BACHELOR

Unsupervised learning for SEM-images of defects by particles on the wafer
The clustering of particles on the wafer based on their morphological features

Fölker, Ilse

Award date:
2020

2WH40 Bachelor final project

Unsupervised learning for SEM-images of defects by particles on the wafer

The clustering of particles on the wafer based on their morphological features.

Student:

Ilse Fölker
0953386
i.folker@student.tue.nl

Supervisors:

Robin de Wit
robin.de.wit@asml.com

Marta Regis
m.regis@tue.nl

Edwin van den Heuvel
e.r.v.d.heuvel@tue.nl

Eindhoven, October 5, 2020

Contents

1 Introduction

From an early stage in life, we learn to detect, identify and compare shapes that we find in everyday life. We learn to identify animals or read handwritten notes, because we learn what to look for and we use past experiences. Detecting, identifying and comparing shapes is not only useful in everyday life situations, but plays an important part in science as well. For example in the medical field identifying irregular blood cell shapes in a blood smear can lead to detecting a certain disease. There are many more applications where examining the shape of a subject has a certain benefit, such as identifying plant types and face recognition, and with the rise of the amount of data the question comes to surface if we can let machines learn these steps i.e. machine learning. Machine learning is the area of computational science that focuses on analyzing and interpreting patterns and structures in data to enable learning, reasoning, and decision making outside of human interaction. What makes machine learning so interesting is that it replaces the labor intensive and expensive tasks done by humans, while also eliminating human error.
Machine learning is a large and complicated field with many applications and options. In this study we will only see the tip of the iceberg, nonetheless this is a real-life example of a study that can benefit from machine learning and its applications. One of the primary learning models is unsupervised learning, which we will consider in this study. In unsupervised learning, the machine is provided with a set of data and is not provided with any labels for it. Given the huge amount of data, the machine may identify trends of similarity. The algorithm will then identify clusters or groups with similar properties [26].
In this section a case study will be addressed that can benefit from an implementation of machine learning. In section 2 the problem description is presented with the research questions and the scope of the study. Afterwards in section 3, we gather source references from the literature that are closely related to the case study. This will be followed by a more in-depth explanation and gathering of background information in section 4, that is needed to set up a method for the case study. Then in section 5 we illustrate chosen methods based on the background information in section 4 and how they will be applied to the case study at hand. Results are then visualized and discussed in section 6. Furthermore in section 7 the case study will be shortly summarized and the most important points will be highlighted. Lastly section 7 will include some recommendations for further research.

1.1 Case study

1.1.1 About ASML

ASML is an innovation leader in the semiconductor industry. They provide chip manufacturers with everything they need – hardware, software and services – to mass produce patterns on silicon through lithography. ASML is also the only producer of lithography machines that uses extreme ultraviolet light (EUV), i.e. the EUV system [1]. The lithography system is a projection system that projects light through a blueprint in order to print a pattern onto a photosensitive silicon wafer. The wafer is a substrate that is essential to the chip manufacturing [2].

1.1.2 EUV System Particle Contamination Department

During the process particles can contaminate the wafer, which can result into the damaging of printed chips. Since the contamination with particles can have a negative impact on product quality, ASML has dedicated departments to resolve and minimize the particle contamination.
The EUV System Particle Contamination Department focuses on the particle contamination of the EUV system and there is another department that focuses on the DUV (deep ultraviolet) lithography system. This case study is limited to the EUV system only. In order to understand the defects found on the wafer, the properties of particles must be obtained. After the initial wafer is passed through the EUV system for the chip printing process, the exposed wafer is analysed by a couple of different methods. One of these methods is the scanning electron microscope (SEM) review, which is a type of electron microscope that scans the surface of the particle with a focused beam of electrons in order to obtain an image. The SEM review outputs multiple types of SEM-images of the observed particles on the wafer. This data will be the subject of the current case study.

1.1.3 Shape analysis of the particles contaminated on the wafer

Once the SEM-images are obtained from the contaminated particles on the wafer, these images, together with the results from other methods, can be analyzed in order to find the root cause of contamination. In this case study we will study the SEM-images of the particles, in order to distinguish different types based on their morphological features. The data considered in this case study consists of several types of SEM-images for each of the particles on a wafer from various EUV systems. Only one particle is visualized in each of the SEM-images and the data does not have any labels for the indicated particle. Additionally the SEM analysis sometimes (partly) misses the particle such that no particle or only half is visible in the SEM-image. These images are also included in the data and need to be dealt with. An example of a 'missed' image is visible in Figure 2.

Two types of SEM-images for each particle are addressed in this case study as a starting point. For related studies, other types can be used. The first type is the 'Internal' image, which depicts a large contrast between the background and the foreground object shown in Figure 1 a. The other type of images are the 'Topography' images, which consists of four subcategories: 'Topography 1', 'Topography 2', 'Topography 3'and 'Topography 4'. The distinction between the four subcategories of the 'Topography' images is the beam position that enlightens the particle. Two subcategories of the 'Topography' type with various beam position are visible in Figure 1 b and c. Using all subcategories of the 'Topography' images a new type is created, of which an example is visible in Figure 1 d, by calculating the mean value of each pixel of the four 'Topography' images. What makes this new type notable is that the pixel values within the border of the particle show little difference and it is also the lightest color in the image, i.e. highest pixel values.

From these images the goal of this study is to observe the particle in the image and group particles that share a certain similarity, i.e. clustering the images. As there are many clustering methods, we will explore only two, which will be discussed further in section 2.

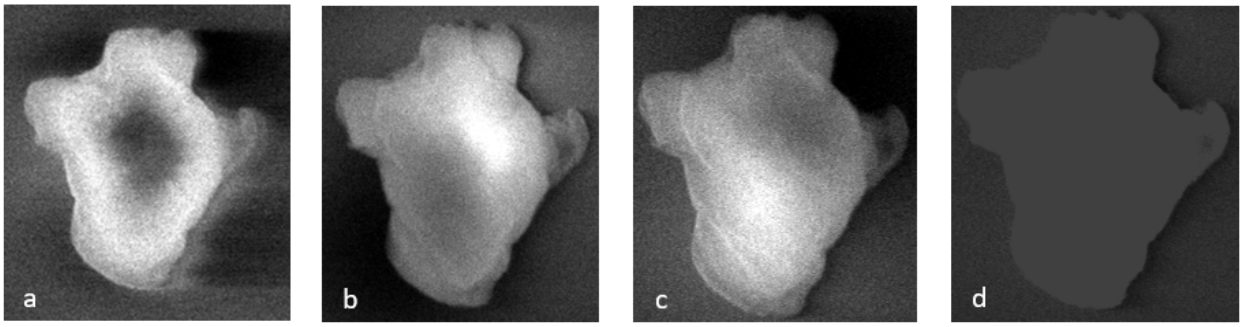


Figure 1: a) An 'Internal' type SEM-image b) An image with the beam positioned in the upper right corner. b) An image with the beam positioned in the lower left corner. d) obtained image with mean values of all four beam position images

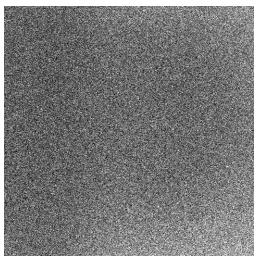


Figure 2: A 'missed' image

2 Problem Description

2.1 Main question

The main question that this study aims to answer is:

> *What are the advantages and disadvantages of two clustering methods with different cost-functions and how can this be evaluated?*

Images in the study do not have a label of which class they should belong to. Thus we will approach the problem via unsupervised learning. Furthermore the morphological features of the image have yet to be defined, as well as the preliminary processing of the images before clustering can take place. Preliminary processing includes, but is not limited to, selecting images of interest and image segmentation within the selected images to detect the region of interest. This is done because (parts off) the raw data might not be usable for this study. In order to answer the main question this project has been divided in several sub-questions.

2.2 Sub-questions

1. *Which part of the raw data is usable for this study and how can this be evaluated?*
2. *What is the best method to segment the particle in the image from the background?*
3. *What are the most interesting morphological features of the segmented particle?*
4. *What are the best suited clusterings according to various evaluation methods?*

2.3 Scope

The current study is of exploratory nature, which means that it is not build on previous work and might require further research before concrete implementations can take place. The scope of the current study is quite extensive as it contains multiple steps that are already very elaborate on their own. Considering the limited time, each step within the study will consider a limited amount of methods or morphological features. Based on related works, the most promising methods and morphological features will be considered and evaluated. However for further research many more fields can be explored for each of the steps, on which we will elaborate in section 7.

3 Related research

The case study is inspired by the work of Panuju et al. [30] who applied supervised learning to distinguish diverse defects, including but not limited to particle contamination, on the wafers of ASML's deep ultraviolet (DUV) lithography systems [30]. This research proposed to use machine learning to analyze wafer defects, since it showed desired results. However, supervised learning was the proposed method in the research by Panuju et al. [30] since labels were given for the data. This differs from the current case study, consequently related studies in other fields were explored. No studies were found with identical data and method, so closely related studies were viewed.

3.1 Tumor spheroids

With rising interest in machine learning over the course of the years, there are many fields in which machine learning is applied. In order to be able to answer the main question, it is helpful to look for studies that have a similar data set and goal. An example is provided by [5], in which unsupervised learning is proposed to classify tumor spheroids (i.e. artificial cancer cells that form multi-cellular structures) based on their morphology, into "biologically meaningful" groups. Like in our case study, the data they consider is mostly that the data considered is mostly spherical and the study aims to detect the irregularities, and cluster them into different groups.

3.2 Leukemia

As the article mentioned above uses a quite novel way of describing shape, it is important to look at similar papers that use more common descriptors of shape. Another medical field where the shape and size of particles in the image data is of vital importance, is the hematology department. Morphological analysis of the blood cells provide information relevant for blood diseases like leukemia [19]. In the work done by Mohapatra et al [2015] morphological features are used to describe the shape of the nucleus of lymphocytes and lymphoblasts in order to distinguish leukemic and healthy cells [23]. Again, with promising results, the data included in the study is quite similar to the data that is used in the current case study.

3.3 Current image Segmentation method

In order to analyze the shape of a particle detected in SEM-images, it is important to define the area within the image wherein the particle is contained. This process is called image segmentation. There exists already an algorithm, provided by ASML, that deals with image segmentation. The algorithm can be found in appendix B.2.1 and the method is explained in section 4.2.1.

4 Background Information

4.1 Unsupervised learning

To understand unsupervised learning, it is important to understand what labeled data means. When there is a correct answer to a question related to the data then that is labelled data. For example, when the data is images of handwritten digits, the label is the written digit. When there is no right or defined category, the data is unlabelled [26]. When dealing with unlabelled data, the only type of learning is unsupervised learning. In unsupervised learning the machine is given a huge amount of data, where the machine may identify trends or similarity. The algorithm will identify clusters or groups of similar items [26]. The end result of the clustering is highly dependant on which algorithm is used to identify these similarities. There are many methods that deal with different types of data and clustering. The bases for constructing these algorithms is distance and similarities. To recognize relationships among quantitative data, distance is preferred over similarity [33]. Clustering is by nature exploratory as it is looking for novel patterns. There is not one correct clustering algorithm as the clustering is in the eye of the beholder [12]. As there are many clustering methods, we will explore only two clustering methods, with their advantages and their disadvantages. Furthermore, besides choosing the method of clustering, it is important to explore methods to evaluate the clustering. Over the years, many methods for cluster evaluation have been proposed, however only some demonstrate promising and consistent performance [18]. These will be discussed afterwards.

4.1.1 Centroid-based clustering

The first type of clustering we consider here is the centroid-based clustering. In centroid-based clustering, the clusters are represented by central points. These central points are not necessarily observations in the dataset. For each of the other data points the assigned centroid is then the centroid that is closest (usually measured using euclidean distance) to that data point [18]. The most common method of centroid-based clustering is k-means. Essentially k-means seeks to minimize the total squared euclidean distance between all data points and their respective centroids [12]. It does this by repeating two steps until convergence [16]:

1. Generate a new partition by assigning each data point to its closest cluster centroid.
2. Compute the new cluster centroid, which is the mean value of the data-points in the cluster.

Advantages of k-means is that the complexity is relatively low and the computing efficiency is generally high [33]. However, since the algorithm wants to minimize the total squared euclidean distance between all data points and their assigned cluster center, it is very sensitive to outliers. Therefore the k-means method only works best for well separated data. Another disadvantage is that the number of clusters has to be decided upfront [12].

4.1.2 Distribution-based clustering

The second type of clustering considered in this work is the distribution-based clustering. In distribution-based clustering it is assumed that the data has an underlying probability distribution. Each cluster is assumed to be sampled by a different component [13]. One common model under this category is the Gaussian mixture model (GMM). In GMM, the data is assumed to be a sample from several independent Gaussian distributions and the parameters of the distribution are normally estimated using the Expectation Maximization (EM) algorithm [33]. Just like in k-means, the amount of clusters has to be decided upfront. Suppose the data comes from a mixture of K clusters then the EM algorithm maximizes the log-likelihood by using the posterior probability of cluster K given the data. Each iteration consists of the following three steps:

1. Estimate the expected value for each latent variable.
2. Optimize the parameters of the distribution using maximum likelihood.
3. Check for convergence of the parameters. If the parameters have converged, terminate the process. Otherwise, repeat steps 1 and 2.

Progressively the algorithm then maximizes the log-likelihood [18]. One of the advantages of GMM with respect to k-means is that clusters can overlap each other, therefore it can be seen as an improved version of

the k-means algorithm. Disadvantages of the method are the relatively high complexity of the EM algorithm and the high dependency on the initialization of the algorithm and normality assumption of the data [33].

4.1.3 Evaluation of clustering

For both the k-means method and the Gaussian Mixture model, the number of clusters has to be decided upfront. Thus normally the procedure involves clustering the data with a varying number of clusters, and then evaluate the clustering with a cost-function. We consider multiple cost-functions.
One type of cost-function considers a variation of the clustering algorithm cost-function, which is the total squared euclidean distance between all data points and their assigned cluster center for k-means and the log-likelihood for the GMM. Commonly used is the elbow method for k-means [26]. Furthermore, the I-index and the Bayes Information Criterion are studied as well since they have shown consistent performance for k-means and GMM clustering respectively [18] [20]. Other types of cost-function are geometry-based, by examining the distance between clusters, centroids and/or data points within clusters. Commonly used are the Silhouette analysis, Calinski-Harabasz Index and Davies-Bouldin Criterion [33]. These methods can be applied to both k-means and GMM.
For each of the clustering evaluation methods we consider the following:

Suppose $X = \{x_1, x_2, ..., x_n\}$ is the data set of n observations, K the amount of clusters, C_k the clusters with centroids z_k and $k \in [1, K]$ and U(X) the partition matrix with elements u_{kj}, $k \in [1, K]$,$j \in [1, n]$ and

$$u_{kj} = \begin{cases} 1 & \text{if } x_j \text{ belongs to cluster } k \\ 0 & \text{otherwise} \end{cases}$$

Elbow method As seen in section 4.1.1, the k-means method tries to minimize the total squared euclidean distance. Therefore the total squared euclidean distance can be considered as the cost-function. For each number of clusters K compute the cost-function J and plot the curve J as a function of K. As K increases, J decreases and the plot forms an 'elbow', i.e. the highest decrease in J value between cluster number K and K+1 [26]. The cost-function J is defined in equation 1.

$$J = \sum_{j=1}^{n} \sum_{k=1}^{K} u_{kj} \left\| x_j - z_k \right\|^2 \tag{1}$$

An example of a plot with a clear elbow is visible in Figure 3. However it is possible that the K-J curve does not show a clear elbow, making it not very useful in determining the number of clusters (see Figure 4).

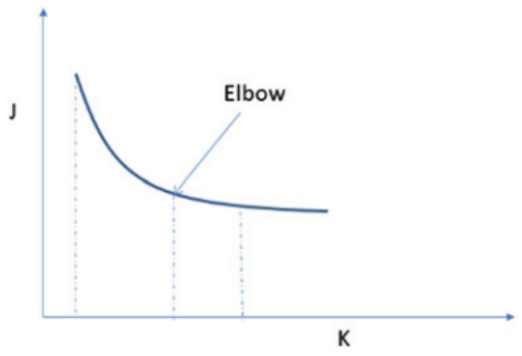


Figure 3: Cost function of the elbow method for k-means evaluation derived from the book 'An introduction to machine learning' [26] with clear elbow

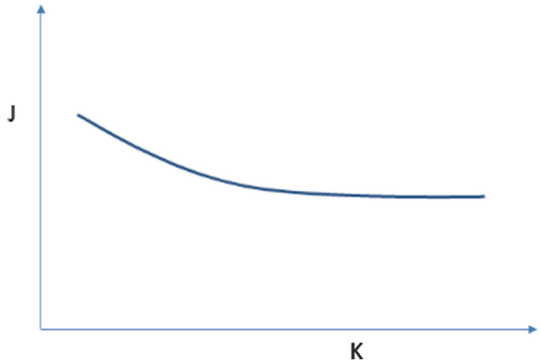


Figure 4: Cost function of the elbow method for k-means evaluation derived from the book 'An introduction to machine learning' [26] with no clear elbow.

I-index The I-index is a recently developed cluster validity index that performs very well for k-means [20]. It is a combination of three factors and is formally described in equation 2. The first factor penalizes the increase of amount of clusters to prevent over-fitting. The second factor is the ratio of total euclidean distance for 1 cluster and K clusters, which increases as the amount of clusters increases. Lastly, the third factor measures the maximum separation between two clusters over all possible pairs, which increases when the number of clusters K increases [20]. Thus the higher the I-index the better the solution.

$$
\begin{aligned}
I_K &= \left(\frac{1}{K} \times \frac{E_1}{E_K} \times D_K \right)^2 \\
E_K &= \sum_{k=1}^{K} \sum_{j=1}^{n} u_{kj} \left\| x_j - z_k \right\| \\
D_K &= \max_{i,j=1}^{K} \left\| z_i - z_j \right\|
\end{aligned}
\tag{2}
$$

Bayes Information Criterion (BIC) BIC attempts to balance the likelihood criterion of the model with a term that penalizes increasing complexity [18]. The formal definition of BIC can be found in equation 3:

$$
\mathrm{BIC} = K \ln(n) - 2 \ln(\widehat{L}) \tag{3}
$$

Where K is the number of clusters, n the number of observations and $\widehat{L}$ the maximized value of the likelihood function of the model. The lower the value of BIC, the better the clustering number.

Silhouette analysis Let x_i be an arbitrary observation in cluster C_i. Then a_i is the average distance between observation x_i and all other observations in C_i, d(x_i,C_j) is the average distance of x_i to all observations in cluster C_j and b_i is the minimum of all d(x_i,C_j) values of all the clusters C_1, C_2, ..., C_K except C_i. Then the silhouette value s_i is calculated using equation 4.

$$
s_i = \frac{b_i - a_i}{\max\{a_i, b_i\}} \tag{4}
$$

Each s_i value is in between -1 and 1, where values close to 1 and -1 represent good and bad clustering respectively. The average silhouette score indicates how good the clustering is [27].

Calinski–Harabasz Index (CH index) The Calinski–Harabasz Index is one of the most succesfull evaluations of clustering [18]. It looks at the relation between the overall between-cluster variance and the overall within-cluster variance. In equation 5 SS_B is the overall between-cluster distance, SS_W is the overall within-cluster distance, n_i is the number of observations in cluster C_i and m the mean of the sample data.

$$
\begin{aligned}
CH &= \frac{SS_B}{SS_W} \times \frac{(n-K)}{(K-1)} \\
SS_B &= \sum_{i=1}^{k} n_i \left\| z_i - m \right\|^2 \\
SS_W &= \sum_{i=1}^{k} \sum_{x \in C_i} \left\| x - z_i \right\|^2
\end{aligned}
\tag{5}
$$

Since in the ideal situation SS_B is maximized and SS_W is minimized, the higher the CH value the better the clustering.

Davies-Bouldin Criterion The Davies-Bouldin Criterion is another common method that looks at the worst relations between different clusters. Again, in equation 6 K is the number of clusters and $D_{i,j}$ represents the relationship between cluster i and cluster j. $\bar{d}_i$ in the nominator of $D_{i,j}$ in equation 6 is the sum of the average distance between each point in cluster i and its centroid. The denominator is the distance between the centroid of cluster i and the centroid of cluster j [33].

$$DB = \frac{1}{K}\sum_{i=1}^{k} \max_{j \neq i} \{D_{i,j}\} \tag{6}$$

$$D_{i,j} = \frac{(\bar{d}_i + \bar{d}_j)}{d_{i,j}}$$

Since for each cluster the algorithm takes the max of the $D_{i,j}$ values, it looks at the worst separations between clusters. Therefore lower DB criterion values are better than higher values.

4.2 Image segmentation

As mentioned in chapter 2 there are some steps that need to be dealt with before performing clustering. One of them is separating the particle from the background in the given gray-scaled images. We will describe the method that is currently in use at ASML, as well as two alternative methods, since the current method does not segment the particle from the background as well as needed in the current study and it does not deal with the 'missed' images in the data as mentioned in section 1.1.

4.2.1 Gradient

The algorithm makes use of the gradient of the image. The gradient calculates the central difference for internal data points. For example, consider a matrix with unit-spaced data, A, that has horizontal gradient G = gradient(A). The interior gradient values are

$$G_{i,j} = 0.5 * (A_{i,j+1} - A_{i,j-1}) \tag{7}$$

for $i, j \in (1, N-1)$, where N is the horizontal size of the image. For the border points of the image the gradient values are:

$$G_{i,1} = A_{i,2} - A_{i,1} \tag{8}$$

$$G_{i,N} = A_{i,N} - A_{i,N-1} \tag{9}$$

Both the horizontal and the vertical gradients are calculated for each of the image pixels and the new matrix is composed by the euclidean distances between the obtained vertical and horizontal gradients. Let G_h and G_v be the horizontal and vertical gradients respectively, then the newly obtained image I is defined as

$$I_{i,j} = \sqrt{G^2_{h_{i,j}} + G^2_{h_{i,j}}} \tag{10}$$

for $i, j \in (1, N)$.
After the new image I is obtained, the best threshold value is determined with an algorithm of ASML. The threshold splits the image into two parts. The pixels with a grey value below the threshold are replaced with a black pixel and the pixels with a grey value above the threshold will be replaced by a white pixel. After the splitting, if all goes well, the white surface should contain the contour of the particle which then completes the segmentation of the particle from the background.
Unfortunately the current process is not accurate enough for all the particles. The mask obtained covers the entire particle, but contains redundant pixels that should not be contained within the borders of the particle. A sample of extracted contours of four particles can be seen in Figure 5. A larger sample can be found in the appendix A.1

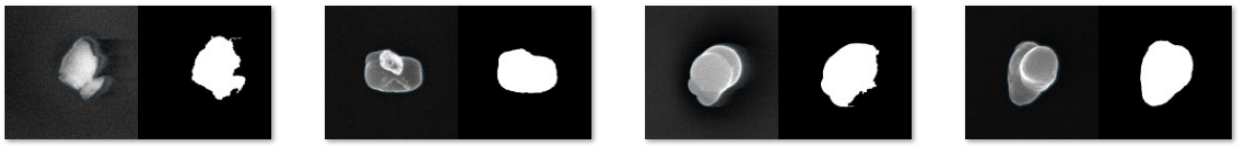


Figure 5: Gradient segmentation method (currently in use at ASML)

4.2.2 Image histogram analyses

A simple approach to image segmentation consists of disregarding the position of the pixels and focus only on the grey-values of the image. An image histogram is a gray-scale value distribution showing the frequency of occurrence of each gray-level value [25]. Even though the original image cannot be retrieved from the image histogram since it only describes the frequency of the grey-values, many useful image processing operations can be derived from it [6]. For example, the image histogram is used to find the threshold to distinguish the background from the desired object in the image. Threshold determination from the image histogram is probably one of the most widely used applications [22]. The image histogram analysis is based on the assumptions that the foreground object has a different distribution than the background of the image. Therefore the background and the foreground object will be two distinguishable peaks in the image histogram. It is possible that two peaks overlap, however a minimum between the two peaks can be detected in order to separate the two objects [25]. An example of this process can be found in Figure 6. Nonetheless, the method sometimes fails to succesfully separate the object from the background when no minimun can be detected, e.g. when the background is non-uniform. Therefore this method only provides a rough assessment of the image segmentation.

Figure 6: Removal of background from anatomical structures. (a) Original image; (b) histogram (arrow marks the threshold value); (c) thresholded image: anatomical structures remaining in the image are marked in white. Image obtained from Ewa Pietka, in Handbook of Medical Imaging, 2000

4.2.3 Active contours

Unlike the two methods described above, which use a threshold pixel value to form a binary image, called pixel segmentation, the active contour method uses a curve segmentation. In general this means that it is looking for a regular parametric curve that defines the image feature of interest by minimizing an energy function [8]. The energy of the basic snake, E, constitutes of two factors: the internal and external energy. The internal energy is a combination of the continuity of the contour E_{cont} and the smoothness of the contour E_{smooth}. The external energy is the force within the image itself E_{image}. In this case it consists of the gradient of the image, which tends to attract the contour to the border of the image of interest [7]. Suppose the snake $\mathcal{C}(s) : [0, 1] \rightarrow \mathbf{R}^2$ then the snake energy is formulated in equation 11.

$$E = \int_0^1 \left(E_{\text{cont}}\left(\mathbf{C}(s)\right) + E_{\text{smooth}}\left(\mathbf{C}(s)\right) + E_{\text{image}}\left(\mathbf{C}(s)\right)\right) ds \tag{11}$$

The internal energy is described by the first and second term, the third term represents the external energy and α, β and λ are real positive constants. However, there are still some issues with the original active contours model. For example the snake is not attracted to the object in the image when the initial contour is too far from the object edge (thus the gradient is very low at the initial contour) and there are several parameters that still need to be estimated. This resulted in many reformulations of the initial energy level and its minimization. One example that has been largely used in medical image processing is the geodesic active contour method [7]. The paper proposed a new scheme to detect the object boundary by using minimal path computations. It introduces a new term in the initial energy function that attracts the deforming curve to the boundary of the foreground object. It also reduces the amount of estimated parameters needed for the energy function and the solution exists and is unique [7]. It can also detect holes in the object, which is not possible with the original snake method. This tells us that low contrast within the boundaries of the particle is important, since the assumption is that the particle does not have any holes.
The geodesic active contour is available from the MATLAB library, which makes it easy to use. However a contour outside the foreground object is needed [3]. Noise in the image can affect the active contour as seen in Figure 7. However, the noise can be reduced by an adaptive filter that smooths the image to limit the effect of noise on the accuracy of the active contour. The adaptive filter used is 'wiener2' in MATLAB which estimates the local mean and variance of each pixel and reduces the variance, i.e. smoothing. The method is selective i.e. higher variances are smoothed less than low variances to preserve edges and other high-frequency parts of the image.

In Figure 7 an example object is recognized using the method in the absence as well as in the presence of (smoothed) Gaussian noise.

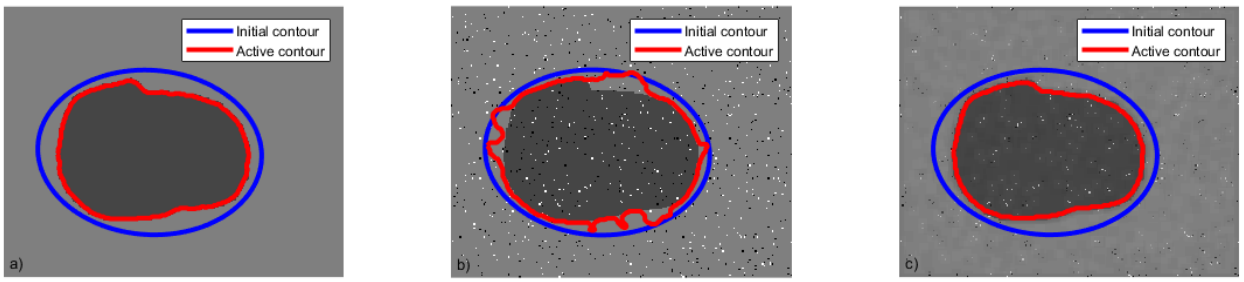


Figure 7: a) Example of the active contour method; b) Example of the active contour method with noise; c) Example of the active contour method with the noise smoothed out using the function 'wiener2' from MATLAB **??**

4.3 Feature selection

After the region of interest is segmented, morphological features are extracted and selected. These features are then used for clustering multiple images. The selection of the features is an iterative process, in which the chosen features are extracted, the data is clustered and the results are evaluated based on the desired outcome. A desired outcome can include, but is not limited to, clearly notable clusters when the observation values are plotted against each other or visible similarity in the images of the same cluster. However, desired results are dependable of the end user which is ASML in this case study. Since it is assumed that most of the images are characterized by roundly shaped particles, it is important to describe features that evaluate the roundness of the particle. Moreover, the roughness (i.e. irregularity) of the particle should be captured as well in order to establish different levels of irregularity of the particles.

4.3.1 Roundness: Ellipse fit

In order to describe the roundness of the particles, an ellipse can be fitted to it. A simple and accurate method to fit an ellipse to the contour of the segmented particle is with the use of the least-squares method [14],

which minimizes the total distance between the points of the contour and their projection on the ellipse. From the best fitting ellipse, the shape of the ellipse and the error of the fit can be determined [5]. The error is formulated in equation 13, where A is the area within the contour and B the area within the ellipse [33] visible in Figure 8. The shape is described by the eccentricity of the ellipse formulated in equation 12, where 'a' is the length of the semi-major axis and 'b' is the length of the semi-minor axis.

$$Eccentricity = \sqrt{1 - \frac{b^2}{a^2}} \tag{12}$$

$$Error = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \tag{13}$$

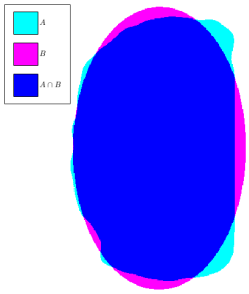


Figure 8: Mask of the particle, ellipse fitted to the particle and the overlap of the two

4.3.2 Roughness method 1: Irregular shape descriptors

When describing the roughness of shapes, commonly used irregularity descriptors are: formfacter, compactness and solidity [24]. Nonetheless, these shape features can be closely related to the roundness features described above 3.1.

Formfactor The formfactor expresses the roughness of a shape by a single number between 0 and 1, where 1 represents a perfect circle and lower numbers represent shapes with increasingly rougher surfaces. Equation 14 shows how the formfactor can be calculated from the area and perimeter of a shape.

$$\text{Formfactor} = \frac{4 \times pi \times \text{Area}}{\text{Perimeter}^{\,2}} \tag{14}$$

Compactness The compactness of a shape is similar to the formfactor. However, it is $4 \times p$ for round shapes and can increase to infinity for irregular shapes.

$$\text{Compactness} = \frac{\text{Perimeter}^{\,2}}{\text{Area}} \tag{15}$$

Solidity The solidity shape parameter measures the relation between the area of the segmented particle and the area of the convex hull as shown in equation 16.

$$\text{Solidity} = \frac{\text{Area}}{\text{ConvexArea}} \tag{16}$$

4.3.3 Roughness method 2: Fast Fourier Transform

In 3.1 the irregularity of the particle is established by using an estimate of the original contour of the particle and calculating the distance between the obtained estimate and the original contour [5]. The original contour is the contour obtained after image segmentation. However, due to lack of information found on the estimate used in the paper, described in section 3.1, a different contour estimate will be analyzed of which the same features can be determined.
Since the image is in 2D, we can image the contour of the particle as points in the complex plane. The Fourier transform can be applied to the points in the complex plane, forming the Fourier series of the contour. The Fourier series can be visualized as stacked arrows that each have their own scale and speed of rotation, which together draw the closed contour given as input. This process is visualized by 3blue1brown [29]. The more rotating arrows that are used, the more precise the contour is approximated [21]. A number of 'rotating arrows', i.e. stacked circles, is chosen and the reversed Fourier transform is applied to obtain an estimate of the original contour. Complex shapes need a higher number of these stacked circles to achieve a closely related estimated contour, while circles and ellipses only need a few [21].

The roughness features are the described distance vector distribution, which is expressed by the following factors:

1. The first four moments, i.e. mean, variance, skewness and kurtosis.
2. The quantiles 0, 0.05, 0.10, ..., 0.95, 1.00 of the cumulative probability

4.3.4 Roughness method 3: HU's moments

Contrary to the common irregularity descriptors, moments can be used to represent and reproduce a shape at any required degree of precision. With enough moments it is even possible to exactly reconstruct the particle shape [10]. These moments can also be normalized so that they are invariant under rotation and change of scale.. These normalized moments are called HU's moments [15].
Moment approximations take the form of series expansions of the type that is visible in equation 17 for a picture function f(x,y) [10].

$$M_{pq} = \sum_x \sum_y x^p y^q f(x, y) \tag{17}$$

They are widely used in image processing and work best for irregular shapes. This method has greater difficulty with shapes that have fine-irregularity or serration [28].

4.4 Dimension reduction: principal component analysis

Often in data analysis the data sets are large and the different features may correlate with each other. Therefore a method is proposed that finds an orthogonal basis of which different individual dimensions in the features are uncorrelated. The procedure is called principal component analysis, where the basis vectors are the principal components.
Geometrically we can look at the n observations x_1, x_2, ..., x_n as vectors in the $\mathcal{R}^d$ space. The first component is composed as the best fit line through the data points in the $\mathcal{R}^d$ space, where the direction is chosen such that the variance is maximized. The second component is orthogonal to the first and the variance is again maximized. The process is repeated to compose d components. Even though d amount of components can be found, it is preferred that most of the variance is explained by $m \leq d$ components, such that the dimensionality is reduced [4].
Principal component analysis requires a few variables: loadings, scores and explained variance. Firstly the explained variance tells us how much of the total variance is explained by each component. Often the number of components m is chosen such that 95% of the total variance is explained by the d number of components. The scores are the representations of the data matrix in the principal component space and the loadings are the coefficients for the features in order to obtain the score values. Suppose 95% of the variance is explained by m components of the d features, the score value $t_{i,1}$ of observation i and component 1, data point $x_i = (x_{i,1}, x_{i,2}, ..., x_{i,m})$ and loadings $p_{1,1}$, $p_{2,1}$, ..., $p_{m,1}$ for component 1. Then the score value $t_{i,1}$ is

obtained by applying equation 18 [9].

$$t_{i,1} = x_{i,1}p_{1,1} + x_{i,2}p_{2,1} + \ldots + x_{i,m}p_{m,1} \tag{18}$$

In general, the score matrix T is retrieved by matrix multiplication between the n observation matrix $X = (x_1, x_2, ..., x_n)$ and the loading matrix P as seen in equation 19.

$$T = XP \tag{19}$$

Before PCA can be applied the features need to be normalized, by subtracting the mean value and dividing over the standard deviation of the feature values. This step is important as it removes any bias from the data and it does not affect the correlation within the data [4].

5 Methodology

In this section we describe our approach to clustering the images in our case study. Each of the steps in the procedure tries to answer one of the sub-questions in section 2. The procedure includes preliminary steps, such as (1) detecting and excluding unusable images, (2) performing image segmentation to separate the particle from the background, (3) extracting meaningful morphological features and (4) reducing the dimension with PCA. The so-obtained features will then be used for clustering. These steps will be largely discussed in the following sections.

5.1 Detecting usable images

The detection of usable images will make use of the image histogram analysis as specified in section 4.2.2. Since this method inspects, independently of position, each of the pixel values of the image to detect a foreground object it is best to have maximum contrast between the foreground object and the background. This is the case for the 'Internal' images described in section 1.1.3 (also see Figure 1a). In section 4.2.2 it is explained that ideally the foreground and background will have distinguishable peaks in the image histogram of which in this case the second peak is the foreground object as the object consists of higher pixel values. Therefore the splitting will be done on the prominence of the second peak which is explained in more detail below.

5.1.1 Assumptions

1. **First peak**: The first peak is the background, made up mostly of black pixels (i.e. low pixel values). The first peak starts in the origin and is symmetric, such that its maximum is in the middle of the peak.
2. **End first peak**: As point one already implies, the index of the end of the first peak is twice the index of the maximum value of the first peak. Additionally, it is assumed that the maximum of the second peak follows the end of the first peak (if there exists a second peak).
3. **Unusable images**: the image histogram has no two individual peaks for images that either do not contain a foreground object or contain a foreground object that is too small/unclear to be detected. These images will be unusable for further analysis.

5.1.2 Parameters

- **Second peak prominence**: The prominence of the second peak is defined as the height of the peak minus the minimum value obtained between the first and the second peak. A significant second peak means thus means a high prominence. A visual example of the peak prominence can be seen in Figure 9
- **Image mean**: The average value of all the grey-values in an image.

5.1.3 Approach

The images will be split into three groups:

1. Images that contain a single particle with clearly visible contour and texture.
2. Images that contain a single particle with clearly visible outline, but with undetectable texture.
3. Unusable images with either no particles or poor distinction between particles and background.

The distinction between the first and the second/third group is the absence of a prominent second peak in the image histogram. The prominence of the second peak is described in the parameters, which is and will be different for each of the images. When the prominence of the second peak is above a certain threshold the image is labelled as group 1. To differ between group 2 and group 3, we will look at the mean value of the image. The two groups have no prominent second peak, which means the image histogram is dominated by the first peak of the background. In the case of group 2, the background is still mostly black, otherwise the particle would not have a clearly visible outline, while group 3 will contain more noise. The noise consist

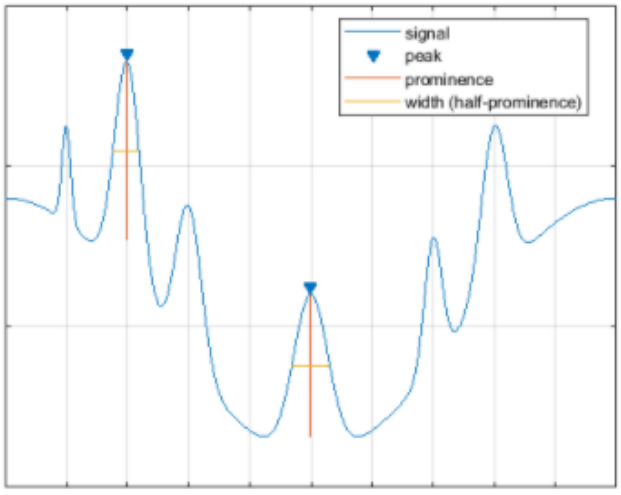


Figure 9: Visual example of peak prominence for arbitrary peaks.

of higher pixel values, which results in a higher mean. Therefore the mean was chosen to split group 2 from group 3. When the image mean crosses a certain threshold the image is split into group 3.
Combining the two split steps, using the prominence and the image mean thresholds, the images are split into the three groups described above. A schematic overview of this process is visible in Figure 10.

The issue that is left to be dealt with is the threshold values of the prominence and the image mean. In order to deal with this matter, 300 images were manually labelled with the preferred group based on the 'Internal' images. This labelling is very subjective, therefore this method will only give a rough indication in which group each image should go. By looking at the prominence of the second peak for each label, the prominence threshold is chosen such that images with label 1 are split into group 1 while also limiting the amount of images with label 2 or 3 in group 1. The same method is applied to find the image mean threshold. The previous steps are done for 70% of the labelled data. The last 30% of the data is used to validate the end-results.

To show the result of the splitting, we use a confusion matrix of the validation data. In the confusion matrix the row index represents the label and the column index represents the group the image is split into. Each entry a_{ij} is the amount of images that have label i and are split into group j.

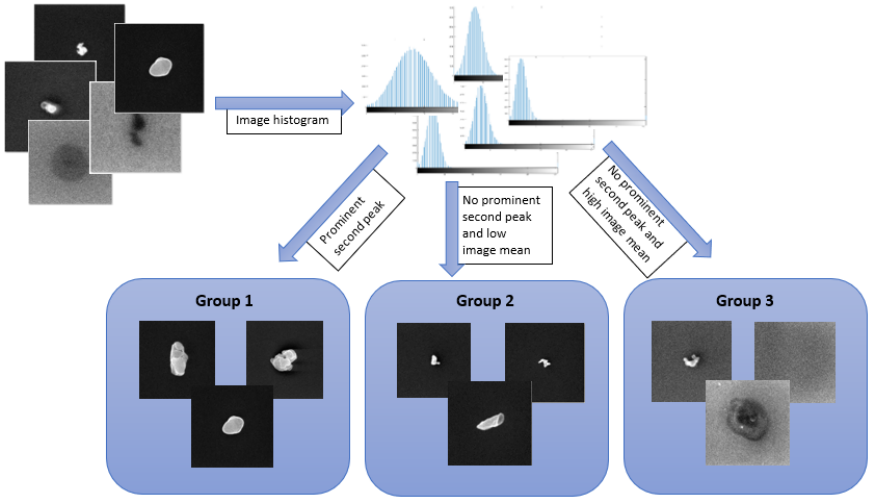


Figure 10: Overview step 1 splitting

5.2 Image segmentation

The set of usable images, group 1, are segmented to obtain the mask of the particle needed for further analysis. To advance the image segmentation provided by ASML the geodesic active contour method is used as described in section 4.2.3. Furthermore the MATLAB code of the image segmentation method can be found in appendix B.2.2 and the previous image segmentation method provided by ASML can be found in B.2.1.

5.2.1 Assumptions

1. **Particle**: In each image only one particle is visible, which doesn't contain any holes. Only fully visible particles can be used for further analysis. Fully visible means that the particle does not touch the border of the image, since we cannot know what shape the particle takes outside the image borders.
2. **Different image types**: The position of the particle in the different types of images does not change. Types of images are explained in section 1.1.3.

5.2.2 Approach

The preceding image segmentation method used the images of type 'Internal' as specified in section 1.1.3 and visualized in Figure 1a, as it shows the highest contrast between the background and the foreground object around the edges. However, for the geodesic active contour it is important that the contrast within the boundaries of the particle is low such that no 'holes' are detected, which makes the image type shown in Figure 1d a better option. Since the particle does not change in position in each type of image, we can use a different type.

To apply the snake image segmentation method, both an initial contour C and input image I is needed. The MATLAB adaptive filter 'wiener2' is used to form the input image I from the image shown in Figure 1d, since it reduces noise and enhances the snake method (see Figure 7) **??**. The input image is then binarized and holes are filled using the MATLAB function 'imbinarize' and 'imfill' respectively to form a mask M_1 [31]. As mentioned in section 1.1.3 the pixel-values within the border of the particle are the highest in the input image, which means that mask M_1 should at least contain all the pixels within the border of the particle. The boundary of the mask formed by the intersection between the mask M_2, obtained after the gradient segmentation method, and mask M_1 forms the initial contour C, i.e. $C = \partial(M_1 \cap M_2)$. The fact that the chosen snake method is biased to contract inward and both masks M_1 and M_2 are a coverage of the particle with some redundant pixels that should be excluded makes C a valuable initial contour.
Images are removed when multiple particles are visible or when the particle is not fully in frame. These type of images are detected when the mask M_1 is not fully connected and when the mask M_1 touches the border. After the snake image segmentation method the mask M_{snake} is obtained which is used for further analysis.

5.3 Morphological features extraction

When the M_{snake} is obtained, from the image segmentation step, the interesting features can be extracted. The selected features are described in section 4.3. The MATLAB function 'regionprops' was used to calculate the irregular shape descriptors [31]. Furthermore the MATLAB code of the feature extraction can be found in appendix B.3.

5.3.1 Assumptions

1. **Invariant property**: all the features are scale and rotation invariant.

5.3.2 Approach

For each of the features it is assumed that it is scale/rotation invariant, therefore this property is evaluated for confirmation as it is a very important property.

Evaluation scale/rotation invariant property method 1: We define data table $\mathcal{D}$ where the rows represent the observations and the columns the various features. We rotate one particle 360 times 1 degree each time to obtain rotation data table $\mathcal{D}_r$, where the row index 'ii' represent the rotation angle 'ii' and the columns the various features.
We also scale the same particle by 0.01, 0.02, ..., 2.98, 2.99, 3.00 to obtain scaled data table $\mathcal{D}_s$, where the rows index 'ii' represent the scaling factor '$\frac{ii}{100}$' and the columns the various features.
The data tables $\mathcal{D}_s$ and $\mathcal{D}_r$ contain the features for each scale factor and rotation angle of that specific particle respectively.
Each column of the data tables $\mathcal{D}$, $\mathcal{D}_s$ and $\mathcal{D}_r$ are normalized using the mean and standard deviation from the columns of $\mathcal{D}$, such that the bias in the data is removed and features with different ranges can be compared. Doing this we obtain the normalized data tables $\tilde{\mathcal{D}}$, $\tilde{\mathcal{D}}_s$ and $\tilde{\mathcal{D}}_r$. We compare the ratio between the range of the features of the scaled particle $\tilde{\mathcal{D}}_s$ (resp. rotated $\tilde{\mathcal{D}}_r$) and the features of the original particle $\tilde{\mathcal{D}}$, to detect if a feature range differs significantly with respect to the original range. When the range of a feature in $\tilde{\mathcal{D}}_s$ (or $\tilde{\mathcal{D}}_r$) is very large with respect to the range of that similar feature in $\tilde{\mathcal{D}}$ then this means that the feature value changes significantly when a particle is rotated or scaled. This implies that the feature is not scale/rotation invariant. Here significant means more than 0.05% of the original range.

Evaluation scale/rotation invariant property method 2: A second method to evaluate the scale/rotation invariant property can be applied after the clustering method has been implemented to data table $\mathcal{D}$. Therefore this will be explained further in section 5.4.

5.4 Clustering algorithms

After the features are extracted, we can attain for each of the n observations x_1, x_2, ... x_n d features, accommodated in the data table $\mathcal{D}$. Within this section we look at the steps taken before clustering and the evaluation of the final clustering. Both are described in more detail below.

5.4.1 Assumptions

1. **Existence and uniqueness**: There exists a solution to the K-means and Gaussian mixture model clustering and the solution is unique.

5.4.2 Approach

As explained in section 4.4 the features of the n observations may correlate, therefore principal component analysis (PCA) is applied before clustering to form an orthogonal basis of the feature vectors. The PCA method also includes the normalization of the data before forming the orthogonal basis. The amount of m components remaining from the PCA are chosen such that 95% of the data variance is explained. The loadings of the m components after PCA are studied to gain knowledge about the features that were most relevant for this case. Furthermore the score values of the m components and n observations are used as input for the clustering algorithms. We choose to apply both clustering methods K-means and Gaussian Mixture model, introduced in Section 4.1.1 and 4.1.2, and implement them in MATLAB using the functions 'cluster', 'fitgmdist' and 'kmeans'.
The clustering evaluation methods explained in section 4.1.3 are used, to evaluate the clustering algorithms and to find an optimal amount of clusters [32]. The function 'evalclusters' of MATLAB is used to implement the evaluation methods in MATLAB [32].

Evaluation scale/rotation invariant property method 2: Recall the normalized data table $\tilde{\mathcal{D}}_s$ and $\tilde{\mathcal{D}}_r$ from section 5.3. We define the score matrices $\mathcal{S}_s$ and $\mathcal{S}_r$ containing the score values obtained from applying the PCA to the data tables $\tilde{\mathcal{D}}_s$ and $\tilde{\mathcal{D}}_r$ respectively. Now, for each score in $\mathcal{S}_s$ and $\mathcal{S}_r$, we use the obtained clusters in the previous step to assign it to the corresponding cluster. Then we observe whether or not this cluster is different from the cluster assigned to the original (not rotated or scaled) observation. The error, total wrongly clustered observations divided by the total observation in $\mathcal{S}_s$ or $\mathcal{S}_r$, indicates the scale/rotation invariance of the PCA components. Therefore it is hard to find exactly which feature is not scale/rotation invariant. We can merely take a guess which feature may have caused an error by looking at the loading values of the components. Thus this evaluation method is a second check to see if the features we chose in the section 5.3 are indeed rotation/scale invariant.

6 Results and discussion

The following section illustrates the results of each of the steps taken to answer the main question. Each step tries to answer one sub-question, in order to give a solution to the main question.

6.1 Detecting usable images

In Figure 11 the prominence of the second peak can be seen for each label of the training images. According

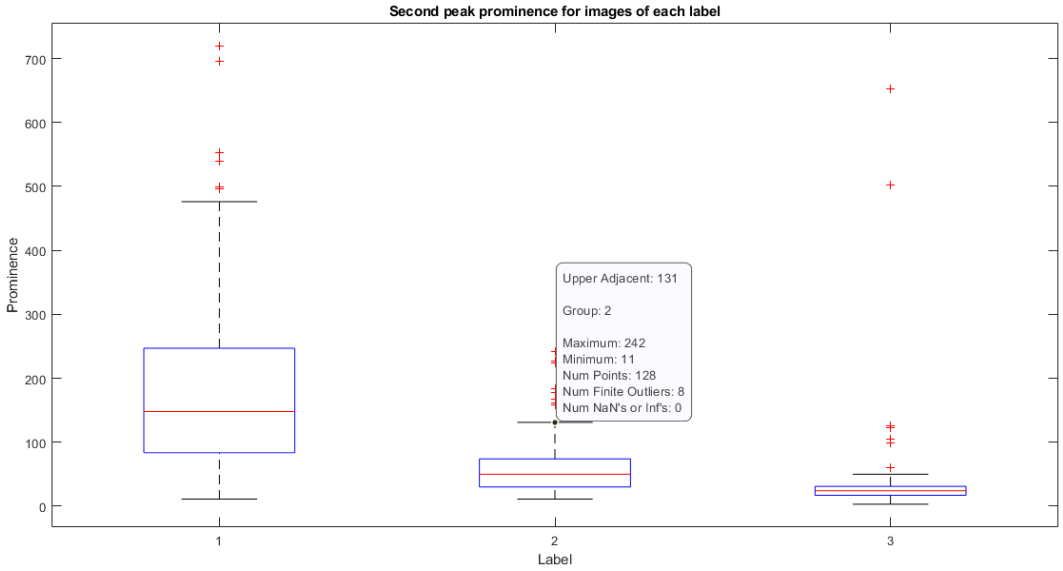


Figure 11: The prominence of the second peak for each of the labels

to Figure 11, when disregarding the outlier values, the prominence threshold can be set to 131 so that the separation between label 1 and label 2 is accurate. However, when we plot the second peak prominence values of images of Label 2 from low to high, visible in Figure 12a, we see that 110 of the 128 values is equal to or lower than 87. That is almost 86% of the total amount of images. Up until that threshold the line in Figure 12a is almost linear as can be seen in Figure 12b since the difference between the values are between 0 and 3 until the 110^{th} value. A sample of the images that have a larger second peak prominence than 87 are visible in Appendix 32. Images with a second peak prominance higher than 87 show a much wider

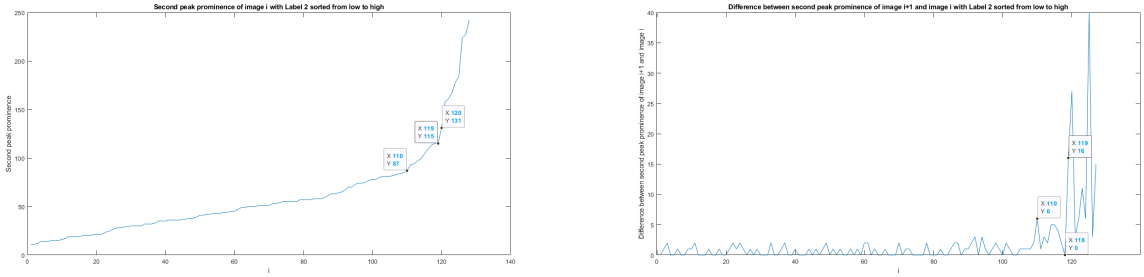


Figure 12: a) Sorted second peak prominence Label 2 b) Difference sorted second peak prominence Label 2

distribution than those lower than 87. Therefore the prominence threshold is set at 87. In the discussion we look at the images with label 1 with a second peak prominance lower than 87 and images with label 2 and 3 with a second peak prominance higher than 87.

After lowering the second peak prominence threshold to 87, we can look at the image mean for each Label and each Group visible in Figure 13. Since we have not made any distinction between group 2 and group

3, all the images that were not assigned to group 1 are arbitrarily assigned to group 3 (since the splitting into groups 2 and 3 happens after this step). In Figure 13 each boxplot represents a label combined with the group the images are in. The label represents the group the image should be assigned to and the group number tells us which group it is currently assigned to. The optimal solution should have all images with label 1 in group 1 etc. The confusion matrix is given in the black lined box in Figure 13.

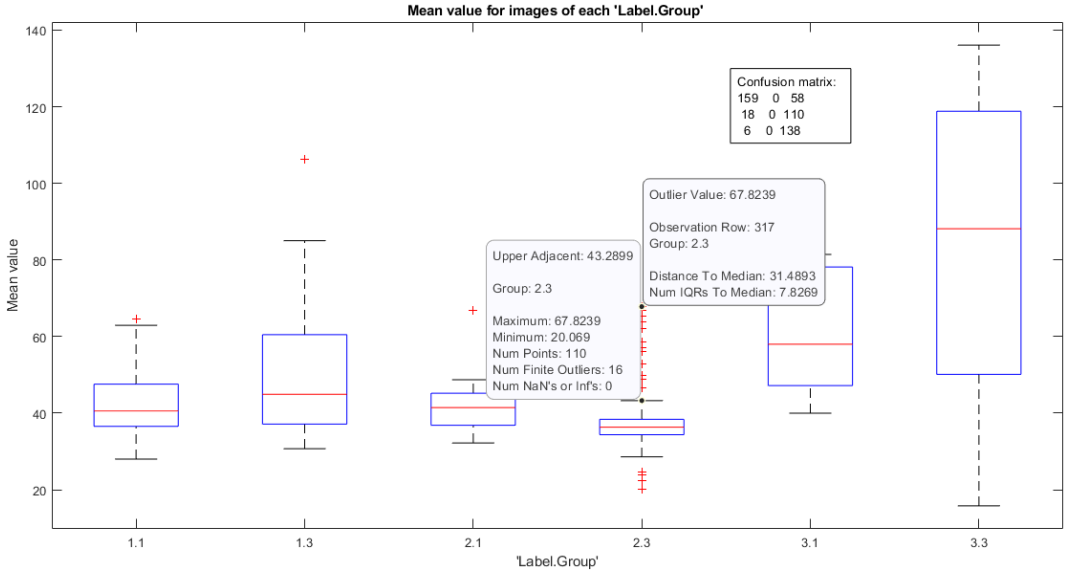


Figure 13: The image mean for each of the labels and groups combination

As seen in Figure 13, the image mean cut-off can be set to 68 such that all the images with label 2 that are currently in group 3 are split to group 2.
Nonetheless, when we disregard the outlier values, which are 16 values of the total 116, the threshold can be set to 44. To examine these outlier values more closely, we look at the sorted image mean values of images with label 2 and also the differences between adjacent elements (see Figure 14). The difference in image mean between the 101^{st} and the 9^{th} image is only 7.05. Again, between these values the line in Figure 14a is almost linear. After the 101^{st} the values have a much wider variation then before, which would mean 39 would be a good cutoff value. Taking into consideration that the other cutoff value options mentioned above, 44 and 68, would increase the range a lot but not include that many more images, the image mean threshold is set at 39. In the discussion we look at the wrongly grouped images.

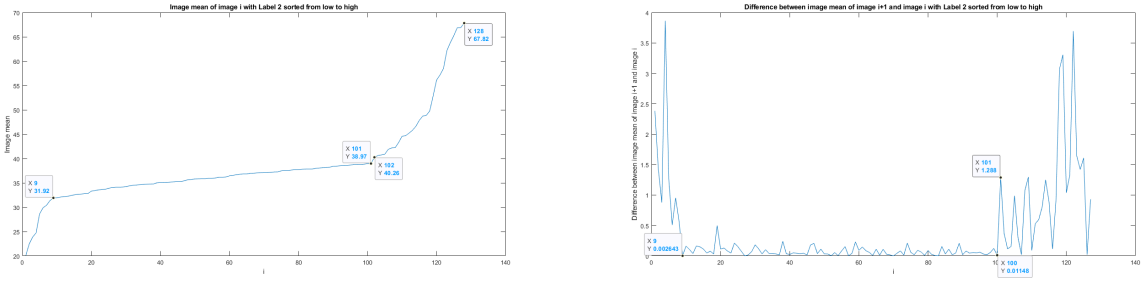


Figure 14: a) Sorted image mean Label 2 b) Difference sorted image mean Label 2

After the image mean cut-off is set to 39, we obtain 2 confusion matrices, i.e. one of the training data set and one of the validation data set. Both can be seen in Table 1.

Both for the training and validation set, the images in group 1 and 3 are mostly from the designated label. However, group 2 consists of many images with label 1 and 3. A sample of images of each Label and Group

	Group 1	Group 2	Group 3
Label 1	159	24	34
Label 2	18	94	16
Label 3	6	29	109

	Group 1	Group 2	Group 3
Label 1	71	8	15
Label 2	7	42	6
Label 3	5	18	39

Table 1: Left: confusion matrix of the training data. Right: confusion matrix of the validation data

combination is visible in Appendix A.3.
In total 27.779 images were split to group 1.

6.1.1 Discussion

In Figures 30 and 31 in Appendix A.3 we are able to distinct the background from the foreground object with the naked eye. Nonetheless based on the image histogram these images are not detected as usable images (since they are split to groups 2 and 3). An explanation could be that the background contains too much noise and that there is little to no contrast between the pixel values of the foreground object and the background. Therefore the two peaks that should appear in the image histogram overlap too much, i.e. only one peak is detected.
Furthermore, in Figures 32 and 35 we see a sample of the images that were split to group 1 but have label 2 or 3. We see in these images that the foreground object should be distinctive from the background as the contrast within pixel values is quite high. However the particle itself is either blurry (no distinctive texture) or half visible (unusable for shape analysis). This indicates that the current method is not usable for predicting a distinct texture within the image.

Based on the results, the assumption that the current method could distinguish in blurry or distinctive texture is rejected. In this study case only the shape is analyzed, which means that it is not important to address this problem further in this study case.

6.2 Image segmentation

Each of the steps taken in the image segmentation process can be seen in image segmentation are visible in Figures 15 and 16. In 'a' and 'b' we can see the image before and after the adaptive filter 'wiener2' respectively. Image 'c' is the binarization of image 'b'. In image 'd' some pixel values are set to 0 using the previous gradient image segmentation method. Examples of the gradient image segmentation method can be found in Appendix A.1 Figure 27. Image 'e' is the final image mask and is formed using the active contour method with image 'b'and mask 'd' as input. It takes approximately 6 seconds to segment one image. To

Figure 15: Example of image segmentation: a) Initial image. b) Smoothed initial image. c) Binarized image. d)Binarized image adapted to previous image segmentation method. e) Final image mask

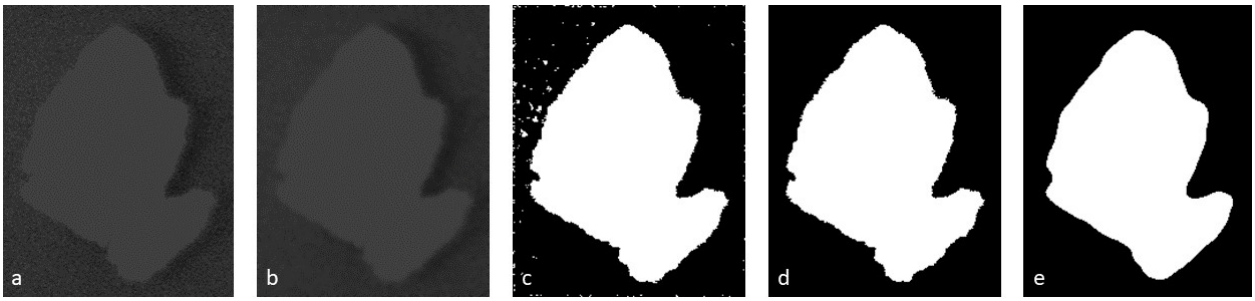


Figure 16: Example of image segmentation: a) Initial image. b) Smoothed initial image. c) Binarized image. d)Binarized image adapted to previous image segmentation method. e) Final image mask

reduce computation time, 5000 of the 27.779 usable images (group 1 images from section 6.1) were randomly selected for image segmentation. A sample of the extracted contours by the active contours method can be found in Figure 17.
A sample of the images that were removed from the data set are visualized in Figure 28 in Appendix A.2. In total 40 images were removed from the data set of 5000 images. The removed images (see Figure 28 in Appendix A.2) contain images that only touch the border of the image for a small amount of pixels of which the shape is almost fully visible.

6.2.1 Discussion

Image 'd' of Figures 15 and 16 already represents a good segmentation of the particle, but the active contour helps to refine the image a bit more. A disadvantage of the smooth factor of the active contour is that small roughness spikes in the contour of the particle are non-detectable, since these are smoothed out. Therefore very smooth particles cannot be distinguished from a little rough particles.
Another improvement could be a cut-off value for the amount of pixels that touch the border. This could limit the amount of removed images. Additionally, in some images, multiple particles were detected whereas on closer inspection by the naked eye no distinguishable second particles were visible. For most of these images the outline is too vague or incomplete and the method seems to fail in such cases.

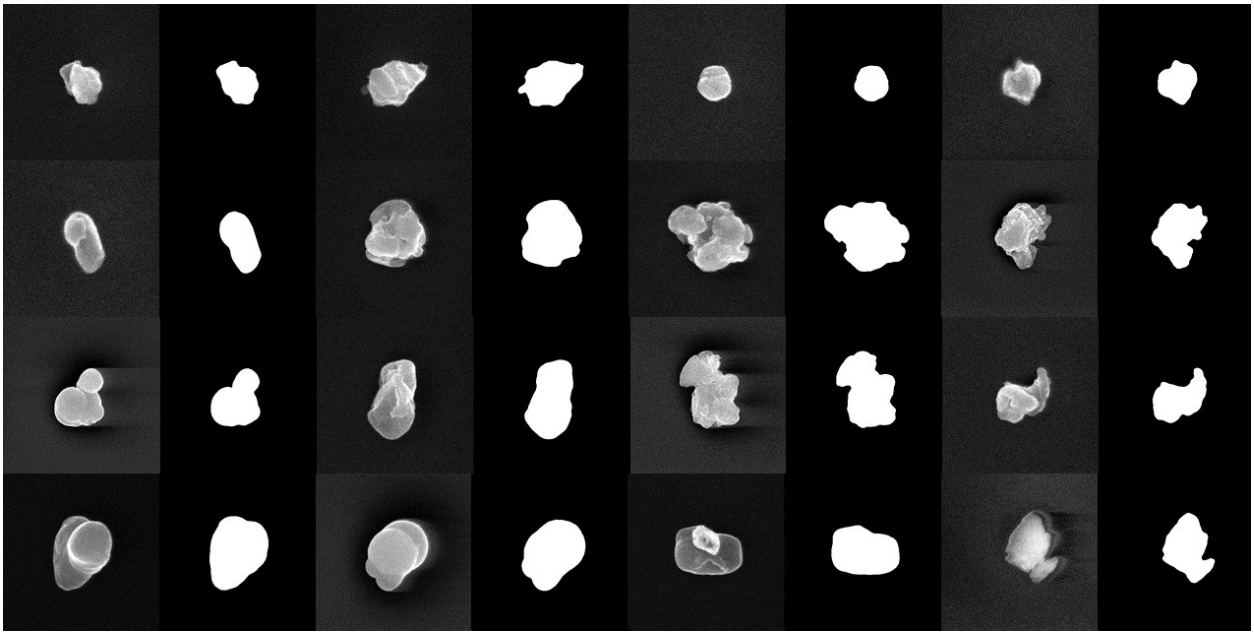


Figure 17: Sample image segmentation active contour method

6.3 Morphological features extraction

Sample images of distinctive values of the eccentricity, ellipse error, formfactor, compactness, solidity, first HU moment, first quantile value of the distance vector and the mean of the distance vector are given in Appendix A.4.1.

6.3.1 Evaluation scale/rotation invariant property method 1

Table 2 contains the ratio of the ranges in data tables $\tilde{\mathcal{D}}_r$ and $\tilde{\mathcal{D}}_s$ and range in data table $\tilde{\mathcal{D}}$ for each feature respectively. We observe that for many features the ratio is above 5%, which means the change in the range of features due to rotation and scaling is significant (given the threshold value chosen of 5%). Each of the observation values of $\tilde{\mathcal{D}}_r$ and $\tilde{\mathcal{D}}_s$ for the obtained features are visualized in Appendix A.4.2. Only the second distance quantile and the distance mean is visualized, as the largest value ratio values are the highest of all the quantiles and moments in Table 2. While the ellipse error values for the rotated particle is about the same, the eccentricity, formfactor, solidity and compactness values are closest to the original particle feature values (X=360) when the rotation is $k * 90$ degrees for $k = [1, 2, 3, 4]$. The values between two $k * 90$ consecutive points seem to repeat. This indicates that the method used to rotate the image in order to obtain the features works best for multiples of 90 degrees, since it consistently gives different values for other rotations. For the scaled particles, the values vary more from the original particle feature value (X=100) when the scale is below 1 (ii=100).
The HU moment values appear to be constant under different rotations, with the exception of the 3_{rd} and 4_{th} moment. The change in the 3_{rd} moment is insignificant while the ratio of the 4_{th} moment is above 0.05. Furthermore, the values of the 4_{th} moment in Appendix A.4.2 are at their highest when the rotation is 0, 180 or 360 degrees and lowest when the rotation is 90 or 270. This observation reveals a high dependency on rotation for the 4_{th} moment of HU.
The HU moments for scaled particles alter more when the scale is below 1 (ii=100), parallel to the eccentricity, formfactor, solidity and compactness values.
All distance quantiles and moments have a significant ratio in Table 2, both for rotation and scaling. Observing the graphs in Appendix A.4.2, we notice a consistent increase in the feature value for larger scaling values. This proves that the distance quantiles and moments strongly rely on the size of the particle, which contradicts the assumed scale-invariance property.

Since many of the features oscillate in feature value when the scale is below 1, we calculate new ratio range values which only contain scaling values above 1 (see Table 2). The observed oscillation could be due to the fact that the amount of pixels of the particle is too low to properly determine the feature of the particle.

As shown in Table 2 for most features the ratio range scaling values are insignificant when the scale is below 1, which confirms the scale invariance of these features. Nonetheless the formfactor is still above 0.05, thus significant, for the scaling feature values.
To summarize, the distance features and formfactor value will be removed as features since both have a significant ratio between the feature range of data table $\tilde{\mathcal{D}}_s$ and feature range of data table $\tilde{\mathcal{D}}$ even after removing scale factors below 1 (which gave trouble for other features). Furthermore the 4_{th} moment of HU is removed from the set of features because of the significant ratio between the feature range of data table $\tilde{\mathcal{D}}_r$ and feature range of data table $\tilde{\mathcal{D}}$.

6.3.2 Discussion

As mentioned, some of the features showed different values when the particle was rotated except for when the rotation was 90 degrees or a multiple thereof. This can be explained by noting that the MATLAB function 'imrotate' used for rotating the particle, alters the contour of the particle a bit upon rotation. This hypothesis is confirmed by looking at a specific particle. This is visualized in Figure 18. Figure 18 the particle that is rotated 90 degrees shows a smooth edge on the upper side of the image, while for other rotations the same edge seems jagged. This will have an indisputable effect on the roughness features as well as the ellipse error feature.

	Formfactor	Solidity	Compactness	Ellipse error	Eccentricity
Ratio rotation	0.0987	0.0062	0.011	0.0037	0.0052
Ratio scaling	1.2537	0.0446	0.0617	0.3319	0.1305
Ratio scaling (scale >1)	0.113	0.0024	0.0126	0.004	0.0075
	HU 1	HU 2	HU 3	HU 4	HU 5
	7.24859E-05	3.17882E-05	0,0014	0,0706	1,57724E-05
	0,0596	0,0159	0.1244	0.1224	0.2383
	0.0005	0.0001	0.0005	0.001	3.55086E-06
	HU 6	HU 7	P(X<a) = 0.0	P(X<a) = 0.05	P(X<a) = 0.1
	0.0002	0.0001	0.1643	0.1582	0.1296
	0.2233	0.1434	0.5704	0.8926	0.8593
	9.86146E-05	8.64905E-06	0.5645	0.6795	0.6154
	P(X<a) = 0.15	P(X<a) = 0.2	P(X<a) = 0.25	P(X<a) = 0.3	P(X<a) = 0.35
	0.1149	0.1090	0.0932	0.0847	0.0807
	0.8320	0.8390	0.8111	0.7709	0.7467
	0.5717	0.5777	0.5534	0.5187	0.4999
	P(X<a) = 0.4	P(X<a) = 0.45	P(X<a) = 0.5	P(X<a) = 0.55	P(X<a) = 0.6
	0.0728	0.0639	0.0536	0.0463	0.0437
	0.7535	0.7451	0.7297	0.7169	0.7079
	0.5105	0.5087	0.4965	0.4874	0.4784
	P(X<a) = 0.65	P(X<a) = 0.7	P(X<a) = 0.75	P(X<a) = 0.8	P(X<a) = 0.85
	0.0424	0.0455	0.0490	0.0498	0.0533
	0.7012	0.7171	0.7357	0.7659	0.8366
	0.4691	0.4869	0.5006	0.5169	0.5703
	P(X<a) = 0.9	P(X<a) = 0.95	P(X<a) = 1.0	Mean(X)	Variance(X)
	0.0601	0.0476	0.0596	0.0490	0.0227
	0.8837	0.8742	0.8695	0.7743	0.6058
	0.5964	0.5845	0.6004	0.5225	0.5392
	Skewness(X)	Kurtosis(X)			
	0.1698	0.1421			
	0.4364	0.2624			
	0.1940	0.1788			

Table 2: The ratio between feature range of data tables $\tilde{\mathcal{D}}_r$ and $\tilde{\mathcal{D}}_s$ and the feature range of data table $\tilde{\mathcal{D}}$. Here X represents the distance vector of the FFT features as explained in section 4.3.3.

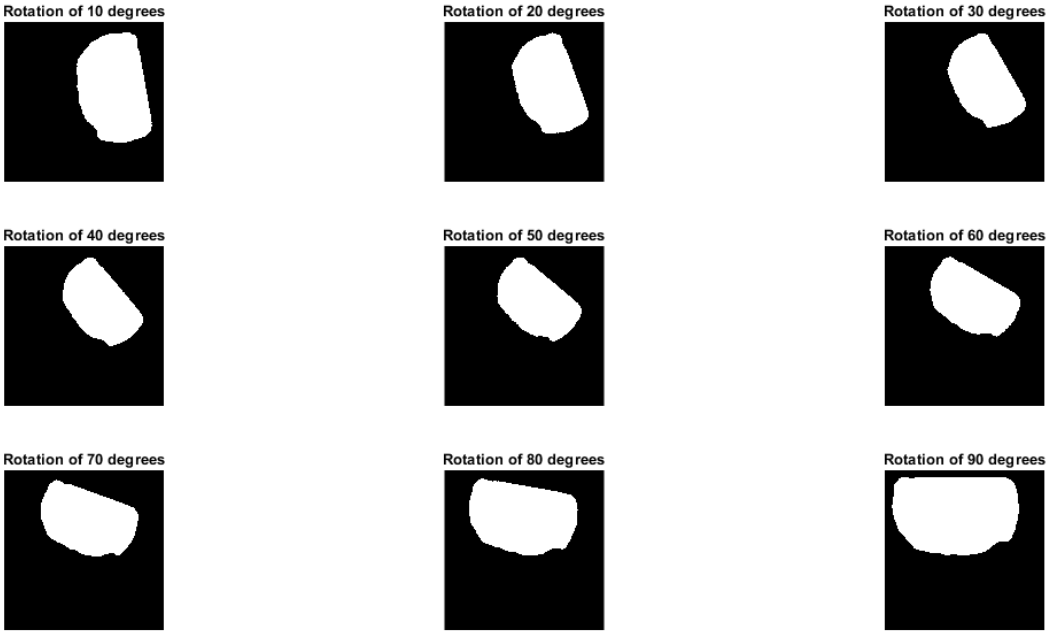


Figure 18: Particle under different rotations.

6.4 Clustering algorithms

In section 6.3 the features were extracted such that clustering algorithms can be applied. Appendix ... shows the score matrices after PCA for the data tables $\tilde{\mathcal{D}}$, $\tilde{\mathcal{D}}_s$ and $\tilde{\mathcal{D}}_r$.
From these score matrices and Table 3 it can be seen that we only need 5 components to explain 95% variance in data. The loading values per feature are visible in Figure 19 and the score values can be seen in Appendix A.5.1.

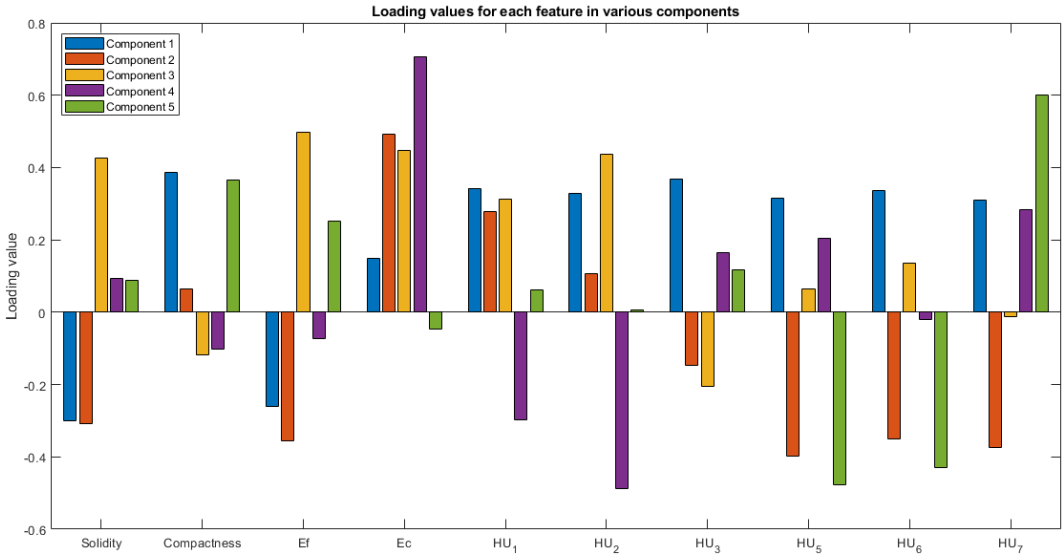


Figure 19: Loading values per feature

	Component 1	Component 2	Component 3	Component 4	Component 5	Component 6	Component 7	Component 8	Component 9	Component 10
Percentage variance	58.7837	16.8911	11.6550	4.4278	3.8874	2.2237	0.8744	0.7309	0.3491	0.1770

Table 3: Percentage of variance explained by each of the components

As illustrated in Table 3 and pointed out in section 4.4 the first component explains most of the variance. The absolute value of the loadings of the first component are relatively similar for all features, with the exception of the eccentricity, which means that the contribution to the first component for these features is high. When looking at the Figures in Appendix A.4.1 we can explain the negative values for the first component of the Solidity and Ellipse error values, by considering that (in contrary to the other values) the lower the value the more irregular the particle is. Furthermore, the loading of the second up to the fourth component for the eccentricity is, in comparison to the other loading values for the second component, very significant. Therefore all features play a significant role in the explanation of the variance in the data.
When observing the Figures in Appendix A.5.1, no distinguishable clusters are visible yet. However the scaled and rotated data show to be compact within the range of the first, second and third components, with some exceptions. This is important for the scale/rotation invariant assumption. However the fourth and fifth component show some more spread data points. The scale/rotation invariant assumption is discussed further in section 6.4.2.

6.4.1 Evaluation number of clusters

In order to determine the number of clusters and to investigate whether k-means or Gaussian mixture models work best for clustering the results, we used both methods to cluster the results (see Figure 20 and 22 respectively).

K-means The 'elbow' in Figure 20 is around 7 clusters, since the decline in total distance is a lot more gradual after 7 clusters.
The I-index, CH and DB values are inconclusive as the I-index and CH get gradually better and the DB

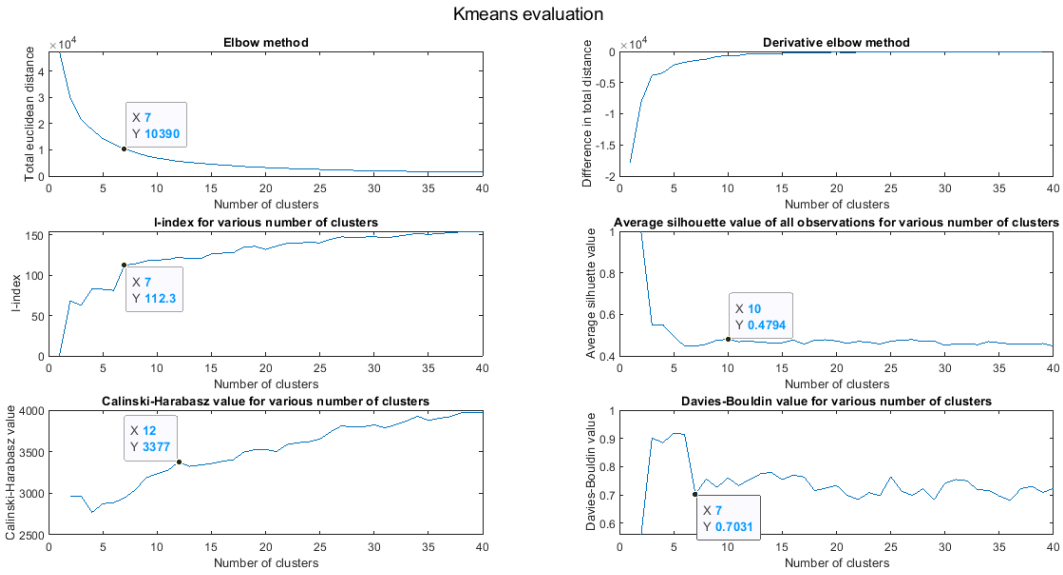


Figure 20: k-means evaluation for various number of clusters

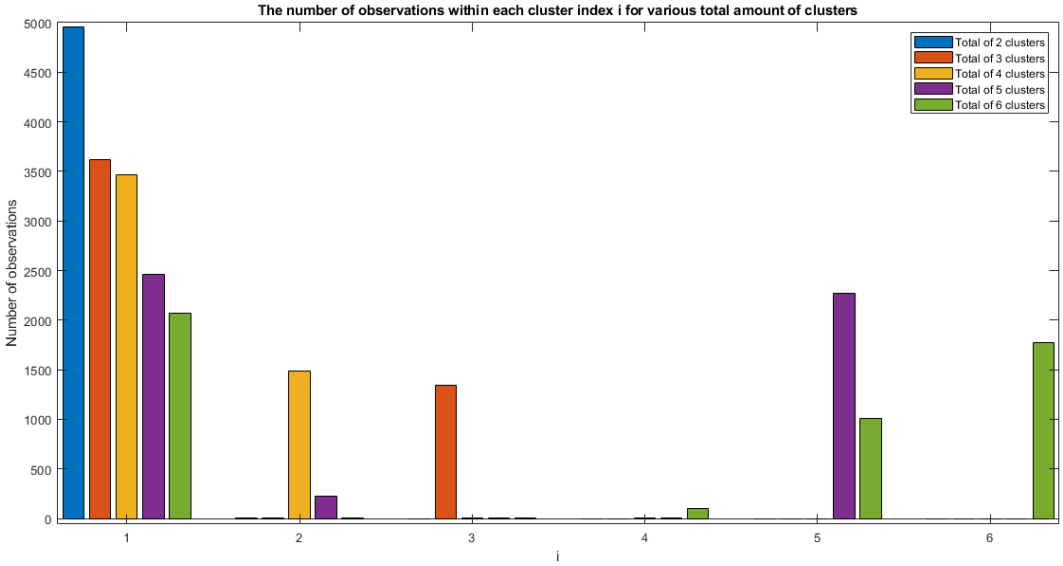


Figure 21: Number of observations within each cluster for various amount of clusters in k-means

values oscillate between 0.7 and 0.8 as the amount of clusters inclines. Considering we suspect the optimal amount of clusters is 7, given the elbow method, we look at the first local maximum/minimum after 6 clusters (maximum for I-index and CH values and minimum for DB values). These are indicated in Figure 20.
The average silhouette values indicate that the optimal amount of clusters is 2. In Figure 21 we notice that the second cluster, when considering 2 clusters, contains little to no observations. Therefore 2 clusters is not a favorable option, as it does not actually split the data. The next highest silhouette values are for 3 or 4 clusters. However, these options only split the data in two significant clusters (see Figure 21) and the silhouette value of 3-4 clusters does not differ much with the next highest silhouette value at 10 clusters. Therefore 10 clusters is chosen as a more favorable option.
To conclude, the most interesting amount of clusters for k-means is 7, 10 and 12. Since 12 clusters gives the best results in most evaluation methods, samples of the images of each cluster are presented in appendix A.7.1.

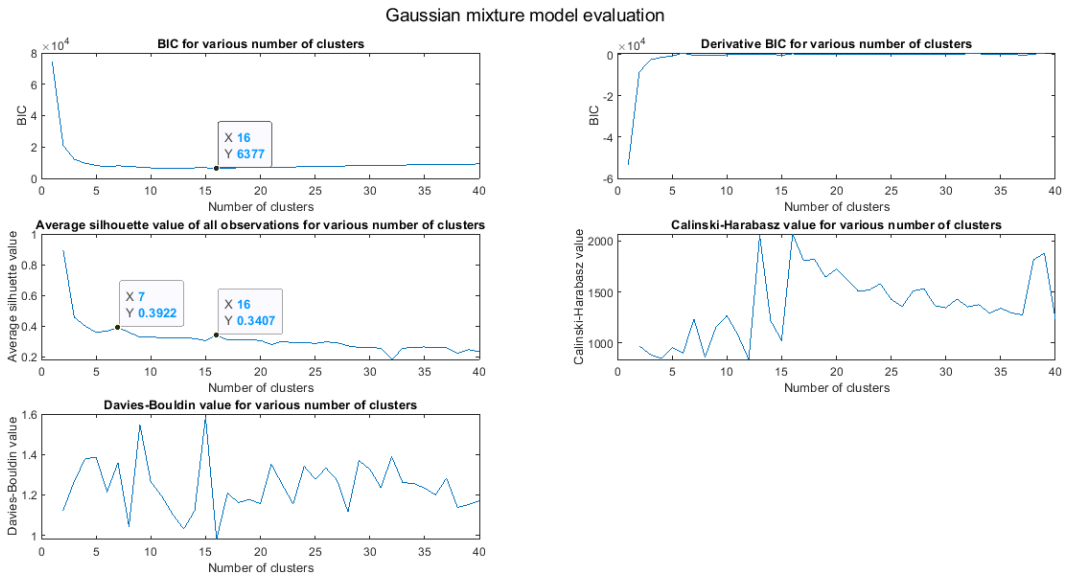


Figure 22: Gaussian mixture models evaluation for various number of clusters

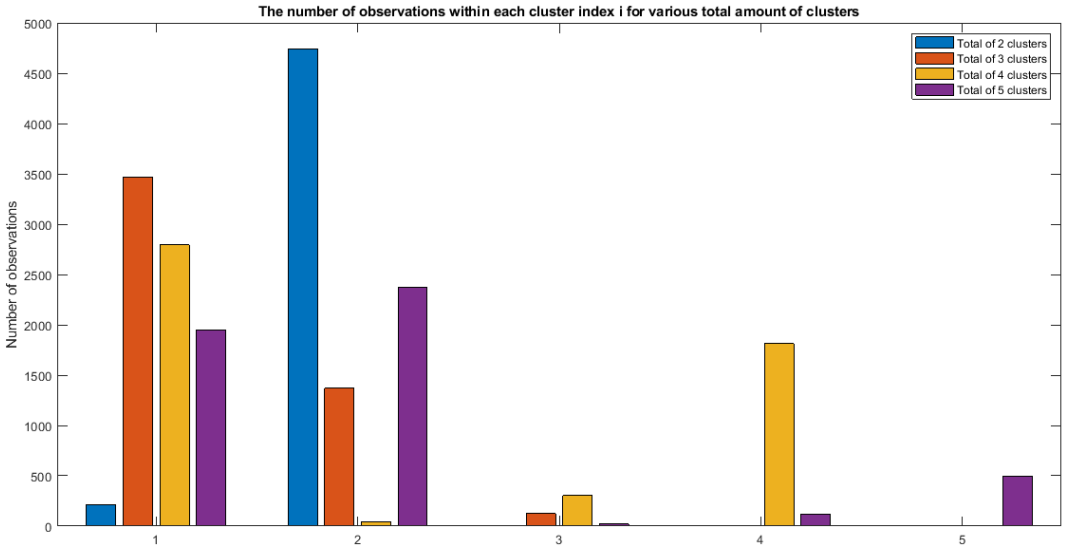


Figure 23: Number of observations within each cluster for various amount of clusters in GMM

Gaussian mixture models Each evaluation method, with the exception of the silhouette, consequently gives 16 clusters as the optimal result. The BIC, CH and DB values for 13 clusters are very similar to the respective values for 16 clusters, which makes 13 clusters an appealing option.
The silhouette values are best for 2-4 clusters. However, these options only split the data in one or two significant clusters (see Figure 23) and the silhouette value of 3-4 clusters does not differ much with the next highest silhouette value at 7 clusters. Therefore 7 clusters is chosen as a more favorable option.
To conclude, the most interesting amount of clusters for GMM is 7, 13 and 16. Since 16 clusters gives the best results in most evaluation methods, samples of the images of each cluster are presented in appendix A.7.2.

Now that we've decided upon some preferred amount of clusters for both the k-means and Gaussian mixture models clustering method, we assemble the corresponding evaluation values in Table 4 in order for us to decide which clustering method gives the best results. Further it gives an overview of the best clustering

results.

	Silhouette	Calinski-Harabasz	Davies-Bouldin	Total distance	I-index	BIC
kmeans, k=7	0.4478	2943.8187	0.7031	10387.4788	112.3290	NaN
kmeans, k=10	0.4794	3234.2605	0.7599	6893.4653	118.7719	NaN
kmeans, k=12	0.4716	3377.0604	0.7550	5575.0433	122.3895	NaN
GMM, k=7	0.3922	1234.8728	1.3588	NaN	NaN	8083.6885
GMM, k=13	0.3270	2055.8553	1.0308	NaN	NaN	6582.2426
GMM, k=16	0.3407	2068.3875	0.9787	NaN	NaN	6377.2913

Table 4: Overview of the leading number of clusters for both kmeans and Gaussian mixture models

Each of the possible amount of clusters for k-means has better evaluation values with respect to the Gaussian mixture model clusterings (seen in Table 4). This matter is debated further in the discussion of this section. The favorable clusterings from Table 4 are visible in Appendices A.6.1 and A.6.2, by giving each cluster a different color in the score data.

6.4.2 Evaluation scale/rotation invariant property method 2

In Figures 24 and 25 the bars at the correct cluster index have a black outline. In Figure 24 only 2 scaled particles are not in the original cluster for each of the k-means clusterings. The wrongly clustered particles had a scale of 0.01 and 0.02, which were the two lowest scaling values. Since the lowest gave the most dissimilar feature values (see section 6.3), it is logical that the particle scaled with 0.01 or 0.02 is clustered differently.

In Figure 25 there are some wrongly clustered scaled particles, of which the scale is one of the lowest scaling values, that are clustered wrongly. However for 16 clusters when the scale is 2, the particle is clustered wrong as well. Moreover some of the rotated particles are in the wrong cluster.

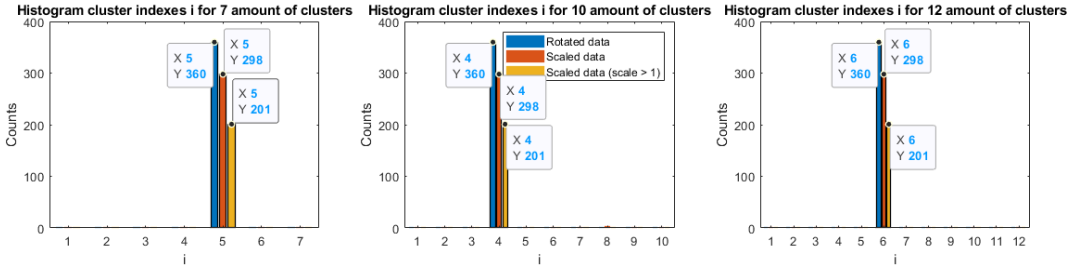


Figure 24: Cluster index for rotated/scaled data after k-means clustering

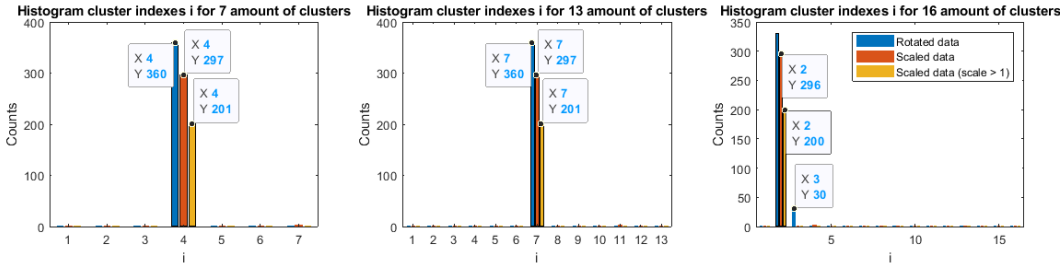


Figure 25: Cluster index for rotated/scaled data after GMM clustering

In order to understand what goes wrong here, we want to know which component causes this error such that we can hopefully link it to a certain feature. Therefore we use the obtained Gaussian models for 16 clusters and limit it to one component space. Using only one component, we compute the percentage of wrongly clustered scaled/rotated particles, i.e. the error.

	Component 1	Component 2	Component 3	Component 4	Component 5
Error for rotated particles	0	0	0	0	0.8528
Error for scaled particles (scale >1)	0	0	0	0	0.6965

Table 5: Percentage of wrongly clustered rotated/scaled particles when considering only one component space

From Figure 5 it becomes clear that the error for rotation and scaling is 0 for all components except the 5_{th}. Therefore the 5_{th} component causes the rotated and scaled particles to be wrongly clustered. Therefore we take a closer look at the dominant features of the fifth component. Looking at the loading values in Figure 19 the features that have a notable loading value for component 5 are compactness, ellipse error, HU_5, HU_6 and HU_7. For these features we compare the original feature values of the particle without rotation or scaling to the wrongly clustered particles with a rotation or scaling by calculating the distance between the feature values obtained from data tables $\tilde{\mathcal{D}}$, $\tilde{\mathcal{D}}_s$ and $\tilde{\mathcal{D}}_r$. Since the data tables $\tilde{\mathcal{D}}$, $\tilde{\mathcal{D}}_s$ and $\tilde{\mathcal{D}}_r$ are normalized we can compare the distance values and see which feature gives the largest distance for the rotated particle.

	Compactness	Ellipse error	HU 5	HU 6	HU 7
33	0.4250	0.0058	1.24E-06	0.0017	0.0003
34	0.3979	0.0269	2.78E-05	0.0012	0.0002
35	0.4272	0.0204	0.0002	0.0044	0.0002
56	0.4051	0.0187	1.24E-05	0.0018	0.0012
58	0.4318	0.0226	1.86E-05	0.0030	0.0011
67	0.4084	0.0066	0.0001	0.0065	0.0007
70	0.4236	0.0023	3.46E-05	0.0019	0.0013
123	0.4250	0.0094	2.35E-06	0.0026	0.0020
124	0.3979	0.0336	8.82E-06	0.0028	0.0020
125	0.4272	0.0240	0.0001	0.0065	0.0014
146	0.4051	0.0258	0.0001	0.0045	0.0004
148	0.4318	0.0332	0.0002	0.0060	0.0004
157	0.4084	0.0127	0.0001	0.0063	0.0004
160	0.4236	0.0123	8.90E-05	0.0032	0.0005
213	0.4250	0.0060	1.64E-05	0.0015	0.0006
214	0.3979	0.0243	4.70E-06	0.0013	0.0006
215	0.4272	0.0156	8.83E-05	0.0044	0.0004
236	0.4051	0.0242	2.57E-05	0.0018	0.0018
238	0.4318	0.0217	0.0001	0.0039	0.0016
247	0.4084	0.0090	4.38E-05	0.0027	0.0011
248	0.3841	0.0158	0.0002	0.0021	0.0012
250	0.4236	0.0079	3.83E-05	0.0014	0.0013
303	0.4250	0.0022	0.0004	0.0057	0.0036
304	0.3979	0.0260	0.0004	0.0055	0.0035
305	0.4272	0.0139	0.0001	0.0004	0.0025
326	0.4051	0.0163	0.0001	0.0037	0.0004
328	0.4318	0.0241	2.02E-05	0.0018	0.0005
337	0.4084	0.0056	1.72E-05	0.0007	0.0003
338	0.3841	0.0131	4.74E-05	0.0032	0.0005
340	0.4236	0.0093	0.0001	0.0047	0.0004

Table 6: Distance in feature values between original particle and particle under various rotation angles. The first column represents the different rotation angles

	Compactness	Ellipse error	HU 5	HU 6	HU 7
2	0.4654	0.0131	1.1754E-05	0.0009	0.0001

Table 7: Distance in feature values between original particle and particle under scaling. The first column represents the scale

In Tables 6 and 7 the highest distance for all the particles is highlighted which is the compactness feature. Therefore we suspect that the compactness plays a role in the wrongly clustered particles. To support this hypothesis, we set the loading value of the compactness feature for the 5_{th} component to 0, then calculate the new score values of the rotated particles and the original particle, and use the GMM clustering for 16

clusters to determine the new cluster indexes. As we can see in Figure 26 the new cluster index is again 2, the rotated data is all clustered correctly and the scaled particle with scaling value 2 is also clustered correctly. This supports the previous hypothesis that the compactness feature may cause the wrongly clustered rotated/scaled particles.

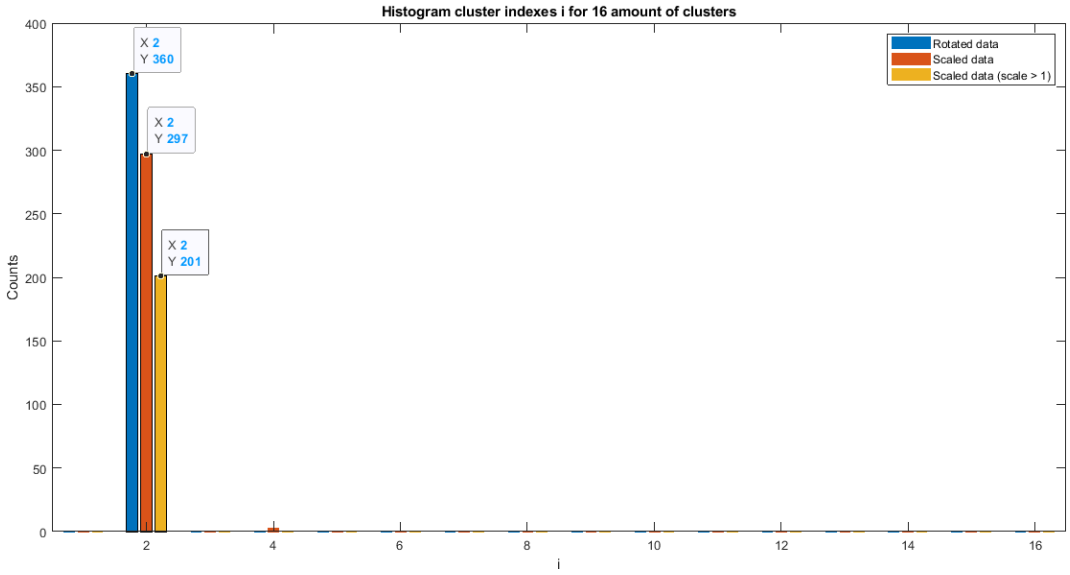


Figure 26: Cluster index for rotated/scaled data after GMM clustering with the loading value of the compactness feature set to 0 for the 5_{th} component

6.4.3 Discussion

It is not a big revelation that the compactness feature caused some rotated/scaled particles to be clustered improperly, since it correlates strongly with the formfactor which was removed as feature because of the same reasoning. Additionally in Table 2 the compactness scored the highest of the remaining features. Lastly the compactness value for scaled particles gives a high unexplained value in Appendix A.4.2, which explains the wrong cluster for the scaled particle with scaling factor 2.
However this doesn't necessarily imply that the compactness value should be left out. As was seen in section 6.3.2 the values differ for certain rotations because of the way the rotated particle is computed (seen in Figure 18).

As mentioned before, there is no clear separation within the data and seen in the component values in Appendix A.5.1. Therefore there is no clear answer to the question 'what is the optimal amount of clusters' that can be observed with the naked eye. Either all particles are very similar or the feature extraction method lacks some features that better separates the various particles.
The k-means clustering method scored better than Gaussian mixture models clustering for each of the common evaluation methods. Nonetheless, there was no optimal amount of clusters within the k-means method for the evaluation methods I-index and CH since the values kept increasing as the amount of clustered advances. Using GMM clustering however, we did see an optimal amount of 16. Also when looking at the definitions of these evaluation methods as spoken about in section 4.1.3 the k-means method might have an advantage compared to the Gaussian mixture models, since the silhouette analysis, CH index and DB criterion all consider either the distance between the clusters or the distance within the clusters. In these evaluations either the distance between the observation and the cluster centroid/other observations should be minimized within the cluster or the distance between observations or centroids of various clusters should be maximized. Since the clusters of GMM might overlap and the GMM method considers variance within the data these evaluation methods might disadvantage the GMM method beforehand. A solution should be a different evaluation method o at least keeping this in mind when comparing the k-means method and the GMM method.

7 Conclusion

In section 2 we defined the following main question:

> *What are the advantages and disadvantages of two clustering methods with different cost-functions and how can this be evaluated?*

The general advantages of k-means were briefly discussed in section 4.1.1. These were the relatively low complexity and the generally high computing efficiency. Disadvantages are that the method is sensitive to outliers and works best for well separated data. There were some outliers in the data and the data was rather accumulated (see Figures of score values in appendix A.6.1). This disadvantage is retrievable in the evaluation of k-means in Figure 20, since:

1. The silhouette evaluation is best for low amounts of clusters but these clusterings only separate the outliers;
2. The I-index, Calinski-Harabasz and Davies-Bouldin evaluation are inconclusive due to the accumulated data.

Nonetheless the best evaluation values for k-means are higher than the best evaluation values for GMM for each of the evaluations that were applied to both cluster methods. In section 4.1.2 the advantages of the

Gaussian mixture models were spoken about. In contrast to the k-means method, the GMM is able to find underlying distribution of accumulated data. This was noticed in the evaluation of the method in Figure 22, where all, except the silhouette evaluation, evaluation methods pointed to the optimum of 16 clusters. Nonetheless the evaluation values for 16 clusters with the GMM method are still slightly poorer than the best clustering of the k-means method.
Therefore either the k-means method is a better method to the data or the evaluation methods are better suited for the k-means method than the GMM method. Discussion on this matter can be found in **??**.

Besides the main question, we considered mutliple sub-questions. All of these are answered accordingly:

1. *Which part of the raw data is usable for this study and how can this be evaluated?*
2. *What is the best method to segment the particle in the image from the background?*
3. *What are the most interesting morphological features of the segmented particle?*
4. *What is the best suited clustering according to various evaluation methods?*

1. Splitting (un)usable images The first step of the process, splitting of (un)usable images, showed very promising results. Group 1, which was then used in the analysis, contained images that contained a particle with a clear-cut shape outline. However a few particles within this group were still a bit vague (e.g. Figure 32) and/or were only partly visible (e.g. Figure 35). Since we only look at the shape of the particle in this study and we furthermore removed the images with partly visible particles, these particles did not affect the study. However for further research that considers the texture of the particle, this may become an issue. Lastly the amount of groups in the first step could be limited to two, considering that extraction of features is complicated for too small particles as seen in section 6.3.1. Therefore the groups 2 and 3 can be combined, as the group 2 images may give ambiguous results.

2. Image segmentation The second step, image segmentation, proposed a new and better method to segment the particle out of the image using geodesic active contours. Visually the particles seemed to be segmented well, but the method does use a smoothing factor which may effect the roughness of each particle. This issue should be addressed in future research.

3. Feature extraction In the third step of the process, feature extraction, we noticed that almost all roughness descriptors had the same rough particle outliers while the other data was a bit cluttered as seen in appendix A.4.1. This was also visible in the scoring values in appendix A.5.1. Therefore the current features

may not have been sufficient to split the data. Some features were removed due to suspicion in scaling and rotation variance. In section 7.1 some new features are proposed.

4. Clustering Most of the evaluation methods in the clustering section, which is the last step of the process, implied that 12 clusters was the optimal amount of clusters for the k-means method when considering only 7, 10 and 12 as options. Nonetheless the evaluation values of the I-index, CH and DB improved as the amount of clusters inclined. The GMM method suggested an optimal solution of 16 clusters, while 13 clusters was a close second as the evaluation values of 13 clusters were close to those of 16 clusters. According to silhouette evaluation 7 clusters was a better option than 13 and 16.

7.1 Recommendations

Overall recommendation zal zijn om nog meerdere clustering methods te bekijken

Below some recommendations are given for further research. These are split into the four steps of the case study, each referring back to a sub-questions: splitting (un)usable images, image segmentation, feature extraction and clustering

1. Splitting (un)usable images As mentioned above, the groups 2 and 3 can be merged. Plus the splitting method could use an enhancement that determines whether the particle contained in the image has a clearly visible texture. A proposed method is to segment the image and look at the pixel-values within the segmentation. Another option is to consider more images of the same particle which may include a better visible texture.

2. Image segmentation The active contour method can be adapted to exclude the smoothing of the particles boundary. This will likely result in better distinction in the roughness of each particle. The segmentation method proposed in this study can be used to detect an initial contour.

3. Feature extraction Since some features that considered the roughness of the particle were dependent the size and/or orientation of the particle, new roughness descriptors could be examined as advanced features. Drolon et al. [11] propose the harmonic wavelet transform as a new roughness descriptor for sediment particles. The data structure that they use is very similar to the structure of our data, which may imply that the roughness descriptors can be beneficial to our study as well. Moreover there are multiple roughness descriptors in "The image processing handbook" that can improve the results [28]. For example, different definitions of compactness are proposed, and it is possible that some of them are more suitable for the specific application.

4. Clustering It is recommended to first perform the feature extraction and clustering algorithms on more images (as we only used 5.000 of 27.779 images now) in order to know whether the results of the clustering are somewhat similar and whether the k-means method is unambiguous about the optimal amount of clusters. Furthermore other clustering methods can be considered as there is no determined wrong or right clustering method. To quote Anil K Jain writer of the paper "Data clustering: 50 years beyond K-means": "The above discussion underscores one of the important facts about clustering; there is no best clustering algorithm. Each clustering algorithm imposes a structure on the data either explicitly or implicitly. When there is a good match between the model and the data, good partitions are obtained. Since the structure of the data is not known a priori, one needs to try competing and diverse approaches to determine an appropriate algorithm for the clustering task at hand. This idea of no best clustering algorithm is partially captured by the impossibility theorem [17], which states that no single clustering algorithm simultaneously satisfies a set of basic axioms of data clustering." [16]

References

[1] About asml — supplying the semiconductor industry. `https://www.asml.com/en/company/about-asml`.

[2] Asml technology — supplying the semiconductor industry. `https://www.asml.com/en/technology`.

[3] Segment image into foreground and background using active contours (snakes) region growing technique - MATLAB activecontour - MathWorks Benelux.

[4] Hervé Abdi and Lynne J. Williams. Principal component analysis, 7 2010.

[5] Ilmari Ahonen, Ville Härmä, Hannu-Pekka Schukov, Matthias Nees, and Jaakko Nevalainen. Morphological Clustering of Cell Cultures Based on Size, Shape, and Texture Features. *Statistics in Biopharmaceutical Research*, 8(2):217–228, 4 2016.

[6] Alan Bovik. *Handbook of Image and Video Processing*. Elsevier Inc., 2005.

[7] Vicent Caselles, Ron Kimmel, and Guillermo Sapiro. Geodesic Active Contours. *International Journal of Computer Vision*, 22(1):61–79, 1997.

[8] Da Chen and Laurent D. Cohen. From active contours to minimal geodesic paths: New solutions to active contours problems by Eikonal equations. In *Handbook of Numerical Analysis*, volume 20, pages 233–271. Elsevier B.V., 1 2019.

[9] Krzysztof J. Cios, Witold Pedrycz, and Roman W. Swiniarski. Data Mining and Knowledge Discovery. In *Data Mining Methods for Knowledge Discovery*, pages 1–26. Springer US, Boston, MA, 1998.

[10] E.R. Davies. Binary shape analysis. In *Computer Vision*, pages 203–238. Elsevier, 1 2018.

[11] H. Drolon, F. Druaux, and A. Faure. Particles shape analysis and classification using the wavelet transform. *Pattern Recognition Letters*, 21(6-7):473–482, jun 2000.

[12] Vladimir Estivill-Castro. Why so many clustering algorithms. *ACM SIGKDD Explorations Newsletter*, 4(1):65–75, 6 2002.

[13] Chris Fraley and Adrian E Raftery. How Many Clusters? Which Clustering Method? Answers Via Model-Based Cluster Analysis. Technical report.

[14] Walter Gander, Gene H. Golub, and Rolf Strebel. Least-squares fitting of circles and ellipses. *BIT*, 34(4):558–578, 12 1994.

[15] Ming Kuei Hu. Visual Pattern Recognition by Moment Invariants. *IRE Transactions on Information Theory*, 8(2):179–187, 1962.

[16] Anil K. Jain. Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, 31(8):651–666, 6 2010.

[17] Jon Kleinberg. An Impossibility Theorem for Clustering. Technical report.

[18] M Kyan, P Muneesawang, K Jarrah, and L Guan. Unsupervised Learning: A Dynamic Approach. 2014.

[19] E. J. Liu, K. V. Cashman, and A. C. Rust. Optimising shape analysis to quantify volcanic ash morphology. *GeoResJ*, 8:14–30, 12 2015.

[20] Ujjwal Maulik and Sanghamitra Bandyopadhyay. Performance evaluation of some clustering algorithms and validity indices. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(12):1650–1654, 12 2002.

[21] John McGarva and Glen Mullineux. Harmonic representation of closed curves. *Applied Mathematical Modelling*, 17(4):213–218, 4 1993.

[22] Fatima A. Merchant and Kenneth R. Castleman. Computer-Assisted Microscopy. In *The Essential Guide to Image Processing*, pages 777–831. Elsevier Inc., 1 2009.

[23] Subrajeet Mohapatra, Dipti Patra, and Sanghamitra Satpathy. Unsupervised Blood Microscopic Image Segmentation and Leukemia Detection using Color based Clustering. Technical report.

[24] FB Neal and JC Russ. Measuring shape. 2012.

[25] Ewa Pietka. Image standardization in PACS. In *Handbook of Medical Image Processing and Analysis*, pages 874–894. Elsevier Inc., 2009.

[26] Gopinath Rebala, Ajay Ravi, and Sanjay Churiwala. *An Introduction to Machine Learning*. Springer International Publishing, Cham, 2019.

[27] Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20(C):53–65, 11 1987.

[28] John C. Russ and F. Brent Neal. *The Image Processing Handbook*. CRC Press, 9 2018.

[29] Grant Sanderson. But what is a fourier series? from heat flow to circle drawings. `https://www.youtube.com/watch?v=r6sGWTCMz2k&ab_channel=3Blue1Brown`.

[30] Septyo, Irene and Panuju, Rini. Label efficient machine learning for multiclass image classification. (30), 9 2019.

[31] Inc. The MathWorks. *Image Processing Toolbox.* ***Funtions****: activecontour, imbinarize, imfill, imrotate, regionprops, wiener2*. Natick, Massachusetts, United State, 2019.

[32] Inc. The MathWorks. *Statistics and Machine Learning Toolbox.* ***Funtions****: cluster, evalclusters, fitgmdist, kmeans*. Natick, Massachusetts, United State, 2019.

[33] Dongkuan Xu and Yingjie Tian. A Comprehensive Survey of Clustering Algorithms. *Annals of Data Science*, 2(2):165–193, 6 2015.

Appendices

A Figures

A.1 Gradient image segmentation method

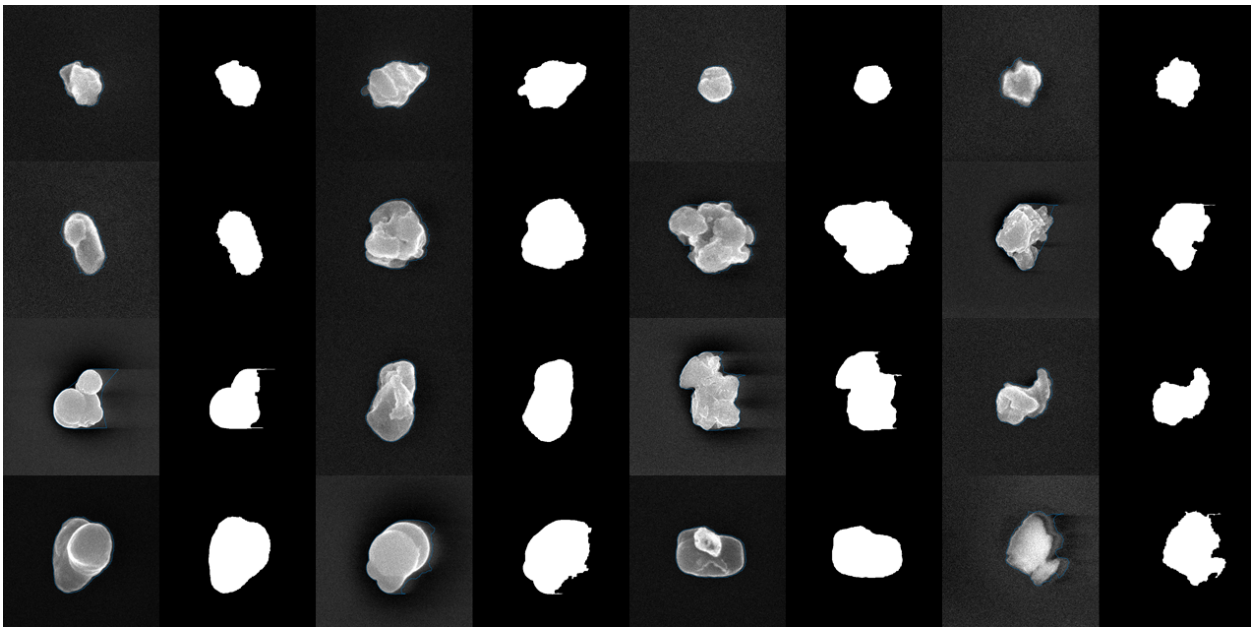


Figure 27: Sample gradient image segmentation method

A.2 Removed images

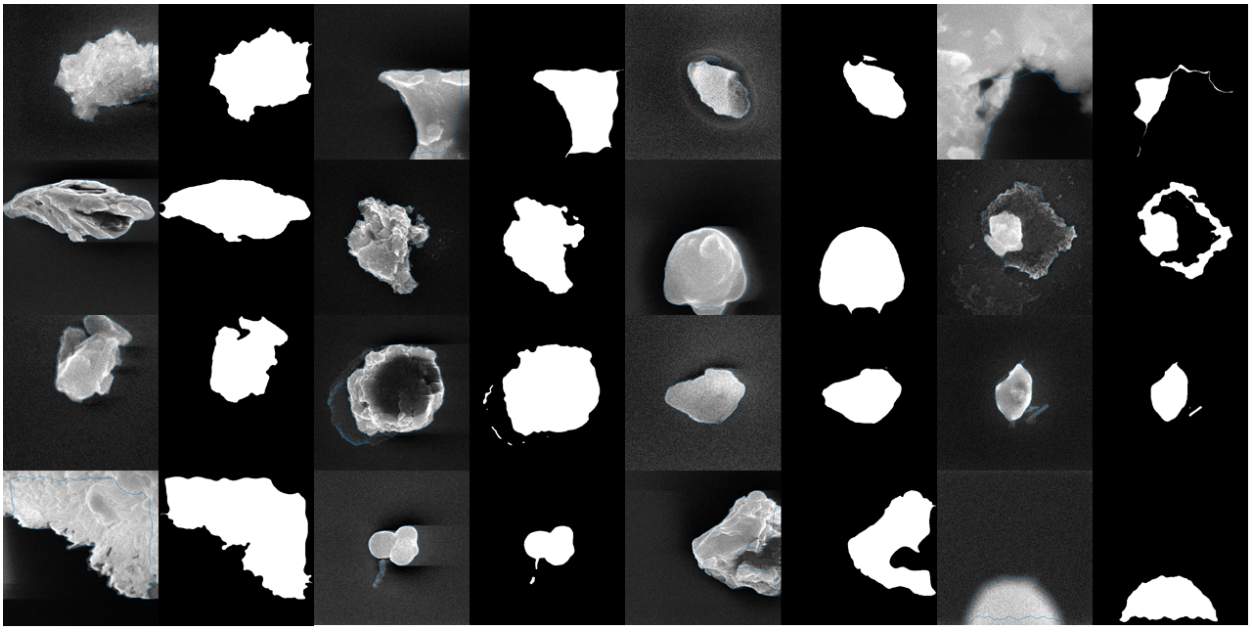


Figure 28: Removed images during image segmentation

A.3 Splitting (un)usable images

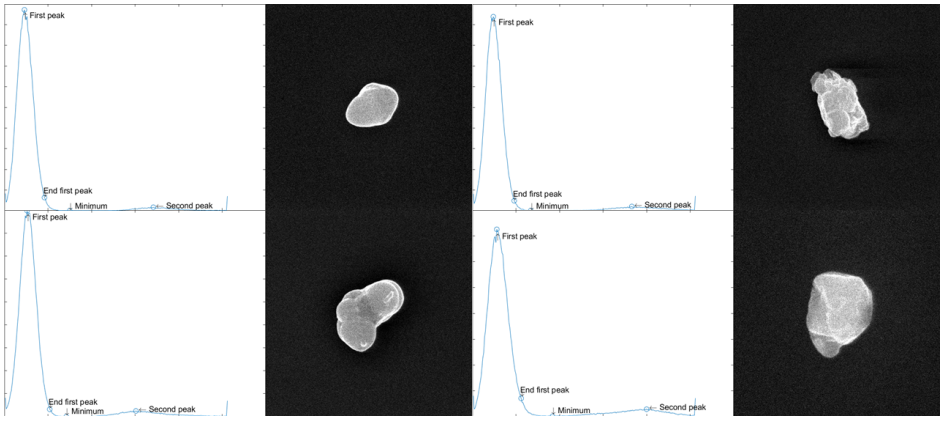


Figure 29: Sample images 1.1

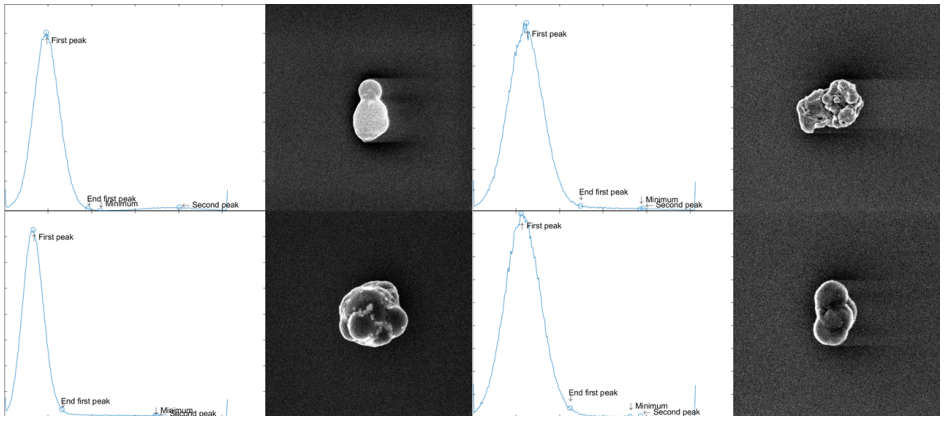


Figure 30: Sample images 1.2

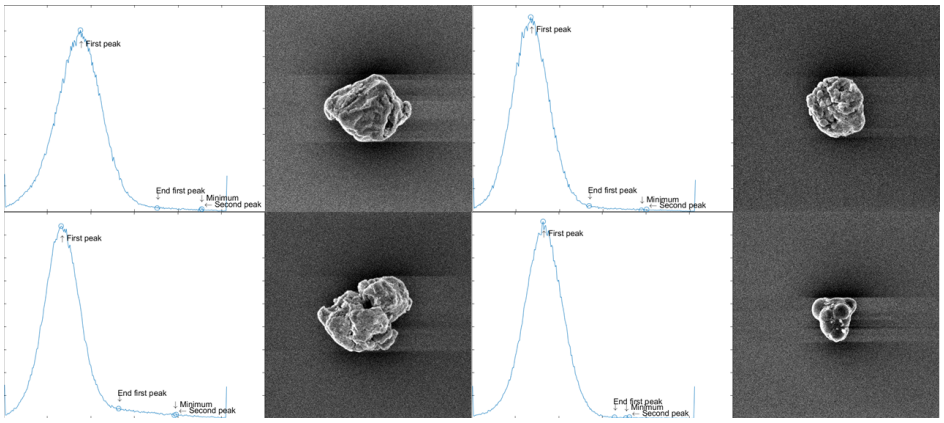


Figure 31: Sample images 1.3

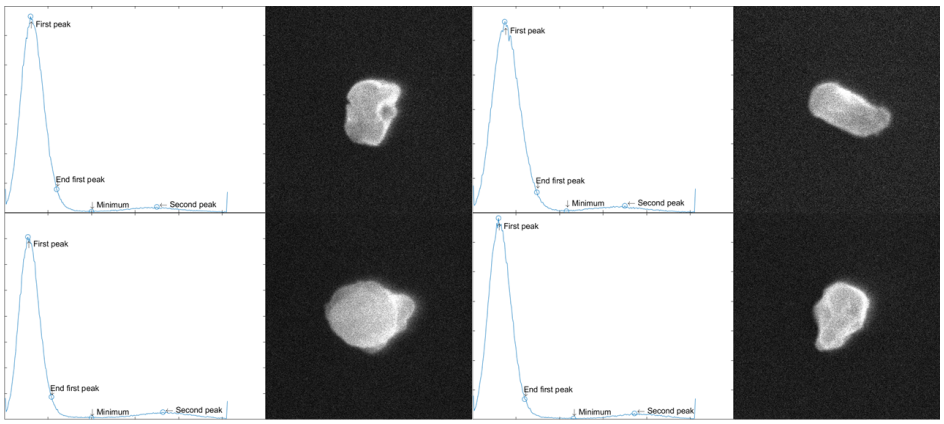


Figure 32: Sample images 2.1

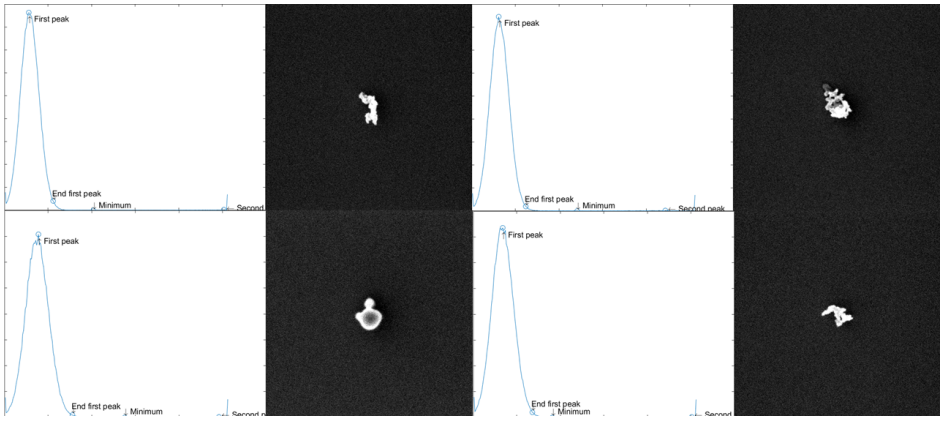


Figure 33: Sample images 2.2

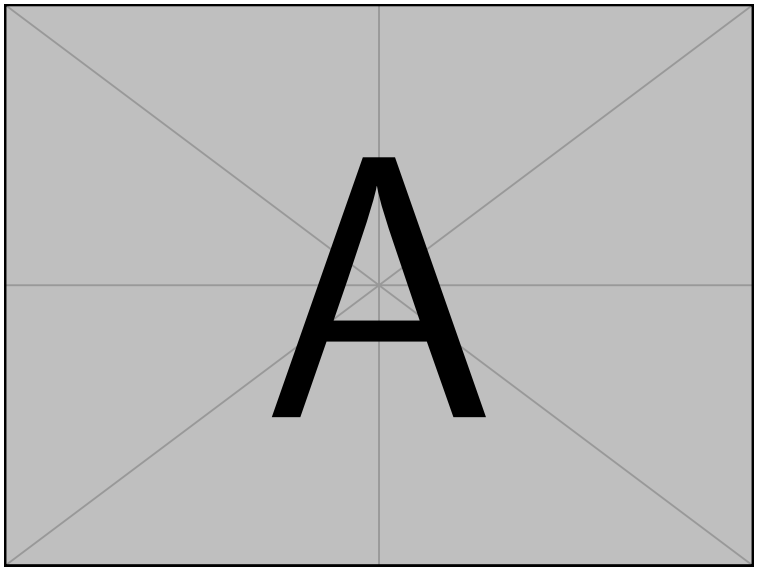


Figure 34: Sample images 2.3

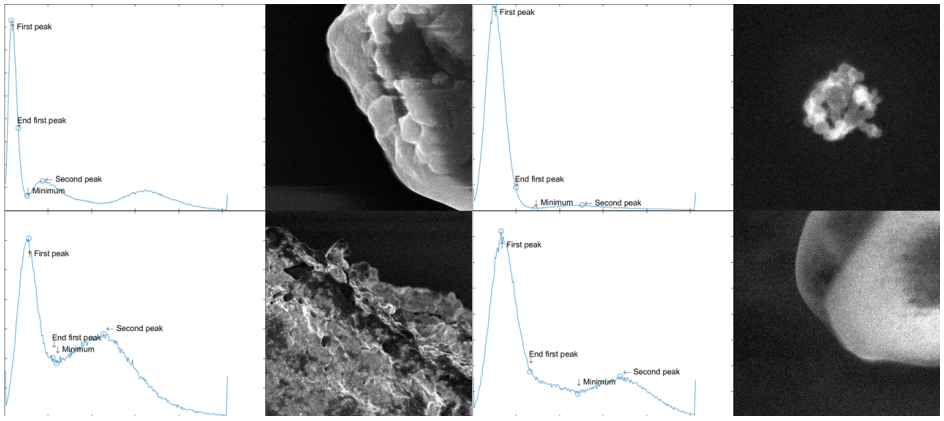


Figure 35: Sample images 3.1

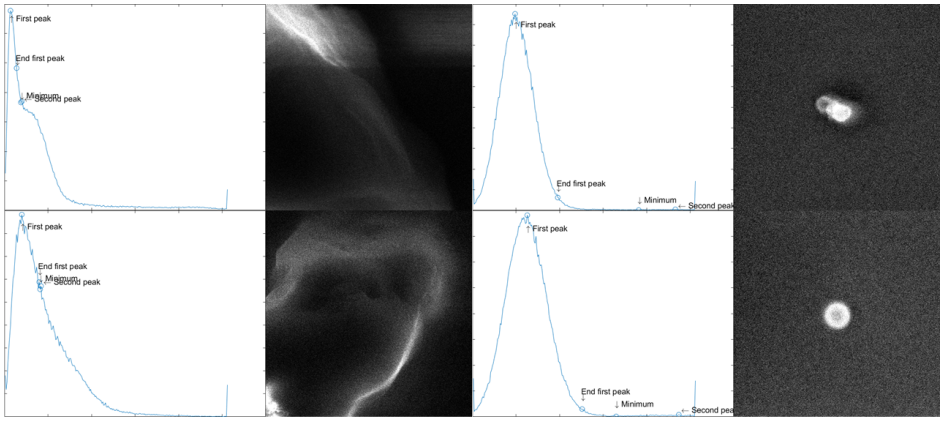


Figure 36: Sample images 3.2

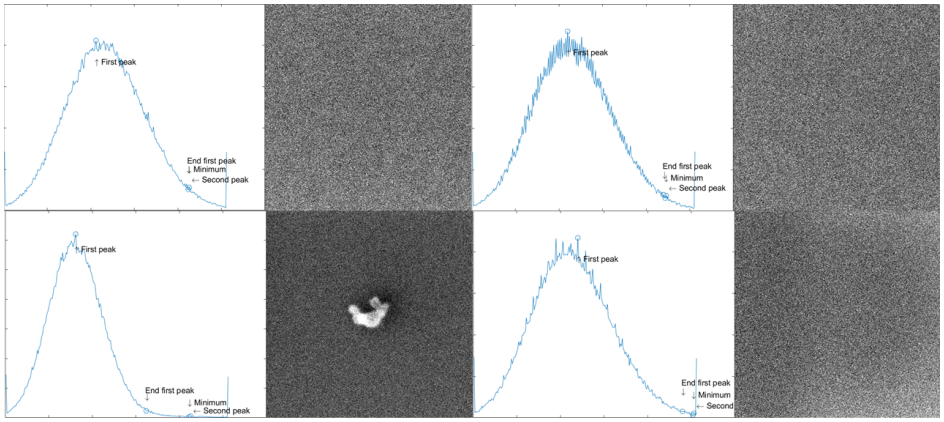


Figure 37: Sample images 3.3

A.4 Feature extraction

A.4.1 Features sample

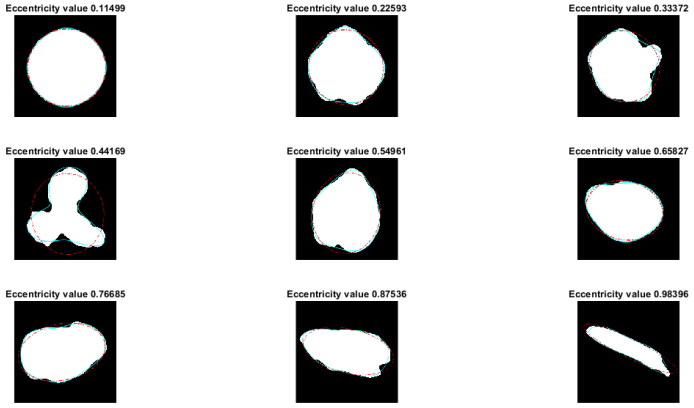


Figure 38: Sample images with different Eccentricity values

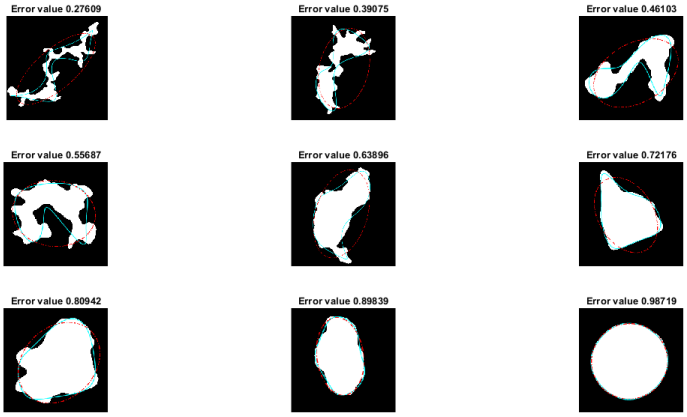


Figure 39: Sample images with different Ellipse error values

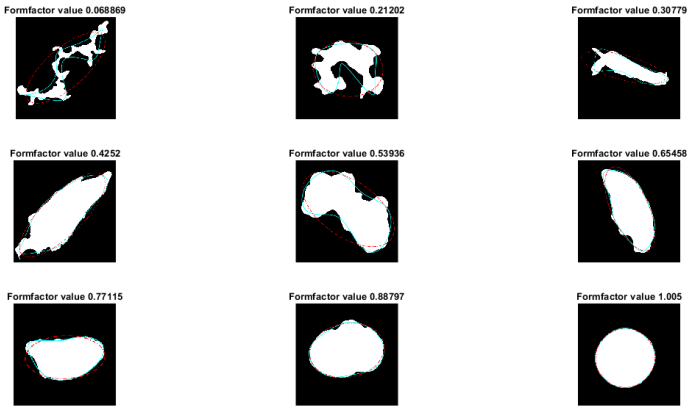


Figure 40: Sample images with different Formfactor values

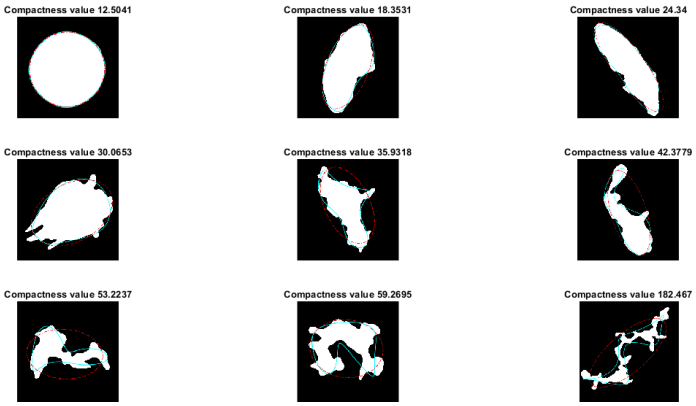


Figure 41: Sample images with different Compactness values

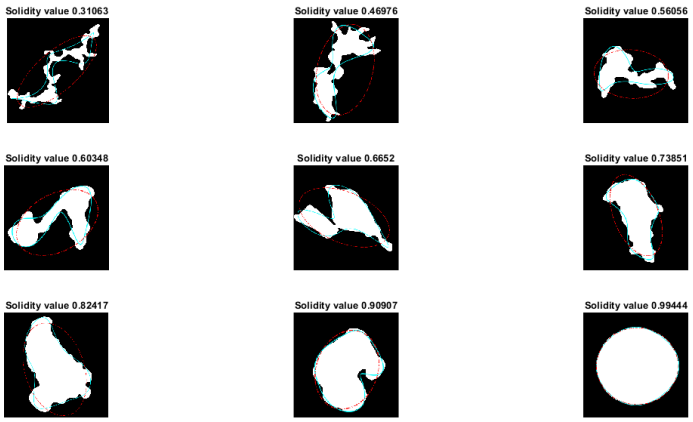


Figure 42: Sample images with different Solidity values

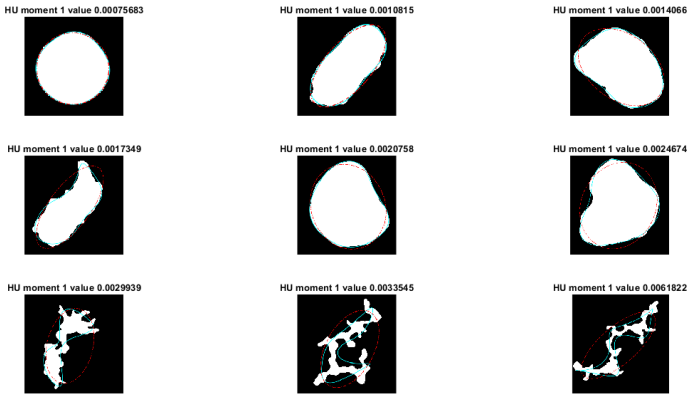


Figure 43: Sample images with different first HU moment values

Figure 44: Sample images with different values of 'a' where $\mathcal{P}(X < a) = 0.0$ and X is the distance factor between the original contour and the FFT estimate

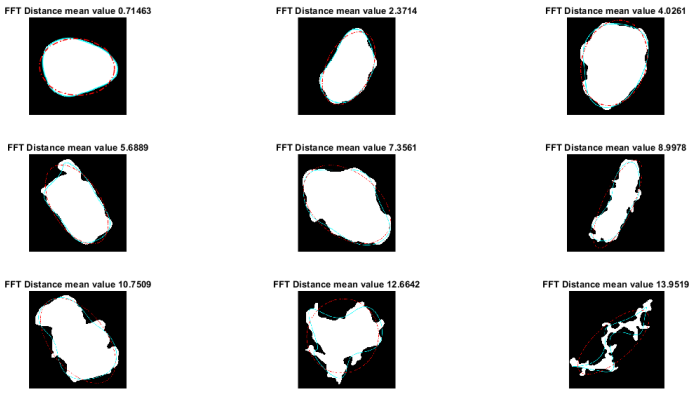
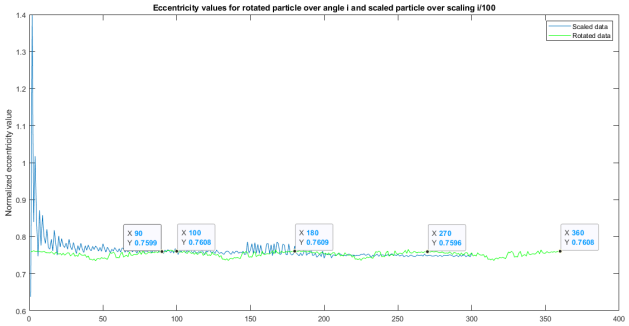
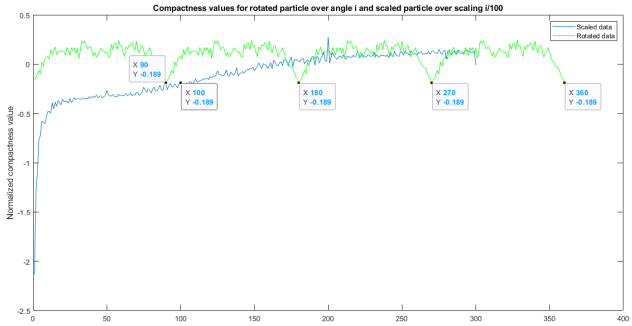
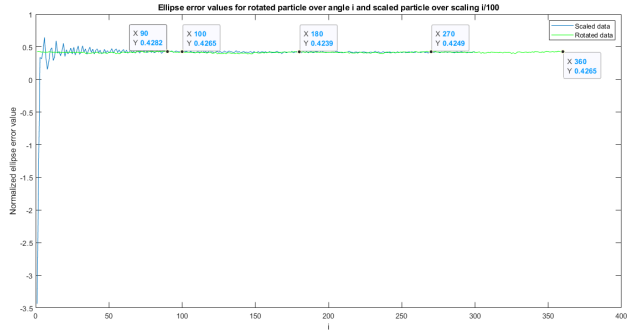


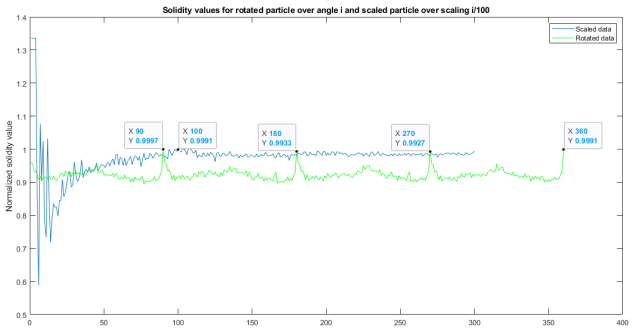
Figure 45: Sample images with different mean values of the distance between the original contour and the FFT estimate

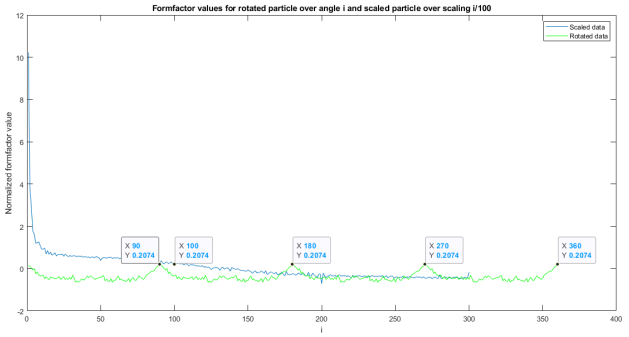
A.4.2 Feature values under rotation and scaling

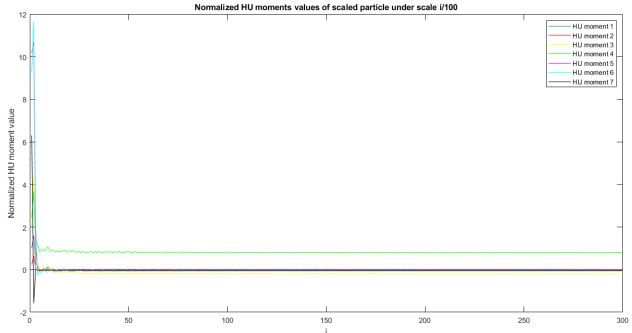


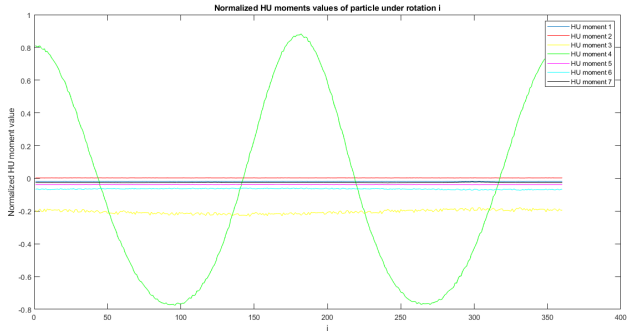


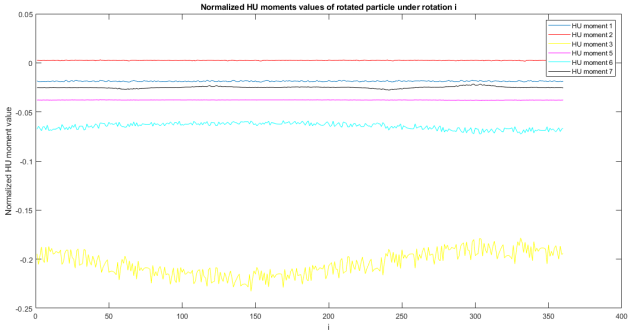


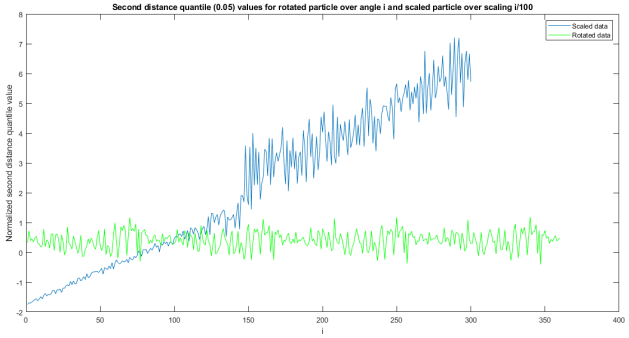


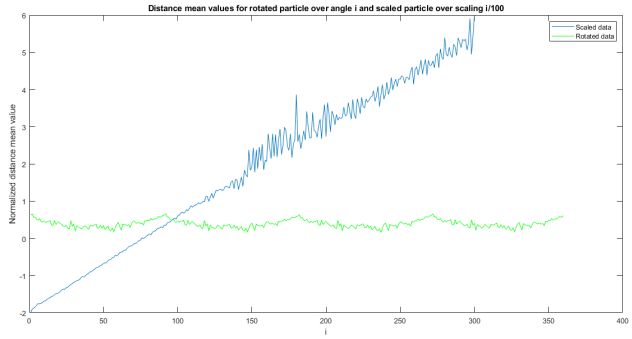






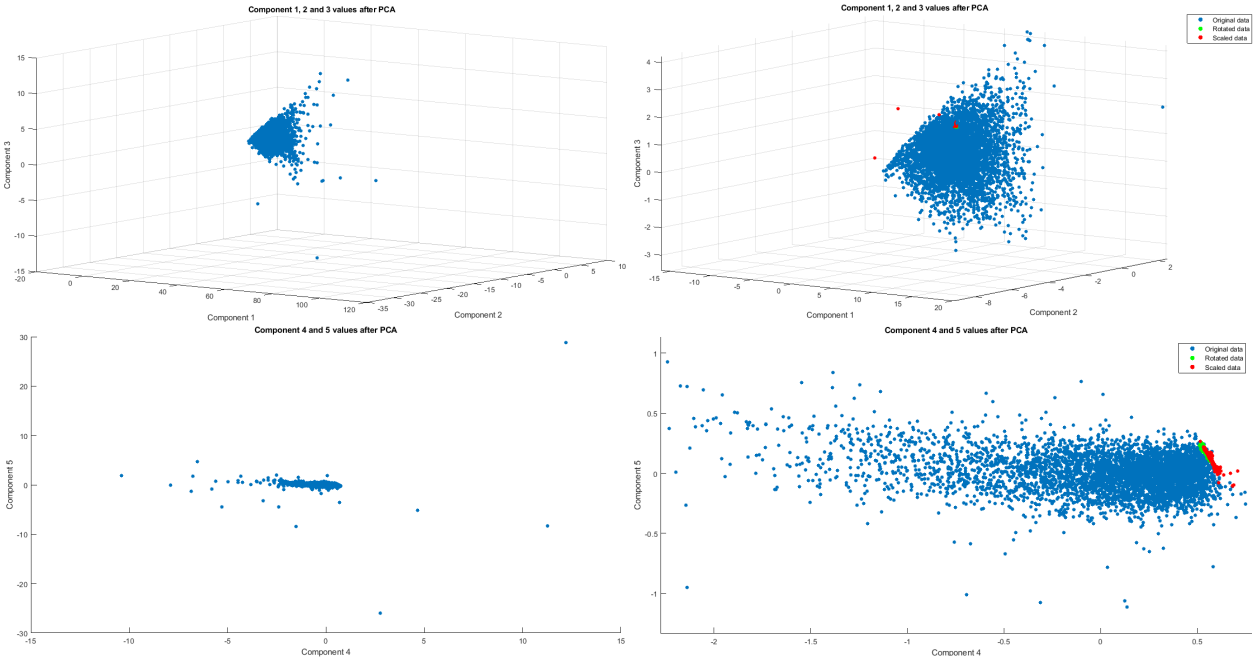






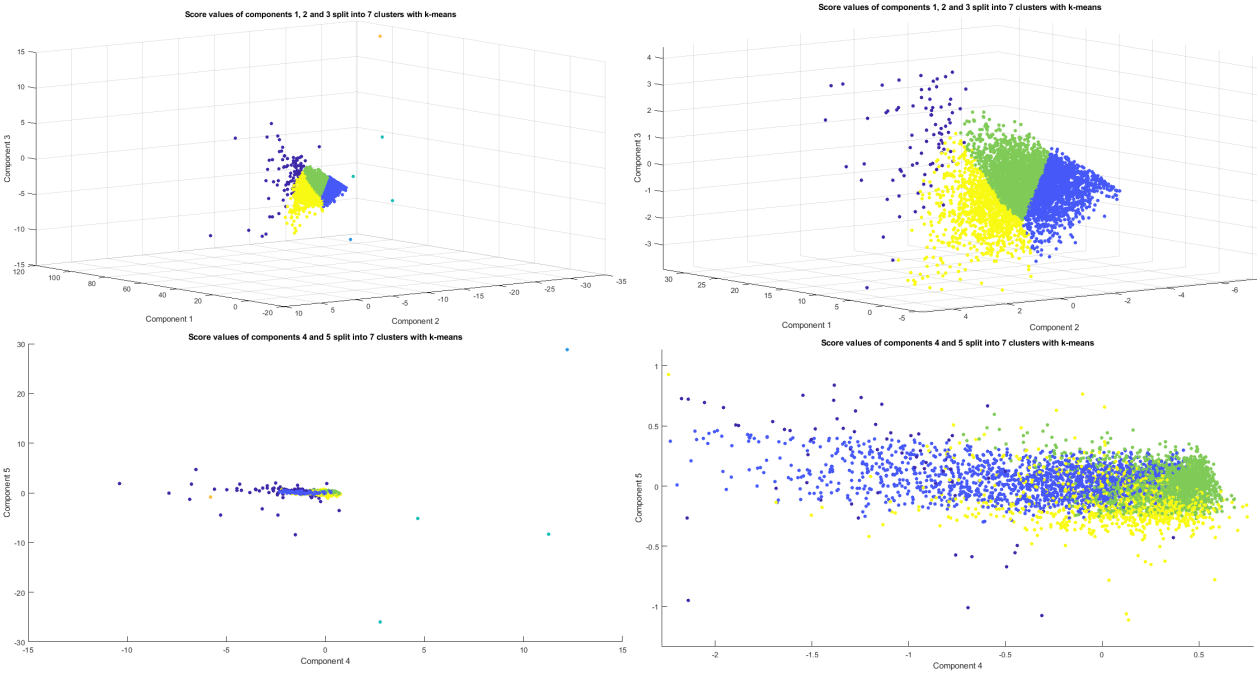
A.5 Principal component analysis

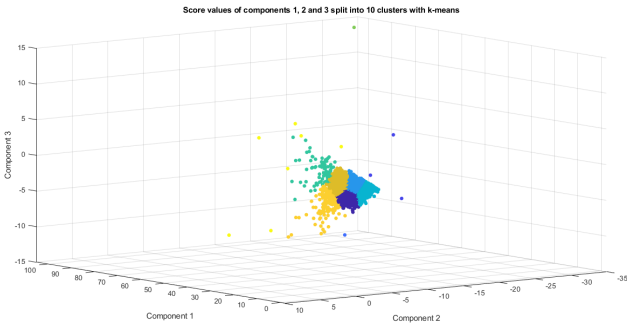
A.5.1 Score values

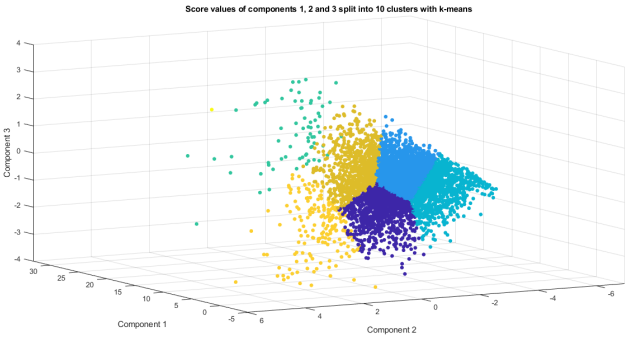


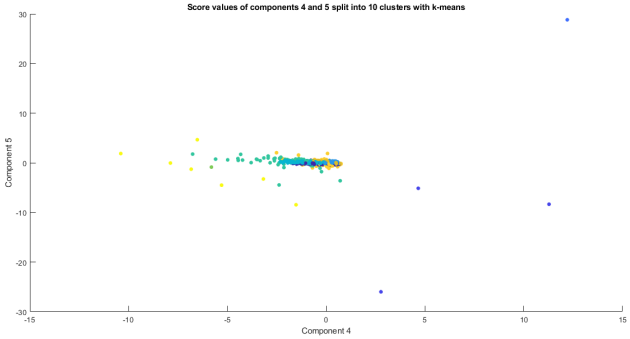
A.6 Clusters in components

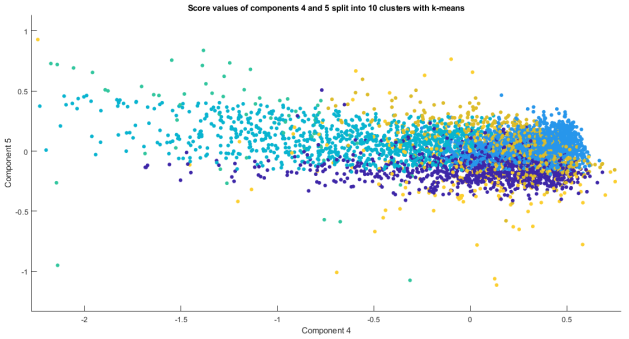
A.6.1 k-means

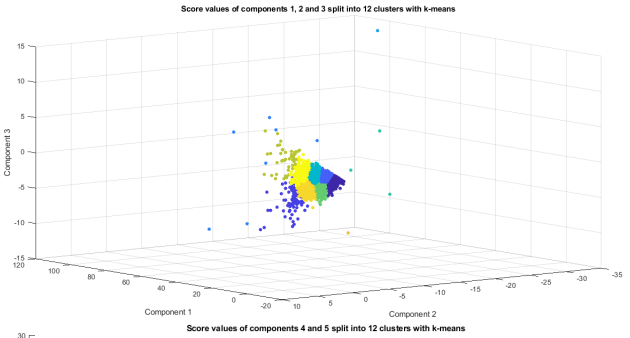


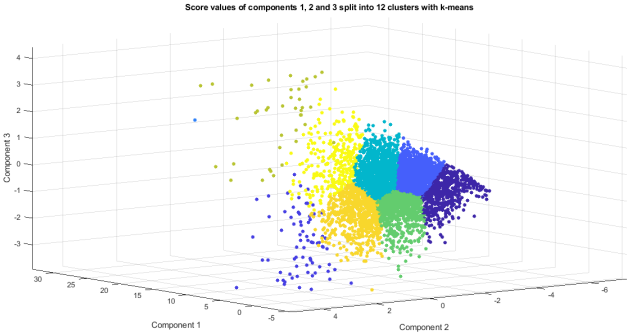

Score values of components 1, 2 and 3 split into 10 clusters with k-means
Component 1
Component 2
Component 3

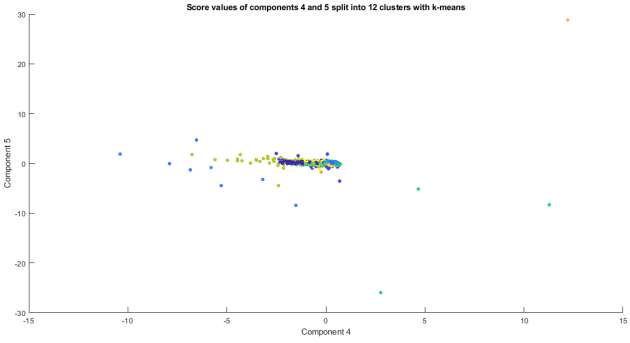

Score values of components 1, 2 and 3 split into 10 clusters with k-means
Component 1
Component 2
Component 3

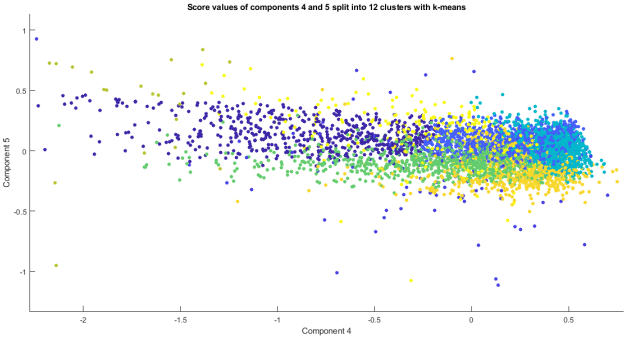

Score values of components 4 and 5 split into 10 clusters with k-means
Component 4
Component 5


Score values of components 4 and 5 split into 10 clusters with k-means
Component 4
Component 5

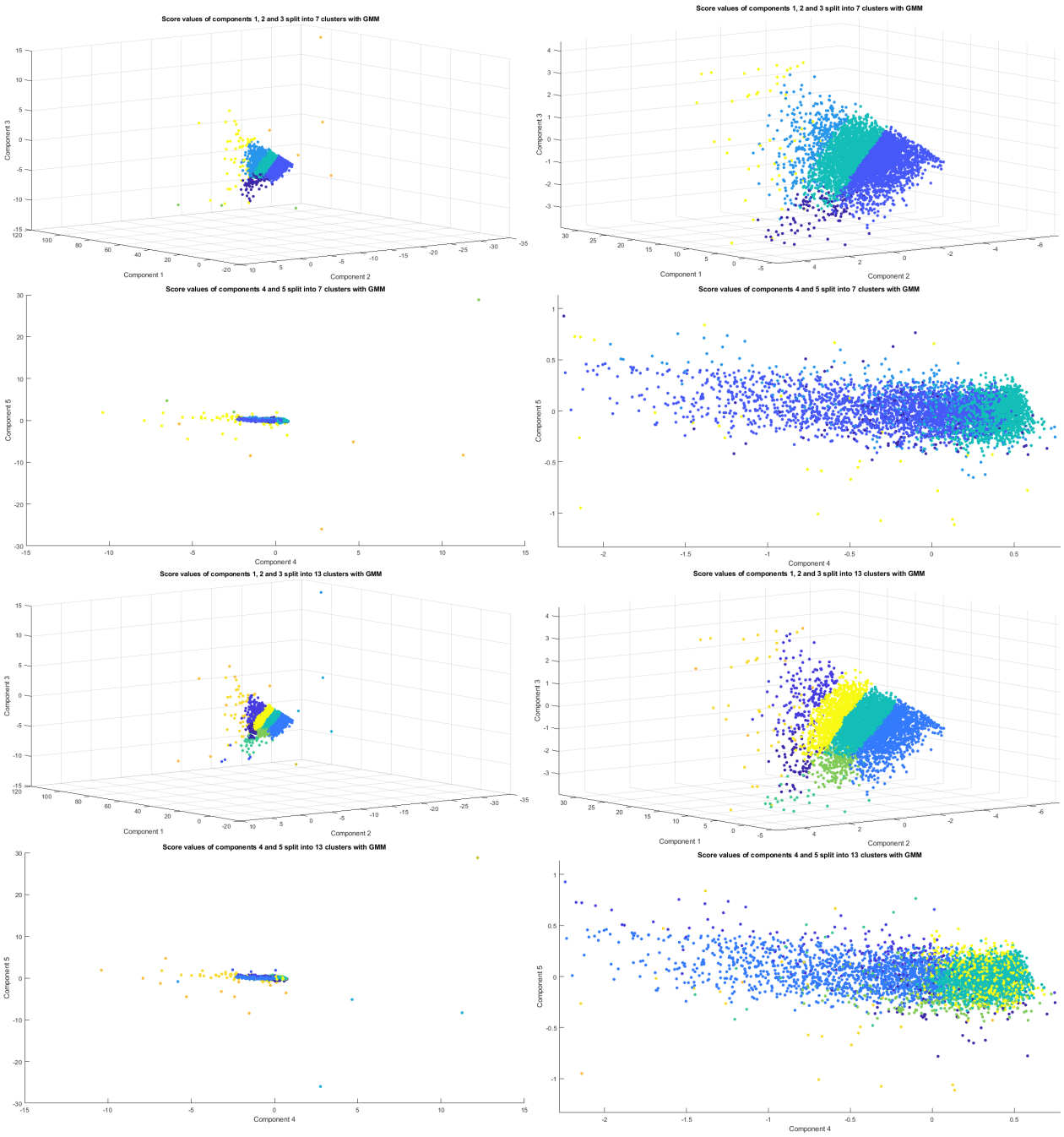

Score values of components 1, 2 and 3 split into 12 clusters with k-means
Component 1
Component 2
Component 3

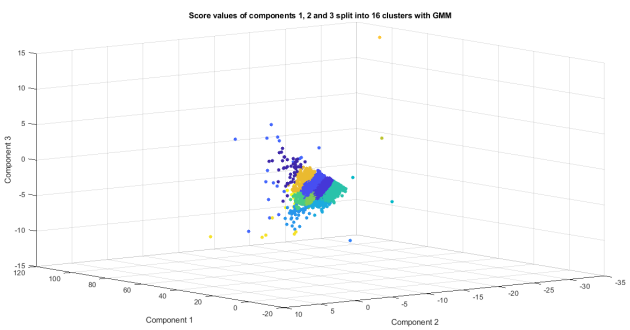

Score values of components 1, 2 and 3 split into 12 clusters with k-means
Component 1
Component 2
Component 3

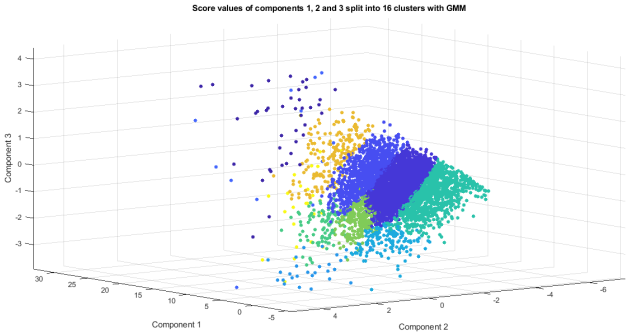

Score values of components 4 and 5 split into 12 clusters with k-means
Component 4
Component 5

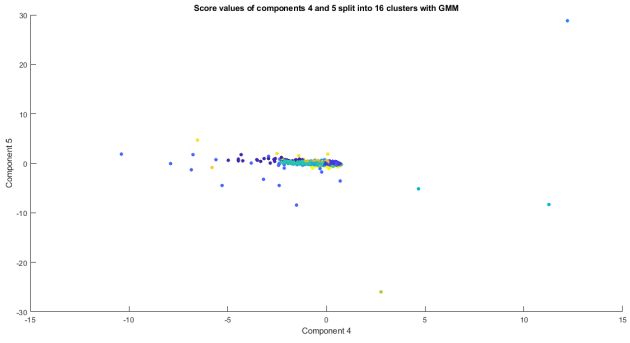

Score values of components 4 and 5 split into 12 clusters with k-means
Component 4
Component 5

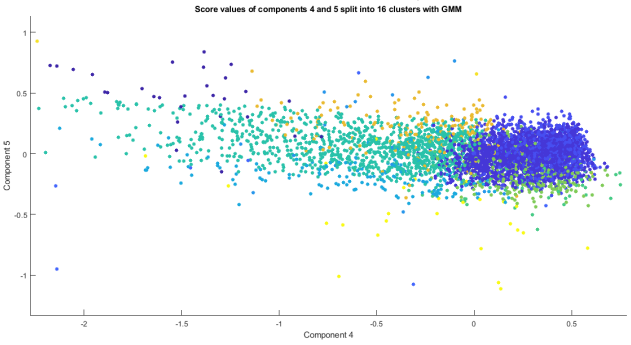
A.6.2 Gaussian mixture models




Score values of components 1, 2 and 3 split into 16 clusters with GMM
Component 3
Component 1
Component 2

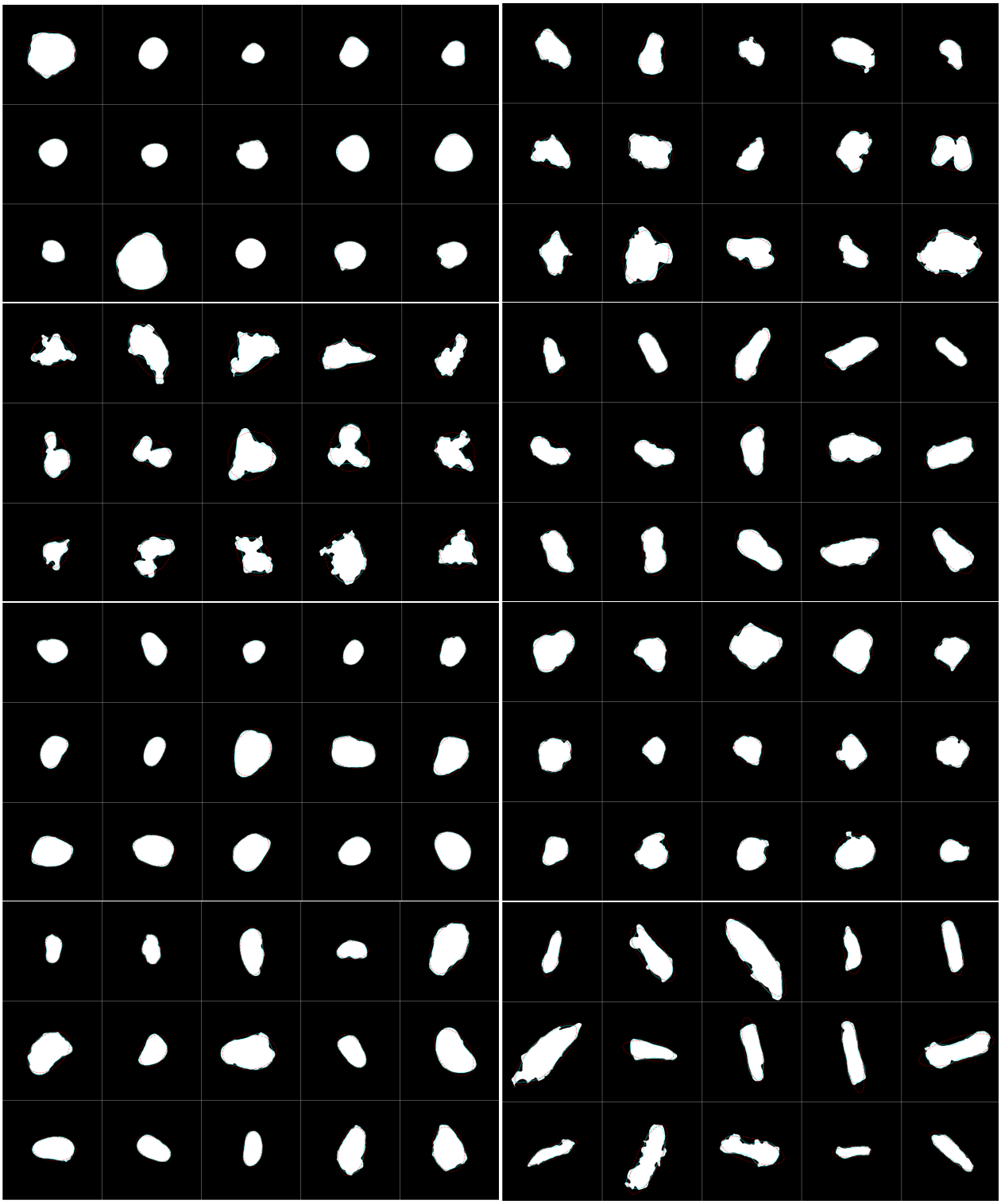

Score values of components 1, 2 and 3 split into 16 clusters with GMM
Component 3
Component 1
Component 2

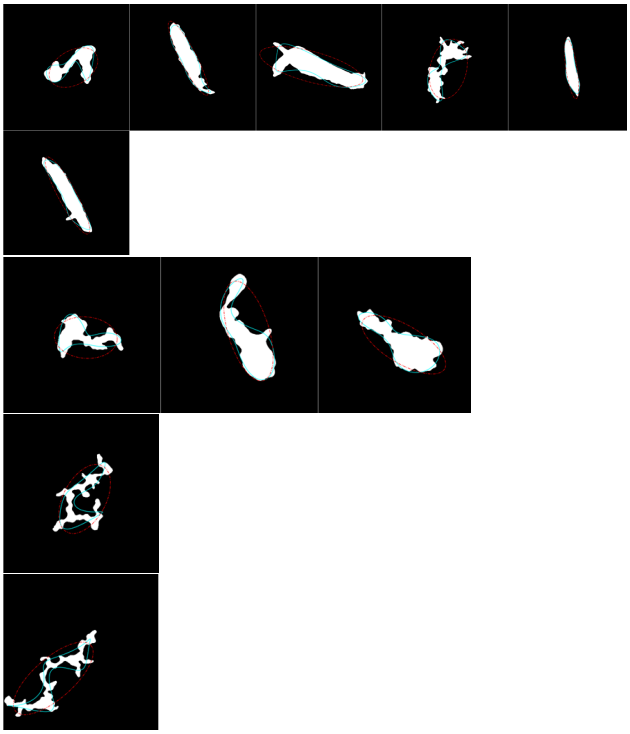

Score values of components 4 and 5 split into 16 clusters with GMM
Component 5
Component 4


Score values of components 4 and 5 split into 16 clusters with GMM
Component 5
Component 4

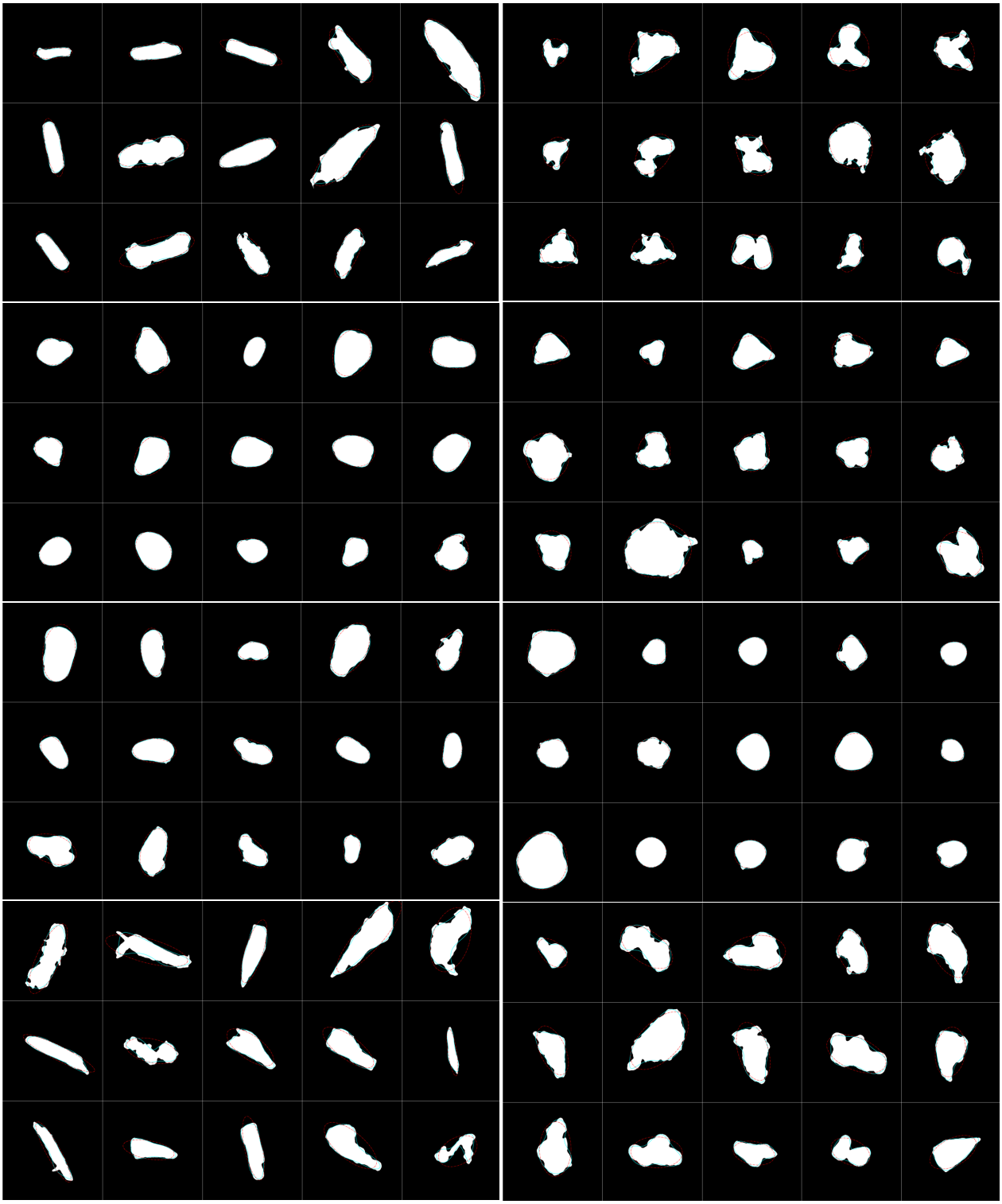
A.7 Cluster images

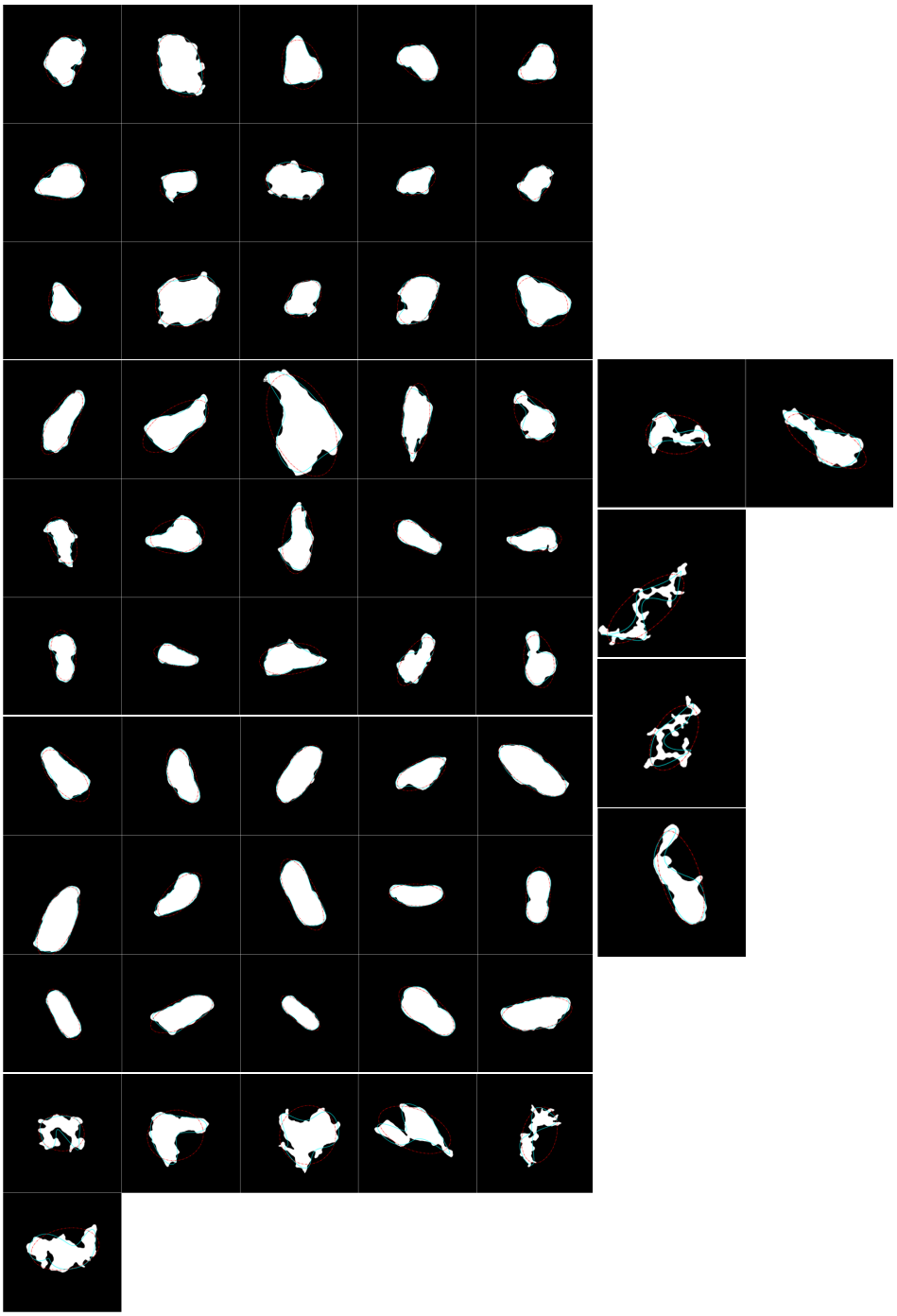
A.7.1 k-means 12 clusters





A.7.2 GMM 16 clusters





B MATLAB code

B.1 Image histogram analysis

```
function [data_table,group_1,group_2, group_3] = split_second_peak(imds, indexes, mu_upperbound,
    min_prominence)
%% Explanation
% Splits the images of the imagedatastore based on the the histogram of the
% grey values. As the background of the images is black, the first peak of
% the histogram will be the largest and heighest. For the images
% containing only noise, the first peak will be the only peak. The
% splitting into groups will be based on the prominence of the second peak
% and the mean value of the grey values.

% The algorithm is based on a few assumptions. The first peak is the
% largest, starts in point 0 and is symmetrical. The index of the second
% peak is therefore twice the index of the maximum of the histogram.

% From the index of the first peak until the end of the histogram, it
% searches for a second peak. It does this by looking for at most 2 peaks
% of at least width 8. The prominence of each peak is then calculated by
% taking the height of the peak and deducting the minimum that is obtained
% between the end of the first peak and the current peak. In case there are
% two peaks, the most prominent one will be seen as the official second
% peak.

%% Function
% Initialize the properties needed for splitting
varNames = {'mu','prominence'};
data = zeros([length(indexes) length(varNames)]);

for ii= 1:length(indexes)
    filename = cell2mat(imds.Files(indexes(ii)));
    I = imread(filename);

    % Calculate the mean of all the grey-values
    data(ii,1) = mean2(I);

    % Define the counts off each grey-value and define the maximum of the
    % histogram
    [counts,~] = imhist(I(:));
    [~, max_index] = max(counts);

    % Based on the assumption that the first peak starts in 0 and is
    % symmetric, we define the end of the peak as twice the max index
    end_peak = 2*max_index;

    % The end of the histogram is at point 256, so if we want at least the
    % possibility for 2 peaks of width 8, the end of the first peak should
    % be below 240.
    if end_peak <= 240
        % If there are 2 peaks we want them to be as far apart as possible.
        % Therefore the width of the peaks is set to half of the indexes
        tmp = floor((255-end_peak)/2);
        [max_height_2, max_index_2,~,~] = findpeaks(counts(end_peak:end-1),end_peak:255,'
            WidthReference','halfheight','NPeaks',2,'MinPeakDistance',tmp);
        nmr_peaks = length(max_height_2);
    else
        nmr_peaks = 0;
        max_height_2 = NaN;
        max_index_2 = NaN;
    end

    % When there is only 1 peak detected, the prominence is calculated like
    % described above.
    if nmr_peaks == 1
        max_prom_2 = max_height_2 - min(counts(end_peak:max_index_2));

    % When there are 2 peaks, the prominence is calculated like described above
    % and the heighest prominence is defined as the second peak.
    elseif nmr_peaks == 2
        max_prom_2 = max((max_height_2(1) - min(counts(end_peak:max_index_2(1)))), ...
            (max_height_2(2) - min(counts(end_peak:max_index_2(2)))));
    else
        max_prom_2 = NaN;
    end
    data(ii,2) = max_prom_2;

end
% Create a table with all the variables
```

```
75  data_table = array2table(data,'VariableNames',varNames);
76  group = zeros(size(indexes));
77  data_table = [table(indexes),data_table, table(group)];
78
79  % Split the group based on the min_prominence and mu_upperbound
80  data_table.group(data_table.prominence > min_prominence) = 1;
81  data_table.group(data_table.group==0 & data_table.mu < mu_upperbound) = 2;
82  data_table.group(data_table.group==0) = 3;
83
84  % Create arrays of all the indexes per group.
85  group_1 = data_table.indexes(data_table.group==1);
86  group_2 = data_table.indexes(data_table.group==2);
87  group_3 = data_table.indexes(data_table.group==3);
```

B.2 Image segmentation

B.2.1 Gradient threshold

```
function [DSize,contX,contY, imOut, mask] = measureG7subPart(filename)
img = imread(filename);
img = double(img)/max(double(img(:)));
imOut = img;

%Using the mean value per pixel seems to
% work better than the max
%% Use SEM FOV and image size to calculate area pixel

info = imfinfo(filename);
tmp = info.XMP;
f = strfind(tmp,'<xapGImg:pixelsize>');
g = strfind(tmp,'</xapGImg:pixelsize>');
str = tmp(f+19:g-1);

% [imagesize] = FindSEM_FOV(filename);
% FOVx = imagesize(1);
% FOVy = imagesize(2);
PixelX = str2double(str)/1000;
PixelY = str2double(str)/1000;
PixelArea = PixelX* PixelY; %in um2
%% Use a 2D moving average filter to surpress noise
windowSize = 7;
crop = (windowSize-1)/2;
kernel = ones(windowSize)/windowSize^2;
img = conv2(img,kernel,'valid');
%% Calculate gradient of the image and normalize
[FX,FY] = gradient(img);
tmp = sqrt(FX.^2+FY.^2);
tmp = tmp-min(tmp(:));
img = tmp/max(tmp(:));
%% Find best threshold value to detect gradient of particle, based on algorithm G4Sizer
steps = 0:.0125:.3;

biggest = NaN(length(steps),1);
idx = NaN(length(steps),1);
CC.Connectivity=4; %Initialize the  CC struct instead of growing it
CC.ImageSize=[0,0];
CC.NumObjects=0;
CC.PixelIdxList= cell(1,0);
CC(length(steps)).Connectivity=4;
for j = 1:1:length(steps)
    BW = im2bw(img,steps(j));
    BW = imfill(BW,'holes');

    CC(j) = bwconncomp(BW,4);
    numPixels = cellfun(@numel,CC(j).PixelIdxList);
    if isempty(numPixels)
        biggest(j) = NaN;
        idx(j) = 1;
    else
        [biggest(j),idx(j)] = max(numPixels);
    end

end

db = diff(biggest);
level = find(db==min(db));
stab = find(db(level:end)> nanmean(db),1);
level=stab+level;
level = steps(level);
%% After threshold has been determined perform final sizing
BW = im2bw(img,level);
BW = imfill(BW,'holes');
BW = imerode(BW,true(size(kernel))); %Compensate for smoothing the data

CC = bwconncomp(BW,4);
numPixels = cellfun(@numel,CC.PixelIdxList);
[biggest,idx] = max(numPixels);
if ~isempty(idx)
    IND = CC.PixelIdxList{idx};
    [y,x] = ind2sub(size(BW),IND);
    try
        k=boundary(x,y,1); %Calculate the tight contour of the particle
        DefectArea = biggest*PixelArea;
        DSize = sqrt(DefectArea/pi())*2;
        DSize=round(DSize*1000)/1000;
```

```
78          contX=x(k)+crop;
79          contY=y(k)+crop;
80
81          imOut = imread(filename);
82          mask=false(size(imOut));
83
84          temp = false(size(BW));
85          temp(IND) = true;
86          mask = false(size(temp,1)+2*crop);
87          mask(1+crop:end-crop,1+crop:end-crop) = temp;
88
89          mask = bwareaopen(mask, 50);
90          imOut(~mask) = 0;
91
92      catch exception
93          disp('error')
94      end
95  end
96
97  end
```

B.2.2 Activecontour

```
function [mask,edge,contX, contY, imOut] = remove_back(filename)
%% Explanation
%Remove_back is a function to determine the contour of an image. Based on
%an Class_1_Internal SEM-image of the particle, it gathers Topography
%images to look at the particle from different angles. Therefore the
%filename should be an Class_1_Internal SEM-image or otherwise the function
%gives an error.
%   First the fucntion decides whether the entire particle is visible in
%   the image. It binarizes the original internal image and sees whether
%   there are true values at the edge of the image. When this is the case,
%   not the entire particle is visible and 'edge' will return as true.
%   Secondly a rough indication of the border is made with the help of
%   a gradient, done by measureG7subPart.
%   Thirdly, the different topography images are loaded and the average of
%   each pixel is calculated to even out the noise in each direction and to
%   obtain a grey area which is close to the mask of the original particle.
%   Then the image is smoothened a bit to cancel out some noise and a
%   binary image is created. The binary is corrected with the rough fit of
%   the contour in the first step.
%   Then lastly activecontour is used to gain a smooth mask of the original
%   image using the method 'edge'. This is the final mask. Of that mask the
%   contour is deducted and imOut is adjusted so that only the particle is
%   visible.

%% Function
I = imread(filename);
imOut = I;
tmp = imbinarize(I);
tmp = bwareafilt(tmp, 1);
tmp(2:end-1,2:end-1)= false;
edge = logical(sum(tmp,'all'));

[~,~,~, ~, mask] = measureG7subPart(filename);

I_1 = imread(strrep(filename,"Internal","Topography1"));
I_2 = imread(strrep(filename,"Internal","Topography2"));
I_3 = imread(strrep(filename,"Internal","Topography3"));
I_4 = imread(strrep(filename,"Internal","Topography4"));
I_new = (I_1+I_2+I_3+I_4)/4;
I_new = wiener2(I_new,[5 5]);
bw = imbinarize(I_new);

bw(~mask)=0;
bw = imfill(bw,'holes');
mask = activecontour(I_new,bw,'edge');

[y, x] = find(mask);
k =boundary(x,y,1);
contX=x(k);
contY=y(k);
contX=[contX;contX(1)];
contY=[contY;contY(1)];
imOut(~mask)=0;

end
```

B.3 Feature Extraction

```
function [data_table,removed_indexes] = feature_extraction(imds, indexes, FFT_N, q_values,
    fLocation, fPrint)
%% Explanation
%Feature_extraction extracts all the features of the input images and
%(optional) outputs them in a folder with indications of the features.
% All the current features that the algorithm extract are named below by
% varNames.

% The features that are calculated can be deduced to 4 subject area's:
% 1. Features 'Perimeter', 'Area', 'ConvexArea', 'Compactness' and
% 'Solidity' are descriptions of the mask, where the first three are
% variant to the size of the particle and 'Compactness' and 'Solidity are
% not. To read more about these features visit
% 'http://www.cyto.purdue.edu/cdroms/micro2/content/education/wirth10.pdf'

%2. 'Ef' and 'Ec' describe the fit of the particle to an ellipse. How the
% ellipse is fitted can be read in fit_ellipse. Ef is the goodness of the
% fit and Ec is the eccentricity ie. how elongated or round the particle is

%3. 'HU' refers to HU's invariant moments. How the function works is best
% described in 'SI_Moment' and 'Hu_Moments'. More information on the
% subject can be found here:
% 'https://en.wikipedia.org/wiki/Image_moment#Moment_invariants'

%4. The last subject area is the Fast Fourier Transform. This is an
% approximation of the actual contour with a certain Amps restriction.
% 'FFT_N' gives us number of Amps that can be used, therefore you restrict
% the goodness of the fit if the number is somewhat low. Then the fit will
% be less good for the particles that have more amplitudes than the number
% given in 'FFT_N. Of this contour we calculate the actual amps and the
% distance between actual contour and the FFT-contour. The distance will
% give a numerical expression of how irregular the particle shape is. Of
% this distance the quantile given by 'q_values' and the first four moments
% are calculated.

%% Function
% Set the names of the variables and the cell 'data' that will collect all
% the features.
varNames = {'indexes'; 'Area'; 'ConvexArea'; 'Solidity'; 'Perimeter'; 'Compactness'; ...
    'Ef'; 'Ec';'HU';'NAmpsFFT'; 'DistQuantiles'; 'DistMoments'};
data = zeros([length(indexes) 47]);
emptyRows = zeros(size(indexes));

for ii=1:length(indexes)
    filename = cell2mat(imds.Files(indexes(ii)));
    try
    % Extract the contour using remove_back algorithm
    [mask,edge,contX, contY, imOut] = remove_back(filename);

    % The logical value 'edge' is 1 when the particle is not fully visable
    % and 0 when it is fully visable.
    % When edge is true, we continue and do not calculate the shape features
    if edge
        % The index will not be used for clustering.
        emptyRows(ii) = 1;
        continue
    end

    % Calculate solidity and compactness based on the area, perimeter and
    % convexarea. Even though we (probably) do not want to cluster in
    % area, perimeter and convexarea we do gather the data.
    measurement = regionprops(maskO,'Area','ConvexArea','Solidity','Perimeter');

    % When the mask consists of multiple particles, the measurement size
    % will be larger than 1. This conflicts with the assumption that only
    % one particle is visible in the image, so the particle will not be
    % observed for clustering
    if length(measurement) > 1
        % The index will not be used for clustering.
        emptyRows(ii) = 1;
        continue
    end

    measurement.Compactness = (measurement.Perimeter^2)/measurement.Area;

    % Safe the data. Remember that the features are in alphabetical order!
    data(ii,1:5) = table2array(struct2table(measurement));

    % Fit an ellipse to the current contour
```

```
79      [ellipse_t,x,y] = fit_ellipse(contX,contY);
80
81      % Create a mask of the ellipse and an overlap_mask that is true for all
82      % the pixels that are true in both the mask of the ellipse and the mask
83      % of the particle
84      mask_ellipse = poly2mask(x,y,size(imOut,2),size(imOut,1));
85      overlap_masks = mask_ellipse & mask;
86
87      m_ellipse = regionprops(mask_ellipse,'Area');
88      m_overlap = regionprops(overlap_masks,'Area');
89
90      % Calculate the error of the fit
91      data(ii,6) = m_overlap.Area/(m_ellipse.Area + measurement.Area - m_overlap.Area);
92
93      % Calculate the eccentricity of the ellipse
94      data(ii,7) = sqrt(1-((ellipse_t.short_axis^2)/(ellipse_t.long_axis^2)));
95
96      %Calculate HU's moments.
97      eta = SI_Moment(mask, mask);
98      inv_moments = Hu_Moments(eta);
99      data(ii,8:14) = inv_moments;
100
101     % Fit an FFT contour to the actual contour using the input 'FFT_N' for
102     % the amount of amplitudes
103     [FFTcontX,FFTcontY,NAmps] = FFTContour(contX,contY,FFT_N);
104     data(ii,15:22) = NAmps';
105
106     % We calculate the distance between the FFTcontour and the actual
107     % contour. Of these distance we calculate the quantiles of the input
108     % 'q_values' and the first 4 moments of the distance values
109     dist = sqrt((contX-FFTcontX).^2+(contY-FFTcontY).^2);
110     data(ii,23:43) = quantile(dist, q_values);
111     data(ii,44:47) = [mean(dist),var(dist),skewness(dist),kurtosis(dist)];
112
113     % When there exists a location variable 'fLocation', we want to print
114     % the images with their corresponding features
115     if exist('fLocation', 'var')
116         % Replace the firectory of the image from ASML database to the
117         % direction 'fLocation' that is given in the input. Insert the
118         % index of the image in the imageDatastore in the name
119         fOut = replaceBetween(filename,1,strfind(filename,'Defect')-1, fLocation);
120         fOut = insertBefore(fOut,'Defect',['Number(',num2str(indexes(ii)),')_']);
121
122         % When the variable 'fPrint' is set to mask, we want to print the
123         % mask with the features added. Any other input gives the original
124         % internal.tiff image from the imageDatastore
125         if contains(fPrint,'mask')
126             print = mask;
127             fOut = insertBefore(fOut,'Number','Mask_');
128         else
129             print = filename;
130         end
131
132         % Extract the values from the data that you have gathered.
133         plot_text = {['compactness ', num2str(data(ii,4))],['solidity ',num2str(data(ii,5))],...
134         ['ef ',num2str(data(ii,6))],['ec ',num2str(data(ii,7))],['dist-m ',num2str(data(ii,44))
                ],...
135         ['dist-v ',num2str(data(ii,45))],['dist-s ',num2str(data(ii,46))],['dist-k ',num2str(data(
                ii,47))]};
136
137         % Start_y states where the text begins on the y-axis and the
138         % step_size_y indicates how much space there is between two text
139         % lines on the y-axis. The start of the text is always 1 on the
140         % x-axis
141         start_y = 6;
142         step_size_y = 16;
143         nmr_groups = length(plot_text);
144         plot_x = ones(1,nmr_groups);
145         plot_y = start_y:step_size_y:start_y-1+step_size_y*nmr_groups;
146
147         % We will print the image (visibility is off) and collect the
148         % framework of the print. This framework is the data we actually
149         % write to the directory.
150         figure('visible', 'off')
151         imshow(print)
152         hold on
153         plot(FFTcontX,FFTcontY,'g-','LineWidth',0.75);
154         plot(x,y,'r-.','LineWidth',0.75);
155         text(plot_x,plot_y,plot_text, 'HorizontalAlignment', 'left','FontSize',9, 'Color', 'b')
156         hold off
157         F = getframe ;
```

```
158             imwrite(F.cdata,fOut)
159
160             % Make sure to close all figures! Otherwise the algorithm will slow
161             % down for a lot of indexes.
162             close all
163         end
164
165         catch
166             % When any of the previous steps fails, the index will be printed
167             % for debugging. Also the index will not be used for clustering.
168             disp(['Failed at: ' num2str(ii)])
169             emptyRows(ii) = 1;
170         end
171     end
172     % We create a data_table containing the feature names and values of the
173     % particles that were fully visible.
174     % First we check if there are any emptyRows
175     if sum(emptyRows)==0
176
177         % If no, we removed_indexes as an empty array and create a data_table
178         % of all the features and their corresponding index
179         removed_indexes = [];
180         data_table = array2table( [indexes,data] );
181     else
182         % If yes, we create a data_table of ll the features and their
183         % corresponding index, without the emptyRows. removed_indexes are the
184         % indexes that were not fully visible.
185         removed_indexes = indexes(emptyRows);
186         data_table = array2table( [indexes(~emptyRows),data(~emptyRows,:)] );
187     end
188     % Some of the features have multiple columns in the table. We merge these
189     % so that it is clear these columns are part of one and the same cluster.
190
191     % Merge HU's moments
192     data_table = mergevars(data_table,9:15);
193     % Merge NAmps of the FFT contour
194     data_table = mergevars(data_table,10:17);
195     % Merge the distance quantiles
196     data_table = mergevars(data_table,11:31);
197     % Merge the distance moments
198     data_table = mergevars(data_table,12:15);
199
200     % Now we add the variableNames.
201     data_table.Properties.VariableNames = varNames;
202     end
```

B.3.1 FFT contour

```
function [FFTcontX,FFTcontY,NAmps] = FFTContour(contX,contY,N)
% Input
% contX: x component of a closed contour contX(1) = contX(end)
% contY: y component of a closed contour contY(1) = contY(end)
% N: Number of frequency components besides the DC term (N=10 should be
% sufficient)
% Output
% FFTcontX: x component of a closed contour after FFT filtering
% FFTcontY: y component of a closed contour after FFT filtering
% NAmps: Normalized amplitudes of first and last N components in the
% frequency domain
Z = contX+1i*contY;
Z=Z(1:end-1);
FT = fft(Z);

FT(N+2:end-N) = 0;


Z2 = ifft(FT);
FFTcontX = real(Z2);
FFTcontY = imag(Z2);
FFTcontX(end+1) = FFTcontX(1);
FFTcontY(end+1) = FFTcontY(1);
NAmps = [abs(FT(2:N+1)); abs(FT(end-N+1:end))]/abs(FT(2));
```