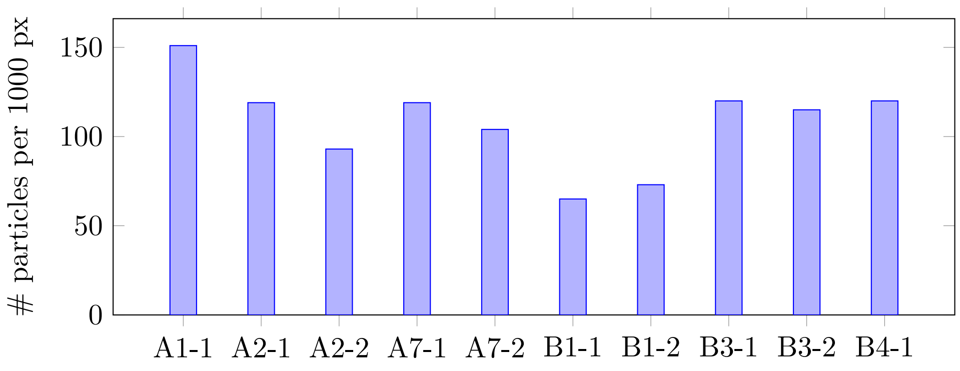
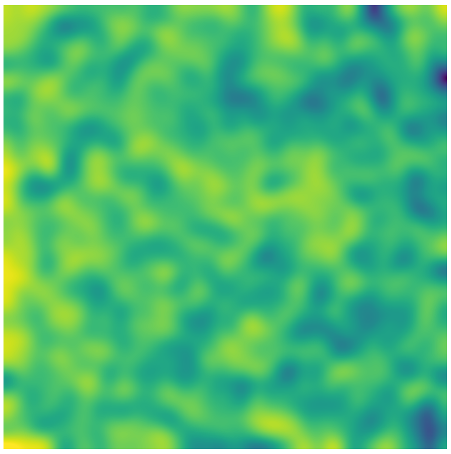
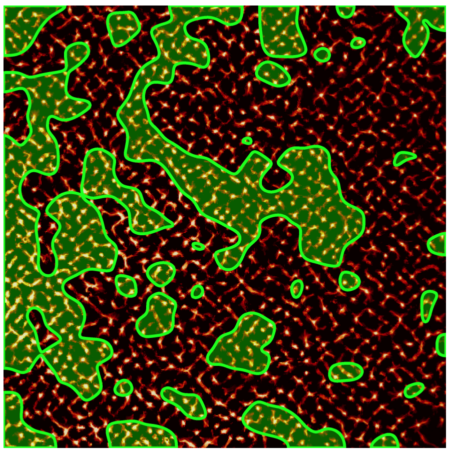


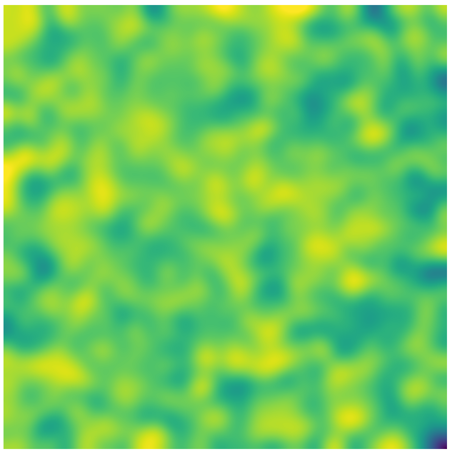
Figure 7.2: Number or particles per px in the input images, method 1

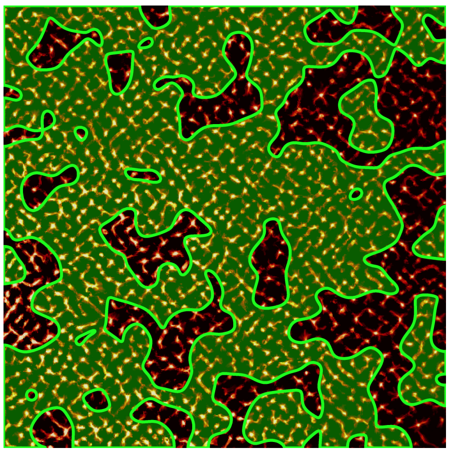
7.2.2 Particle density

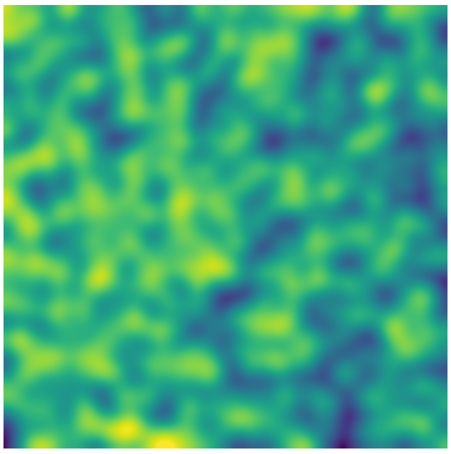
In this section, density maps (described in section 5.6.2) of the image B3-1 are presented.

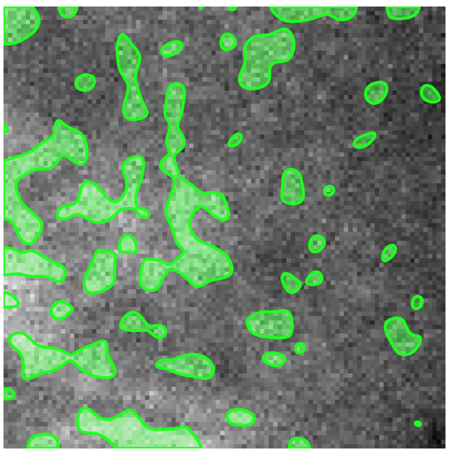

(a) Image B3, ROI 1
SRRF, Local max., H-maxima

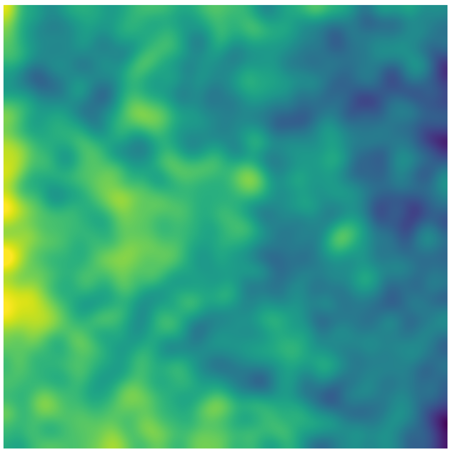

(b) Image B3, ROI 1
SRRF, Local max., H-maxima

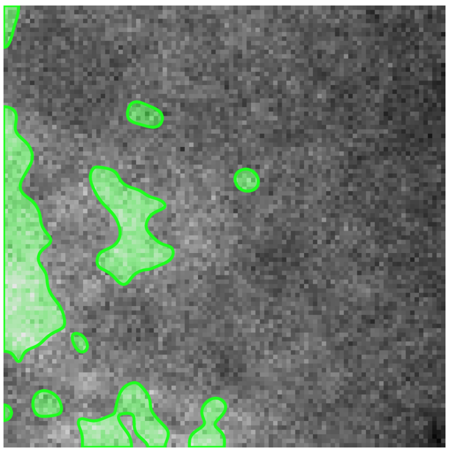

(c) Image B3, ROI 1
SRRF, Gauss b., Threshold


(d) Image B3, ROI 1
SRRF, Gauss b., Threshold


(e) Image B3, ROI 1
Top hat, Threshold

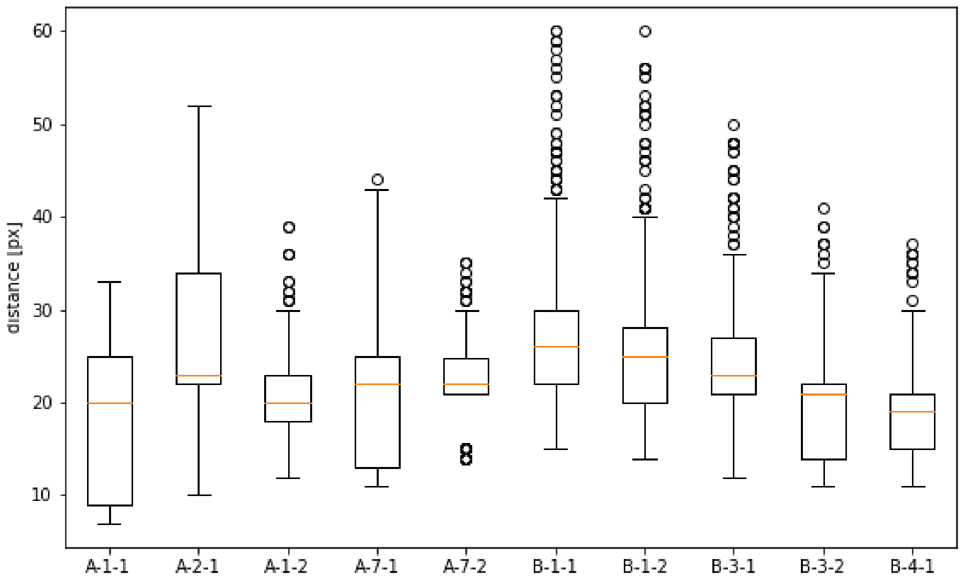

(f) Image B3, ROI 1
Top hat, Threshold


(g) Image B3, ROI 1
Subtract Prewitt, Region growing

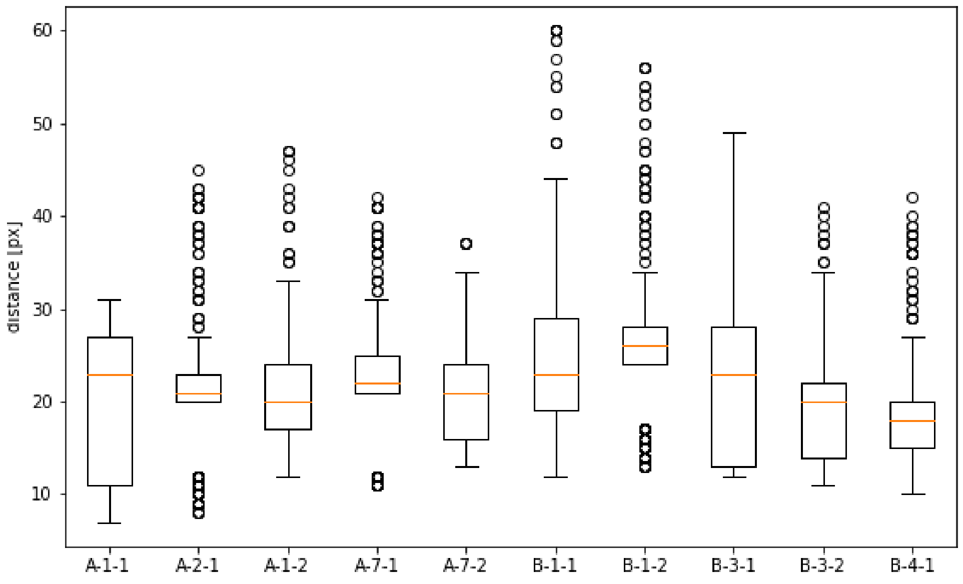

(h) Image B3, ROI 1
Subtract Prewitt, Region growing

7.2.3 Nearest neighbour analysis

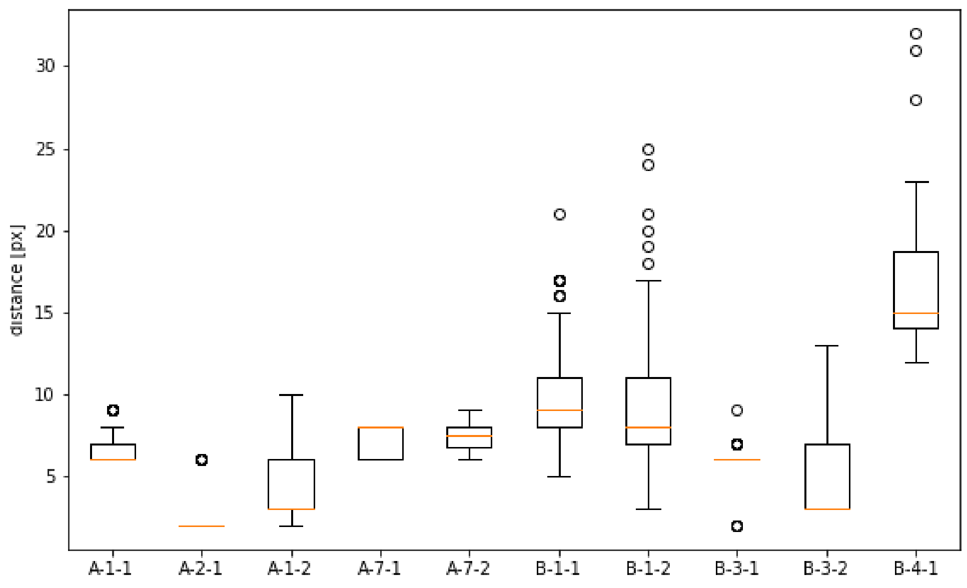
The following plots display distribution of distances to the nearest neighbour. They are described in section 5.6.3.



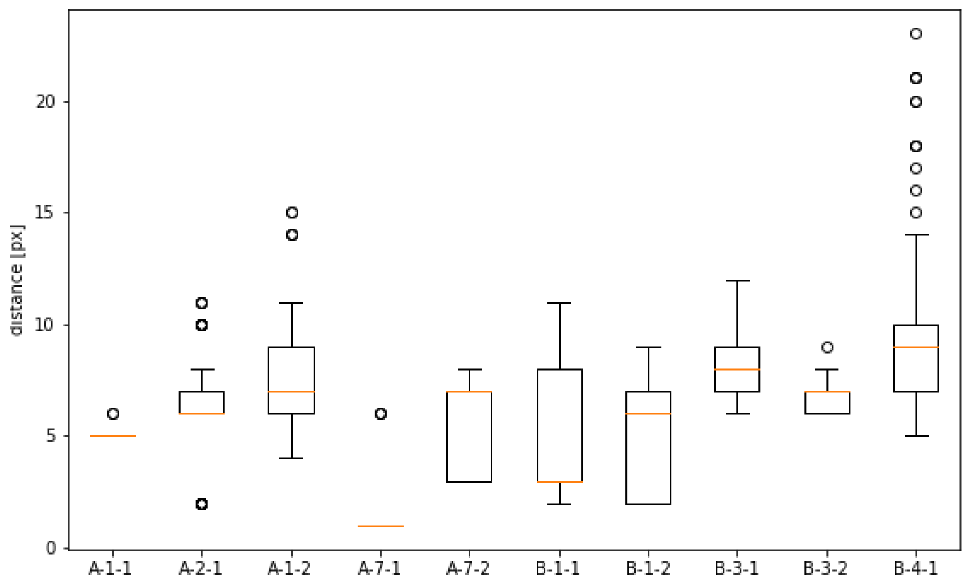
(a) SRRF, Local max., H-maxima



(b) SRRF, Gauss b., Threshold



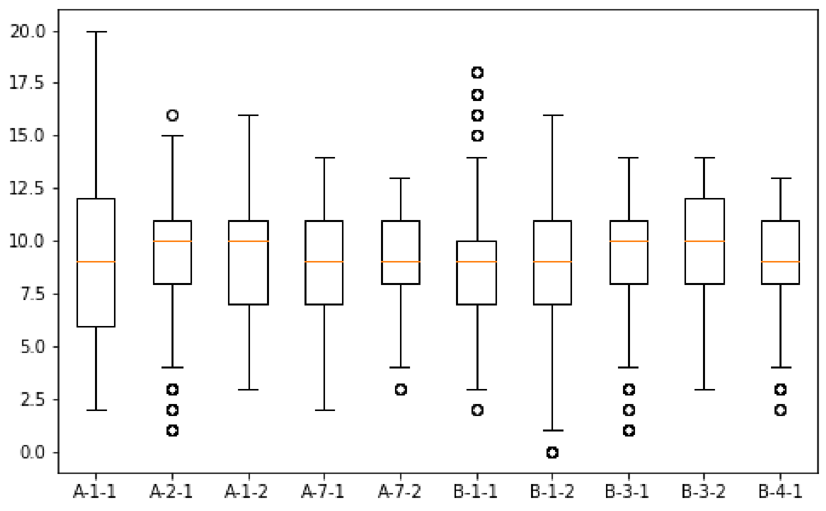
(c) SRRF, Gauss b., Threshold



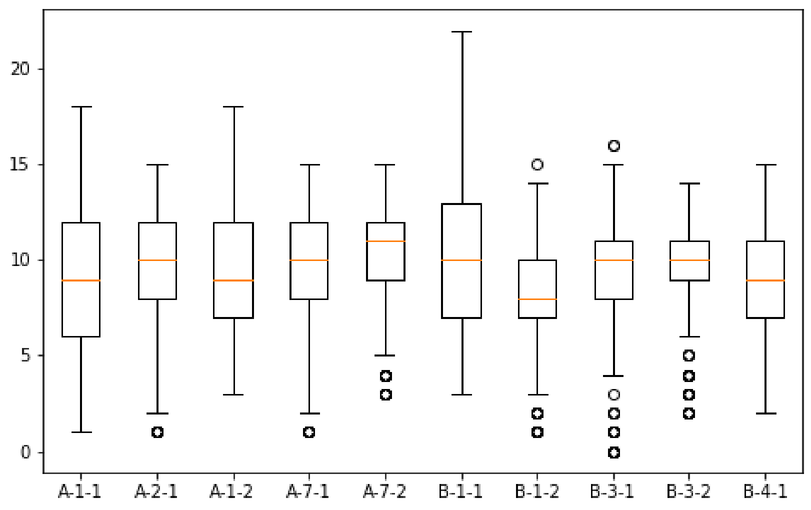
(d) SRRF, Gauss b., Threshold

7.2.4 Number of particles in the radius

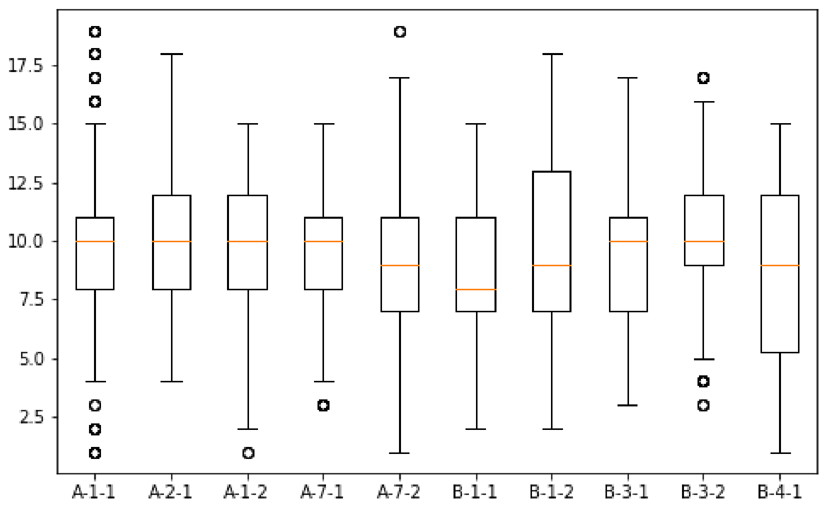
The following plots display distribution of number of the particles within a variable radius – a measure derived from Ripley's K-function. Details of this measure are described in section 5.6.4.



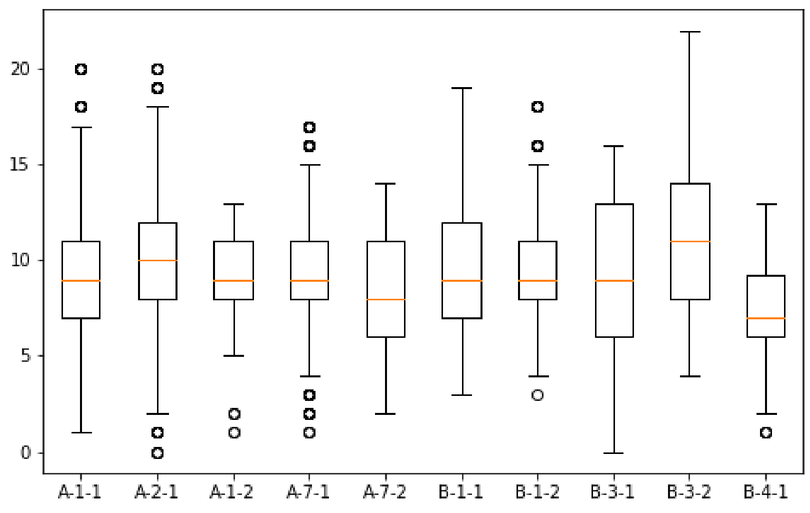
(a) SRRF, Local max., H-maxima



(b) SRRF, Gauss b., Threshold



(c) Top hat, Threshold



(d) Subtraction Prewitt, Region growing

Chapter 8

Testing

This chapter describes methods used to evaluate the single-particle detection methods.

The evaluation algorithm uses two input data sets: protein positions manually annotated by a human (called “reference”) and positions detected by the evaluated algorithm (called “testing”).

The test data are compared to the reference data.

A particle is often marked by some reference point and some testing point that are very close to each other but not exactly on the same pixel. Therefore, it is necessary to assign the particles marking the same protein to each other. The assignment algorithm is further described in the section 8.1.

Precision, recall and f-measure are computed to evaluate the performance of the particle detection methods.

Several interesting measures (like ROC curve and AUC that describe the whole method and not just its one configuration) can’t be computed because those measures need the number of true negative samples. There is an infinite number of points with no protein particle where no particle has been detected – hence the number of true negative would be infinite. It is possible to compute true negative pixels. It is a finite number but it is very high for all reasonable algorithms (so the ROC curves would look almost the same) and it changes when the image resolution changes.

8.1 Assignment of particles

The mapping algorithm pairs particles in the reference set with particles detected in the testing set.

Mapping of the particles is considered an assignment problem.

There are three possible types of assignment to make:

- a reference point to a testing point – means that the reference point and the testing point lie on the same particle
- a reference point to nothing – means that reference point was not detected
- a testing point to nothing – means that the testing point is a false positive

Each of the assignments described above has defined "cost" which denotes how unlikely is the occurrence to happen.

The assignment problem is defined as finding such assignment function f_A, for which the expression:

$$\sum_{m=1}^{n} \sum_{n=1}^{n} f_A(i,j)c(i,j) \tag{8.1}$$

is as minimal as possible [42]. Function $c(i,j)$ in the equation 8.1 is the cost of the assignment of i to j.

Values of the function f_A can be 1 (the objects i and j are assigned to each other) or 0 (the objects are not assigned). Exactly one i can be assigned to one j.

In the case of this thesis, the cost c is defined as:

- for assignment of two points: square distance of those two points,
- for assignment of point to noting (and vice versa): constant of 50 px. Particles with the distance greater than 50 px could be never assigned to each other. The constant is further noted as d_{max}.

There are two possible ways to understand the assignment problem – as a pairing in a bipartite graph or using a matrix.

8.1.1 Matching in a bipartite graph

Assignment problem can be understood as finding a matching[11] in a bipartite graph[12].

Nodes in the graph would represent reference and testing points. Edges between such two nodes would represent assigning the two points to each other. Only points representing points from different sets could be connected,

[11] Set of edges with no common vertex.

[12] Graph whose nodes form two disjoint sets. The graph may contain edges between nodes from different sets but there can be no edges between nodes from the same set.

therefore the graph is bipartite (having two sets of nodes where there could be no edge between nodes in the same set).

There is another restriction of this graph: each node could only have one edge (each point could be assigned to only one point).

The goal of the task is to find such a graph with maximal possible number of edges.

Though this algorithm might be much simpler, it can't utilize distance between two points. Two points very close to each other are equally likely to be assigned to each other than two points with the distance just slightly below d_max. The matrix solution was chosen for this reason.

8.1.2 Assignment matrix

Assignment matrix A is a square matrix, in which each element $A_{i,j}$ is equal to the cost function $c(i,j)$ [42].

The following text describes creation of such a matrix.

Let F_i be the i-th reference point, T_i be the i-th testing point, k be number of the reference points, l be number of the testing point and $d(a,b)$ be the (square Euclidean) distance between a and b.

The assignments matrix is defined as follows:

$$
\begin{array}{c|cccccc|}
 & T_1 & \dots & T_l & X_1 & \dots & X_{n-l} \\
F_1 & d(F_1,T_1) & \dots & d(F_1,T_l) & d_{max} & d_{max} & d_{max} \\
\vdots & \vdots & \ddots & \vdots & d_{max} & d_{max} & d_{max} \\
F_k & d(F_k,T_1) & \dots & d(F_k,T_l) & d_{max} & d_{max} & d_{max} \\
X_1 & d_{max} & d_{max} & d_{max} & d_{max} & d_{max} & d_{max} \\
\vdots & d_{max} & d_{max} & d_{max} & d_{max} & d_{max} & d_{max} \\
X_{n-k} & d_{max} & d_{max} & d_{max} & d_{max} & d_{max} & d_{max}
\end{array}
$$

This matrix serves as an input of Hungarian algorithm [40], which creates the assignment function.

Assigning a column to a row in the matrix is interpreted this way:

- Assignment between F_i and T_j – true positive – The testing point T_j is assigned to the reference point F_i.
- Assignment between X_i and T_j – false positive – The testing point T_j has no counterpart in the reference set.
- Assignment between F_i and X_j – false negative – The reference point F_i has no counterpart in the testing set.
- Assignment between X_i and X_j – no meaning (necessary for technical reasons).

8.2 Observed measures

Simple measures as true positive, false positive and false negative defined in the section 8.1.2 are not sufficient to be used to compare detection methods. Therefore, derived measures have to be computed.

Precision, recall and F-measure were selected.

Precision (defined in the equation 8.2 [43]) captures algorithm's ability to avoid false detections.

$$\text{precision} = \frac{|\text{true positive}|}{|\text{true positive}| + |\text{false positive}|} \tag{8.2}$$

Recall (defined in the equation 8.3 [43]) captures algorithm's ability to detect particles.

$$\text{recall} = \frac{|\text{true positive}|}{|\text{true positive}| + |\text{false negative}|} \tag{8.3}$$

F-measure combines both those measures into one number [43]:

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{8.4}$$

F-measure captures not only algorithm's precision or recall but also their balance [43]. If one of those measures is very high and the second one very low, F-measure will be low.

8.3 Results

The following section contains the result of the tested methods. Measures described in the previous section were used.

The tables below contain selected combinations of algorithms. The combinations were selected manually – preprocessing or validation phases were added to the existing set of algorithms to address some of their weakness.

Different parameters of the methods were tried to obtain the best possible combination of the parameters. Then the algorithm's performance was evaluated on the testing data.

8.3.1 Local maximum

First approach (4-neighbourhood):

Preprocessing/postprocessing method	Precision	Recall	F-measure
No preprocessing/postprocessing	38.5 %	53.9 %	0.450
H-maximum	80.4 %	79.2 %	0.798
Bilateral filter + H-maximum	75.0 %	83.2 %	0.789
Laplace operator + H-maximum	84.2 %	32.1 %	0.465
SRRF	71.3 %	76.2 %	0.737
SRRF + H-maximum	91.0 %	82.0 %	0.862
SRRF + Gaussian filter + H-maximum	91.5 %	82.2 %	0.866
SRRF + Bilateral filter + H-maximum	91.2 %	82.0 %	0.863
SRRF + Wiener filter + H-maximum	99.7 %	29.8 %	0.459

Second approach (sliding window):

Preprocessing/postprocessing method	Precision	Recall	F-measure
No preprocessing/postprocessing	63.5 %	65.5 %	0.645
H-maximum	73.5 %	80.9 %	0.771
SRRF	72.0 %	76.0 %	0.739

The sliding window local maximum algorithm is more complex than the 4-neighbourhood one but it didn't bring almost any advantage in the performance. Therefore, this algorithm was abandoned and all preprocessing methods were tested only for the first one.

8.3.2 Region growing

Preprocessing/postprocessing method	Precision	Recall	F-measure
No preprocessing/postprocessing	65.2 %	73.1 %	0.690
Gaussian blur	64.6 %	72.8 %	0.622
Prewitt subtraction	69.9 %	73.2 %	0.715
Prewitt subtraction + Gaussian filter	75.3 %	60.9 %	0.673
Sobel subtraction	62.5 %	74.7 %	0.681
SRRF	53.9 %	58.8 %	0.562
SRRF + Prewitt subtraction	96.9 %	58.2 %	0.727
SRRF + Gaussian filter + Prewitt subtraction	99.4 %	34.3 %	0.510

8.3.3 Thresholding

Preprocessing/postprocessing method	Precision	Recall	F-measure
No preprocessing/postprocessing	52.5 %	65.0 %	0.580
Top hat	68.6 %	77.6 %	0.729
Laplacian filter + Top hat	55.6 %	78.5 %	0.651
Flatfield correction (subtract)	76.3 %	45.5 %	0.570
Flatfield correction (image normalization)	79.3 %	50.3 %	0.616
Prewitt subtraction	59.2 %	82.3 %	0.689
Flatfield (normalisation) + Prewitt subtraction	59.7 %	79.1 %	0.680
SRRF	80.3 %	79.1 %	0.797
Gaussian filter	90.3 %	80.2 %	0.850
Bilateral filter	78.9 %	80.3 %	0.796
SRRF + Prewitt	87.9 %	80.1 %	0.838
SRRF + Prewitt subtraction	80.7 %	80.5 %	0.806
SRRF + Gaussian filter + Prewitt	89.9 %	78.3 %	0.837
SRRF + Bilateral filter + Prewitt	87.6 %	79.7 %	0.834
SRRF + Top hat	80.2 %	79.1 %	0.797
SRRF + Gaussian filter + Top hat	90.4 %	72.3 %	0.804
SRRF + Flatfield correction (subtract)	78.7 %	80.3 %	0.795
SRRF + Flatfield correction (image normalization)	79.5 %	79.0 %	0.794

Thresholding with the Otsu method was also tested:

Preprocessing/postprocessing method	Precision	Recall	F-measure
No preprocessing/postprocessing	69.6 %	27.4 %	0.393
SRRF	94.8 %	48.0 %	0.637

Chapter 9

Discussion

This section evaluates the previous work and outlines possible future work on the task.

The design part describes several methods, from which many of them proved not very useful for the task. However, combination of SRRF, local maximum detector and h-maximum proved to be sufficient for particle detection.

The four most promising algorithms were selected to create input set of particles for analysis – the best combination was one of them. The reason for selecting more methods than just one was possibility of their comparison.

The analysis aimed to find some differences in spatial distribution of the particles among two different kinds of images. Four analytic methods were chosen to fulfill this task.

The differences in the spatial distribution weren't found. It is possible that the wrong measures were chosen to evaluate the spatial distribution. Another conceivable explanation is that the spatial distributions of both kinds of images don't differ.

When comes to method detection, there is still plenty of unused methods.

The protein particles have approximately circular shape. This feature could be used to detect them – for example by the combination of the first derivation (eg. by already implemented Prewitt or Sobel operator) and Hough transform detecting circular shapes. Such detector would have to deal with several particles that have blend in.

Another not used algorithm is mean shift. It could find local maximums in the image, which could be used for localizing particle centers. This method might be useful for detection of particles on a preprocessed image using seeds that were spread across the image. However, it would be better to use the algorithm in the validation phase to merge double detection of one particle (so the previous detection method is used to obtain the seeds).

Watershed algorithm might be also used in the validation phase. It might help to segment the image once particle positions were obtained. Some algorithms have problems when segmenting the image – for example thresholding: when the value of the particle pixels is very low, the algorithm detects its peak but the bottom part of the particle is considered background. Local maximum doesn't even try to segment the image.

The segmented image created by watershed or another algorithm might serve as the input of another validation algorithms (like already existing particle size). Segmented areas merged, split or filtered out.

All the used methods are prone to some specific failure the other methods are resistant to. In machine learning, this issue can be addressed by an ensemble of the methods.

Creating an ensemble of localisation algorithms is a more difficult task than creating an ensemble of classifiers. Two algorithms might find the same particle on slightly different locations.

One possible solution might be assigning the particles detected by various algorithms to each other by the algorithm described in the Testing chapter, section 8.1.

There is a space to improvement in the implementation too. When installing the ImageJ plugin, the user has to install OpenCV, add its location to the Java library path and manually move the plugin and detection library to the plugins directory. The OpenCV could be distributed together with the library. The most convenient way for the user is to use a plugin update site[13].

The particle marking program (section 6.4) was created as a temporary solution. Therefore, it lacks a lot of features that would make its use more pleasant. It can't move the particle on the image or delete it there – the only way to delete a particle is to delete it on a list (where it is hard to find). If someone would like to use the program, those features are recommended to be added.

The problem with not enough data might be solved using Generative adversarial network (GAN). Many authors solving similar issues mentioned that they use a generated datasets to tests their algorithms. Protein particles, represented by Gaussian functions (their PSF) are scattered around the image. Then a noise or a background illumination is added. This approach provides input images with already known particle locations but it may significantly differ from the real data. GAN might generate the images using parameters of the Gaussian function, parameters of noise and particle density. On the other hand, training GAN for this purpose would be a challenging task with an uncertain result.

[13]`https://imagej.net/How_to_set_up_and_populate_an_update_site`

Conclusion

This thesis focuses on detection of proteins in an image from a microscope and analysis of their distribution.

Existing methods focusing on microscope image processing were explored, primarily those focusing on protein detection. The protein particle detection task is similar to detection of stars on astronomical images. Research in this field has been also performed.

Methods discovered during the research part have been further described in the thesis. They have been implemented in Java using the OpenCV library.

The implemented methods are available as a plugin to popular image processing program ImageJ. The plugin contains a graphical user interface to make the methods accessible even to non-programmers.

The image processing methods have been combined together resulting in several particle detection algorithms.

Particles detected by these algorithms have been compared to a human-annotated data. A labelling program was created to make the annotation more convenient.

To evaluate a particle detection algorithm, it is necessary to know which protein has been detected, which ones have been missed and which detections are superfluous. Therefore, a method of linking the detected points to the manually annotated data had to be created.

The most suitable algorithm for particle detection has been chosen according to its f-measure in the test. The chosen algorithm consists of SRRF upsampling, local maximum detection and “h-maximum” validation.

The data were evaluated using four spatial analysis techniques.

The task of this thesis has been accomplished but there is still a wide field of unused approaches that might improve both single particle detection and analysis. Possible enhancement has been outlined together with the discussion of the results.

Bibliography

1. ANDERSON, C.M. et al. Tracking of cell surface receptors by fluorescence digital imaging microscopy using a charge-coupled device camera. Low-density lipoprotein and influenza virus receptor mobility at 4 degrees C. *Journal of Cell Science*. 1992, vol. 101, no. 2, pp. 415–425. ISSN 0021-9533. Available from eprint: `https://jcs.biologists.org/content/101/2/415.full.pdf`.
2. MASHANOV, G; MOLLOY, J. Automatic Detection of Single Fluorophores in Live Cells. *Biophysical journal*. 2007, vol. 92, pp. 2199–211. Available from DOI: `10.1529/biophysj.106.081117`.
3. ANDERSSON, S. B. Precise localization of fluorescent probes without numerical fitting. In: *2007 4th IEEE International Symposium on Biomedical Imaging: From Nano to Macro*. 2007, pp. 252–255. Available from DOI: `10.1109/ISBI.2007.356836`.
4. PARTHASARATHY, Raghuveer. Rapid, accurate particle tracking by calculation of radial symmetry centers. *Nature methods*. 2012, vol. 9, pp. 724–6. Available from DOI: `10.1038/nmeth.2071`.
5. YOSHIDA, Seiichi. Star detection from an image. *MISAO project*. 1997. Available also from: `http://www.aerith.net/misao/report/seiichi/master/970917-english/node3.html`.
6. HROCH, Filip. The robust detection of stars on CCD images. *Experimental Astronomy*. 1999, vol. 9, no. 4, pp. 251–259. ISSN 1572-9508. Available from DOI: `10.1023/A:1008195518637`.
7. ZHENG, Caixia et al. An improved method for object detection in astronomical images. *Monthly Notices of the Royal Astronomical Society*. 2015, vol. 451, no. 4, pp. 4445–4459. Available from DOI: `10.1093/mnras/stv1237`.

8. SCHÖFER, Christian et al. Mapping of cellular compartments based on ultrastructural immunogold labeling. *Journal of structural biology*. 2004, vol. 147, pp. 128–35. Available from DOI: 10.1016/j.jsb.2004.01.014.
9. GLASBEY, C. A.; ROBERTS, I. M. Statistical analysis of the distribution of gold particles over antigen sites after immunogold labelling. *Journal of Microscopy*. 1997, vol. 186, no. 3, pp. 258–262. Available from DOI: 10.1046/j.1365-2818.1997.2050767.x.
10. SPRING, Kenneth R.; DAVIDSON, Michael W. *Introduction to Fluorescence Microscopy* [online]. Nikon MicroscopyU, [n.d.] [visited on 2019-08-29]. Available from: https://www.microscopyu.com/techniques/fluorescence/introduction-to-fluorescence-microscopy.
11. WU, Qiang et al. *Microscope Image Processing*. 1st. Orlando, FL, USA: Academic Press, Inc., 2008. ISBN 012372578X, ISBN 9780123725783.
12. WEISENBURGER, Siegfried; SANDOGHDAR, Vahid. Light microscopy: an ongoing contemporary revolution. *Contemporary Physics*. 2015, vol. 56, no. 2, pp. 123–143. Available from DOI: 10.1080/00107514.2015.1026557.
13. CULLEY, Siân et al. SRRF: Universal Live-cell Super-Resolution Microscopy. *The International Journal of Biochemistry & Cell Biology*. 2018, vol. 101. Available from DOI: 10.1016/j.biocel.2018.05.014.
14. GUSTAFSSON, Mats G. L. Nonlinear structured-illumination microscopy: Wide-field fluorescence imaging with theoretically unlimited resolution. *Proceedings of the National Academy of Sciences*. 2005, vol. 102, no. 37, pp. 13081–13086. ISSN 0027-8424. Available from DOI: 10.1073/pnas.0406877102.
15. ROTTENFUSSER, Rudi et al. *The Point Spread Function* [online]. Zeiss.com, [n.d.] [visited on 2020-05-19]. Available from: https://www.zeiss.com/microscopy/int/solutions/reference/basic-microscopy/the-point-spread-function.html.
16. FISHER, Robert et al. *Digital Filters*. The University of Edinburgh, 2003. Available also from: https://homepages.inf.ed.ac.uk/rbf/HIPR2/filtops.htm. [Online; cited 07-May-2020].
17. LEVENTHAL, Daniel. *Image Processing* [university lecture]. University of Washington, 2011. Available also from: https://courses.cs.washington.edu/courses/cse457/11au/lectures/image-processing.pdf. [Online; cited 07-May-2020].
18. BURGER, Wilhelm; BURGE, Mark J. Digital Image Processing - An Algorithmic Introduction using Java. In: Springer, 2008, pp. 120–123. Texts in Computer Science. ISBN 978-1-84628-968-2.

19. FISHER, Robert et al. *Feature Detectors – Sobel Edge Detector*. The University of Edinburgh, 2003. Available also from: http://homepages.inf.ed.ac.uk/rbf/HIPR2/sobel.htm. [Online; cited 05-May-2020].

20. KATSAGGELOS, Aggelos. *Image Recovery – part II. Fundamentals of Digital Image and Video Processing* [online course]. Northwestern University, [n.d.].

21. DELMAS, Patrice. *Gaussian Filtering* [unversity lecture]. The University of Auckland, 2010. Available also from: https://www.cs.auckland.ac.nz/courses/compsci373s1c/PatricesLectures/Gaussian%20Filtering_1up.pdf.

22. FISHER, Robert et al. *Gaussian Smoothing*. The University of Edinburgh, 2003. Available also from: https://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm. [Online; cited 09-May-2020].

23. TOMASI, C.; MANDUCHI, R. Bilateral filtering for gray and color images. In: *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*. 1998, pp. 839–846. Available also from: https://www.cs.jhu.edu/~misha/ReadingSeminar/Papers/Tomasi98.pdf.

24. SÝKORA, Daniel. *Non-Linear Filtering* [university lecture]. Czech Technical University in Prague, 2018. Available also from: https://courses.fit.cvut.cz/MI-DZO/media/lectures/05/dzo-l05.pdf.

25. ANJNA, Er. Anjna; KAUR, Er.Rajandeep. Review of Image Segmentation Technique. In: *International Journal of Advanced Research in Computer Science*. 2017, vol. 8.

26. KASK, Peet et al. Flat field correction for high-throughput imaging of fluorescent samples. *Journal of Microscopy*. 2016, vol. 263, no. 3, pp. 328–340. Available from DOI: 10.1111/jmi.12404.

27. RAY, Sidney. Applied Photographic Optics. In: 2nd ed. Woburn: Focal press, 2002, chap. 14, pp. 120–123. ISBN 0-240-51540-4.

28. BEBIS, George. *Region Growing* [university lecture]. University of Nevada, 2004. Available also from: https://www.cse.unr.edu/~bebis/CS791E/Notes/RegionGrowing.pdf.

29. GUSTAFSSON, Nils et al. Fast live-cell conventional fluorophore nanoscopy with ImageJ through super-resolution radial fluctuations. *Nature Communications*. 2016, vol. 7. ISSN 2041-1723. Available from DOI: 10.1038/ncomms12471.

30. HENRIQUES, Ricardo. *High-speed super-resolution data analysis in ImageJ: from QuickPALM to NanoJ* [presentation]. Single Molecule Localization Microscopy Symposium, 2016. Available also from: https://www.youtube.com/watch?v=HjrcM8NfWJE.

31. OWENS, Robyn. *Mathematical Morphology* [university lecture]. University of Edinburgh, 1997. Available also from: http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OWENS/LECT3/node3.html.

32. HLAVÁČ, Václav. *Matematická morfologie* [university lecture]. Czech Technical University in Prague, [n.d.]. Available also from: http://people.ciirc.cvut.cz/~hlavac/TeachPresCz/11DigZprObr/71-3MatMorpholBinCz.pdf.

33. SERRA, Jean. *Opening, Closing* [unversity lecture]. Ecole des Mines de Paris, 2000. Available also from: http://www.cmm.mines-paristech.fr/~serra/cours/pdf/en/ch3en.pdf.

34. TCHESLAVSKI, Gleb. *Morphological Image Processing: Gray-scale morphology* [online]. Lamar University, 2010 [visited on 2020-04-29]. Available from: http://liulong.site/opencv/doc/Gray%20Morphological%20Image%20Processing.pdf.

35. CLOUARD, Régis. *Tutorial: Mathematical Morphology.* University of Caen, 2012. Available also from: https://clouard.users.greyc.fr/Pantheon/experiments/morphology/index-en.html.

36. WUDI, Petr. *Detekce a sledování pohybu osob na základě záznamu z kamerového systému.* 2017. Bachelor's Thesis. Czech Technical University in Prague, Faculty of Information Technology.

37. CLARKE, Keith. *Applications of Feature Measurements* [university lecture]. University of California, Santa Barbara, 2011. Available also from: http://www.geog.ucsb.edu/~kclarke/Geography12/Lecture19.pdf.

38. DIXON, Philip M. Ripley's K function. In: Chichester: John Wiley & Sons, Ltd, 2002, vol. 3, pp. 1796–1803. Encyclopedia of Environmetrics. An optional note.

39. ECKEL, Stefanie Martina. *Statistical Analysis of Spatial Point Patterns.* Ulm, 2008. PhD thesis. Universität Ulm.

40. KUHN, Harold. The Hungarian Method for the Assignment Problem. *Naval Research Logistic Quarterly.* 1955, vol. 2, pp. 83–97.

41. STERN, Kevin. *HungarianAlgorithm* [online]. [N.d.]. Available also from: https://github.com/KevinStern/software-and-algorithms/blob/master/src/main/java/blogspot/software_and_algorithms/stern_library/optimization/HungarianAlgorithm.java.

42. BASIRZADEH, Hadi. Ones assignment method for solving assignment problems. *Applied Mathematical Sciences (Ruse).* 2012, vol. 6.

43. BROWNLEE, Jason. *How to Calculate Precision, Recall, and F-Measure for Imbalanced Classification* [online]. Machine Learning Mastery, 2020 [visited on 2020-05-22]. Available from: https://machinelearningmastery.com/precision-recall-and-f-measure-for-imbalanced-classification/.

Appendix A

Notation

The following list contains meaning of variables and operations used in this theis if they are not explicitly defined differently in the scope.

I	Input image – either a function or a matrix
x, y	Image coordinates
p	Protein particle
τ	A threshold
h	A threshold, usually refers to height of some object
d	Distance (Eucleidian) between two points in the image
$a * b$	Convolution of a and b
f	Original function – in a filter
g	Filtered function – in a filter
h	Filter kernel function – in a filter
F, G, H	Frequency equivalents of the previous
G	Gaussian function (collides with the previous but it is commonly used)
$\hat{f}$	Estimate of the function f
S	Shading function
σ	Standard deviation
F	Set of the manually annotated reference points
T	Set of “testing” points detected by the algorithm
$c(\cdot, \cdot)$	Cost function

Appendix B

Contents of enclosed CD

```
readme.txt ........................ the file with CD contents description
jar ................................. the directory with Java packages
src ...................................... the directory of source codes
    core ..................... detector and analysis library source sources
    imagej-plugin ........................ ImageJ plugin source sources
    evaluation ..................... evaluation program source sources
    marker ................................ particle marker source sources
    thesis ............... the directory of LaTeX source code of the thesis
thesis.pdf ............................. the thesis text in PDF format
dependencies .......................... libraries used by the programs
    opencv-build ................................ OpenCV library build
    opencv-4.0.0 ........................ OpenCV library source code
```

Appendix C

ImageJ plugin user manual

The plugin enhances ImageJ image processing program. It is recommended to use the Fiji distribution available at `https://imagej.net/Fiji/Downloads`.

C.1 Installation

The plugin depends on the OpenCV library to work. The library must be delivered to ImageJ/Fiji by the user.

There are two ways to obtain OpenCV:

- Use the precompiled files available at the disc attached to this thesis. This is a more user-friendly way but there is no guarantee the program will work on each combination of software and hardware. The files were compiled for Windows and Linux operation systems for AMD64 architecture.
- Use the OpenCV source code available on the attached disc or download Opencv 4.0.0 from `https://opencv.org/opencv-4-0/`. Compile the source code or install OpenCV on your system using the installation program. The necessary files are available in the installation at `java/opencv-400` and `java/x64/opencv_java400.(so|dll)`.

Another files necessary for installation are the plugin itself and the detection library. They are located in the `jar` directory.

To install the program, copy (relative to the ImageJ root directory):

- `protein-detection-1.0-SNAPSHOT.jar` to `lib`
- `protein-detection-imagej-1.0-SNAPSHOT.jar` to `plugins`
- `opencv-400.jar` to `jars`

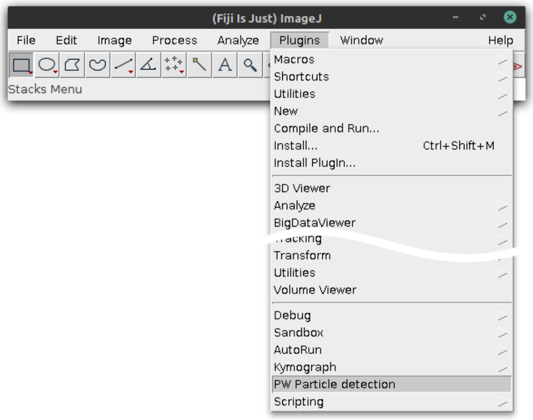
- `opencv_java400.dll` (Windows) or `opencv_java400.so` (Mac, Linux) to `lib`

C.2 Usage

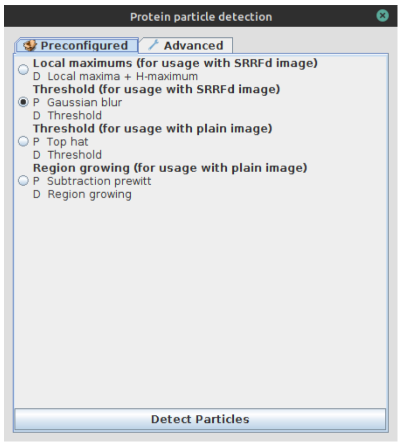
To detect locations of particles on an image, follow those steps:

1. start ImageJ/Fiji,
2. choose the image to be processed using File → Open,
3. click Plugins button on the menu bar,
4. click PW Particle detection in the menu (Figure C.1a),
5. a window should appear (Figure C.1b),
6. select detection, preprocessing and validation method,
7. click Proceed to parameter setting,
8. select parameters of methods selected on during previous steps,
9. click Detect Particles,
10. processing of the image takes mostly up to few seconds to regular images,
11. an image with detected protein particles marked by points appears (Figure C.1e).

Manual configuration of the methods is also available. Its configuration appears after clicking on the ”Advanced” tab on top of the window. The manual configuration consists of two screens displayed on figures C.1c and C.1d.

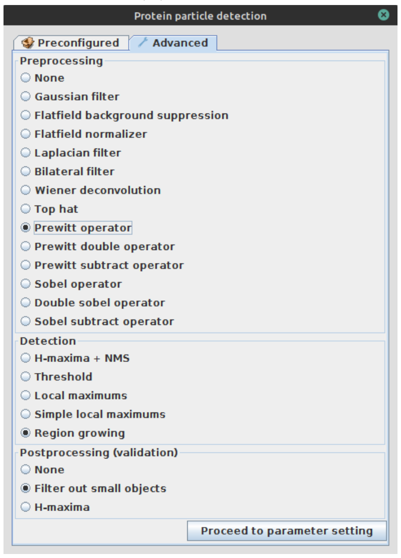


(a) Menu item selection



- ( ) **Local maximums (for usage with SRRFd image)**
  D Local maxima + H-maximum
- (●) **Threshold (for usage with SRRFd image)**
  P Gaussian blur
  D Threshold
- ( ) **Threshold (for usage with plain image)**
  P Top hat
  D Threshold
- ( ) **Region growing (for usage with plain image)**
  P Subtraction prewitt
  D Region growing

(b) Preconfigured method selection

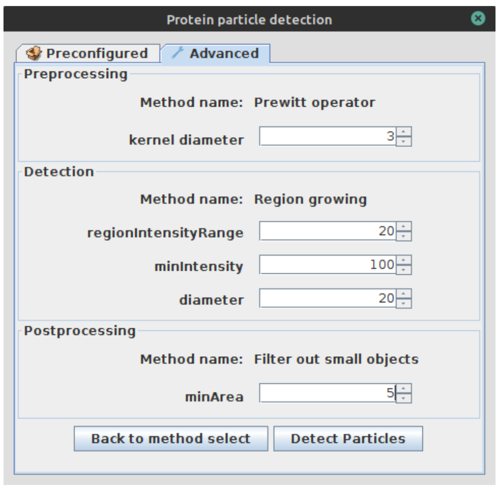


Preprocessing
- ( ) None
- ( ) Gaussian filter
- ( ) Flatfield background suppression
- ( ) Flatfield normalizer
- ( ) Laplacian filter
- ( ) Bilateral filter
- ( ) Wiener deconvolution
- ( ) Top hat
- (●) Prewitt operator
- ( ) Prewitt double operator
- ( ) Prewitt subtract operator
- ( ) Sobel operator
- ( ) Double sobel operator
- ( ) Sobel subtract operator

Detection
- ( ) H-maxima + NMS
- ( ) Threshold
- ( ) Local maximums
- ( ) Simple local maximums
- (●) Region growing

Postprocessing (validation)
- ( ) None
- (●) Filter out small objects
- ( ) H-maxima

(c) Manual method selection

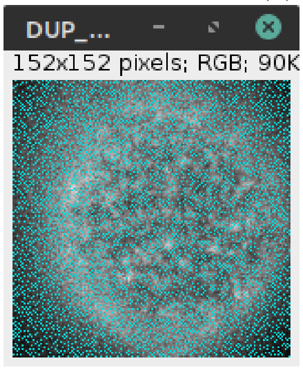


Preprocessing
Method name: Prewitt operator
kernel diameter 3

Detection
Method name: Region growing
regionIntensityRange 20
minIntensity 100
diameter 20

Postprocessing
Method name: Filter out small objects
minArea 5

(d) Parameter selection



(e) The output.

Figure C.1: Usage of the ImageJ plugin