CZECH TECHNICAL UNIVERSITY IN PRAGUE

FACULTY OF BIOMEDICAL ENGINEERING
Department of Biomedical Technology

Automatic Determination of Size and Shape of Nanoparticles for Biomedical Applications

Monika Pulcová

Study program: Biomedicínská technika
Bachelor thesis supervisor: doc. Ing. Vladimíra Petráková, PhD.
Consultant: Ing. Miroslav Hekrdla, PhD. (ÚFCHJH AV ČR, v.v.i.)

Bachelor thesis

Kladno 2023

BACHELOR‘S THESIS ASSIGNMENT

I. PERSONAL AND STUDY DETAILS

Student's name:	**Pulcová Monika**	Personal ID number:	**499901**
Faculty:	**Faculty of Biomedical Engineering**		
Department:	**Department of Biomedical Technology**		
Study program:	**Biomedicínská technika**		

II. BACHELOR’S THESIS DETAILS

Bachelor’s thesis title in English:

Automatic determination of shape and size of nanoparticles for biomedical applications

Bachelor’s thesis title in Czech:

Automatické určení tvaru a velikosti nanočástic pro biomedicínské aplikace

Guidelines:

Prepare a sample of gold nanoparticles and nano-rods for transmission electron microscopy (TEM). Take TEM images of nanoparticles with assistance. Design algorithms to evaluate TEM images of nanoparticles to determine their size, shape, and size distribution. Segment TEM images from real measurement data so that segmentation is feasible even on images that have a jagged background with measurement artifacts. Perform segmentation in Python. Automatically evaluate the pixel size for the specified scale. Evaluate the sizes and shapes of the nanoparticles in a few selected images that have a jagged background. Implement circular and elliptic Hough transforms to solve the problem of overlapping nanoparticles. Evaluate the success of segmentation and determination of nanoparticle shapes and sizes. Discuss the limitations of the proposed algorithms. Create a graphical user interface (GUI) for automatic shape and size evaluation.

Bibliography / sources:

[1] Chiwoo P. et al., Segmentation, inference and classification of partially overlapping nanoparticles, IEEE transactions on pattern analysis and machine intelligence, ročník 35, číslo 2, 2012
[2] Grulke, Eric A., et al., Differentiating gold nanorod samples using particle size and shape distributions from transmission electron microscope images, Metrologia, ročník 55, číslo 2, 2018
[3] Groom, D. J. et al., Automatic segmentation of inorganic nanoparticles in BF TEM micrographs, Ultramicroscopy, ročník 194, číslo 1, 2018

Name of bachelor’s thesis supervisor:

doc. Ing. Vladimíra Petráková, Ph.D.

Name of bachelor’s thesis consultant:

Ing. Miroslav Hekrdla, Ph.D. (ÚFCHJH AV ČR, v.v.i.)

Date of bachelor’s thesis assignment: **14.02.2023**
Assignment valid until: **20.09.2024**

doc. Ing. Martin Rožánek, Ph.D.	prof. MUDr. Jozef Rosina, Ph.D., MBA
Head of department	Dean

Prohlášení

Prohlašuji, že jsem Bakalářskou práci s názvem „Automatic Determination of Size and Shape of Nanoparticles for Biomedical Applications“ vypracovala samostatně a použila k tomu úplný výčet citací použitých pramenů, které uvádím v seznamu přiloženém k semestrálnímu projektu 2.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č.121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů.

V Kladně dne ....

Monika Pulcová

Poděkování

Ráda bych poděkovala mé vedoucí doc. Ing. Vladimíře Petrákové, PhD. za její čas, spoustu dobrých rad a v neposlední řadě za pomoc, kdykoli jsem ji potřebovala. Dále bych ráda poděkovala mému konzultantovi Ing. Miroslavu Hekrdlovi, PhD. za velmi výraznou pomoc s metodami zpracování obrazu, s kódem v jazyce Python i s textem práce, dále bych mu chtěla poděkovat za vyvětlení všeho, co jsem potřebovala a za odpovědi na spoustu mých dotazů. Také bych ráda poděkovala Ing. Niklasovi Hansenovi za pomoc při měření dat.

ABSTRACT

Automatic Determination of Size and Shape of Nanoparticles for Biomedical Applications:

Nanotechnology is widely used in biomedical research. For its usage it is important to know sizes and shapes of particles. These parameters can be measured by transmission electron microscopy (TEM), but analysis of images from transmission electron microscope is complicated and tedious. It would be easier if method for automatic evaluation of these parameters would exist. The goal of this project is to create program for automatic determination of sizes and shape of nanoparticles and nanorods from TEM images.

TEM samples of nanoparticles and nanorods of various sizes were used and TEM images were acquired. Image analysis methods were used for image adjustement and two image segmentation methods were used. First method is watershed transform and second method is circle Hough transform.

Python program was created and tested on various images of nanoparticles and nanorods. Sizes and shapes of nanoparticles and nanords were calculated and histograms of sizes were plotted.

Key words

nanoparticles, transmission electron microscopy, image segmentation, watershed transform, Hough transform

Table of Content

List of Symbols and Shortcuts

List of symbols

Symbol	Unit	Meaning
cx, cy	px	Coordinates of center of circle in image
r	px	Circle radius in image

List of shortcuts

Shortcut	Meaning
NP	Nanoparticle
AuNP	Gold nanoparticle
NR	Nanorod
GNR	Gold nanorod
TEM	Transmission electron microscopy
HT	Hougt transform
CHT	circle hough transform
ROI	Region of interestt
LUT	Look up table
GUI	Graphical user interface

List of Figures

List of Tables

List of Algorithms

1 Introduction

Nanoparticles and nanorods are widely used in biomedical research. For their usage, it's essential to know the size and shape of these particles in the solution. The most used method for this estimation is transmission electron microscopy (TEM), which is a very well-known method, but data evaluation is very complicated and includes several image processing methods. The biggest part of image processing of TEM images is image segmentation. There are plenty of segmentation algorithms, but each set of data is unique and requires a totally different approach. Another huge problem is that nanoparticles and nanorods are often overlapping, so the perfect method should divide them well while also taking into account overlapping areas. TEM images also contain a lot of noise, particles may have an in-homogeneous background, and even particles may not be of homogeneous intensity. Simply there are so many sizes and shapes of nanoparticles or nanorods, that it is challenging to create an algorithm which can automatically analyze TEM images of nanoparticles.

In this work, we tackle this challenge by developing a program, which is able to analyze different TEM images of gold nanoparticles. The program uses the watershed transform algorithm, which is one of the segmentation methods, to separate particles. We focus on the usage of TEM image scales for estimating parameters used in image adjustment algorithms. We also test the hough transform segmentation algorithm and its use for separating overlapping particles.

2 State of the Art

2.1 Nanotechnology and Nanoparticles

Nanotechnology is a rapidly developing field, which is widely used in diagnostics, therapeutics, and generally biomedical research. Innovations in this discipline of science brings big opportunities not only for biomedicine but also for electronics, photonics, or energetics. There are plenty of structures, sizes, and shapes of nanoparticles (NPs) and knowledge of their sizes and shapes is crucial for their usage [1, 2, 3].

Nano means 10^{-9}, so nanotechnology works with structures with size in the order of nanometers. For example, a hydrogen atom has a diameter of 0.1 nm. If one of the dimension of the material lies between 1 to 100 nm then it has different physical and chemical properties than bulk material. It is considered nanotechnology. It stands on the border of the macroworld described by classical physics and the microworld described by quantum physics. The properties of nanomaterials can be very different than the same bulk material, it can change color, electrical conductivity, toxicity, and others. For example solution of gold nanoparticles has red color. Another important parameter is fact, that nanomaterial has a far greater surface and the mass is so small, that electromagnetic force becomes more important for the behavior of the particles. Inorganic nanomaterials are also able to interact with living systems, which has huge use in medicine. Nanomaterials can be created by two methods. The first method is called Top-down and starts with greater structures and ends with smaller particles. The opposite method is called Bottom-up and creates nanoparticles from single atoms. There are plenty of types of NPs, the most famous are carbon NPs - for example carbon nanotubes, but gold or silver NPs are also very important due to their properties [4, 5, 6, 7, 8, 9, 10, 11].

Nanotechnology is a quite new and developing field, first use of the word 'nanotechnology' was in 1974 by N. Taniguchi. Since then nanotechnology is developing rapidly. But ideas, that there would be some opportunities in the small world, were even earlier. In 1931 TEM (transmission electron microscopy) was created and in 1959 Richard Freyman, a physicist who got the Nobel prize, said: 'There is plenty of room at the bottom, an invitation to enter a new field of physics.' Nanotechnology was even used before people knew about it. Archeologists for example found cup from the Roman empire changing color depending on the angle of view. The effect was caused by nanoparticles [6, 7, 10, 11].

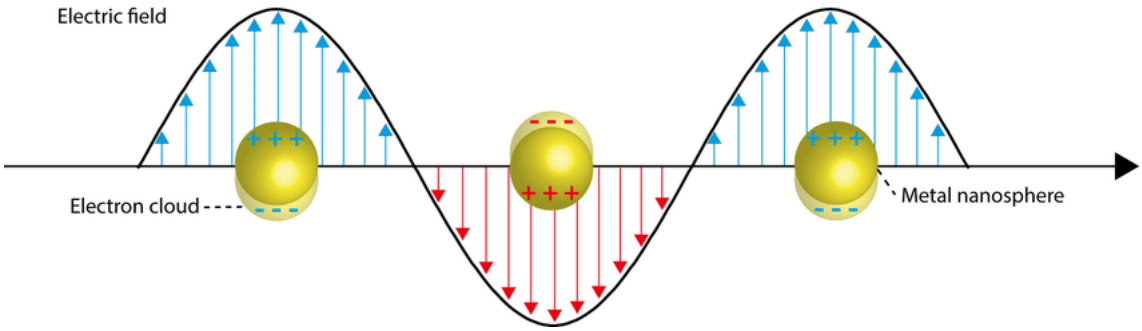


Fig. 2.1: Principle of interaction plasmon with an electrical field [16].

2.1.1 Plasmon Resonance Nanoparticles

A special type of NPs with huge utilization in biomedicine includes surface plasmons, which are a special type of electrons on the surface of NPs. These NPs can interact with light or generally electromagnetic radiation of wavelength much bigger than the size of the NPs and localized surface plasmons are consequence of this radiation. This gives them outstanding optical and physical properties like strong absorption which facilitates using them for various optical sensors and other diagnostic methods. Localized surface plasmon resonance (LSPR) is an oscillation of surface-free electrons in spherical metal NPs and polarizing them. As a consequence of the induced dipole, the electron cloud is oscillating. The principle is demonstrated by Fig. 2.1 Optical responses of these phenomena can be calculated from Maxwell equations. Gold NPs (AuNPs) belong to the group of plasmonic NPs and are used in medicine and biology due to their good chemical stability and biocompatibility. Noble metal NPs in general are also easily bioconjugated (conjugation with at least one biomolecule). Hence, they are a popular choice for biodetection, gene therapy for musculoskeletal regeneration, cancer diagnostics and therapy or Covid Antigen test [12, 13, 14, 15].

This phenomenon is used for plasmonic biosensors. There are two types of these biosensors, first one works on the principle of surface plasmon resonance and is made from a thin film of noble metal. The second one uses nanoparticles which, are smaller than the wavelength of light, and their property of LSPR. Due to LSPR, NPs are able to absorb and scatter light with high intensity. It is possible to observe it using dark-field microscopy. Thus these are highly applicable to biochemical sensors and cellular imaging. NPs are also useful as transducers because the scattering spectra depend on the local refractive index and the spectra are shifted even with small changes in the refractive index. When the refractive index increases, the scattering spectra shifts to red (bigger wavelength). It has also an advantage in comparison with film biosensors, because NPs can be used for very small volumes and gives a better spatial resolution. The spectra of these NPs depend among other things on their size and shape. There are spherical NPs and nanorods which can be of various aspect ratios (AR) [17, 18, 10, 11].

NRs are able to give two LPSRs (2.2) in comparison with spherical NPs because the electrons can oscillate in two paths. Oscillation of electrons inside the longer paths (in the longitudinal dimension of the NR) is called longitudinal resonance and produces light of longer wavelength, the other is called transversal resonance and produces light of shorter wavelength. GNR solutions can have various colors depending on their size and aspect ratio (2.3) [18, 19, 20, 21, 22]. For both NPs and NRs, it is very important to know their sizes and shapes.

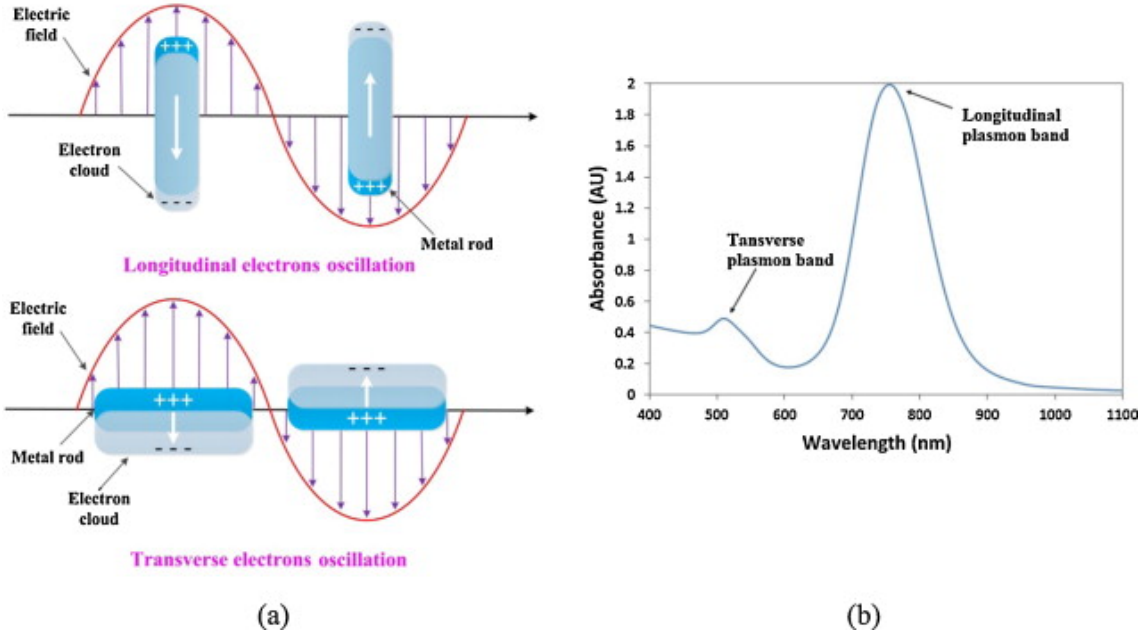


Fig. 2.2: Principle of LPSR in GNRs [23].

2.2 Transmition Electron Miscroscopy

One of the most common methods for the determination of the size and shape of AuNPs and GNRs 2.4 is transmission electron microscopy (TEM) which is a very accurate and well-known technique. It is just necessary to make the images of the sample and use a computer to count the number and size of the NPs. But there are some disadvantages. For example, there is the complicated and nontrivial preparation of the sample, and the measurement itself can take a long time, as well as the requirement of the actual microscope itself. On the other hand, it can be used for every size and shape of AuNPs and even GNRs [25].

The TEM microscope 2.5 works on the principle of electrons passing through the conductive sample and measuring the intensity of the beam that passes through. It consists of an electron gun, a system of lenses, and a fluorescent screen or digital detector [21].

Fig. 2.3: GNR solutions of different colors [24].

The electron gun is made of a cathode which is heated and has a negative potential. The potential is equal to the accelerating voltage. Then there is an anode with positive potential. The electrons come from the cathode to the anode and pass through an axial hole in the anode. Then there are condenser lenses which are used for focusing the electron beam. In most cases, there are two lenses. The first lens reduces the image and the second focuses it onto the specimen. The specimen is made by a measured sample and TEM grid. It is situated in the specimen stage which can be taken out of the microscope. Then there is an objective lens which produces a real image and projects it onto the projector lens which produces an enlarged image. There are more than one projector lens due to the fact that one lens can provide just a magnification of 5:1 and every next lens increases the magnification range. The last step is detecting the image. There are two options. The electron beam may be absorbed on the fluorescent screen which produces visible light. Or it may be detected by a camera and displayed on the monitor and saved [21].

2.3 Segmentation Methods

With TEM it is more complicated because the result can be obtained only after image processing. It is necessary to use segmentation methods to get the number, sizes, and shapes of NPs and NRs. Segmentation is an image processing technique, which transforms the image into regions, each representing one object, and includes dividing objects and background. Image processing can be divided into three layers,

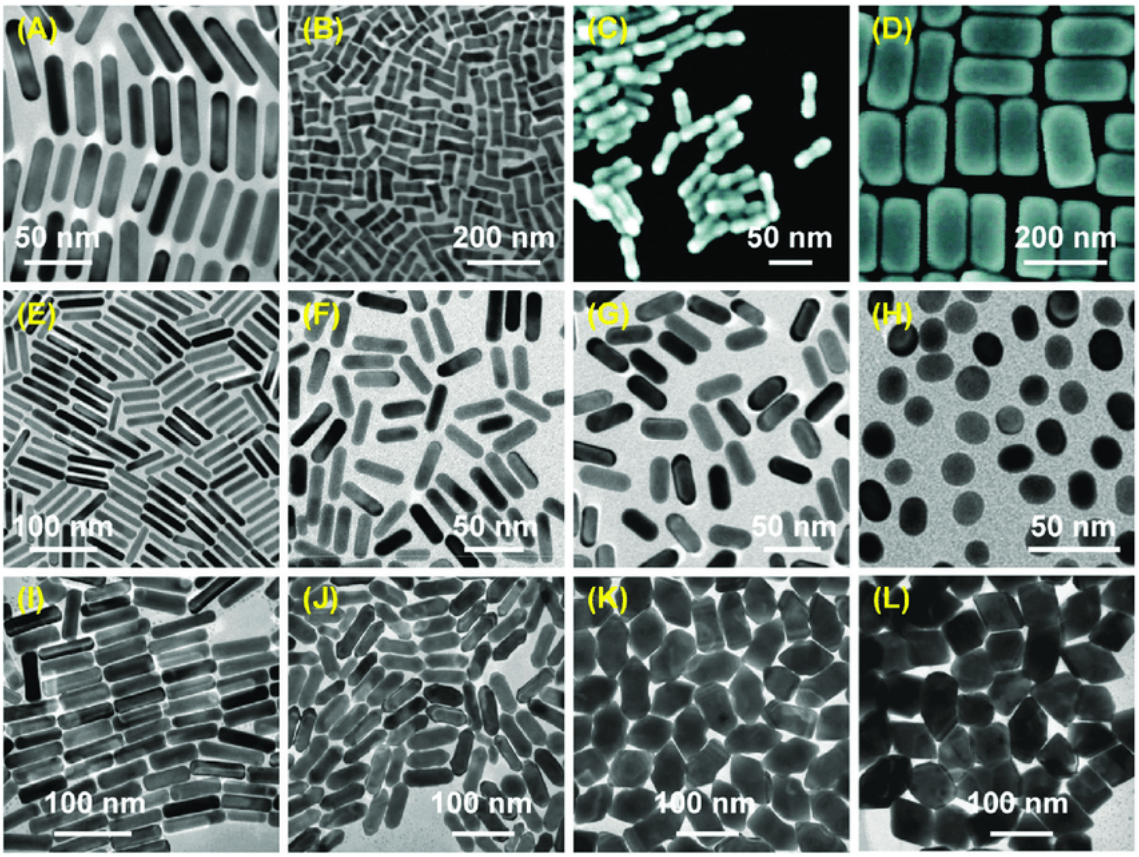


Fig. 2.4: TEM images of various shape and size GNRs [16].

segmentation belongs to the low layer, then there is image analysis which belongs to the middle layer, and image understanding in the high layer. There are plenty of methods of image segmentation but it is not easy to use them. Image segmentation is widely used in many diagnostic systems. It is used for automatic evaluation and quantitative analysis of microscopic and medical images. It is crucial that the segmentation is precise because if it is not, the algorithm will not work. Segmentation lies in finding boundaries of objects, it can be cells, anatomical structures, or nanoparticles. But there are more steps than the segmentation, the result has to be evaluated using other algorithms. There are more types of techniques, and some of them are based on edge detection using discontinuities in pixel values. Other techniques find similarities in regions, there are techniques like region growing, clustering and thresholding. There are, clustering methods, connectivity-preserving techniques, and region-based methods which collect regions with similar intensity, color, texture, etc [17, 26, 27, 28].

2.3.1 Thresholding Methods

The most basic and even fast methods are region-based methods. This group includes for example threshold segmentation and regionally grown segmentation. The first

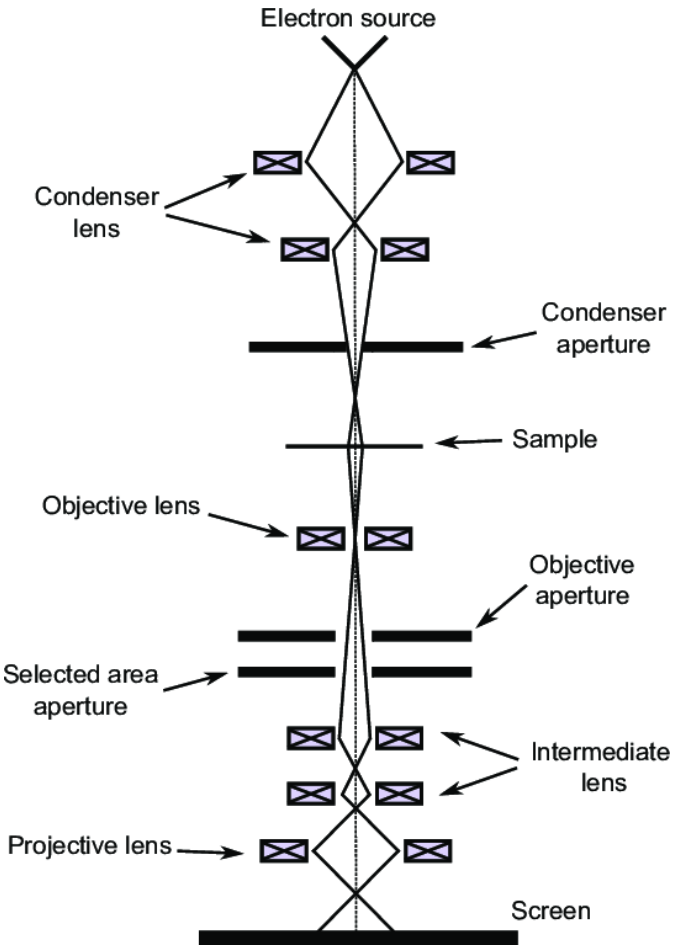


Fig. 2.5: TEM instrument schematic [16].

mentioned technique is very simple and fast, it just divides pixels depending on their value. Thresholding does not take into account the position of a pixel, it works just with its intensity. There are two types of thresholds, the global threshold method has the same threshold value for the whole image and the adaptive or local threshold method calculates the value for each point using average region intensity. The advantage of adaptive threshold is that it accommodates inhomogenous lightning of image. There are various ways how to determine the threshold value. Thresholding can be also divided according to how many thresholds there are. Bi-level thresholding is based on one threshold and divides pixels into two groups and multilevel thresholding selects multiple thresholds, thus there are several groups of pixels. One of the most used methods is called Otsu threshold (Fig. 2.6) and consists in finding value, where weighted within-class variance is minimal. This method expects that the histogram of the image is bimodal, which means there is sufficient difference between the background and region of interest intensity [27, 29, 28, 30].

There are also entropy-based techniques. Entropy is originally used in thermodynamics and it describes the level of disorder of a physical system. Shannon described entropy in informatics as a unit for measuring the efficiency of information in a sys-

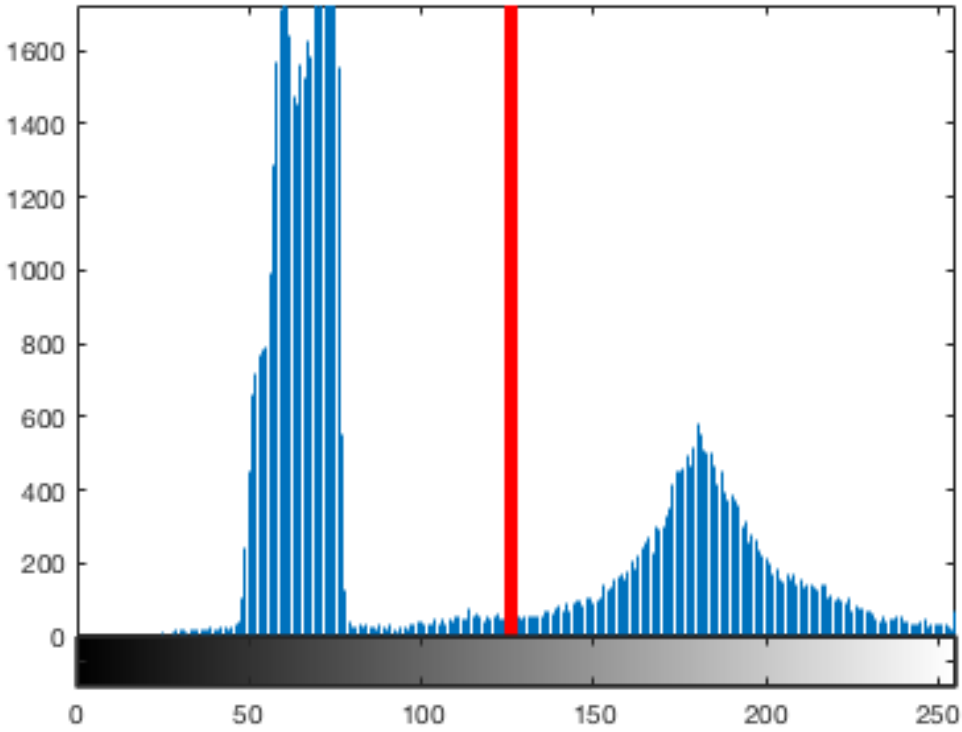


Fig. 2.6: otsu threshold in bimodal histogram [16].

tem. Entropic thresholding works on the principle of finding a threshold where the sum of the entropy of the background and the entropy of ROI is maximal. Another method is called minimum error thresholding and it considers histogram as a probability density function which includes a combination of two populations, the intensity of ROI pixels and background pixels and it determines threshold as value, where the overlap of these two populations is minimal. It uses the estimation of parameters of mean value, standard deviation, and probability. This algorithm works with a 2-D histogram, which can show the relationship between two variables. There is also a 2-D minimum error method which takes into account also position of a pixel. Multilevel thresholding determines more than one threshold and divides an image into more levels depending on intensity and color (in RGB images). It is ideal for images with color inhomogeneous background. Thus it is possible for example to assign both dark and bright pixels to zero and middle gray pixels to one. A disadvantage of these algorithms is it is often not possible to detect the number of thresholds automatically. There are methods which can determine it using several times lowpass or highpass filters in image histogram to decrease or increase the number of local maxima and minima. Another region-based type of image segmentation algorithm is called regional growth segmentation. It assumes that pixels of one region have similar values. The algorithm first selects the seed pixel and then joins similar pixels in the neiborhood connected with this pixel into the region. It

requires a threshold value which is used as the maximum difference between the seed pixel and pixels assigned to the region. A disadvantage of these methods is that it is slow and susceptible to noise [27, 29, 28, 30].

Another segmentation algorithm is called the watershed segmentation algorithm (Fig. 2.7) and is also a region-based method. The name and even principle comes from nature and it is applied to grayscale or binary images and binary mask. The goal of this algorithm is to find all connections in the image. Connected pixels mean pixels that create one object (for example nanoparticle). It is necessary to find small connected regions, from which can watershed begin to fill the mask. Theese regions are called seeds. There are more methods, how to find the seeds, one of them is called distance map. The first step of distance map algorithm works on the principle of distance from nonzero pixel which creates a distance map. Objects have to be of value zero and background of zero one, if it is reversed, the image has to be negated first. In the case of grayscale images, the distance is calculated from local minima. So objects have to be of lower value than the background. When the distance map is calculated, watershed transform can be applied, it starts filling pixels from the highest value (local maxima) of distance and stops before two connected objects join into one. According to [31] there is method called ultimate erosion, which works on principle of morphological erosion. It can be applied on binary image, where ROI has pixel value one and background zero. Algorithm uses erosion repeatedly until each region has a convex shape. Theese convex shapes can be used as seeds for watershed algorithm. Finding of theese seeds can be called decomposition into catchment basins. Basins represent individual objects or regions. Every pixel of an image is classified into one of the regions or the background, so noise can be a problem because it creates a large number of small regions. It is called over-segmentation. Thus image has to be filtered before applying watershed transform [26, 32].

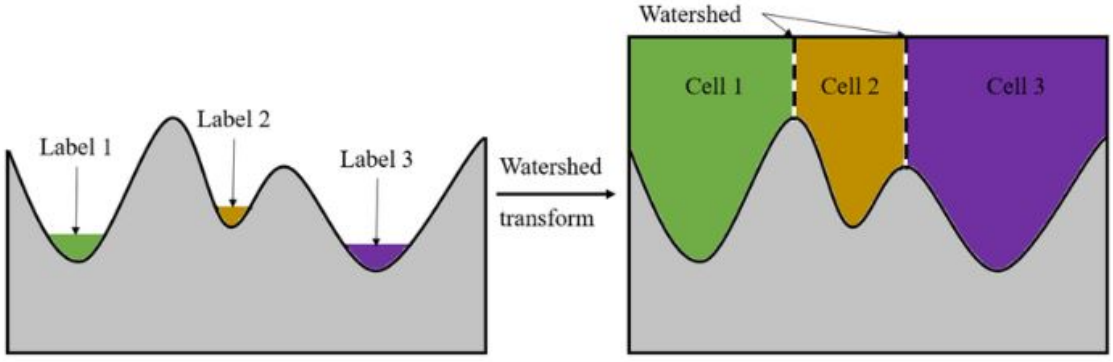


Fig. 2.7: Watershed principle [16].

2.3.2 Edge Detection Methods

The edge detection technique works on the principle of finding local changes and discontinuities in the image [33]. It works with spatial domain instead of histogram in contrast with thresholding. Edge detection means detecting high frequencies in an image, so it works like a high-pass filter. Spatial frequency is generally a number of cycles per distance, it has two components, frequency on the x-axis and y-axis. Spatial frequency has units pairs of lines per millimeter, which means how quickly low-value pixels shift changes with high-value pixels. This spectrum can be obtained using a discrete 2D Fourier transform, which is a function of x and y. Thus high-pass filter extracts high frequencies and the low-pass filter low frequencies [34]. There are four types of edges The step edge where the value changes from one pixel to another. The ramp edge, where the change is more gradual. The line edge has a steep change but then returns to its original value. The roof edge is made of slow change and in the peak value it also slowly returns back to the original value. There are more steps in the edge detection algorithm, first step is filtering, which is used for removing noise. For example salt and pepper noise, which means random white and black pixels in the image. It has to be very well balanced because filtering also reduces edge strange. Then there is the enhancement phase which points up pixels with a greater change of value. This step uses calculating gradient. The last step is called detection and it removes pixels with nonzero gradients which are not edges [32]. Greater changes in value can be detected using differential operators which can calculate derivatives. There are several types of these discrete operators, Sobel and Laplacian operators are of the most used. The Sobel operator (Fig. 2.8) calculates the first derivative in each pixel. The algorithm uses two 3x3 matrices, a transverse matrix which founds gradients in a horizontal direction and a longitudinal matrix which founds gradients in a vertical direction [28, 33]. Then the size of the gradient is calculated using the equation [33]:

$$G = \sqrt{G_x^2 + G_y^2} \tag{2.1}$$

And vector of the gradient can be calculated using the equation [33]:

$$\Theta = \arctan \frac{G_x}{G_y} \tag{2.2}$$

Zero angle corresponds with a longitudinal edge, where the pixels on left have a lower value than the ones in right.

Another type of differential operator is called the Laplacian operator (Fig. 2.9), which is matrix, which calculates the second derivative and detects zero-crossing of image intensity. It is more accurate than the Sobel operator, but it detects isolated

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Fig. 2.8: Sobel operators, horizontal and vertical [33].

pixels better than edges, so it is not possible to use it for images with noise [33]. Thus before using the Laplacian operator, it is necessary to perform a low-pass filter, which reduces noise. For example, the median filter can be used, because it reduces noise while keeping sharp edges. It is also possible to combine it with an edge-matching filter, which has minimal response to isolated pixels and good response to merged edges [35]. It consists of only one symmetric matrix, thus it is used for both directions. It can be also calculated using the equation [33]:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{2.3}$$

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

Fig. 2.9: Laplacian operator [33].

2.3.3 Clustering Methods

Clustering techniques work on the principle of dividing pixels into various groups (clusters) depending on their characteristics. A cluster can be defined as a group of pixels with similar properties to each other in the same cluster and diverse with each other clusters [36]. There are two types of clustering algorithms, hierarchical clustering is based on merging pixels with similar characteristics or splitting pixels with opposite characteristics and finding all possible combinations. Partitional clustering is an iteration method with successive optimization. Hierarchical clustering can be divided into two groups, first is called the agglomerative or bottom-up method and works on the principle of merging data based on their similarities, first, it defines every pixel as one cluster and then in every iteration merges similar clusters into one, until it reaches given number of clusters. The divisive or top-down method takes all pixels into one cluster and then in every cycle divides the most diverse pixels and from each cluster creates two [37, 36]. One of the most used clustering methods is called k-means (Fig. 2.10), which comes under partitional methods. The advantage of this method is, that it is easy, fast and the results are of high quality. The algorithm first randomly selects a defined number (k) of cluster centers and assigns each pixel to the nearest cluster. In every iteration, algorithm calculates new center for

every cluster using the mean value of all pixels in this cluster and then reallocates pixels depending on their new distances from centers. The algorithm finishes when there is no significant change in centers [36]. A disadvantage of this method is that it is difficult to estimate the k parameter.

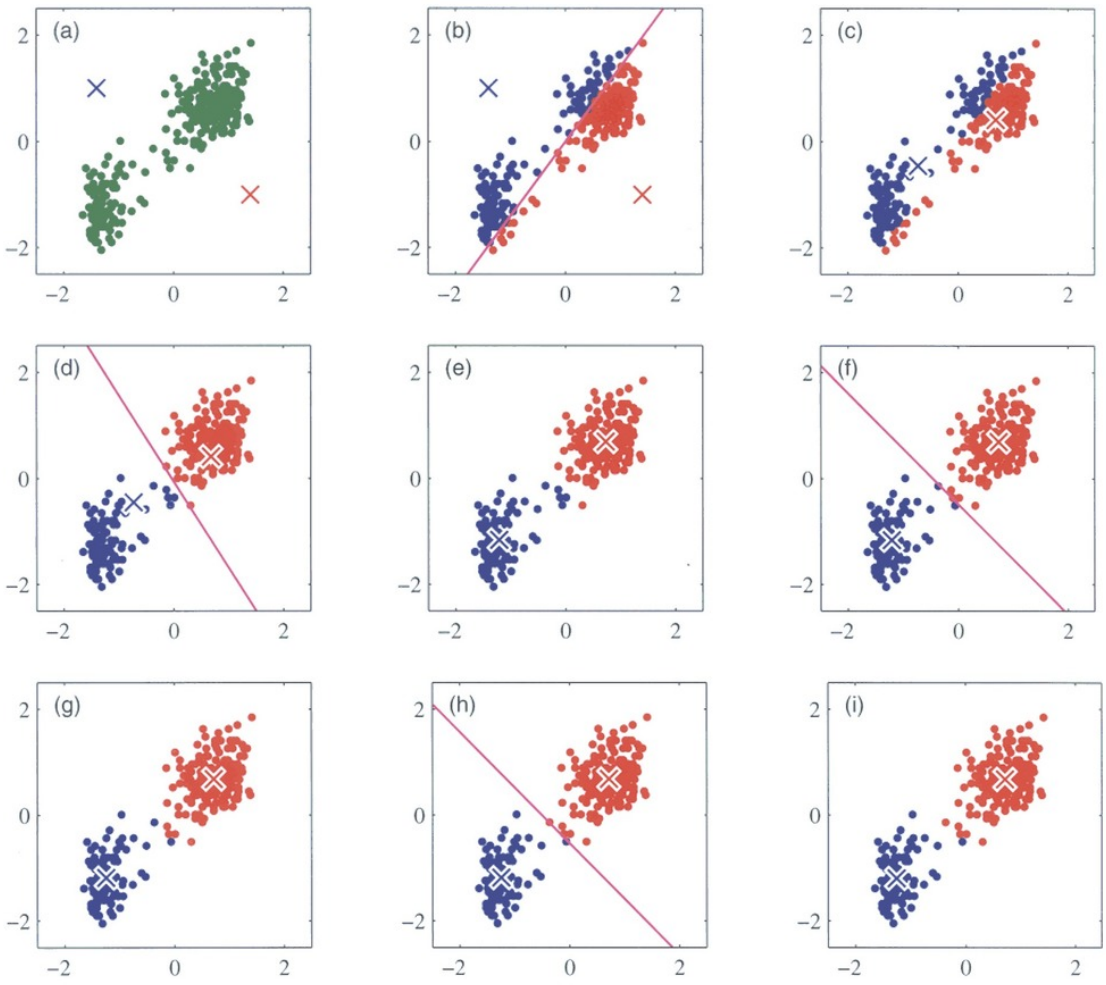


Fig. 2.10: K-means principle [38].

Another iterative algorithm is called fuzzy C-means clustering. It is based on fuzzy sets which do not assign each point just into one cluster. Instead of this, it calculates the vector of memberships for each pixel, which is a measure of how strongly it belongs to each cluster. It has to be numbers between 0 and 1, the sum of all memberships of one pixel must be 1. It is an advantage in comparison with k-means because it allows the overlapping of clusters. There is also a defined number of clusters (c). The steps of C-means are similar to k-means, the algorithm first randomly selects c centers, then calculates membership and updates cluster centers. A disadvantage is the same as in k-means, it is necessary to specify C parameter [36, 39, 40]. There are more clustering algorithms, for example, Gaussian clustering, but it has a disadvantage, that is very complex. It is also an iterative method and works on the principle of calculating probability [39].

2.3.4 Machine Learning Methods

There are also image segmentation techniques which use deep learning, which is a subset of machine learning using neural network architecture with many layers. These systems require a large number of sample images and very good computation power. In biomedicine, it is used for example for the automatic detection of cancer cells. Each neural network is composed of an input layer, some amount of hidden layers, and an output layer. Deep learning neural networks may contain about 150 or more hidden layers, in comparison with traditional neural networks, which have 2-3. It can also learn automatically just from large data sets. Neural networks are formed from nodes, which are connected with other nodes, these nodes are called neurons (Fig. 2.11). Each connection has weight, thus some of them are more important than others [41]. The output of neuron can be calculated using the equation:

$$y = f(\sum_{i=0}^{p} w_i x_i - \Theta) \tag{2.4}$$

where y is output, x_i is i-th input, w_i is weigth of i-th input, f is activation function and Θ is threshold of neuron [41].

There are two types of operations, the output of a neuron can be a linear function of the weighted sum of inputs which in each connection is multiplied by a weight, the bigger is input, the bigger is output. Or there can be a threshold and if the weighted sum of inputs reaches this value, the neuron becomes active and there is a signal, which is multiplied in each connection with weight, other way there is no signal [41].

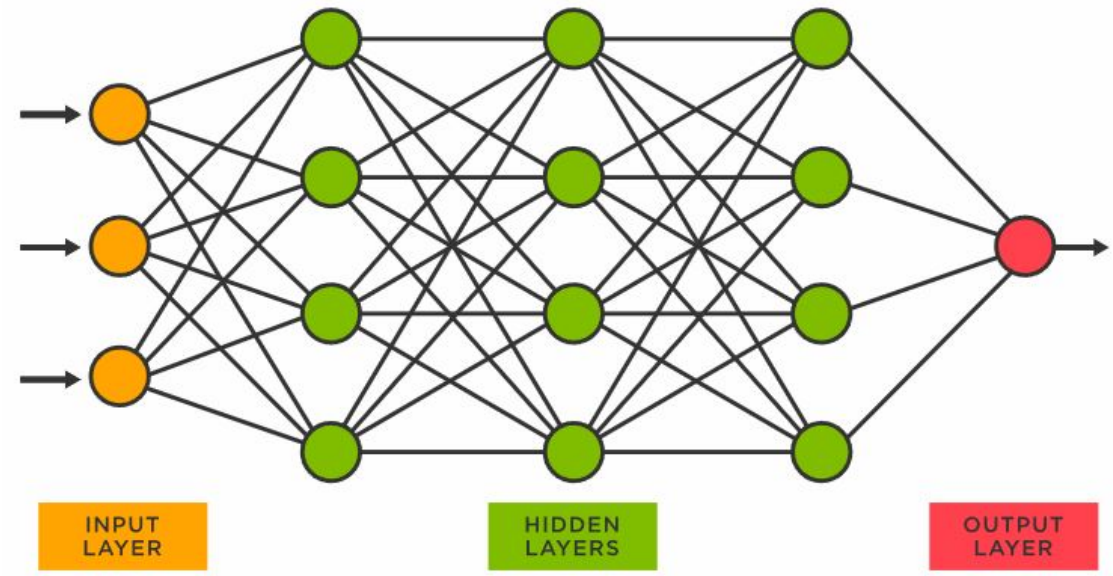


Fig. 2.11: Neural networks layers [16].

Artificial neural network (ANN) is a basic model of deep neural networks. It is composed of perceptrons, which means that the signal moves just forward and the input is connected to the output. it uses back-propagation (feedback) to reduce errors while learning, during this it updates the weights of connections depending on the error of NN results. The NN tries to reduce the error into required value. This learning process is called adviser learning [42]. Convolutional neural networks (CNN) are types of deep neural networks that are used for images because it contains 2D layers. It learns directly from sample images [26]. In CNN signal moves just forward through all the hidden layers and output is connected with input through these layers. CNN has three types of hidden layers, the convolutional layer, which contains the convolution of the kernel matrix (filtering), which includes learned parameters, and the image, which creates a new matrix called the activation map. Each CNN contains multiple of these filters. The pooling layer merges each region (a group of pixel) into one value, there are more methods of this pooling function, it can be the mean value, the weighted value depending on the distance from a center pixel of a region, or the max value of the region. It reduces the size of output, thus computational complexity is decreased. Then there is a fully connected layer, which connects all neurons from the previous layer with each neuron from the following layer. Fully connected layer works as classical NN and does not include image shape. Then there usually are also nonlinear layers placed after the convolutional layer, these layera use non-linear operators to activation map. A prerequisite for this method is a large number of relevant sample labeled images for training [42, 43]. A recurrent neural network (RNN) is not a feed-forward neural network, it means the signal does not move just forward like in ANN and CNN. There is a recurrent path which is able to affect input. It is connected with feedback, which works like memory, which connects a sequence of data. RNN works with previous data, so it can take it into account and output depends on it. It can work well for time dependent data, but it is exploitable also for images, where these sequences of data represent in image processing spatial domain [42].

3 Goals

The goal of the project is to create a program for the automatic determination of sizes and shapes of nanoparticles and nanorods from TEM images. Particular goals are:

- Prepare TEM samples of gold nanoparticles and nanorods.
- Acquire TEM images of nanoparticles and nanorods with assistance.
- Suggest proper segmentation algorithms for TEM images.
- Create program for segmentation of the images and automatical evaluation of sizes and shapes of nanoparticles using Python programming language and its libraries (OpenCV, Scikit-image, Numpy).
- Suggest and test the possibility of using the Hough transform for solving the problem with overlapping particles.
- Perform image segmentation on real data with jagged background and measurement artifacts.
- Automatically evaluate pixel size for a given scale.
- Evaluate the size and shape of nanoparticles on the sample images.
- Evaluate algorithm performance using proper methods.
- Discuss the limitation of the algorithm.
- Create a graphical user interface (GUI) for the Python program.

4 Methods

4.1 Measurement

TEM measurement includes sample preparation. It is very important to have the sample dry completely during the TEM measurement, so the preparation must start a few hours before. The very first step is getting the copper grids (Fig. 4.1) with carbon coating. The reason for using a copper grid is its conductivity, so the electrons from the TEM can go through the grid easily and it also is not as expensive as gold or platinum grids. In the second step, it is necessary to adsorb the solution that is to be measured onto the grid. The volume of the pipetted solution has to be proportional to the size of the grid. The samples are being prepared using drop-casting, pipette with volume 1 μL was used. After that, the samples have to become dry. For this reason it was prepared several hours before the measurement.

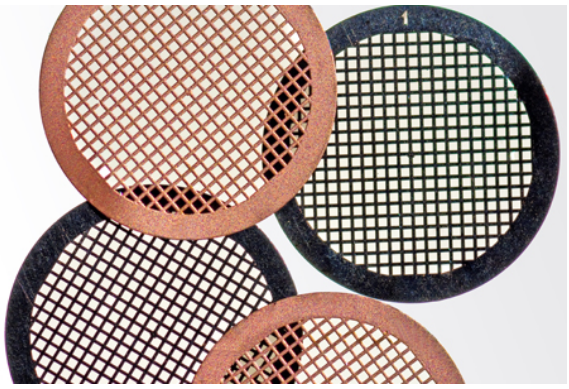


Fig. 4.1: Image of copper grid [44].

We made two measurements. We used AuNPs and GNRs NanoXact samples purchased from nanoComposix San Diego, USA. For the first measurement, we used samples shown in table 4.1 and for the second measurement we used samples shown in table 4.2.

Tab 4.1: First measurement samples

catalog number	size [nm]	absorption peak [nm]	type
TJC0090	-	660	GNRs
SAM0010	20	-	AuNPs
DAG3774	50	-	AuNPs
DAG3567	80	-	AuNPs

The electron microscope measurement itself was done by a microscope technician. The measurement begins with putting the grid with the sample into the microscope. Then it is necessary to focus on the place where the sample is. It is

Tab 4.2: Second measurement samples

catalog number	size [nm]	absorption peak [nm]	type
AUCN40-50M	40	-	AuNPs
AUCN80-50M	80	-	AuNPs
-	-	800	GNRs

good to make more than one picture of each sample. It is also important to find places where there are some particles but where these do not overlap too much. We used UHR TEM (ultra high-resolution transmission electron microscope) JEM-2100Plus with a maximum accelerating voltage of 200kV, guaranteed resolution of 0.14 nm, and magnification x30 to x800 000. We used an acceleration voltage of 80kV [45].

4.2 Image Analysis

Another step is creating a program for image analysis. We chose to use Python programming language. We used several Python libraries, predominantly Scikit-image and OpenCV-2 Python libraries. Scipy ndimage was also used in some functions. All these libraries are using Numpy, which is a library used for work with matrices. We also used Matplotlib Pyplot library for plotting and saving images.

4.2.1 Image Adjustment

The first step after loading the TEM image is converting it into a grayscale, so it could be filtered. TEM images can be of different scales, which has an impact on evaluating the number and size of visualized nanoparticles on each image. We chose to use the median filter for denoising images. Filtering works on the principle of convolution and the size and appearance of the convolution mask determines the type of the filter. The median filter belongs to nonlinear filters which means that the output is not a linear function of the input. The median filter simply counts the median value from the neighborhood of a certain pixel for every pixel in the image. The advantage of this filter is that it preserves edges and removes noise. The size of the convolutional kernel or pixel neighborhood depends on the scale of the input image and also on the type of particles because nanorods are usually smaller than nanoparticles so it has to be blurred less. The principle of the median filter can be seen in Fig. 4.2.

Then grayscale image has to be transformed into a binary image. It is usually done by finding the appropriate threshold. Pixels with a value below the threshold are transformed to zero and pixels with a value above the threshold are transformed into 1 or 255 (it depends if binary image values are of type boolean with just 0 and 1 or 8-bit unsigned integer with values from 0 to 255). In the case of this project binary

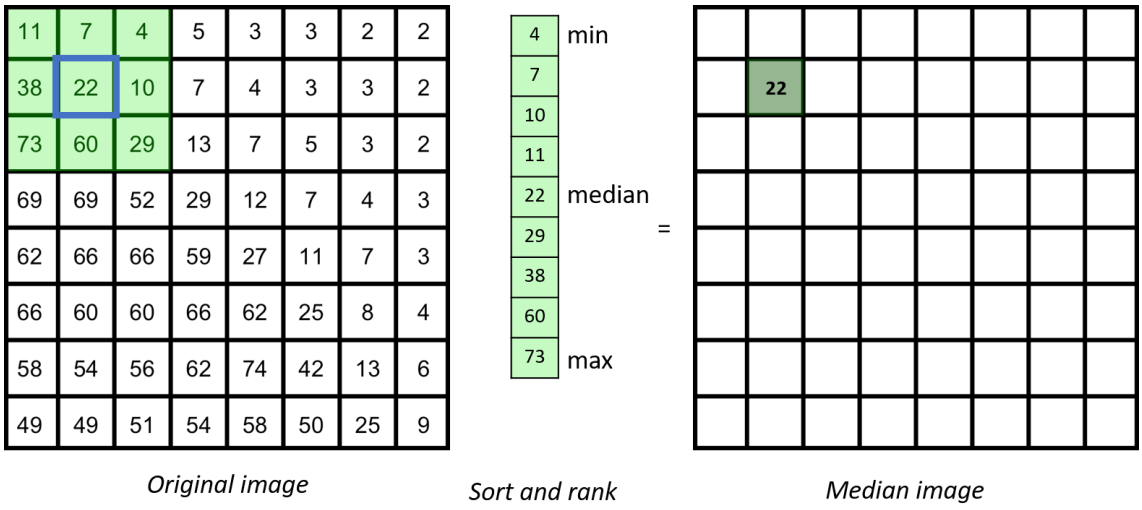


Fig. 4.2: Principle of median filter [46].

image is also inverted. In TEM images particles are black on a white background so in binary image particles pixels should have value zero and background pixels should have value one and for this application, it is better to invert it. The final equation, which consists of thresholding and inverting image can be seen on equation 4.1.

$$F(x) = \begin{cases} 1 & \text{for} \quad x(i,j) < threshold \\ 0 & \text{for} \quad x(i,j) <= threshold \end{cases} \tag{4.1}$$

Where $F(t)$ is LUT (look up table) function and $x(i,j)$ is the value of a pixel with i,j coordinates.

Several types of thresholding techniques exist. These techniques use the histogram for calculating the appropriate threshold. We used two different techniques in this project - the minimum threshold method and the Otsu threshold method. The minimum threshold method finds pixel value with minimum frequency in the histogram and this value is determined as the threshold t.

$$t = min(histogram(I)) \tag{4.2}$$

where I is a set of pixel values in an input image.

Another method is called Otsu thresholding and it is a bit more complicated algorithm. It consists of calculating within-class variance, which is defined as the weighted sum of variances of background class and particle class. Within-class variance can be calculated using equation 4.3 [47].

$$\sigma^2(t) = n_{bg}(t)/N \cdot \sigma^2_{bg}(t) + n_p(t)/N \cdot \sigma^2_p(t) \tag{4.3}$$

Where $\sigma^2(t)$ is within-class variance for current threshold, $n_{bg}(t)$ is number of pixels below threshold t (in background class), N is number of all pixels in image, $\sigma^2_{bg}(t)$ is inside-class variance for pixels below threshold t, $n_p(t)$ is number of pixels above threshold t (in particle class) and σ_p is variance for pixels above threshold t [47].

The algorithm of Otsu thresholding can be seen in Alg. 1.

Algorithm 1: Otsu thresholding [47].

```
Input  : bitmap [I_{i,j}], i = 0, 1, ··· , 255, j = 0, 1, ··· , 255
Output: threshold value t
n = length(width(I))
σ² = inf
for t_0 ← 0 to 255 do
    B = [ ] /*Set of background pixels*/
    F = [ ] /*Set of foreground pixels*/
    for i ← 0 to length(I) do
        for j ← 0 to width(I) do
            if I_ij < t_0 then
                B.append(I_ij)
            else
                F.append(I_ij)
            end
        end
    end
    w_B = length(B)/n /*Weight of background*/
    w_F = length(F)/n /*Weight of foreground*/
    σ2_within = σ²_B × w_B t + σ²_F × w_F t
    if σ²_within < σ² then
        t = t_0
    end
end
return t
```

It is done iteratively for every pixel value. The background class consists of pixel values below the current value and the particle class consists of values above the current value [47]. Otsu thresholding is a more advanced method and would work better, but in nanoparticle images with larger scales (there were just a few dart particles in the center of the image) this method calculated too high threshold. More than just particle pixels were determined as foreground. So it was chosen to use the minimum thresholding method for nanoparticles with larger scale (scale larger than 200 nm) and the Otsu thresholding method for remaining nanoparticles and all nanorods. The principle of Ostu thresholding can be seen in Fig. 4.3.

After binarizing the image we use mathematical morphology operation to clean up the image. Two basic operations of mathematical morphology are called erosion

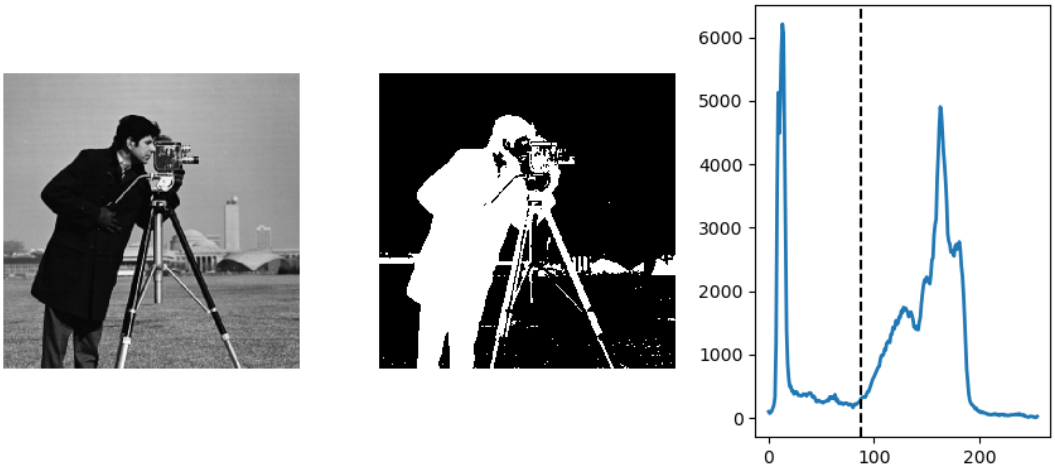


Fig. 4.3: Principle of Otsu thresholding [48].

and dilation. These methods work on the principle of interaction with the structuring element, which determines the shape of the pixel neighborhood. Dilation is used for filling holes and increases the size of objects. The algorithm changes pixel value to one if there are enough white pixels in its neighborhood. Erosion is the opposite of dilation, it is used for removing small objects and decreases the sizes of objects. The algorithm changes the pixel value to zero if there are not enough white pixels in its neighborhood. There are also two methods which consist of both erosion and dilation. Opening first uses erosion and later dilation, it is used for removing small objects. Closing first uses dilation and later erosion, it is used for removing small holes. Opening and closing are better because it changes the sizes of objects less than just erosion and dilation. The Scikit-image and OpenCV libraries also contains functions designed perfectly for removing small objects or holes. We used these scikit-image and OpenCV functions in this project for removing artifacts from the background and fill holes in particles. The principle of these operations can be seen in Fig. 4.4.

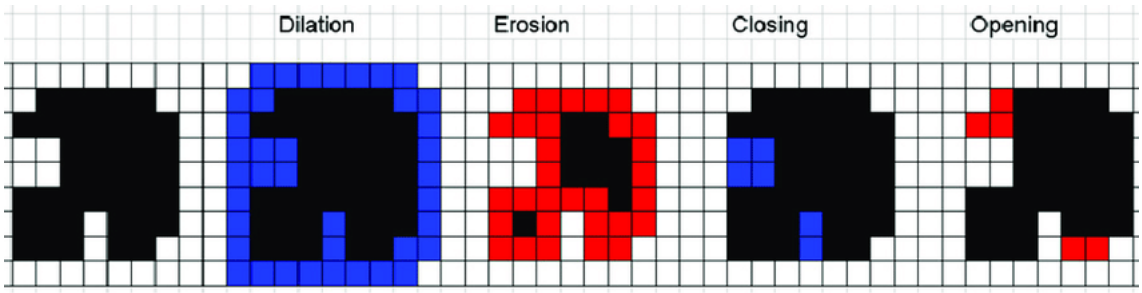


Fig. 4.4: Principle of mathematical morphology operations, red pixels are removed and blue pixels are added [16].

4.2.2 Image Segmentation

The main part of this project is image segmentation, which separates particles so these could be analyzed. There are plenty of methods used for segmentation and it is a quite difficult task to choose the best method and implement it. For this project we have chosen watershed transformation as the main method. This algorithm starts with seeds and binary mask. Seeds are points or small clusters of points. In a perfect world, there should be only one seed per particle in the image. The algorithm takes seeds and starts filling binary mask from these points. It is called a watershed because seeds can be imagined as valleys (or local minima) and the algorithm fills every valley with water of different value starting from local minima (these are represented using the seeds). It ends when it reaches the edge of the binary mask or when it meets water from another valley. A labeled image is created, each pixel in one label has the same number, and the background has zero. The principle of a watershed can be seen in Fig. 4.5 [26, 48].

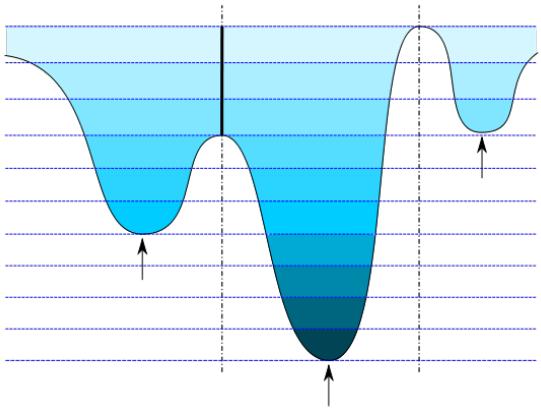


Fig. 4.5: Waterhsed transform principle [49].

There are more methods how to find seeds, in this project we tried to use a method called a distance map. It works on the principle of finding euclidean distance (number of pixels) for each one pixel to the closest zero pixel. Distance map can be created using several algorithms, the easiest but inefficient way can be implemented using multiple morphological erosion operations. Each step removes one layer of pixels. The value of each pixel is determined as a number of erosiosn needed to remove this particular pixel [50, 48].

It was also decided to try method described in [31] called ultimate erosion. Due to this reference it should be better way to find seeds for watershed segmentation. It works on principle of repeating morphological erosion applied on the binary image. In each step of the cycle algorithm evaluates the convexity of each connected region in the image and if it is convex, the region is classified as seed and removed from

set of regions for next erosion cycle, otherwise it stays in this set [31]. Principle of this algorithm can be seen in Alg. 2.

Algorithm 2: Ultimate erosion algorithm [31].

```
Input  : binary mask (set of connected regions) I
Output: Binary mask of seeds (set of connected regions) I
I^(0) = I
while I^(k-1) ≠ I^k do
    for R_i in I^(0) do
        /*Where R_i is connected pixels in I*/
        for A_i^(k-1) in R_i do
            /*Where A_i^(k-1) is connected component in k-1 iteration in R_i
            if A_i.isConvex() = False then
                R_i = erode(A_i^(k-1))
            end
        end
    end
end
return I
```

Watershed perfectly divides non-overlapping particles. When dealing with overlapping particles, two cases may occur. Watershed divided particles well but did not count with their parts, which overlap, or it does not divide particles well so it finds one big label. Due to this problem, we decided to use also another segmentation method. First ROI (region of interest) is defined. ROI is defined for each overlapping particle. Each ROI is cropped image which contains only these overlapping particles. Detection of overlapping particles is done by finding each connected region. It is done by evaluating convexity of each label and if the region is not convex, it is classified as overlapping particles, and the ROI of this object is created. We decided to perform a Hough transform (HT) to detect circles in the image. HT transform works on the principle of transforming image space into Hough space, which means mathematical transformation from 2-D image spatial axis (x, y) into 3-D axis of circle centers coordinates (cx, cy) and cirlce radii (r). Circle Hough transform (CHT) takes every point in the binary edge image, considers it a circle edge, and finds every potential center of this edge point for given set of radii (Fig. 4.6) [51].

A circle can be described by a circle equation (see eqn: 4.4).

$$(x - cx)^2 + (y - cy)^2 = r^2 \tag{4.4}$$

Where x, y are the axis, cx, cy are center coordinates and r is the radius of a circle [52].

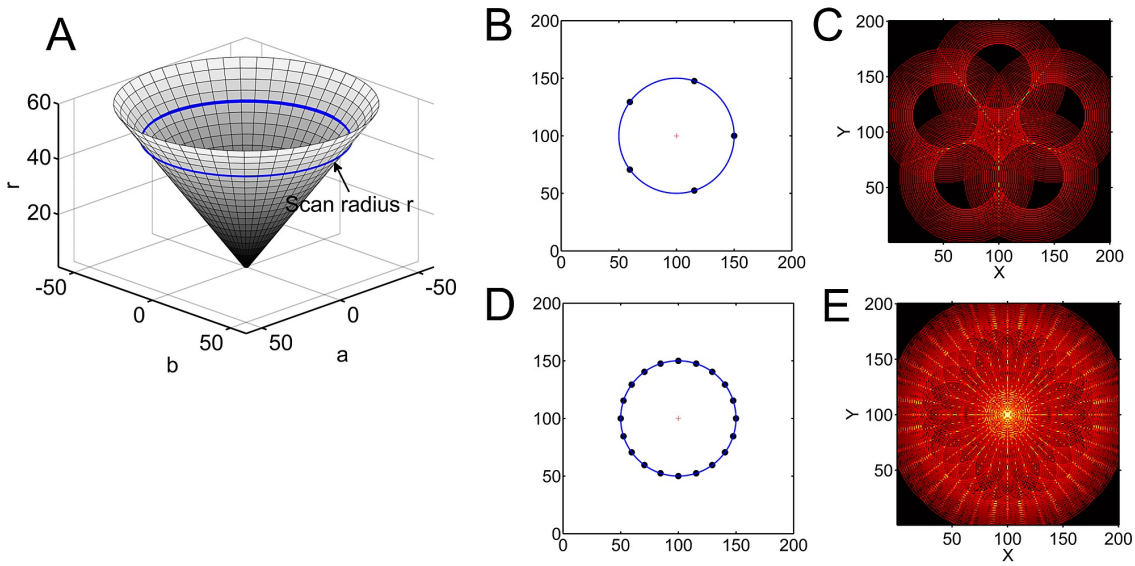


Fig. 4.6: Principle of circle Hough transform, taken from [16].

$$\text{(a)} \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{(b)} \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Fig. 4.7: Horizontal (a) and vertical (b) Sobel kernel, [53].

It means for each x and y in the potential circle edge there is a function of cx, cy, and r, which describes the coordinates of potential circle centers and potential circle radius. For each radius, there is a circle in the Hough space for each point in the image space and a point in the Hough space for each circle in the image space. Limits of cx and cy are given by image size, but there is no limitation for radii. So the limitation must be defined manually. Function in scikit image library calculates cx, cy, r, and weight of each detected circle, which means how detected circle fits the edges. The first step in Hough transform is performing Canny edge detection, which is an algorithm containing high pass filtrating (marks-up edges) and binarizing, so the result is a binary image with marked-up edges [48]. Edge detection uses the Sobel operator (Fig. 4.7).

Then gradient image can be calculated using equation 4.5 [53].

$$|G| = \sqrt{I_x^2 + I_y^2} \tag{4.5}$$

Where I_x, I_y are images filtered by the vertical and horizontal kernel [53].

The next step is removing lines, which are too thick using Non-Max Suppression. It works on the principle of going through all pixels in a gradient image and selecting maximum values in direction of the edge. The last step in this algorithm

is double thresholding. Two thresholds are defined, pixel values below the first threshold are determined as non-relevant and pixel values above the second threshold are determined as strong, pixel values between two thresholds are determined as weak. Strong pixels are assigned as one, and non-relevant pixels are assigned as zero. Weak pixels are divided using the hysteresis mechanism, which works on the principle of assigning weak pixels into strong ones if at least one strong pixel lies in the neighborhood of the current weak pixel, otherwise, it is assigned as a non-relevant pixel. This process can be seen in Alg. 3 [53].

Algorithm 3: Canny edge detection, [53].

Input : bitmap $[img_{i,j}]$, $i = 0, 1, \cdots, 255$, $j = 0, 1, \cdots, 255$
Output: Binary edge image B
$I = gaussianFilter(I, sigma = 1)$
$M_H \leftarrow$ Sobel Horizontal Operator 4.7 (a)
$M_V \leftarrow$ Sobel Vertical Operator 4.7 (b)
$H = convolution(I, M_H)$
$V = convolution(I, M_V)$
for *i in I* **do**
  for *j in I* **do**
    $G_{ij} = \sqrt{H_{ij}^2 + V_{ij}^2}$
  end
end
$G = nonMaxSuppresion(G)$
$t_1 = 0.05$
$t_2 = 0.09$
for i, j *in image* **do**
  if $G_{ij} < t_1$ **then**
    $I_{ij} = 0$
  else if $G_{ij} > t_2$ **then**
    $img_{ij} = 255$
end
for *i in weak pixels (other than 0 and 255)* **do**
  for *j in weak pixels* **do**
    if *any strong pixel in neighborhood of* w_{ij} **then**
      /*Where $w_{ij} is weak pixel$i,j*/
      $I_{ij} = 255$
    else
      $I_{ij} = 0$
    end
  end
end
return I

These edges can be used in function for HT. CHT returns the weight called accumulator, center coordinates, and radius for each detected circle. Then the result has to be filtrated, it may contain more than one detected circle for just one circle in

the image or it may detect wrong edges (too small or too big in comparison to other circles). So it is defined minimum distance between two centers and the minimum weight of the detected circle. Then circles smaller than half of the biggest circle are removed [48, 51].

4.3 Statistics

Jaccard or Dice (F1 score) indices are usually used for the evaluation of segmentation fruitfulness (and also for neural network results evaluation). These indices have both the same use and gain values between zero and one, the bigger the value is, the better performance segmentation provides. Jacccard index can be calculated using equation 4.6 and Dice index can be calculated using equation 4.7 [54, 55, 56].

$$JSC = \frac{TP}{TP + FP + FN} \tag{4.6}$$

$$DSC = \frac{2TP}{2TP + FP + FN} \tag{4.7}$$

Where TP (true positive) are pixels, which were classified well as ROI, FP (false positive) are pixels, which were classified as ROI instead of background and FN (false negative) are pixels, which were classified as background instead of ROI.

Dice and Jaccard indices can be also represented using set theory. First set consists of pixels of ROI from segmentation result and second set consists of pixels from true ROI. Jaccard index (Fig. 4.8) is defined as intersection over union, which means number of pixels, which belong to both of the sets divided by pixels, which belong only to one the sets (number of well detected pixels over number of badly detected or not detected pixels) [55].

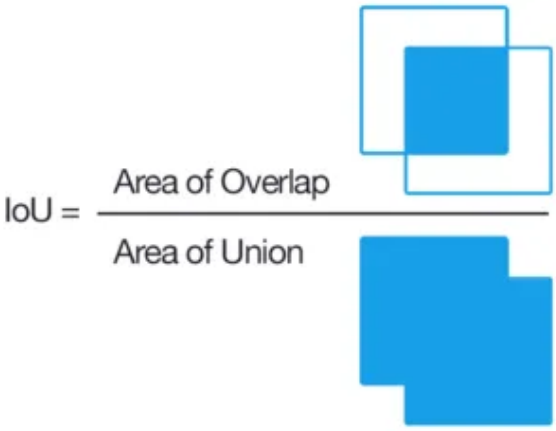


Fig. 4.8: Set representation of Jaccard index [54].

Dice index (Fig. 4.9) can be represented as two times overlap over sum of cardinalities of both sets [56].

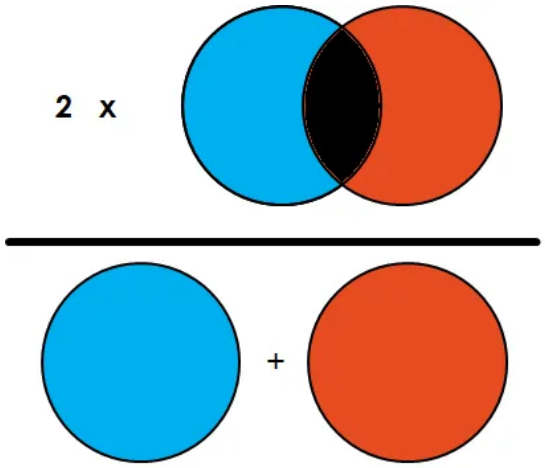


Fig. 4.9: Set representation of Dice index [54].

This method requires accurate mask of pixels, which remain into ROI and background, for each image. Due to absence of these masks, we decided to use simplified method, which just takes in account number of NPs. We classified well detected NPs and NRs as TP, badly detected NPs and NRs as FP (oversegmented or undersegmented) and non-detected NPs and NRs (removed from the selection knowingly or in error) as FN.

4.3.1 Summary of Methods

In summary, non-overlapping particles are detected by the watershed algorithm, and overlapping particles are detected by Hough transform. There is also a Hough transform for ellipse, which may be used for nanorods, but in this project, nanorods are detected only by watershed. After it, it is important to calculate particle properties. For nanoparticles area and diameter is calculated and for nanorods area and major axis and minor axis length are calculated. As segmentation metric are usually used Dice and Jaccard indices, in this project metrics inspired by these indices were used.

5 Results

5.1 TEM Images

TEM samples were prepared and TEM images were acquired. Various shapes and sizes of nanoparticles and nanorods were investigated. There were two samples of nanorods with unknown sizes and four sizes of nanoparticles with diameters of 20 nm, 40 nm, 50 nm, and 80 nm. There are examples of 20 nm nanoparticles (Fig: 5.1), 40 nm nanoparticles (Fig: 5.2), and nanorods (Fig: 5.3) with absorption peak on wavelength 800 nm. 20 nm nanoparticle image (Fig: 5.1) was chosen as sample images for evaluating and for optimization and testing algorithm.

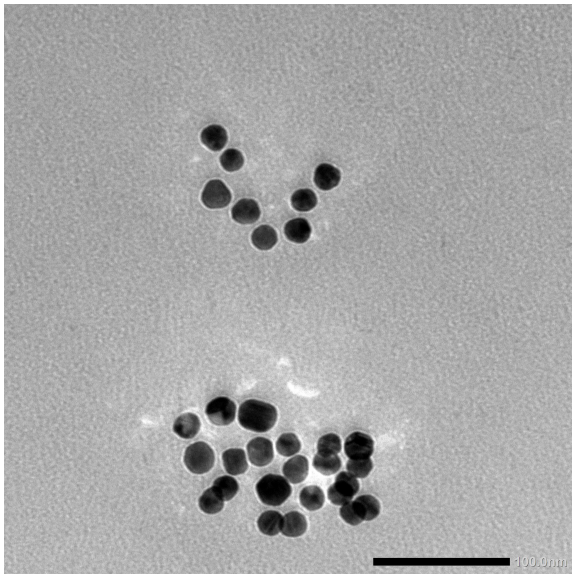


Fig. 5.1: Raw TEM image of 20nm NPs.

5.2 Python Program

Python program was created. The scheme of the program can be seen on Fig. 5.4 The program is able to process more images from the same dataset at once and it can run from the command line with arguments, a user may define a folder with images, specify, if the background should be suppressed or decide if results should be plotted or not. The program requires metadata in JSON format, which includes information about the type of particles (nanoparticles or nanorods) and the scale of the TEM image. The program performs image segmentation and returns labeled images, histograms of all sizes from the dataset and .txt file with mean value calculated using median, interquartile range and information about outliers and detection of small particles called seeds.

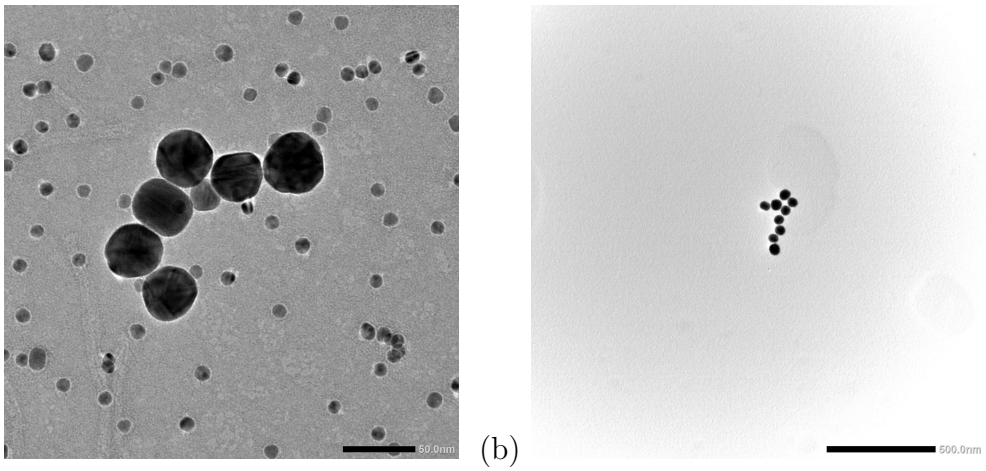


Fig. 5.2: Raw TEM images of 40nm (a) and 5 nm (b) NPs.

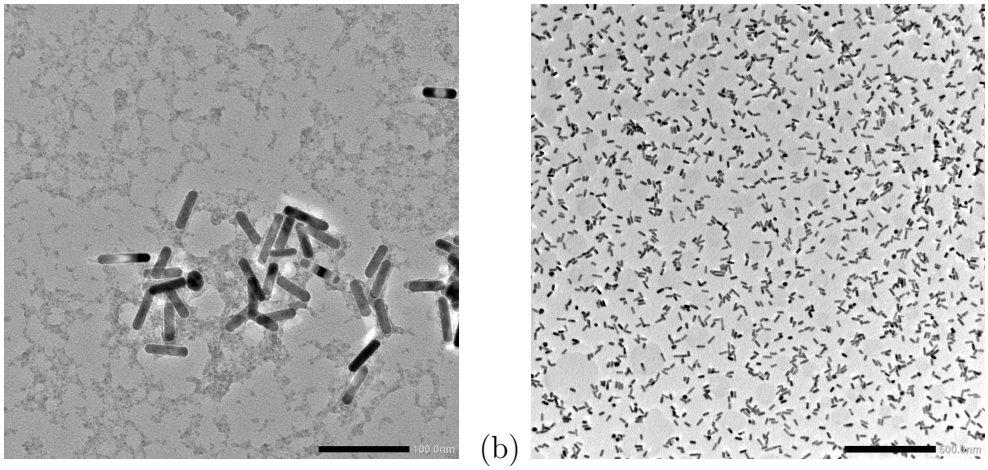


Fig. 5.3: Raw TEM images of NRs.

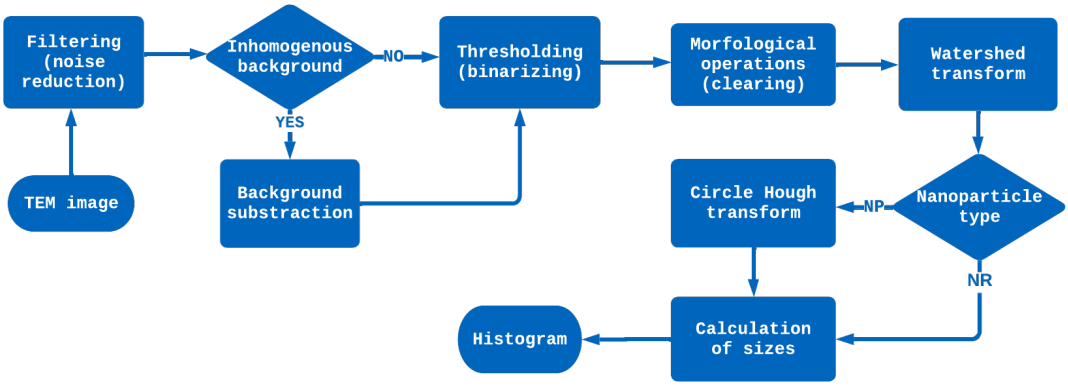


Fig. 5.4: Scheme of algorithm workflow.

The program can be also run using GUI, this option needs no metadata, and everything is set using GUI. It is also possible to run whole dataset, but it is necessary to specify scale for each image. GUI contains four pages, a home page with options, a page with plotted labeled images, a page with plotted histogram, and a page with a description of the program. The home page can be seen in Fig 5.5.

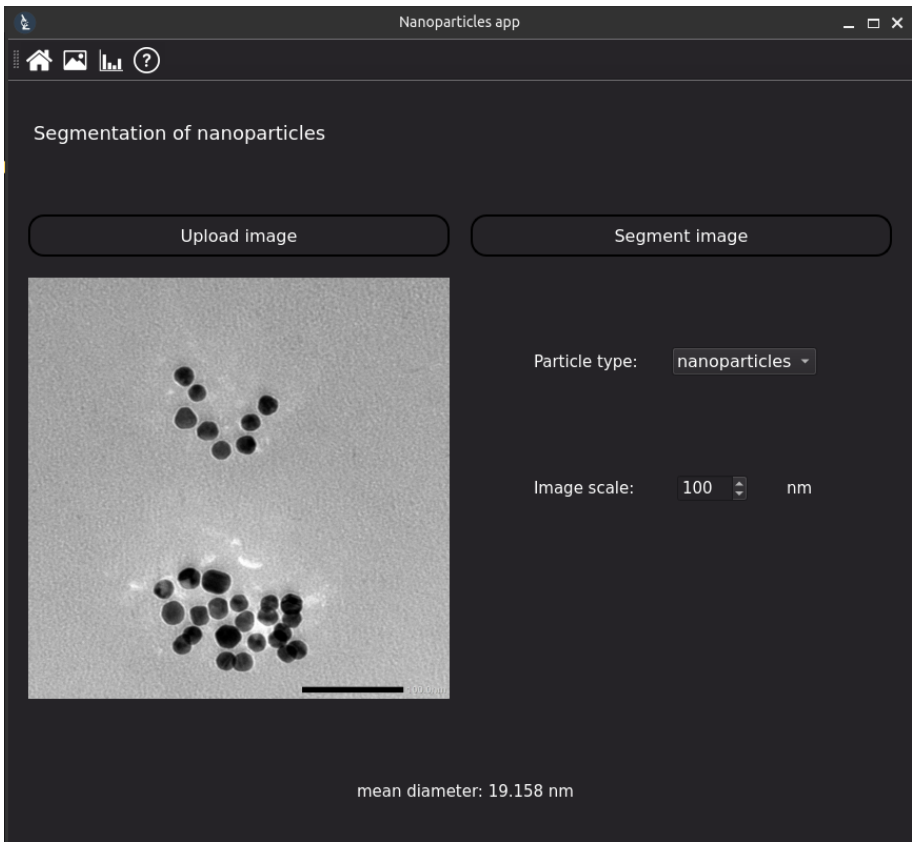


Fig. 5.5: GUI main page look.

There are key functions of this project. One of the most important functions is watershed transform, which can be seen in listing 1. This function takes binary mask as an argument, then performs watershed transform, which consists of experimental implementation of ultimate erosion and watershed transform itself, function returns the labeled image.

```
1 def watershed_transform(binary, np_type, pixel_size):
2     """Labeling image using watershed transform. Seeds for the
    algorithm are determined using morphological erosion.
3
4     Args:
5         binary (numpy.ndarray): binary image
6
7     Returns:
8         numpy.ndarray: labeled image
9     """
```

```
    if np_type == 'nanoparticles':
        seeds = np.zeros(binary.shape, dtype=np.uint8)
        mask = binary.copy()

        while np.sum(mask) > 0:
            regions = erosion(mask)
            regions = label(regions)

            for i in range(1, np.max(regions) + 1):
                roi = regions == i
                convex_hull = convex_hull_image(roi)
                diff = convex_hull != roi
                diff = np.sum(diff.astype(np.float32))
                area = roi.astype(np.float32).sum()
                convex = (diff == 0 or 10 * diff < area) and area > 5 / pixel_size
                if convex:
                    seeds[roi] = 1
                    regions[regions == i] = 0
            mask = regions > 0

        seeds = dilation(seeds)

        distance = ndi.distance_transform_edt(binary)
        img = -distance

    else:
        distance = ndi.distance_transform_edt(binary)
        img = -distance
        mask =  binary
        for _ in range(10):
            mask = remove_small_holes(mask)
            mask = remove_small_objects(mask)
        for _ in range(10):
            mask = erosion(mask)
            seeds = mask

    markers, _ = ndi.label(seeds)
    labels = watershed(img, markers, mask=binary)

    return labels
```

Listing 1: Watershed function

Another important function is the function for CHT, which can be seen in listing 2. The function takes arguments one-channel image, which is filtered by the median filter, then performs CHT, which consists of Canny edge detection and CHT itself, and returns circle parameters, which are coordinates of the center, radius, and

accumulator for each circle in the image. This function is applied to ROIs, where overlapping particles are detected.

```
def hough_segmentation(img, pixel_size, np_type):
    """Segmentation of given ROI using CHT.

    Args:
        img_filtered (numpy.ndarray): grayscale image

    Returns:
        numpy.ndarray, list: RGB image with plotted
                circles/elipses, list of NP parameters
    """
    canny_edge = canny(img, sigma=4)

    start = 10 / pixel_size
    end = 100 / pixel_size
    min_dist = 10/pixel_size
    hough_radii = np.arange(start, end, 1)
    hough_res = hough_circle(canny_edge, hough_radii)

    accums, x, y, r = hough_circle_peaks(
        hough_res, hough_radii, min_xdistance=int(min_dist),
    min_ydistance=int(min_dist)
    )
    x = np.uint16(np.around(x))
    y = np.uint16(np.around(y))
    r = np.uint16(np.around(r))

    a = accums
    l = len(x)
    circles = [(x[i], y[i], r[i]) for i in range(l) if a[i] > 0.2]

    return circles
```

Listing 2: CHT function

Another key function is function, which loads image, because it contains scale bar detection and pixel size calculation. This function consists of loading the image itself, then cropping the image, because only the right corner, which contains the scale bar, is needed. The image is also binarized, small objects are removed and then pixel size is calculated. This function can be seen in listing 3.

```
def loading_img(img_path, scale):
    """Loading image from given file and croping it. Also
    calculating pixel size from image scale bar and given scale.

    Args:
        img_path (str): path to image file
```

```
        scale (int): scale from microscope

    Returns:
        numpy.ndarray: RGB image
        flaot: size of pixel in raw image
    """
    try:
        img_raw = imread(img_path)
    except Exception as err:
        print(err)
        raise Exception("wrong input: image cannot be open")

    width = img_raw.shape[1]

    line = img_raw[-100:, int(width / 2) :, :]
    line = cv2.cvtColor(line, cv2.COLOR_RGB2GRAY)

    mask = line < 10
    mask = remove_small_objects(mask)
    indices = np.where(mask)
    length = max(indices[1]) - min(indices[1])
    global pixel_size
    pixel_size = scale / length

    img_raw = img_raw[0:-100, :]

    return img_raw, pixel_size
```

Listing 3: CHT function

5.3 Program Outputs

The program was tested with various input data, examples of results can be seen in Fig. 5.6, Fig. 5.7, and Fig. 5.8.

Calculated sizes were saved, average area and diameter were calculated for nanoparticles, and average area, major axis length, minor axis length, and aspect ratio were calculated for nanorods. Results of chosen data are presented in tables. For nanoparticles see Tab. 5.1.

Tab 5.1: Example data outputs for NPs.

mean diameter [nm]	interquartile range [nm]	supposed diameter [nm]	Dice index [-]
19.632	0.040	20	0.749
39.039	4.702	40	1.000
51.156	17.678	50	0.799
73.300	18.954	80	8.853

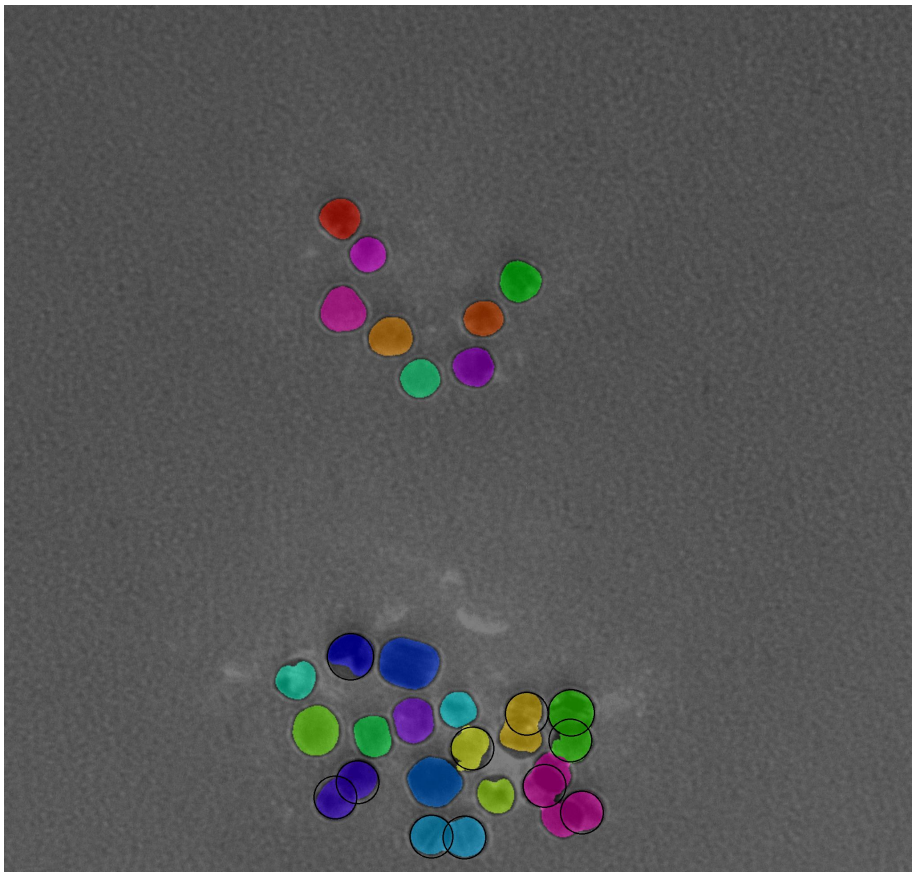


Fig. 5.6: Result labeled image of 20nm NPs.

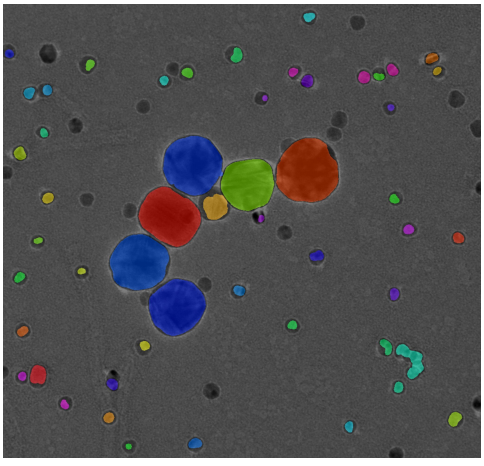
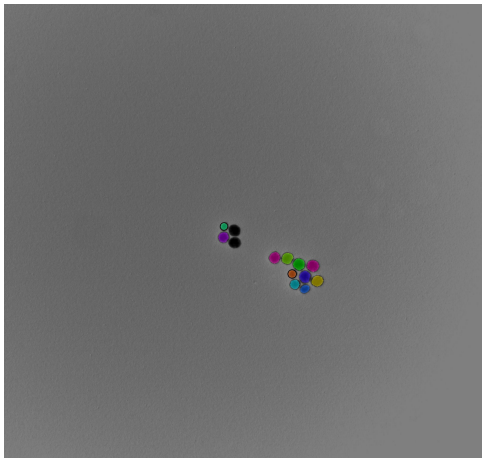
(a)  (b) 

Fig. 5.7: Result labeled images 40nm (a) and 50nm NPs (b).

NRa sample with absorption peak on 800 nm was chosen. Mean minor axis length is 11.044 nm, interquartile range is 0.48 nm, mean major axis is 36.003 nm, interquartile range is 1.566 nm and Dice index is 0.947.

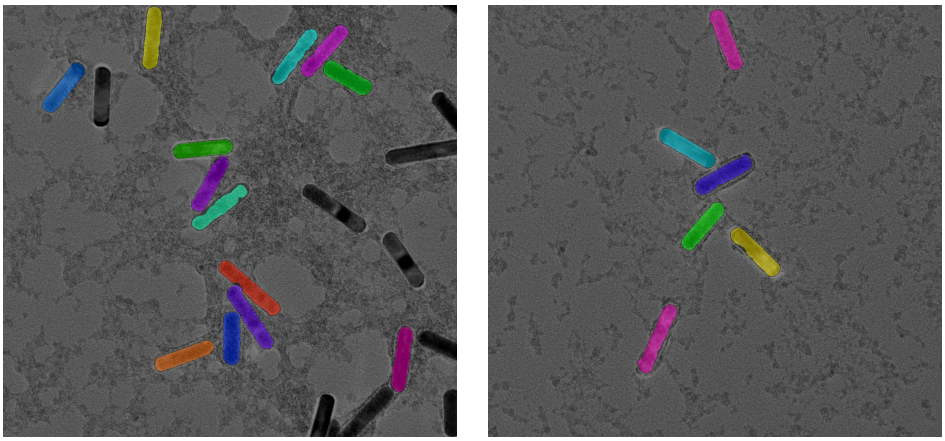


Fig. 5.8: Result labeled image on NRs with absorption peak on 800nm with inhomogeneous background

Histograms of sizes were calculated and plotted, the histogram of nanoparticles shows a layout of diameters in the input images (Fig: 5.9). A histogram of the 40 nm NPs sample can be seen in Fig. 5.9. For comparison of images of the same sample the boxplot was plotted. The boxplot shows differences in mean values and variance between images of the same sample. An example boxplot can be seed in Fig 5.10. Histograms of major and minor axis length in NRs sample can be seen in Fig. 5.11 and Fig. 5.12.

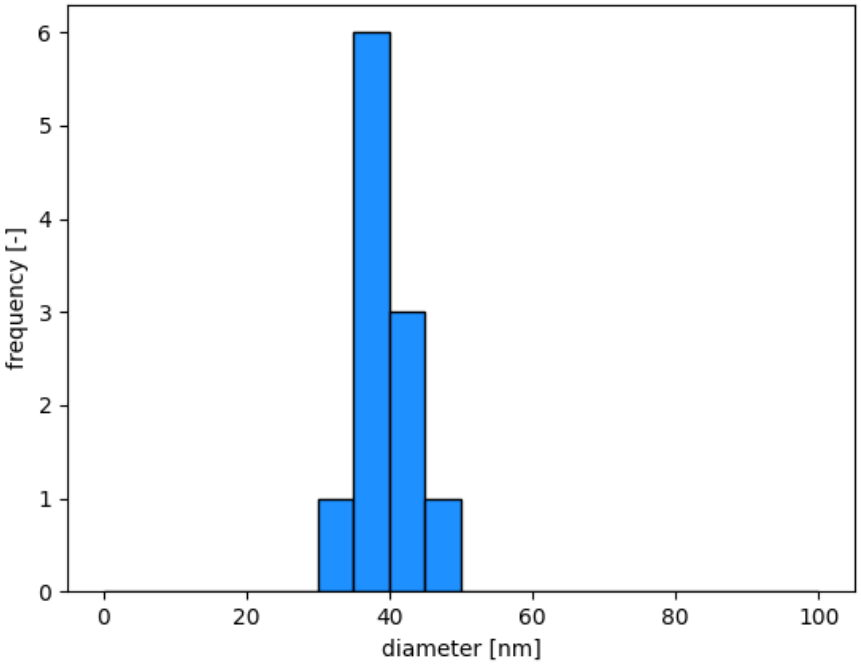


Fig. 5.9: Histogram of sizes of 40 nm NPs.

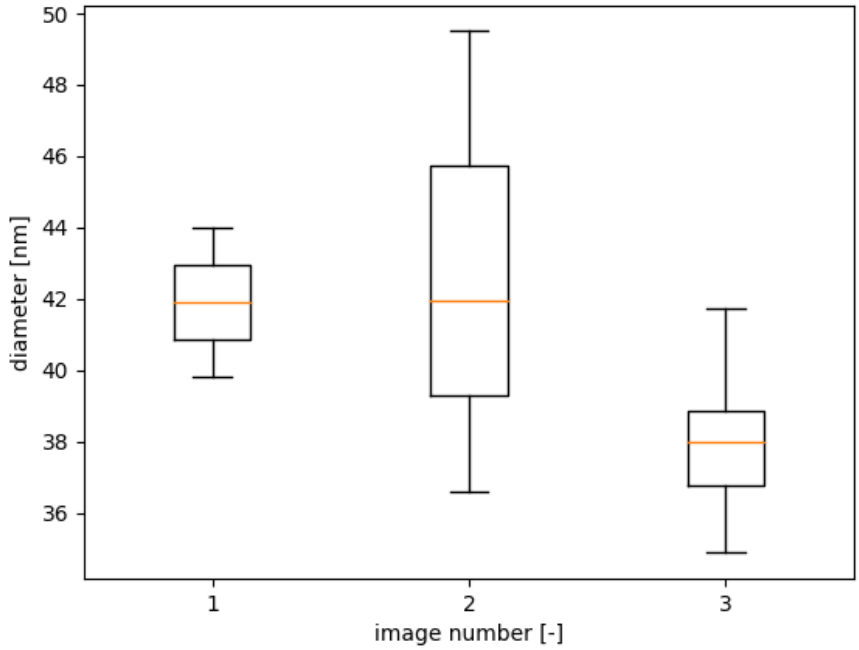


Fig. 5.10: Boxplot comparing sizes of NPs from three different images of 40 nm NPs.

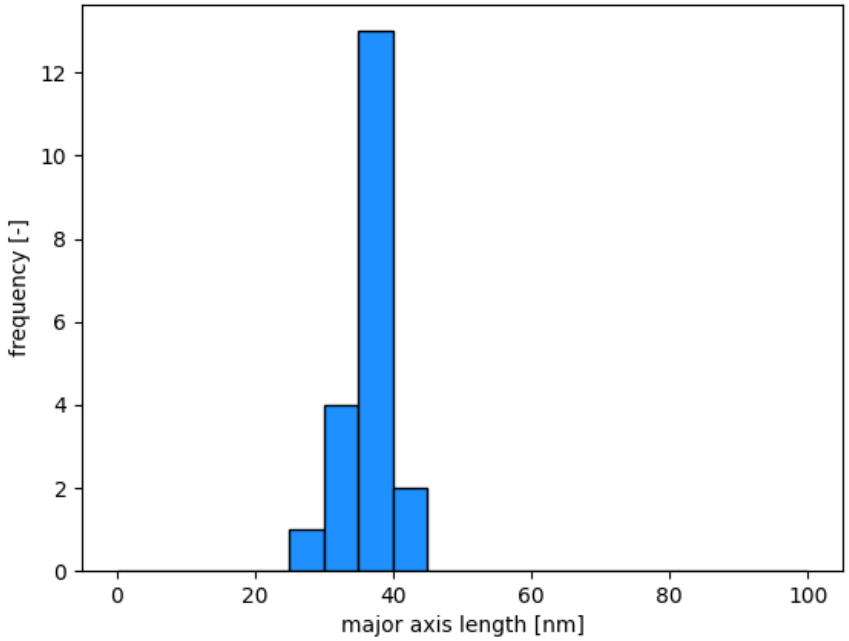


Fig. 5.11: Histogram of major axis lengths in NRs sample with absorption peak on 800 nm.

5.4 Statistical Evaluation of Segmentation Fruitfulness

Segmentation performance was evaluated using method inpired by Dice and Jaccard index. Therse valuse were calculated for each sample and Dice index can be found

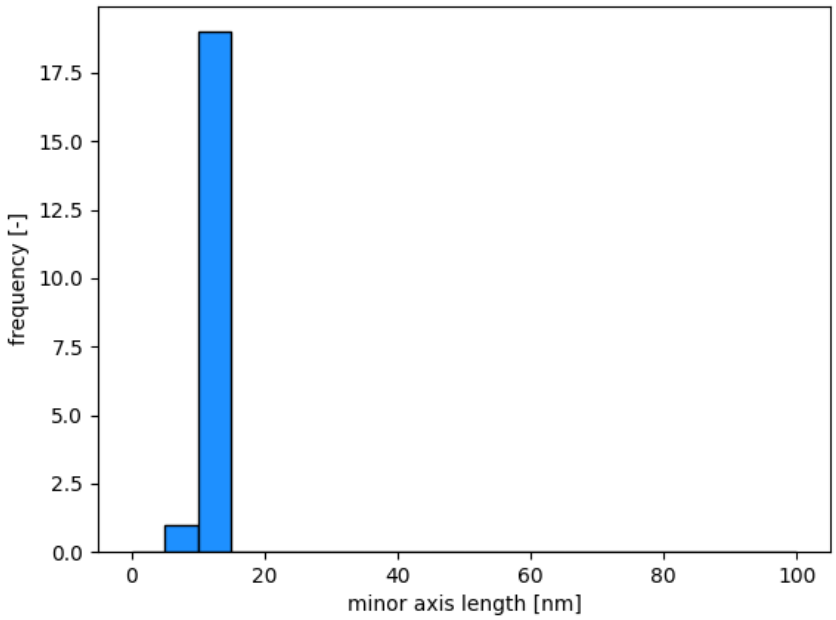


Fig. 5.12: Histogram of minor axis lengths in NRs sample with absorption peak on 800 nm.

in Tab. 5.1. Mean value of Dice index for all the samples is 0.869 and mean Jaccard index is 0.781. Particles were manually sorted into three groups - true positive, where well detected particles were placed in, false positive, where oversegmented or undersegmented particles were placed in and false negative, where particles, which were removed from evaluation, where placed in.

6 Discussion

6.1 Results Evaluation

The main result of this project is Python code for the automatic analysis of TEM images of gold nanoparticles and nanorods. This code was optimized using various input images (real data). During the measurement we found some errors in the data. Evaluation of algorithm performance is described in 6.2 Algorithm Performance. The errors are described in 6.3 Sample Preparation Errors and 6.4 TEM Measurment Errors. The program has some limitations, these are described in 6.5 Limitations.

Two samples were taken out of the selection because of inconvenient quality. First removed sample was sample TJC0090, which contains NRs with absorption peak on 660 nm. Second removed sample was sample DAG3567, which contains 80 nm NPs from first measurement. Images on both samples were all too small because of large scale and were overlapping too much.

6.2 Algorithm Performance

For the performance evaluation used methods inspired by Dice and Jaccard indices were used. Whole nanoparticles were calculated instead of single pixels. The reason is absence of masks of true ROI, which are required for calculation Dice and Jaccard indices. Mean Dice index was 0.869 and Jaccard index 0.744, which are satisfactory values. We treated badly detected particles as FP (false positive) and not detected particles as FN (false negative). These values mean, that there were far more well detected nanoparticles, than badly or not detected.

Algorithm performs with some error, which depends on quality of the data. Values, which are far bigger than median value, were classified as outliers. These outliers can be also seen in histogram. We have chosen to use median for value and interquartile range, because these parameters are outliers resistant.

6.3 Sample Preparation Errors

The goals of the project were to prepare TEM samples on copper gird and acquire TEM images. Samples of nanoparticles and nanorods were prepared, but during TEM measurement there were found other small particles on some samples. These small particles were probably caused by imperfect preparation of the NPs solution. See fig. 6.1. The algorithm detects these seeds using k-mean clustering, inform the user and removes the seeds in evaluation.

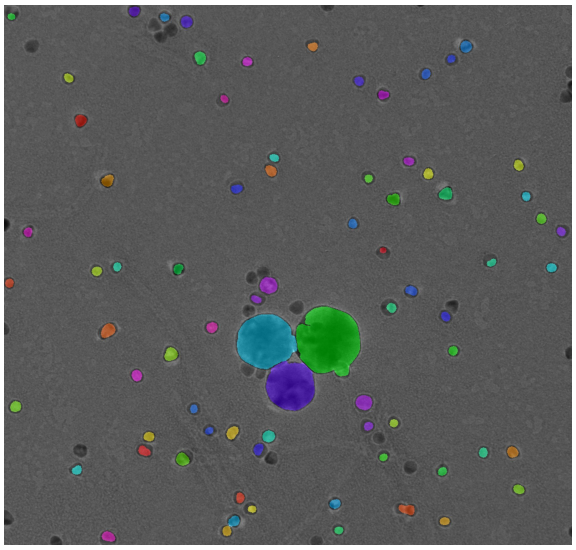


Fig. 6.1: Detected small particles.

Another source of segmentation errors may be insufficiently dried samples. Samples of nanorods with an absorption peak of 800 nm can be seen in image 6.2, this image has inhomogeneous background due to insufficiently dried TEM grids. This is an error which may happen more frequently so it is treated in software, however, it declines the quality of the image. A wet sample can be seen in figure 6.2.

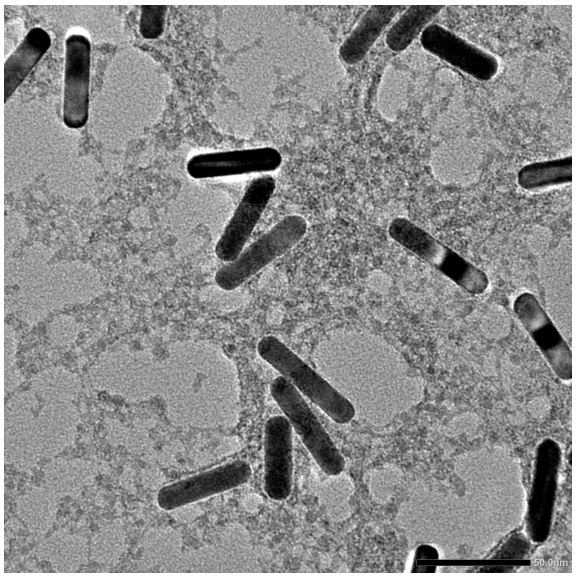


Fig. 6.2: Inhomomogeneous background caused by wet sample.

6.4 TEM Measurment Errors

Another source of potential errors can be the quality of TEM images. There are several issues which affect the results of the image analysis program. The first issue is resolution. Microscope computer saves images in very high resolution of 2044 x 2044 pixels and due to the computational complexity of most functions, image resolution has to be reduced. For most images rescaling does not matter, but there are images where particles are very tiny in comparison with image size. After rescaling, these particles have a diameter for example 8 or 9 pixels, so the accuracy of measurement extremely decreases. A solution for this problem could be manually cropping these images before automatic analysis, but there is an issue. Images contain a scale bar, which is located in corner of the image and is used to determine pixel size, so cropping the image would remove this information.

Another issue with TEM image quality can be contrasted. The algorithm for particle analysis works on the principle of subtracting particles from the background based on pixel intensity. Particles in some images contain stains with a higher intensity which is similar to or even higher than the background intensity. So the algorithm assigns this pixel to the background and not to particles. This issue is partly treated in code by using morphological operations for filling holes. This solution can solve smaller bright spots but it cannot do anything with cases where for example half of the particle is bright. Some TEM images have low contrast overall so thresholding methods may not be as accurate as they could be. An example of low-contrast nanorods with even bright spots can be seen in figure 6.3 (a) in chapter Results. In this image wet background is also a problem, so it decreases the quality of the image even more. Bigger bright spots may cause over-segmentation of the image. Oversegmented NRs had to be removed from the evaluation using erosion. 6.3.

The last issue which is related to image quality and occurred in sample images are badly focused image. It is problem especially in images with a large number of small particles because particles in samples are overlapping and in this case, it is almost impossible to divide them well. An example of this case can be seen in figure 6.4.

When talking about overlapping particles there is also one issue with this, which cannot be solved. Some samples contain places with a lot of particles laying on each other. In some extreme cases, there is not even possible to see individual particles. These images obviously cannot be used for analysis. The program detects these huge clusters as one particle and it would bring enormous error in results so the program has treated these cases by removing such huge areas which were detected as one particle. An example of an image with this case can be seen in figure 6.5.

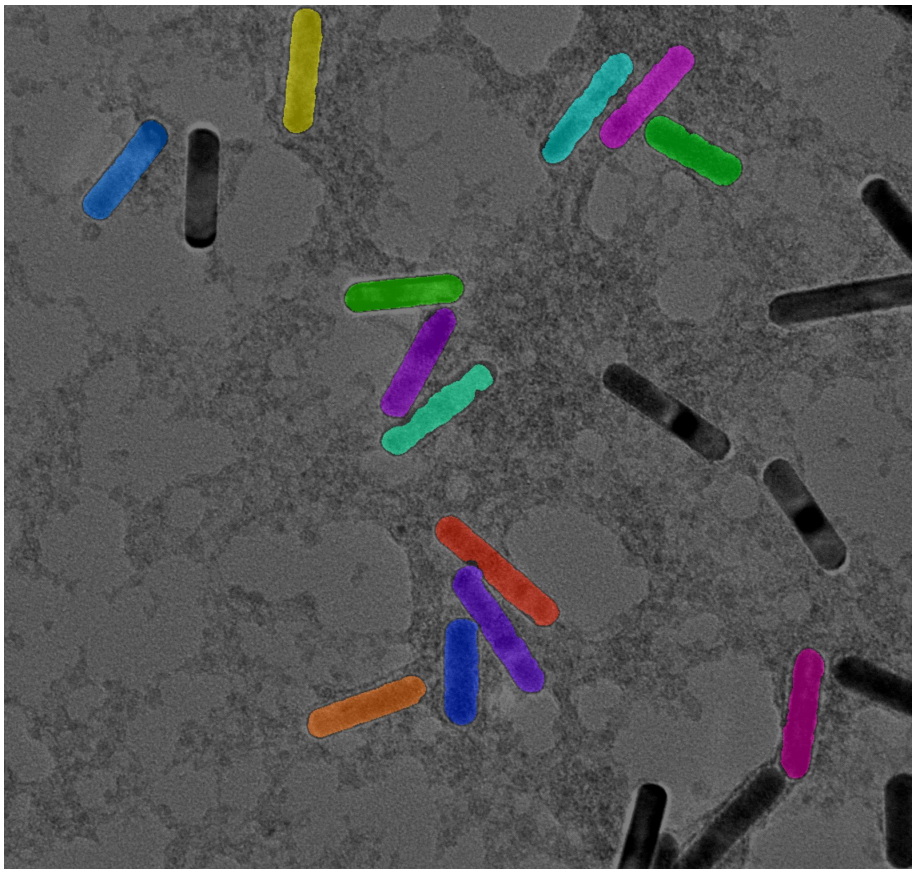


Fig. 6.3: Removed nanorods due to the oversegmentation.

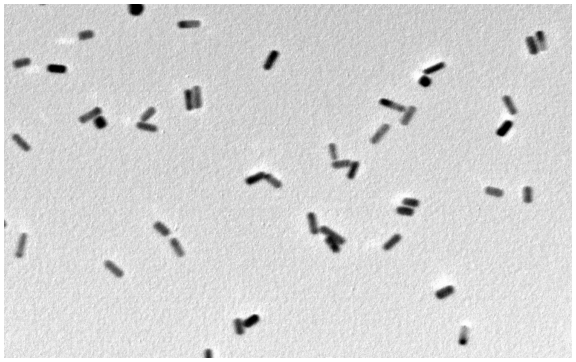


Fig. 6.4: TEM micrograph with insufficient resolution.

6.5 Limitations

First limitation is in type of NPs, for which the program can be used. Each type of NPs are totally different and needs specific approach, so this program can be used just for evaluating TEM images of AuNPs and GNRs, because the algorithm was optimized using AuNPs and GNRs images and it means it is feasible only for this

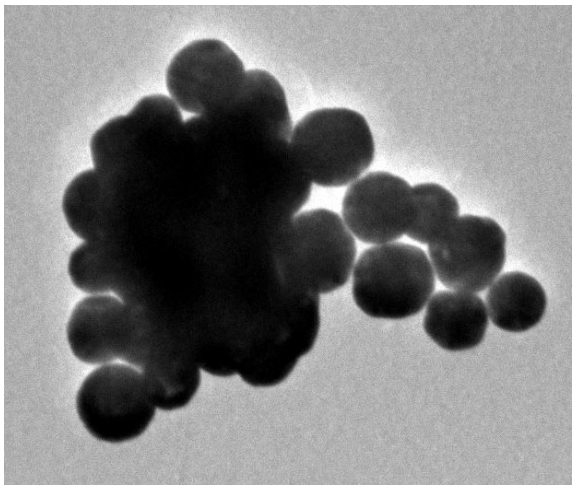


Fig. 6.5: Image with huge cluster of particles.

type of NPs. Due to this limitation it is in the pipeline to create modular program, where parameters and chosen methods could be changed by user to fit the data.

Second limitation is in size of NPs and NRs and TEM image scale. The program takes in account sizes found between 1 nm and 100 nm according to definition of nanotechnology [6]. The TEM image scale is limited, because the algorithm was optimized and tested on images with scale bar from 20 nm into 500 nm, so there is no guarantee it would work for another scale bars. The 500 nm scale bar is frontier, because with this scale NPs are yet too small, so the accuracy of evaluation can be decreased in comparison with images with scale bar about 100 nm. This problem was described better in 6.4 TEM Measurment Errors.

Other limitations are about quality of sample and TEM image. The algorithm cannot be used for bad focused images or images, where individual particles cannot be seen (it may occur, when NPs are overlapping too much). The limitation can be also, that algorithm cannot detect particles, which contain bright spots in it, so these are removed from the evaluation. These limitation were discussed in 6.4 TEM Measurment Errors. The algorithm can handle with some of sample preparation errors as wet sample or small exotic NPs, which are not part of the measured NPs. These problems were discussed in 6.3 Sample Preparation Errors.

7 Conclusion

Project included several goals. which were achieved, presented in the chapter Results and discussed in the chapter Discussion.

- First goal was to prepare TEM samples. Five AuNPs samples and two GNRs samples were prepared. NPs have sizes 20 nm, 40 nm, 50 nm and 80 nm and NRs samples have absorption peaks on 660 nm and 800 nm.
- Second goal was to acquire TEM images. It was done by microscopy technician. We obtained multiple images of each sample.
- Another goals were about image analysis and the program. Watershed transform was suggested as main segmentation algorithm.
- Python program for automatic evaluation of sizes and shapes of nanoparticles was created and the main functions are presented in the chapter Results.
- Using HT for segmentation of overlapping NPs was tested and used in the program. The limitation of this method was discussed in the chapter Discussion.
- Segmentation was performed on various real data images, NRs images had jagged background. Example of result images can be seen in the chapter Results.
- Pixel size is automatically evaluated in loading function in the program using scale bar in the corner of the TEM image. This function can be seen in the chapter Results.
- Sizes and shapes of the NPs and NRs were evaluated and are presented using histogram of sizes and boxplot for comparison of particular images of the same sample. Diameters for NPs and major and minor axis length and aspect ratio for NRs were calculated. The mean value (using median) and interquartile range were also determined.
- Metrics inspired by Dice and Jaccard indices were used as statistical parameter for segmentation evaluation. Average Dice index inspired metric result was 0.869 and Average Jaccard index inspired metric result was 0.781.
- Limitation of the algorithm were discussed in the chapter Discussion. The main limitations are, that the program can be used just for evaluation AuNRs and GNRs with sizes between 1 and 100 nm on images with scale bars up to 500 nm.The quality of TEM images has also great impact on the results.
- GUI was created and is presented in the chapter Results.

Bibliography

1. PETERSEN, Elijah J; BUSTOS, Antonio R Montoro; TOMAN, Blaza; JOHNSON, Monique E; ELLEFSON, Mark; CACERES, George C; NEUER, Anna Lena; CHAN, Qilin; KEMLING, Jonathan W; MADER, Brian et al. Determining what really counts: modeling and measuring nanoparticle number concentrations. *Environmental Science: Nano*. 2019, roč. 6, č. 9, s. 2876–2896.
2. SHANG, Jing; GAO, Xiaohu. Nanoparticle counting: towards accurate determination of the molar concentration. *Chemical Society Reviews*. 2014, roč. 43, č. 21, s. 7267–7278.
3. KHLEBTSOV, Nikolai G. Determination of size and concentration of gold nanoparticles from extinction spectra. *Analytical chemistry*. 2008, roč. 80, č. 17, s. 6620–6625.
4. CHEN, Huanjun; SHAO, Lei; LI, Qian; WANG, Jianfang. Gold nanorods and their plasmonic properties. *Chemical Society Reviews*. 2013, roč. 42, č. 7, s. 2679–2724.
5. NOUAILHAT, Alain. *An introduction to nanoscience and nanotechnology*. Sv. 10. John Wiley & Sons, 2010.
6. KOLÁŘOVÁ, Lucie. *Úvod do nanovědy a nanotechnologií*. Univerzita Palackého, 2014.
7. HORNYAK, Gabor L; TIBBALS, Harry F; DUTTA, Joydeep; MOORE, John J. *Introduction to nanoscience and nanotechnology*. CRC press, 2008.
8. HOŠEK, Jan. *Úvod do nanotechnologie*. České vysoké učení technické, 2010.
9. *Introduction to Nanotechnology: Basic Concepts Explained* [online]. [cit. 2019-08-03]. Dostupné z: `https://www.technology.org/2019/08/03/introduction-to-nanotechnology-basic-concepts-explained/`.
10. PETERSEN, Nils O. *Foundations for nanoscience and nanotechnology*. CRC Press, 2017.
11. BINNS, Chris. *Introduction to nanoscience and nanotechnology*. John Wiley & Sons, 2021.
12. LONG, Yi-Tao; JING, Chao. *Localized surface plasmon resonance based nanobiosensors*. Springer, 2014.
13. KHLEBTSOV, Nikolai G; DYKMAN, Lev A. Optical properties and biomedical applications of plasmonic nanoparticles. *Journal of Quantitative Spectroscopy and Radiative Transfer*. 2010, roč. 111, č. 1, s. 1–35.

14. CHEN, Tianhong; POURMAND, Mahshid; FEIZPOUR, Amin; CUSHMAN, Bradford; REINHARD, Björn M. Tailoring plasmon coupling in self-assembled one-dimensional Au nanoparticle chains through simultaneous control of size and gap separation. *The journal of physical chemistry letters*. 2013, roč. 4, č. 13, s. 2147–2152.

15. PRAMANIK, Avijit; GAO, Ye; PATIBANDLA, Shamily; MITRA, Dipanwita; MCCANDLESS, Martin G; FASSERO, Lauren A; GATES, Kalein; TANDON, Ritesh; RAY, Paresh Chandra. The rapid diagnosis and effective inhibition of coronavirus using spike antibody attached gold nanoparticles. *Nanoscale Advances*. 2021, roč. 3, č. 6, s. 1588–1596.

16. *ResearchGate*. 2008-2022. Dostupné také z: `https://www.researchgate.net/`.

17. PREIM, Bernhard; BOTHA, Charl P. *Visual computing for medicine: theory, algorithms, and applications*. Newnes, 2013.

18. KUMAR, Rahul; BINETTI, Leonardo; NGUYEN, T Hien; ALWIS, Lourdes SM; AGRAWAL, Arti; SUN, Tong; GRATTAN, Kenneth TV. Determination of the Aspect-ratio Distribution of Gold Nanorods in a Colloidal Solution using UV-visible absorption spectroscopy. *Scientific reports*. 2019, roč. 9, č. 1, s. 1–10.

19. ELAHI, Narges; KAMALI, Mehdi; BAGHERSAD, Mohammad Hadi. Recent biomedical applications of gold nanoparticles: A review. *Talanta*. 2018, roč. 184, s. 537–556.

20. ANKER, Jeffrey N; HALL, W Paige; LYANDRES, Olga; SHAH, Nilam C; ZHAO, Jing; VAN DUYNE, Richard P. Biosensing with plasmonic nanosensors. *Nature materials*. 2008, roč. 7, č. 6, s. 442–453.

21. JOY, David C. *transmission electron microscope* [online]. [cit. 2022-10-10]. Dostupné z: `https://www.britannica.com/technology/transmission-electron-microscope`.

22. CHEN, Huanjun; SHAO, Lei; LI, Qian; WANG, Jianfang. Gold nanorods and their plasmonic properties. *Chemical Society Reviews*. 2013, roč. 42, č. 7, s. 2679–2724.

23. CAO, Jie; SUN, Tong; GRATTAN, Kenneth TV. Gold nanorod-based localized surface plasmon resonance biosensors: A review. *Sensors and actuators B: Chemical*. 2014, roč. 195, s. 332–351.

24. *Nano Nano* [online]. [cit. 2014-07-15]. Dostupné z: `https://xsigsummer.wordpress.com/2014/07/15/nano-nano/`.

25. AMENDOLA, Vincenzo; MENEGHETTI, Moreno. Size evaluation of gold nanoparticles by UV- vis spectroscopy. *The Journal of Physical Chemistry C*. 2009, roč. 113, č. 11, s. 4277–4285.

26. *MathWorks*. 1994-2022. Dostupné také z: `https://www.mathworks.com`.

27. MAHMOUDI, Linda; EL ZAART, Ali. A survey of entropy image thresholding techniques. In: *2012 2nd international conference on advances in computational tools for engineering applications (ACTEA)*. IEEE, 2012, s. 204–209.

28. SENTHILKUMARAN, N; RAJESH, Reghunadhan. Image segmentation-a survey of soft computing approaches. In: *2009 International Conference on Advances in Recent Technologies in Communication and Computing*. IEEE, 2009, s. 844–846.

29. MALIK, Mateusz; SPUREK, Przemysław; TABOR, Jacek. Cross-entropy based image thresholding. *Schedae Informaticae*. 2015, roč. 24.

30. ARORA, Siddharth; ACHARYA, Jayadev; VERMA, Amit; PANIGRAHI, Prasanta K. Multilevel thresholding for image segmentation through a fast statistical recursive algorithm. *Pattern Recognition Letters*. 2008, roč. 29, č. 2, s. 119–125.

31. PARK, Chiwoo; HUANG, Jianhua Z; JI, Jim X; DING, Yu. Segmentation, inference and classification of partially overlapping nanoparticles. *IEEE transactions on pattern analysis and machine intelligence*. 2012, roč. 35, č. 3, s. 1–1.

32. PATIL, Dinesh D; DEORE, Sonal G. Medical image segmentation: a review. *International Journal of Computer Science and Mobile Computing*. 2013, roč. 2, č. 1, s. 22–27.

33. YUHENG, Song; HAO, Yan. Image segmentation algorithms overview. *arXiv preprint arXiv:1707.02051*. 2017.

34. YESILYURT, Emre. *What is the Difference Between Hierarchical and Partitional Clustering* [online]. [cit. 2022-03-26]. Dostupné z: `https://www.cs.auckland.ac.nz/courses/compsci773s1c/lectures/ImageProcessing-html/topic1.htm`.

35. WANG, Xin. Laplacian operator-based edge detectors. *IEEE transactions on pattern analysis and machine intelligence*. 2007, roč. 29, č. 5, s. 886–890.

36. SHARMA, Priyansh; SUJI, Jenkin. A review on image segmentation with its clustering techniques. *International Journal of Signal Processing, Image Processing and Pattern Recognition*. 2016, roč. 9, č. 5, s. 209–218.

37. YESILYURT, Emre. *What is the Difference Between Hierarchical and Partitional Clustering* [online]. [cit. 2022-03-26]. Dostupné z: https://medium.com/machine-learning-t%5C%C3%5C%BCrkiye/what-is-the-difference-between-hierarchical-and-partitional-clustering%2084d5a4ceb4c0.

38. *Spatial Frequency Domain*. Dostupné také z: http://dendroid.sk/2011/05/09/k-means-clustering/.

39. *Data Clustering Algorithms* [online]. [cit. 2020-08-30]. Dostupné z: https://sites.google.com/site/dataclusteringalgorithms/home?authuser=0.

40. AYDILEK, Ibrahim Berkan; ARSLAN, Ahmet. A hybrid method for imputation of missing values using optimized fuzzy c-means with support vector regression and a genetic algorithm. *Information Sciences*. 2013, roč. 233, s. 25–35.

41. KRAJČA, Vladimír; MOHYLOVÁ, Jitka. *Číslicové zpracování neurofiziologických signálů*. České vysoké učení technické, 2011.

42. SEO, Hyunseok; BADIEI KHUZANI, Masoud; VASUDEVAN, Varun; HUANG, Charles; REN, Hongyi; XIAO, Ruoxiu; JIA, Xiao; XING, Lei. Machine learning techniques for biomedical image segmentation: an overview of technical aspects and introduction to state-of-art applications. *Medical physics*. 2020, roč. 47, č. 5, e148–e167.

43. MISHRA, Mayank. *Convolutional Neural Networks, Explained* [online]. [cit. 2020-08-26]. Dostupné z: https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939.

44. *Science services*.

45. *NANOCARBON GROUP*. Dostupné také z: https://www.nanocarbon.cz/facilities/.

46. *Median filter*. Dostupné také z: https://neubias.github.io/training-resources/median_filter/index.html.

47. MUTHUKRISHNAN. *Otsu's method for image thresholding exmplained and implemented*. Dostupné také z: https://muthu.co/otsus-method-for-image-thresholding-explained-and-implemented/.

48. *Scikit image documantation*. Dostupné také z: https://scikit-image.org/docs/stable/.

49. *rahnampress*. Dostupné také z: https://shop65002.rahnamapress.net/category?name=watershed%5C%20transform.

50. *Distance transform*. 2003. Dostupné také z: https://homepages.inf.ed.ac.uk/rbf/HIPR2/distance.html.

51. *Hough circle transform*. Dostupné také z: https://theailearner.com/tag/hough-circle-transform-algorithm/.

52. *math is fun*. Dostupné také z: https://www.mathsisfun.com/algebra/circle-equations.html.

53. *Canny edge Detection step by step in Python - Computer Vision*. Dostupné také z: https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d8123.

54. *Metrics to Evaluate your Semantic Segmentation Model*. Dostupné také z: https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2.

55. *Jaccard Index*. Dostupné také z: https://deepai.org/machine-learning-glossary-and-terms/jaccard-index.

56. *F1 Score in Machine Learning: Intro Calculation*. Dostupné také z: https://www.v7labs.com/blog/f1-score-guide.