STEPHEN KOCKENTIEDT

A SYSTEM FOR THE AUTOMATIC DETECTION AND IDENTIFICATION OF ENGINEERED NANOPARTICLES IN SCANNING ELECTRON MICROSCOPY IMAGES

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

INF FACULTY OF COMPUTER SCIENCE

A SYSTEM FOR THE AUTOMATIC DETECTION AND IDENTIFICATION OF ENGINEERED NANOPARTICLES IN SCANNING ELECTRON MICROSCOPY IMAGES

Dissertation
zur Erlangung des akademischen Grads
Doktoringenieur (Dr.-Ing.)

angenommen durch die Fakultät für Informatik
der Otto-von-Guericke Universität Magdeburg

Gutachterinnen/Gutachter
PROF. DR. KLAUS TÖNNIES
PROF. DR. XIAOYI JIANG
PROF. DR. MYRA SPILIOPOULOU

von DIPL.-INF. STEPHEN KOCKENTIEDT
geb. am 19.09.1985 in Coesfeld

Magdeburg, den 07.09.2017

Stephen Kockentiedt: *A System for the Automatic Detection and Identification of Engineered Nanoparticles in Scanning Electron Microscopy Images,* PhD Thesis, September 2017

ABSTRACT

Small particles are all around us and even nanoparticles (smaller than 100 nm in diameter) can stem from natural sources. However, intentionally created (so-called engineered) nanoparticles are used in more and more products such as deodorants and sun creams. Research is still ongoing if such particles pose a health risk. Especially people handling engineered nanoparticles at their workplace and potentially breathing them in could be at risk. Therefore, it is desirable to measure their concentration in the air in work environments.

One promising approach to do so is to gather the particles in the air and image them using a scanning electron microscope (SEM). After that, the engineered nanoparticles in the images have to be distinguished from all other particles in order to estimate their concentration. Currently, this can only be done manually, which takes much time.

This thesis introduces a fully automatic system to classify such particles in SEM images into engineered nanoparticles and other particles. As far as we know, it is the first system of its kind. It is able to save time for the user while improving the classification quality.

We compile and outline literature on automatic image-based particles analysis and propose many novel algorithms as the available methods are not suitable for our problem.

A segmentation method with a completely new noise estimation algorithm targeted at SEM is developed. In our evaluation, this method provides a good segmentation and is able to find about 10 % more engineered particles than a manual search.

We select appropriate features for the classification and devise a new one able to capture the size of local shape features. In addition, we show a completely new statistically derived method to estimate the detected electron count of each pixel of an SEM image. We derive features from these electron counts, which prove to be the most informative features we use for classification.

This thesis selects suitable classifiers, parameters, performance measures and preprocessing steps for the particle classification. In addition, we propose an automatic model selection method so that no classification expert is needed to train the system for new particle types. In our evaluation, the classification performance (measured as the geometric mean of the true positive rate and the true negative rate) of our system is on average about 18 % better than that of human experts.

The user can save about 3.6 h when analyzing a typical workplace sample using the trained system. This amounts to about 44 % of the time when considering sampling and imaging and almost all of the time after the images have been made.

Finally, this thesis proposes a method to predict the classification performance under the assumption that more particles are added to train the system. In our evaluation, the median relative error of the predicted classification performance change is about 0.39.

ZUSAMMENFASSUNG

Kleine Partikel gibt es fast überall. Sogar Nanopartikel (Partikel mit einem Durchmesser von weniger als $100\,\mathrm{nm}$) können natürlich auftreten. Allerdings werden auch immer mehr künstlich hergestellte Nanopartikel in Produkten wie Deodorants und Sonnencremes verwendet. Es wird aktuell daran geforscht, ob diese Partikel gesundheitsschädlich sind. Dieses Risiko wäre besonders hoch für Personen, die an Arbeitsplätzen mit solchen Partikeln arbeiten und sie einatmen könnten. Daher ist es wünschenswert, die Konzentration von künstlich hergestellten Nanopartikeln in der Luft an Arbeitsplätzen zu messen.

Ein erfolgversprechender Ansatz ist, die Partikel in der Luft zu sammeln und Bilder von ihnen mit einem Rasterelektronenmikroskop (REM) zu machen. Danach müssen die künstlich hergestellten Nanopartikel von allen anderen Partikeln unterschieden werden, damit man ihre Konzentration schätzen kann. Aktuell kann dieser Arbeitsschritt nur manuell durchgeführt werden, was viel Zeit in Anspruch nimmt.

In dieser Dissertation wird ein vollautomatisches System vorgestellt, um solche Partikel in REM-Bildern als künstlich hergestellte Nanopartikel oder andere Partikel zu klassifizieren. Soweit wir wissen ist es das erste System dieser Art. Es kann den Zeitaufwand für den Nutzer reduzieren, während die Klassifikationsqualität verbessert wird.

Wir sammeln Literatur zur automatischen bildbasierten Partikelanalyse und fassen sie zusammen. Außerdem haben wir viele neue Algorithmen entwickelt, da die vorhandenen nicht für unser Problem geeignet sind.

In dieser Dissertation wird eine Segmentierungsmethode mit einem komplett neuen Rauschschätzungsalgorithmus für REM-Bilder vorgestellt. In unserer Evaluation zeigt diese Methode eine gute Segmentierungsqualität und findet ungefähr $10\,\%$ mehr künstlich hergestellte Partikel als eine manuelle Suche.

Wir wählen passende Klassifikationsmerkmale aus und stellen ein neues vor, welches die Größe von lokalen Konturmerkmalen erfasst. Zusätzlich entwickeln wir eine komplett neue statistisch hergeleitete Methode, die die Anzahl der detektierten Elektronen für jeden Pixel eines REM-Bildes schätzt. Von diesen Elektronenzahlen leiten wir Merkmale her, die sich als wertvollste aller von uns verwendeten Merkmale erweisen.

In dieser Dissertation werden passende Klassifikatoren, Parameter, Klassifikationsgütemaße und Vorverarbeitungsschritte für die Partikelklassifikation ausgewählt. Zusätzlich stellen wir ein automatisches Verfahren zur Modellauswahl vor, sodass kein Klassifikationsexperte nötig ist, um das System für neue Partikeltypen zu trainieren. In unserer Evaluation ist die Klassifikationsgüte (berechnet als geometrisches Mittel aus Sensitivität und Spezifität) unseres Systems im Durchschnitt ungefähr 18 % besser als die menschlicher Experten.

Der Benutzer des trainierten System kann bei der Auswertung einer typischen Arbeitsplatzprobe etwa 3.6 h sparen. Das entspricht etwa 44 % der Zeit, wenn Probenahme und Bilderfassung berücksichtigt werden, und nahezu der gesamten Zeit, falls man diese vernachlässigt.

Zuletzt wird in dieser Dissertation eine Methode zur Vorhersage der Klassifikationsgüte beim Hinzufügen von weiteren Trainingspartikeln vorgestellt. In unserer Evaluation ist der Median des relativen Vorhersagefehlers der Veränderung der Klassifikationsgüte etwa 0.39.

DANKSAGUNG

An dieser Stelle möchte ich allen Personen danken, die mich bei meiner Forschung und der Erstellung der Doktorarbeit unterstützt haben.

Zuerst danke ich den Mitarbeitern der BAuA, die mir die Doktorandenstelle überhaupt erst ermöglicht haben und mir bei der Ausführung dieser geholfen haben. Es würde mir falsch vorkommen, einzelne Personen aus diesem tollen Team herauszugreifen. Sie standen mir mit Rat und Tat zur Seite und haben meine Meinung wertgeschätzt. Es war immer schön, euch besuchen zu kommen.

In diesem Zuge möchte ich auch Ulrich Gernert vom ZELMI nennen, mit dessen Zusammenarbeit die Bilder in dieser Dissertation entstanden sind (DFG INST 131/631-1). Er hat mir immer weitergeholfen, wenn ich Fragen bezüglich der Bildgebung hatte.

Als nächstes möchte ich dem ISG danken, wo einem immer problemlos und unkompliziert geholfen wird. Ich danke Klaus Tönnies, der mir viel Freiraum bei der Forschung gelassen und im richtigen Moment mit wertvollen Tipps geholfen hat. Auch möchte ich Charlotte danken, die mir eine angenehme Zimmernachbarin war und mir oft gute Gespräche und Ratschläge geboten hat. Zudem danke ich Basti und Clemens, die große Teile dieser Dissertation korrekturgelesen und ihre Erfahrungen mit mir geteilt haben.

Zuletzt möchte ich meiner Familie danken, die mich in der ganzen Zeit unterstützt und motiviert hat. Ich habe euch lieb!

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

LIST OF ACRONYMS

Ag	Silver. 1, 2, 4, 5, 39–45, 47, 54–56, 60, 92, 101–103, 115, 117, 118, 120, 122–125, 127, 128, 137, 142, 163, 167–177, 192, 195, 199
BAuA	German Federal Institute for Occupational Safety and Health. 3, 4, 14–16, 31–33, 37, 43–45, 49, 56, 113, 162–164, 178, 179
CCSEM	Computer-controlled scanning electron microscopy. 49, 50, 107, 200
CPG	Connected particle group. 56, 57, 62, 64, 84, 87, 89, 90, 96, 100–109, 111–118, 120–122, 124–126, 128, 129, 131, 132, 134, 137, 140–144, 146, 147, 159, 163–165, 167, 169–171, 176, 177, 179, 180, 183, 192–195, 199–201
EDX	Energy-dispersive X-ray spectroscopy. 36, 49, 50, 107, 201
RBF	Radial basis function. 145, 154–156
ROI	Region of interest. 53
SEM	Scanning electron microscope. 16, 17, 19, 20, 24, 31, 33–36, 41, 49, 64, 65, 73, 80, 93, 98, 178–180, 200
SEM	Scanning electron microscopy. 3–14, 17, 20–24, 27–29, 31, 33, 37–42, 45, 47, 49, 50, 54, 55, 59–62, 64–67, 71, 80, 81, 84, 89–93, 97, 103–105, 111–114, 128, 131, 140, 141, 163, 164, 177, 178, 183, 198
SMOTE	Synthetic minority oversampling technique. 147
SVM	Support vector machine. 134, 144–147, 149, 150, 154–157
TEM	Transmission electron microscopy. 3, 47, 49, 50, 60
TiO_2	Titanium dioxide. 1, 4, 6, 20–24, 38, 39, 44, 45, 63, 81, 91, 101, 103, 104, 112, 114, 167, 168, 170–176, 180, 199
TU Berlin	Technische Universität Berlin. 37

ZELMI	Zentraleinrichtung Elektronenmikroskopie. 36
ZnO	Zinc oxide. 1, 4, 7, 14–16, 19, 22, 24, 44, 45, 51, 84, 97, 100, 101, 103, 167, 168, 171–177, 195, 199

LIST OF SYMBOLS

a	The variable corresponding to the contrast setting of the SEM ($a \in \mathbb{R}_{>0}$). 66–76, 78, 79, 81, 90, 93, 94, 97, 98, 128
$\hat{a}$	An estimate of a ($\hat{a} \in \mathbb{R}$). 76, 78, 80–83, 93, 94, 129
$\vec{\alpha}$	A vector containing estimates of the noise parameters of an SEM image ($\vec{\alpha} = \begin{pmatrix} \hat{a} \\ -2\hat{a}\hat{b} \end{pmatrix} \in \mathbb{R}^2$). 76, 78, 205
B	The Bayes error parameter of the power-law function to predict a future classification error rate ($B \in [0, 1]$). 189, 190
b	The variable corresponding to the brightness setting of the SEM ($b \in \mathbb{R}$). 66–76, 78, 79, 81, 90, 93, 94, 97, 98, 128
$\hat{b}$	An estimate of b ($\hat{b} \in \mathbb{R}$). 76, 78, 80–83, 93, 94, 129
β	The decay rate parameter of the power-law function to predict a future classification error rate ($\beta \in \mathbb{R}_{\geqslant 0}$). 189, 190
$\mathcal{C}$	The set of all possible CPGs C ($\mathcal{C} := \{C \subseteq P \mid C \text{ is connected}\}$). 57, 106, 115–117, 120, 121, 125, 126, 128, 129
C	A connected subset of pixels representing a CPG ($C \in \mathcal{C}$). 57, 89, 105, 106, 115–117, 120, 121, 125–129, 132
c	Defined as $\frac{s-2b}{a}$ ($c \in \mathbb{N}$). 68–70, 203
$\hat{c}$	An estimate of the value of $\tilde{c}$ ($\hat{c} \in \mathbb{R}$). 81
$\tilde{c}$	A random variable representing the number of detected electrons at a certain point on the specimen over multiple images at the same position ($\tilde{c} : \Omega \to \mathbb{N}$). 65–69, 80, 203

$\mathcal{D}$	The multiset of all datasets D containing samples and their corresponding classes (A multiset $\mathcal{D} := \{D \mid D \sqsubseteq \mathcal{X} \times \mathcal{Y}\}$). 132, 144, 146, 147, 188, 190
D	A dataset containing samples and their corresponding classes (A multiset $D \sqsubseteq \mathcal{X} \times \mathcal{Y}$). 132, 160, 188–193
d	The difference of the intensities of two pixels ($d \in \mathbb{R}$). 77
$\tilde{d}$	A random variable representing the difference of the intensities of two pixels ($\tilde{d} : \Omega \to \mathbb{R}$, $\tilde{d} := \tilde{g}_1 - \tilde{g}_2$). 67–70, 72, 73
$\mathcal{D}'$	A multiset of datasets D (A multiset $\mathcal{D}' \sqsubseteq \mathcal{D}$). 160, 188
Δ	An offset between two pixels ($\Delta = (\Delta_x, \Delta_y) \in P - P := \{p - p' \mid p, p \in P\}$). 116, 126, 127
δ	The offset added to the intensity that has been found to obtain the final image threshold ($\delta \in \mathbb{R}$). 87, 88, 90, 98, 99
$\mathbb{E}(\tilde{v})$	The expected value of a random variable $\tilde{v} : \Omega \to S$ ($\mathbb{E}(\tilde{v}) \in S$). Further, $\mathbb{E}(\tilde{v} \mid S') \in S$ denotes the conditional expected value of $\tilde{v}$ given the event $S' \subseteq \Omega$. 65–73, 76, 203
E	An error rate ($E \in [0, 1]$). 189, 190
e	Euler's number ($= 2.718\,281\,828\ldots$). 65, 83, 120, 144, 145
E_{FN}	The false negative rate of a classifier. 186, 189, 190, 193, 194
$\hat{E}_{FN}$	An estimate of E_{FN}. 188–190, 192–194
E_{FP}	The false positive rate of a classifier. 186, 189, 190, 193, 194
$\hat{E}_{FP}$	An estimate of E_{FP}. 188–190, 192–194
E_{PC}	The desired average relative error of the particle concentration estimation ($E_{PC} \in \mathbb{R}_{>0}$). 18
ε	A small constant value ($\varepsilon \in \mathbb{R}_{>0}$). 125, 205
η	The filtering parameter of the non-local means algorithm ($\eta \in \mathbb{R}_{>0}$). 82, 83, 88, 90, 96, 97
F	A cross-validation fold (A multiset $F \in \mathcal{D}$). 160, 165, 188, 189
f	An image ($f : P \to G$). 55, 77, 78, 81–84, 87, 88, 106, 115, 117, 120, 125–129, 132

$\mathcal{G}$	A Gaussian kernel used in the non-local means algorithm ($\mathcal{G} : \{-r_n, \dots, r_n\} \times \{-r_n, \dots, r_n\} \to \mathbb{R}_{>0}$). 82, 83
G	The set of possible image intensities. 55, 77, 78, 81, 83, 84, 87, 88, 94, 99, 106, 115, 117, 120, 125–129
g	An intensity ($g \in G$). 84, 87, 94, 99, 125–127
g_T	A threshold used for the segmentation of images to separate the particles from the image background. 87, 88
$\tilde{g}$	A random variable representing the intensity of a pixel corresponding to a certain point on the specimen over multiple images at the same position ($\tilde{g} : \Omega \to \mathbb{R}$, $\tilde{g} := a\tilde{c} + b$). 66–68, 72
Γ	A function $\Gamma : \mathbb{N} \times \mathbb{N} \to \mathbb{R}$. 203
γ	The (normalized) cumulative angle of a contour point of a CPG ($\gamma \in \mathbb{R}$). 120, 121
h_R	The hypothesis of logistic regression with parameters $R := \left(R_0, \vec{R}\right)$ ($h_R : \mathcal{X} \to [0,1], h_R(\vec{X}) := \frac{1}{1+e^{-\left(R_0 + \vec{R}^T \vec{X}\right)}}, \forall\, R := \left(R_0, \vec{R}\right) \in \mathbb{R} \times \mathbb{R}^{n_\Phi}, \vec{X} \in \mathcal{X}$). 144
H_f	The absolute histogram of image f ($H_f : G \to \mathbb{N}, H_f(g) := \lvert\{p \in P \mid f(p) = g\}\rvert, \forall\, g \in G, f : P \to G$). 84, 87, 125
ι	A function that calculates the population similarity of a parameter population Θ ($\iota(\Theta) := \frac{\min_{(\theta,\rho)\in\Theta} \rho}{\max_{(\theta,\rho)\in\Theta} \rho}$). 151, 152
$\iota_{\max}$	The maximum population similarity ($\iota_{\max} \in]0, 1]$). 151, 152, 171
K	A kernel function ($K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$). 145, 146, 155, 156
K_{lin}	The linear kernel. 145, 147, 155, 156
$K_{\text{RBF},\theta_\gamma}$	A radial basis function kernel with parameter $\theta_\gamma \in \mathbb{R}_{>0}$. 145, 147, 155, 156, 171
κ	The expected number of particles per image ($\kappa \in \mathbb{R}_{>0}$). 18
L	Returns the true class of a sample ($L : \mathcal{X} \to \mathcal{Y}$). 185–188

l	A classifier. $l(T) : \mathcal{X} \to \mathcal{Y}$ is a trained or learned classifier trained on a training set T and $l(T)(\vec{X}) \in \mathcal{Y}$ is the class of sample $\vec{X}$ as predicted by classifier l trained on the training set T ($l : \mathcal{D} \to \mathcal{Y}^{\mathcal{X}}$). 132, 149, 155, 156, 185–188, 190–193
$l_{\text{full},\theta}$	A classifier representing the full classification pipeline. 147, 149, 165
$l_{\text{LR},\theta_\lambda,w_{\text{cls}}}$	The logistic regression classifier with the ridge parameter θ_λ and the class weight function w_{cls} ($l_{\text{LR},\theta_\lambda,w_{\text{cls}}} : \mathcal{D} \to \mathcal{Y}^{\mathcal{X}}$). 144, 147, 155, 156
$l_{\text{SVM},K,\theta_C,w_{\text{cls}}}$	An SVM classifier with the kernel K, the regularization parameter θ_C and the class weight function w_{cls} ($l_{\text{SVM},K,\theta_C,w_{\text{cls}}} : \mathcal{D} \to \mathcal{Y}^{\mathcal{X}}$). 146, 147, 155, 156
λ	The wavelength of a wave ($\lambda \in \mathbb{R}_{>0}$). 119, 120, 122
$M_{1,C,f,\Delta}$	An intermediate value for computing Haralick features. 127
$M_{2,C,f,\Delta}$	An intermediate value for computing Haralick features. 127
M_{CPG}	The minimum size of a CPG. 89, 90, 100
μ	The average number of detected electrons at a certain point on the specimen over multiple images at the same position ($\mu \in \mathbb{R}_{>0}$). 65–67, 69, 71, 93, 98, 203
$\mathbb{N}$	The set of natural numbers starting from 0 ($\mathbb{N} := \{0, 1, 2, \dots\}$). 57, 65, 67–70, 72, 73, 77, 84, 89, 125, 152, 160, 161, 186, 203
n_+	The number of positive samples in a training set ($n_+ \in \mathbb{N}$). 186–194
n_-	The number of negative samples in a training set ($n_- \in \mathbb{N}$). 186–194
$\mathbb{N}_{<i}$	The set of natural numbers up to but not including $i \in \mathbb{N}$ ($\mathbb{N}_{<i} := \{j \in \mathbb{N} \mid j < i\}$). 55
$\mathbb{N}_{>0}$	The set of positive natural numbers ($\mathbb{N}_{>0} := \{1, 2, 3, \dots\}$). 18, 55, 69, 76–78, 82, 83, 88–90, 121, 128, 129, 139, 186–191, 205
$\mathbb{N}_{\geqslant i}$	The set of natural numbers from $i \in \mathbb{N}$ ($\mathbb{N}_{\geqslant i} := \{j \in \mathbb{N} \mid j \geqslant i\}$). 151, 152, 160, 187
$\tilde{n}_+$	The number of positive samples in the random training set $\tilde{T}$ ($\tilde{n}_+ : \Omega \to \mathbb{N}$). 186–188

$\tilde{n}_{-}$	The number of negative samples in the random training set $\tilde{T}$ ($\tilde{n}_{-} : \Omega \to \mathbb{N}$). 186–188
N_4	The 4-connected neighborhood of a pixel ($N_4 : P \to \mathcal{P}(P)$). 57, 89
n_γ	The number of cumulative angles γ ($n_\gamma \in \mathbb{N}_{>0}$). 120, 121
n_C	The number of CPGs $C_0, \dots, C_{n_C - 1} \subseteq P$ ($n_C \in \mathbb{N}$). 57, 89, 105
$n_{\hat{E}}$	The number of error rate estimates to compute in order to predict a future error rate ($n_{\hat{E}} \in \mathbb{N}_{>0}$). 190–192
n_F	The number of folds F used for cross-validation ($n_F \in \mathbb{N}_{\geqslant 2}$). 139, 160, 166, 187–189
n_f	The number of analyzed images. ($n_f \in \mathbb{N}_{>0}$). 18
n_{FN}	The number of false negatives produced as a result of a cross-validation ($n_{FN} \in \mathbb{N}$). 160, 161, 168
n_{FP}	The number of false positives produced as a result of a cross-validation ($n_{FP} \in \mathbb{N}$). 160, 161, 168
n_ϕ	The number of features ($n_\phi \in \mathbb{N}_{>0}$). 106, 132, 133, 136, 144, 146, 154–156
n_{pp}	The number of pixel pairs ($n_{pp} \in \mathbb{N}_{>0}$). 69–77
$\vec{n}_{pp}$	A vector containing values of n_{pp} ($n_{pp} \in \mathbb{R}^{\lvert G+G \rvert \cdot {n_{tile}}^2}$). 77, 78
n_T	The number of training sets used to classify each sample in order to estimate the error rates of a classifier ($n_T \in \mathbb{N}_{>0}$). 188–193
n_θ	The number of parameter configurations in the population of the automatic parameter finding method ($n_\theta \in \mathbb{N}_{>0}, n_\theta \geqslant 2$). 151–153
n_{tile}	The horizontal and vertical number of tiles, the image is split into to calculate the estimates for the noise estimation ($n_{tile} \in \mathbb{N}_{>0}$). 75–78, 83, 88, 90, 93–95, 97, 128, 129
n_{TN}	The number of true negatives produced as a result of a cross-validation ($n_{TN} \in \mathbb{N}$). 160, 161, 168
n_{TP}	The number of true positives produced as a result of a cross-validation ($n_{TP} \in \mathbb{N}$). 160, 161, 168

$n_{\vec{X}}$	The number of samples $\vec{X}$. 144, 148, 149, 184–186, 189, 190
n_x	The width of an image ($n_x \in \mathbb{N}_{>0}$). 55, 117
n_y	The height of an image ($n_y \in \mathbb{N}_{>0}$). 55, 117
n_z	The number of coordinates z in a contour ($n_z \in \mathbb{N}_{>0}$). 116, 117, 121
$O_{C,f,\Delta}$	The co-occurrence matrix of the CPG C in the image f with offset Δ ($O_{C,f,\Delta} : G \times G \to [0,1]$). 126, 127
$\tilde{o}$	A random variable representing the image noise of a pixel corresponding to a certain point on the specimen over multiple images at the same position ($\tilde{o} : \Omega \to \mathbb{R}$, $\tilde{o} \coloneqq \tilde{g} - \mathbb{E}(\tilde{g})$). 66, 67, 73
Ω	The set that is the sample space as defined in probability theory underlying all random variables used in this thesis. 65–67, 69, 72, 186
ω	An evaluation metric ($\omega : \mathbb{N}^4 \to \mathbb{R}$). 152, 160
ω_F	The F-measure evaluation metric ($\omega_F : \mathbb{N}^4 \to [0,1]$). 161, 168
ω_G	The G-mean evaluation metric ($\omega_G : \mathbb{N}^4 \to [0,1]$). 161, 165, 166, 168
ω_{TN}	The true negative rate evaluation metric ($\omega_{TN} : \mathbb{N}^4 \to [0,1]$). 161, 168, 170
ω_{TP}	The true positive rate evaluation metric ($\omega_{TP} : \mathbb{N}^4 \to [0,1]$). 161, 168, 170
P	The domain of an image. It is a set containing all pixels of an image ($P \coloneqq \{0,\dots,n_x - 1\} \times \{0,\dots,n_y - 1\}$). 55, 57, 77, 78, 81, 83, 84, 87–89, 105, 106, 115, 117, 120, 125, 126, 128, 129
p	A pixel ($p = (p_x, p_y) \in P$). 55, 57, 81–84, 88, 116, 125, 126, 128, 129
P_{BG}	The background of an SEM image. It represents all pixels that do not correspond to a particle ($P_{BG} \subseteq P$, $P_{BG} \coloneqq P \setminus P_{FG}$). 87
P_{FG}	The foreground of an SEM image. It represents all pixels that correspond to a particle ($P_{FG} \subseteq P$). 87–89
$\hat{\mathbb{P}}(S)$	The estimate of $\mathbb{P}(S)$ calculated as the observed frequency of S over multiple repetitions ($\hat{\mathbb{P}}(S) \in [0,1]$). 187, 188

$\mathbb{P}(S)$	The stochastic probability of an event $S \subseteq \Omega$ ($\mathbb{P}(S) \in [0,1]$, $\mathbb{P}(S \mid S') := \frac{\mathbb{P}(S \cap S')}{\mathbb{P}(S')}, \forall S, S' \subseteq \Omega, \mathbb{P}(S') > 0$). 65, 186, 187, 203
$\mathcal{P}(S)$	The power set of a set S. It is defined as the set of all subsets of S ($\mathcal{P}(S) := \{S' \mid S' \subseteq S\}$). 57
Φ	The set of all features $\phi \in \Phi$. 106, 115, 117, 119, 124, 128
ϕ	A numerical feature ($\phi : \mathcal{P}(P) \times G^P \times \mathbb{R}_{>0} \to \mathbb{R}$). 106, 132
φ	A result of the convolution of the cumulative angle of a contour with a wavelet. 120
ϕ_{area}	The projected area of a CPG, defined as the area it occupies in an image measured in nm. 115, 117
$\phi_{\text{HC},\Delta}$	The contrast Haralick feature for offset Δ. 126, 127
$\phi_{\text{HCP},\Delta}$	The cluster prominence Haralick feature for offset Δ. 126, 127
$\phi_{\text{HCS},\Delta}$	The cluster shade Haralick feature for offset Δ. 126, 127
$\phi_{\text{HEne},\Delta}$	The energy Haralick feature for offset Δ. 126, 127
$\phi_{\text{HEnt},\Delta}$	The entropy Haralick feature for offset Δ. 126, 127
$\phi_{\text{hist},\,i}$	The value of the i-th bin of the normalized relative histogram ($\phi_{\text{hist},\,i} \in \Phi, i = 0, \dots, 9$). 124, 125
$\phi_{\text{HLH},\Delta}$	The local homogeneity Haralick feature for offset Δ. 126, 127
$\phi_{\text{HMP},\Delta}$	The maximum probability Haralick feature for offset Δ. 126, 127
ϕ_{IICP}	The in-image contour percentage of a CPG, which is the percentage of the contour which does not touch an image edge. 117–120
ϕ_{IQ}	The isoperimetric quotient, defined as the ratio of the CPG's area ϕ_{area} and the area of a circle having a perimeter equal to the CPG's perimeter including inner contours $\phi_{\text{perimeter+}}$. 117
$\phi_{\text{max EC}}$	The estimated maximum electron count of a CPG. 128, 129
$\phi_{\text{max I}}$	The maximum intensity of a CPG. 128

$\phi_{\text{MCAWR},\,\lambda}$	The mean contour angle wavelet response for the wavelength λ measured in nm. It is defines as the mean absolute response of the contour function to a convolution with a Morlet wavelet of wavelength λ. 119, 120, 122
$\phi_{\text{mean EC}}$	The estimated mean electron count of a CPG. 128, 129
$\phi_{\text{min EC}}$	The estimated minimum electron count of a CPG. 128, 129
$\phi_{\text{perimeter}}$	The perimeter of a CPG, defined as the length of its outer contour. 115, 117
$\phi_{\text{perimeter}+}$	The perimeter including inner contours of a CPG, defined as the length of its outer contour plus the lengths of its inner contours. 117
Q	The learning rate parameter of the power-law function to predict a future classification error rate ($Q \in \mathbb{R}_{\geqslant 0}$). 189, 190
q_{mut}	The mutation probability for the automatic model selection method ($q_{\text{mut}} \in [0, 1]$). 157
q_{θ_1}	The probability that for a given parameter, the parameter value is taken from θ_1 instead of from θ_2 when recombining to parameter configurations ($q_{\theta_1} \in [0, 1]$). 154, 156, 157
$\mathbb{R}$	The set of real numbers. 55, 66–69, 72, 76–78, 81, 83, 87, 88, 90, 106, 116, 120–122, 129, 144–146, 152, 160, 203
R	The internal parameters of the logistic regression classification algorithm ($R := \left(R_0, \vec{R}\right) \in \mathbb{R} \times \mathbb{R}^{n_\phi}$). 144
$\mathbb{R}_{<0}$	The set of negative real numbers defined as $\{\Lambda \in \mathbb{R} \,\vert\, \Lambda < 0\}$. 66, 81
$\mathbb{R}_{>0}$	The set of positive real numbers defined as $\{\Lambda \in \mathbb{R} \,\vert\, \Lambda > 0\}$. 18, 65, 66, 81–83, 88, 90, 106, 115, 117, 119, 120, 125, 126, 128, 129, 144–147, 205
$\mathbb{R}_{\geqslant 0}$	The set of non-negative real numbers defined as $[0, \infty)$. 126, 144, 148, 189
$\mathbb{R}^k$	The set of column vectors of size $k \in \mathbb{N}_{>0}$ with values in $\mathbb{R}$. $\vec{X}(i)$, $i \in \{1, \ldots, k\}$, is the ith entry of $\vec{X} \in \mathbb{R}^k$. 76–78, 106, 144, 149, 205

$\mathbb{R}^{k\times m}$	The set of matrices with $k \in \mathbb{N}_{>0}$ rows and $m \in \mathbb{N}_{>0}$ columns with values in $\mathbb{R}$. $\overset{\square}{A}(i,j)$, $i \in \{1,\dots,k\}$, $j \in \{1,\dots,m\}$, is the entry of $\overset{\square}{A} \in \mathbb{R}^{k\times m}$ in row i and column j. 78, 205
r_n	The neighborhood radius parameter for the non-local means algorithm ($r_n \in \mathbb{N}_{>0}$). 82, 83, 88, 90, 96
r_s	The search window radius parameter for the non-local means algorithm ($r_s \in \mathbb{N}_{>0}$). 82, 83, 88, 90, 96
$\vec{R}$	A vector of internal parameters of a classification algorithm ($\vec{R} \in \mathbb{R}^{n_\Phi}$). 144
$\vec{r}$	The vector of residuals in the iteratively reweighted least squares algorithm. 205
ρ	The estimated classification performance of a classifier-parameter combination ($\rho \in \mathbb{R}$). 151–153, 160
S	An undefined set. 11
s	The sum of the intensities of two pixels ($s \in a\mathbb{N} + 2b := \{ai + 2b \in \mathbb{R} \mid i \in \mathbb{N}\}$). 68–77
s_p	The pixel size $s_p \in \mathbb{R}_{>0}$, which is the distance of the center points of two horizontally or vertically neighboring pixels, in nm. 106, 115, 117, 120, 125–129, 132
$\overset{\square}{s}$	A matrix containing values of s in its first column and 1s in its second column ($\overset{\square}{s} \in \mathbb{R}^{(\lvert G+G\rvert \cdot {n_{\text{tile}}}^2)\times 2}$). 76, 78, 205
$\tilde{s}$	A random variable representing the sum of the intensities of two pixels ($\tilde{s} : \Omega \to \mathbb{R}$, $\tilde{s} := \tilde{g}_1 + \tilde{g}_2$). 67–73, 76
$\vec{s}$	A vector containing values of s ($\vec{s} \in \mathbb{R}^{\lvert G+G\rvert \cdot {n_{\text{tile}}}^2}$). 77, 78
$\hat{\sigma}_{\tilde{v}}$	The estimated standard deviation of $\tilde{v}$ ($\hat{\sigma}_{\tilde{v}} \in \mathbb{R}_{>0}$). 78
σ_X	The standard deviation of the values of a feature. 149
$\hat{\sigma}$	The estimated standard deviation of the data in the iteratively reweighted least squares algorithm. 205
T	A training set containing samples and their corresponding classes (A multiset $T \sqsubseteq \mathcal{X} \times \mathcal{Y}$). 132, 144, 147–149, 152, 160, 165, 166, 185, 188

t	The time a process may take to run. 152, 165, 166
$\tilde{T}$	A random training set ($\tilde{T} : \Omega \to \mathcal{X} \times \mathcal{Y}$). 186–188
τ	A number between 0 and 1 ($\tau \in [0, 1]$). 148, 153–157, 190
τ_n	The maximum percentage of samples to use for the error rate estimation in order to predict a future error rate ($\tau_n \in (0, 1]$). 190–193
Θ	A parameter population used in the automatic parameter selection method. It is a set of tuples (θ, ρ) of a parameter θ and its estimated classification performance ρ. 151–153
θ	A parameter of a classification algorithm. 147, 149, 151–157, 160, 165, 166, 188
θ_C	The regularization parameter of an SVM ($\theta_C \in \mathbb{R}_{>0}$). 145–147, 155, 156, 171
θ_γ	The parameter of a radial basis function kernel K_{RBF} ($\theta_\gamma \in \mathbb{R}_{>0}$). 145, 147, 155, 156, 171
θ_K	A parameter describing the kernel to be used and its parameters. 156
θ_l	A parameter describing the classifier to be used and its parameters. 147, 149, 155, 156
θ_λ	The ridge parameter of the logistic regression classifier used for regularization ($\theta_\lambda \in \mathbb{R}_{\geqslant 0}$). 144, 145, 147, 155, 156, 171
θ_{MS}	The maximum spread parameter of our resampling method. 147–149, 155, 156, 171
θ_N	The oversampling percentage of SMOTE in percent. 147–149, 155, 156, 171
θ_ϕ	The indices of the features selected by our feature selection method ($\theta_\phi \subseteq \{1, \dots, n_\phi\}$). 146, 147, 149, 154–156
θ_w	The weight given to a samples of the positive class for the training of a classifier.. 147, 149, 155, 156, 171
u	The average difference of the intensities of multiple pixel pairs ($u \in \mathbb{R}$). 75, 77
$\tilde{u}$	A random variable representing the average difference of the intensities of multiple pixel pairs ($\tilde{u} : \Omega \to \mathbb{R}$, $\tilde{u} := \frac{1}{n_{pp}} \sum_{i=1}^{n_{pp}} \tilde{d}_i$). 72, 73, 76
$\vec{u}$	A vector containing values of u ($\vec{u} \in \mathbb{R}^{\lvert G+G \rvert \cdot {n_{\mathrm{tile}}}^2}$). 77, 78

ν	The average squared difference of the intensities of multiple pixel pairs ($\nu \in \mathbb{R}$). 75, 77
$\tilde{\nu}$	A random variable representing the average squared difference of the intensities of multiple pixel pairs ($\tilde{\nu} : \Omega \to \mathbb{R}$, $\tilde{\nu} := \frac{1}{n_{pp}} \sum_{i=1}^{n_{pp}} \tilde{d}_i^2$). 69–71, 73, 76, 78
$\vec{\nu}$	A vector of values of ν ($\vec{\nu} \in \mathbb{R}^{\|G+G\| \cdot {n_{tile}}^2}$). 77, 78, 205
$\mathrm{Var}(\tilde{o})$	The variance of a random variable $\tilde{o} : \Omega \to S$ ($\mathrm{Var}(\tilde{o}) := \mathbb{E}\Big((\tilde{o} - \mathbb{E}(\tilde{o}))^2\Big)$, $\mathrm{Var}(\tilde{o} \,\|\, S') := \mathbb{E}\Big((\tilde{o} - \mathbb{E}(\tilde{o} \,\|\, S'))^2 \,\Big\|\, S'\Big)$, $\forall S' \subseteq \Omega$). 65–67, 69, 70, 73
w_{cls}	A function defining the weight of a given class ($w_{\mathrm{cls}} : \mathcal{Y} \to \mathbb{R}_{>0}$). 144, 146, 147, 149
$\overset{\square}{W}$	A diagonal matrix containing weights in the iteratively reweighted least squares algorithm. 205
w_{px}	A function defining the weight of a given pixel in the non-local means algorithm ($w_{\mathrm{px}} : P \to \mathbb{R}_{>0}$). 83
$\mathcal{X}$	The feature space ($\mathcal{X} := \mathbb{R}^{n_\phi}$). 106, 132, 144–149, 152, 160, 185–188, 190
x	The horizontal position of a coordinate in an image ($x \in \mathbb{R}$). 116, 117, 121, 122
$\bar{X}$	The mean value of a feature. 149
$\tilde{X}$	A random variable representing the distribution of randomly gathering a CPG in a realistic scenario and calculating its feature values ($\vec{X} : \Omega \to \mathcal{X}$). 186, 187
$\vec{X}$	A sample or, more precisely, the feature vector of a CPG ($\vec{X} \in \mathcal{X}$). 132, 144, 145, 147–149, 160, 165, 166, 185, 187, 188
ξ	The threshold that determines the maximum intensity of an image to be considered empty ($\xi \in \mathbb{R}$). 87, 88, 90, 100
$\mathcal{Y}$	The set of classes a sample can be assigned to by the classifier ($\mathcal{Y} := \{Y_+, Y_-\}$). 132, 136, 144, 146–149, 152, 160, 185–188, 190
Y	A class a sample can be assigned to by the classifier ($Y \in \mathcal{Y}$). 132, 136, 144, 147–149, 160, 185–188
y	The vertical position of a coordinate in an image ($y \in \mathbb{R}$). 116, 117, 121, 122

Y_+	The class of all CPGs that contain engineered nanoparticles ($Y_+ \in \mathcal{Y}$). 132, 144, 145, 147–149, 160, 186, 188, 190
Y_-	The class of all CPGs that do not contain engineered nanoparticles ($Y_- \in \mathcal{Y}$). 132, 144, 147, 148, 160, 186, 188, 190
z	A coordinate in an image ($z = (x, y) \in \mathbb{R}^2$). 116, 117, 121
ζ	The minimum probability of a configuration to be chosen as a parent in the automatic parameter selection method, measured relative to the probability of the configuration with the best classification performance ($\zeta \in [0, 1]$). 153, 154

1 INTRODUCTION

Small particles are a part of daily life. Sugar, salt, flour or cocoa powder are examples that are used in many kitchens around the world. The small particle sizes lead to a large surface and, thus, to a high reactivity, which is needed for the chemical processes involved when preparing food. House dust is another example of a mixture of small particles commonplace in every home.

However, there are even smaller particles than that. These are called nanoparticles or ultrafine particles and have diameters between $1\,\mathrm{nm}$ and $100\,\mathrm{nm}$. As a comparison, the average size of wheat flour particles is in the order of $100\,000\,\mathrm{nm}$. Nanoparticles can be natural or produced by humans. Examples of natural sources are volcanic eruptions and forest fires [Oberdörster et al., 2007]. Human-caused nanoparticles can be further categorized into incidental particles and engineered nanoparticles. The first category contains all byproducts of production processes or any other human action. Major examples are automobile exhausts and factory fumes.

Engineered nanoparticles are intentionally produced particles at a scale of a few nm. These have considerably gained importance in recent years as their large surface area and high reactivity are advantageous in many applications. To name a few examples, silver (Ag) nanoparticles are used in deodorants, toothpaste and fabrics for their antimicrobial activity [Sung et al., 2009]. Zinc oxide (ZnO) particles are used in sun creams to reflect the sunlight [Crosera et al., 2009]. Titanium dioxide (TiO_2) nanoparticles are used as a catalyst in self-cleaning surfaces [Wagner et al., 2009].

In 2004, an estimated $1000\,\mathrm{t}$ to $2000\,\mathrm{t}$ of nanoparticles such as TiO_2 and ZnO were used in skincare products alone [The Royal Society & The Royal Academy of Engineering, 2004, p. 26]. The amount of nanomaterials used in structural applications is estimated to go up from approximately $10\,\mathrm{t}$ in 2004 to between $10\,000\,\mathrm{t}$ and $100\,000\,\mathrm{t}$ in 2020 [The Royal Society & The Royal Academy of Engineering, 2004, p. 27]. The total production of TiO_2 nanoparticles is expected to grow from about $50\,000\,\mathrm{t}$ in 2010 to about $200\,000\,\mathrm{t}$ in 2015 [Future Markets Inc., 2011]. It must be noted, however, that these numbers are only rough estimates as the production amounts are usually kept confidential by the producing companies.

1.1 MOTIVATION

The potential toxicity of engineered nanoparticles has not yet been sufficiently explored [Gonzalez et al., 2008; Ostrowski et al., 2009; Johnston et al., 2010]. In general, because of their high surface reactivity and the ability to cross cell membranes, nanoparticles are suspected to have adverse health effects [The Royal Society & The Royal Academy of Engineering, 2004, p. ix]. In studies conducted with rats, Ag nanoparticles are detectable in the blood after inhalation [Johnston et al., 2010]. Low concentrations can also be measured in several organs including the brain. Cha et al. [2008] observe liver inflammation in mice after ingestion of microparticulate and nanoparticulate Ag. Bhabra et al. [2009] show that cobalt-chromium nanoparticles can damage human cells and DNA if they are directly exposed to these particles. Current research focuses on acute toxicity instead of chronic effects [Ostrowski et al., 2009].

The Royal Society & The Royal Academy of Engineering [2004, p. ix] expects a low likelihood of nanoparticles fixed or embedded in products being released from them. "Nevertheless (depending on the way in which they are manufactured, stored, transported or incorporated into products), there is the potential for some nanopowders to be inhaled in certain workplaces in significant amounts." [The Royal Society & The Royal Academy of Engineering, 2004, p. 80] Among these workplaces are factories where the nanoparticles are produced. That raises the need to measure the exposure to specific nanoparticles of humans such as workers. This has two reasons. Firstly, to further research the potential toxicity, it is necessary to measure the amount of particles a person is exposed to in order to discover correlations with health issues. Secondly, as long as the toxicity of a particular particle type has not be ruled out, there need to be strict exposure limits. These limits have to be enforced by directly measuring the exposure.

While the exact circumstances of the toxicity of nanoparticles are not yet known, engineered nanoparticles may be more toxic than particles of the same size from other sources due to their chemical composition [Savolainen et al., 2010]. Therefore, a measurement system has to differentiate the engineered nanoparticles from so-called background particles. These are other particles which also occur in ambient air. An example of a background particle type is diesel soot. It is produced by diesel engines, which are used in cars, ships and trains.

There are several automatic techniques to measure the concentration of nanoparticles in the air: condensation particle counter, optical particle counter, fast mobility particle sizer, scanning mobility particle sizer, electric low pressure impactor, aerosol diffusion charger and tapered element oscillating microbalance [Methner et al., 2009]. However, these techniques have several disadvantages. Some are not able to measure the size of the particles. Some of the instruments only work in specific size ranges. Many of them are also not portable, are complex to use or bear a high cost [Methner et al., 2009]. All of these instruments have in common that they are not able to differentiate engineered nanoparticles from background particles [Savolainen et al., 2010; National Institute for Occupational Safety and Health (NIOSH), 2009, p. 27]. In addition, these instruments cannot detect groups of nanoparticles that stick together to form so-called agglomerates. Rather, they are regarded as particles larger than 100 nm and, thus, not as nanoparticles.

Therefore, the National Institute for Occupational Safety and Health (NIOSH) [2009] suggests the use of scanning electron microscopy (SEM) or transmission electron microscopy (TEM) to analyze particles gathered from the air in environments where engineered nanoparticles may occur to supplement other measurement techniques [Methner et al., 2009]. This method has the advantage that it is possible to count the engineered nanoparticles separately and that the device used to gather the particles can be very small and can be carried by a worker. To measure his or her exposure, the gathered particles are later analyzed by taking images of them using SEM or TEM.

1.2 PROBLEM FORMULATION

The German Federal Institute for Occupational Safety and Health (BAuA) seeks to develop guidelines on how to measure the concentration of engineered nanoparticles at workplaces where they are produced or handled. The current version of the recommended process is described in the following paragraphs.

As a first step, using either a thermal or electrostatic precipitator, particles are gathered from the air at the workplace and deposited onto a silicon wafer (see Section 4.1). The electrostatic precipitator has a high flow rate and, thus, can gather a large amount of particles in a short time. However, it is also relatively big and heavy. The thermal precipitator, on the other hand, can be worn by a worker to measure his or her exposure directly instead of measuring the particle concentration at a stationary position. The downside is that it is only able to gather one thousandth of the particles in the same time compared to the electrostatic precipitator. In addition to the precipitators, particle counters are used to directly measure number and size distribution of particles in the air. However, these are not able to determine the composition of these particles.

In a next step, SEM images of the silicon wafer onto which the particles have been deposited by the precipitators are taken. By using electrons, SEM can achieve higher magnifications than light microscopy. This is necessary because of the small size of the nanoparticles.

Currently, the BAuA concentrates on Ag, TiO_2 and ZnO nanoparticles, because they are contained in the list of representative manufactured nanomaterials for safety testing by the Organisation for Economic Co-operation and Development (OECD) [2010] and are commonly used in commercial products.

In the next paragraphs, we will describe three pairs of example SEM images. Both images of each pair show the same location of the wafer. However, the first image has a magnification of 5000 and the second of 20 000. The particles in these images have been collected using an electrostatic precipitator as described above.

Figures 1.1 and 1.2 show Ag nanoparticles with average diameters in the range of 50 nm to 100 nm. The images illustrate very well that the nanoparticles usually do not occur individually. Instead, multiple particles stick together to form a so-called agglomerate. Figure 1.2 shows two agglomerates. The approximately spherical structures inside them are the actual nanoparticles. Figure 1.1 illustrates that the sizes of such agglomerates can vary to a high degree.

Figures 1.3 and 1.4 show TiO_2 nanoparticles with an average diameter of 25 nm. Similar to the Ag particles, they form agglomerates. However, due to the different average particle size, the appearance is very different. Even in Figure 1.4, with a magnification of 20 000, the single nanoparticles are only barely perceptible.

Figures 1.5 and 1.6 show ZnO nanoparticles with an average diameter of 10 nm. These illustrate that the nanoparticle size is a significant factor for the appearance of the agglomerates. In addition, the single nanoparticles are not discernible anymore, even at a magnification of 20 000.

While gathering the particles from the air at a workplace, it is expected that, in addition to engineered nanoparticles, other particles from the ambient air are also deposited onto the silicon wafer. In Figures 1.7 to 1.9, a selection of these background particles can be seen. Figure 1.7 shows diesel soot particles which are produced and exhausted by any kind of diesel engine. Just as the engineered nanoparticles, these have a tendency to form agglomerates, which look very similar to the ones formed by TiO_2 or ZnO. Figure 1.8 shows particles which were gathered during abrasive cutting of steel. The nearly spherical particle in the upper left corner is probably composed of steel. The origin of the other particles is unknown. In Figure 1.9, one can see a sample gathered from the ambient air in an urban environment.

For the sake of brevity, we will from now on refer to engineered nanoparticles, nanoparticle agglomerates and background particles simply as *particles*. If we want to refer to single nanoparticles, either as part of an agglomerate or not, we will call them *primary particles*. In order to further clarify these terms, a short list of definitions follows:

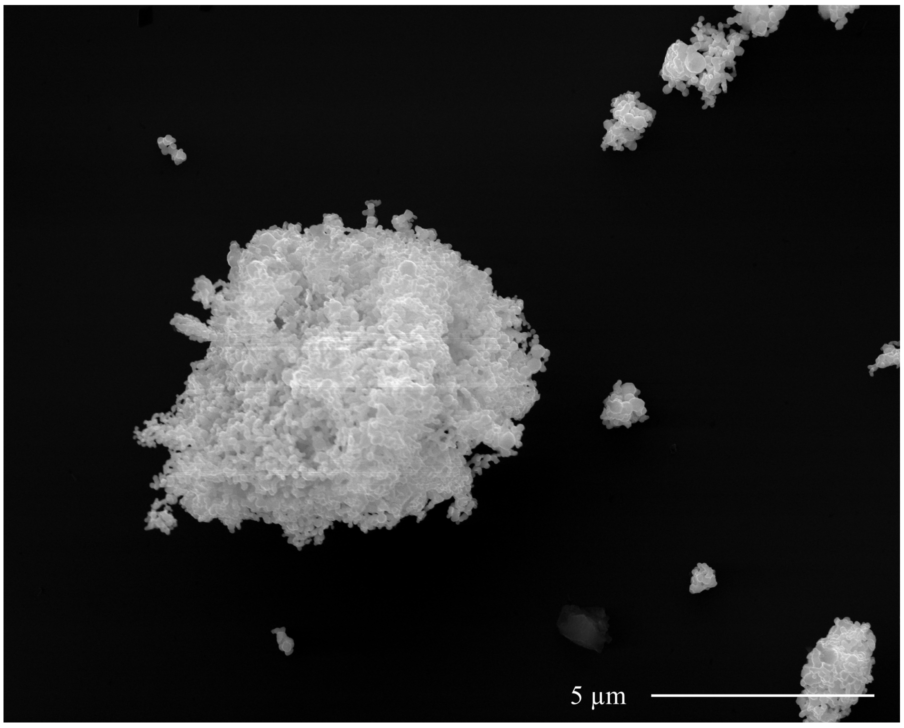


Figure 1.1: An SEM image (1 $\mathrm{pixel} = 5.08\,\mathrm{nm}$) of Ag nanoparticles with an average diameter of $75\,\mathrm{nm}$.

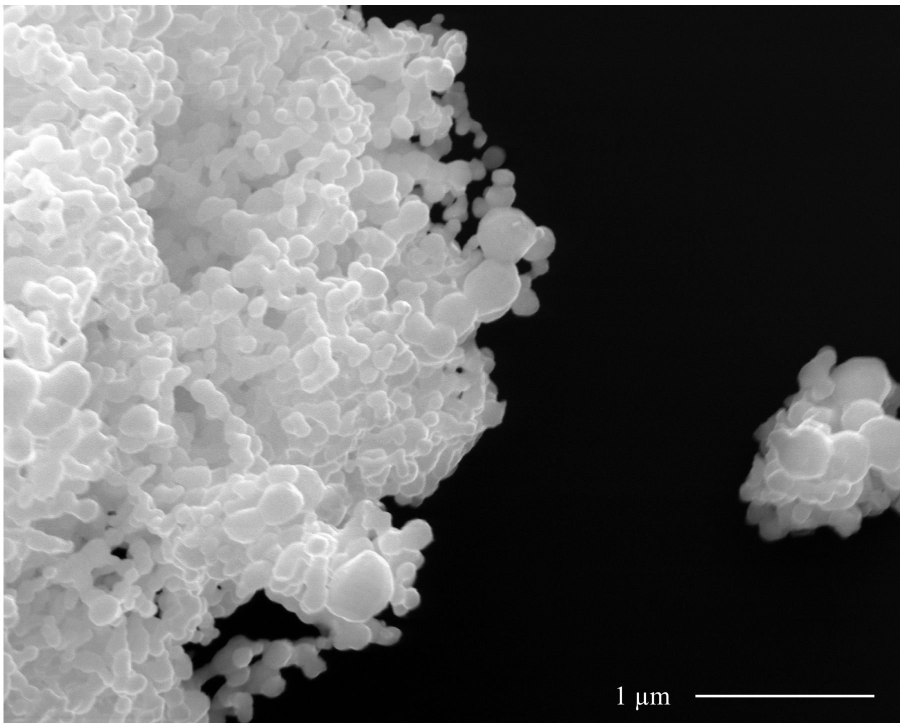


Figure 1.2: An SEM image (1 $\mathrm{pixel} = 1.27\,\mathrm{nm}$) of Ag nanoparticles with an average diameter of $75\,\mathrm{nm}$. It shows the two agglomerates at the center of Figure 1.1.

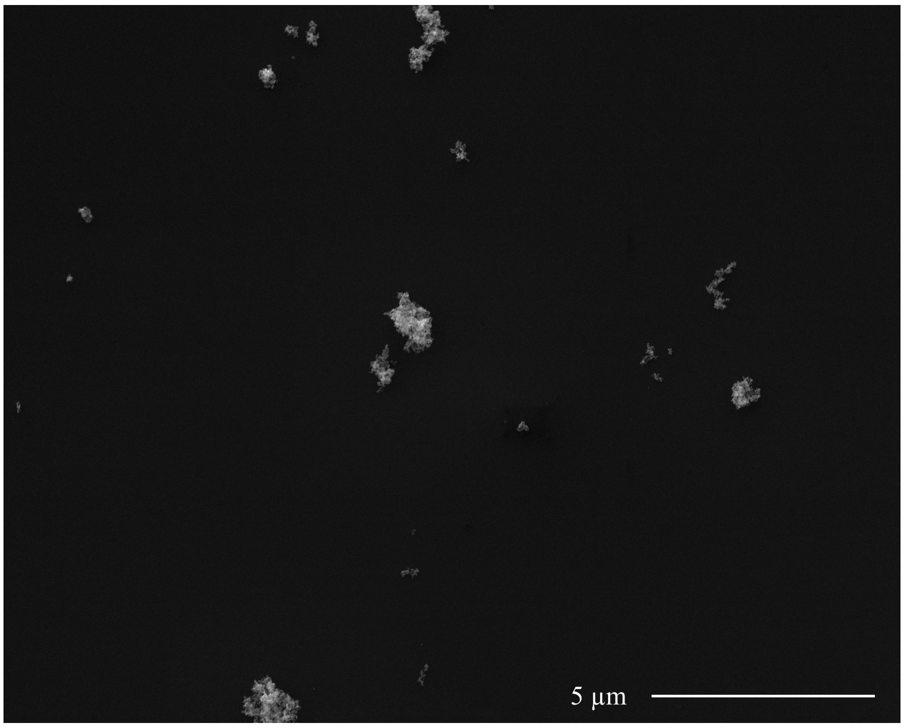


Figure 1.3: An SEM image (1 pixel = 5.08 nm) of TiO_2 nanoparticles with an average diameter of 25 nm.

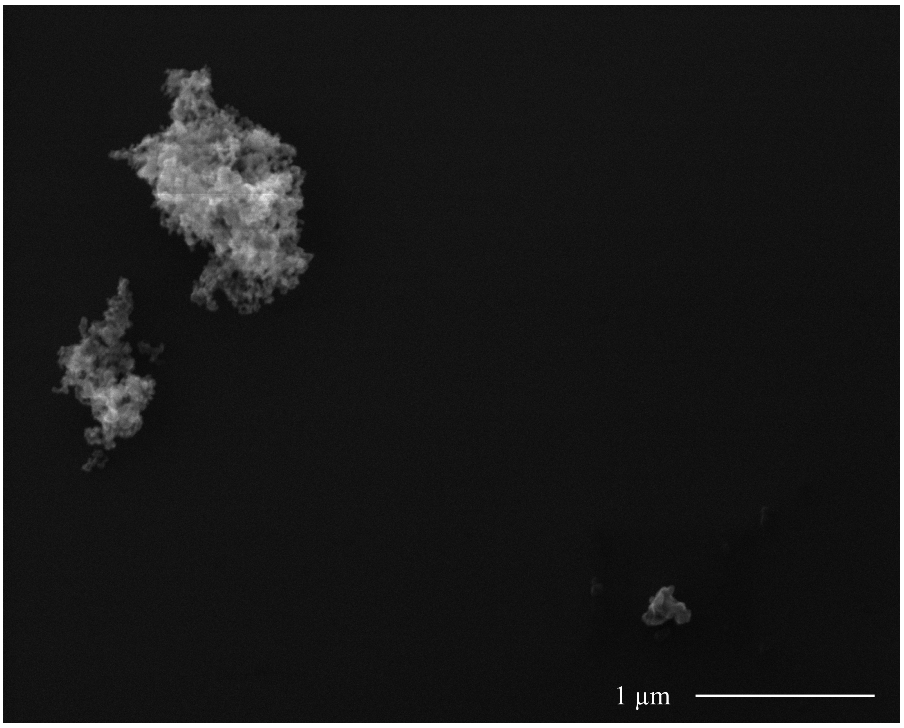


Figure 1.4: An SEM image (1 pixel = 1.27 nm) of TiO_2 nanoparticles with an average diameter of 25 nm. It shows the three agglomerates at the center of Figure 1.3.

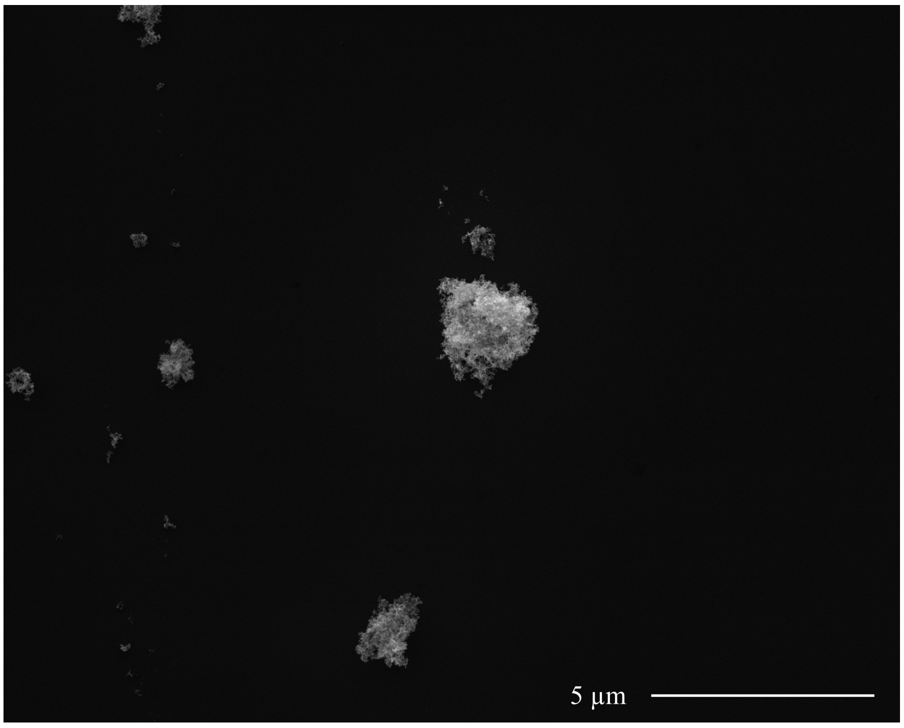


Figure 1.5: An SEM image (1 pixel = 5.08 nm) of ZnO nanoparticles with an average diameter of 10 nm.

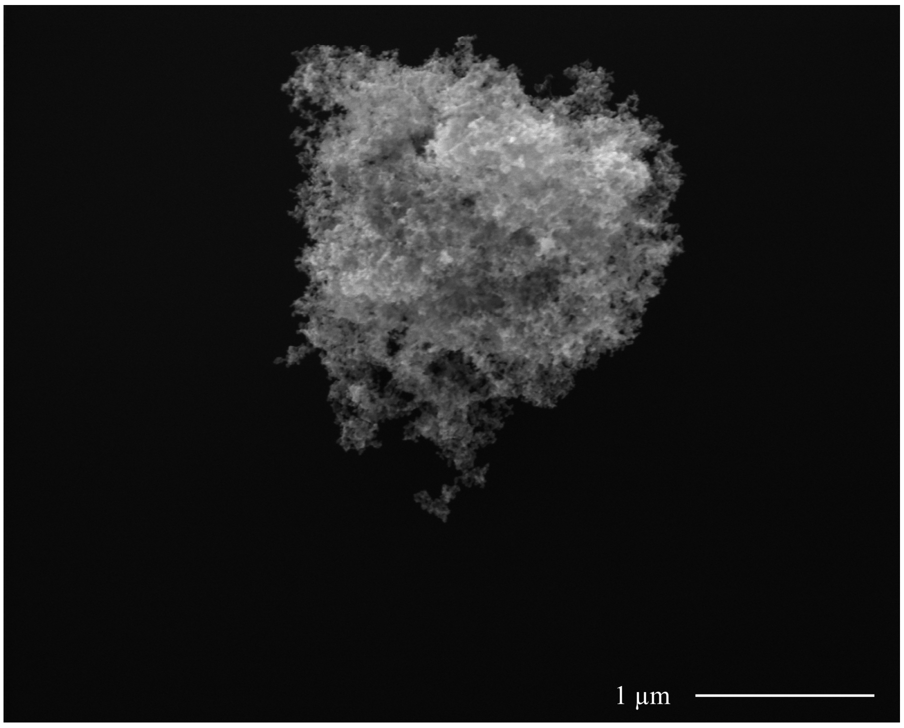


Figure 1.6: An SEM image (1 pixel = 1.27 nm) of ZnO nanoparticles with an average diameter of 10 nm. It shows the agglomerate at the center of Figure 1.5.

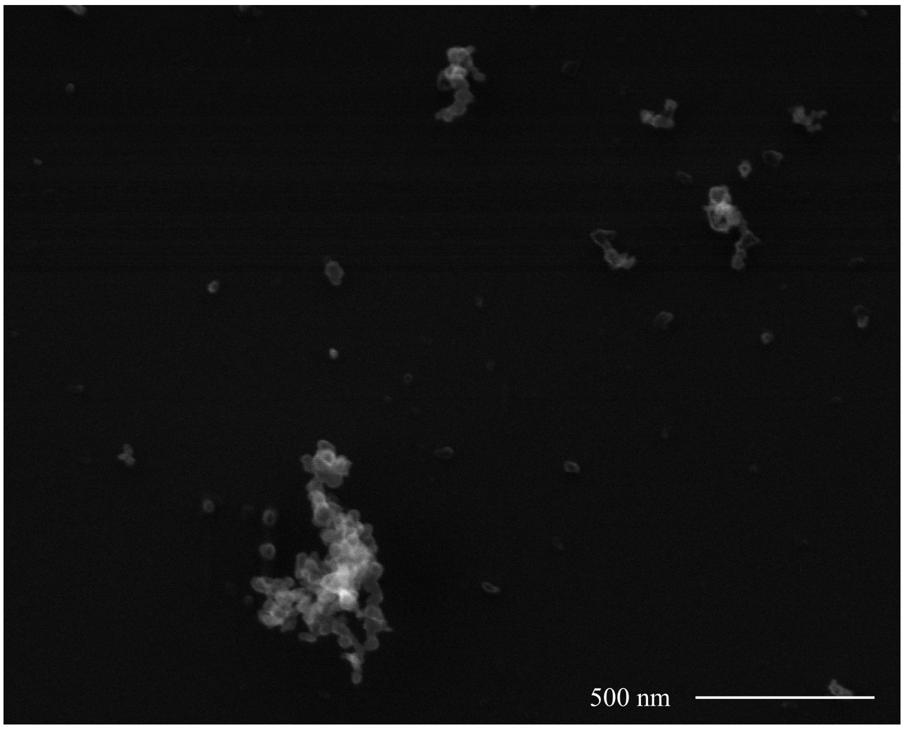


Figure 1.7: An SEM image detail (1 pixel = 1.27 nm) of diesel soot particles.

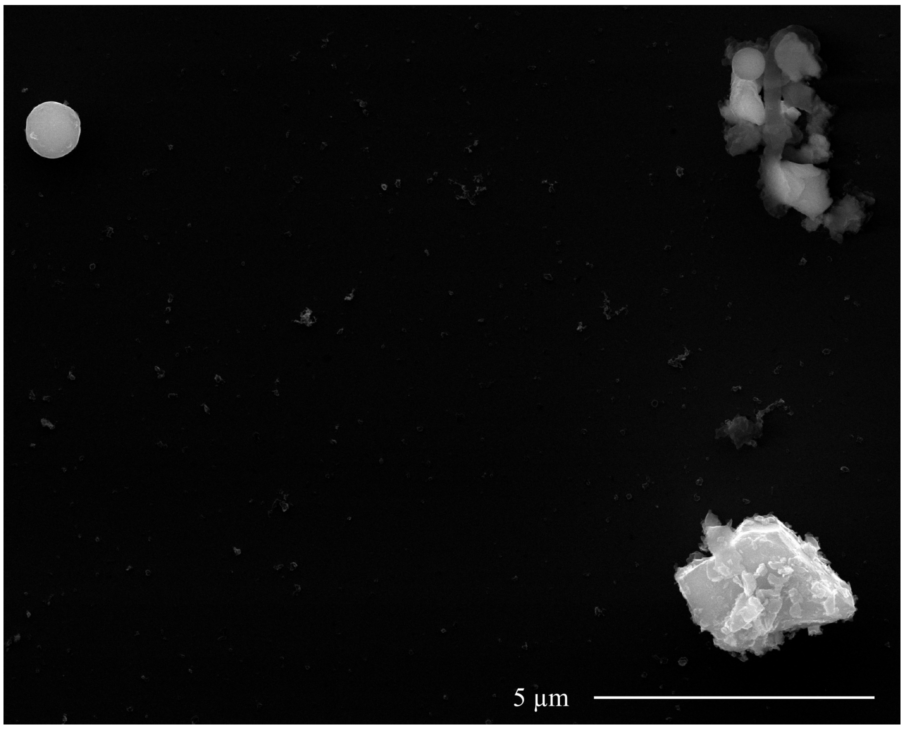


Figure 1.8: An SEM image detail (1 pixel = 5.08 nm) of particles gathered during abrasive cutting of steel. While the specific source of each of the particles is unknown, we believe that the upper left particle is steel solidified in the form of a sphere. The upper right particle presumably also derives from cutting process while the lower right particle probably is a silicate, a typical component of dust.

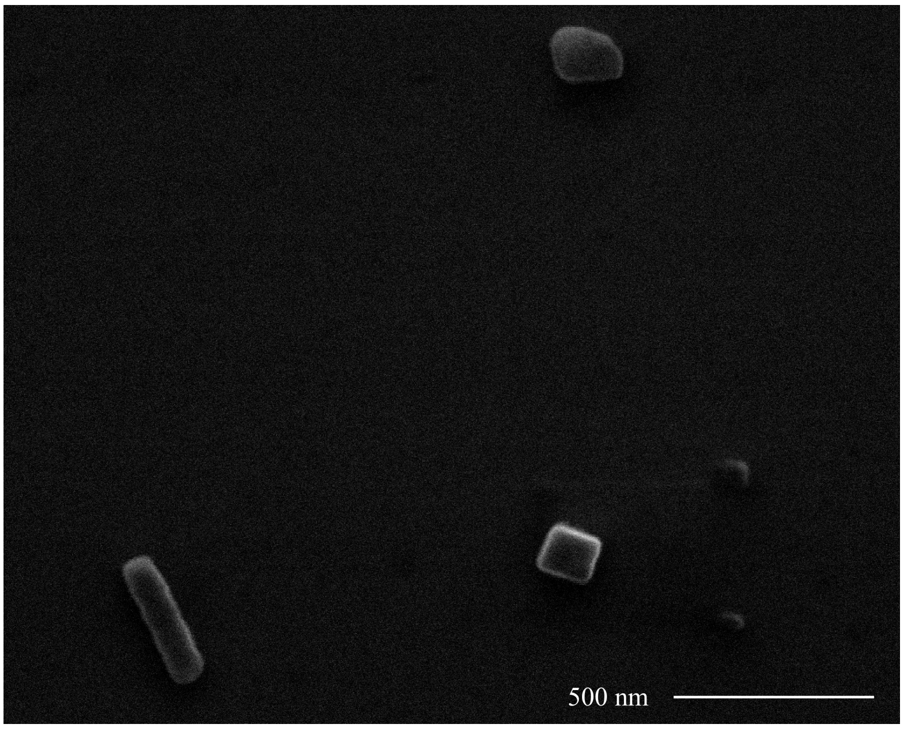


Figure 1.9: An SEM image detail (1 pixel = 1.27 nm) of particles from the ambient air in an urban environment. While the composition of the particles is unknown to us, we presume that the particle in the lower left is organic, possibly a bacterium. The rectangular particle in the lower right is a crystal such as salt.

PRIMARY PARTICLE A single nanoparticle which is not a compound of other particles. It may be part of an agglomerate or not.

NANOPARTICLE AGGLOMERATE An assembly of primary particles which are sticking to each other.

ENGINEERED NANOPARTICLE An intentionally produced primary particle with a diameter of 1 nm to 100 nm. It may be part of an agglomerate or not.

BACKGROUND PARTICLE A particle or an agglomerate of particles which has not been intentionally produced. If it is between 1 nm and 100 nm in size, it is also called *incidental nanoparticle*.

PARTICLE A collective term for primary particle or agglomerate, made up of engineered nanoparticles or background particles. In most cases, we will use this term to refer to a primary particle or agglomerate which forms a connected component in an SEM image and is surrounded by the image background.

After SEM images of particles have been created, they usually have to be manually examined. This involves finding every particle on each image and determining its type [Methner et al., 2009; Peters et al., 2009; Savolainen et al., 2010]. In other words, a person has to tell for each particle in every image if it is an engineered nanoparticle or a background particle. This is necessary because the concentration of a certain particle type in the air can directly be calculated from the number of particles of this type per image area. This process is very cumbersome and time-consuming as in some cases there may be in the order of one thousand particles per image. In addition, it is not enough to examine one image. Several images need to be analyzed to obtain reliable measurements. The actual number can range from a few to several hundreds [Bundesanstalt für Arbeitsschutz und Arbeitsmedizin (BAuA), 2012].

Therefore, the goal of this thesis is to develop a system to significantly reduce the manual work necessary to analyze such images. In the following section, we will shortly describe its main goals. A more detailed description of the exact circumstances of the application of the system and its goals can be found in Chapters 2 and 3.

1.3 GOALS

To reduce the manual work involved in the analysis of particles in SEM images, the minimum requirements are that our system is able to detect particles and agglomerates in the images and decide for every particle or agglomerate if it consists of engineered nanoparticles or not. This allows an estimation of the concentration of engineered nanoparticles in the air where the particles have been gathered. In addition, the system shall be able to adapt to changing requirements. This is desirable as nanoparticle research is still in its early days and further insights may change the focus of exposure measurement.

The system shall be implemented in the form of software running on a conventional PC. This has several potential advantages over the manual approach:

- An automatic system can work faster than a human.
- The user is not required for the identification of the particles. Therefore, the automatic approach can save most of the user's time.
- The system is able to find even small and faint particles, which a human could overlook.
- The system cannot only measure the number concentration of engineered particles, but can also estimate the volume and surface concentrations. Especially the surface concentration seems to be very important in predicting the potential toxicity of nanoparticles [Crosera et al., 2009].
- The computer cannot get fatigued and, thus, avoids errors after identifying many particles.

- The saved time and effort makes it possible to provide the system with more images than a human would be able to examine. This allows for more accurate concentration estimates.

1.4 ABOUT THIS THESIS

Although this thesis and the described work have been written and developed by me, for reasons of familiarity and legibility, I will stick to the plural *we* in the rest of the text.

This thesis contains a decent amount of mathematical formulas. As a result, while reading these, the reader may wonder about the definition of a specific symbol. For these cases, this thesis contains a list of symbols, which begins on Page xx. For each symbol, it contains a short definition and a list of pages on which it occurs. If the reader wishes more detail than the short definition, a more detailed description can usually be found on the page on which a symbol first occurs. In our opinion, it is crucial to make good use of the list because several symbols are used many pages after the point at which they are first defined. In addition, there is a list of figures beginning on Page xiv, a list of tables on Page xvii, a list of algorithms on Page xviii and a list of acronyms on Page xix.

In addition, this thesis uses special mathematical syntax. As an example, we use the expression $S' \sqsubseteq S$ to signify that S' is a multiset that contains subsets of the set S. Furthermore, we try to consistently use accents to indicate the type of a variable. In particular, $\vec{\cdot}$ is used for vectors, $\overset{\square}{\cdot}$ for matrices, while $\tilde{\cdot}$ stands for random variables and $\hat{\cdot}$ for estimates.

Because of its subject, this thesis contains many SEM images. In order to assure that these images have a consistent look in print as well as in digital form, we have resized these images so that they appear at a pixel density of 600 dpi. For this, we have chosen nearest neighbor interpolation because it allows the reader to clearly see the individual pixels of the original image on a magnified image detail. In addition, we have enhanced the contrast of the images by linearly scaling the image intensities so that the whole intensity range is used. We have chosen this approach in order to allow the reader to see low-contrast details even in difficult conditions such as on a monitor in the sunlight.

In this chapter, we have introduced the topics covered by this thesis and the target of the proposed system. In the remainder of this thesis, we will give details on the challenges encountered and the steps that are necessary when creating such a system. Chapter 2 will outline a realistic usage scenario used as the basis of the design considerations of our approach. In Chapter 3, we will define concrete goals to be accomplished by the proposed system. Chapter 4 will describe how the SEM images are created and which characteristic properties they possess. Chapter 5 will be the first of four chapters that describe the details of our system. In particular, it will outline its global architecture. In Chapter 6, we will explain how particle agglomerates can be differentiated from the image background. Chapter 7 will describe how each agglomerate can be represented as a small set of numerical features in order to enable the distinction of different classes of particles. The details of this distinction process will be the subject of Chapter 8. In Chapter 9, we will present an extension to our system that is able to predict its performance given the addition of more training data. Finally, Chapter 10 will look back at all previous chapters and provide some concluding remarks.

2 USAGE SCENARIO

The study of engineered nanomaterials is a relatively new field, which has its roots in the 1990s. It comes as no surprise that nanotoxicology, the research of the toxicity of nanomaterials, is even younger. Actually, the term nanotoxicology itself was only coined in 2004 [Donaldson and Seaton, 2012]. As a result, the insights gained are still preliminary and the properties of nanomaterials in focus are still shifting. Among others, possible factors for the toxicity of inhaled nanoparticle agglomerates are:

- Agglomerate count
- Primary particle count
- Agglomerate size
- Primary particle size
- Composition
- Mass
- Surface area

It is unclear how important the individual properties are in the assessment of the toxicity of such particles [Savolainen et al., 2010].

Therefore, it is not possible to reliably predict which classes of nanoparticles or nanomaterials will be of interest for the field of nanotoxicology in the future. This has two implications:

- An automatic system to assist in the exposure measurement of nanoparticles at workplaces has to be flexible enough to adapt to changing requirements.
- It is currently not possible to accurately determine the requirements and circumstances such a system will encounter in the near future.

Thus, when developing a system to automatically find and identify engineered nanoparticles in SEM images, one has to make some assumptions about the requirements and working conditions of the system. To the best of our knowledge, our work is the first approach to an automatic solution to image-based identification of engineered nanoparticles in a realistic scenario.

Given these insights from today's standpoint, we will outline a realistic scenario for the application of our system in this chapter. A workflow representing the usage scenario is described and requirements

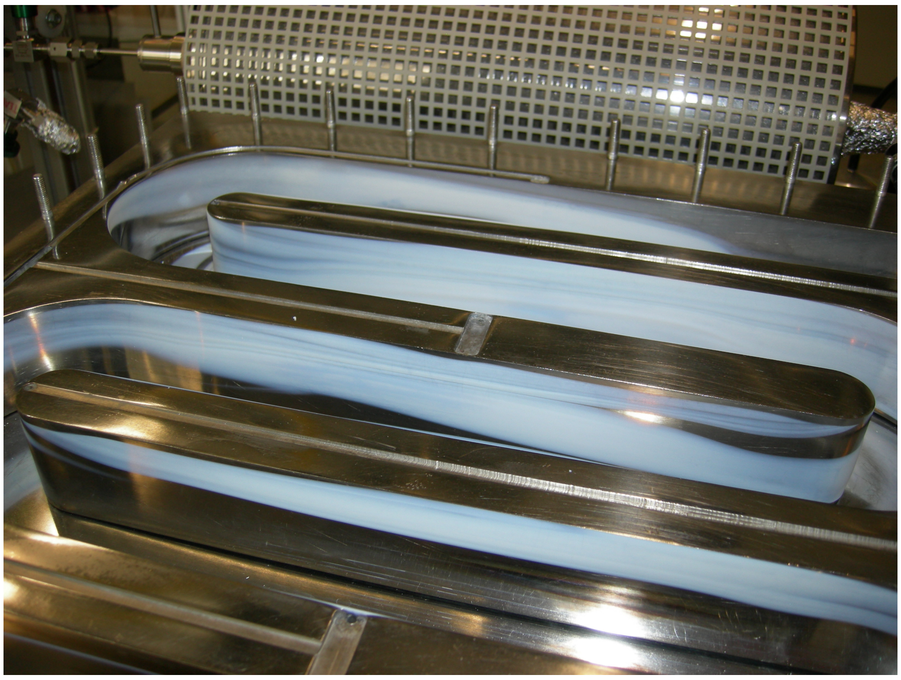


Figure 2.1: Picture of ZnO nanoparticle production on a laboratory scale. In the background, there is a hot wall reactor producing gaseous ZnO. It then deposits on the walls of the collector, which can be seen in the foreground. The deposited ZnO is visible as a white coating. Picture by BAuA.

and challenges the system has to fulfill and overcome are derived. The workflow has three steps:

SAMPLING Gathering particles from the air.

MICROSCOPY Taking SEM images of the gathered particles.

IMAGE ANALYSIS Analyzing the images using the system proposed in this thesis.

In Sections 2.1 to 2.3, these steps will be described in detail.

2.1 WORKPLACES AND SAMPLING

Nanoparticles are produced and processed either in a large industrial scope or in a smaller laboratory setting as in the example described above. The amount of produced nanoparticles ranges from less than 1 kg to more than 100 t per year. The production or processing sites may be fully enclosed, equipped with different types of ventilation systems or it may lack any such measures. Workers may or may not wear personal protective equipment such as respiratory masks at different points in time [Plitzko et al., 2013].

Figures 2.1 to 2.3 show an example of laboratory scale ZnO nanoparticle production. The hot wall reactor in the background of

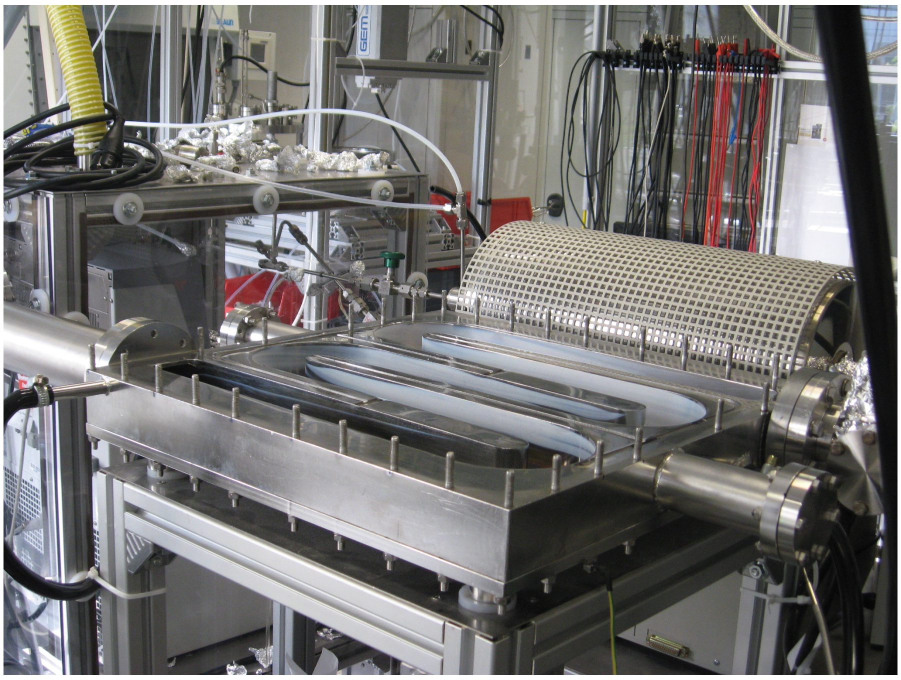


Figure 2.2: Picture of ZnO nanoparticle production on a laboratory scale. In the foreground, the collector with a white coating of nanoparticles can be seen. Behind it, on the right, there is a hot wall reactor used to produce gaseous ZnO, which then deposits on the walls of the collector. Picture by BAuA.

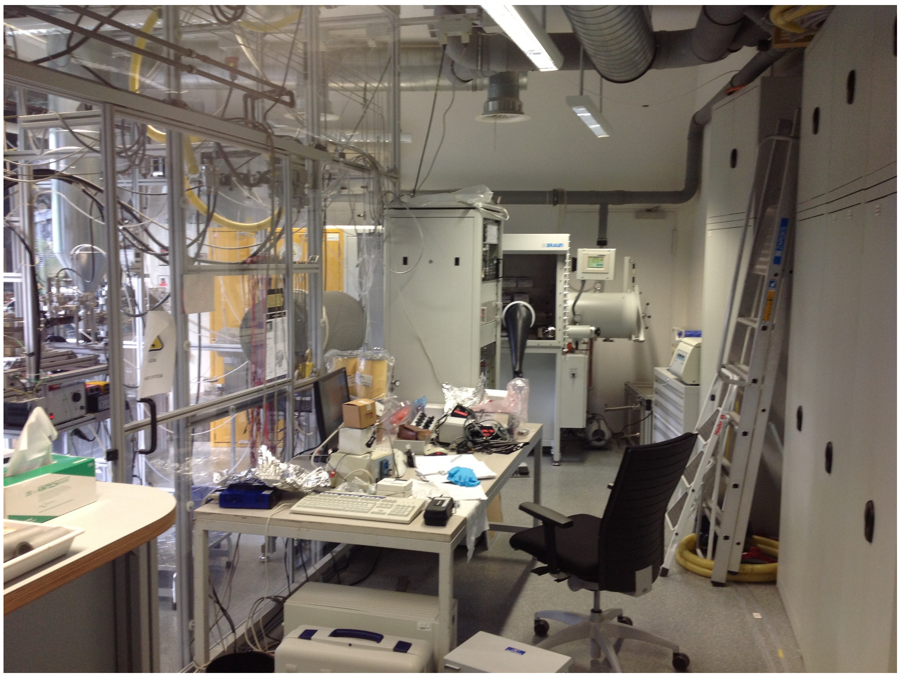


Figure 2.3: Picture of ZnO nanoparticle production on a laboratory scale. On the left, the enclosure which is housing the production apparatus is visible. On the far left, the enclosure has an opening, which can be closed by a sliding door. Picture by BAuA.

Figure 2.1 produces gaseous ZnO, which then is directed through the collector in the center of the picture. Subsequently, the ZnO deposits on the walls of the collector. It is visible as a white coating, which, afterwards, is scraped off to obtain nanoparticles. The surroundings of the collector are visible in Figure 2.2. In Figure 2.3 on the left, an enclosure, which houses the production apparatus, is visible. It shall prevent nanoparticles from escaping into the surrounding air. However, as can be seen in Figure 2.3, it is not always closed and for certain operations such as scraping off the nanoparticles or cleaning, workers have to go inside the enclosure. Therefore, it is likely that they are exposed to the nanoparticles and should wear personal protective equipment.

All of these circumstances have an effect on the amounts of engineered nanoparticles and background particles a worker is exposed to. In addition, the position of the worker with respect to the production or processing site and the process step, such as production or equipment cleaning, have an impact on the particle concentrations. Therefore, the relative amounts of engineered and background particles are not known beforehand. Hence, it would not be appropriate to make any assumptions regarding the ratio of engineered nanoparticles to background particles.

To gather particles a worker is exposed to, he or she wears a so-called precipitator, which picks up particles from the air (see Section 4.1). This allows a more precise concentration assessment than it would be possible with a fixed measurement system. At the same time, the worker's movement is not restricted. Additionally, it is possible to gather particles at a specific location using a stationary precipitator. In both cases, the particles are deposited onto a silicon wafer, which is later scanned using a scanning electron microscope (SEM).

The BAuA uses two particle counters in conjunction with the precipitators to directly assess the particle concentration in the air. The first is a TSI Condensation Particle Counter 3007 [TSI, 2012]. It is portable and can measure the particle concentration. However, it cannot provide any information about the sizes of these particles. The second is a stationary Grimm Scanning Mobility Particle Sizer (SMPS+C) [Grimm Aerosol Technik, 2012]. It is able to measure the size distribution of the airborne particles. Both counters are not able to distinguish engineered nanoparticles from background particles or liquid aerosols. Therefore, their measurements are not directly comparable to those using a precipitator and do not give direct information on the concentration of engineered particles in the air.

Typically, there is only one nanoparticle type at a time produced at a specific site. Therefore, we assume that, aside from background particles, only one type of engineered nanoparticles is gathered by the precipitators. This type is known beforehand. Hence, for the computation of the nanoparticle concentration, it is sufficient to be able to differentiate this one nanoparticle type from all background particles. From now on, we will assume that the system distinguishes one engineered nanoparticle type from background particles at a time. However, this does not mean that it is specialized to only one type of engineered nanoparticles. It merely means that the system will know beforehand which type it needs to identify.

Some background particle types typical for urban and industrial environments are:

- Diesel soot emitted by automobiles.
- Particles contained in welding fumes.
- Silicates such as quartz crystals, which commonly occur in soils.
- Soot from factory fumes or fireplaces.
- Organic materials such as bacteria.
- Rubber dust from car tires.
- Various salts from sources such as salt water.
- Metal dust from pantographs of electrically operated vehicles such as streetcars.
- Carbon dust from the brushes of electric motors.

These may all occur at a workplace in varying amounts. Particles of other origins may appear, as well.

2.2 MICROSCOPY

Using a precipitator, particles have been deposited onto a silicon substrate. In a next step, it is put into an SEM to take images of it. Apart from the corresponding metadata, such as resolution, these images are the only information available to our system. It cannot rely on the chemical composition to identify the particle types. The reasons for this will be explained in Section 2.2.1. Section 2.2.2 will gives details on how to chose the locations on the substrate of which images are taken. In Section 2.2.3, we will describe issues to be considered when choosing proper microscope settings. Finally, Section 2.2.4 will present properties of the SEM images which pose a challenge for the analysis of these images.

2.2.1 *Chemical Composition*

While most SEMs have the capability to use energy-dispersive X-ray spectroscopy to gain information about the chemical composition of the samples, it is not feasible to use such information for the concentration estimation of engineered nanoparticles. This is due to the following reasons:

- In order to obtain information about the composition of the particles, the microscope operator would have to find every particle on each image. Due to small size and faintness of many particles, this is a difficult task. In our experience, it would not be possible to detect 100 % of the particles.
- The microscope operator would need to focus on every single particle and perform an energy-dispersive X-ray spectroscopy analysis of it. Apart from the time needed to focus on the particle, zoom in, zoom out again and returning to the original image area, the analysis itself takes in the order of minutes per particle [Goldstein et al., 2003, p. 346]. Due to the potentially large number of visible particles, this would severely increase the time of the microscope operator spent per image.

2.2.2 *Image Locations*

The locations on the substrate where images are taken have to be chosen carefully. To obtain reliable statistical information about the concentration, the particle frequency in the images has to be representative of the whole substrate. Given that the average particle frequency is homogeneous on a certain area of the substrate, there are two possibilities to choose the image locations inside this area:

- Choose the image locations in a predictable pattern such as a grid. The individual images should not overlap.
- Choose the image locations randomly inside the homogeneous area.

To keep the average relative error of the particle concentration estimation below a given value, a certain number of images have to be analyzed. The necessary image count $n_f \in \mathbb{N}_{>0}$ can be calculated as

$$n_f = \frac{1}{{E_{PC}}^2 \kappa}, \tag{2.1}$$

where $\kappa \in \mathbb{R}_{>0}$ is the expected number of particles of the respective type per image and $E_{PC} \in \mathbb{R}_{>0}$ is the desired average relative error of the concentration estimation [Bundesanstalt für Arbeitsschutz und Arbeitsmedizin (BAuA), 2012]. If, for example, there is on average 1 particle per image and we want to achieve a relative error of 25 %, 16 images are necessary. If, on the other hand, we want to achieve a relative error of only 5 %, we need 400 images. Of course, κ depends on the used magnification.

2.2.3 *Microscope Settings*

As an SEM has a multitude of parameters in order to alter the properties of the resulting images, decisions have to be made on the settings of these parameters. Among them are the following:

- Magnification
- Image pixel count
- Brightness
- Contrast

2.2.3.1 *Magnification and Image Pixel Count*

The settings for magnification and image pixel count (number of pixels per image) determine the pixel resolution, which is measured as the number of pixels per length of the sample. They should be chosen according to the expected sizes of the agglomerates to be identified. These may vary considerably. In our images of ZnO, the smallest particle occupies an area of about $80\,\mathrm{nm}^2$ while the largest agglomerate takes up about $34\,\mu\mathrm{m}^2$. This corresponds to a ratio of 1:425 000. Therefore, there is a trade-off between a low and a high pixel resolution. A low pixel resolution has the following drawbacks:

- Small agglomerates are hard to detect.
- Small features disappear.
- Single primary particles cannot be discerned.
- Texture and contour of the agglomerates in the image are less informative.
- The primary particle size is reflected in the image only to a small degree.
- Different nanoparticle types which form similar agglomerates are hard to distinguish.

A high pixel resolution, on the other hand, also has disadvantages:

- Large agglomerates are not fully visible if the image shows a smaller area of the sample. As a consequence, the agglomerate size cannot be determined. It can even happen that an image is fully occupied by a single agglomerate.
- Images with a high pixel count take much time to be recorded.
- A higher pixel resolution leads to less sample area being scanned in the same time. To make accurate estimates of the particle concentration, a certain sample area needs to be analyzed. Therefore, the microscopy process takes longer.

An optimal solution would be to record images with a high pixel count and a moderately high pixel resolution. However, long image acquisition times cause a higher risk of vibrations disturbing the recording process (see Section 4.2.2). As a result, there is no solution combining all advantages. A trade-off has to be found to balance the disadvantages.

2.2.3.2 *Brightness and Contrast*

The brightness and contrast settings need to be adjusted in such a way that the full range of the signal can be mapped to the range of image intensities. In practice, many SEMs only use 256 or less different intensities. At the same time, the contrast of each image shall be maximized so that small signal differences are still visible in the image. To achieve both goals, it is necessary to adjust the brightness and contrast settings between images due to varying conditions (see Section 2.2.4.1). Hence, there is no fixed correlation between signal strength and image intensity. This also means that there is no direct relationship between particle properties and image intensity.

2.2.4 *Image Analysis Challenges*

Due to the nature of SEM and the nanoparticles, there are some challenges which have to be overcome to successfully analyze the images. One such issue is the low signal-to-noise ratio of SEM, similar to other nanoscale imaging techniques, which leads to high levels of noise [Ribeiro and Shah, 2006]. This noise can be reduced by increasing image acquisition times. However, besides the increased cost this brings about, the drawbacks described in Section 2.2.3.1 have to be kept in mind.

Another problem is that different particles appear with varying intensities, which will be explained in Section 2.2.4.1. Section 2.2.4.2 will illustrate the fact that some engineered particle types look very similar to certain background particles.

2.2.4.1 *Variable Intensity*

Small particles often show lower intensities than larger particles. The reason is that in large particles, the electron beam passes through more material. This leads to a greater number of emitted electrons. For a more detailed explanation, see Section 4.2.2. Figures 2.4 and 2.5 illustrate this very well. Figure 2.4 shows an SEM image detail of TiO_2 nanoparticles with an average diameter of 25 nm. The large agglomerate on the right with a diameter of about 1700 nm has a high contrast and is clearly visible. Figure 2.5 shows the same image, but in addition, every particle is marked with a rectangle around it. The single primary particle closest to the agglomerate has a diameter of

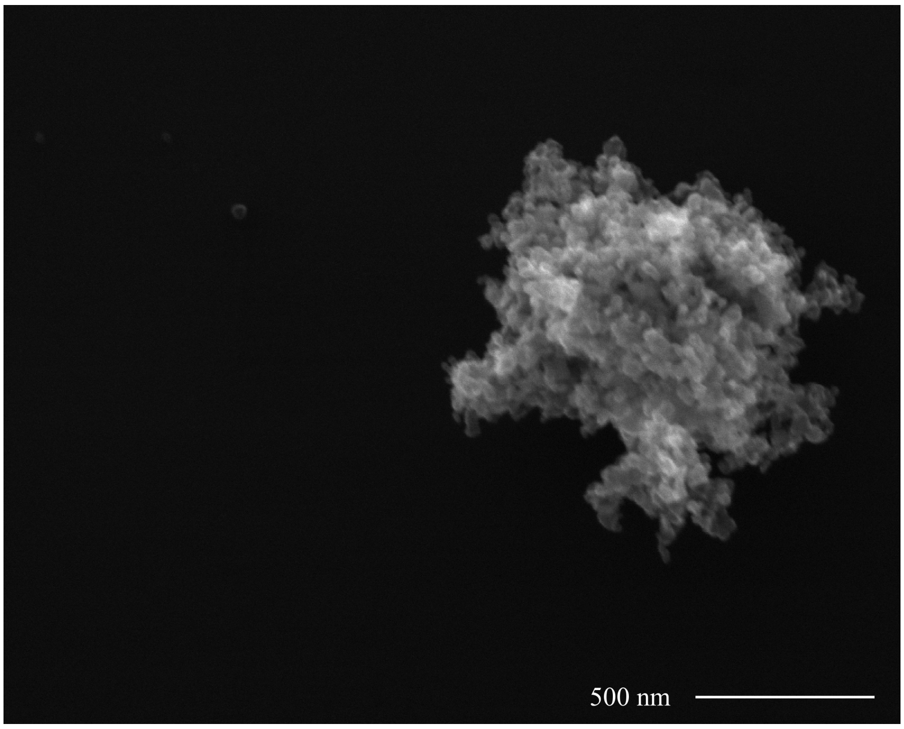


Figure 2.4: An SEM image detail (1 pixel = 1.27 nm) of TiO_2 nanoparticles with an average diameter of 25 nm.

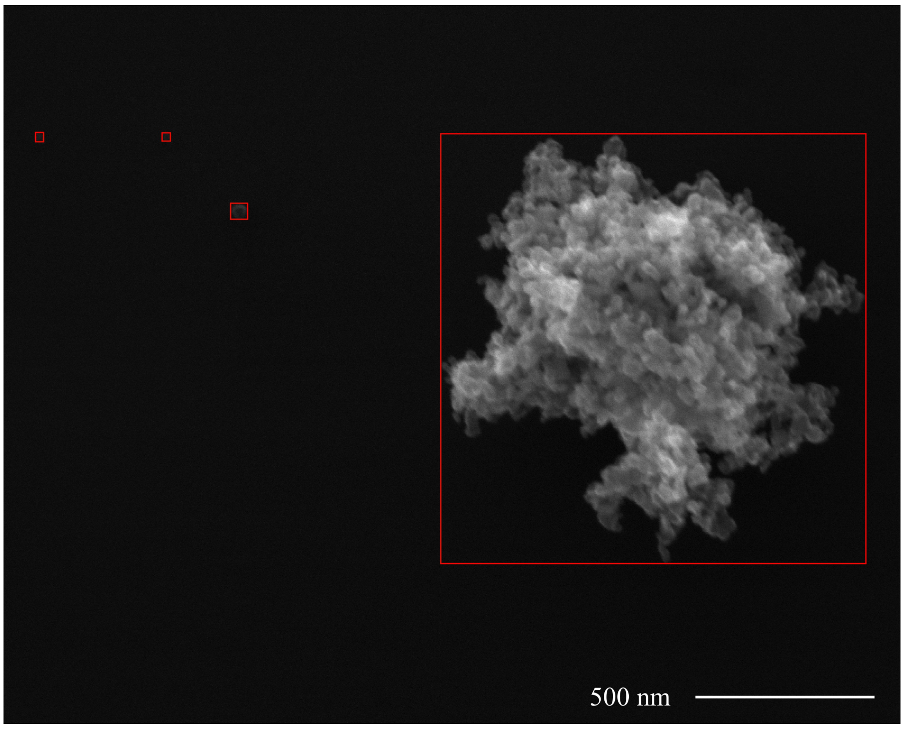


Figure 2.5: The same image detail as in Figure 2.4 with a rectangle around each particle.

about 50 nm. It can be spotted relatively well in the original image. The two primary particles at the left, however, can only be seen if one looks very closely. They have diameters of about 20 nm to 25 nm.

This poses a problem for human inspection as well as for an automatic solution. A human has to take a close look at every small part of the image in order to see such faint particles. We have found in our experiments that it is very likely that a human overlooks such small particles (see Section 6.4.5). For an automatic solution, small particles pose a problem as it has to differentiate them from noise artifacts.

2.2.4.2 *Similarity to Background Particles*

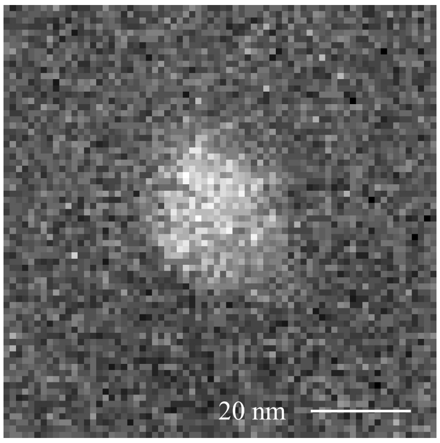
A major challenge in the task of identifying engineered nanoparticles in SEM images is that types with certain sizes behave very similar to incidental nanoparticles such as diesel soot. Due to comparable primary particle sizes, they form agglomerates in a similar way and often look almost identical in SEM images. Figure 2.6 shows a comparison of TiO_2 nanoparticles with a mean primary particle size of 25 nm and diesel soot particles. Figure 2.6a displays a single TiO_2 primary particle whereas, in Figure 2.6b, an individual diesel soot primary particle can be seen. Apart from the contrast, there is virtually no difference between the two image details. In Figures 2.6c to 2.6f, two small agglomerates of both TiO_2 and diesel soot are shown, which further illustrate that it is a big challenge to differentiate certain types of engineered nanoparticles from incidental nanoparticles like diesel soot. We have also observed these problems with ZnO particles with an average diameter of 10 nm.

2.3 IMAGE ANALYSIS

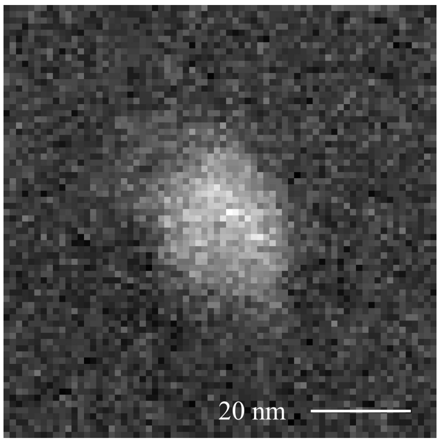
The image analysis of an automatic solution comprises two steps. In the first one, the system is trained to recognize a specific type of engineered nanoparticles. This will be explained in Section 2.3.1. The second step, where the actual particle identification is performed, will be described in Section 2.3.2.

2.3.1 *Training*

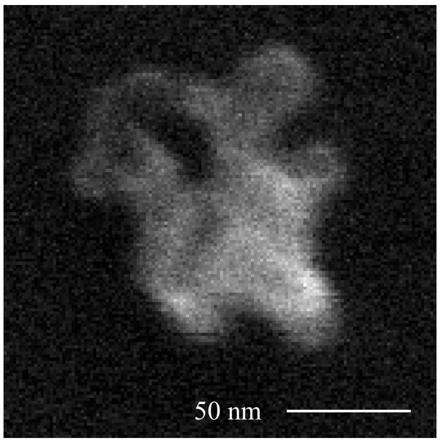
In order to learn the specific properties of a nanoparticle type and to be able to differentiate it from background particles, the system needs to be trained. Therefore, it needs to be provided with images of that particle type and, in addition, of a wide variety of background particles. This is necessary so that the system is able to characterize the difference between that particular type and particles which may occur in an image of a sample taken at a workplace.



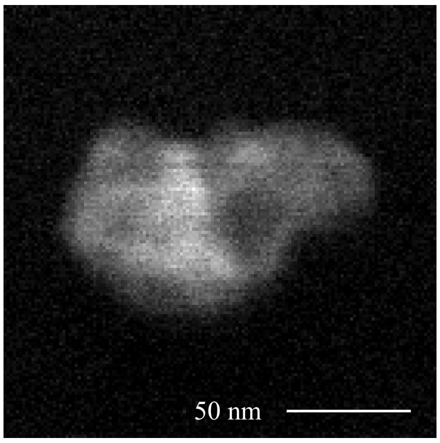
(a) A single TiO_2 primary particle.



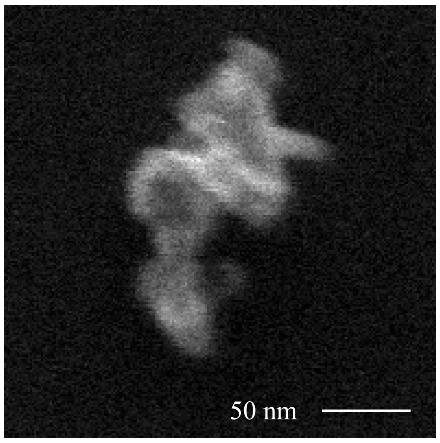
(b) A single diesel soot primary particle.



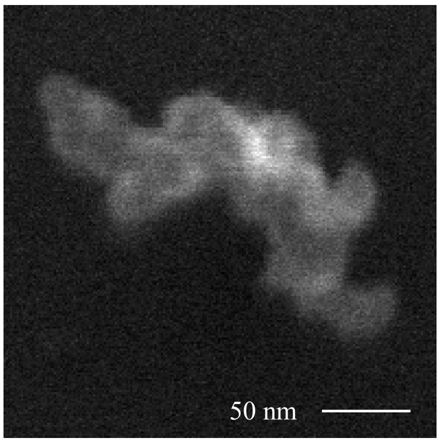
(c) A TiO_2 agglomerate.



(d) A diesel soot agglomerate.



(e) A TiO_2 agglomerate.



(f) A diesel soot agglomerate.

Figure 2.6: A comparison of SEM image details ($1\,\mathrm{pixel} = 1.27\,\mathrm{nm}$) of TiO_2 nanoparticles with an average diameter of $25\,\mathrm{nm}$ on the left and diesel soot on the right.

If a human operator is able to unambiguously distinguish the engineered nanoparticles from the background particles, images where both types occur at the same time can be used for the training. These may be images of samples gathered at workplaces. In cases such as the TiO_2 and ZnO samples described above, however, this procedure is not applicable. Because it is not possible to manually distinguish them from diesel soot, it would not be achievable to generate reliable training data for the system. Providing it with wrongly labeled particles as its training data would lead to errors in the automatic classification.

Therefore, the usual case is to give the system images with either only engineered particles of that specific type or only background particles. The former can be produced by releasing nanoparticle samples into a closed environment and gather them using a precipitator (see Section 4.3). Images containing only background particles can be created by taking air samples in environments where typical background particles but no engineered nanoparticles occur. The images which only contain background particles are reusable each time the system is trained to identify a new particle type. The images containing engineered nanoparticles have to be generated anew for each new particle type.

Taking SEM images is time-consuming and SEMs are expensive so that access is often shared among multiple parties. Therefore, it may be expected that only few images will be available for the training of the system, especially those containing engineered particles. This makes it hard to train the system in a way that it reliably classifies unknown particles because it is only provided with few examples. The problem is amplified by the fact that it is very difficult to optimize the number of particles per training image. If too many are present, the probability that they overlay each other is high, which changes the characteristics of the particles. However, the surface density cannot directly be measured when the sample is taken. Therefore, the sampling time must be chosen conservatively to avoid overlapping particles. This leads to fewer particles per image.

2.3.2 *Particle Identification*

After the system has been trained with images of a type of engineered nanoparticles, it can be used to identify them in images of samples gathered at workplaces. For that, SEM images recorded as described in Section 2.2 and the corresponding metadata are given as input to the system. It is assumed that the images have been recorded with the same microscope settings as those used for the training images, because particle properties may look different otherwise. This does not apply to the focus, brightness and contrast settings as these need to be adjusted to obtain sharp images with high contrast. The system then detects and identifies the particles of a specific engineered particle type. Finally, the software estimates and displays the concentration of engineered nanoparticles in the air where the sample was taken.

In this chapter, we have painted a realistic usage scenario for our system and derived the circumstances it has to cope with. In the next chapter, we will describe the goals such a system has to accomplish in order to work well in this scenario and to significantly reduce the manual work necessary for particle concentration measurement.

3 GOALS

After we have described the circumstances of its application, we will, in this chapter, further elaborate on the goals of our system to measure the concentration of engineered nanoparticles by analyzing SEM images of sampled airborne particles. Our primary goal was to reduce the manual work in analyzing the images. We wanted to develop a system to automate the monotonous work of detecting and identifying particles in SEM images. In addition, the goal was to reduce the time the user has to spend in front of the computer to a minimum. Therefore, apart from providing the system with training data, that is images of particles with corresponding type labels, and, of course, looking at the results, the user shall need to spend as little time as possible to use the system. This implies that the user shall not need to provide any parameters except for application-related metadata such as the used precipitator or microscope settings. In other words, the system should be fully automatic.

To adopt the work previously done by humans, the minimum requirements for the system are that it performs these two tasks:

DETECTION Locate the engineered nanoparticles and (optionally) the background particles in every image.

IDENTIFICATION Decide for every found particle if it either consists of engineered nanoparticles of a specific type or is a background particle.

In addition to these minimum requirements, further goals targeted towards further reducing the work for the user of the system are:

FUTURE VIABILITY Be able to adapt to new particle types and requirements.

PERFORMANCE PREDICTION Predict the influence of additional training data on the quality of the system's particle identification to guide the user's decision if more training samples shall be produced.

In the following paragraphs, we will describe these goals in more detail.

DETECTION In the first step, the system shall find all agglomerates of engineered nanoparticles in each SEM image. This includes primary particles of the same type not connected to any other particle. This means to determine the location and shape of each particle. In this case, a particle is considered as a collection of adjacent or overlapping primary particles. In particular, agglomerates are considered as single particles. In other words, a connected component in an SEM image is regarded as one particle.

Necessarily, the system will also find background particles in this step. However, it is not required to find all of them because they have no influence on the concentration estimation of engineered nanoparticles.

IDENTIFICATION Once every particle has been found, the system has to decide for each of them to which of the following two categories it belongs:

- The particle is an engineered nanoparticle of a predetermined type or an agglomerate of such particles.
- The particle does not contain engineered nanoparticles.

The second category contains all particles occurring in ambient air in industrial scenarios not intentionally produced by humans. (For a list of examples, see Section 2.1.) As explained in Section 2.1, it is sufficient for the system to be able to differentiate one nanoparticle type at a time from background particles. The specific type is known beforehand.

FUTURE VIABILITY As described in Chapter 2, the circumstances the proposed system will face cannot completely be foreseen. The requirements and the nanoparticle types the system is targeted at may change in the future due to further research on the health implications. Therefore, the system shall not be targeted at a specific type of nanoparticles. In addition, in order to train it to recognize nanoparticles similar to the ones already supported by the system, no changes to the source code shall be required. Furthermore, the training shall not require the user to set any parameters.

The system shall be easily adaptable to changing requirements. Therefore, it shall have a modular architecture so that single parts can be replaced or adapted to fit the new demands.

PERFORMANCE PREDICTION Generally, the more training samples a system can learn from, the better the classification performance will be. However, producing SEM images and manually labeling the particles costs time and money. So, the system shall be able to help the user make the decision if generating more training samples is worth the associated costs. More precisely, it shall be able to predict the influence further training samples will have on the performance of its ability to differentiate engineered nanoparticles from background particles. This helps the user to make a more informed cost-benefit analysis.

On one hand, this can save time and money by preventing the generation of additional samples which would have no effect on the classification performance. On the other hand, it can tell the user that a much better performance can be achieved with samples generated using a reasonable amount of work.

Additionally to these goals, we want to assess how well a system can identify engineered nanoparticles in SEM images compared to humans. We want to see whether an automatic solution is able to achieve similar or even better results compared to a manual inspection. A solution that only identifies a small fraction of the engineered nanoparticles will not produce helpful output. Therefore, an evaluation of such a system needs to incorporate the quality of its output. In order to act as a replacement for manual work, the system quality should be at least comparable to that of a human doing the same job.

Furthermore, we want to stress that our work is explicitly a proof of concept. To the best of our knowledge, nobody else has proposed a system targeted at detecting and identifying engineered nanoparticles in ambient air based on SEM images in a realistic scenario. Therefore, the following items are not the target of our work.

- An optimal runtime performance of all the parts of the system.
- Implementing and comparing all possible algorithms to find the best one for every part of the system and its workflow.
- Development of an optimal user interface.

These would have exceeded the scope of this thesis and can be the target of future research. We have the goal to develop a system which can be used and tested within reasonable time constraints. As there is no need for manual parameter optimization, the algorithms can run unsupervised, without requiring user interaction. The operation of the system is not time-critical. In addition, we have conducted literature research and developed our own methods within the scope of this project to compose a system which works well and is easy to use.

Using the types, count and projected shapes of the particles in the SEM images, different types of concentrations of these particles in the air at the workplace can be estimated. However, it is not our goal to determine how accurate these concentration estimates are and which formulas should be used to obtain them. These questions have to be considered by physicists. To answer them would have exceeded the scope of this thesis. Therefore, we limit ourselves to providing accurate data on the types, count and projected shapes of the found agglomerates.

Our main contributions are the following:

- Gathering and analyzing literature on automatic image-based particle analysis.
- Finding an appropriate workflow and architecture for our system.
- Identifying and analyzing the properties of the problem to be solved as well as those of the particles to be recognized and selecting suitable methods for every module of the system.
- Developing new algorithms tailored to the characteristics of the problem, especially for image preprocessing, feature computation and classification performance estimation.
- Creating an algorithm able to predict the performance of classifiers trained on training sets of different sizes and compositions.

More details on our contributions can be found in Section 10.1.

In this chapter, the goals we have set ourselves for the development of our system have been outlined. The next chapter will describe the images used as its input and the steps necessary for them to be created.

4 MATERIALS

In this chapter, we will describe how the images used as input for our system are created. It is important to have a basic understanding of this process in order to understand the characteristics of the images and the depicted particles. The basic workflow is as follows:

- Particles are gathered using a Precipitator.
- Images of the samples are taken using an SEM.
- The images and the used parameters are given as input to the system.

In reality, this process is repeated at least twice: First, particles of known type are sampled and used for the training of the system. Afterwards, unknown particles at a workplace are sampled and our system in the form of software running on a conventional desktop computer is used to identify them and calculate their concentration. In the following sections, we will describe the procedures and devices used for each of the steps as well as the images and particles used for our experiments. Section 4.1 will outline the sampling process. In Section 4.2, we will describe how the SEM images are created. The process to generate training samples will be portrayed in Section 4.3. Finally, Section 4.4 will describe the particles used for this thesis.

4.1 SAMPLING

As a first step, particles are gathered from the air at the workplace and deposited onto a silicon wafer. This is done with either a thermal precipitator or an electrostatic precipitator. They will be described in Sections 4.1.1 and 4.1.2.

4.1.1 *Thermal Precipitator*

The first sampling device is a so-called thermal precipitator. The model used by the BAuA is described by Azong-Wara et al. [2013]. Its dimensions are $4.5\,\mathrm{cm} \times 3.2\,\mathrm{cm} \times 9.7\,\mathrm{cm}$ and it weighs only $140\,\mathrm{g}$. Additionally, it can be powered by batteries. This allows it to be worn during an entire shift to assess a worker's personal exposure to nanoparticles.

The precipitator is equipped with a pump, which sucks in the surrounding air. This air is channeled through a pair of opposing metal

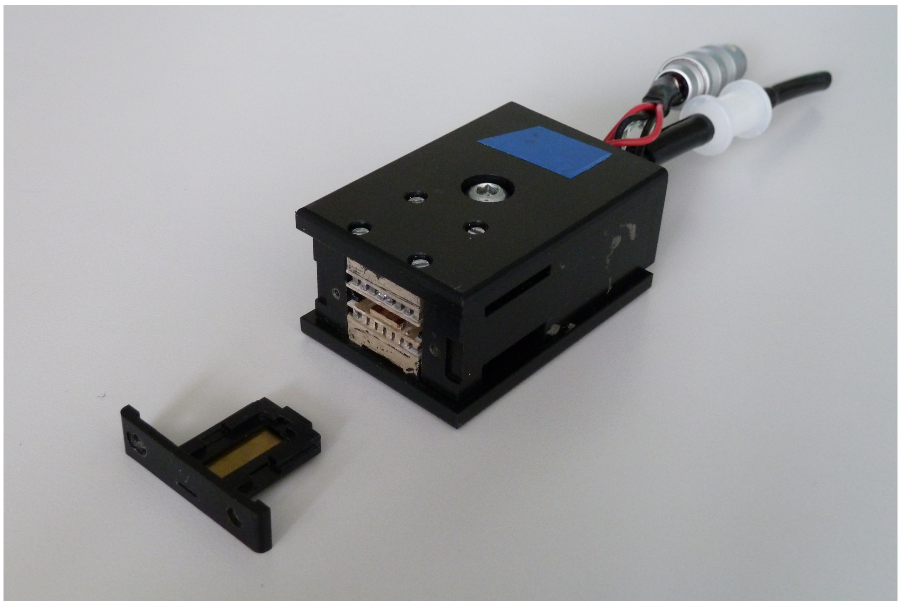


Figure 4.1: The thermal precipitator used by the BAuA. At the front, the tray which holds the silicon substrate onto which the particles are deposited can be seen. In the back, there is the cable and the tube which connect to the power source and the air pump, respectively. Picture by BAuA.

plates, which are 1 mm apart. One of them is heated so that their temperature difference is 15 °C. As a result of a mechanism called thermophoresis, the air is accelerated from the warmer towards the colder plate. Along with the air, the contained particles are accelerated and a certain percentage is deposited onto a silicon substrate with a surface of 10 mm × 20 mm. In Figure 4.1, the thermal precipitator used by the BAuA is shown. It has an air throughput of 2 ml/min.

4.1.2 *Electrostatic Precipitator*

The second device is called electrostatic precipitator. First, it electrically charges the airborne particles. This is usually done by a corona discharge caused by an electrode carrying a high voltage. The air stream with the charged particles is then directed through an electric field [Parker, 1997]. This drives the particles towards the silicon substrate, which is similar to the one used in the thermal precipitator. The BAuA uses a TSI Nanometer Aerosol Sampler 3089 [TSI, 2013], which can be seen in Figure 4.2. It weighs 3.75 kg and its dimensions are 20.3 cm × 25.6 cm × 22.8 cm. Thus, it is unsuitable to be carried by a person while doing work and, therefore, cannot be used to measure a worker's personal exposure to nanoparticles. However, its air throughput of up to 2.5 l/min allows it to collect particles at a fixed location much quicker than the thermal precipitator.

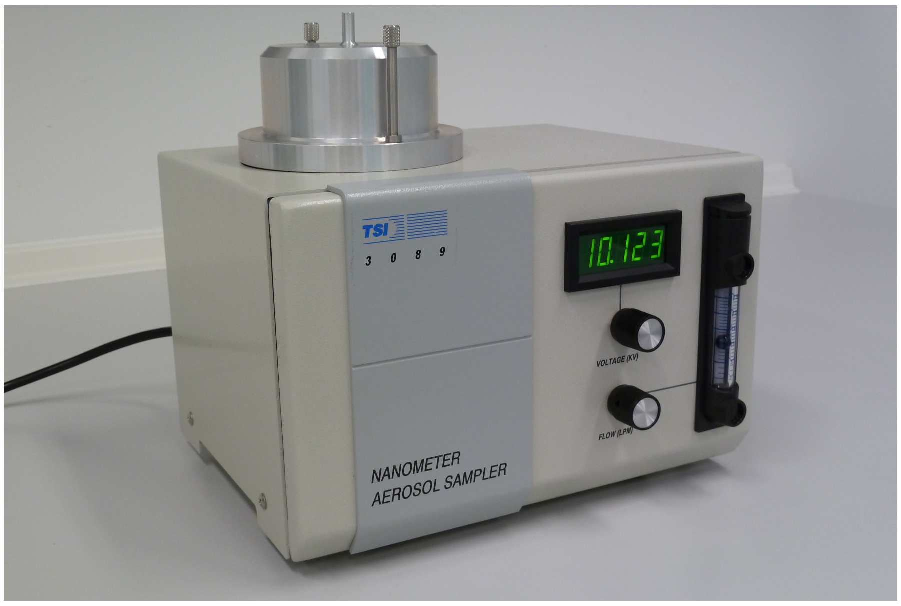


Figure 4.2: The electrostatic precipitator used by the BAuA. The silicon substrate onto which the particles are deposited is located inside of the metal cap on top of the precipitator. On top of that is a small duct through which the air is sucked in. Picture by BAuA.

4.2 IMAGING

In a next step, images of the silicon wafer onto which the particles have been deposited are taken using an SEM. Its functional principle will be explained in Section 4.2.1. In Section 4.2.2, we will explain properties that are typical for SEM images.

4.2.1 *Scanning Electron Microscopy*

The main concept of SEM is that a beam of electrons scans the surface of the specimen and resulting electron emissions are measured. It can achieve higher magnifications than light microscopy by using electrons instead of visible light. Figure 4.3 shows a schematic diagram of an SEM. The main components are located in a chamber under vacuum in order to keep the electron beam from interacting with air molecules. The electron gun at the top of the so-called microscope column generates electrons and accelerates them downwards. The energy to which each electron is accelerated is measured in eV (electron volt, $1\,\text{eV} \approx 1.6 \times 10^{-19}\,\text{J}$) and can be controlled to take values of $0.1\,\text{keV}$ to $30\,\text{keV}$ [Goldstein et al., 2003, p. 22]. This directly corresponds to the acceleration voltage, which is measured in V. Thus, as an example, using an acceleration voltage of $10\,\text{kV}$ would give every electron an energy of $10\,\text{keV}$.

Since the resulting electron beam is too coarse, its diameter needs to be reduced using two or more so-called electron lenses. After that,

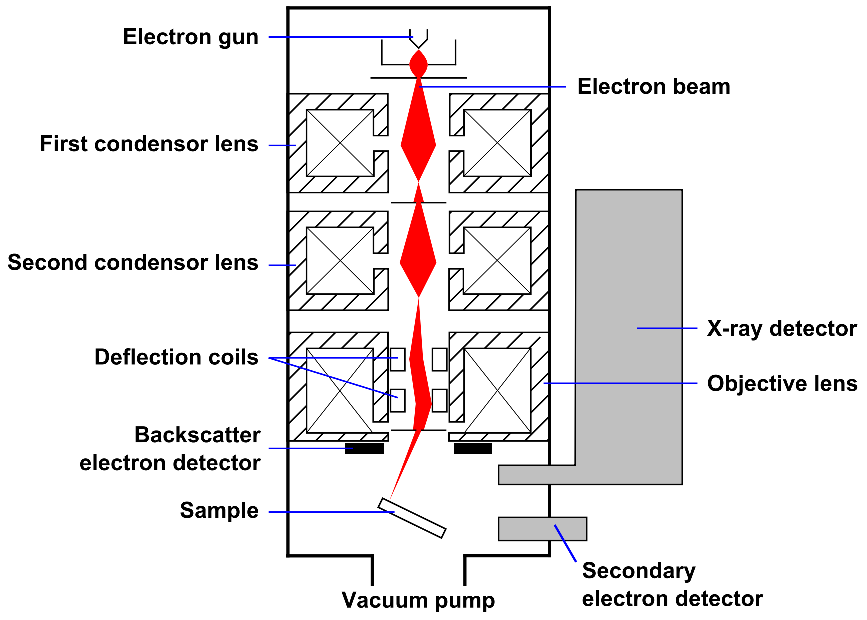


Figure 4.3: A schematic diagram of an SEM. An electron beam is generated by an electron gun. Its diameter is reduced using so-called electron lenses and it is deflected by deflection coils to scan the sample surface. Finally, detectors sense emitted electrons and X-rays. "Diagram of a scanning electron microscope with English captions" (`http://commons.wikimedia.org/wiki/File:Schema_MEB_(en).svg`, original by Steff, modified by ARTE and MarcoTolo) is licensed under CC BY-SA 3.0 (`http://creativecommons.org/licenses/by-sa/3.0/deed.en`).

the electron beam is deflected using so-called deflection coils so that it scans the surfaces of the specimen in a rectangular grid or raster. Each point on this grid corresponds to a pixel in the final image. The amount of deflection generated by the coils directly corresponds to the magnification of the image, which is defined as the ratio of the image length to the length of the raster on the specimen. If a higher magnification shall be achieved, the electron beam is deflected by a smaller amount so that the grid points are closer together.

Every SEM has at least one detector which collects the electrons ejected from the sample. The image intensity directly represents the amount of detected electrons using a linear relationship. This means that if the detector counts more electrons at a point in the grid, the corresponding pixel will have a higher intensity. The parameters of this relationship can be altered by the microscope operator using the contrast and brightness settings [Goldstein et al., 2003, p. 25].

There are two different kinds of electrons which can be detected: secondary electrons and back-scattered electrons. Most SEMs have separate detectors for both types (see Figure 4.3). Secondary electrons are ejected from the atoms of the specimen by the interaction with the electron beam. These have low energies and can only escape the

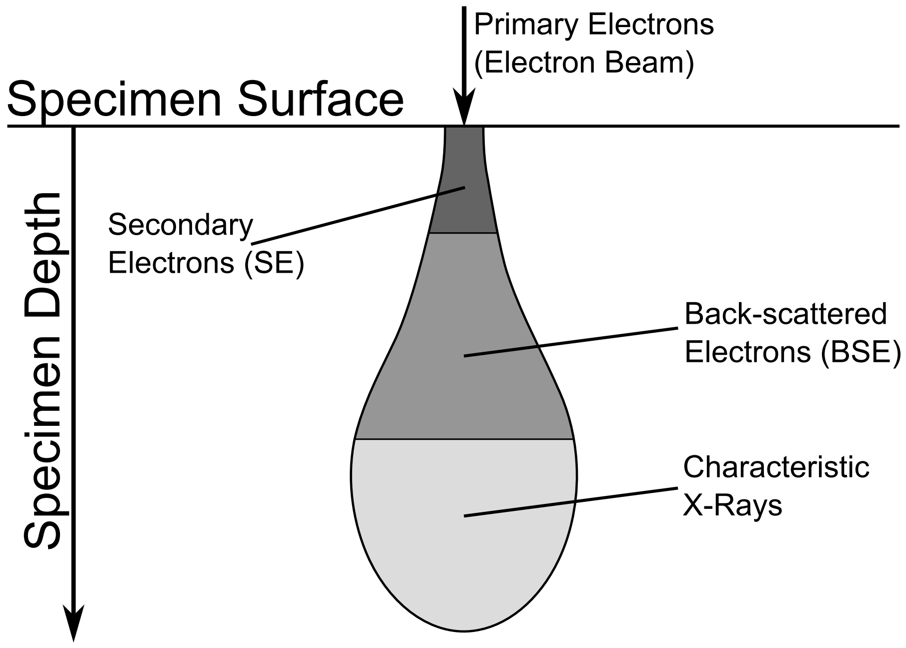


Figure 4.4: A schematic diagram of a part of the interaction volume of the electron beam of an SEM inside the specimen. The area above the horizontal line represents the vacuum inside the microscope. The area below it represents the inside of the specimen. The primary electrons, which compose the electron beam, penetrate the surface of the specimen at the top. The three delineated areas represent the volumes where secondary electrons, back-scattered electrons and characteristic X-rays, respectively, are generated with enough energy to escape the specimen and be detected. Secondary electrons are generated near to the specimen surface while back-scattered electrons stem from an area deeper inside the specimen. Characteristic X-rays are generated still further away from the surface. This work is a derivative of a graphic by Freundchen (`http://commons.wikimedia.org/wiki/File:Pear_interaction_SEM_german.svg`).

specimen if they are generated near the surface. Therefore, detected secondary electrons stem from a relatively small volume around the impact point of the electron beam (see Figure 4.4). This allows for images with a high resolution which contain surface details of the specimen.

Back-scattered electrons are electrons from the beam which are reflected by the specimen and have high energies. They can penetrate the sample to a great depth before returning and being detected. Thus, the detected electrons stem from a volume which is far less local to the impact point of the electron beam, both along the surface and perpendicular to it (see Figure 4.4). Consequently, the signal of the back-scattered electrons contains less information about the specimen surface and has a lower resolution compared to the signal generated by the secondary electrons. In our case, we want to take im-

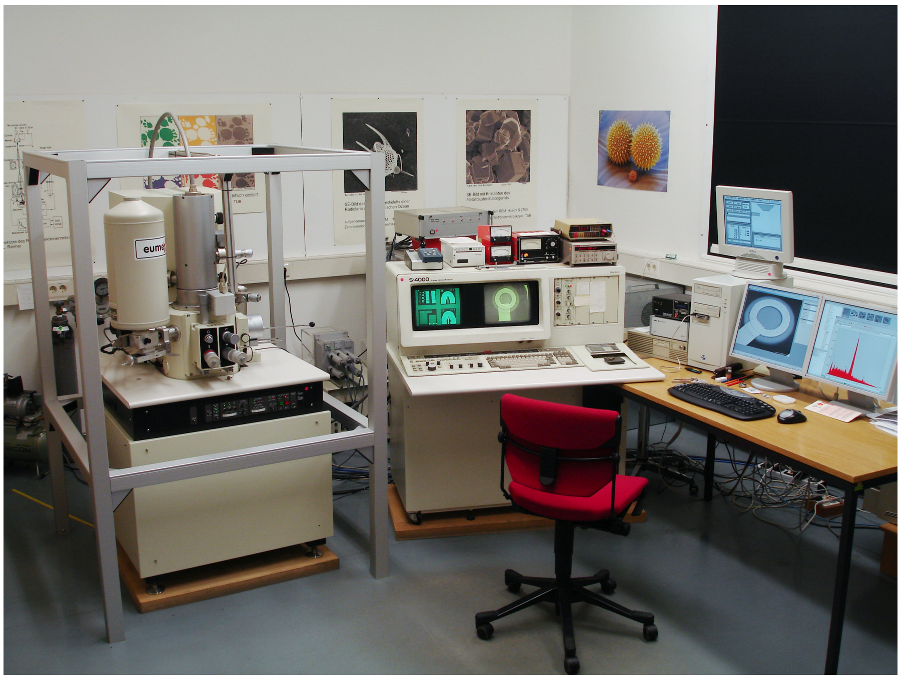


Figure 4.5: The Hitachi S-4000 SEM at the ZELMI. It has been used to take the images used for this thesis. On the left, the microscope itself can be seen. It stands on a special table to reduce vibrations, which could lead to image artifacts, and is surrounded by a frame to reduce the impact of electromagnetic radiation on the images. In the center, there is the console used to control the microscope. On the right, one can see the computer used to digitally save the images. Picture by ZELMI.

ages of nanoparticles, which are inherently small. Therefore, we have decided to use the secondary electron signal, as it allows for images with higher resolution. In addition, back-scattered electrons are more likely to pass the nanoparticles and penetrate the silicon substrate before being detected. This means that only part of the signal stems from the particles themselves and the substrate may contribute to the pixel intensity.

Most SEMs also have a detector to sense characteristic X-rays generated by the interaction of the electron beam with the specimen. This allows for so-called energy-dispersive X-ray spectroscopy (EDX), also abbreviated as EDS, which can be used to get information about the chemical composition of the sample. However, performing just a qualitative analysis takes a relatively long time (10 s to 100 s) [Goldstein et al., 2003, p. 357]. This means that it is not feasible to get composition information for every pixel or, in our case, for a grid fine enough to cover every separate particle. We have several images containing close to or more than one thousand separate particles. Therefore, the chemical composition of the particles is not available to us.

For our experiments, we use images taken with a Hitachi S-4000 SEM at the Zentraleinrichtung Elektronenmikroskopie (ZELMI) of the Technische Universität Berlin (TU Berlin) by BAuA employees. It can achieve a resolution of 1.5 nm and can be seen in Figure 4.5. For the images used in this thesis, a relatively high acceleration voltage of 20 kV has been chosen in order to improve the effective resolution. The working distance has been set to 10 mm. To achieve a relatively good signal-to-noise ratio, we have chosen a high oversampling factor of 128, which corresponds to the time used to record each pixel. The microscope operator has been advised to adjust the contrast and brightness settings for every image so that the average background intensity is about $\frac{1}{8}$ of the maximum possible intensity and the maximum intensity does not exceed the maximum possible intensity to avoid clipping (see Section 4.2.2.3). We have chosen a large image size of 4000×3200 pixels so that big agglomerates can be fully captured even at high magnifications. In order to be able to compare the performance at different pixel sizes, each chosen position on the substrate has been captured using two different magnifications. These are 5000 and 20 000, which correspond to pixel sizes of 5.08 nm and 1.27 nm, respectively.

4.2.2 *Properties of Scanning Electron Microscopy Images*

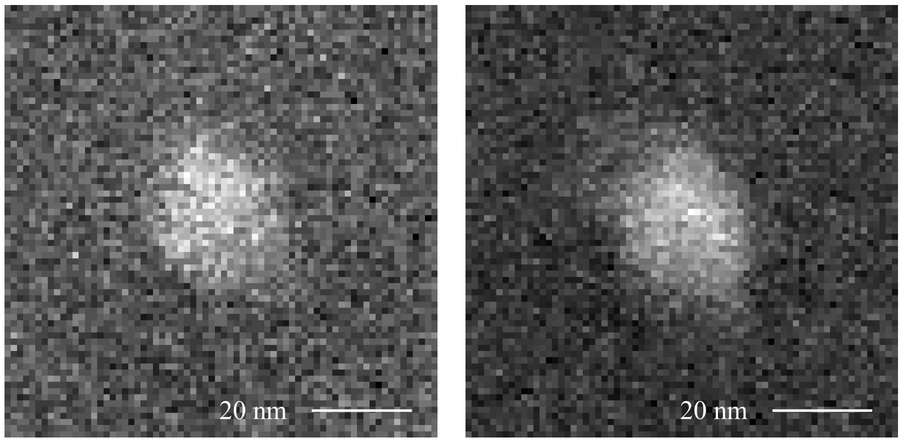
Images captured using SEM have several properties which may mask or distort the main image signal. These are:

- A low signal-to-noise ratio.
- Artifacts caused by vibrations or electromagnetic radiation.
- Intensity clipping.
- Variable particle intensity.
- Deposition of gas residuals.
- Intensity fluctuations around large particles.

We will describe them in detail in the following sections.

4.2.2.1 *Noise*

The number of detected electrons, which generate the image signal, follows a Poisson distribution. This means that the standard deviation divided by the expected value will get larger if the mean decreases. In essence, the signal-to-noise ratio will be low if the expected number of counted electrons is low. Compared to conventional optical images such as photographs, SEM images have an extremely low signal-to-noise ratio because of the low electrons counts [Ribeiro and Shah, 2006]. In addition, keeping imaging times low and achieving a high resolution further decrease the number of electrons. The two images in Figure 4.6 illustrate the low signal-to-noise ratio very well.



(a) A single TiO_2 primary particle. (b) A single diesel soot primary particle.

Figure 4.6: Two SEM image details ($1\,\mathrm{pixel} = 1.27\,\mathrm{nm}$) of a TiO_2 nanoparticle with an average diameter of $25\,\mathrm{nm}$ and a diesel soot particle, respectively. The particles look very similar. In addition, the signal-to-noise ratio of both images is very low.

In Section 6.3.1, we will present a custom algorithm to estimate the strength of the noise in each image. Furthermore, in Section 6.3.2, we will propose a noise removal method specifically targeted at SEM images using the output of the noise estimation algorithm to tune itself specifically for each image.

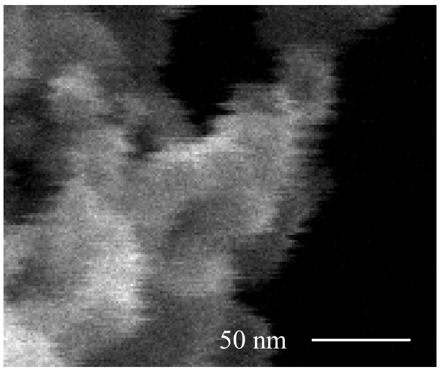
4.2.2.2 *Vibrations and Electromagnetic Radiation*

Due to the fact that SEM works at the nanoscale, even slight vibrations or electromagnetic radiation affect the image [Goldstein et al., 2003, pp. 59f.]. This includes human voices and vibrations and radiation emitted by trains and other machines. Such influences may affect the impact point of the electron beam or the signal amplification. Because of the fact that SEM images are captured row by row, this may cause horizontal edges which are not present in the sample. Such edges can be seen in Figure 4.7b. In addition, vertical edges may become jagged as pixels in different rows may picture points on the sample with different horizontal positions. This is illustrated well in Figure 4.7a, where all vertical edges have become jagged.

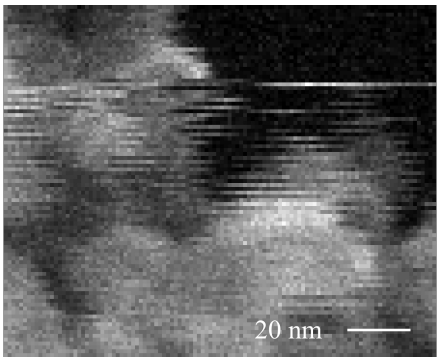
For the segmentation, these artifacts should not pose a big problem. For the classification step, however, the numerical features describing the shape and texture of particles have to be robust enough to deal with these distortions.

4.2.2.3 *Intensity Clipping*

In some circumstances, the electron detector detects more (or fewer) electrons than the amount that can be mapped to the available intensity range. In these cases, the maximum (or minimum) intensity is



(a) Jagged edges in an image detail (1 pixel = 1.27 nm) of TiO_2 nanoparticles.



(b) Artificial edges in an image detail (1 pixel = 1.27 nm) of TiO_2 nanoparticles.

Figure 4.7: SEM image artifacts caused by vibrations or electromagnetic radiation.

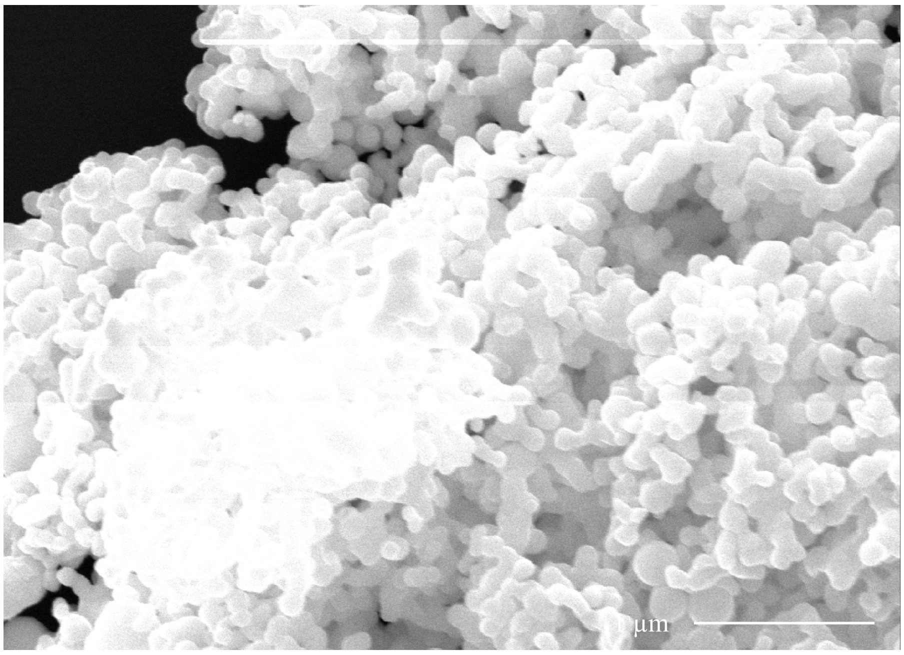


Figure 4.8: An SEM image detail (1 pixel = 5.08 nm) of Ag nanoparticles with an average diameter of 75 nm showing intensity clipping in the lower left corner. In addition, the artificial edge along the top is probably caused by a fluctuating beam current.

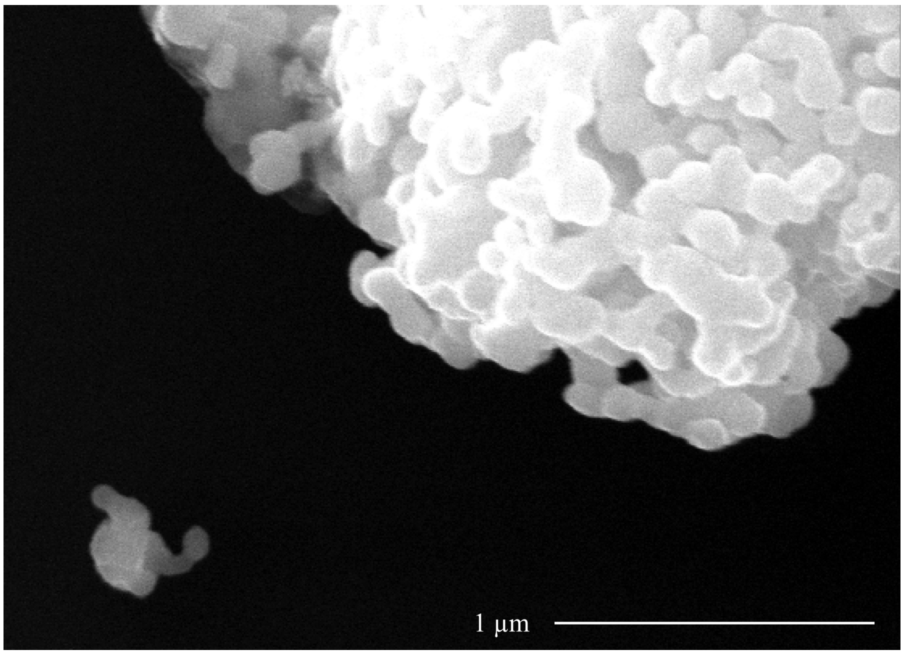


Figure 4.9: An SEM image detail ($1\,\mathtt{pixel} = 5.08\,\mathrm{nm}$) of Ag nanoparticles with an average diameter of $75\,\mathrm{nm}$ showing that larger agglomerates appear brighter than small ones partly due to electrostatic charging. Also note that shallow parts of the large agglomerate are not as bright as other parts.

assigned to the corresponding pixels. This leads to image areas whose pixels all contain the same intensity even though the sample contains texture at the corresponding position [Goldstein et al., 2003, p. 178f.]. One reason for this is a wrong combination of the contrast and brightness settings. The operator has to adjust these before taking the image using only a more noisy live image having a much lower resolution. Therefore, it is not easy to always find the right values.

Another reason for this can be that the current of the electron beam sometimes fluctuates [Goldstein et al., 2003, p. 34]. This leads to more (or fewer) electrons being accelerated towards the sample which in turn leads to more (or fewer) detected electrons. Figure 4.8 shows such a case, where the signal has been too intense to be mapped to the intensity range of the image. Therefore, the intensities have been clipped to the highest possible value and the part of the image has lost its texture.

Just like with artifacts caused by radiation and vibrations, the features describing the texture of particles need to be robust to deal with intensity clipping.

4.2.2.4 *Variable Particle Intensity*

In addition to the fact that the intensity may differ between images due to different contrast and brightness settings (see Section 2.2.3.2), large agglomerates usually appear brighter than small ones. This is visible in Figure 4.9 as well as in Figures 2.4 and 2.5 on Page 21, where small particles have much lower intensities than larger agglomerates. This is due to the fact that the electron beam travels further through a large agglomerate compared to small particles and has thus more chances to release secondary electrons. The beam can penetrate the sample up to a depth of several µm [Goldstein et al., 2003, p. 66]. This is also the reason why shallower parts of agglomerates appear not as bright as other parts, which can be seen in Figure 4.9, as well.

Apart from these differences, the intensity is mainly dependent on the surface orientation of the specimen [Goldstein et al., 2003, p. 96]. Therefore, the sides of the primary particles usually appear brighter than the parts facing upwards as can be seen in Figure 4.9. The composition of the particles has little influence on their intensity. There are exceptions, however, such as gold, which generates about two times as many secondary electrons as other materials [Goldstein et al., 2003, p. 95]. We have observed that the Ag nanoparticles appear much brighter than other particles in our images (see Figures 1.1 to 1.9 on Pages 5 to 9).

Small dark particles may pose a problem to the segmentation algorithm used in our system because they are difficult to distinguish from noise artifacts. However, this problem is not limited to an automatic solution. Humans searching for particles in SEM images may overlook them as well.

4.2.2.5 *Deposition of Gas Residuals*

The main SEM components and the sample is kept in a vacuum when images are taken. However, residuals of air still remain in the microscope column. These can react with the electron beam spot and be deposited onto the substrate [Ribeiro and Shah, 2006]. Such deposits are also called contaminations. This can lead to a rectangular layer on the sample which is clearly visible and may mask small details. However, the deposits can also be local at a corner of the view frame where the beam remains longer than at the other points of the frame. Figure 4.10 shows three examples of such deposits. They clearly look very similar to nanoparticles of a certain size range, such as the primary particle in Figure 4.6a. Such deposits cannot be avoided as the operator has to zoom in on a particle to adjust the focus for each new position on the substrate.

In addition, it is also possible that these deposits build up along the edge of an image to form a bright line. An example of this can be seen in the lower right image in Figure 4.10. This is caused by the fact that the electron beam shortly remains at the same position at the start of each line.

For our system, it will be very difficult to distinguish deposits inside the image from engineered nanoparticles. In fact, they do not

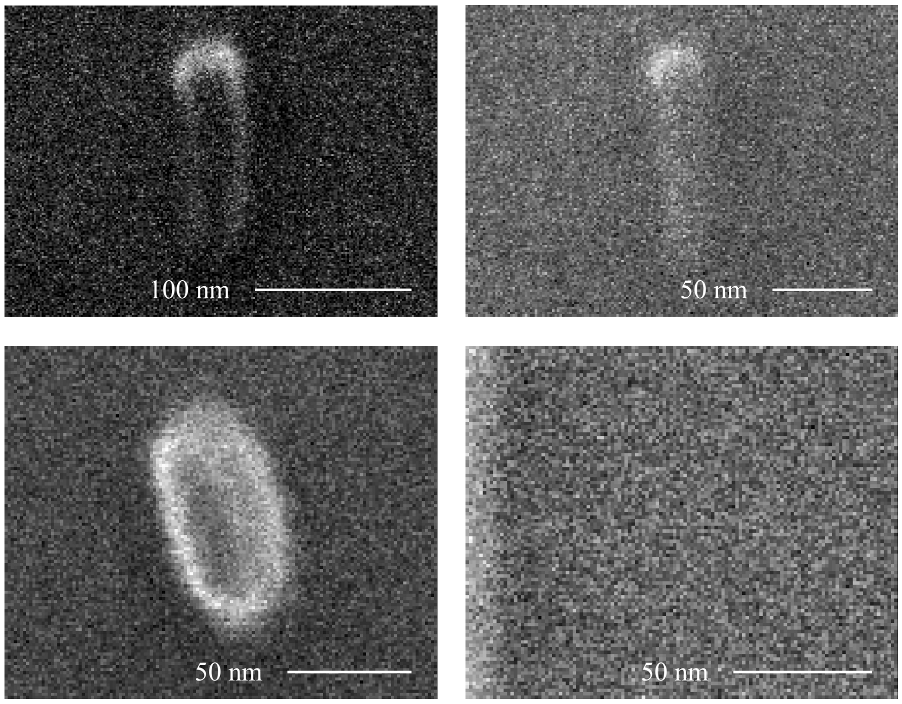


Figure 4.10: SEM image details (1 pixel = 1.27 nm) of deposits of gas residuals on the silicon substrate. These can easily be mistaken for engineered nanoparticles of a similar size, such as the one in Figure 4.6a. The image detail in the lower right shows the left edge of an image. Here, the leftmost pixels appear brighter because of gas deposition.

only seem like but are indeed nanoparticles created by the electron beam. Deposits along an image edge may be detected as particles by our system. The classification steps then has to distinguish them from engineered nanoparticles.

4.2.2.6 *Intensity Fluctuations around Particles*

Image areas around a large agglomerate can appear much darker or brighter than other areas. An example of this can be seen in Figure 4.11. This image shows a part of the image background to the lower right of a large Ag agglomerate. The area nearest to it appears much darker than the other areas. This is due to the fact that the agglomerate is located directly between this part of the substrate and the electron detector. This causes secondary electrons emitted in this area and normally detected by the detector to be blocked by the agglomerate. These blocked electrons never reach the detector leading to a lower intensity.

On the other hand, areas next to a large agglomerate in the direction of the detector appear brighter than normal. Here, electrons emitted in this area colliding with the agglomerate may be reflected

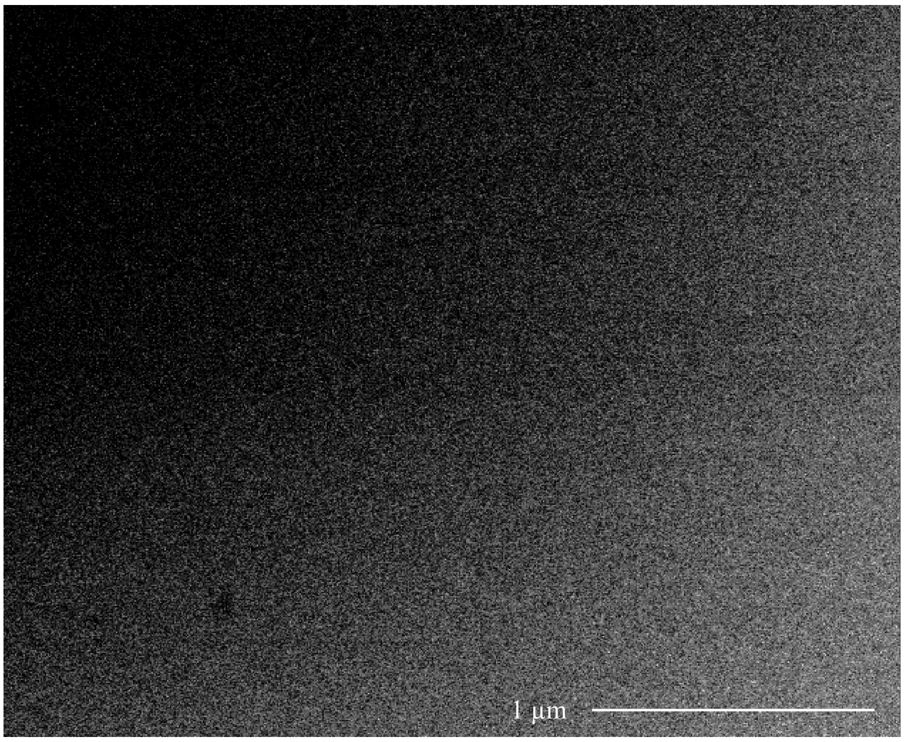


Figure 4.11: A part of the image background directly besides a large Ag agglomerate (1 pixel = 5.08 nm). The area in the upper left corner of the image, which is nearest to the agglomerate, appears much darker than the rest of the background.

towards the detector or release additional secondary electrons that are detected, as well.

Small particles in the shadow of larger ones are hard to detect because they appear darker than the image background in other parts of the image. We have observed this effect in one of our experiments. It is explained in Section 6.4.5.

4.3 TRAINING SAMPLE GENERATION

If particles of multiple types are gathered at once, it is often impossible to differentiate the types afterwards. However, incorrectly labeled training samples may confuse the system and may cause it to make false decisions when classifying unknown samples. For background particles, it is sufficient to only gather training samples in environments where one can be sure that no engineered nanoparticles occur. These can be urban environments near roads or other surroundings where only background particles occur. For engineered particles, however, it is important to only gather exactly one type and no other engineered nanoparticles or background particles. This can be ensured by sampling the nanoparticles in an enclosed container where other particles are unlikely to occur. In case of the BAuA, small

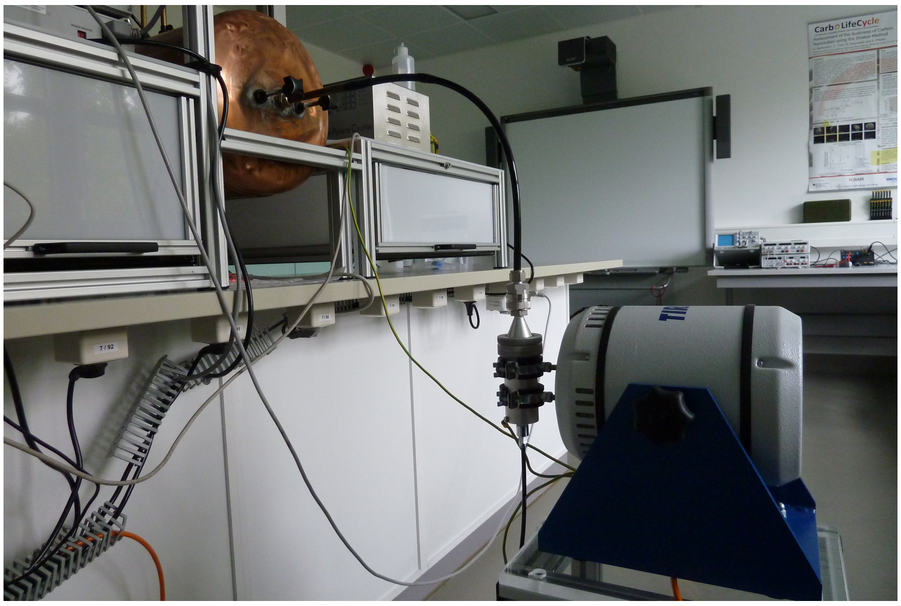


Figure 4.12: The equipment used by the BAuA to generate training samples of engineered nanoparticles. The apparatus on the lower right shakes the nanoparticles so that they become airborne and move into the tank on the upper left. Connected to the tank but not visible is a precipitator to gather the particles and deposit them onto a silicon substrate. Picture by BAuA.

amounts of nanoparticles are requested from the manufacturer. These are then brought into the air inside an airtight tank using the so-called shaker process, which is described by Spurny et al. [1975]. The used apparatus can be seen in Figure 4.12. Connected to the tank is a TSI Nanometer Aerosol Sampler 3089 [TSI, 2013], which gathers the particles as described in Section 4.1.2.

4.4 PARTICLES USED IN THIS THESIS

For the experiments in this thesis, we use three different types of engineered nanoparticles. These are:

- A mixture of Ag nanoparticles with average diameters of $50\,\mathrm{nm}$ and $100\,\mathrm{nm}$.
- TiO_2 nanoparticles with an average diameter of $25\,\mathrm{nm}$.
- ZnO nanoparticles with an average diameter of $10\,\mathrm{nm}$.

The average diameters given here are according to the manufacturers' specifications. The diameters of individual nanoparticles may deviate from that specification to a relatively high degree.

The BAuA has chosen these particle types because they are on the OECD's list of representative manufactured nanomaterials for safety testing [Organisation for Economic Co-operation and Development

Table 4.1: A list of all SEM images and particles used for the experiments in this thesis. Above the horizontal line are engineered nanoparticles and below background particles are listed. The particle counts in this table are given as counted by our system. Here, multiple primary particles being connected in an image are regarded as a single particle.

PARTICLE TYPE	IMAGE COUNT	PARTICLE COUNT
Ag (50 nm and 100 nm)	26	109
TiO_2 (25 nm)	44	845
ZnO (10 nm)	48	1774
Abrasive cutting dust	8	1424
Ambient air	12	156
Cigarette smoke	2	5
Construction dust	13	42
Diesel soot	14	6860
Welding smoke	12	792

(OECD), 2010] and available to the BAuA. In addition, these particle types are commonly used in commercial products. TiO_2 and ZnO have also been chosen as they are very similar to accidentally created nanoparticles such as diesel soot (see Section 2.2.4.2).

Furthermore, we use images of background particle samples that are typical of urban or industrial environments. These contain a selection of the particle types listed in Section 2.1.

A list of all SEM images and particles used for the experiments in this thesis is given in Table 4.1. We use images of the following background particle samples:

- A sample taken during abrasive cutting of steel.
- A sample taken from the ambient air in an urban environment.
- A sample taken while someone is smoking a cigarette.
- A sample taken on a construction site.
- A sample taken of the exhaust fumes of a diesel car.
- A sample taken during welding.

In almost all cases, two images have been taken of each selected position on the substrate, one using a magnification of 5000 (1 pixel = 5.08 nm) and one using 20 000 (1 pixel = 1.27 nm). This means that about half of the images have a pixel size of 5.08 nm while the rest has a pixel size of 1.27 nm. In addition, the area covered by an image with a magnification of 20 000 is usually also covered by an image with a magnification of 5000. In a few cases, there are one, three or four images of a single position on the substrate, respectively. However, we have no images in our database that have a pixel size different from the two mentioned above.

Example images of our engineered nanoparticles can be seen in Figures 1.1 to 1.6 on Pages 5 to 7. Examples of background particles can be found in Figures 1.7 to 1.9 on Pages 8 and 9.

In this chapter, we have described the equipment used to create the images for this thesis and the particles used in our experiments. In the next chapter, we will present literature related to the problem we have chosen to solve and we will establish the architecture and workflow of our system.

5 SYSTEM DESIGN

In this chapter, we will first give an overview of the literature on automatic image-based particle detection and classification in Section 5.1. In Section 5.2, we will then, based on the literature, devise the architecture and workflow of our system.

5.1 RELATED WORK

We have found and examined 33 publications on automatic image-based particle analysis. Not included are papers that say that a commercial image processing solution has been used but do not describe details of the process [Vezey and Skvarla, 1990; Reetz et al., 2000; Donskoi et al., 2007]. The literature on problems concerning particles with similar characteristics to ours is very sparse. Table 5.1 gives an overview of the types of particles analyzed in the examined literature. We have left out prior publications by the same author groups covering earlier versions of their systems targeted at the same particle types. It is easily visible that the publications cover a wide range of particle types with substantially different shapes and characteristics. Therefore, the methods used to detect and classify them may not be appropriate to be applied to nanoparticle agglomerates.

To the best of our knowledge, there are only three papers on automatic image-based analysis of engineered nanoparticles. We will briefly describe them now.

Fisker et al. [2000] analyze TEM images of nearly spherical nanoparticles. In contrast to our problem, the particles form no agglomerates and do not overlap in the images. Therefore, the image characteristics are very different. Furthermore, the paper is only interested in estimating the particle size distribution and does not perform any kind of classification.

Flores et al. [2003] classify small nanoparticles in high resolution TEM images. The particles are composed of very few atoms, at most a few thousand. The image characteristics differ to a great degree from ours as the single atoms of the particles are clearly visible and no agglomerates are formed. The nanoparticles shall be classified based on the arrangement of their atoms rather than their size or composition.

Oleshko et al. [1996] examine agglomerates of Ag nanoparticles in TEM images. Their shapes are very similar to the agglomerates analyzed by us. However, their texture in the images is very different due to the functional differences of SEM and TEM. In addition, no classification is performed. In fact, only the fractal dimension of the

Table 5.1: Overview of the examined particle types in the literature on automatic image-based particle analysis.

PARTICLE TYPE	PUBLICATIONS
airborne particles	Germani and Buseck [1991] Kindratenko et al. [1994, 1996] Wienke et al. [1995] Oster [2010] Genga et al. [2012]
aquatic life forms	Kindratenko and Van Espen [1996] Luo et al. [2004]
engineered nanomaterials	Oleshko et al. [1996] Fisker et al. [2000] Flores et al. [2003] Oster [2010]
crystals	Kindratenko et al. [1996, 1997] Calderon De Anda et al. [2005] Yu et al. [2007]
pollen grains	Langford et al. [1990] Rodriguez-Damian et al. [2006] Ranzato et al. [2007]
polyethylene powders	Greco and Maffezzoli [2004]
printing dust	Hopke and Song [1997]
quartz grains	Thomas et al. [1995] Drolon et al. [2000] Oster [2010]
sedimentary particles	Orford and Whalley [1983]
urine particles	Ranzato et al. [2007] Hans et al. [2010]
wear particles	Xu et al. [1998] Peng and Kirk [1999] ap Gwynn and Wilson [2001] Laghari [2003] Raadnui [2005] Stachowiak et al. [2008]

agglomerates is extracted, which we identified as not very useful for differentiating agglomerates of engineered nanoparticles from those composed of background particles (see Section 7.2.2).

Although it does not target engineered nanoparticles, we want to mention the bachelor's thesis by Oster [2010] at this point. The author also worked with the BAuA and used images captured with the same SEM as those used in this thesis. Rather than nanoparticles, the approach tries to find carbon nanotubes. Nevertheless, the image characteristics and the problem statement are very similar to ours. However, the problem is simplified and somewhat artificial in order to match the scope of a bachelor's thesis. Each image contains exactly one particle in its center and, apart from carbon nanotubes, only quartz dust and diesel soot may occur in the images. These have sufficiently different appearances to be distinguished using relatively simple numerical features.

The first three mentioned publications use images obtained by TEM. While it, just like SEM, uses electrons to depict the sample, its functional principle is different. With TEM, the electrons go through the sample and the image intensity reflects how many electrons are able to pass through it. While SEM images are comparable to photographs in that they depict the surface of the object, TEM is more like light microscopy or X-ray radiography. Here, the image represents a three-dimensional volume and all layers of the object are superimposed in it. TEM allows for very high resolutions as can be seen in the paper by Flores et al. [2003]. However, the sample has to be very thin in order that the electrons can pass it. In addition, the texture of nanoparticles imaged using TEM is quite different to that obtained by SEM, which becomes clear when looking at the images in the paper by Oleshko et al. [1996]. More information on TEM can be found in the book by Kohl and Reimer [2008].

In addition to TEM, the publications we examined use various other imaging techniques. Table 5.2 lists these and the corresponding publications. As in Table 5.1, we have left out prior publications by the same author groups covering earlier versions of their systems targeting the same particle types.

Several publications use a process called computer-controlled scanning electron microscopy (CCSEM). Here, a computer instead of an operator controls the microscope. It automatically detects particles on the substrate and takes an image of each of them. In addition, it performs an EDX analysis of each particle. However, this technique only works for particles with a diameter larger than $50\,\mathrm{nm}$ to $100\,\mathrm{nm}$ [Poelt et al., 2002]. Two of the three nanoparticle types we target are smaller than that. Therefore, using this technique is not an option for us.

The imaging methodologies used in the literature are diverse and each provides a set of data with characteristics typical for the corre-

Table 5.2: Overview of the used imaging techniques in the literature on automatic image-based particle analysis. The publications whose techniques are marked with 3D obtain and use three-dimensional data of the particles.

IMAGING TECHNIQUE	PUBLICATIONS
CCSEM + EDX	Germani and Buseck [1991]
	Kindratenko et al. [1994, 1996]
	Wienke et al. [1995]
	Hopke and Song [1997]
confocal microscopy (3D)	Peng and Kirk [1999]
light microscopy	Xu et al. [1998]
	Drolon et al. [2000]
	Laghari [2003]
	Greco and Maffezzoli [2004]
	Luo et al. [2004]
	Calderon De Anda et al. [2005]
	Rodriguez-Damian et al. [2006]
	Ranzato et al. [2007]
	Yu et al. [2007]
	Hans et al. [2010]
SEM	Langford et al. [1990]
	Thomas et al. [1995]
	Kindratenko and Van Espen [1996]
	Kindratenko et al. [1997]
	ap Gwynn and Wilson [2001]
	Oster [2010]
SEM (3D)	Stachowiak et al. [2008]
SEM + EDX	Genga et al. [2012]
TEM	Fisker et al. [2000]
	Flores et al. [2003]
TEM + EDX	Oleshko et al. [1996]

sponding technique. Often, these differ substantially for the different methodologies. Therefore, many of the applied algorithms are not useful for our problem.

The main differences between the problems covered in the literature examined by us and the challenge we face are the following:

- In many cases, the differences between the classes to distinguish are bigger in the related work. As seen in Section 2.2.4.2, our smaller engineered nanoparticles and diesel soot are very difficult to distinguish, even for humans.
- As the literature covers a wide range of particle types, their characteristics differ by a large amount compared to the nanoparticles which usually occur in agglomerates. The particles may for example be circular or have regular features.
- In 9 of the 33 examined publications, the authors have access to the chemical composition of the particles.
- As far as we know, none of the publications target particles of a wide size range. As mentioned before, the ratio of the areas occupied by the smallest and the largest ZnO particles in our images is 1:425 000.
- Several of the publications try to detect particles with a constant appearance. In our case, each nanoparticle agglomerate has a different shape.

5.2 SYSTEM ARCHITECTURE

Despite these major differences in the examined literature, we have found a typical system workflow common to almost all examined papers:

1. Segmentation of the image to separate the particles from the image background.
2. Computation of numerical features for every particle.
3. Optional: Reduction of the number of features (dimensionality reduction).
4. Classification of the particles based on the computed features.

A graphical representation of the workflow can be seen in Figure 5.1. The following paragraphs describe each step in detail:

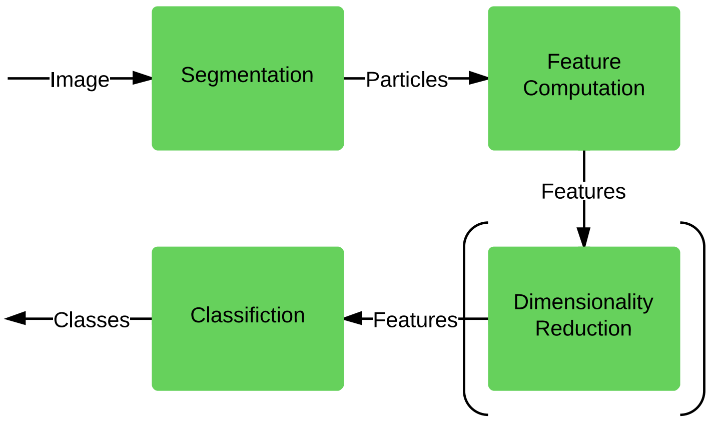


Figure 5.1: The typical workflow for the task of particle detection and classification.

SEGMENTATION This step consists of finding the particles by deciding which pixels of the original image belong to a particle and which are part of the background. Furthermore, as an image potentially shows more than one particle, the algorithm has to determine which pixels belongs to which particle. This is often done by simply grouping non-background pixels which form a connected component. As a result of the segmentation process, the location and shape of every particle in the original image is known.

FEATURE COMPUTATION In this step, every particle found in the segmentation process is described by a fixed set of features, which are expressed as numbers. The goal is that particles can be compared solely based on their feature values. To do this, features should capture important aspects of the particle such as its shape or texture. Particles which are similar in the image should also have similar feature values. To achieve that, features have to be carefully chosen to suit the requirements of specific applications. The whole set of feature values of a particle is also called its feature vector.

DIMENSIONALITY REDUCTION To reduce the complexity of the classification, another step called dimensionality reduction can be performed before the feature vector is passed to the classifier. Its goal is to reduce the number of features in order to make the classification less complex. For a more thorough analysis of the reasons to perform this step, see Section 8.1.2. There are two approaches to reach this goal: feature extraction and feature selection. In the literature, these terms are used inconsistently and interchangeably. However, we will use them according to Jain et al. [2000] as follows:

- Feature extraction combines or transforms existing features to create new ones.

- Feature selection chooses a subset of the original features to obtain a smaller feature vector.

The two approaches are often combined by first using feature extraction to create new features and then reducing their number by applying feature selection.

CLASSIFICATION In this last step, a so-called classifier assigns exactly one class to each particle. In our case, there are two classes:

- Engineered nanoparticles
- Background particles

The classifier bases its decision solely on the feature vector of each particle. To reliably predict the class of a particle, the classifier has to be trained using manually classified particles. This step uses techniques from the fields of pattern recognition and machine learning. For a brief overview of pattern recognition, see Jain et al. [2000].

As said before, almost all of the publications follow this workflow. However, there are a few exceptions, which we will briefly describe.

Langford et al. [1990] use manually extracted texture samples to classify pollen grains instead of relying on an automatic segmentation. Other publications do not use segmentation because each image only contains one centered particle or the exact shape is not of interest. Instead, a region of interest (ROI) is used. In our case, however, it is not feasible to manually mark the position of each particle as there are up to a thousand particles in a single image.

Podsiadlo and Stachowiak [2005] do not compute a feature vector to classify the particles. Instead, they calculate the dissimilarities of the wear particle to be classified with all particles of the training set. The classification is then performed based on these dissimilarities. This poses a problem as the approach cannot easily be combined with features not based on dissimilarity. Another disadvantage is that the computational complexity of the classification grows as more particles are added to the training set.

Some of the publications do not cover automatic classification [Orford and Whalley, 1983; Germani and Buseck, 1991; Kindratenko et al., 1994, 1996; Oleshko et al., 1996; Fisker et al., 2000; ap Gwynn and Wilson, 2001; Greco and Maffezzoli, 2004; Genga et al., 2012]. Instead, features are computed to characterize different types of particles. However, the first two steps of the workflow are the same as in the classification approach. Therefore, we have included these papers, as well.

We have decided to adopt the presented workflow because the general set-up of our problem is the same as in many of the analyzed papers. In addition, because of the lack of large training sets and the complex nature of nanoparticle agglomerates, the use of numerical features to describe the particles is a sensible choice. We will explain this in more detail in Section 8.2.1.

As this workflow is linear and each step only depends on the previous, we will organize the remaining thesis based on it. Chapter 6 will present our segmentation approach. In Chapter 7, we will describe the features we chose for our system. Chapter 8 will address dimensionality reduction and classification. Finally, Chapter 9 will present an additional feature of our system, which enhances the usability and operates in parallel to this workflow. Each of these chapters will additionally present related work on the corresponding topic and will feature an evaluation of the applied algorithms and approaches.

In order to illustrate each individual step of the workflow in a practical way, we will use a single Ag agglomerate on the left in Figure 1.2 on Page 5 and present the intermediate results for this agglomerate in the corresponding passages of the text.

In this chapter, we have presented literature on automatic image-based particle analysis and have derived a workflow for our system. The next chapter will address the first step of this workflow, the segmentation of SEM images containing engineered nanoparticles.

6 SEGMENTATION

This chapter will address the segmentation of nanoparticle images. Section 6.1 will give a detailed description of its goals. In Section 6.2, we will summarize the algorithms used in the literature on automatic image-based particle analysis. Section 6.3 will describe the methods used by our system and Section 6.4 will evaluate them.

6.1 GOALS

As the input of our system, we have SEM images. For the purpose of this thesis, we define an image f as a function

$$f : P \to G. \tag{6.1}$$

Here, P is the two-dimensional pixel space, defined as a set

$$P := \mathbb{N}_{<n_x} \times \mathbb{N}_{<n_y}, \tag{6.2}$$

where $n_x \in \mathbb{N}_{>0} := \{1, 2, 3, \dots\}$ and $n_y \in \mathbb{N}_{>0}$ are the width and height of the image, respectively, and $\mathbb{N}_{<n_x} := \{0, \dots, n_x - 1\}$. The set G is, in the case of grayscale images, a subset of the real numbers:

$$G \subseteq \mathbb{R}. \tag{6.3}$$

We will only address grayscale images in this thesis. An element $p \in P$ is called pixel and has an intensity $f(p) \in G$, also called gray level, grayscale value or pixel value. We will use the convention that the pixel $(0, 0)$ is in the top-left corner of the image. The images used for this thesis have a width of $n_x = 4000$ and a height of $n_y = 3200$. Their pixels can take intensities from 0 to 255, so that $G = \{0, \dots, 255\}$.

Simply put, the goal of the segmentation task is to find the shape and position of each particle in the image. As mentioned before, the term *particle* can denote a single nanoparticle, also called primary particle, but it can also refer to an agglomerate of many particles. When looking at Figure 6.1, it becomes clear that it is impossible to locate every nanoparticle in an agglomerate. The depicted agglomerate is made up of thousands of Ag nanoparticles with an average diameter of 75 nm. Due to the nature of SEM, contours of the primary particles vanish. Instead, neighboring nanoparticles seem to be fused together as if the agglomerate was a single solid object.

Another reason why it is impossible to locate every primary particle is that a majority of them is masked by other nanoparticles. Because the agglomerate is a three-dimensional object, it is impossible

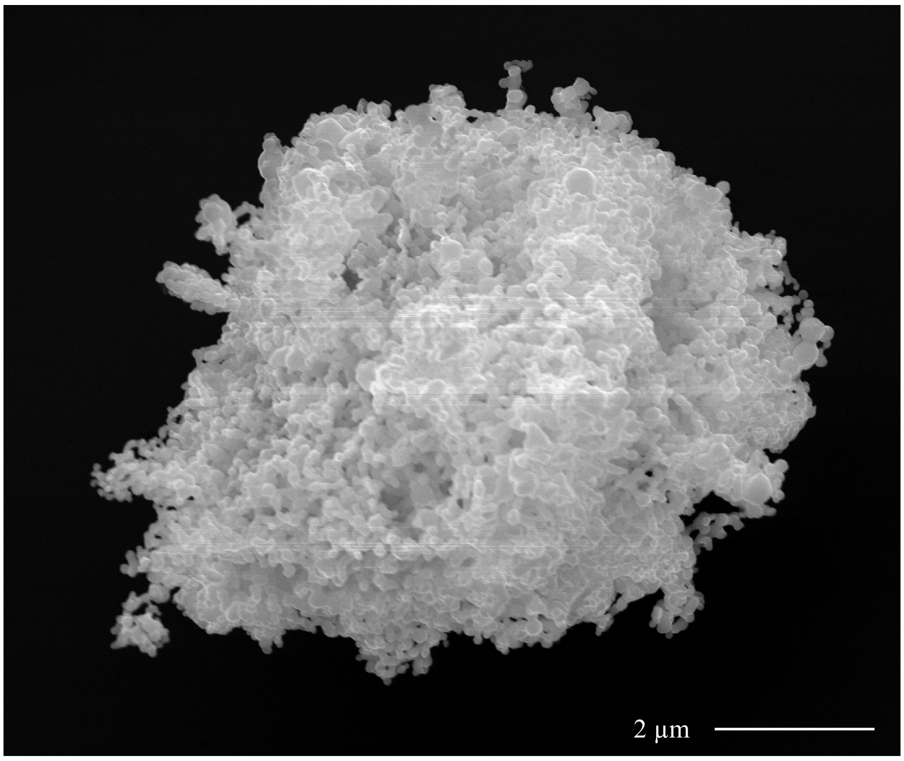


Figure 6.1: An image detail of an agglomerate of Ag nanoparticles with an average diameter of 75 nm.

to take an image that shows all primary particles on every side of the agglomerate. Equally impossible is an image that shows the primary particles inside the agglomerate. Therefore, we have chosen to treat agglomerates as a whole instead of trying to decompose them into their components. More specifically, two particles connected by a chain of other particles in the image are regarded as belonging to one particle by our system. To avoid ambiguities, we will use the term *connected particle group (CPG)* to refer to a set of connected particles in an image. A single solid particle such as a nanoparticle will also be called CPG if it is not connected to any other particles.

One conceivable disadvantage of the approach to treat agglomerates as a whole would be that CPGs made up of multiple particle types could not be separated. However, according to experts from the BAuA, a case where engineered nanoparticles and background particles form an agglomerate is very rare. In light of this and the fact that it simplifies the system workflow to a great degree, we have chosen to classify each CPG as consisting either of engineered nanoparticles or of background particles.

Optimally, our system should be able to find all CPGs consisting of engineered nanoparticles as well as all CPGs of background particles. In our experience, however, background particles can have highly varying sizes, shapes and textures. Especially, very small and dark particles might be hard to differentiate from noise artifacts of the highly noisy image background. Engineered nanoparticles, on the other hand, while still relatively variable in appearance, are quite similar in our experience. We have therefore decided to concentrate on the detection of the latter. We will develop and tune our method so that it optimally finds engineered nanoparticles in order to allow for a precise concentration estimation. An overlooked background particle is irrelevant with respect to a correct concentration estimate.

To phrase the goal of the segmentation step in a more formal way, we want to find $n_C \in \mathbb{N} := \{0, 1, 2, \ldots\}$ different CPGs $C_1, \ldots, C_{n_C} \subseteq P$, which are not empty and mutually disjoint:

$$|C_i| > 0, \quad \forall\, i \in \{1, \ldots, n_C\}, \tag{6.4}$$

and

$$C_i \cap C_j = \emptyset, \quad \forall\, i, j \in \{1, \ldots, n_C\}, i \neq j. \tag{6.5}$$

To define what it means for a CPG to be connected, we need to define the 4-connected neighborhood $N_4 : P \to \mathcal{P}(P)$ of a pixel $p \in P$:

$$N_4(p) := \left\{p' \in P \,\middle|\, \left(\left|p_x - p'_x\right| + \left|p_y - p'_y\right|\right) = 1\right\}, \quad \forall\, p \in P. \tag{6.6}$$

Here $\mathcal{P}(P) := \{S \,|\, S \subseteq P\}$ is the power set of P, which is defined as the set of all subsets of P. Now, we can recursively define the notion of connectedness, where $C \subseteq P$ and $p \in P$:

1. C is connected if $|C| = 1$.
2. $C \cup \{p\}$ is connected if C is connected and $\exists\, p' \in C : p \in N_4(p')$.

Each CPG must be connected. In addition, we said that agglomerates should be treated as a whole. This means that two CPGs should not be in the neighborhood of each other:

$$C_i \cap \bigcup_{p \in C_j} N_4(p) = \emptyset, \quad \forall\, i, j \in \{1, \ldots, n_C\}, i \neq j. \tag{6.7}$$

In addition, let $\mathcal{C} \subseteq \mathcal{P}(P)$ be the set of all possible CPGs $C \subseteq P$:

$$\mathcal{C} := \{C \subseteq P \,|\, C \text{ is connected}\}. \tag{6.8}$$

Now that we have defined the goal of the segmentation step, both informally and mathematically, we want to look at how the literature on automatic image-based particle analysis deals with the problem. We will do that in the following section.

6.2 RELATED WORK

In about half of the publications on automatic image-based particle analysis, the subject of segmentation is not covered at all. The reason may be that each image contains only one centered particle or that the segmentation is done manually. In those papers which address the matter, we have found three general particle segmentation approaches:

- Finding round structures
- Edge-based segmentation
- Thresholding

In the following sections, these will be explained further.

6.2.1 *Finding Round Structures*

The approaches presented in this section assume that the particles to be found are circular or elliptical. The used methods try to use this feature to directly locate the particles.

Fisker et al. [2000] use an elliptic deformable template model to find single elliptical nanoparticles and estimate their size distribution directly. The authors state that their method "gives reliable estimates of the size distribution when the particles are well separated and the particle shape can be well approximated with ellipses." In our case, this is neither the case for the primary particles (most of them are occluded by others) nor for the agglomerates (which have irregular shapes and can even have holes in them). Therefore, the proposed solution is unsuitable for our problem.

Rodriguez-Damian et al. [2006] propose a similar solution. They use a circle Hough transform to detect circular pollen grains instead of using classical segmentation. As with the previously mentioned solution, this can only be used if the particles are round and well separated.

Ranzato et al. [2007] filter their images at two scales with a difference of Gaussians and look for extrema to find biological particles in human urine and airborne pollen grains. The algorithm is designed for circular particles and requires their approximate size to be known beforehand. As mentioned before, the sizes of agglomerates in our images can vary to a great degree (see Section 2.2.3.1). Therefore, we cannot use this solution either.

6.2.2 *Edge-based Segmentation*

Only one method falls into this category. It first tries to find the contours of the particles in order to locate them. The method is proposed and used by Calderon De Anda et al. [2005] to find organic crystals. It is also used by Yu et al. [2007].

The approach consists of the following steps:

1. Edge detection
2. Morphological closing
3. Region filling

4. Morphological opening

5. Removal of artifacts

The first step is done using the Canny edge detector and is performed twice using two different Gaussian kernel widths. Their results are combined to find the edges of the crystals. The closing step ensures that there are no gaps in the contours. As a next step, the areas surrounded by closed edges are filled. The opening step removes the remaining edges which have not been filled. The last step consists of removing all regions smaller than a minimum number of pixels to ensure the removal of remaining artifacts.

The method is targeted at images where the background is irregular. It is unclear if the method is usable if the particles have holes. A potential weakness of the approach is that a particle is completely disregarded if its contour is not detected at all points.

6.2.3 *Thresholding*

Thresholding is by far the most popular segmentation approach in the publications we have examined. The main idea is that the particles, depending on the imaging technique, are brighter or darker than the image background. Therefore, every pixel that has an intensity above (below) a given threshold is regarded as belonging to a particle and all other pixels as part of the image background. The main problem is the determination of the optimal threshold. In addition, there are some modifications to the base algorithm in the literature. An overview of the used methods is given in this section.

Kindratenko et al. [1994] and Stachowiak et al. [2008] use thresholding but do not give any details while ap Gwynn and Wilson [2001] manually determine the threshold per image.

Germani and Buseck [1991] propose an iterative approach to find the optimal threshold for the detection of atmospheric particles in SEM. It consist of the following steps:

1. Calculate the mean and standard deviation of the image intensity.

2. Reject every pixel whose intensity differs more than two times the standard deviation from the mean.

3. Repeat Steps 1 and 2 until the change in the relative standard deviation is less than 0.2. Note that for the computation of the mean and standard deviation, only pixels which were not rejected are used.

4. The threshold is set to the mean plus three times the standard deviation, calculated without the rejected pixels.

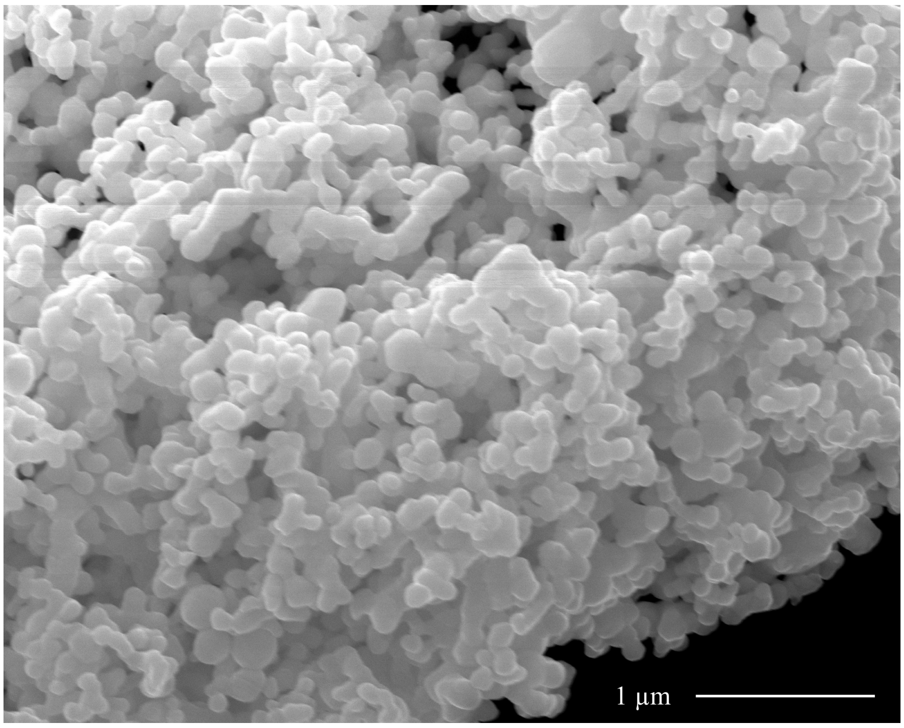


Figure 6.2: An SEM image ($1\,\mathtt{pixel} = 1.27\,\mathrm{nm}$) of Ag nanoparticles with an average diameter of $75\,\mathrm{nm}$. Almost the entire image is occupied by a single Ag agglomerate.

For the algorithm, it is assumed that the particles have a higher intensity than the background. The authors say that a "particle loading (projected particle area per SEM field-of-view area) of < 0.2 is necessary for effective automatic thresholding." Some of our images such as the one in Figure 6.2 have a much higher particle loading than that. Therefore, the algorithm is unsuitable for our problem.

Oleshko et al. [1996] use a threshold finding method by Brink [1989]. The threshold is chosen such that the coefficient of correlation between the original image and the binary image resulting from the thresholding is maximized. The method works well for the TEM images used in the paper because they have only very little contrast inside of the particles while the difference between background and particles is large. Our images, however, have in some cases a high contrast inside the particles compared to the particle-background difference. An example of this can be seen in Figure 6.3.

Oster [2010] uses a method by Zack et al. [1977] to find the threshold for the segmentation of nanomaterials in SEM images. It chooses the threshold based on the image histogram using geometrical means. As the images used by Oster [2010] are very similar to ours, we have found that the algorithm works well for the images examined in this work. An explanation of the method will be given in Section 6.3.3.

In addition to the classical thresholding approach, there are some variations known as dynamic or adaptive thresholding. Rather than

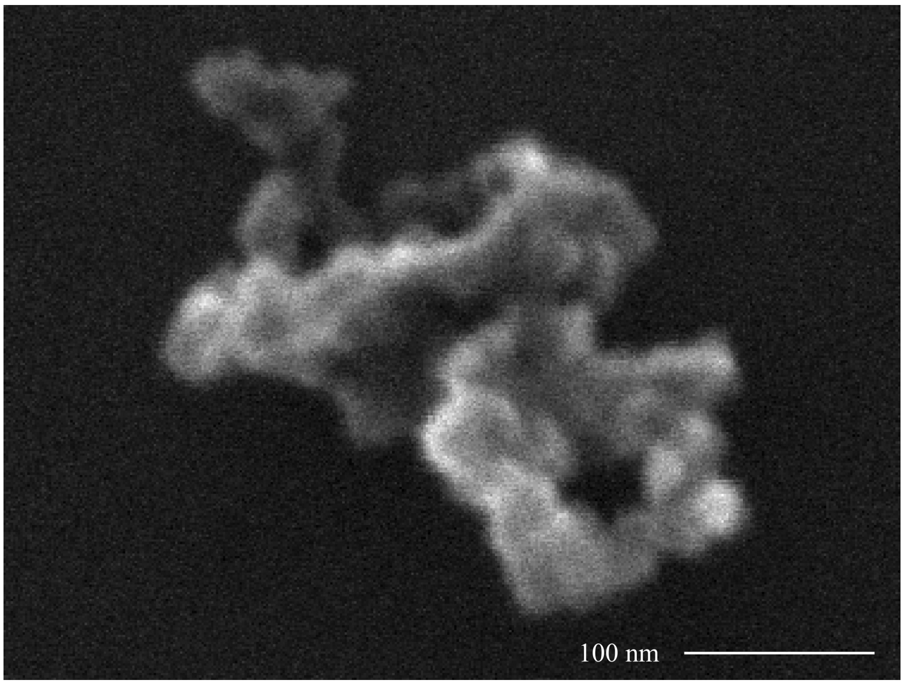


Figure 6.3: An SEM image detail (1 pixel = 1.27 nm) of engineered TiO_2 nanoparticles.

a constant threshold, these methods use a function which can give a different threshold for every pixel of the image. The same approach can also be interpreted as follows: A second image, such as an approximation of the background, is subtracted from the original one. After that, classical thresholding is applied to the resulting image.

One such method is used by Flores et al. [2003]. They simply use a smoothed version of the original image as the threshold function. For large particles, the smoothing has to be strong, so that background pixels are taken into account. However, for small and faint particles, the smoothing must not be too strong so that neighboring particles with high intensities do not have too much influence on the smoothed image. Therefore, the approximate size of the particles has to be known beforehand. For the images targeted in this work, where large bright and small faint particles can occur, the method is unsuitable.

Hans et al. [2010] use another variation to handle background intensity variations. A morphological opening of the original image with a round structuring element is performed. Its size must be larger than the particle size so that the resulting image becomes an approximation of the image background. The result is then subtracted from the original and a thresholding as proposed by Otsu [1979] is performed. It works by minimizing the weighted sum of the variances of the pixel intensities below and above the threshold. The weights are determined by the number of pixels in each class.

The requirements for the method used by Hans et al. [2010] are that the structuring element is larger than the largest particle. In our case, a large part of the image can be occupied by an agglomerate. Therefore, the structuring element would have to be only slightly smaller than the whole image. This would have no advantage as the method converges to classical thresholding as the structuring element gets larger. In addition, the computational complexity grows.

6.3 METHOD

In this section, we will describe the method we use to perform a segmentation of each image and to find the CPGs in it.

We have chosen to use thresholding to segment the images. This has the following reasons:

- As is clearly visible in Figure 6.1, the engineered nanoparticles always have a higher intensity than the image background.
- The background usually has a very homogeneous intensity.
- The segmentation method needs to be able to detect holes in an agglomerate. An example of a hole can be seen in Figure 6.1. In the lower right part of the agglomerate, a portion of the image background is fully surrounded by nanoparticles. Most segmentation methods apart from thresholding are not able to handle such holes.
- Oster [2010] obtains good results using a thresholding approach on images comparable to ours. In fact, the images have been taken using the same microscope with similar parameters.

As the signal-to-noise ratio of SEM images is relatively low, applying thresholding to the raw images leads to small holes in the segmentation of particles and background. This effect can be seen when thresholding the image detail in Figure 6.4. The result is visible in Figure 6.5. Many pixels of the background are regarded as foreground pixels by the thresholding. In addition, the segmentation of the agglomerate contains small holes and its contour is very rough. Oster [2010] successfully solves this problem by removing image noise using a filter prior to thresholding. Therefore, we have chosen to use a noise removal method to avoid small holes in the segmentation, as well.

Due to the different contrast and brightness settings used, each image has individual noise parameters. In order to optimally remove the noise in every image, the parameters of the noise removal method have to be adjusted for each image individually. We therefore use an algorithm to estimate the noise parameters of each image.

As a result of the thresholding, we obtain a binary image with foreground and background pixels. In accordance with the decision

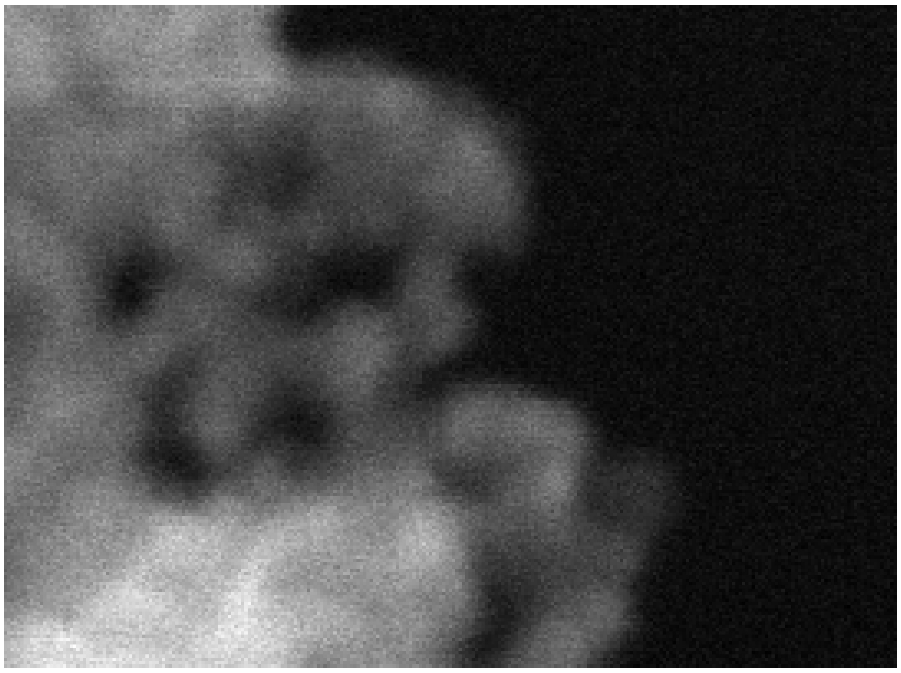


Figure 6.4: An image detail (1 pixel = 1.27 nm) of an agglomerate of TiO_2 nanoparticles with an average diameter of 25 nm.

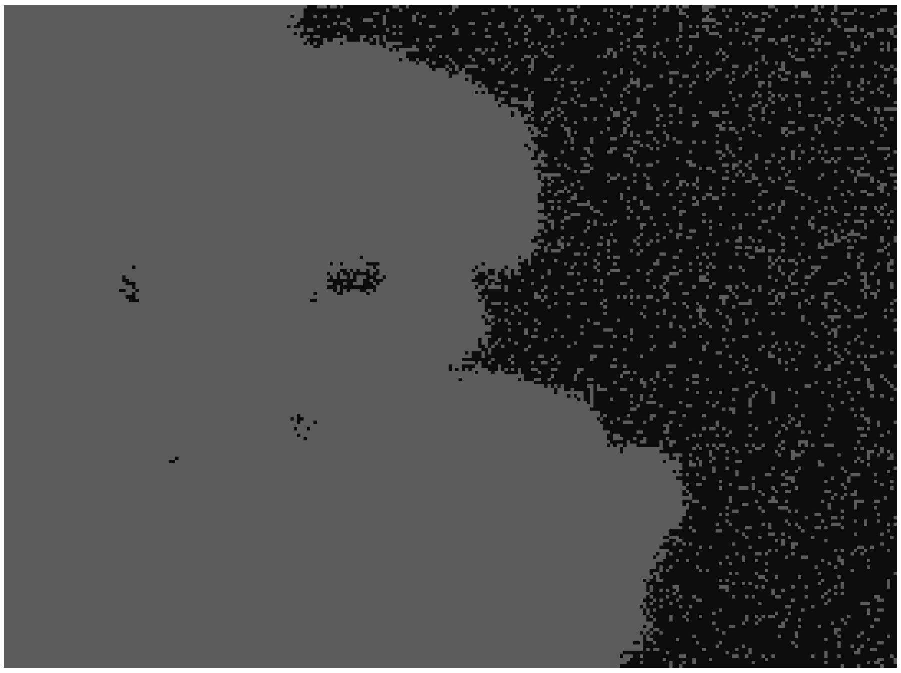


Figure 6.5: A binary image resulting from thresholding the image in Figure 6.4 with a threshold of 20.5.

to treat CPGs as a whole as outlined in Section 6.1, each connected component of foreground pixels is treated as one CPG.

All in all, the workflow of the segmentation of every image is as follows:

1. Noise estimation
2. Noise removal
3. Threshold determination
4. CPG search in thresholded image

These steps are explained in the following sections.

6.3.1 *Noise Estimation*

To optimally remove the noise in an image, the strength and distribution of the noise have to be known. Since these are unknown, they have to be estimated. Typically, it is assumed that the noise is Gaussian distributed and uniform for each pixel [Olsen, 1993; Rank et al., 1999]. However, this assumption does not hold for SEM images.

The electron generation and detection in an SEM is a Poisson process [Joy, 2008]. Therefore, the number of detected electrons is Poisson distributed. This means that the noise is stronger in image parts with higher intensities than in dark parts of the image. Noise having this property is called shot noise.

As most noise estimation methods assume white noise [Olsen, 1993; Rank et al., 1999], they are inappropriate to be used on SEM images. However, we found four existing approaches targeting signal-dependent noise estimation, which we will briefly describe.

Healey and Kondepudy [1994], Ito et al. [2008] and Liu et al. [2008] propose three different noise estimation methods targeted at CCD camera images. All of these model signal-dependent noise. However, CCD images have characteristics different from SEM images. For example, the signal pipelines of CCD cameras contain steps such as gamma correction. Furthermore, the method by Healey and Kondepudy [1994] requires estimating camera parameters using multiple pairs of images of homogeneous surfaces. Using their approach, it is not possible to estimate parameters for a single image. The algorithms by Ito et al. [2008] and Liu et al. [2008] both need a so-called camera response function, which models the signal pipeline processes and is individual for each camera.

Foi et al. [2008] propose a method to estimate the noise parameters of the raw data coming from an imaging sensor. It models Poissonian and Gaussian Noise components.

While all of the mentioned solutions model signal amplification in some form, none of their models include an additive term which is constant for all pixels of the image. In other words, all models assume that an intensity of 0 corresponds to the case that no photons (or in our case electrons) have been detected. This is not the case for SEM because the brightness setting adds a constant term to the intensity of every pixel of the image. Therefore, we will propose a novel noise model for SEM images in the next section. In Section 6.3.1.2, we will present an algorithm based on this model to estimate noise parameters from a single image, which is specifically targeted at SEM images.

6.3.1.1 *Scanning Electron Microscopy Noise Model*

As explained in Section 4.2.1, an SEM shoots electrons at the specimen and detects how many secondary electrons escape the specimen. If we assume that all microscope parameters including the position of the specimen are fixed and that it takes an infinite number of images, we can look at the probability distribution of the number of detected electrons at each point of the specimen. In the following sections, we will look at that distribution for a fixed point on the specimen. This point corresponds to the same pixel in each image because the specimen position and the microscope parameters are not changed. We can interpret the number of electrons detected at that point as a random variable $\tilde{c} : \Omega \to \mathbb{N}$. It reflects the probability of detecting a certain number of electrons if we look at a random image from the infinite set. We use the convention that the set Ω is a sample space as defined in probability theory underlying all random variables in this thesis. It is only used to define random variables and will not be defined itself.

The random variable $\tilde{c}$ follows a Poisson distribution [Joy, 2008]. The probability of detecting exactly $k \in \mathbb{N}$ electrons is

$$\mathbb{P}(\tilde{c} = k) = \frac{\mu^k}{k!} e^{-\mu}, \quad \forall k \in \mathbb{N}. \tag{6.9}$$

Here, $\mu \in \mathbb{R}_{>0}$ is the average number of electrons detected at the current point over multiple images of the same specimen at the same position. This distribution occurs in scenarios where the number of independent events is counted in a fixed time interval. Such scenarios are called Poisson processes.

The expected value of $\tilde{c}$ is equal to μ:

$$\mathbb{E}(\tilde{c}) = \mu. \tag{6.10}$$

In fact, the variance of $\tilde{c}$ is also equal to μ:

$$\mathrm{Var}(\tilde{c}) = \mu. \tag{6.11}$$

As explained in Section 4.2.1, the microscope operator is able to adjust the brightness and contrast settings in order to map the range of electron counts to the intensity range. As this mapping is linear, we can define the intensity of the pixel corresponding to the point on the specimen we have fixed as a random variable $\tilde{g} : \Omega \to \mathbb{R}$ as follows:

$$\tilde{g} \coloneqq a\tilde{c} + b, \tag{6.12}$$

where $a \in \mathbb{R}_{>0}$ corresponds to the contrast setting and $b \in \mathbb{R}_{<0}$ corresponds to the brightness setting. The values of a and b are constant for each pixel of an image. However, their values are generally different for separate images. Therefore, if we want to determine the noise parameters, we have to estimate the values of a and b from a single image.

Note, that we define $\tilde{g}$ as a real-valued random variable. In reality, the intensities of our images can only take on integer values. However, we assume that the effects of quantization are considerably smaller than the inherent noise because of the low signal-to-noise ratio of SEM (see Section 4.2.2.1). Modeling the quantization would make the maths considerably more complex.

With this definition, we can compute the expected value of the intensity $\tilde{g}$:

$$\begin{aligned} \mathbb{E}(\tilde{g}) &\underset{(6.12)}{=} \mathbb{E}(a\tilde{c} + b) \\ &= a\,\mathbb{E}(\tilde{c}) + b \\ &\underset{(6.10)}{=} a\mu + b. \end{aligned} \tag{6.13}$$

In a perfect image, $\tilde{g} = \mathbb{E}(\tilde{g})$ would hold for every pixel. Given the values of a and b, which are determined by the microscope operator, $\mathbb{E}(\tilde{g})$ only depends on μ, which is a direct property of the specimen at the current point.

In a given image, $\tilde{g} = \mathbb{E}(\tilde{g})$ will generally not hold. We can interpret this noise as a random variable $\tilde{o} : \Omega \to \mathbb{R}$, which is added to the optimal intensity $\mathbb{E}(\tilde{g})$ to yield the observed intensity $\tilde{g}$:

$$\tilde{o} \coloneqq \tilde{g} - \mathbb{E}(\tilde{g}). \tag{6.14}$$

The expected value of the noise is 0:

$$\begin{aligned} \mathbb{E}(\tilde{o}) &\underset{(6.14)}{=} \mathbb{E}(\tilde{g} - \mathbb{E}(\tilde{g})) \\ &= \mathbb{E}(\tilde{g}) - \mathbb{E}(\tilde{g}) \\ &= 0. \end{aligned} \tag{6.15}$$

This is so by definition because, in average, the pixel's intensity is $\mathbb{E}(\tilde{g})$.

To determine the strength of the noise, we have to compute its variance $\mathrm{Var}(\tilde{o})$:

$$\begin{aligned} \mathrm{Var}(\tilde{o}) &\underset{(6.14)}{=} \mathrm{Var}(\tilde{g} - \mathbb{E}(\tilde{g})) \\ &= \mathrm{Var}(\tilde{g}). \end{aligned} \tag{6.16}$$

Moreover, we can calculate:

$$\begin{aligned} \operatorname{Var}(\tilde{g}) &\underset{(6.12)}{=} \operatorname{Var}(a\tilde{c}+b) \\ &= \operatorname{Var}(a\tilde{c}) \\ &= a^2 \operatorname{Var}(\tilde{c}) \\ &\underset{(6.11)}{=} a^2\mu. \end{aligned} \tag{6.17}$$

Further, we can express it in terms of the expected intensity $\mathbb{E}(\tilde{g})$:

$$\begin{aligned} \operatorname{Var}(\tilde{g}) &\underset{(6.17)}{=} a^2\mu \\ &= a(a\mu+b-b) \\ &= a(a\mu+b)-ab \\ &\underset{(6.13)}{=} a\,\mathbb{E}(\tilde{g})-ab. \end{aligned} \tag{6.18}$$

Using Equations (6.16) and (6.18), we get

$$\operatorname{Var}(\tilde{o}) = a\,\mathbb{E}(\tilde{g}) - ab. \tag{6.19}$$

This result shows that, contrary to the common assumption of Gaussian noise, SEM images have signal-dependent noise. The noise variance is a linear function of the average pixel intensity. Areas with a higher intensity have a higher noise variance than darker image areas. But this also means that in order to calculate the noise variance for a given image and intensity, we only need to determine the contrast setting a and the brightness setting b. As we do not know them, they have to be estimated.

In order to do this, let us assume that we have two pixels with the same probability distribution, which means that they have the same average electron count μ. In addition, we assume that both pixels have been recorded using the same values of a and b. Let the random variables $\tilde{c}_1 : \Omega \to \mathbb{N}$ and $\tilde{c}_2 : \Omega \to \mathbb{N}$ be their electron counts and $\tilde{g}_1 : \Omega \to \mathbb{R}$ and $\tilde{g}_2 : \Omega \to \mathbb{R}$ their intensities. These are independent and have the same distribution as $\tilde{c}$ and $\tilde{g}$, respectively. Let $\tilde{s} : \Omega \to \mathbb{R}$ be their sum:

$$\tilde{s} := \tilde{g}_1 + \tilde{g}_2, \tag{6.20}$$

and $\tilde{d} : \Omega \to \mathbb{R}$ their difference:

$$\tilde{d} := \tilde{g}_1 - \tilde{g}_2. \tag{6.21}$$

With this, we can compute the expected squared difference given the sum $s \in a\mathbb{N} + 2b := \{ai + 2b \in \mathbb{R} \,|\, i \in \mathbb{N}\}$, using the definition $c := \frac{s-2b}{a} \in \mathbb{N}$ for brevity:

$$\begin{aligned}
&\mathbb{E}\big(\tilde{d}^2 \,\big|\, \tilde{s} = s\big) \\
\underset{(6.21)}{=}\; &\mathbb{E}\big((\tilde{g}_1 - \tilde{g}_2)^2 \,\big|\, \tilde{s} = s\big) \\
\underset{(6.20)}{=}\; &\mathbb{E}\big((\tilde{g}_1 - \tilde{g}_2)^2 \,\big|\, \tilde{g}_1 + \tilde{g}_2 = s\big) \\
\underset{(6.12)}{=}\; &\mathbb{E}\big((a\tilde{c}_1 + b - a\tilde{c}_2 - b)^2 \,\big|\, a\tilde{c}_1 + b + a\tilde{c}_2 + b = s\big) \\
=\; &\mathbb{E}\Big((a(\tilde{c}_1 - \tilde{c}_2))^2 \,\Big|\, a(\tilde{c}_1 + \tilde{c}_2) + 2b = s\Big) \\
=\; &\mathbb{E}\left(a^2(\tilde{c}_1 - \tilde{c}_2)^2 \,\middle|\, \tilde{c}_1 + \tilde{c}_2 = \frac{s-2b}{a}\right) \\
=\; &a^2\,\mathbb{E}\big((\tilde{c}_1 - \tilde{c}_2)^2 \,\big|\, \tilde{c}_1 + \tilde{c}_2 = c\big) \\
\underset{(A.1)}{=}\; &\frac{a^2}{2^c}\sum_{i=0}^{c}(i - (c-i))^2\binom{c}{i} \\
=\; &\frac{a^2}{2^c}\sum_{i=0}^{c}(2i - c)^2\binom{c}{i} \\
=\; &a^2 c \\
=\; &a^2\frac{s-2b}{a} \\
=\; &as - 2ab, \quad \forall\, s \in a\mathbb{N} + 2b.
\end{aligned} \tag{6.22}$$

Here, we have used Equation (A.1), which can be found in Appendix A because it is not within the scope of this section.

This means that the expected squared difference of two independent and identically distributed pixel intensities is linearly dependent on their sum. This can be used to estimate the values of a and b. To do this, we could take one pixel pair for each possible sum s and use linear regression to estimate the slope a and y-intercept $2ab$. To see how good such estimates can be, let us calculate the variance of the squared difference given the sum $s \in a\mathbb{N} + 2b$:

$$\begin{aligned}
&\operatorname{Var}\bigl(\tilde{d}^2 \bigm| \tilde{s} = s\bigr) \\
&= \mathbb{E}\Bigl(\bigl(\tilde{d}^2 - \mathbb{E}(\tilde{d}^2 \mid \tilde{s} = s)\bigr)^2 \Bigm| \tilde{s} = s\Bigr) \\
&\underset{(6.22)}{=} \mathbb{E}\Bigl(\bigl(\tilde{d}^2 - a^2 c\bigr)^2 \Bigm| \tilde{s} = s\Bigr) \\
&\underset{(6.22)}{=} \mathbb{E}\Bigl(\bigl(\tilde{d}^2 - a^2 c\bigr)^2 \Bigm| \tilde{c}_1 + \tilde{c}_2 = c\Bigr) \\
&\underset{(6.22)}{=} \mathbb{E}\Bigl(\bigl(a^2(\tilde{c}_1 - \tilde{c}_2)^2 - a^2 c\bigr)^2 \Bigm| \tilde{c}_1 + \tilde{c}_2 = c\Bigr) \\
&= a^4\, \mathbb{E}\Bigl(\bigl((\tilde{c}_1 - \tilde{c}_2)^2 - c\bigr)^2 \Bigm| \tilde{c}_1 + \tilde{c}_2 = c\Bigr) \\
&\underset{(A.1)}{=} \frac{a^4}{2^c} \sum_{i=0}^{c} \bigl((i - (c-i))^2 - c\bigr)^2 \binom{c}{i} \\
&= 2a^4 c(c-1) \\
&= 2a^2 c\bigl(a^2 c - a^2\bigr) \\
&\underset{(6.22)}{=} 2\,\mathbb{E}\bigl(\tilde{d}^2 \bigm| \tilde{s} = s\bigr)\bigl(\mathbb{E}(\tilde{d}^2 \mid \tilde{s} = s) - a^2\bigr), \quad \forall s \in a\mathbb{N} + 2b.
\end{aligned} \tag{6.23}$$

We want the variance of an estimate to be considerably smaller than its expected value. This is not the case for $\mathbb{E}(\tilde{d}^2 \mid \tilde{s} = s)$ and $\operatorname{Var}(\tilde{d}^2 \mid \tilde{s} = s)$.

To solve this problem, let use take $n_{pp} \in \mathbb{N}_{>0}$ pixel pairs that have been taken using the same values of a and b instead of only one. The two pixels of each pair share the same average electron count μ. However, the pixels of one pair may have a different μ from that of another pair. In other words, the two pixels of a pair share the same distribution while two pixels from different pairs may not. In addition, we assume that the random variables representing the electron counts of all involved pixels are mutually independent.

For the pixel pair with the index $i \in \{1, \dots, n_{pp}\}$, let the random variables $\tilde{s}_i : \Omega \to \mathbb{R}$ and $\tilde{d}_i : \Omega \to \mathbb{R}$ be the sum and difference of their intensities, respectively. Then, we can define average squared intensity difference as a random variable $\tilde{v} : \Omega \to \mathbb{R}$:

$$\tilde{v} := \frac{1}{n_{pp}} \sum_{i=1}^{n_{pp}} \tilde{d}_i^2. \tag{6.24}$$

If we assume that all pixel pairs have the same sum, we can show that the expected value of $\tilde{v}$ given the sum is equal to $\mathbb{E}(\tilde{d}^2 \mid \tilde{s} = s)$:

$$\begin{aligned}
&\mathbb{E}\left(\tilde{v} \,\middle|\, \tilde{s}_1 = \cdots = \tilde{s}_{n_{pp}} = s\right) \\
\underset{(6.24)}{=}\; &\mathbb{E}\left(\frac{1}{n_{pp}} \sum_{i=1}^{n_{pp}} \tilde{d}_i^2 \,\middle|\, \tilde{s}_1 = \cdots = \tilde{s}_{n_{pp}} = s\right) \\
=\; &\frac{1}{n_{pp}} \sum_{i=1}^{n_{pp}} \mathbb{E}\left(\tilde{d}_i^2 \,\middle|\, \tilde{s}_1 = \cdots = \tilde{s}_{n_{pp}} = s\right) \\
=\; &\frac{1}{n_{pp}} \sum_{i=1}^{n_{pp}} \mathbb{E}\left(\tilde{d}_i^2 \mid \tilde{s}_i = s\right) \\
\underset{(6.22)}{=}\; &\frac{1}{n_{pp}} \sum_{i=1}^{n_{pp}} a^2 c \\
=\; &a^2 c \\
=\; &as - 2ab, \quad \forall\, s \in a\mathbb{N} + 2b.
\end{aligned} \tag{6.25}$$

In the third step, we have used that the random variables representing the electron counts of different pixels are independent.

With this result, we can group pixel pairs by their sum s and use linear regression on their average squared difference to estimate the values of a and b. We can also show that the variance of $\tilde{v}$ given the sum is n_{pp} times smaller than $\mathrm{Var}(\tilde{d}^2 \mid \tilde{s} = s)$:

$$\begin{aligned}
&\mathrm{Var}\left(\tilde{v} \,\middle|\, \tilde{s}_1 = \cdots = \tilde{s}_{n_{pp}} = s\right) \\
\underset{(6.24)}{=}\; &\mathrm{Var}\left(\frac{1}{n_{pp}} \sum_{i=1}^{n_{pp}} \tilde{d}_i^2 \,\middle|\, \tilde{s}_1 = \cdots = \tilde{s}_{n_{pp}} = s\right) \\
=\; &\frac{1}{{n_{pp}}^2} \mathrm{Var}\left(\sum_{i=1}^{n_{pp}} \tilde{d}_i^2 \,\middle|\, \tilde{s}_1 = \cdots = \tilde{s}_{n_{pp}} = s\right) \\
=\; &\frac{1}{{n_{pp}}^2} \sum_{i=1}^{n_{pp}} \mathrm{Var}\left(\tilde{d}_i^2 \,\middle|\, \tilde{s}_1 = \cdots = \tilde{s}_{n_{pp}} = s\right) \\
=\; &\frac{1}{{n_{pp}}^2} \sum_{i=1}^{n_{pp}} \mathrm{Var}\left(\tilde{d}_i^2 \mid \tilde{s}_i = s\right) \\
\underset{(6.23)}{=}\; &\frac{1}{{n_{pp}}^2} \sum_{i=1}^{n_{pp}} 2a^4 c(c-1) \\
=\; &\frac{2}{n_{pp}} a^4 c(c-1) \\
=\; &\frac{2}{n_{pp}} a^2 (s-2b)((s-2b)-a), \quad \forall\, s \in a\mathbb{N} + 2b.
\end{aligned} \tag{6.26}$$

Again, in steps three and four, we used the fact that the random variables are independent. As the variance shrinks with n_{pp}, one can obtain a good estimate by using many pixel pairs. Also note that the variance depends on the value of s. This is called heteroscedasticity and has to be taken into account when performing the linear regression.

In order to estimate the values of a and b, we need good estimates of $\mathbb{E}\left(\tilde{v}\,\middle|\,\tilde{s}_1 = \cdots = \tilde{s}_{n_{pp}} = s\right)$ for multiple values of s. To obtain these estimates, we need many pixel pairs where both pixels have the same average electron count μ. Healey and Kondepudy [1994] suggest using corresponding pixels of two images taken with the same configuration. In our case, that would mean taking two images at the same position of the specimen using the same microscope settings. However, this is infeasible due to several reasons:

- The approach would be very time-consuming. As mentioned before, it takes very long to record SEM images with a reasonable resolution and signal-to-noise ratio. Taking two images of each specimen position would result in only having half the number of images for analysis.
- Existing images of which only one version is available could not be used at all because the exact circumstances under which they were taken are not reproducible in order to take a second image.
- The electron beam may alter the specimen. An example for this effect is that gas residuals can form a layer on the specimen (see Section 4.2.2.5). This way, a second image would have different characteristics than the first [Ribeiro and Shah, 2006].
- The point on the specimen that maps to a given pixel may change between the two images. In other words, the electron beam may scan different points on the specimen for the same pixel even if the specimen position is not altered. This effect can be caused by small fluctuations such as vibrations (see Section 4.2.2.2).

Due to these reasons, we have decided to assume that neighboring pixels in the same image have the same probability distribution. There are several factors that permit this assumption:

- Our images have high magnifications and pixel counts. That means that neighboring pixels represent points on the specimen that are very close together. These are very likely to have similar material properties and surface orientations.
- Most of the images we use have large homogeneous regions where neighboring pixels have the same expected intensity.
- The images are slightly blurry because the electron beam diameter is larger than the pixel size. Therefore, neighboring pixels represent overlapping areas on the specimen.

Although this assumption is reasonable, it is not always the case, especially at edges in the images. Therefore, we need a way to exclude pixel pairs that violate our assumption from the estimation. In order to do that, let us define the average intensity difference of the pixel pairs as a random variable $\tilde{u} : \Omega \to \mathbb{R}$:

$$\tilde{u} := \frac{1}{n_{pp}} \sum_{i=1}^{n_{pp}} \tilde{d}_i. \tag{6.27}$$

We want to calculate its expected value given the sum. To do that, we need the expected intensity difference of two pixels given the sum:

$$\begin{aligned} &\mathbb{E}(\tilde{d} \mid \tilde{s} = s) \\ \underset{(6.21)}{=}\; &\mathbb{E}(\tilde{g}_1 - \tilde{g}_2 \mid \tilde{s} = s) \\ \underset{(6.20)}{=}\; &\mathbb{E}(\tilde{g}_1 - \tilde{g}_2 \mid \tilde{g}_1 + \tilde{g}_2 = s) \\ =\; &\mathbb{E}(\tilde{g}_1 \mid \tilde{g}_1 + \tilde{g}_2 = s) - \mathbb{E}(\tilde{g}_2 \mid \tilde{g}_1 + \tilde{g}_2 = s) \\ =\; &\mathbb{E}(\tilde{g}_1 \mid \tilde{g}_1 + \tilde{g}_2 = s) - \mathbb{E}(\tilde{g}_1 \mid \tilde{g}_2 + \tilde{g}_1 = s) \\ =\; &0, \quad \forall\, s \in a\mathbb{N} + 2b. \end{aligned} \tag{6.28}$$

Here, we have used that $\tilde{g}_1$ and $\tilde{g}_2$ have the same probability distribution. With this result, we can calculate the expected value of the average intensity difference given the sum:

$$\begin{aligned} &\mathbb{E}\left(\tilde{u} \,\middle|\, \tilde{s}_1 = \cdots = \tilde{s}_{n_{pp}} = s\right) \\ \underset{(6.27)}{=}\; &\mathbb{E}\left(\frac{1}{n_{pp}} \sum_{i=1}^{n_{pp}} \tilde{d}_i \,\middle|\, \tilde{s}_1 = \cdots = \tilde{s}_{n_{pp}} = s\right) \\ =\; &\frac{1}{n_{pp}} \sum_{i=1}^{n_{pp}} \mathbb{E}\left(\tilde{d}_i \,\middle|\, \tilde{s}_1 = \cdots = \tilde{s}_{n_{pp}} = s\right) \\ =\; &\frac{1}{n_{pp}} \sum_{i=1}^{n_{pp}} \mathbb{E}(\tilde{d}_i \mid \tilde{s}_i = s) \\ \underset{(6.28)}{=}\; &\frac{1}{n_{pp}} \sum_{i=1}^{n_{pp}} 0 \\ =\; &0, \quad \forall\, s \in a\mathbb{N} + 2b. \end{aligned} \tag{6.29}$$

In the second step, we have once again used the fact that the random variables are independent.

This means that if neighboring pixels have the same intensity distribution, the average difference of multiple pairs will be close to 0. If this is not the case, our assumption will probably be violated. We can use this to remove these pixels from the estimation. In order to see how small the average difference has to be, we want to calculate the variance of $\tilde{u}$. To do that, we need to compute $\operatorname{Var}(\tilde{d} \mid \tilde{s} = s)$ first:

$$\begin{aligned} &\operatorname{Var}(\tilde{d} \mid \tilde{s} = s) \\ =\ &\mathbb{E}\Big(\big(\tilde{d} - \mathbb{E}(\tilde{d} \mid \tilde{s} = s)\big)^2 \,\Big|\, \tilde{s} = s\Big) \\ \underset{(6.28)}{=}\ &\mathbb{E}(\tilde{d}^2 \mid \tilde{s} = s) \\ \underset{(6.22)}{=}\ &as - 2ab, \quad \forall s \in a\mathbb{N} + 2b. \end{aligned} \tag{6.30}$$

Now, we can calculate $\operatorname{Var}\Big(\tilde{u} \,\Big|\, \tilde{s}_1 = \cdots = \tilde{s}_{n_{pp}} = s\Big)$:

$$\begin{aligned} &\operatorname{Var}\Big(\tilde{u} \,\Big|\, \tilde{s}_1 = \cdots = \tilde{s}_{n_{pp}} = s\Big) \\ \underset{(6.27)}{=}\ &\operatorname{Var}\left(\frac{1}{n_{pp}} \sum_{i=1}^{n_{pp}} \tilde{d}_i \,\middle|\, \tilde{s}_1 = \cdots = \tilde{s}_{n_{pp}} = s\right) \\ =\ &\frac{1}{{n_{pp}}^2} \sum_{i=1}^{n_{pp}} \operatorname{Var}\Big(\tilde{d}_i \,\Big|\, \tilde{s}_1 = \cdots = \tilde{s}_{n_{pp}} = s\Big) \\ =\ &\frac{1}{{n_{pp}}^2} \sum_{i=1}^{n_{pp}} \operatorname{Var}(\tilde{d}_i \mid \tilde{s}_i = s) \\ \underset{(6.30)}{=}\ &\frac{1}{{n_{pp}}^2} \sum_{i=1}^{n_{pp}} as - 2ab \\ =\ &\frac{1}{n_{pp}}(as - 2ab), \quad \forall s \in a\mathbb{N} + 2b. \end{aligned} \tag{6.31}$$

If the average intensity difference of a set of pixel pairs with the same sum deviates from 0, we can compare that deviation to the standard deviation $\sqrt{\frac{as-2ab}{n_{pp}}}$. This way, we can assess whether our assumption is violated.

6.3.1.2 *Noise Estimation Algorithm*

Our goal is to calculate the noise variance $\operatorname{Var}(\tilde{o})$ as a function of the local image intensity. We can do this using Equation (6.19) if we have estimates of a and b. In Equation (6.25), we have seen that there is a linear relationship between s and $\mathbb{E}\Big(\tilde{v} \,\Big|\, \tilde{s}_1 = \cdots = \tilde{s}_{n_{pp}} = s\Big)$, whose parameters depend on the values of a and b. This means that if we estimate $\mathbb{E}\Big(\tilde{v} \,\Big|\, \tilde{s}_1 = \cdots = \tilde{s}_{n_{pp}} = s\Big)$ for different values of s, we will be able to estimate the values of a and b with linear regression.

To calculate these estimates, we want to use neighboring pairs of pixels from the given image. This is possible because the probability distributions of the electron counts of the two pixels are very similar in most cases.

The SEM we use captures images row-wise. Therefore, we have chosen to use horizontally neighboring pixel pairs as they are captured in direct succession. Between rows, small fluctuations may change the probability distribution governing the electron counts which violates our assumption. Additionally, each pixel is used in only one pixel pair. Otherwise, different pixel pairs would not be independent.

While the theoretical model describes the pixel generating process relatively well, it does not take intensity clipping into account. If the amplified signal is too high (or low) for the available intensity range, its value is clipped to the highest (or lowest) available intensity (see Section 4.2.2.3). To reduce the impact of intensity clipping on our estimates, we have chosen to remove all pixel pairs from the estimation process, where at least one pixel has the lowest or highest available intensity. This cannot fully remove the impact of intensity clipping. However, it reduces its impact.

Figure 6.6 shows a plot which has been generated by grouping all pixel pairs of the image in Figure 1.2 on Page 5 by the sum s of their intensities, calculating the average squared intensity difference using Equation (6.24) and plotting these pairs. As indicated by the black line, there is a clear linear relationship. For the values of s from about 30 to 250, there are outliers in the plot. These stem from the edges in the image where our assumption is not valid. Note that we have not removed points that are likely to violate our assumption for this plot. In addition, for very low and very high values of s, the average squared intensity difference is smaller than expected. This is due to intensity clipping. Because of these outliers, we have chosen to use a robust estimation method instead of simple linear least squares. We use an M-estimation algorithm called iteratively reweighted least squares because it is reliable and easy to compute [Maronna et al., 2006, p. 87].

As shown in Equation (6.26), the variance of the estimates changes based on the value of s and the number of pixel pairs used to obtain the estimate. This means that the estimates are better for smaller values of s and a higher pixel pair count. Figure 6.6 is color-coded to indicate how many pixel pairs where used to calculate each estimate. It can be seen that the data points following the linear relationship have been obtained using far more pixel pairs than the outliers and, thus, are better estimates. In order to use this information, we divide the coordinates of each data point by the expected standard deviation derived from Equation (6.26) during the regression as suggested by Maronna et al. [2006, p. 154]. However, to calculate the standard deviation, we need the values of a and b. Therefore, in the first run, we divide the data by $\sqrt{\frac{1}{n_{pp}}}$ to obtain initial estimates of a and b and then perform a second run where the points are divided by their standard deviations.

As mentioned before, Equations (6.29) and (6.31) allow us to detect values of s for which our assumption is violated. If, for instance, in the area of a particle edge, the right pixels of each pair are commonly

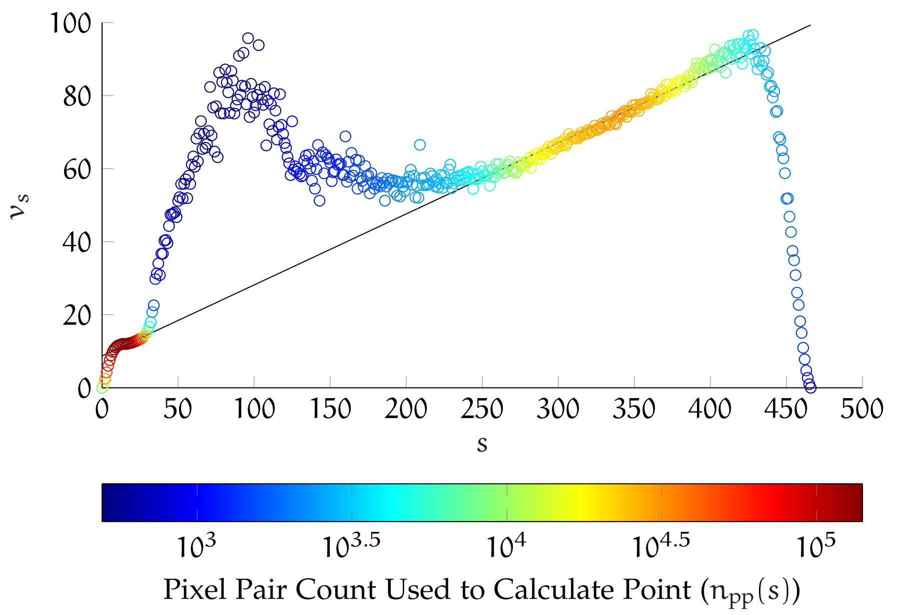


Figure 6.6: The average squared intensity differences v_s of all horizontal pixel pairs of the image in Figure 1.2 on Page 5 grouped by their sum s. The y-coordinate of each point represents the average squared intensity difference v_s of all pixel pairs whose intensity sum s is equal to the x-coordinate of the point. Its color codes the number $n_{pp}(s)$ of pixel pairs that have that sum s. The black line illustrates the linear relationship between s and v_s. The data has been calculated using Algorithm 6.1 with $n_{tile} = 1$, which will be introduced later in this chapter.

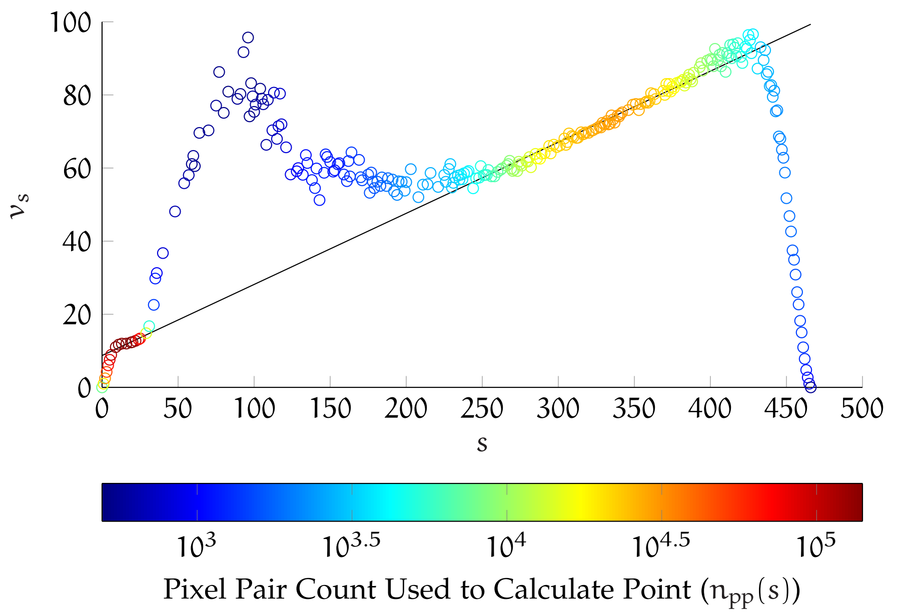


Figure 6.7: The same data as in Figure 6.6 except that points for which $|u_s| \geqslant \sqrt{\frac{1}{n_{pp}(s)}(as - 2ab)}$ holds have been removed (see Equation (6.31)). The removal has been performed as outlined in Algorithm 6.2 in Line 12.

brighter than the left pixels, the average intensity difference of the pairs deviates from 0. In this case, we want to remove those pixel pairs from the estimation. We have chosen to remove the sets of pixel pairs where the average intensity sum deviates more than the standard deviation from 0. This way, we remove approximately 32 % of the estimates that meet our assumption. However, the estimates that violate our assumption are more likely to be removed. Consequently, the fraction of pixel pairs violating our assumption gets smaller, which leads to more accurate results. Figure 6.7 shows the same data as Figure 6.6 with the exception that the points where the absolute average intensity difference is larger than the standard deviation have been removed. Especially the number of outliers with values of s from about 30 to 250 has been reduced. This is the expected result because these stem from the edges in the image. Since relatively good estimates of a and b are required to calculate the standard deviation, we only apply this step in a third regression pass in addition to dividing each point by its standard deviation to compensate for heteroscedasticity.

One remaining problem is that left and right edges cancel each other out in the computation of the average intensity difference. This prevents estimates violating our assumption from being removed. To avoid this, we split the image into $n_{\text{tile}} \times n_{\text{tile}}$ uniform tiles with $n_{\text{tile}} \in \mathbb{N}_{>0}$ and calculate the estimates separately for each tile. This way, the left and right edges of a larger particle are likely to lie in different tiles which prevents them from canceling each other out.

The goal of our algorithm is to find the vector $\vec{\alpha} = \left(\begin{smallmatrix} \hat{a} \\ -2\hat{a}\hat{b} \end{smallmatrix}\right) \in \mathbb{R}^2$ where $\hat{a}$ and $\hat{b}$ are estimates of a and b, respectively. Equation (6.25) tells us that if these estimates were correct, $\vec{\alpha}$ would solve the following equation:

$$\begin{pmatrix} s_1 & 1 \\ s_2 & 1 \\ s_3 & 1 \\ \vdots & \vdots \end{pmatrix} \cdot \vec{\alpha} = \begin{pmatrix} \mathbb{E}\left(\tilde{v} \,\middle|\, \tilde{s}_1 = \cdots = \tilde{s}_{n_{\text{pp}}} = s_1\right) \\ \mathbb{E}\left(\tilde{v} \,\middle|\, \tilde{s}_1 = \cdots = \tilde{s}_{n_{\text{pp}}} = s_2\right) \\ \mathbb{E}\left(\tilde{v} \,\middle|\, \tilde{s}_1 = \cdots = \tilde{s}_{n_{\text{pp}}} = s_3\right) \\ \vdots \end{pmatrix}, \tag{6.32}$$

where $s_1, s_2, s_3, \cdots \in \mathbb{R}$ are the possible sums of two intensities.

The steps of our method are detailed in Algorithms 6.1 and 6.2. The main algorithm is Algorithm 6.2. Its first step is to call Algorithm 6.1, which computes values of n_{pp}, $\tilde{v}$ and $\tilde{u}$ grouped by intensity sum s and image tile. These values are then returned as vectors to the main algorithm. As a next step, a matrix $\overset{\square}{s}$ similar to the one in Equation (6.32) is constructed and used to obtain a first estimate $\vec{\alpha}$ using simple linear least squares. Then, iteratively reweighted least squares is used three times to improve the estimate. First, it is used on the original data. In the second and third round, the data is divided by the estimated standard deviation to account for the heteroscedasticity.

Algorithm 6.1 : `ExtractEstimates`(f, n_{tile})

```
Input : An image $f : P \to G$ and the parameter $n_{\text{tile}} \in \mathbb{N}_{>0}$, which
        determines into how many tiles the image is split.
Output : Vectors $\vec{s}, \vec{n}_{\text{pp}}, \vec{v}, \vec{u} \in \mathbb{R}^{|G+G| \cdot {n_{\text{tile}}}^2}$.
1  Split the image f into $n_{\text{tile}} \times n_{\text{tile}}$ uniform tiles.
2  for each tile $i \in \{1, \ldots, {n_{tile}}^2\}$ do
3      Find as many horizontally neighboring pixel pairs in tile $i$ so
       that no pixel is shared between two pixel pairs.
4      for every possible intensity sum $s \in G + G$ do
5          Let $n_{\text{pp}}(s, i) \in \mathbb{N}$ be the number of pixel pairs in tile $i$
           whose intensity sums are equal to $s$ and of which neither
           possesses the minimum or maximum possible intensity.
6          Let $d_{s,i,1}, \ldots, d_{s,i,n_{\text{pp}}(s,i)} \in \mathbb{R}$ be the intensity differences
           of these pairs.
7          if $n_{pp}(s, i) > 0$ then
8              $v_{s,i} := \frac{1}{n_{\text{pp}}(s,i)} \sum_{j=1}^{n_{\text{pp}}(s,i)} d^2_{s,i,j}$. // See Equation (6.24).
9              $u_{s,i} := \frac{1}{n_{\text{pp}}(s,i)} \sum_{j=1}^{n_{\text{pp}}(s,i)} d_{s,i,j}$. // See Equation (6.27).
10         end
11     end
12 end
13 Define the vectors $\vec{s}, \vec{n}_{\text{pp}}, \vec{v}, \vec{u} \in \mathbb{R}^{|G+G| \cdot {n_{\text{tile}}}^2}$ as follows. If any of
   the $v_{s,i}$ or $u_{s,i}$ are not defined (i.e. if $n_{\text{pp}}(s, i) = 0$), leave out the
   corresponding entries in *all* vectors. In this case, all vectors have
   fewer than $|G + G| \cdot {n_{\text{tile}}}^2$ entries:
```

$$
\begin{aligned}
\vec{s} &:= \begin{pmatrix} s_1 \\ \vdots \\ s_1 \\ s_2 \\ \vdots \\ \vdots \\ s_{|G+G|} \end{pmatrix}, \quad \vec{n}_{\text{pp}} := \begin{pmatrix} n_{\text{pp}}(s_1, 1) \\ \vdots \\ n_{\text{pp}}(s_1, {n_{\text{tile}}}^2) \\ n_{\text{pp}}(s_2, 1) \\ \vdots \\ \vdots \\ n_{\text{pp}}(s_{|G+G|}, {n_{\text{tile}}}^2) \end{pmatrix}, \\
\vec{v} &:= \begin{pmatrix} v_{s_1,1} \\ \vdots \\ v_{s_1,{n_{\text{tile}}}^2} \\ v_{s_2,1} \\ \vdots \\ \vdots \\ v_{s_{|G+G|},{n_{\text{tile}}}^2} \end{pmatrix}, \quad \vec{u} := \begin{pmatrix} u_{s_1,1} \\ \vdots \\ u_{s_1,{n_{\text{tile}}}^2} \\ u_{s_2,1} \\ \vdots \\ \vdots \\ u_{s_{|G+G|},{n_{\text{tile}}}^2} \end{pmatrix}.
\end{aligned}
\tag{6.33}
$$

```
return $\vec{s}, \vec{n}_{pp}, \vec{v}, \vec{u}$.
```

Algorithm 6.2 : *EstimateNoiseParameters*(f, n_{tile})

```
   Input : An image $f : P \to G$ and the parameter $n_{\text{tile}} \in \mathbb{N}_{>0}$, which
           determines into how many tiles the image is split.
   Output : Estimates $\hat{a}, \hat{b} \in \mathbb{R}$ of the noise parameters $a$ and $b$ of
            the image $f$.
 1 $\vec{s}, \vec{n}_{pp}, \vec{v}, \vec{u}$ := ExtractEstimates$(f, n_{\text{tile}})$. // See Algorithm 6.1.
   // Define the matrix $\overset{\square}{s} \in \mathbb{R}^{(|G+G| \cdot n_{\text{tile}}{}^2) \times 2}$ with
      $|G+G| \cdot n_{\text{tile}}{}^2$ rows and 2 columns so that the first
      column is equal to $\vec{s}$ and the second column contains
      only 1s (Note that vector and matrix entries are
      accessed using indices inside parentheses):
 2 $\overset{\square}{s}(k,1) := \vec{s}(k),\ \overset{\square}{s}(k,2) := 1, \quad \forall\, k \in \left\{1, \dots, |G+G| \cdot n_{\text{tile}}{}^2\right\}$
   // Calculate an initial estimate $\vec{\alpha}_0 \in \mathbb{R}^2$ of the solution
      to the equation $\overset{\square}{s}\vec{\alpha}_0 = \vec{v}$ (see Equation (6.32)) using
      linear least squares ($\overset{\square}{s}^{T}$ is the transpose of $\overset{\square}{s}$):
 3 $\vec{\alpha}_0 := \left(\overset{\square}{s}^{T}\overset{\square}{s}\right)^{-1}\overset{\square}{s}^{T}\vec{v}$. // Initial estimate. Can optionally be
      weighted using $\vec{n}_{pp}$ as in Lines 6 to 9.
 4 for $j = 1, 2, 3$ do
      // Define $\overset{\square}{s}_j \in \mathbb{R}^{(|G+G| \cdot n_{\text{tile}}{}^2) \times 2}$ and $\vec{v}_j \in \mathbb{R}^{(|G+G| \cdot n_{\text{tile}}{}^2)}$ by
         dividing each row of $\overset{\square}{s}$ and $\vec{v}$ by the estimated
         standard deviation of $\tilde{v}$ (see Equation (6.26)) as
         suggested by Maronna et al. [2006, p. 154]:
 5    for $k \in \left\{1, \dots, |G+G| \cdot n_{tile}{}^2\right\}$ do
 6       $\hat{\sigma}_{\tilde{v}} := \begin{cases} \sqrt{\frac{1}{\vec{n}_{pp}(k)}}, & \text{if } j = 1, \\ \sqrt{\frac{2\hat{a}_{j-1}^2\left(\vec{s}(k) - 2\hat{b}_{j-1}\right)\left(\left(\vec{s}(k) - 2\hat{b}_{j-1}\right) - \hat{a}_{j-1}\right)}{\vec{n}_{pp}(k)}}, & \text{else.} \end{cases}$
 7       $\overset{\square}{s}_j(k,1) := \overset{\square}{s}(k,1)/\hat{\sigma}_{\tilde{v}}$.
 8       $\overset{\square}{s}_j(k,2) := \overset{\square}{s}(k,2)/\hat{\sigma}_{\tilde{v}}$.
 9       $\vec{v}_j(k) := \vec{v}(k)/\hat{\sigma}_{\tilde{v}}$.
10    end
11    if $j = 3$ then
12       Remove all rows $k \in \left\{1, \dots, |G+G| \cdot n_{\text{tile}}{}^2\right\}$ of $\overset{\square}{s}_j$ and $\vec{v}_j$
         for which $|\vec{u}(k)| \geqslant \sqrt{\frac{1}{\vec{n}_{pp}(k)}\left(\hat{a}_{j-1}\vec{s}(k) - 2\hat{a}_{j-1}\hat{b}_{j-1}\right)}$ holds.
         // See Equation (6.31).
13    end
      // Perform iteratively reweighted least squares
         [Maronna et al., 2006, p. 105] to obtain the
         estimate $\vec{\alpha}_j$ as the solution to the equation
         $\overset{\square}{s}_j\vec{\alpha}_j = \vec{v}_j$ with the initial estimate $\vec{\alpha}_{j-1}$:
14    $\vec{\alpha}_j$ := IterativelyReweightedLeastSquares$(\overset{\square}{s}_j, \vec{v}_j, \vec{\alpha}_{j-1})$.
         // See Algorithm B.1 in Appendix B on Page 205.
      // Extract the estimates $\hat{a}_j \in \mathbb{R}$ and $\hat{b}_j \in \mathbb{R}$ from $\vec{\alpha}_j$:
15    $\hat{a}_j = \vec{\alpha}_j(1)$.
16    $\hat{b}_j = -\frac{\vec{\alpha}_j(2)}{2\hat{a}_j}$.
17 end
18 return $\hat{a} := \hat{a}_3,\ \hat{b} := \hat{b}_3$.
```

In the third pass, the data points where the absolute average intensity difference deviates more than its standard deviation from 0 are removed, as well.

The iteratively reweighted least squares algorithm [Maronna et al., 2006, p. 105] used in our method is a robust replacement of simple least squares. Understanding its functional principle is not necessary to understand our method. However, the interested reader can find its steps in Algorithm B.1 in Appendix B on Page 205.

Using the method described in this section, we are able to estimate the values of a and b for a given image. This allows us to calculate the noise variance given the expected pixel intensity using Equation (6.19) and, thus, to adjust the parameters of a noise removal method to fit the noise of the image. This process is explained in the next section.

6.3.2 *Noise Removal*

Having estimated the contrast-setting a and the brightness-setting b of an image, we know the characteristics of the image noise via Equation (6.19). This knowledge can be used to adjust a noise filter to the noise characteristics of each image. Removing heteroscedastic noise can be done using one of two approaches [Foi, 2009]:

- Use an algorithm specifically designed for heteroscedastic noise.
- Employ a variance-stabilizing transformation to make the image noise homoscedastic, which means that the noise of each pixel has the same variance, and then use any filter designed for Gaussian noise.

We have chosen to use the second approach because much more research has been done on removing white noise than signal-dependent noise. As a result, a much greater number of methods targeting homoscedastic noise have been proposed. Section 6.3.2.1 describes how the variance-stabilizing transformation works and Section 6.3.2.2 details the removal of the homoscedastic noise.

6.3.2.1 *Variance-Stabilizing Transformation*

Noise removal using variance-stabilizing transformation has three steps [Foi, 2009]:

1. The value of each pixel is converted using a transformation formula so that the noise of the resulting image has an approximately fixed variance.
2. A denoising algorithm for additive white noise is applied to the transformed image.

3. The inverse transformation is applied to each pixel of the resulting image.

The result of this process is a denoised image.

For signals resulting from a Poisson process, the Anscombe transformation (also called Anscombe transform) proposed by Anscombe [1948] is used for this purpose. Its formula is as follows:

$$2\sqrt{\tilde{c} + \frac{3}{8}}, \tag{6.34}$$

where $\tilde{c}$ is the original signal, which, in our case, is the electron count. Simply using the inverted formula as the inverse transformation leads to a biased result [Mäkitalo and Foi, 2011b]. We therefore use the unbiased inverse transformation proposed by Mäkitalo and Foi [2011a].

In addition to the basic transformation, we first have to invert the scaling performed by the SEM using the estimated contrast setting $\hat{a}$ and brightness setting $\hat{b}$ gained using Algorithm 6.2. At the end of the whole procedure, the scaling is reapplied to the resulting image.

The whole algorithm is described in Algorithm 6.3. Here, the parameter `RemoveNoise` is the denoising algorithm targeted at additive white noise. We use a method called non-local means for this task. It will be explained in the next section.

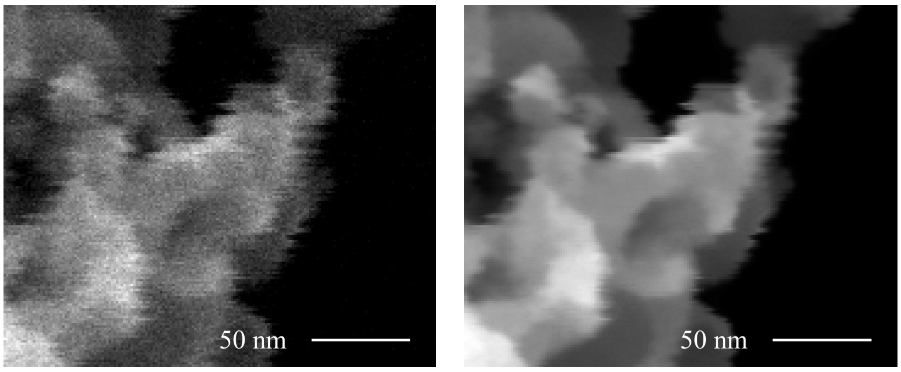
6.3.2.2 *Homoscedastic Noise Removal*

The Anscombe transform, which we described in the previous section, allows us to use a standard noise removal method targeted at white noise for our SEM images. We have chosen to use an algorithm called non-local means by Buades et al. [2005b] for this task. This has several reasons:

- It makes no concrete assumptions about the image content such as the direction of edges. Assumptions true for most types of images could be false for SEM images as these do not stem from an optical process.
- Non-local means proves to be the best algorithm in a review by Buades et al. [2005a], especially for images with a low signal-to-noise ratio, as is the case for SEM images.
- In our experience, the algorithm removes no visible image details. It even preserves small image artifacts that are not the effect of noise. This can be seen in Figure 6.8.
- A microscopy expert confirmed that non-local means coupled with our noise estimation method and the Anscombe transform is able to improve the effective resolution of SEM images [Kockentiedt et al., 2013].

Algorithm 6.3 : *ScaledAnscombeTransform*$(f, \hat{a}, \hat{b},$ *RemoveNoise*$)$

```
Input : An image $f : P \to G$ with scaled Poisson noise, estimates
        $\hat{a} \in \mathbb{R}_{>0}, \hat{b} \in \mathbb{R}_{<0}$ of the noise parameters $a$ and $b$ and an
        algorithm RemoveNoise to remove white noise.
Output : An image $f' : P \to G$, where the noise has been
         removed.
1  Let $f_1 : P \to \mathbb{R}$ be an empty image. for $p \in P$ do
       // The estimated electron count of pixel $p$ (see
          Equation (6.12)):
2      $\hat{c} := \frac{f(p)-\hat{b}}{\hat{a}}$.
       // Perform the Anscombe transform [Anscombe, 1948]:
3      $f_1(p) := 2\sqrt{\hat{c}+\frac{3}{8}}$.
4  end
   // $f_1$ now has an approximately uniform noise variance of
      1. Apply the noise removal method to it:
5  $f_2 :=$ RemoveNoise$(f_1)$.
6  Let $f_3 : P \to \mathbb{R}$ be an empty image. for $p \in P$ do
       // Perform the inverse Anscombe transform [Mäkitalo
          and Foi, 2011a]:
7      $\hat{c}_{avg} :=$
       $\frac{1}{4}(f_2(p))^2 + \frac{\sqrt{6}}{8}(f_2(p))^{-1} - \frac{11}{8}(f_2(p))^{-2} + \frac{5\sqrt{6}}{16}(f_2(p))^{-3} - \frac{1}{8}$.
8      $f_3(p) := \hat{a}\hat{c}_{avg} + \hat{b}$. // See Equation (6.12).
9  end
10 Obtain the final image $f' : P \to G$ by rounding the values of $f_3$ to
   values in $G$.
11 return $f'$.
```



(a) Jagged edges caused by vibrations or electromagnetic radiation in an image detail (1 pixel = 1.27 nm) of TiO_2 nanoparticles.

(b) The same image detail after applying our noise removal approach. Global intensity differences are due to different intensity scalings.

Figure 6.8: SEM image artifacts before and after application of our noise removal method. The result is a very smooth image where edges and artifacts are preserved.

- As Figure 6.8 shows, the method is able to provide smooth results without disturbing edges in the images. This is especially important as the resulting images are only used for thresholding. They are not used for further analysis.

The premise of the non-local means algorithm is that the value of each pixel of the resulting image is a weighted average of all pixels of the original image. This is similar to the concept of a Gaussian filter. In contrast to the Gaussian filter, however, the weight of a pixel does not depend on its distance to the center pixel. Instead, the weight depends on the similarity of the pixel's neighborhood to the neighborhood of the center pixel, where the size of each neighborhood is $(2r_n + 1) \times (2r_n + 1)$ pixels and $r_n \in \mathbb{N}_{>0}$ is the neighborhood radius. In other words, if the neighborhood of pixel p in the original image is more similar to the neighborhood of pixel p_1 than to the one of pixel p_2, the weight of p_1 in the computation of the new value of p will be greater than the weight of p_2.

However, since it is computationally not feasible to compare the neighborhoods of every pixel pair, the search for similar pixels is restricted to a so-called search window of size $(2r_s + 1) \times (2r_s + 1)$ pixels around the center pixel, where $r_s \in \mathbb{N}_{>0}$ is the search window radius.

In addition, the algorithm has a filtering parameter $\eta \in \mathbb{R}_{>0}$, which determines how strong the image is smoothed.

The steps of the method are listed in Algorithm 6.4. For the sakes of brevity and understandability, we left out image edge handling in the listing. We use a modified version of the implementation by Kroon [2010]. It restricts the search window to the bounds of the image and mirrors the pixels inside the image to compute the pixel values of a neighborhood that lie outside the image.

When comparing two neighborhoods, pixels closer to the center of the neighborhood have more influence than pixels at the edges. This is achieved by weighting the convolution using the Gaussian kernel $\mathcal{G}$. In the implementation we use, its value is calculated as $\frac{2r_n+1}{4}$, which is a fourth of the kernel width.

Now, we have all the building blocks for estimating the parameters of scaled Poisson noise and removing it. Algorithm 6.5 shows the full process as it is used by us. In the listing, the expression $\mathit{ScaledAnscombeTransform}(f, \hat{a}, \hat{b}, \mathit{NonLocalMeans}(\cdot, r_n, r_s, \eta))$ signifies that *NonLocalMeans* is used as the parameter *RemoveNoise* of the algorithm *ScaledAnscombeTransform* using the fixed parameters r_n, r_s and η.

Note that we use noise removal only for the thresholding and not for any other image analysis task such as the computation of features. The process yields optically superior images to the originals and we do not observe that the method removes any meaningful image de-

Algorithm 6.4 : *NonLocalMeans*(f, r_n, r_s, η)

```
Input : An image $f : P \to \mathbb{R}$ with white noise, the neighborhood
        radius $r_n \in \mathbb{N}_{>0}$, the search window radius $r_s \in \mathbb{N}_{>0}$ and
        the filtering parameter $\eta \in \mathbb{R}_{>0}$.
Output : An image $f' : P \to \mathbb{R}$, in which the noise has been
         removed.
1 Let $\mathcal{G} : \{-r_n, \dots, r_n\} \times \{-r_n, \dots, r_n\} \to \mathbb{R}_{>0}$ be a Gaussian kernel
  with sum 1.
2 Let $f' : P \to \mathbb{R}$ be an empty image. for $p = (p_x, p_y) \in P$ do
      // Define the search window for pixel p:
3     $P' := \{p_x - r_s, \dots, p_x + r_s\} \times \{p_y - r_s, \dots, p_y + r_s\}$.
      // Calculate the weights of the pixels in the search
         window by comparing their neighborhood to the
         neighborhood of p (An appropriate image edge
         handling method has to be used.):
4     for $p' = (p'_x, p'_y) \in P'$ do
5         $w_{px}(p') := e^{-\frac{\sum_{i,j=-r_n}^{r_n} \mathcal{G}(i,j)\cdot(f(p_x+i, p_y+j) - f(p'_x+i, p'_y+j))^2}{\eta^2}}$.
6     end
      // Calculate the new value of pixel p as the weighted
         average of the pixel values in the search window:
7     $f'(p) := \frac{1}{\sum_{p' \in P'} w_{px}(p')} \sum_{p' \in P'} w_{px}(p') \cdot f(p')$.
8 end
9 return $f'$.
```

Algorithm 6.5 : *RemoveScaledPoissonNoise*$(f, n_{\text{tile}}, r_n, r_s, \eta)$

```
Input : An image $f : P \to G$ with scaled Poisson noise, the
        parameter $n_{\text{tile}} \in \mathbb{N}_{>0}$, which determines into how many
        tiles the image is split, the neighborhood radius
        $r_n \in \mathbb{N}_{>0}$, the search window radius $r_s \in \mathbb{N}_{>0}$ and the
        filtering parameter $\eta \in \mathbb{R}_{>0}$.
Output : An image $f' : P \to G$, where the noise has been
         removed.
1 $\hat{a}, \hat{b}$ := EstimateNoiseParameters$(f, n_{\text{tile}})$.
2 $f'$ := ScaledAnscombeTransform$(f, \hat{a}, \hat{b},$ NonLocalMeans$(\cdot, r_n, r_s, \eta))$.
3 return $f'$.
```

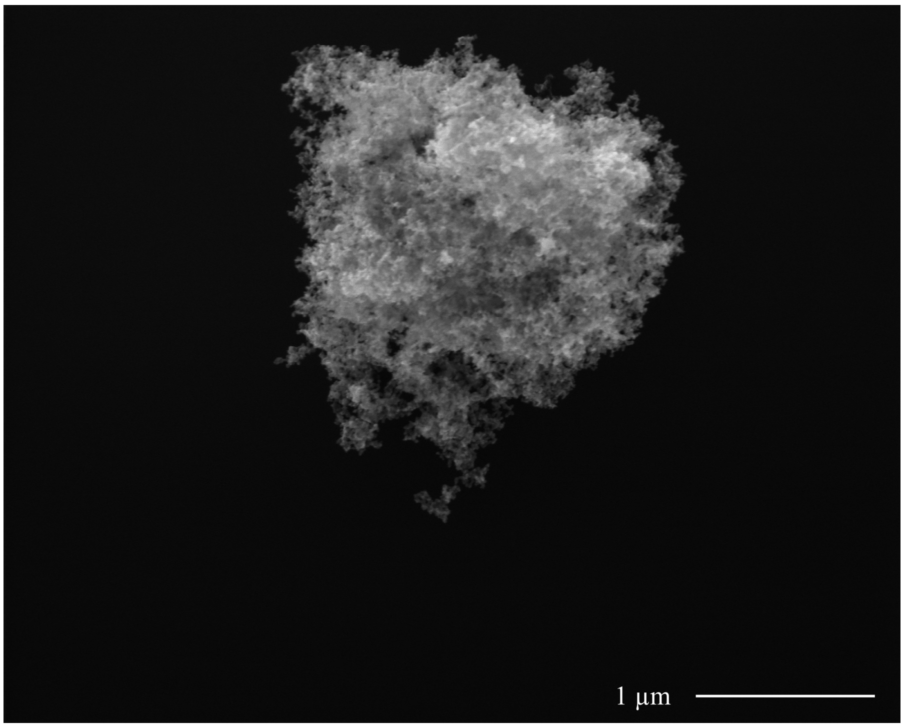


Figure 6.9: An SEM image (1 pixel = 1.27 nm) of ZnO nanoparticles with an average diameter of 10 nm.

tails. However, we cannot rule out the possibility that in addition to noise, the algorithm removes small image variations that a feature can use to differentiate particle types. In fact, we have observed that computing our features on filtered images leads to a reduced classification performance.

After removing the image noise, the next step is to determine the threshold used to segment the image. This will be explained in the next section.

6.3.3 *Threshold Determination*

As a next step, we want to find a threshold to split the image into background and foreground in order to find all CPGs in it. To do that, we have to define the so-called absolute histogram $H_f : G \to \mathbb{N}$ of an image $f : P \to G$:

$$H_f(g) := \sum_{p \in P} \begin{cases} 1, & \text{if } f(p) = g, \\ 0, & \text{else,} \end{cases} \quad \forall f : P \to G, g \in G. \tag{6.35}$$

This function can also be represented by a graph. Figure 6.10 shows the histogram of the image in Figure 6.9. It has only one prominent peak. This is typical for the images we use because most of the area of a particular image is taken up by the background. In addition, the background has a narrow intensity range in contrast to most particles.

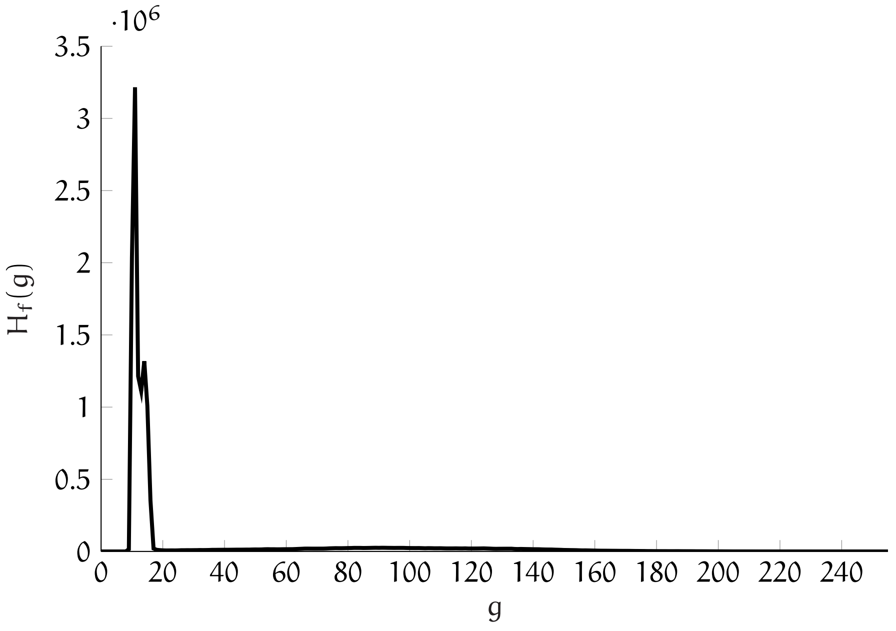


Figure 6.10: A graph of the histogram of the image in Figure 6.9. The peak at an intensity of 11 represents the image background. The intensity distribution of the image foreground is barely visible and has a peak at an intensity of about 90.

Most thresholding methods assume that the histogram is bimodal. This means that the intensities consist of two distinct distributions which are clearly discernible and of approximately equal proportions in the histogram. However, as seen in Figure 6.10, this is not the case for the images we use. While there are in fact two distributions, the one stemming from the particles is usually much less prominent than the background distribution in the histogram. Although the agglomerate takes up a considerable portion of the image because it is very large, its intensity distribution with a peak between 80 and 100 is only barely visible. In a typical image, which shows much smaller agglomerates, the foreground distribution is not visible without magnification.

Oster [2010] successfully uses a method targeted at images with only one peak in their histograms on images very similar to ours. In fact, those images have been taken using the same microscope that was used to record our images. Therefore, we have decided to use the same method, which has been proposed by Zack et al. [1977] and further analyzed by Rosin [2001].

The algorithm assumes that the image intensities stem from two distributions, from which the first has lower intensities and is much more prominent than the second. It does not require that the second distribution is discernible in the histogram, which makes it a good choice for our images. The approach works by drawing a line from the tallest peak in the histogram to the intersection of the highest

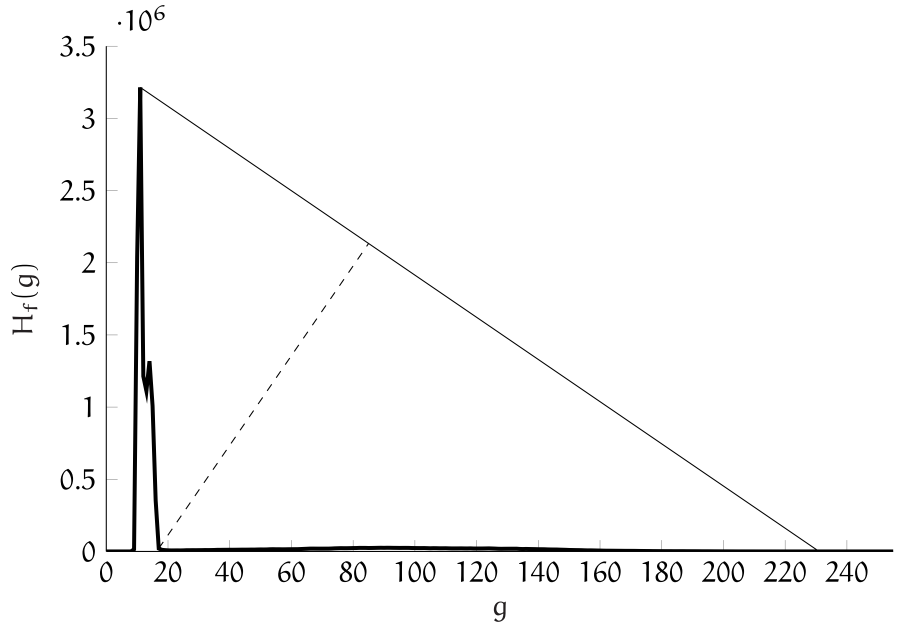


Figure 6.11: An illustration of the thresholding method by Zack et al. [1977] using the histogram from Figure 6.10. The tallest peak is connected to the highest occurring intensity using a straight line (solid line). Then, the point on the histogram between these two intensities that has the greatest distance to the line is chosen (dashed line). Finally, a constant offset is added to the found intensity to obtain the threshold (not displayed).

occurring intensity and the x-axis. As a next step, the point between these two intensities with the greatest distance to the line is taken. Finally, a fixed offset is added to the found intensity to obtain the final threshold. This process is illustrated in Figure 6.11 and formally described in Algorithm 6.6.

In addition to this method, we have also tested an algorithm by Kittler and Illingworth [1986], which shows the best results in a comparison with 40 other approaches [Sezgin and Sankur, 2004]. The workflow of the technique, which is called minimum error thresholding, is as follows:

1. Split the histogram at every possible threshold into two parts.
2. For every threshold and for each of the two parts, compute the mean, standard deviation and relative frequency and model a weighted Gaussian distribution using these parameter values.
3. For every threshold, compute the value of a criterion function, which indirectly reflects the amount of overlap between the Gaussian models of the two parts.
4. Take the threshold with the minimum value of the criterion function.

Algorithm 6.6 : *UnimodalThresholding*(f, δ)

```
  Input : An image $f : P \to G$ and an offset parameter $\delta \in \mathbb{R}$.
  Output : A threshold $g_T$.
  // Find the mode of the histogram:
1 $g_1 := \arg\max_{g \in G} H_f(g)$.
  // Find the maximum intensity of the image:
2 $g_2 := \max(\{g \in G \mid H_f(g) > 0\})$.
  // Find the point in the histogram with the greatest
     distance to the line between the points $(g_1, H_f(g_1))$
     and $(g_2, 0)$:
3 $g_3 := \arg\max_{g \in G, g_1 \leqslant g < g_2} ((g_2 - g_1)(H_f(g_1) - H_f(g)) - (g - g_1)H_f(g_1))$.
4 return $g_T := g_3 + \delta$.
```

In contrast to the algorithm by Zack et al. [1977], this method assumes that there are two distinct and prominent modes in the histogram. As a consequence, it has shown worse results in our experiments (see Section 6.4.3). We therefore use the method by Zack et al. [1977] in our system.

After finding an appropriate threshold, the next step is to search for CPGs in the image. This will be explained in the next section.

6.3.4 *CPG Search*

After a threshold has been found, the image can be split into foreground $P_{FG} \subseteq P$ and background $P_{BG} \subseteq P$. The foreground represents the area of the image where any kind of particle is visible and the background is its complement:

$$P_{BG} := P \setminus P_{FG}. \tag{6.36}$$

The foreground is found by taking every pixel whose intensity is higher than the calculated threshold. This approach works well for images that contain particles. However, in cases where an image contains no particles, the threshold finding method sometimes fails. As a consequence, large portions of the background have an intensity higher than the threshold and are thus regarded as foreground.

To avoid this, we need an algorithm to detect empty images. In our dataset, after the noise removal step, the difference of the highest intensity of any empty image and the one of the darkest non-empty image is 13 (see Section 6.4.4). Therefore, using a threshold $\xi \in \mathbb{R}$ to determine if an image is empty works well for our dataset.

The whole algorithm to find the image foreground P_{FG} is listed in Algorithm 6.7. Figure 6.12 shows the foreground of the image in Figure 1.2 on Page 5.

Algorithm 6.7 : *FindImageForeground*$(f, n_{\text{tile}}, r_n, r_s, \eta, \xi, \delta)$

```
Input : An image $f : P \to G$ with scaled Poisson noise, the
        parameter $n_{\text{tile}} \in \mathbb{N}_{>0}$, which determines into how many
        tiles the image is split for the noise estimation, the
        neighborhood radius $r_n \in \mathbb{N}_{>0}$, the search window radius
        $r_s \in \mathbb{N}_{>0}$, the filtering parameter $\eta \in \mathbb{R}_{>0}$, the empty
        image threshold $\xi \in \mathbb{R}$ and the threshold offset $\delta \in \mathbb{R}$.
Output : The image foreground $P_{FG} \subseteq P$.
1 $f' :=$ RemoveScaledPoissonNoise$(f, n_{\text{tile}}, r_n, r_s, \eta)$.
2 if $\max_{p \in P}(f'(p)) > \xi$ then
3     $g_T :=$ UnimodalThresholding$(f', \delta)$.
4     $P_{FG} := \{p \in P \mid f'(p) > g_T\}$.
5 else
6     $P_{FG} := \emptyset$.
7 end
8 return $P_{FG}$.
```

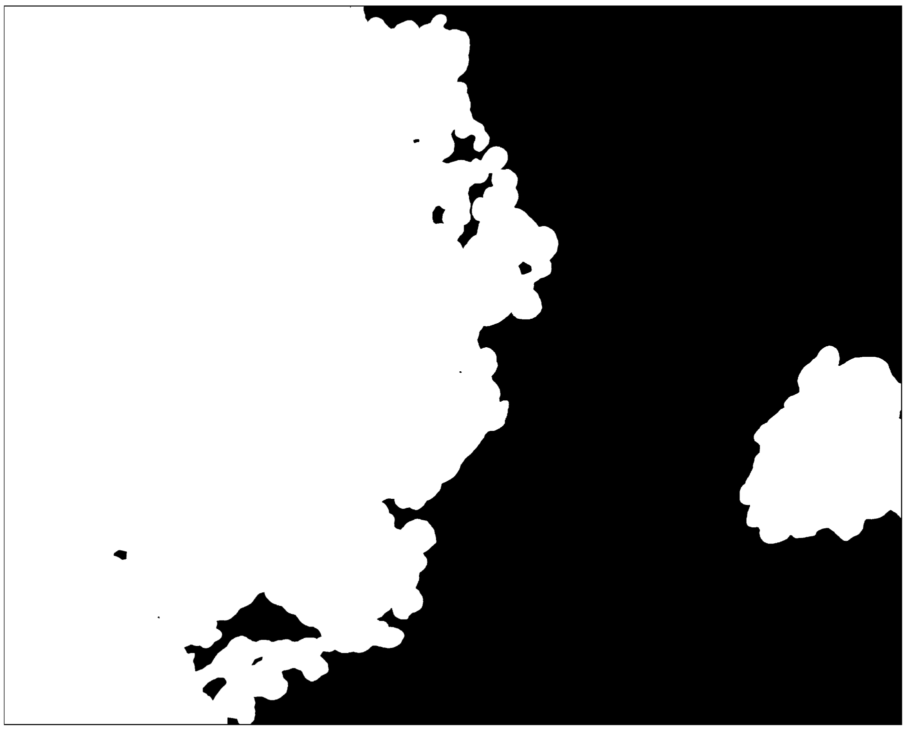


Figure 6.12: The foreground P_{FG} of the image in Figure 1.2 on Page 5 represented as white pixels on a black background.

As a next step, the image foreground has to be split into CPGs $C_1, \dots, C_{n_C} \subseteq P_{FG}$ as introduced in Section 6.1. Each one has to be connected as defined on Page 57 and they must satisfy Equation (6.7), that is, two CPGs may not be directly adjacent. This can be accomplished using an algorithm to find connected components in a binary image such as the one described by Shapiro and Stockman [2001, p. 76].

In addition, each CPG shall have a minimum size $M_{CPG} \in \mathbb{N}_{>0}$. While the noise removal method we use works well, it is still possible that the thresholding produces components that contain only one or a few pixels. Since the nanoparticles we want to find have a well defined size and the magnification has been chosen so that the pixel size is considerably smaller than the nanoparticle size, it is unlikely that such small components represent a particle. Therefore, it is reasonable to introduce a minimum CPG size M_{CPG}. A formal description of the method we use to find the CPGs can be found in Algorithm 6.8.

```
Algorithm 6.8 : FindCPGs($P_{FG}, M_{CPG}$)
  Input : The image foreground $P_{FG} \subseteq P$ and the minimum CPG
          size $M_{CPG} \in \mathbb{N}_{>0}$.
  Output : The CPGs $C_1, \dots, C_{n_C} \subseteq P$.
1 Let $C'_1, \dots, C'_{n_C'} \subseteq P$ with $n_C' \in \mathbb{N}$ and $\bigcup_{i=1}^{n_C'} C'_i = P_{FG}$ be the
  components of $P_{FG}$ so that each one is connected using the
  4-connected neighborhood $N_4$ as defined on Page 57 and they
  satisfy Equation (6.7).
  // We find these components using the algorithm from
     Shapiro and Stockman [2001, p. 76].
  // Retain only those components with a size $\geqslant M_{CPG}$:
2 Define $n_C \in \mathbb{N}$ and $C_1, \dots, C_{n_C} \subseteq P$ so that
  $C_i \neq C_j, \forall i, j \in \{1, \dots, n_C\}$ and
  $\{C_1, \dots, C_{n_C}\} = \left\{C' \in \left\{C'_1, \dots, C'_{n_C'}\right\} \middle| |C'| \geqslant M_{CPG}\right\}$.
3 return $C_1, \dots, C_{n_C}$.
```

In this section, we have introduced a method to find CPGs in an SEM image. With this, we can accomplish the goal we have stated in Section 6.1. In the next section, we will find appropriate values for the parameters used in this chapter and evaluate our method to segment SEM images.

6.4 evaluation

In this chapter, we have defined goals for the segmentation of the images provided to our system. Then, we have looked at the solutions to related problems proposed in other publications. Based on these foundations, we have devised a method specifically targeted at SEM images containing engineered nanoparticles. As a last step, we will evaluate our approach. This serves two purposes:

- Find fixed values for the algorithm parameters.
- Determine how well our proposed system achieves the defined goals.

Our segmentation method contains several parameters that need to be specified before it is applied. In Chapter 3, however, we have stated that our system shall be fully automatic. This means that the user should not have to pick their values. Therefore, we need to find fixed values for the following parameters beforehand:

- The parameter $n_{\text{tile}} \in \mathbb{N}_{>0}$ determines into how many tiles the image is split for the noise estimation.
- The neighborhood radius $r_n \in \mathbb{N}_{>0}$ defines the size of the neighborhood which is compared to determine the weights in the non-local means noise removal algorithm.
- The search window radius $r_s \in \mathbb{N}_{>0}$ specifies the size of the part of the image that is averaged to compute the resulting intensity of a particular pixel in the non-local means noise removal algorithm.
- The filtering parameter $\eta \in \mathbb{R}_{>0}$ determines how strong the non-local means noise removal algorithm smoothes the image.
- The empty image threshold $\xi \in \mathbb{R}$ defines the maximum intensity of an image that is regarded as empty.
- The threshold offset $\delta \in \mathbb{R}$ is added to the output of the unimodal thresholding to obtain the final threshold.
- The minimum CPG size $M_{\text{CPG}} \in \mathbb{N}_{>0}$ defines the minimum size of of a connected component of the foreground to be regarded as a CPG.

We will evaluate the segmentation steps in their respective order beginning with noise estimation, which we will evaluate in the next section. Finally, in Section 6.4.5, we will evaluate the results of the segmentation pipeline as a whole.

6.4.1 *Noise Estimation*

To test the estimation of SEM noise parameters, some sort of ground truth is needed. For images that stem from a real SEM, the true parameters a and b are not available. Therefore, we have chosen to create

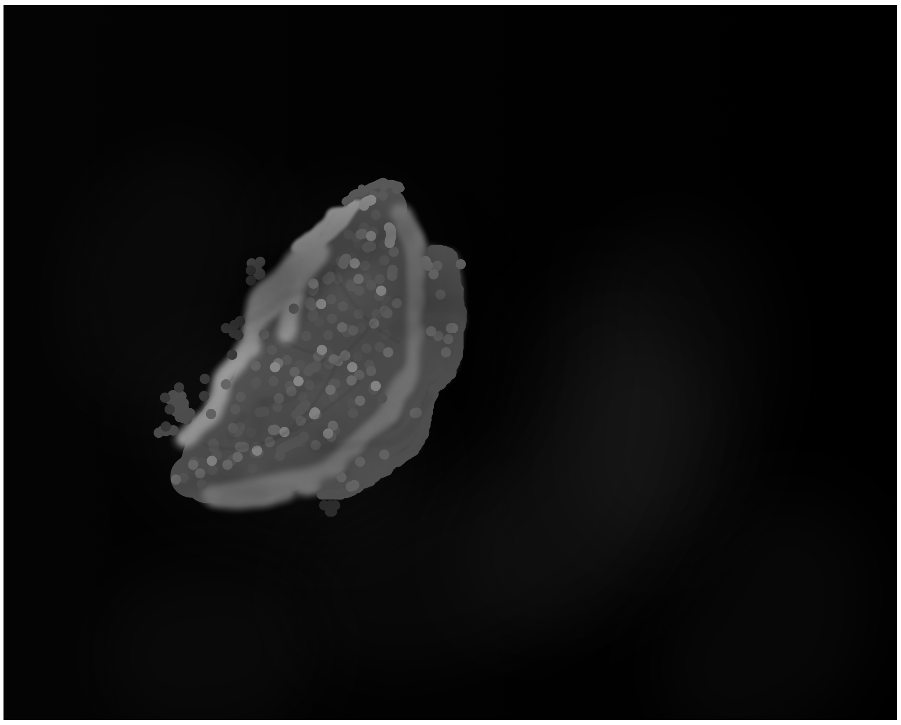


Figure 6.13: An artificial image without noise loosely based on the SEM image in Figure 6.14. It is used to test our noise estimation approach.

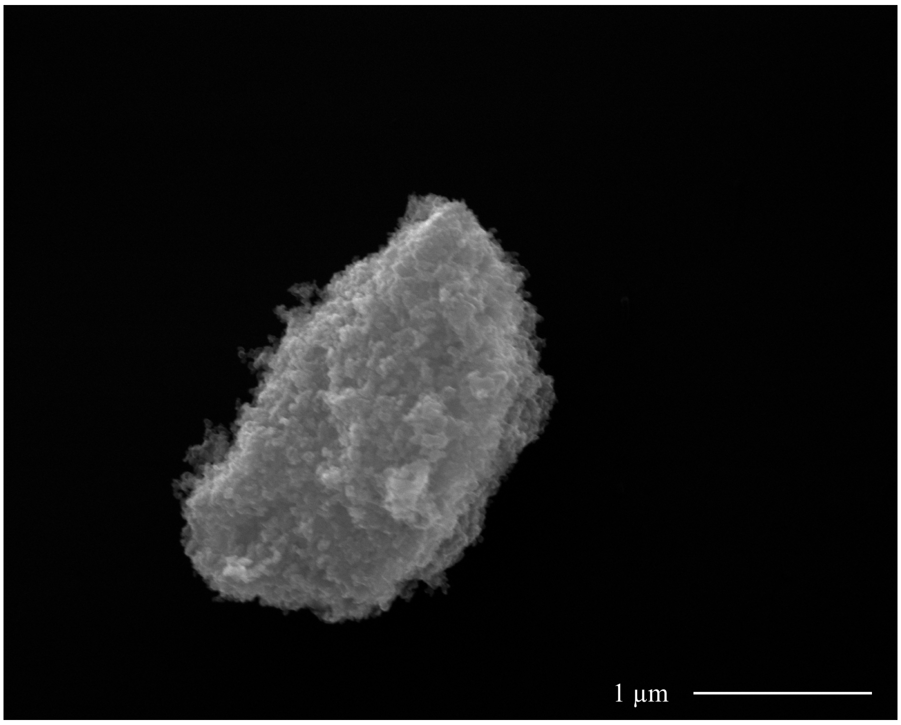


Figure 6.14: An SEM image (1 pixel = 1.27 nm) of an agglomerate of TiO_2 nanoparticles with an average diameter of 25 nm.

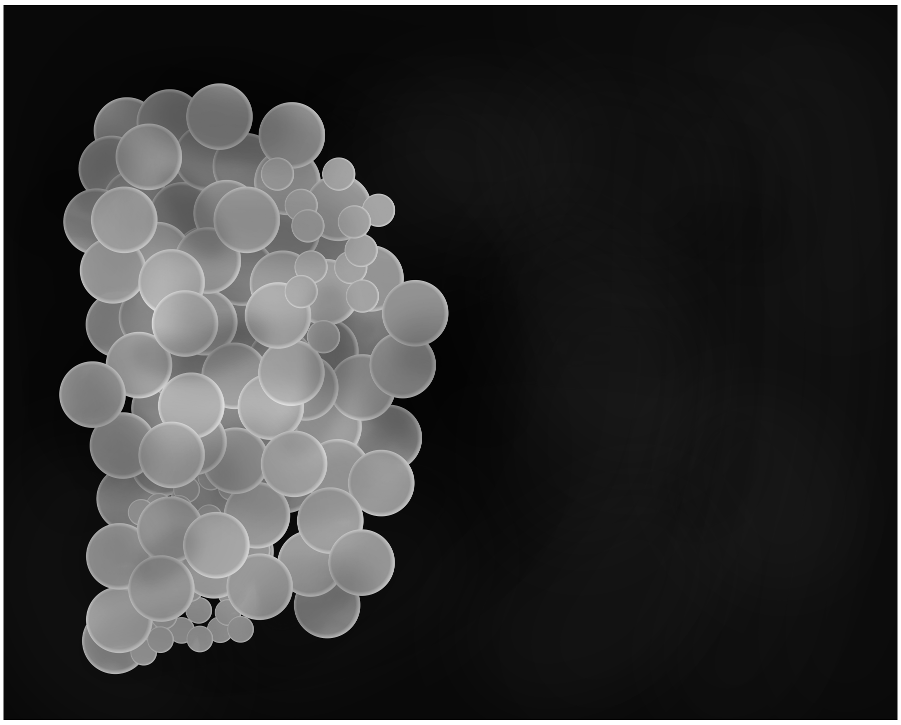


Figure 6.15: An artificial image without noise loosely based on the SEM image in Figure 6.16. It is used to test our noise estimation approach.

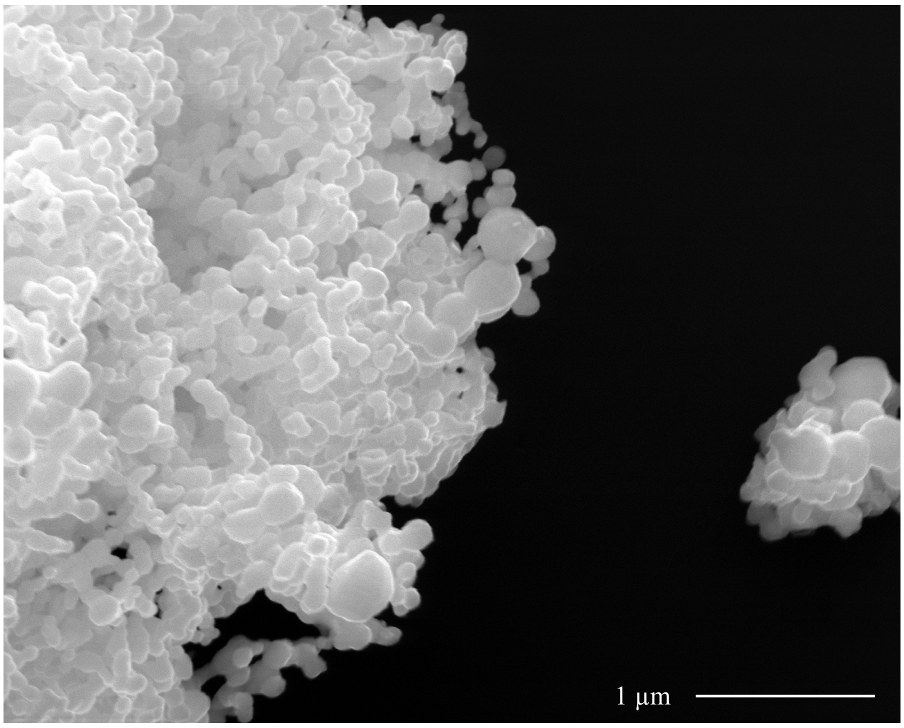


Figure 6.16: An SEM image (1 $\mathtt{pixel}$ = 1.27 $\mathtt{nm}$) of Ag nanoparticles with an average diameter of 75 $\mathtt{nm}$.

Table 6.1: The parameter values used to simulate Poisson noise to test our noise estimation method and the values of the parameter n_{tile} of our noise estimation method to be tested.

PARAMETER	USED PARAMETER VALUES
Gaussian standard deviation	2, 4 and 8
contrast setting a	$\frac{1}{8}$, $\frac{1}{4}$, $\frac{1}{2}$ and 1
brightness setting b	−20, −10 and −5
noise estimation parameter n_{tile}	1, 2, 4, 8, 16, 32, 64 and 128

artificial images, add SEM noise with varying known parameters a and b and test how well our method is able to estimate their values.

The artificial images used to test our approach can be seen in Figures 6.13 and 6.15. They contain structures imitating nanoparticle agglomerates. In addition, we added intensity gradients in the fore- and background to further simulate real images. However, in contrast to SEM images, the artificial ones contain no noise at all. This allows us to add noise with specific properties to test our noise estimation method. We do this as follows:

1. Choose values for a, b and the standard deviation of the Gaussian kernel used to smooth the simulated image.
2. Use the inverse of Equation (6.13) with the values of a and b to compute μ for every pixel of the artificial image. This can be done because the image is assumed to be optimal.
3. Apply a Gaussian Filter using the predetermined standard deviation to the resulting image in order to simulate blur introduced by the SEM.
4. Use the computed value of μ for each pixel to simulate a Poisson process.
5. Use Equation (6.12) with each pixel of the resulting image to compute the intensity from the electron count.
6. Convert the resulting image to 8-bit intensities.
7. Use our noise estimation method to obtain estimates $\hat{a}$ and $\hat{b}$.

Figure 6.17 shows the image in Figure 6.15 with added blur and artificial noise using the above procedure.

We have performed this procedure for every combination of the parameter values in Table 6.1. Because there is no ground truth, we did not know typical values for these parameters. Therefore, we have applied our noise estimation method to several images and tried to

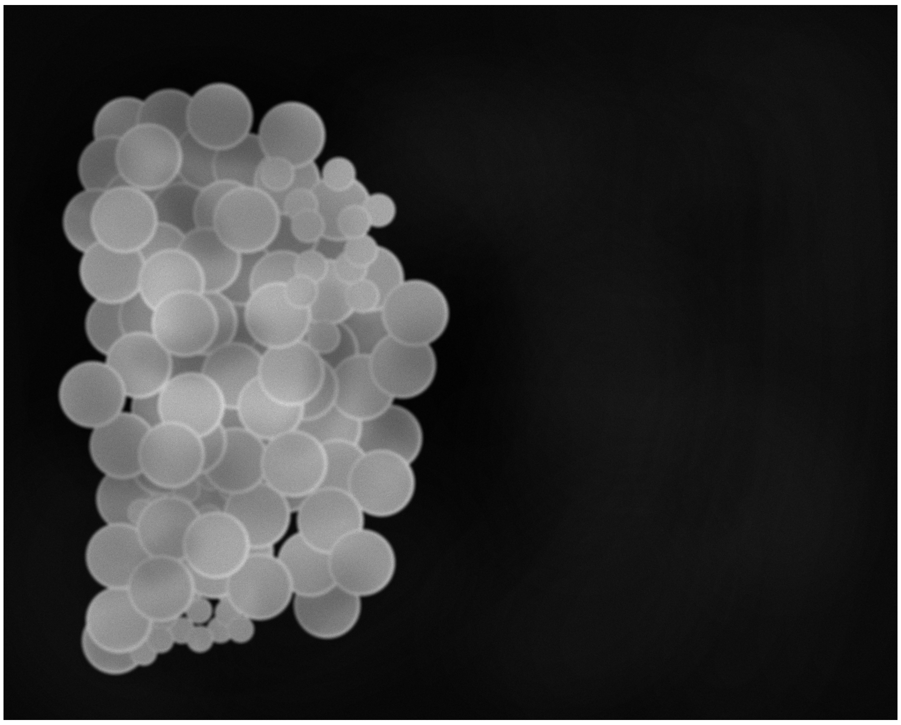


Figure 6.17: An image that has been obtained by blurring the artificial image in Figure 6.15 and adding artificial noise. A Gaussian standard deviation of 8, a contrast setting a of 1 and a brightness setting b of -5 have been used.

cover the resulting values of $\hat{a}$ and $\hat{b}$ with the range of parameter values to be used for the evaluation. In addition, we have chosen to scale the values exponentially in order to cover a wide range of possible values while also including small numbers. For the standard deviation of the Gaussian filter, we use relatively large values to match the high degree of blur of some of our images.

To obtain an error metric for a given n_{tile}, we use the following calculation:

1. For a given parameter combination and intensity $g \in G$, using the estimates $\hat{a}$ and $\hat{b}$ from our algorithm, we estimate the standard deviation of the noise to be expected for a pixel with the intensity g using Equation (6.19).

2. Then, we calculate the relative absolute error of this estimate compared to the true standard deviation of the noise calculated using the real values of a and b.

3. Finally, we average these relative absolute errors over all $g \in G$ and all parameter combinations of Gaussian standard deviation, contrast setting a and brightness setting b.

Doing this, we find that, for both images, all noise estimation parameter values n_{tile} from 1 to 32 produce stable results with errors between

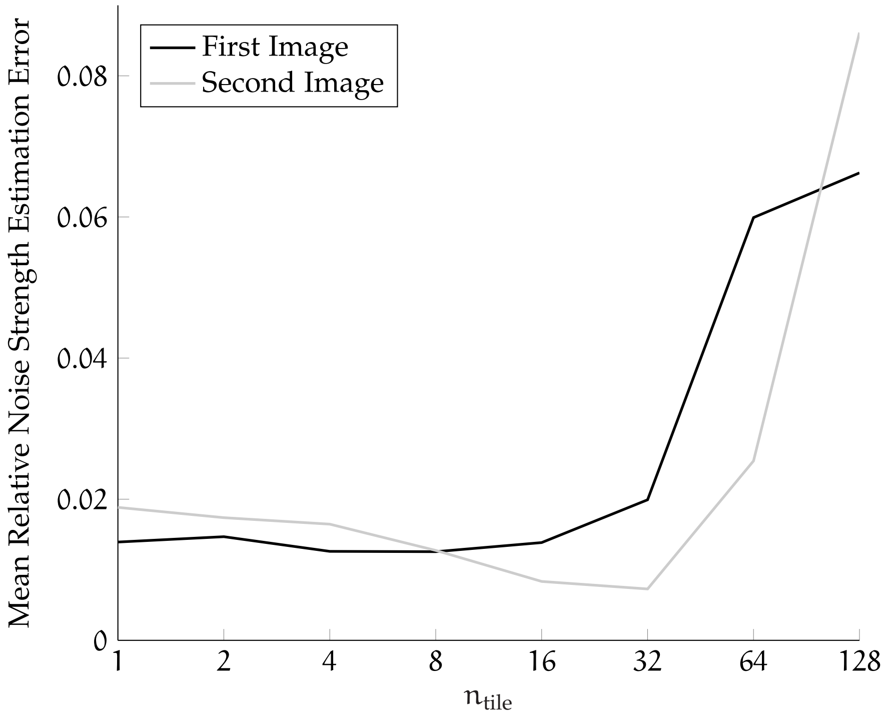


Figure 6.18: The mean relative error of the noise standard deviation estimation by our method for different values of n_{tile} measured using two artificial images. The image in Figure 6.13 is referred to as the first image and the one in Figure 6.15 as the second.

0.007 and 0.02. This means that our algorithm's estimates of the noise standard deviation differ on average less than 2 % from the true value. For the image in Figure 6.13, $n_{\text{tile}} = 8$ produces the lowest error of 0.013 and for the image in Figure 6.15, $n_{\text{tile}} = 32$ yields an error of 0.007. For higher values of n_{tile}, the error grows up to a value of 0.086. A graph of the errors for both images can be seen in Figure 6.18.

These results tells us the following:

- The algorithm is stable for a wide range of values of n_{tile} from 1 to 32 without the parameter being carefully chosen.
- The error is very small for most parameter values and even in the worst case, a relative error of 0.086 is still acceptable.

We have decided to use a fixed value of $n_{\text{tile}} = 16$ in our system because it produced the lowest error of 0.011 averaged between both test images. We will also use this value for the remaining experiments in this thesis.

6.4.2 *Noise Removal*

The noise removal method by Buades et al. [2005b] has three parameters (see Section 6.3.2.2):

- The neighborhood radius r_n.
- The search window radius r_s.
- The filtering parameter η.

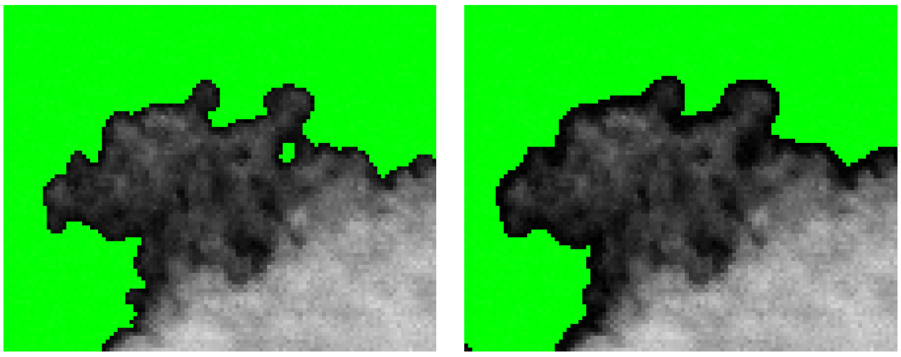
The choice of the first two parameters is mainly a trade-off between quality and execution time. For larger parameters, the runtime of the algorithm grows and the results become slightly better. We have decided to adopt the values given by the authors in the original paper as they yield a good balance between quality and time. Therefore, for the rest of this thesis and in our system, we use a neighborhood radius of $r_n = 3$ and a search window radius of $r_s = 10$.

The filtering parameter η determines how strong the algorithm smooths the image. The larger the value, the smoother the image becomes. We have not adopted the value used by Buades et al. [2005b] because our setup is different. Firstly, because we use a variance-stabilizing transformation, the images used as input to the noise removal method have a fixed noise variance of 1. Secondly, the resulting images are only used to apply a thresholding. This means that it is acceptable for a filtered image to be very smooth and lose details inside the particles as long as the borders between image background and particles is unaffected.

To find a good value for η, we have chosen three images of each of the three nanoparticle types. Among these nine images there are some that pose a problem to our thresholding algorithm because parts of the image background have a high intensity. We have found an appropriate parameter value of η using the following approach:

1. For each of these images, apply noise estimation and removal as described in Section 6.3 starting with a low value of η.
2. Manually choose an optimal threshold for each image so that as much particle area as possible but no large parts of the background are recognized as foreground.
3. If a particle in any image contains small holes in the segmentation resulting from the manual thresholding, repeat the process with a higher value of η.
4. If no particle contains any small holes, use the current value of η.

Using this approach to find a value for η, we ensure that the found CPGs do not contain small holes. Additionally, we avoid that the images are smoothed too much. As with the noise estimation parameters, we have chosen to exponentially scale the parameter values to be tested. Therefore, we have started with a value of $\eta = \frac{1}{2}$ and multiplied the current value with $\sqrt{2}$ to obtain the next value. Using this


(a) Segmentation after removing noise using the method proposed in this chapter.

(b) Segmentation after removing noise using a Gaussian filter.

Figure 6.19: Two segmentations of an SEM image detail ($1\,\mathrm{pixel} = 5.08\,\mathrm{nm}$) of ZnO nanoparticles after noise removal using the proposed method and a Gaussian filter, respectively. The segmentation of the background is represented by green pixels.

approach, we have found that $\eta = 2$ is a good choice. We use this value in our system and in all following experiments.

Figure 6.19 shows a comparison of two segmentations of a ZnO nanoparticle agglomerate. In the left case, the noise removal method proposed in this chapter is used. The right image is the result of instead using a Gaussian filter as the noise removal method before segmenting the image. The green pixels in each image show the area which has been segmented as image background. The segmentation using our method is much more precise than using a Gaussian filter. The segmentation border in the latter case is a few pixels further from the particle border. In addition, the small hole in the agglomerate near the center of the image has not been detected.

In addition, we wanted to test how our noise removal pipeline performs compared to a conventional method. For that, we have chosen non-local means because it is a state-of-the-art method and it allows us to directly quantify the influence of our noise estimation approach. We have opted not to use a variance-stabilizing transformation for the comparison approach because in order to use such a method the noise parameters have to be known. This is usually not the case without a noise estimation method. The test we have performed is very similar to the experiment for finding the best noise estimation parameter n_{tile}, which is explained in Section 6.4.1. It has the following steps:

1. For each of the two artificial images in Figures 6.13 and 6.15, do:

    a) For each combination of the values for a, b and the standard deviation of the Gaussian kernel in Table 6.1 on Page 93, do:

i. Use the inverse of Equation (6.13) with the values of a and b to compute μ for every pixel of the artificial image. This can be done because the artificially created images contain no noise.

ii. Apply a Gaussian Filter using the predetermined standard deviation to the resulting image in order to simulate blur introduced by the SEM.

iii. Use the computed value of μ for each pixel to simulate a Poisson process.

iv. Use Equation (6.12) with each pixel of the resulting image to compute the intensity from the electron count.

v. Convert the resulting image to 8-bit intensities.

vi. Apply our noise removal pipeline as well as simple non-local means to the resulting image.

vii. For each approach, compute the root-mean-square error of the pixel differences between the blurred image of Step 1(a)ii and the filtered image resulting from Step 1(a)vi.

2. For each approach, compute the average of the root-mean-square errors resulting from Step 1(a)vii.

Note that we do not convert the filtered images from Step 1(a)vi to 8-bit intensities because this could pose a source of error for the results. In addition, we compare the resulting images to the blurred ones instead of the unaltered test images because the noise removal methods are not designed for removing blur.

The resulting average root-mean-square errors are shown in Figure 6.20. As a baseline, the average error of the unfiltered images from Step 1(a)v is also shown. The average error of simple non-local means (1.657) is almost three times as high as the one of our method (0.585). Moreover, our approach has outperformed simple non-local means for every combination of test image and parameters in our experiments.

6.4.3 *Threshold Determination*

To find the threshold offset δ, we use the nine images and the manually chosen thresholds from the noise removal evaluation as described in Section 6.4.2. We apply the thresholding algorithm by Zack et al. [1977] and the method called minimum error thresholding by Kittler and Illingworth [1986] to the filtered images and compare the found thresholds to the manual thresholds. By computing the mean error and the error variance of both algorithms compared to the manual values, we can see which performs better and how large δ has to be.

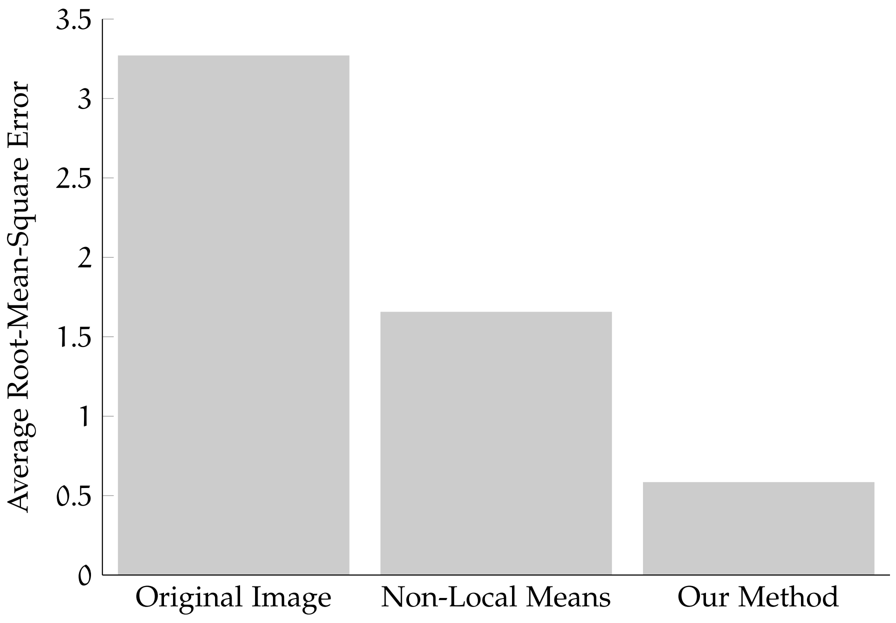


Figure 6.20: The average root-mean-square errors of non-local means (1.657) and our noise removal method (0.585) in our experiment. As a comparison, the average error of the unfiltered noisy images (3.271) is also shown.

The average error of the method by Zack et al. [1977] is −1.17, while the one by minimum error thresholding is −2.11. This means that both algorithms return values that are on average slightly below the ground truth consisting of the manual thresholds while the first yields a better result. However, the error variance paints a much clearer picture. For the algorithm by Zack et al. [1977], it is equal to 2.5, while the approach by Kittler and Illingworth [1986] has an error variance of 7.86. This means that the first one is much more precise at predicting the best possible threshold. In fact, for six of the nine images, the error of Zack's method is either −0.5 or 0.5. This is the best possible result because the algorithm always returns a single intensity while the manually chosen thresholds are defined as $g + 0.5$ for $g \in G$ so that they always lie between two intensities. The three images for which the absolute error of the method by Zack et al. [1977] is higher than 0.5 are exactly the ones which we have added to the test set because they pose a challenge to thresholding algorithms. For these, the error is between −4.5 and −3.5. Minimum error thresholding's best result has an error of −1.5, while its largest error is −22.5. Such a high error would presumably make any further processing of the image impossible.

Due to these results, we have decided to use the thresholding approach by Zack et al. [1977] and a threshold offset of $\delta = 1.17$ in our system and all remaining experiments. This value coincides with the mean negative error achieved in our experiment in order to counteract the average error.

6.4.4 *CPG Search*

The empty image threshold ξ determines if an image is considered empty by our system. If the highest intensity of an image is above the threshold, it is considered not empty and vice versa. The highest intensity is determined after our noise removal method has been applied.

The highest intensity in any empty image in our dataset is 21. On the other hand, the brightest pixel in the image with the lowest peak intensity that contains particles has a value of 34. Therefore, we have chosen to set $\xi = 27.5$ for our system, which is exactly in the middle of these two values.

The parameter M_{CPG} determines the minimum size of a CPG in pixels. This value depends on the size of the nanoparticles and the pixel size in nm^2. However, for smaller particles, usually a smaller pixel size is chosen by the user. Therefore, we have decided to take the particles and image magnification we use for this thesis as a basis. The smallest engineered particles we use in our experiments are ZnO nanoparticles with a diameter of $10\,nm$. In one of our images with a magnification of 20 000, such a particle has a size of approximately 48.7 pixels. We have chosen to use 90 % of that, which corresponds to 43.8 pixels, as the value of M_{CPG}. This ensures that even a single nanoparticle can be found at a magnification of 20 000.

6.4.5 *Full Segmentation Pipeline*

Fully evaluating our segmentation pipeline is difficult because of the lack of ground truth. However, while one cannot objectively determine if the algorithm correctly segmented an agglomerate, we are able to objectively determine if it has detected the CPG at all. In addition, we can subjectively assess how well the segmentation result is. Using a random number generator, we have chosen two images of each type of engineered nanoparticles. For these, we have assessed how many CPGs the pipeline detects and how well the resulting segmentations are.

Table 6.2 lists the results of our quantitative evaluation. In the six assessed images, we have manually found 73 CPGs containing engineered nanoparticles. Our segmentation pipeline finds all of these, as well. In fact, it finds 7 additional CPGs containing engineered nanoparticles, which we have overlooked while searching the images manually. All of these are recognized as a whole and none are split into multiple CPGs. Apart from these CPGs, the images also contain gas deposits as described in Section 4.2.2.5 and some background

Table 6.2: The number of CPGs of engineered nanoparticles we have found manually compared to the number our pipeline detects automatically.

PARTICLE TYPE	MANUALLY	AUTOMATICALLY
Ag	12	12
TiO_2	29	30
ZnO	32	38

particles accidentally introduced onto the silicon substrate. We have manually found 11 background CPGs, all in the same image containing Ag nanoparticles. Our pipeline finds 6 of these, while it splits two of them into two and three parts, respectively. The other images contain 2 gas deposits, which we have overlooked. In contrast, our algorithm finds both of them.

Apart from the CPGs present in the images, the algorithm also marks segments at the left edge of an image containing TiO_2 nanoparticles. This is caused by the fact that the leftmost columns appear brighter than the rest of the background. This effect has been explained in Section 4.2.2.5. These extra segments do not impair the segmentation because we can train the classification step of our system to label them as not being engineered nanoparticles.

All in all, we have manually found 84 CPGs while our pipeline detects 88 CPGs, of which it splits two into multiple parts. Our method overlooks 5 CPGs. All of them are in the same image and each consists of a single small background particle. This has two reasons:

- These particles are very small and the image has a relatively low magnification of 5000. Therefore, each of them occupies relatively few pixels.
- The particles have very low intensities. In addition, the images contain a very large Ag agglomerate having a very high intensity due to its material properties (see Section 4.2.2.4). This has two implications: Firstly, in order to avoid intensity clipping, the microscope operator has adjusted the brightness and contrast settings so that these small particles appear very dark. Secondly, the large agglomerate causes the background on the upper left of it to appear brighter than normal. This effect has been explained in Section 4.2.2.6. It leads to the fact that these small particles are darker than the background next to the Ag agglomerate.

The first point tells us that while this magnification is enough to detect engineered Ag nanoparticles, if we wanted to analyze particles like the ones overlooked, we would need to choose a higher magnification resulting in a better resolution and a smaller pixel size.

The second issue is more difficult to solve. If parts of the background have a higher intensity than a particle, this issue is inherent to thresholding. However, we have made the point in Section 6.2 and at the beginning of Section 6.3 that the other approaches in the related work are not suitable for the types of images our system has to deal with. One could argue that dynamic thresholding could be a possible candidate. However, our database contains images where over 90 % of the area is occupied by a single agglomerate. Our approach segments this image very well. We would argue that dynamic thresholding would not be able to handle such an image correctly, because the local threshold for some parts of the image would only be determined by intensities belonging to the agglomerate. This would lead to some parts of the agglomerate being segmented as image background.

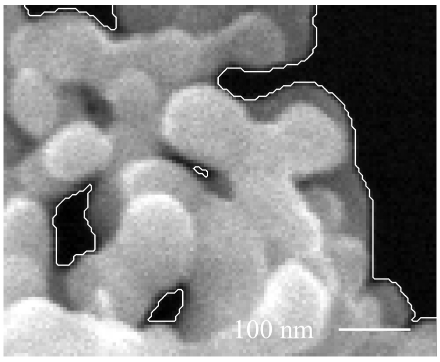
While criticism of this result is certainly justifiable, we want to argue that our pipeline delivers results superior to manual detection. For the randomly chosen images, our approach not only detects more particles overall than we have been able to find manually. In addition, the pipeline is able to find every single CPG containing engineered nanoparticles in these images in contrast to the manual search, which has missed about 9 % of them.

Still, one could argue that a case like that of the particles that have not been detected by our pipeline may also occur in an instance where small engineered nanoparticles are overlooked. This may be possible but we believe that such a case is unlikely. For this to happen, a large background particle that appears very bright would also need to be present in the image. However, we have never observed any kind of background particle that appears similar to these Ag nanoparticles. This is due to the properties of Ag. It is very unlikely that background particles made of Ag appear in an image of other engineered nanoparticles.

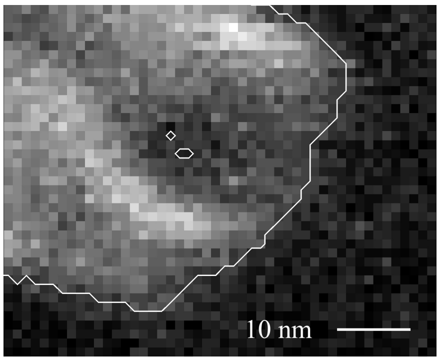
In addition, for the analysis of smaller engineered nanoparticles, one would choose a higher magnification. This would result in a smaller visible area of the sample. Due to this smaller area, the intensity differences between different parts of the image would not be so high. Therefore, it would be less likely that a small particle is darker than the image background in other parts of the image.

One additional advantage of our solution over manual detection is that it yields information about the size and shape of the detected CPGs. This data can be used to estimate CPG volume, mass and nanoparticle count in addition to the CPG count. In order to use this advantage, the segmentation itself should be good. We will assess its quality in the following paragraphs.

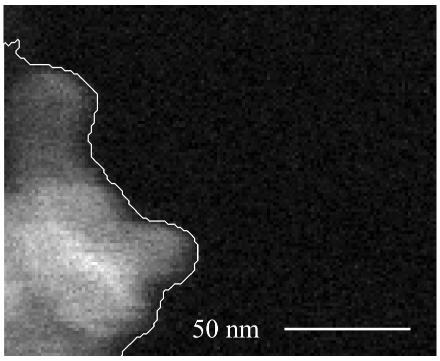
When assessing the results of our experiment, we find that our pipeline makes no major errors in the segmentation of the 80 CPGs



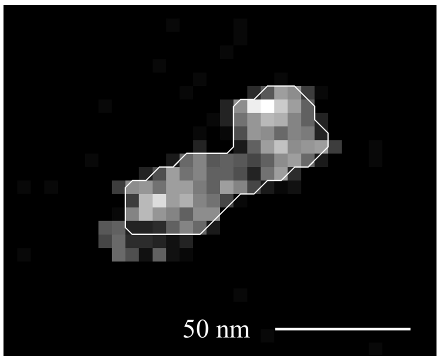
(a) One of the holes in this Ag nanoparticle agglomerate is not correctly segmented (1 pixel = 5.08 nm).



(b) Small holes in the segmentation of a TiO_2 agglomerate (1 pixel = 1.27 nm).



(c) The segmentation of this TiO_2 agglomerate contains a small part of the background in the upper left corner due to an SEM artifact (1 pixel = 1.27 nm).



(d) This ZnO agglomerate is not fully segmented (1 pixel = 5.08 nm).

Figure 6.21: SEM image details containing segmentation errors. The segmentation is represented by a white line.

containing engineered nanoparticles in the randomly chosen images. The borders of the resulting segments match very well with the outlines of the CPGs and the algorithm also detects holes in them. In addition, no large parts have been added to or removed from any CPG. We find only minor imperfections in these segmentations. These are the following:

- Small holes inside the CPGs are sometimes smaller in the segmentation than they are in the the image. An example of this can be seen in Figure 6.21a, where the hole in the center of the image is not correctly segmented. The impact of this kind of error is usually negligible because small holes only have a minor influence on the complete segmentation of a CPG.

- Despite the fact that we apply a noise removal method before thresholding, in a few cases, the segmentation contains small holes of only one or two pixels. An instance of this is visible in

Figure 6.21b. Similar to the previous issue, the impact of this should be small.

- In one case, the segmentation of a TiO_2 agglomerate located at the left edge of the image contains a small part of the image background as can be seen in Figure 6.21c. This is due to an SEM artifact where the leftmost pixel columns appear brighter than the rest of the background as described in Section 4.2.2.5. This part represents only a very small percentage of the agglomerate's segmentation.

- Finally, in four cases, small parts of an agglomerate are not captured by the segmentation. An example of this can be seen in Figure 6.21d where some pixels in the lower left are not part of the segmentation.

The last two issue might have an impact on every calculation involving the sizes of CPGs. However, these segmentation errors are mostly minor and only 5 CPGs are affected, representing about 6 % of all CPGs. In all other cases, the sizes are measured correctly.

Taking all these facts into consideration, the pipeline presented in this chapter does a very good job at detecting and segmenting engineered nanoparticles in SEM images. This is especially true when compared to performing this task manually, which is tedious, error-prone and not able to produce a detailed segmentation.

In this chapter, we have developed a pipeline specifically targeted at detecting and segmenting engineered nanoparticles in SEM images. We have developed and selected various algorithms in order to match the unique properties of these images. In the next chapter, we will characterize the properties differentiating CPGs containing engineered nanoparticles from those consisting of background particles and present numerical features capturing these characteristics in order to perform a successful classification.

7 FEATURE COMPUTATION

This chapter will address the task of computing numerical features for each CPG found by the segmentation pipeline presented in the last chapter. The goal is to capture the characteristics differentiating engineered nanoparticles from background particles. In Section 7.1, we will give a detailed description of the task to be performed and the particle characteristics that the features shall capture. Section 7.2 will give an overview of the features used in the literature on automatic image-based particle analysis. Finally, in Section 7.3, we will describe the features used by our system.

This chapter does not contain an evaluation. Instead, the features used by our system will be evaluated in the next chapter. The reason for this is that we will assess their quality by the impact they have on the classification result. In order to do this, we will first need to introduce our classification pipeline, which we will also do in the next chapter.

7.1 GOALS

The segmentation pipeline of our system outputs a list of CPGs, which we have defined as $C_1, \dots, C_{n_C} \subseteq P$ in Section 6.1, for each image. Each of these represents the position and shape of a group of connected particles visible in the SEM image. The goal of the system is to decide for each CPG into which of these two classes it belongs:

- CPGs consisting of engineered nanoparticles.
- CPGs consisting of background particles.

In order to do that, the first step is to capture the characteristics of a CPG into a fixed set of numerical values. These are called feature values and are produced by functions called features. Each feature produces a feature value for each CPG.

In order to allow for a successful classification, the following conditions should be fulfilled:

- Similar CPGs should have similar feature values. As similarity is always subjective, each feature focuses on a specific attribute of a CPG.
- Different classes may have similar feature values. For a successful classification, however, it should be possible to differentiate two classes based solely on a combination of feature values of their members.

- The CPGs should be described by as little features as possible. This may seem counter-intuitive at first as more information about a CPG allows for a better informed decision. However, given a fixed number of CPGs, an increased number of features may lead to random patterns in the feature values that seem to differentiate the classes well. However, when looking at a different set of CPGs, these patterns are not at all able to tell the classes apart. When having too many features, finding those that allow for a successful classification is difficult. As few features as possible should be selected. We will give a more detailed explanation of this so-called curse of dimensionality in Section 8.1.2.

In a more formal way, a feature ϕ is defined as a function

$$\phi : \mathcal{C} \times G^P \times \mathbb{R}_{>0} \to \mathbb{R}. \tag{7.1}$$

Here G^P is the set of all images $f : P \to G$. Using this definition, if $C \subseteq \mathcal{C}$ is a CPG, $f : P \to G$ is the corresponding image and $s_p \in \mathbb{R}_{>0}$ is its pixel size in nm, then $\phi(C, f, s_p) \in \mathbb{R}$ is the feature value of feature ϕ for the CPG C. For brevity, we will define the set of all features Φ:

$$\Phi := \mathbb{R}^{\mathcal{C} \times G^P \times \mathbb{R}_{>0}}, \tag{7.2}$$

so that $\phi \in \Phi$.

If $\phi_1, \ldots, \phi_{n_\phi} \in \Phi$ are the features used to describe all particles, then $\left(\phi_1(C, f, s_p), \ldots, \phi_{n_\phi}(C, f, s_p)\right) \in \mathcal{X} := \mathbb{R}^{n_\phi}$ is the so-called feature vector of the CPG C. Sometimes, a group of features belonging together is also called a feature. In this case, a feature yields multiple feature values instead of only one. However, we can split such a feature into multiple "sub-features" that fit with our definition.

7.2 RELATED WORK

We have again surveyed the 33 publications on automatic image-based particle analysis first introduced in Section 5.1. The features used in these publications can be grouped into four categories:

- External information
- Basic geometric features
- Shape features
- Texture features

Features based on external information rely on any knowledge from sources other than the image itself. The only feature of this kind mentioned in the papers we investigated was the chemical composition of the particles [Germani and Buseck, 1991; Kindratenko et al., 1994, 1996; Wienke et al., 1995; Oleshko et al., 1996; Hopke and Song, 1997; Genga et al., 2012]. That feature would be very helpful for our purpose as engineered nanoparticles are made of metals while environmental particles are usually composed of non-metals such as carbon. However, as mentioned in Section 4.2.1, it is not feasible for the microscope operator to probe every particle using EDX. As explained in Section 5.1, using a CCSEM to obtain EDX data is also not an option because it cannot handle particles with diameters smaller than 50 nm to 100 nm. Therefore, we cannot use the chemical composition as a feature.

Basic geometric features only depend on the shape of a CPG in the image. They produce a single feature value and in most cases, their computational complexity is low. Often, they are calculated as ratios of two geometric measures such as area, diameter or perimeter.

There is no clear dividing line between basic geometric features and shape features. The latter are based on the shape of the CPG, as well. However, they often require more involved computations and produce multiple feature values.

Texture features examine the intensity distribution inside the CPG's area. They may be sensitive to particular patterns such as circles. Others calculate specific statistical properties such as the frequency of intensity combinations at specific offsets. Most texture features generate multiple feature values.

In the following sections, we will describe image-based features used in the literature we have surveyed. However, as there are many different features, we will only present the ones which are used by more than one research group or seem important to us.

7.2.1 *Basic Geometric Features*

Basic geometric features can be further categorized into two groups: size-dependent measures and scale-invariant measures. The difference is that the latter do not change their value as the particle changes its size without altering its shape. In the remainder of this section, we will discuss a selection of these measures.

7.2.1.1 *Size-dependent Features*

The most used feature is the area a particle occupies in the image [Germani and Buseck, 1991; Xu et al., 1998; Peng and Kirk, 1999; Laghari, 2003; Luo et al., 2004; Rodriguez-Damian et al., 2006; Stachowiak et al., 2008; Genga et al., 2012]. This can be a useful feature for our system, as well. For example, agglomerates made of large engineered nanoparticles are on average larger than diesel soot agglomerates.

Another popular measure is the diameter. However, there are multiple definitions for it. A Feret diameter is the "distance between two parallel tangents on opposite sides of the image" of the particle [Merkus, 2009, p. 15]. In the publications using automatic classification, only the maximum Feret diameter is used [Xu et al., 1998; Peng and Kirk, 1999; Rodriguez-Damian et al., 2006]. A Martin diameter is the length of a line which divides the area of the particle image into two equal halves [Merkus, 2009, p. 15]. Germani and Buseck [1991] use the minimum, maximum and average Martin diameter. Fisker et al. [2000] calculate the geometric means of the minor and major diameters of ellipses fitted to images of spherical particles. Ap Gwynn and Wilson [2001] use the maximum diameter but do not specify which kind of diameter is used.

We find that a diameter feature is not appropriate for our system. Two agglomerates made of the same type of engineered nanoparticles can have very different global—as opposed to local—shapes. One can be very elongated while the other is relatively round. A diameter would therefore not help to capture the characteristics of nanoparticle agglomerates. In addition, a random diameter is not very meaningful while a minimum, maximum or average diameter requires much computational effort.

The last size-dependent measure is the perimeter [Germani and Buseck, 1991; Stachowiak et al., 2008; Genga et al., 2012]. It is defined as the length of the contour of the particle. We adopt this feature in our system because it can help to distinguish CPGs with jagged edges such as nanoparticle agglomerates from ones with smooth borders. Moreover, it is relatively easy to compute.

7.2.1.2 *Scale-invariant Features*

Scale-invariant features do not change their value when the scale of the particle is changed. They are also called *shape factors*. The naming of the individual features is very inconsistent so that there are different names for the same measure while there are also distinct features with the same name.

The most popular class of scale-invariant features we have found compares the shape of the particle with that of a circle. The most common way to do this is to compute the ratio of the particle area and the area of a circle which has the same perimeter as the particle. Its formula is $\frac{4\,\pi\,\text{area}}{\text{perimeter}^2}$. This ratio or variants multiplied with a constant value are called circularity [ap Gwynn and Wilson, 2001; Greco and Maffezzoli, 2004], isoperimetric quotient [Oster, 2010] or roundness [Rodriguez-Damian et al., 2006] in the literature. Its inverse is called compactness [Rodriguez-Damian et al., 2006], circularity or roundness factor [Raadnui, 2005]. A similar measure which is calculated using the formula $\frac{4\,\text{area}}{\pi\,(\text{maximum Feret diameter})^2}$ is called roundness by Peng and Kirk [1999] and Greco and Maffezzoli [2004]. In addition, features with the names compactness [Stachowiak et al., 2008], roundness [Genga et al., 2012] and roundness factor [Laghari, 2003] are used without being defined. Similar to the perimeter, this feature can be of use in our system to distinguish CPGs with jagged edges such as nanoparticle agglomerates from those with smooth borders.

Several publications use a feature called aspect ratio. However, their definitions vary. Fisker et al. [2000] define it as the ratio of major and minor diameter of an ellipse fitted to the particle. Greco and Maffezzoli [2004] call their feature aspect ratio or elongation and define it as the ratio of maximum to minimum Feret diameter. Raadnui [2005] employs a shape factor also called aspect ratio and defines it as the ratio of width to length of the minimum enclosing rectangle. Laghari [2003], Stachowiak et al. [2008] and Genga et al. [2012] use measures with the same name but do not give a definition. We do not employ the aspect ratio as a feature due to the same reasons we mentioned in the case of the diameter.

7.2.2 *Shape Features*

The fractal dimension can be seen as either a basic geometric or as a shape feature. It is one-dimensional and relatively complex to compute. Still, it is very popular [Orford and Whalley, 1983; Kindratenko et al., 1994, 1996; Oleshko et al., 1996; Peng and Kirk, 1999; ap Gwynn and Wilson, 2001; Greco and Maffezzoli, 2004; Raadnui, 2005; Stachowiak et al., 2008; Genga et al., 2012]. The idea is that a contour can have self-similarity at different scales and thus have a dimension between 1 and 2. Therefore, it is called fractal dimension. Its value can be obtained by measuring the perimeter of the particle with measuring sticks of different sizes. Using a large measuring stick, the measured length of the contour will be shorter. Using a more detailed representation of the contour, however, its measured length will be longer. If the contour is fractal, the measured perimeter is proportional to the length of the stick to the power of one minus the fractal dimension [Kindratenko et al., 1996]. However, real objects are not fractal as the self-similarity is only present in a limited size range [Kindratenko et al., 1996].

The computation of this measure is complex and the feature is suitable only for irregular shapes. In addition, a shape has only one fractal dimension so that it is not able to successfully discriminate complex shapes [Drolon et al., 2000]. Kindratenko et al. [1997] write that the fractal dimension attempts "to condense all the details of the shape into a single number. There can, however, be an unlimited number of visually different shapes with the same fractal dimension." These facts limit the applicability of this feature. The most important reason, however, for not using the fractal dimension is that it is scale-invariant. Therefore, it cannot give any information about the size of contour features. For nanoparticle agglomerates, though, the size of the nanoparticles is reflected in the contour and it is a crucial factor for distinguishing them from other types of particles.

Another shape feature that is used in many publications are the so-called Fourier descriptors. They aim at capturing the raggedness of the particle contour at different scales and are calculated as follows:

1. Express the contour of the particle as a real- or complex-valued periodic function.
2. Apply the Fourier transform to this function.
3. Alter the Fourier coefficients to ensure translation-, rotation- and (optionally) scale-invariance.
4. Take a number of the resulting coefficients and use them as feature values.

The third step is done by removing the zeroth coefficient, taking only the magnitude and dividing all coefficients by the first one. When applying this type of feature, one has to decide how to express the particle contour as a function. There are several possibilities to do so:

- angle $\mapsto$ centroid distance
- contour length $\mapsto$ centroid distance
- contour length $\mapsto$ curvature
- contour length $\mapsto$ coordinates expressed as complex number

In the first variant, the position of every contour point is expressed in polar coordinates using the particle centroid as the origin of the coordinate system. The function that is used for the Fourier transform maps the angle to the distance of these coordinates [Kindratenko et al., 1996, 1997; Raadnui, 2005]. However, only for convex particles, this mapping is guaranteed to be unique. Therefore, it is not applicable to complex shapes.

The functions used in the three remaining options map the contour length from the starting point to the contour point in question to different measures of that point. For the second variant, the distance of the point and the centroid is used [Yu et al., 2007] while the third option uses the point's curvature [Xu et al., 1998]. The last and most popular variant, however, interprets the Cartesian coordinates of the point as a complex number where the real part is the x-coordinate and the imaginary part is the y-coordinate of the point [Thomas et al., 1995; Kindratenko and Van Espen, 1996; Greco and Maffezzoli, 2004; Calderon De Anda et al., 2005; Rodriguez-Damian et al., 2006].

Kindratenko et al. [1996] state that "Fourier description is very attractive for particles with regular shapes, especially for those with axes of symmetry." That means that it is well suited to detect particles whose shapes are recurring. Nanoparticle agglomerates, however, can have very different shapes and two of them will almost never look the same. To identify these agglomerates, it is necessary to examine them at exact predetermined scales to detect nanoparticles of specific sizes. Fourier descriptors, however, are in general not able to examine two particles using the same scales. This is because the wavelengths used to analyze a particle are always equal to its contour length divided by a natural number. Therefore, these wavelengths are in general not the same for two different particles.

Drolon et al. [2000] propose another approach called multiscale roughness descriptor, which is also used by Greco and Maffezzoli [2004]. In it, the contour of the particle is represented as a function where the contour length maps to the Cartesian coordinates of the contour point interpreted as a complex number. (See the discussion of Fourier descriptors above.) Then, a modified version of the harmonic wavelet transform by Newland [1993] is applied to that function. Drolon et al. [2000] replace the Fourier coefficients used in the computation of the wavelet transform by shift-invariant versions. This makes the coefficients of the wavelet transform translation-, rotation- and scale-invariant. As a last step, the energies of these coefficients are calculated to obtain the final feature values.

The multiscale roughness descriptor, as the name suggests, is able to represent the roughness of the particle's shape at different scales. However, the resulting feature values are scale-invariant. As mentioned before, in our case, the exact size of the primary particles constituting an agglomerate is crucial to distinguish different types of particles. Therefore, for a feature to be used in our workflow, it has to examine every particle at exactly the same scales.

7.2.3 *Texture Features*

The least popular feature category in the literature are texture features. Only one is used in more than one publication. The so-called Haralick features [Langford et al., 1990; Flores et al., 2003; Laghari, 2003; Rodriguez-Damian et al., 2006; Stachowiak et al., 2008] are based on the statistics of gray-level combinations at specific pixel offsets. An advantage of this approach is that it can be applied to large particles as well as very small ones. Therefore, we use this feature in our system. An explanation of the method is given in Section 7.3.3.2.

Oster [2010] applies band-pass filters to the SEM images in order to detect frequencies corresponding to structures of a specific scale in the particle textures. This feature may be useful in order to find the signatures of nanoparticles of a specific size in the texture of agglomerates. However, it cannot be applied to CPGs of similar or smaller size compared to the filter's wavelengths because the image background and particle edges influence the filter responses of pixels near the border of a CPG.

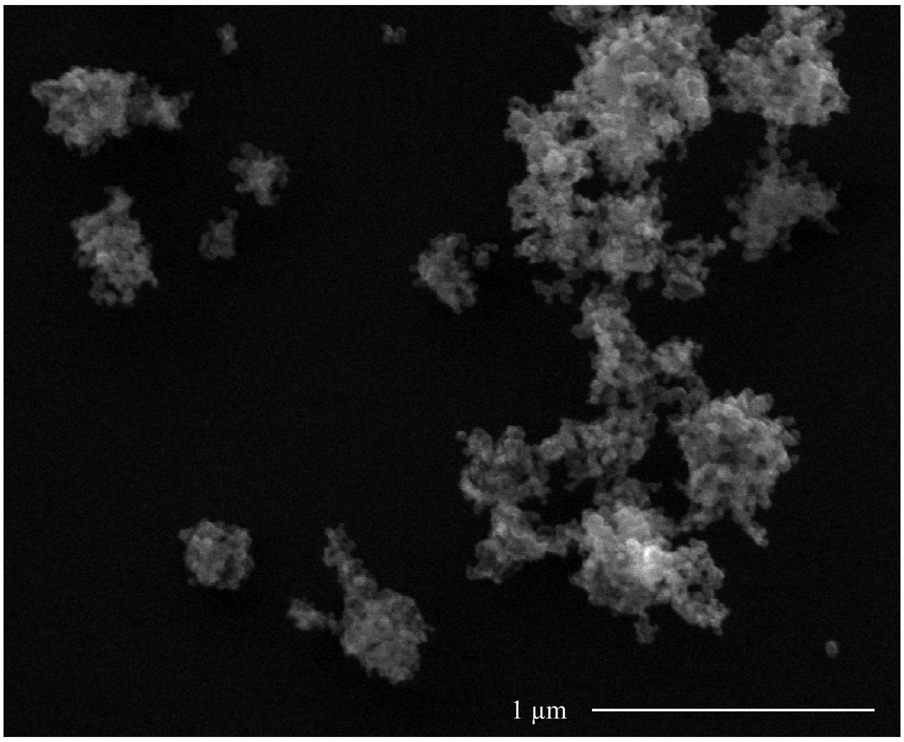


Figure 7.1: An SEM image detail (1 pixel = 5.08 nm) of TiO_2 nanoparticles. The global shapes of the visible agglomerates differ very much.

Other methods from the literature cannot be used because they require extra data such as multispectral images [Hans et al., 2010] or 3D data [Raadnui, 2005] or are not applicable to small and irregularly shaped CPGs because rectangular image patches are needed for their computation [Yu et al., 2007; Stachowiak et al., 2008].

7.3 METHOD

The automatic recognition of engineered nanoparticles in SEM images poses a unique challenge. As explained in Section 5.1 on Page 51, there are several crucial differences between this task and the problems treated in the literature. For the selection of features, this means that we have to find a unique selection of new and existing features suitable for this task. These need to fulfill the following properties:

- The area occupied by a CPG in our images can vary by a factor of several hundred thousand. This means that every feature we use has to be applicable to very small particles as well as very large ones.
- The shape of nanoparticle agglomerates is very variable. Not only does their size vary a lot, their global shape also does not allow any conclusions about the nanoparticles they are made of. This can be seen in Figure 7.1. Some of the agglomerates

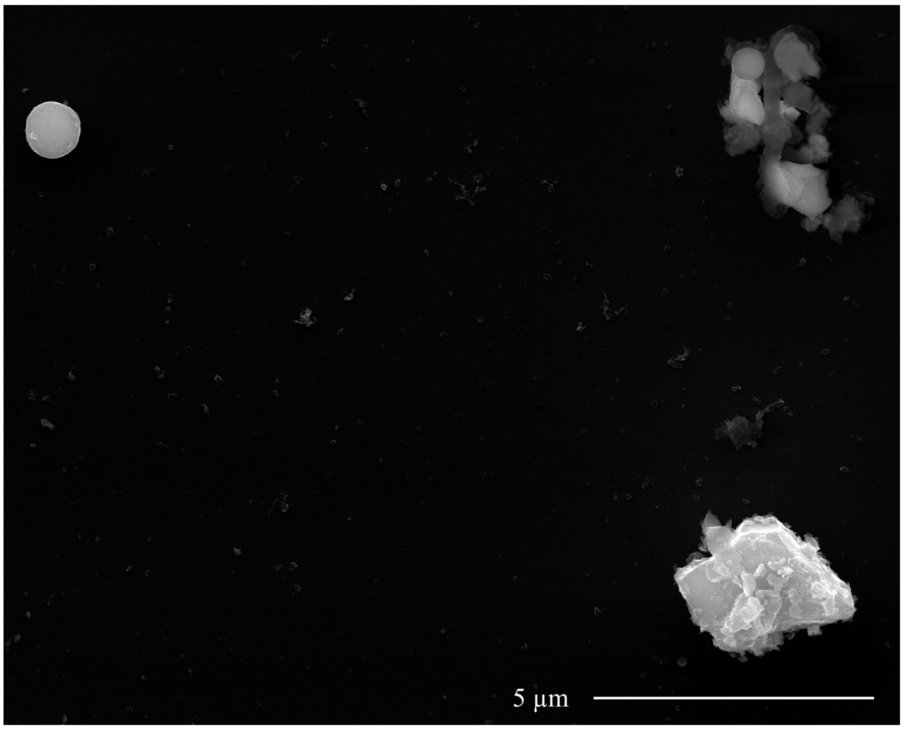
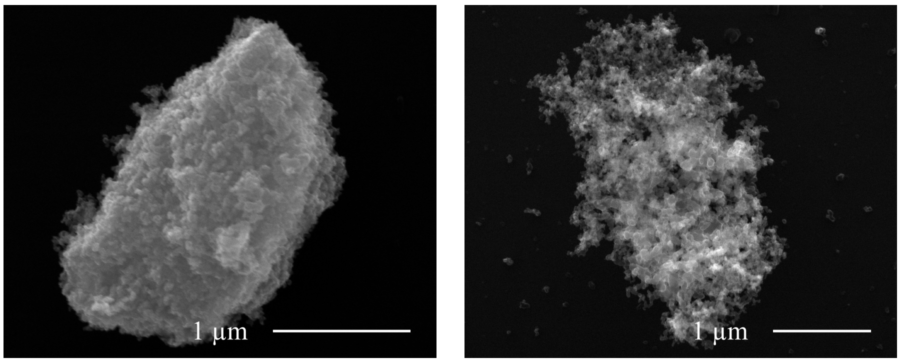


Figure 7.2: An SEM image detail (1 pixel = 5.08 nm) of particles gathered during abrasive cutting of steel. The specific source of each of the particles is unknown. According to employees of the BAuA, the upper left particle may be steel solidified in the form of a sphere, the upper right particle presumably also derives from a cutting process while the lower right particle probably is a silicate, which is a typical component of dust.

in the image are relatively round, some have small tails and the one on the right has a very complex shape. Note, however, that this does not apply if only short segments—in other words local features—of their contours are regarded. These show similarities for agglomerates of nanoparticles with similar sizes. We will elaborate on this shortly. This means that we need to select features that are invariant to changes of the global shape of CPGs.

- As explained in Section 4.2.2.4, the absolute intensity of CPGs does not allow direct conclusions about their composition. Therefore, features should not rely on the absolute intensity.

Although we face a unique challenge, we use some common basic geometric features in order to give the classifier some basic information about the CPG in addition to more specialized features. Specifically, we use the area, variants of the perimeter and the isoperimetric quotient as used by Oster [2010]. These help identify very simple shapes which usually occur in background particles such as the sphere in the upper left corner of Figure 7.2. In addition, to help



(a) An agglomerate of TiO_2 nanoparticles. The primary particles have relatively constant sizes.

(b) A diesel soot agglomerate. It contains small primary particles as well as large ones, especially in its center.

Figure 7.3: Two SEM image details ($1\,\mathrm{pixel} = 1.27\,\mathrm{nm}$) showing different primary particle sizes.

the classifier cope with particles touching the image border, we have added a feature giving the percentage of the contour which does not touch the image border.

While agglomerates of engineered nanoparticles have various sizes and shapes, the sizes of their primary particles are usually relatively constant for a given nanoparticle type. In the case of background particles that form agglomerates such as diesel soot, this is usually not the case. Figure 7.3 shows an example. The primary particles of the TiO_2 agglomerate in Figure 7.3a all have similar sizes. No large particles are visible. The diesel soot agglomerate in Figure 7.3b, however, contains some large primary particles in its center as well as many small ones.

The primary particle sizes influence two aspects of agglomerates in SEM images: Their contour and their texture. They generate local features of a size spectrum corresponding to their size. The numerical features in our system need to be sensitive regarding these spectra. In addition, the scales to which they are sensitive should optimally be configurable. Because of the low signal-to-noise ratio of SEM images and the fact that our segmentation pipeline reconstructs the contours of CPGs well (see Section 6.4.5), we have first concentrated on features characterizing the contour. Therefore, we have developed a feature called mean contour angle wavelet response. Similar to the Fourier descriptors and the multiscale roughness descriptor described in Section 7.2.2, it captures the roughness of the contour at different scales. However, in contrast to all other mentioned features, the mean contour angle wavelet response is not scale-invariant and we are able to exactly specify the wavelength of contour features we are interested in. This is important in order to identify nanoparticle agglomerates.

In order to make intensity information available to the classifier, we calculate a histogram for each CPG. However, since we cannot rely on absolute intensities, these histograms are normalized between the modal background intensity and the maximum intensity of the CPG.

For the goal of capturing the nanoparticle signatures in the texture, we adopt the most commonly used texture feature in the literature, namely Haralick features, because they can be applied to large CPGs as well as small ones.

As mentioned before, the correlation between the number of electrons detected by the electron detector and image intensity can change in every image due to different values of the contrast and brightness settings of the microscope. Therefore, we introduce a novel algorithm to estimate the number of electrons detected for every single pixel in each image. This allows us to use the minimum, maximum and mean electron counts of each CPG as features.

All in all, we use the following features:

- Basic geometric features:
  - Projected area in nm^2
  - Perimeter in nm
  - Perimeter including inner contours in nm
  - Isoperimetric quotient
  - In-image contour percentage
- Shape features:
  - Mean contour angle wavelet response
- Intensity-based features:
  - Normalized relative histogram
  - Haralick features
  - Electron counts

We will explain these features in more detail in the following sections.

7.3.1 *Basic Geometric Features*

The first basic geometric feature we use is the projected area $\phi_{\text{area}} \in \Phi$ measured in nm^2. It is defined as the area a CPG occupies in an image:

$$\phi_{\text{area}}(C, f, s_p) := |C| \cdot {s_p}^2, \quad \forall\, C \in \mathcal{C}, f \in G^P, s_p \in \mathbb{R}_{>0}. \qquad (7.3)$$

The Ag agglomerate on the left in Figure 1.2 on Page 5 occupies 6 322 817 pixels in the image, whose pixel size is 1.27 nm. Therefore, the projected area of the CPG is about 10 198 072 nm^2.

The next feature is the perimeter $\phi_{\text{perimeter}} \in \Phi$ measured in nm. It is the length of the contour that surrounds the CPG in the image. To compute the feature, we first need to locate this contour. This is

Algorithm 7.1 : *FindContour*(C)

```
Input : A CPG $C \in \mathcal{C}$.
Output : The contour $(z_1, \dots, z_{n_z})$ surrounding the CPG $C$ as a
         sequence of contour points $z_i = (x_i, y_i) \in \mathbb{R}^2$. $C$ is
         assumed to be four-connected.
// Find a pixel in the topmost row of the CPG:
1  Let the current pixel $p_{cur}$ be any pixel in the set
   $\{p \in C \,|\, p_y = \min(\{p'_y \,|\, p' \in C\})\}$.
   // $p_{cur}$ will always lie inside $C$.
   // Define the search direction as "up":
2  $\Delta := (0, -1)$.
   // The pixel above the current one is guaranteed to lie
      outside the CPG:
3  $p_{nxt} := p_{cur} + \Delta$
4  $i := 1$. // The index of the next contour point.
   // The first contour point lies between the two pixels:
5  $z_i := \frac{p_{cur} + p_{nxt}}{2}$.
   // Each contour point lies between a pixel inside $C$ and
      a pixel outside it.
   // Define the search direction as "right" so that the
      contour is to our left:
6  $\Delta := (1, 0)$.
7  repeat
8      $p_{nxt} := p_{cur} + \Delta$.
9      while $p_{nxt} \in C$ do
10         $p_{cur} := p_{nxt}$.
           // Turn left to find the next pixel outside of $C$:
11         $\Delta := (\Delta_y, -\Delta_x)$.
12         $p_{nxt} := p_{cur} + \Delta$.
13     end
       // We have found the next contour point:
14     $z := \frac{p_{cur} + p_{nxt}}{2}$.
15     if $z \neq z_1$ then // Is this a new one? Else we are done.
16         $i := i + 1$.
17         $z_i := z$.
           // Turn right so that we do not travel outside $C$:
18         $\Delta := (-\Delta_y, \Delta_x)$.
19     end
20 until $z = z_1$ // We have reached the beginning.
21 $n_z := i$. // The number of contour points.
22 return $(z_1, \dots, z_{n_z})$.
```

done using Algorithm 7.1, which is similar to an algorithm by Zahn [1966]. It starts at a pixel in the top row of the CPG. Then, the algorithm travels along the CPG's edge and records every boundary point it encounters. Each of these points lies between a pixel inside and one pixel outside of the CPG. In every case, these lie in the four-neighborhood of each other.

Given the outputs $(z_1, \dots, z_{n_z})$ of Algorithm 7.1, $\phi_{\text{perimeter}}$ is defined as:

$$\phi_{\text{perimeter}}(C, f, s_p) := s_p \left(\|z_1 - z_{n_z}\|_2 + \sum_{i=1}^{n_z - 1} \|z_{i+1} - z_i\|_2 \right), \quad \forall\, C \in \mathcal{C}, f \in G^P, s_p \in \mathbb{R}_{>0}. \tag{7.4}$$

Here, $\|\cdot\|_2$ stands for the Euclidean norm. The perimeter of the Ag agglomerate mentioned at the beginning of the section is about $17\,054\,\text{nm}$.

The perimeter including inner contours $\phi_{\text{perimeter}+} \in \Phi$ is computed exactly as $\phi_{\text{perimeter}}$ with the only difference that contours of holes inside the CPG are also incorporated into the calculation of the length of the contour. In other words, it is defined as the length of the outer contour plus the lengths of the inner contours. For the Ag agglomerate, the value of this feature is approximately $22\,069\,\text{nm}$.

The next feature, the isoperimetric quotient $\phi_{\text{IQ}} \in \Phi$ is defined as the ratio of the particle's area ϕ_{area} and the area of a circle having a perimeter equal to $\phi_{\text{perimeter}+}$. It is calculated as

$$\phi_{\text{IQ}}(C, f, s_p) := \frac{4\pi\phi_{\text{area}}}{{\phi_{\text{perimeter}+}}^2}, \quad \forall\, C \in \mathcal{C}, f \in G^P, s_p \in \mathbb{R}_{>0}. \tag{7.5}$$

If the particle has a circular shape, its value is 1. If the shape gets more complex, the feature becomes smaller. This complexity can either stem from global features such as a very elongated shape or local properties such as a rugged shape. The Ag agglomerate has an isoperimetric quotient of about 0.263.

The last basic geometric feature we use is the in-image contour percentage $\phi_{\text{IICP}} \in \Phi$. It is defined as the percentage of the CPG's surrounding contour that does not touch an image edge:

$$\phi_{\text{IICP}}(C, f, s_p) := 1 - \frac{s_p |\{(x, y) \in \{z_1, \dots, z_{n_z}\} \mid x \cdot y < 0 \lor x > n_x - 1 \lor y > n_y - 1\}|}{\phi_{\text{perimeter}}(C, f, s_p)}, \quad \forall\, C \in \mathcal{C}, f \in G^P, s_p \in \mathbb{R}_{>0}. \tag{7.6}$$

The set in the equation contains all points of the surrounding contour which lie at the border of the image. Its size multiplied with the pixel size s_p gives us the length of the outer contour that is touching an image edge. Dividing it by the perimeter $\phi_{\text{perimeter}}$ yields the percentage of the outer contour touching an image edge. Finally, by subtracting this value from 1, we get the in-image contour percentage ϕ_{IICP}. For particles which are completely visible, it is equal to 1. The Ag agglomerate has an in-image contour percentage of 0.558.

As mentioned before, we have selected these basic geometric features in order to give the classifier some context. Using these features, it is measured if the CPG is fully visible and if it has a complicated shape. In the next section, we will introduce a feature that will give the classifier much more detailed information about the CPG's contour.

7.3.2 *Shape Features*

The aforementioned features are suitable to distinguish round shapes from irregular ones or small objects from large ones. Still, the agglomerates often only differ in the size distribution of the primary particles. The agglomerates of engineered nanoparticles typically have a small variance in the primary particle sizes while background particles such as diesel soot agglomerates can contain primary particles from a wider size range. These size distributions are typically reflected in the texture and the contour of the agglomerates. Because the texture is greatly affected by the noise and blur in some images, we first concentrate on the contour. In order to identify nanoparticle agglomerates, a shape feature has to have the following properties:

- It has to be insensitive against the global shape of an agglomerate.
- It has to be sensitive towards local contour changes in specific size ranges caused by primary particles of certain sizes.
- It has to be sensitive towards local structures of the *same* size for every CPG regardless of its overall size.

As explained in Section 7.2.2, the shape features used in the literature are not suitable for our purpose. The fractal dimension only allows a rough division of particles because it yields a single feature value. It does not give nearly enough information in order to differentiate agglomerates of engineered nanoparticles from particles such as diesel soot agglomerates. In addition, it is relatively complicated to calculate.

The other popular type of shape features are Fourier descriptors. They capture global frequencies in the function representation. However, local changes such as fluctuations of the phase of these frequencies can pose a problem to the descriptors. In addition, two particles are generally not examined using the same frequencies.

The multiscale roughness descriptor by Drolon et al. [2000] is based on the wavelet transform instead of the Fourier transform. It is able

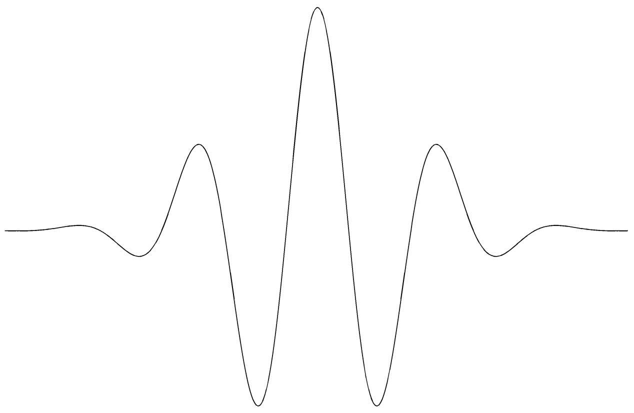


Figure 7.4: The form of the real part of a Morlet wavelet.

to give more local information than the latter. However, similarly to the Fourier transform, the absolute frequencies of the wavelets are in general distinct for two particles due to the different lengths of their contours.

In the next section, we will introduce a feature we have developed that does not have these drawbacks. In addition, we are able to exactly specify the wavelengths of the structures it searches for. It is called mean contour angle wavelet response.

7.3.2.1 *Mean Contour Angle Wavelet Response*

The feature $\phi_{\text{MCAWR},\,\lambda} \in \Phi$ is calculated as follows:

1. Express the contour of the particle as a periodic function.
2. Convolve this function with a wavelet of the predetermined wavelength $\lambda \in \mathbb{R}_{>0}$ measured in nm.
3. Use the mean absolute value of the resulting function as the feature value.

The first step is similar to the first step in the calculations of the Fourier descriptors and the multiscale roughness descriptor. As a result of the convolution, we get a strong response in areas whose structures correspond to the frequency of the wavelet. By taking the average of the absolute values, we get a measure of how prevalent this frequency is in the functional representation of the contour.

The exact method is listed in Algorithm 7.2. The functional representation of the contour, whose details we will discuss shortly, are convoluted with the real part of the complex-valued Morlet wavelet. Its form is visible in Figure 7.4. As the next step, the mean absolute response is calculated. Finally, to obtain the feature value, the result is divided by the in-image contour percentage ϕ_{IICP}. The rationale behind this is that the convolution with parts of the contour that coincide with an image edge usually result in a value of 0. For particles

Algorithm 7.2 : *MeanContourAngleWaveletResponse*(λ, C, f, s_p)

```
Input : The wavelength $\lambda \in \mathbb{R}_{>0}$ of the structures to be detected
        measured in nm, a CPG $C \in \mathcal{C}$, its image $f : P \to G$ and
        the pixel size $s_p \in \mathbb{R}_{>0}$ measured in nm.
Output : The mean contour angle wavelet response
        $\phi_{\mathrm{MCAWR},\,\lambda}(C, f, s_p) \in \mathbb{R}$.
1 $\lambda^* := \frac{\lambda}{s_p}$. // The wavelength measured in pixels.
2 $(\gamma_0, \ldots, \gamma_{n_\gamma - 1}) :=$ NormalizedCumulativeAngularFunction$(C)$.
     // The cumulative angles of the contour of $C$,
     measured at equidistant points one pixel size apart
     from each other (see Algorithm 7.3).
  // Convolve the function $\gamma$ with the real part of a
     Morlet wavelet of wavelength $\lambda^*$:
3 for $i = 0, \ldots, n_\gamma - 1$ do
4     $\varphi_i := \sum_{j=-\lceil 4\lambda^* \rceil}^{\lceil 4\lambda^* \rceil} \gamma_{(i-j) \bmod n_\gamma} \cos\left(2\pi \frac{j}{\lambda^*}\right) e^{-\left(\frac{j}{\lambda^*}\right)^2}$. // For
        brevity, we omitted the constant multiplicative
        and additive terms of the wavelet making its mean 0
        and its $L_1$ norm 1.
5 end
  // To get the feature value, compute the mean absolute
     response of the convolution and divide it by the
     in-image contour percentage in order to account for
     CPGs where only part of the contour is inside the
     image:
6 return $\phi_{\mathit{MCAWR},\,\lambda}(C, f, s_p) := \frac{1}{\phi_{\mathit{IICP}}(C, f, s_p)} \cdot \frac{\sum_{i=0}^{n_\gamma - 1} |\varphi_i|}{n_\gamma}$.
```

such as the Ag agglomerate on the left in Figure 1.2 on Page 5, this would result in a very low feature value. Therefore, we have decided to perform the division, which approximately reverts this effect.

To express the contour as a periodic real-valued function, we have chosen the normalized cumulative angular function also known as normalized tangent-angle function originally proposed by Zahn and Roskies [1972]. It is applicable to non-convex shapes and the function's shape is invariant under translation and rotation. The computation of the function is explained in Algorithm 7.3. At first, the contour of the CPG is expressed as a series of equidistant points progressing clockwise. Then, for each consecutive pair of these points, the angle

Algorithm 7.3 : *NormalizedCumulativeAngularFunction*(C)

```
Input : A CPG $C \in \mathcal{C}$.
Output : The normalized cumulative angular function
         $(\gamma_1, \dots, \gamma_{n_\gamma})$ of the contour surrounding C expressed
         as a sequence of values $\gamma_i \in \mathbb{R}$.
1  $(z_1, \dots, z_{n_z}) := $ FindContour(C). // The contour surrounding C
      as a series of points (see Algorithm 7.1) progressing
      clockwise.
   // Resample the contour so that consecutive points have a
      distance of 1:
2  Let $(z'_0, \dots, z'_{n_\gamma - 1})$ with $n_\gamma \in \mathbb{N}_{>0}$ be a resampled version of
   $(z_1, \dots, z_{n_z})$ so that $\left\| z'_{i+1} - z'_i \right\|_2 = 1, \forall\, i = 0, \dots, n_\gamma - 2$ and
   $\left\| z'_0 - z'_{n_\gamma - 1} \right\|_2 \leqslant 1$.
3  $\gamma := 0$.
4  for $i = 0, \dots, n_\gamma - 1$ do
       // Compute the angle of the vector from $z'_i$ to
          $z'_{(i+1) \bmod n_\gamma}$:
5      $\gamma' := \operatorname{atan2}\left(y'_{(i+1) \bmod n_\gamma} - y'_i, x'_{(i+1) \bmod n_\gamma} - x'_i\right)$.
       // Compute the cumulative angle so that the function
          is continuous:
6      while $|\gamma' - \gamma| > \pi$ do
7          $\gamma' := \begin{cases} \gamma' + 2\pi, & \text{if } \gamma > \gamma', \\ \gamma' - 2\pi, & \text{else.} \end{cases}$
8      end
9      $\gamma := \gamma'$.
       // Normalize the value so that the function is
          continuous and periodic:
10     $\gamma_i := \gamma - \frac{i}{n_\gamma} 2\pi$.
11 end
12 return $(\gamma_1, \dots, \gamma_{n_\gamma})$.
```

of the connecting line segment is calculated. This is done using the function atan2 : $\mathbb{R} \times \mathbb{R} \to (-\pi, \pi]$:

$$\operatorname{atan2}(y, x) := \begin{cases} \arctan\left(\frac{y}{x}\right), & \text{if } x > 0, \\ \arctan\left(\frac{y}{x}\right) + \pi, & \text{if } y \geqslant 0, x < 0, \\ \arctan\left(\frac{y}{x}\right) - \pi, & \text{if } y < 0, x < 0, \\ +\frac{\pi}{2}, & \text{if } y > 0, x = 0, \\ -\frac{\pi}{2}, & \text{if } y < 0, x = 0, \end{cases} \quad \forall y, x \in \mathbb{R}, yx \neq 0. \tag{7.7}$$

We could interpret these values as a function describing the contour of a CPG. However, if the contour was a circle, this function would jump from π to $-\pi$ after half of the function values. To avoid that, we need to calculate the cumulative angle, which is defined as the change of the contour angle between the starting point and the current point. This is done in the loop starting in Line 6. After that step, the function is continuous from the first to the last value. However, we also want the function to be periodic. For a circular contour the first value would be 0 and the last value would be 2π. Interpreted as a periodic function, it would not be continuous. Therefore, we have to normalize the function values. This is done in Line 10. Now, we have a periodic and continuous function. A circle's normalized cumulative angular function would be constantly 0.

The normalized cumulative angular function of the Ag agglomerate on the left in Figure 1.2 on Page 5 is depicted in Figure 7.5. The straight parts of the function are caused by the image edges. Figure 7.6 shows the absolute result values of a convolution of this function with a Morlet wavelet having a wavelength of about 101, which corresponds to $128\,\mathrm{nm}$ (see Algorithm 7.2). Averaging these values, we get about 0.0838. Finally, we get the final feature value of 0.154 by dividing this result by the in-image contour percentage, which is equal to 0.558.

The system uses the versions of the feature $\phi_{\mathrm{MCAWR},\,\lambda}$ with the following wavelengths: $4\,\mathrm{nm}$, $8\,\mathrm{nm}$, $16\,\mathrm{nm}$, $32\,\mathrm{nm}$, $64\,\mathrm{nm}$, $128\,\mathrm{nm}$, $256\,\mathrm{nm}$, $512\,\mathrm{nm}$, $1024\,\mathrm{nm}$ and $2048\,\mathrm{nm}$. We have chosen to scale the values exponentially in order to cover a wide range of frequencies while having a good coverage in the range of the diameters of engineered nanoparticles. Smaller values would not make sense because the images we use have a pixel size of at least $1.27\,\mathrm{nm}$. We have included values much larger than the nanoparticle diameters because multiple primary particles may form substructures inside agglomerates that have sizes in this range. We do not use values larger than $2048\,\mathrm{nm}$ because the contours of most CPGs have a perimeter smaller than this value. The feature values of the Ag agglomerate are listed in Table 7.1.

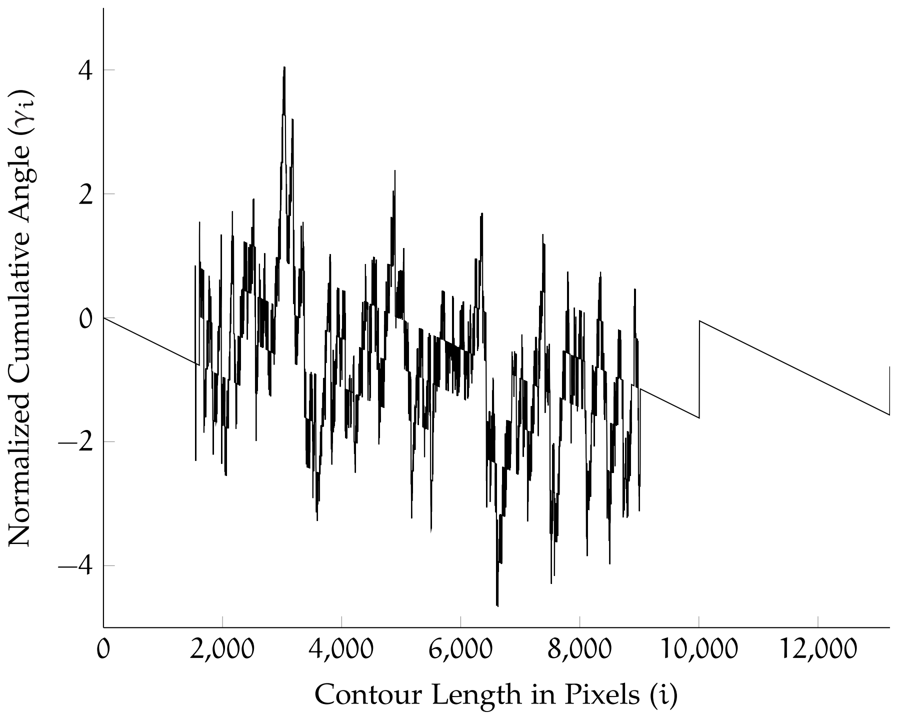


Figure 7.5: The normalized cumulative angular function of the contour of the Ag agglomerate on the left in Figure 1.2 on Page 5 as calculated in Algorithm 7.3.

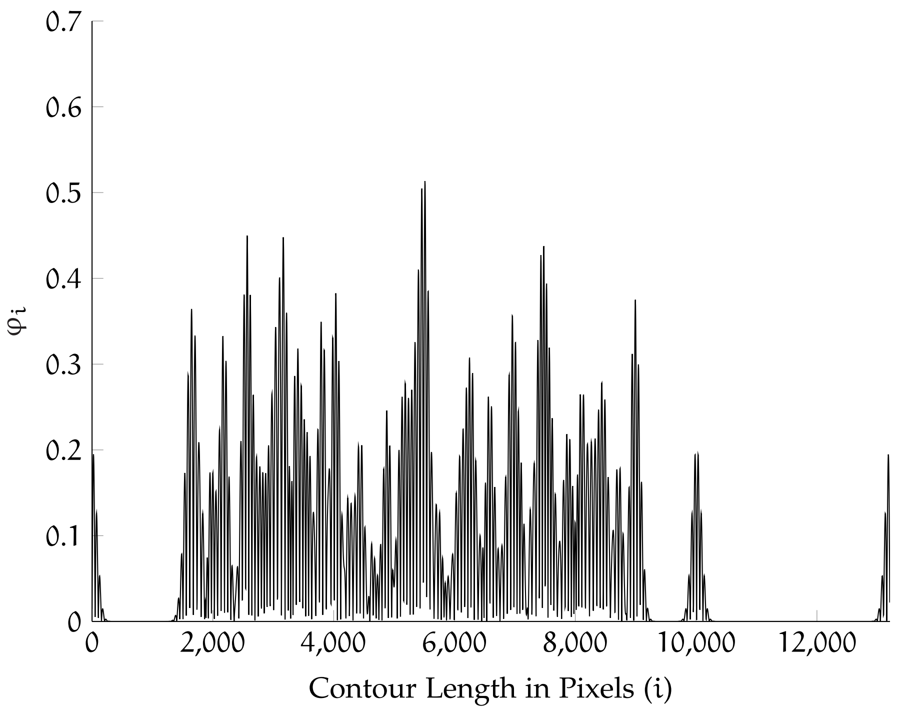


Figure 7.6: The absolute results of convolving the function from Figure 7.5 using a Morlet wavelet with a wavelength corresponding to 128 nm. The details of the computation are listed in Algorithm 7.2.

Table 7.1: The feature values of the mean contour angle wavelet response feature of the Ag agglomerate on the left in Figure 1.2 on Page 5 for different wavelengths.

WAVELENGTH	FEATURE VALUE
$4\,nm$	0.098
$8\,nm$	0.067
$16\,nm$	0.050
$32\,nm$	0.056
$64\,nm$	0.092
$128\,nm$	0.154
$256\,nm$	0.199
$512\,nm$	0.247
$1024\,nm$	0.282
$2048\,nm$	0.246

7.3.3 *Intensity-based Features*

The intensity-based features we use have to be able to deal with the fact that the absolute intensity of a particle has little significance. As explained in Section 4.2.2.4, the intensity can vary between images and even within a single image. We have to take that into consideration when selecting intensity-based features.

In the literature, intensity-based feature are the least-popular category compared to basic geometric and shape features. As mentioned in Section 7.2.3, Haralick features are the most popular kind of intensity-based features. We also use them in our system to capture the structures inside agglomerates introduced by their primary particles. Additionally, they are applicable to very small and also large particles.

In addition, we use a histogram feature to describe the intensity distributions of the CPGs. We use a normalized calculation method to cope with variable intensities. Finally, we introduce a new feature based on an estimation of the number of electrons detected for each pixel. We will describe these features in the following sections.

7.3.3.1 *Normalized Relative Histogram*

One distinguishing factor of the different particle types are the varying intensity distributions. As explained in Section 4.2.2.4, the Ag agglomerates often appear brighter than many other particle types. To include information about the CPGs' intensity distributions, we compute a normalized relative histogram $\phi_{\text{hist},\,i} \in \Phi, i = 0, \ldots, 9$ with 10

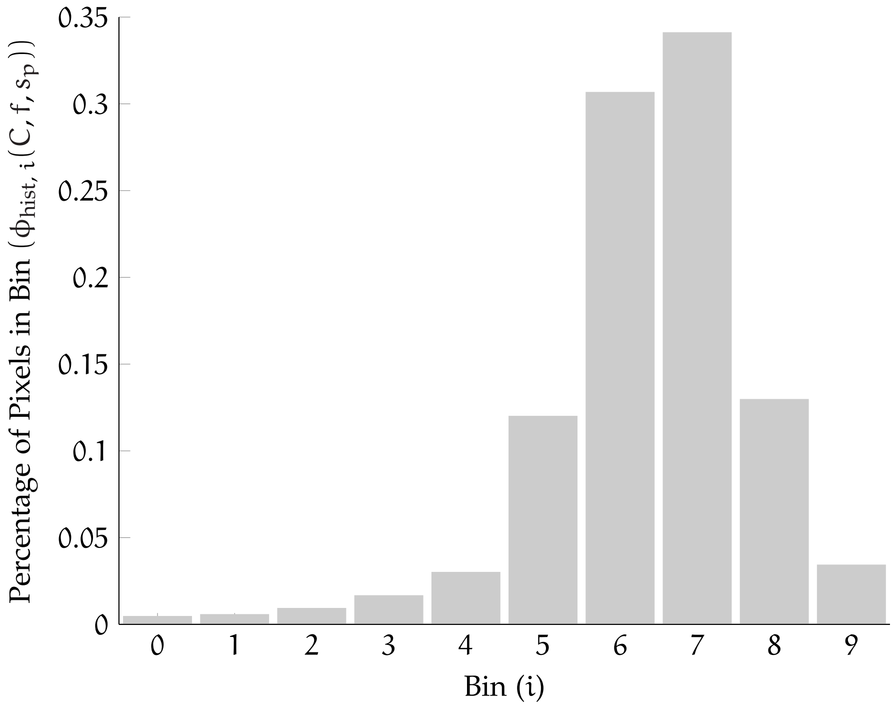


Figure 7.7: The normalized relative histogram of the Ag agglomerate on the left in Figure 1.2 on Page 5.

bins $\phi_{\text{hist},\,0}, \ldots, \phi_{\text{hist},\,9}$ for the pixels of every CPG. The features are calculated as follows:

$$\phi_{\text{hist},\,i}(C, f, s_p) :=$$
$$\frac{\left| \left\{ p \in C \;\middle|\; \frac{i}{10} \leqslant \frac{f(p) - \arg\max_{g \in G} H_f(g)}{\max_{p' \in C} f(p') + \varepsilon - \arg\max_{g \in G} H_f(g)} < \frac{i+1}{10} \right\} \right|}{|C|},$$
$$\forall\, C \in \mathcal{C}, f \in G^P, s_p \in \mathbb{R}_{>0}, i = 0, \ldots, 9. \quad (7.8)$$

Here, $H_f : G \to \mathbb{N}$ is the absolute histogram of the image f (see Equation (6.35)) and $\varepsilon \in \mathbb{R}_{>0}$ is a small positive value. The distribution of the bins is normalized between the modal image intensity $\arg\max_{g \in G} H_f(g)$ and the maximum intensity of the CPG $\max_{p' \in C} f(p')$. This way, the parameter values are invariant under linear transformations of the image intensities such as changes to the contrast and brightness settings of the microscope. The intensities of the CPG are guaranteed to lie above the modal image intensity because the threshold used for the segmentation is always higher than this value (see Section 6.3.3). The ε in Equation (7.8) is introduced to guarantee that pixels with intensities equal to $\max_{p' \in C} f(p')$ are included in the last bin. We ensure that the values do not depend on the size of the CPG by dividing everything by the total number of pixels in the CPG $|C|$.

The histogram of the Ag agglomerate is shown in Figure 7.7.

7.3.3.2 *Haralick Features*

In order to use the additional information given by the texture of the particles, we have decided to use Haralick features. This technique is used in several publications on particle detection and classification [Langford et al., 1990; Flores et al., 2003; Laghari, 2003; Rodriguez-Damian et al., 2006; Stachowiak et al., 2008]. In fact, it is the only intensity-based feature used in more than one publication examined by us. We have chosen the approach because it is applicable to small and irregularly shaped particles.

Haralick features are derived from the so-called co-occurrence matrix $O_{C,f,\Delta}(g,g') \in \mathbb{R}_{\geqslant 0}$, $g, g' \in G$. The value $O_{C,f,\Delta}(g,g')$ is defined as the probability that a pair of intensities g and g' occurs in the CPG C with the offset $\Delta \in P - P := \{p - p' \mid p, p' \in P\}$. The formal definition is the following:

$$O_{C,f,\Delta}(g,g') := \frac{|\{p \in C \mid p + \Delta \in C, f(p) = g, f(p+\Delta) = g'\}|}{|\{p \in C \mid p + \Delta \in C\}|},$$
$$\forall\, C \in \mathcal{C}, f \in G^P, \Delta \in P - P, g, g' \in G. \quad (7.9)$$

The entries of the co-occurrence matrix could be used directly as features but assuming 256 different image intensities, this would amount to 65 536 different features per offset. This would result in a computationally expensive classification training and, because of the curse of dimensionality (see Section 8.1.2), in an inferior classification performance. This is the reason why several measures derived from the co-occurrence matrix are used instead.

Haralick et al. [1973] originally describes 14 statistics. However, several other measures have been proposed. We have chosen the set of features used by Stachowiak et al. [2008] because they report a good performance for wear particles having similar characteristics to nanoparticle agglomerates. They use the following measures [Stachowiak et al., 2005]:

- Contrast $\phi_{\mathrm{HC},\Delta}$
- Energy $\phi_{\mathrm{HEne},\Delta}$
- Entropy $\phi_{\mathrm{HEnt},\Delta}$
- Local homogeneity $\phi_{\mathrm{HLH},\Delta}$
- Cluster shade $\phi_{\mathrm{HCS},\Delta}$
- Cluster prominence $\phi_{\mathrm{HCP},\Delta}$
- Maximum probability $\phi_{\mathrm{HMP},\Delta}$

For $C \in \mathcal{C}, f \in G^P, s_p \in \mathbb{R}_{>0}, \Delta \in P - P$, they are defined as follows [Stachowiak et al., 2005]:

$$\phi_{\mathrm{HC},\Delta}(C,f,s_p) := \sum_{g,g' \in G} (g-g')^2 O_{C,f,\Delta}(g,g'), \quad (7.10)$$

Table 7.2: The feature values of the Haralick features of the Ag agglomerate on the left in Figure 1.2 on Page 5 for different offsets.

FEATURE	FEATURE VALUE FOR OFFSET			
	$(1,0)$	$(1,1)$	$(0,1)$	$(-1,1)$
$\phi_{\mathrm{HC},\Delta}$	70.496	86.926	85.231	87.449
$\phi_{\mathrm{HEne},\Delta}$	0.000 418 9	0.000 381 8	0.000 383 1	0.000 380 8
$\phi_{\mathrm{HEnt},\Delta}$	11.779	11.922	11.911	11.927
$\phi_{\mathrm{HLH},\Delta}$	0.1428	0.1303	0.1313	0.1301
$\phi_{\mathrm{HCS},\Delta}$	−351 620	−343 721	−351 401	−347 069
$\phi_{\mathrm{HCP},\Delta}$	1.246×10^8	1.213×10^8	1.241×10^8	1.224×10^8
$\phi_{\mathrm{HMP},\Delta}$	0.001 201	0.001 109	0.001 090	0.001 056

$$\phi_{\mathrm{HEne},\Delta}(C, f, s_p) := O^2_{C,f,\Delta}(g, g'), \tag{7.11}$$

$$\phi_{\mathrm{HEnt},\Delta}(C, f, s_p) := -\sum_{g,g' \in G} O_{C,f,\Delta}(g, g') \cdot \log_2\left(O_{C,(}(g, g')\right), \tag{7.12}$$

$$\phi_{\mathrm{HLH},\Delta}(C, f, s_p) := \sum_{g,g' \in G} \frac{1}{1 + (g - g')^2} O_{C,f,\Delta}(g, g'), \tag{7.13}$$

$$\phi_{\mathrm{HCS},\Delta}(C, f, s_p) := \sum_{g,g' \in G} (g - M_{1,C,f,\Delta} + g' - M_{2,C,f,\Delta})^3 O_{C,f,\Delta}(g, g'), \tag{7.14}$$

$$\phi_{\mathrm{HCP},\Delta}(C, f, s_p) := \sum_{g,g' \in G} (g - M_{1,C,f,\Delta} + g' - M_{2,C,f,\Delta})^4 O_{C,f,\Delta}(g, g'), \tag{7.15}$$

$$\phi_{\mathrm{HMP},\Delta}(C, f, s_p) := \max_{g,g' \in G} O_{C,f,\Delta}(g, g'), \tag{7.16}$$

where

$$M_{1,C,f,\Delta} = \sum_{g,g' \in G} g O_{C,f,\Delta}(g, g'), \tag{7.17}$$

$$M_{2,C,f,\Delta} = \sum_{g,g' \in G} g' O_{C,f,\Delta}(g, g'). \tag{7.18}$$

We compute these features for the following offsets: $(1, 0)$, $(1, 1)$, $(0, 1)$, $(-1, 1)$. The feature values of the Ag agglomerate are listed in Table 7.2.

7.3.3.3 *Electron Counts*

For each pixel of each SEM image, the intensity is determined by counting the electrons detected by the electron detector at the corresponding point on the specimen and amplifying this signal (see Section 4.2.1). This process can be modeled using Equation (6.12) on Page 66. The electron count is multiplied with the value of the contrast setting a and the value of the brightness setting b is added to obtain the intensity. However, the values of these settings are unknown to us and generally change for every image. This means that the absolute intensities of an image are only weakly connected to the electron counts and reveal very little about the specimen. Knowing the electron counts of the pixels of a CPG, however, would be very insightful. Samples made from carbon, for example, which is the main component of diesel soot, generate only about half as many secondary electrons as other materials [Goldstein et al., 2003, p. 95]. Knowing the electron counts of the pixels of diesel soot agglomerates could therefore be crucial to distinguish them from agglomerates made of engineered nanoparticles.

The noise estimation method we have developed and described in Section 6.3.1 is able to calculate estimates of the contrast and brightness setting values a and b for each image. Using these and Equation (6.12) on Page 66, we are able to directly estimate the electron count of every pixel in each image. From these, we can derive features that describe the minimum, maximum and mean estimated electron counts $\phi_{\text{min EC}}, \phi_{\text{max EC}}, \phi_{\text{mean EC}} \in \Phi$ of each CPG. The computation of these features is described in Algorithm 7.4. The feature values of the Ag agglomerate are approximately 240, 1712 and 1259, respectively. For background particles in our database taken from the ambient air, the maximum estimated electron count can take values below 20. We observe such values for several particles in multiple images. In such a case, according to the Poisson distribution, the standard deviation of the noise equals to more than 20 % of the mean signal value (see Section 6.3.1.1). This shows that the signal-to-noise ratio of these images is indeed extremely low.

The parameter $n_{\text{tile}} \in \mathbb{N}_{>0}$ is needed for the estimation of the contrast and brightness setting values. We use a fixed value of 16, which we have found to be best in the evaluation of the noise estimation in Section 6.4.1.

In order to compare the electron counts to another feature and to test the hypothesis that absolute intensities are not very meaningful in the case of SEM, we will include the maximum intensity of each CPG as a feature $\phi_{\text{max I}} \in \Phi$:

$$\phi_{\text{max I}}(C, f, s_p) := \max_{p \in C} f(p), \quad \forall\, C \in \mathcal{C}, f \in G^P, s_p \in \mathbb{R}_{>0}. \tag{7.19}$$

Algorithm 7.4 : *EstimateElectronCounts*$(n_{\text{tile}}, C, f, s_p)$

```
Input : The parameter $n_{\text{tile}} \in \mathbb{N}_{>0}$ for the noise estimation, a CPG
        $C \in \mathcal{C}$, its image $f : P \to G$ and the pixel size $s_p \in \mathbb{R}_{>0}$
        measured in $nm$.
Output : The minimum, maximum and mean estimated electron
         counts
         $\phi_{\text{min EC}}(C, f, s_p), \phi_{\text{max EC}}(C, f, s_p), \phi_{\text{mean EC}}(C, f, s_p) \in \mathbb{R}$
         of the CPG C.
  // Calculate estimates $\hat{a}, \hat{b} \in \mathbb{R}$ of the contrast and
     brightness settings:
1 $\hat{a}, \hat{b} := EstimateNoiseParameters(f, n_{\text{tile}})$. // See Algorithm 6.2
     on Page 78.
  // Estimate the feature values using Equation (6.12) on
     Page 66:
2 $\phi_{\text{min EC}}(C, f, s_p) := \min_{p \in C} \frac{f(p) - \hat{b}}{\hat{a}}$.
3 $\phi_{\text{max EC}}(C, f, s_p) := \max_{p \in C} \frac{f(p) - \hat{b}}{\hat{a}}$.
4 $\phi_{\text{mean EC}}(C, f, s_p) := \frac{1}{|C|} \sum_{p \in C} \frac{f(p) - \hat{b}}{\hat{a}}$.
5 return $\phi_{\textit{min EC}}(C, f, s_p), \phi_{\textit{max EC}}(C, f, s_p), \phi_{\textit{mean EC}}(C, f, s_p)$.
```

In this chapter, we have described every feature our system uses to describe the CPGs. In the next chapter, we will explain the classification system that uses the values of these features to distinguish CPGs made of engineered nanoparticles from those made of background particles. It will also contain an evaluation of the features described in this chapter because the quality of a feature can best be assessed by the impact it has on the classification result.

8 CLASSIFICATION

This chapter will address the task of deciding for each CPG that has been found if it is composed of engineered nanoparticles. Furthermore, we will present a method that is able to automatically select classification parameters so that the system is able to adapt to new particle types. In Section 8.1, we will look at the remaining tasks to be fulfilled by our system in order to be able to successfully classify engineered nanoparticles. Section 8.2 will explain how other publications have solved similar problems. In Section 8.3, we will describe how our system performs these tasks. Finally, Section 8.4 will perform a thorough evaluation of our system as a whole.

8.1 GOALS

In Chapter 3, we have outlined goals that our system shall be able to fulfill. Our primary target was to reduce the amount of manual work in the detection and identification of engineered nanoparticle in SEM images. In particular, we formulated the following goals:

- Minimum goals:

  DETECTION Locate the engineered nanoparticles and (optionally) the background particles in every image.

  IDENTIFICATION Decide for every found particle if it either consists of engineered nanoparticles of a specific type or is a background particle.

- Optional goals:

  FUTURE VIABILITY Be able to adapt to new particle types and requirements.

  PERFORMANCE PREDICTION Predict the influence of additional training data on the quality of the system's particle identification to guide the user's decision if more training samples shall be produced.

We have shown in Chapter 6 that our system is able to successfully perform the first task. In addition, we have partly addressed the second goal in Chapter 7 by describing how the system computes characteristic numerical features of each found CPG. The missing part of this goal is solved using statistical classification and additional dimensionality reduction, whose general concepts we will explain in Sections 8.1.1 and 8.1.2. After that, we will address the goal of future viability in Section 8.1.3. The last goal will be the topic of Chapter 9.

8.1.1 *Classification*

As the result of the feature computation step of our pipeline, we have a list of feature values $(\phi_i(C, f, s_p))_{i=1,\dots,n_\phi} \in \mathcal{X}$ for each CPG C, its so-called feature vector. The job of the classification step is then to assign each CPG to a class $Y \in \mathcal{Y} := \{Y_+, Y_-\}$. Here, the "positive" class Y_+ is the class of all CPGs made of engineered nanoparticles. Conversely, the "negative" class Y_- is the class of all CPGs made of background particles.

In practice, this is done using a so-called classifier. It has to be trained using a set of CPGs that have manually been labeled as belonging to either of the classes. This produces a learned classifier, which is able to classify previously unknown CPGs on its own.

More formally, a classifier l is trained using a training set T to produce a learned classifier $l(T)$, which is able to assign a class $l(T)\big(\vec{X}\big) \in \mathcal{Y}$ to a CPG with the feature vector $\vec{X} \in \mathcal{X}$. Here, the training set $T \sqsubseteq \mathcal{X} \times \mathcal{Y}$ is a multiset with elements in $\mathcal{X} \times \mathcal{Y}$. Each member $\big(\vec{X}, Y\big) \in \mathcal{X} \times \mathcal{Y}$ of T means that there is a CPG of class Y with the feature vector $\vec{X}$. We have defined T as a multiset because there can be multiple CPGs with the same feature values of the same class in a single training set. If we define $\mathcal{D} := \{D \mid D \sqsubseteq \mathcal{X} \times \mathcal{Y}\}$, then the classifier is a function $l : \mathcal{D} \to \mathcal{Y}^{\mathcal{X}}$ where $\mathcal{Y}^{\mathcal{X}}$ is the set of all functions from $\mathcal{X}$ to $\mathcal{Y}$. In most cases, a classifier l has a set of parameters that can be adjusted in order to change its behavior. Such parameters are always changed before training the classifier.

Often, l as well as $l(T)$ are simply called classifier. However, we will use the convention of calling l classifier, unlearned classifier or untrained classifier while we will refer to $l(T)$ as learned or trained classifier.

8.1.2 *Dimensionality Reduction*

As explained in the previous section, a classifier l is given a training set T in order to produce a function $l(T) : \mathcal{X} \to \mathcal{Y}$ that is able to tell for any CPG to which class $Y \in \mathcal{Y}$ it belongs solely based on its feature vector $\vec{X} \in \mathcal{X}$. The quality of this function depends, among others, on the following factors [Jain et al., 2000]:

- The size of the training set $|T|$.
- The complexity of the classifier.
- The number of features n_ϕ.

In general, a larger training set leads to better results. This observation, however, is only valid under the assumption that the distribution of the samples in the training set is the same as for the samples the quality of the learned classifier is assessed with.

The classifier has to be chosen specifically for each individual classification problem. Given a large enough training set, a complex classifier can improve the results. For relatively small training sets, a simple classifier is usually a better choice. We will elaborate on that in Section 8.3.1.1.

The feature count has a strong influence on the classification results. Without any features, no classifier can perform well if there is more than one class. In some cases, one feature may be enough to achieve a good classification result. In most cases, though, using multiple features leads to a better classification. This is a predictable result because the classifier can base its decision on more information. However, it has been observed that, at a certain point, adding more features deteriorates the classification results. This effect is known as the curse of dimensionality or the peaking phenomenon. Jain et al. [2000] provide an intuitive explanation of this effect: Most classifiers produce the learned classifier simply by altering the parameters of a base function. The more features are used, the more parameters this function has. Given a fixed training set size, the ratio of training samples to unknown parameters decreases. This makes the parameter estimation less reliable.

Another explanation is as follows: We can think of each sample as a point in n_ϕ-dimensional space, the so-called feature space, if we interpret its feature vector as coordinates. The job of the classifier is to estimate the probability of a sample in a specific region of this space of belonging to a certain class. If we increase n_ϕ and add more features, the dimension of the feature space increases. Assuming a fixed training set size, the density of the training samples in this space decreases accordingly. This makes the estimation of the probabilities more difficult.

The ideal number of features depends on the training set size, the classifier complexity, the types of features and the problem domain and cannot be predicted in advance. In addition, one cannot predict which features are best suited to represent the characteristics of the samples. Due to these reasons, in most cases, the number of available features is too high. This is especially true if only few training samples are available. In such cases, dimensionality reduction can be used to improve the classification performance. Its goal is to reduce the number of features in order to improve the classification quality. The classifier is then trained using the reduced set of features. Due to the relatively low number of samples in our database, dimensionality reduction can be a useful addition to our system.

8.1.3 *Future Viability*

In Chapter 2, we have outlined that the requirements a system like ours will have to meet may change in the future. In particular, the types of particles it shall be able to identify may change. If these are very different from the three particle types in our images, new features may have to be added and the segmentation approach may even have to be changed. Such changes must be made by a computer scientist or programmer and cannot be performed by the user of our system. However, if the new particle types are similar to the ones the system is currently targeted at, using a different selection of the already implemented features and changing the classifier or its parameters may be sufficient to achieve a good classification performance. In such a case, we want the system to be able to adapt itself without needing the user to set any parameters. More specifically, when training the system for any type of engineered nanoparticles, it shall be able to fully automatically choose an appropriate classifier, its parameters and the subset of features to use. In the literature on pattern recognition and machine learning, this task is called model selection.

8.2 RELATED WORK

The publications on automatic image-based particle analysis use a wide array of classification algorithms. The used approaches range from expert systems [Xu et al., 1998] over linear discriminant analysis [Langford et al., 1990; Thomas et al., 1995], k-nearest neighbors [Kindratenko et al., 1997; Oster, 2010] and support vector machines (SVMs) [Luo et al., 2004; Rodriguez-Damian et al., 2006; Stachowiak et al., 2008] to neural networks [Wienke et al., 1995; Kindratenko and Van Espen, 1996; Xu et al., 1998; Laghari, 2003; Calderon De Anda et al., 2005; Raadnui, 2005; Rodriguez-Damian et al., 2006]. However, instead of looking at these publications we will concentrate on the general literature on pattern recognition and machine learning. The reason is that the feature representation of CPGs is abstract and not domain-dependent. Classification algorithms are designed to learn the distributions of any problem domain.

In the next section, we will give an overview of the literature on statistical classification. Then, Section 8.2.2 will present different approaches to dimensionality reduction. Finally, Section 8.2.3 will address the task of finding an appropriate classifier and matching parameters for a given classification task.

8.2.1 *Classification*

First, we want to define some terminology as we use it in this thesis. Other publications may use the same terms differently.

The fields of machine learning and pattern recognition are strongly connected. Both aim at enabling computers to make autonomous decisions. Some even say that both "activities can be viewed as two facets of the same field" [Bishop, 2006, p. vii]. We will not try to give an exact definition of these terms.

Both fields make a distinction between unsupervised classification or learning and supervised classification or learning [Jain et al., 2000]. Both tasks have in common that an unknown sample shall be assigned to a class. In the former case, the classes that the samples are assigned to are unknown beforehand. In the case of supervised classification or learning, there are predefined classes. For this thesis, only the latter case is relevant. Therefore, the definition given in Section 8.1.1 describes supervised learning. If we refer to classification in this thesis, we will always mean supervised classification.

Jain et al. [2000] categorize classification approaches into four groups:

- Template matching
- Syntactic or structural matching
- Statistical classification
- Neural networks

The idea of template matching is that there is a prototypical instance (the template) of each class, which every new sample is compared to. Then, the sample is assigned to the class whose template it's most similar to. The approach is unsuitable if there are large variations among the samples of a single class [Jain et al., 2000], as is the case for nanoparticle agglomerates. Therefore, we do not use this method.

The syntactic approach is based on the assumption that the pattern to be recognized is composed of subpatterns which in turn are composed of even simpler patterns, called primitives. The idea is that the pattern is defined by the relationship between these primitives. These relationships can be described by a grammar, which is automatically inferred from the training set. However, to use this approach, a large training set is required. In our case, generating samples is accompanied by considerable costs. Therefore, this method is unsuitable for our purpose.

In the case of statistical classification, the samples are expressed in terms of feature values as explained in Section 7.1. Using these feature values, the classification algorithm then assigns the sample to a class as described in Section 8.1.1.

Neural networks are a special case of statistical classifiers. They are composed of many nodes, which perform simple computations. The feature values of a single sample are used as the input for the first set of nodes. The results of the computations of these nodes are then propagated through a series of nodes until the final set of nodes outputs the class of the sample. Although the computations performed by a single node are very simple, neural networks are able to infer complex relationships between feature values and classes.

A variant of neural networks, which has become popular in recent years, are convolutional neural networks [Krizhevsky et al., 2012]. These do not work with features. Instead, their input are the image pixels themselves. Through a series of nodes organized in layers, which perform convolutions and combine the inputs locally, the amount of information is reduced to such a degree that it can be used as the input of a conventional fully-connected neural network.

Another special case of statistical classification is the so-called outlier detection or anomaly detection [Dokas et al., 2002]. Such approaches group samples into two categories: normal and abnormal. They are trained by learning the characteristics of normal samples. Any unknown sample that does not fit these characteristics is then classified as abnormal. Outlier detection is suitable for applications with the following properties:

- The characteristics of all subtypes of normal samples are well known. Otherwise, a normal sample of an unknown subtype would be classified as abnormal.
- The characteristics of abnormal samples are not well known. Otherwise, a normal statistical classifier is better suited because it can use this information to better distinguish the two classes.

However, our problem does not fit these properties. There are many kinds of airborne particles and only a minority of them are engineered nanoparticles. Therefore, it is impossible to create a training set that contains every possible type of background particles. This would contradict the first property if we would define background particles as "normal". In addition, we can relatively easily create training images of background particles. Therefore, defining them as "abnormal" would disagree with the second property.

Because of the reasons mentioned above, namely relatively small training sets and complex classes, we have decided to use statistical classifiers. As mentioned before, each sample can be thought of as a point in the n_ϕ-dimensional feature space if we interpret its feature vector as coordinates. Using this interpretation, a random sample of class $Y \in \mathcal{Y}$ has a certain probability of being located in a given region of the feature space. These probabilities are called class-conditional densities. If they are completely known, then the optimal Bayes decision rule can be used to assign a class to a previously unknown sample [Jain et al., 2000]. If at least their form is known, their parameters can be estimated using the training set. This approach is called the Bayes plug-in classifier. However, we know nothing about the class-conditional densities of our classification problem. Therefore, the classifier we use has to either estimate the class-conditional densities or operate based on a learned decision boundary. The latter is defined as a hypersurface in the feature space that splits the samples that are assigned to one class from those that are assigned to another.

8.2.2 *Dimensionality Reduction*

According to Jain et al. [2000], there are two types of dimensionality reduction: feature extraction and feature selection. These terms are often used interchangeably in the literature. We will use them as defined by Jain et al. [2000].

Feature extraction methods create new features by calculating them from existing ones. The hope is that these features are more expressive than the old ones and are more useful for the classification.

Feature selection algorithms select a subset of features, which is then used as the input of the classifier. Optimally, the most useful features are selected and the ones that are not helpful to differentiate the classes are rejected.

Also note that both methods are often used together. In these cases, feature extraction is used in order to produce additional features and feature selection is applied subsequently to reduce the overall feature count.

8.2.3 *Model Selection*

We want our system to automatically select an appropriate classifier and good values for the classifier parameters for each type of engineered nanoparticles. This task is called model selection. For that, three questions have to be answered:

- How do we evaluate the quality of a classifier-parameter combination?
- How do we estimate the error of a given classifier-parameter combination in order to calculate the evaluation metric?
- How do we select classifier-parameter combinations to be tested?

To answer the first question, we need to select an evaluation metric or performance measure. Most often used is the accuracy [Japkowicz and Shah, 2011, p. 7]. It is defined as the percentage of the samples that are correctly classified. However, this metric is unsuitable if, as in our case, samples of one class are much rarer than those of another class. For example, the ratio of Ag nanoparticle CPGs to background CPGs in our database is about 1:85.

The shortcoming of the accuracy metric can be illustrated by the following example: We want to evaluate cancer screening tests for the general public. Test A always predicts that a subject is healthy. Therefore, it is not able to detect a single cancer case. Test B, on the other hand, is able to detect every single cancer case and has a relatively low probability of 0.1 % of predicting cancer in a healthy subject. Test B is clearly the better choice. However, let us assume that 99.99 % of the public are healthy. Then, Test A achieves an accuracy of 99.99 % while test B only obtains an accuracy of about 99.90 %. We will describe more appropriate metrics in Section 8.3.3.

To answer the second question, we need to find a way to test a given classifier-parameter combination in order to estimate its error on real-world data. The most obvious solution of using all labeled samples as the training set and testing the performance of the classifier on the same data is not appropriate. In this case, the error is underestimated because the learned classifier is biased towards the samples in the training set. A simple lookup table that remembers the classes of all labels in the training set would commit no errors if it was tested on the training set.

Several other error estimation approaches have been proposed [Japkowicz and Shah, 2011, p. 163]:

- Holdout
- Random subsampling
- Bootstrapping
- Cross-validation
- Repeated cross-validation

The holdout method randomly splits the dataset into two separate sets, the training set and the test set. The classifier is trained on the training set and the performance of the learned classifier is then evaluated on the test set. The advantage of this approach is that the training and the test set are independent. However, in order to successfully use this method, the number of labeled samples of each class has to be relatively high. If this is not the case, we have to designate a large part of the dataset as the test set so that the error estimate is reliable. However, this means that the training set is small, which results in a poorly trained classifier leading to a higher error estimate.

Random subsampling repeats the holdout method multiple times with different choices of the training and test sets. This makes the error estimate more reliable. However, the limited size of the training sets is still a problem.

Bootstrapping is similar to random subsampling. However, the training set has the same size as the full dataset and is drawn with replacement from it. This means that the training set can contain multiple instances of a single sample. The test set is defined as the whole dataset without the samples that are in the training set. This procedure is repeated multiple times. Bootstrapping has the advantage that a larger training set is available to the classifier. However, it is unsuitable for classifiers such as the nearest neighbor algorithm that cannot make use of repeated training samples.

In the case of cross-validation, the dataset is randomly split into $n_F \in \mathbb{N}_{>0}, n_F \geqslant 2$, separate sets of roughly equal size. These sets are called folds. Then, the classifier is trained on $n_F - 1$ folds and evaluated on the remaining fold. This procedure is repeated n_F times so that each fold is used exactly once for evaluation. This method allows the classifier to be tested on every sample in the dataset. At the same time, it is always trained on a sufficiently large training set. A special case where n_F is equal to the dataset size is called leave-one-out. Here, in each iteration, the classifier is trained on all but one sample and tested on the remaining one. A slightly modified version of this method is called stratified cross-validation. Here, the folds are generated such that the class distributions of every fold are approximately equal.

Repeated cross-validation works by applying cross-validation multiple times on the same dataset. This can be done to obtain a more reliable error estimate. However, this approach increases the computation time by a factor equal to the number of repetitions.

The final question to be answered is how to select specific classifier-parameter combinations to be evaluated using the approaches explained above. Several types of algorithms have been proposed to do this:

- Grid-search
- Random search
- Bayesian optimization
- Evolutionary algorithms

Grid-search works by selecting some fixed values for each parameter and then testing each combination of these values [Hsu et al., 2010]. This approach is usually too time-consuming because its runtime grows exponentially with the number of parameters.

Random search does not rely on predetermined parameter values. Instead, as the name suggests, it chooses classifier-parameter combinations randomly. This approach has been shown to perform better than grid-search [Bergstra and Bengio, 2012]. However, a major disadvantage is that random search does not make use of the evaluation results of previously chosen classifier-parameter combinations in the selection of new ones. For example, it does not prefer combinations similar to those which have previously proven to provide good results.

Bayesian optimization tries to build a statistical model of the relationship between parameter values and classification error [Thornton et al., 2013]. It chooses the next configuration based on a trade-off between probing unexplored regions in the parameter space and choosing promising configurations. The model takes its knowledge from all previously evaluated classifier-parameter combinations.

Finally, evolutionary algorithms are used for model selection [Friedrichs and Igel, 2005]. Here, a certain number of configurations is randomly chosen and evaluated to form a so-called population. Then, inspired by biological evolution, new configurations are created using recombination and mutation of existing members of the population. Only the configurations that have shown the best classification performance are kept in the population. As with Bayesian optimization, this approach is able to find a trade-off between probing unexplored regions in the parameter space and choosing the most promising configurations.

8.3 METHOD

In this section, we will explain the classification pipeline of our system. At first, we want to describe the general workflow that is used to automatically identify engineered nanoparticles of a given type:

- Training data creation:
  1. The user makes a set of SEM images and corresponding metadata (pixel sizes) available to the system.
  2. The system segments these images as explained in Chapter 6 in order to detect the CPGs in them.
  3. The user manually labels the CPGs of known types with the corresponding classes (specific engineered nanoparticle class or the generic background particle class).
- Classifier training:
  1. The user tells the system to train the classification of a specific engineered nanoparticle type and which images shall be used for this.
  2. The system computes (only once per CPG) and saves the features of every CPG to be used as described in Chapter 7.
  3. The system assembles a training set of all CPGs that are in the images selected by the user and are either labeled as belonging to the class of the selected engineered nanoparticle type or the background particle class.
  4. The system automatically selects an appropriate subset of features and a matching classifier-parameter combination using the training set as we will explain in Section 8.3.2.
  5. The system trains the selected classifier using the chosen parameters and features on the training set.

- Classification:
  1. The user provides a set of SEM images containing engineered nanoparticles of a specific type and background particles and corresponding metadata (pixel sizes) to the system.
  2. The system segments these images as explained in Chapter 6 in order to detect the CPGs in them.
  3. The user tells the system to identify engineered nanoparticles of the specific type in the images.
  4. The system applies the learned classifier from the training step to the CPGs found in the images using the selected feature subset as we will describe in Section 8.3.1.
  5. The system presents the classification results and concentration estimates to the user.

The three steps can be independently initiated by the user. However, running a step requires that the previous steps have been performed at least once for the specific engineered nanoparticle type.

Note that the responsibilities of the user are restricted to the following three tasks:

- Supplying images and the corresponding metadata.
- Labeling the CPGs of the training set.
- Tell the system to learn or classify a specific engineered nanoparticle type.

The second task seems most demanding. However, this only needs to be done once per engineered nanoparticle type. In addition, different particle types are usually on separate images (see Section 4.3). Therefore, the manual classification is very easy. Also note that the user does not need to supply any parameters whatsoever.

We will begin with describing the classification process in Section 8.3.1. Then, Section 8.3.2 will explain our automatic model selection algorithm. Finally, Section 8.3.3 will address the task of estimating the classification performance of a given classifier-parameter combination.

8.3.1 *Classification*

In this section, we will describe the algorithms we use to classify the CPGs. Section 8.3.1.1 will explain the requirements our classifiers have to meet. In Section 8.3.1.2, we will present the classification algorithms themselves. Finally, Section 8.3.1.3 will describe the preprocessing steps each sample goes through before it is presented to the classification algorithm.

8.3.1.1 *Classifier Requirements*

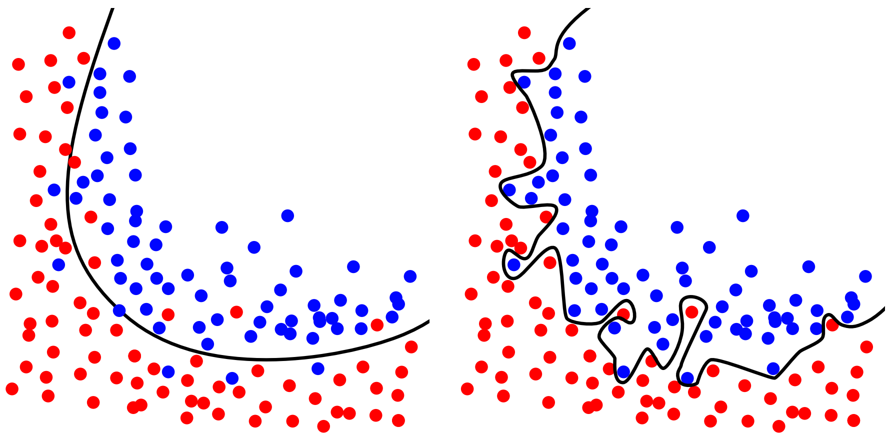
The choice of a classifier strongly depends on the application domain, the class distributions and the training set size. Our system shall be able to automatically adapt to new particle types. While the application domain stays the same, the class distribution and the training set size may depend on the particle type. Therefore, it would not be advisable to select a single classifier to be used by our system. We apply an automatic model selection approach, which we will explain in the next section. It is able to automatically select an appropriate classifier for a given classification task. However, we have to select a set of classifiers it can choose from.

While each new particle type brings with it a new classification problem, we can assume that they share the following two characteristics:

- The training sets will be relatively small because of the high costs of generating training images.
- The proportion of CPGs containing engineered nanoparticles in the training set will be relatively low because background particles are easily obtained and their images can be used in the training sets of multiple engineered nanoparticle types.

Both properties are reflected in our dataset, which we have described in Section 4.4. We have only 109 CPGs of Ag compared to 9279 background CPGs. The overall size of the dataset may seem large. However, we have very complex classes due to the variability of nanoparticle agglomerates. In addition, 109 positive samples are few compared to our feature count of almost 60.

The small training set size means that there is a high risk of the classifier to overfit the training set. Overfitting means that the classifier learns random variations of the training set that are not representative of the true class-conditional densities. Figure 8.1 illustrates this well. The graphics visualize an artificial classification problem with two classes and two features. The points represent the samples of the training set with their colors signifying their classes. Their horizontal and vertical positions represent their feature values. The two black lines depict the decision boundaries of two classifiers learned on the training set. The decision boundary in Figure 8.1a approximates the border between the true class-conditional densities well. The learned classifier would not classify every single sample in the training set correctly. However, it would presumably generalize well to unknown samples. The classifier corresponding to Figure 8.1b, on the other hand, has overfit the training set. It has produced a learned classifier which correctly classifies every single sample in the training set. However, it is obvious that the decision boundary does not match the true distributions.



(a) This decision boundary matches the true class distributions well.

(b) Here, the classifier has overfit the training set.

Figure 8.1: An artificial example of overfitting. The black lines represent the decision boundaries of two classifiers learned using the same training set and two features. The samples of the training set are represented by points. Their position is defined by their feature values interpreted as coordinates and their color represents their class. These works are derivatives of a graphic by Chabacano (`https://commons.wikimedia.org/wiki/File:Overfitting.svg`) licensed under CC BY-SA 4.0 (`https://creativecommons.org/licenses/by-sa/4.0/`).

The fact that the training set contains much more samples of one class than those of another is called class imbalance. In Section 8.2.3, we have briefly covered one problem of this condition. In addition, some types of classifiers tend to overrate the importance of the prevalent class. This leads to a high classification error of the minority class. Our classifier choice has also take this into consideration.

Complex classifiers are susceptible to overfitting on small training sets [Jain et al., 2000]. Therefore, simple classifiers are best suited for our system. In particular, we do not use neural networks because they tend to overfit the data [Zhang, 2000]. In addition, backpropagation neural networks perform poorly in the case of class imbalance [Sun et al., 2009]. In particular, we do not use convolutional neural networks because, such as conventional neural networks, they are quite complex. In addition, the image region used as the input to such a net has a fixed shape (usually a rectangle). This means that pixels not belonging to a CPG may influence its classification. Since nanoparticles and background particles may be next to each other on an image, this is undesirable. Similarly, decision trees, Bayesian classification, associative classifiers and K-nearest neighbor have been reported to perform badly under class imbalance [Sun et al., 2009].

8.3.1.2 *Classifiers*

We have chosen to use logistic regression with a ridge estimator [Le Cessie and Van Houwelingen, 1992] because it has only $n_\phi + 1$ internal parameters that are learned from the training set and it is based on a simple model. In addition, we can avoid overfitting further using regularization by means of the ridge estimator. As a second classifier, we use support vector machines (SVMs) because they are less prone to class imbalance [Sun et al., 2009] than other classifiers. Both are geometric classification approaches and directly construct the decision boundary. This is appropriate because the distributions of our features are unknown [Jain et al., 2000].

For the explanation of the classification algorithms, we will define the positive class, to which every CPG containing engineered nanoparticles belongs, as $Y_+ := 1$ and the negative class as $Y_- := 0$. This means that the class of a sample is equal to 0 or 1.

Logistic Regression models the posterior probability of a sample $\vec{X} \in \mathcal{X}$ belonging to class Y_+ as [Bishop, 2006]

$$h_R(\vec{X}) := \frac{1}{1 + e^{-\left(R_0 + \vec{R}^T \vec{X}\right)}}, \quad \forall\, R := \left(R_0, \vec{R}\right) \in \mathbb{R} \times \mathbb{R}^{n_\phi}, \vec{X} \in \mathcal{X}. \quad (8.1)$$

Here, $R := \left(R_0, \vec{R}\right) \in \mathbb{R} \times \mathbb{R}^{n_\phi}$ are the internal parameters of the trained classifier.

A maximum likelihood estimation, whose details can be read in the book by Bishop [2006, pp. 203ff.], leads to a minimization problem given the training set $T := \left\{\left(\vec{X}_1, Y_1\right), \ldots, \left(\vec{X}_{n_{\vec{X}}}, Y_{n_{\vec{X}}}\right)\right\}$:

$$\begin{aligned} R_{\theta_\lambda, w_{\text{cls}}, T} := \underset{R \in \mathbb{R} \times \mathbb{R}^{n_\phi}}{\arg\min} \sum_{i=1}^{n_{\vec{X}}} -w_{\text{cls}}(Y_i) \ln\left(1 - \left|h_R(\vec{X}_i) - Y_i\right|\right) + \theta_\lambda \left\|\vec{R}\right\|_2^2, \\ \forall\, T := \left\{\left(\vec{X}_1, Y_1\right), \ldots, \left(\vec{X}_{n_{\vec{X}}}, Y_{n_{\vec{X}}}\right)\right\} \sqsubseteq \mathcal{X} \times \mathcal{Y}, \\ \theta_\lambda \in \mathbb{R}_{\geqslant 0}, w_{\text{cls}} : Y \to \mathbb{R}_{>0}. \quad (8.2) \end{aligned}$$

The term $\theta_\lambda \left\|\vec{R}\right\|_2^2$ is not present in the original minimization problem. However, it is added to regularize the classification in order to avoid overfitting [Le Cessie and Van Houwelingen, 1992]. The parameter $\theta_\lambda \in \mathbb{R}_{\geqslant 0}$ controls the amount of regularization. The function $w_{\text{cls}} : Y \to \mathbb{R}_{>0}$ is also an addition to the original problem. It is a function that determines the weight of each class $Y \in \mathcal{Y}$. If class Y_+ is weighted higher than Y_-, then samples are more likely to be assigned to Y_+ compared to a classifier trained with equal weights. In the standard configuration, w_{cls} always returns 1.

The definition of the classifier $l_{\text{LR}, \theta_\lambda, w_{\text{cls}}} : \mathcal{D} \to \mathbb{R}^{\mathcal{X}}$ is then given by

$$\begin{aligned} l_{\text{LR}, \theta_\lambda, w_{\text{cls}}}(T)(\vec{X}) := h_{R_{\theta_\lambda, w_{\text{cls}}, T}}(\vec{X}), \\ \forall\, \theta_\lambda \in \mathbb{R}_{\geqslant 0}, T := \left\{\left(\vec{X}_1, Y_1\right), \ldots, \left(\vec{X}_{n_{\vec{X}}}, Y_{n_{\vec{X}}}\right)\right\} \sqsubseteq \mathcal{X} \times \mathcal{Y}, \\ \vec{X} \in \mathcal{X}, w_{\text{cls}} : Y \to \mathbb{R}_{>0}. \quad (8.3) \end{aligned}$$

Note that this definition returns a real value instead of a class. This value can be interpreted as the probability of sample $\vec{X}$ belonging to class Y_+. To obtain the most probable class of the sample, the return value has to be rounded to 0 or 1.

Our model selection algorithm, which we will explain in Section 8.3.2, can choose from the following values for the regularization parameter θ_λ: 0.001, 0.003, 0.01, 0.03, 0.1, 0.3 and 1. We have chosen relatively high values because our classification problem is prone to overfitting (see Section 8.3.1.1).

SVMs, in contrast to logistic regression, operate based on the premise that the decision boundary should be influenced only by the samples closest to it in feature space, the so-called support vectors [Bishop, 2006, pp. 326ff.]. The concept was first introduced by Vapnik and Lerner [1963] for cases where the two classes are linearly separable in feature space. The decision boundary in the form of a hyperplane was chosen such that the distance of the closest sample in the training set was maximized. However, the concept has been extended to work with non-separable data and non-linear decision boundaries.

We will not describe the exact calculations of an SVM in this thesis because, in our opinion, they add little value. The exact formulas are not very intuitive at first glance and their derivation is beyond the scope of this work. However, the details of SVMs can be found in the book by Bishop [2006, pp. 326ff.] or in the excellent lecture notes by Ng [2014].

For their calculations, SVMs use a so-called kernel $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. In the original algorithm, $K(\vec{X}, \vec{X}')$ was defined as $\vec{X}^T \vec{X}'$ corresponding to a linear decision boundary. Our model selection algorithm may choose from the following two kernels:

$$K_{\text{lin}}(\vec{X}, \vec{X}') := \vec{X}^T \vec{X}', \tag{8.4}$$

$$K_{\text{RBF},\theta_\gamma}(\vec{X}, \vec{X}') := e^{-\theta_\gamma \|\vec{X}-\vec{X}'\|_2^2}, \quad \forall\, \theta_\gamma \in \mathbb{R}_{>0}, \vec{X}, \vec{X}' \in \mathcal{X}. \tag{8.5}$$

The first is called linear kernel and corresponds to the original case of a linear decision boundary. The second is called radial basis function (RBF) kernel and generates non-linear decision boundaries. It has a parameter $\theta_\gamma \in \mathbb{R}_{>0}$, which determines the area of influence of each support vector. Smaller values lead to simpler decision boundaries. Our model selection algorithm can choose these values for θ_γ: 0.001, 0.003, 0.01, 0.03 and 0.1. These are relatively small corresponding to simple decision boundaries. This is reasonable because, as explained in the previous section, our problem calls for a simple classifier.

In addition, an SVM has a regularization parameter $\theta_C \in \mathbb{R}_{>0}$. However, in contrast to θ_λ in logistic regression, a small value of θ_C corresponds to a high degree of regularization. The model selection can select one of these values for it: 1, 3, 10, 30, 100, 300 and 1000. Again, this corresponds to a relatively high degree of regularization.

Furthermore, we use the sequential minimal optimization algorithm to train the SVMs in our system [Platt, 1998]. We will represent an SVM classifier as $l_{\mathrm{SVM},K,\theta_C,w_{\mathrm{cls}}} : \mathcal{D} \to \mathcal{Y}^{\mathcal{X}}$, where $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is its kernel, $\theta_C \in \mathbb{R}_{>0}$ the regularization parameter and $w_{\mathrm{cls}} : \mathcal{Y} \to \mathbb{R}_{>0}$ the class weight function.

8.3.1.3 *Preprocessing*

Before the samples are passed to the classifier, they are preprocessed. The goal of this step is to optimize the classification result. The preprocessing performs three tasks:

- Dimensionality reduction
- Feature scaling
- Weighting and resampling

Dimensionality reduction reduces the number of features. Then, feature scaling normalizes the feature values to equalize the impacts of all features. Finally, weighting and resampling increase the impact of positive samples on the learned classifier. The steps will be explained in more detail in the remainder of this section.

We have discussed dimensionality reduction in Sections 8.1.2 and 8.2.2. As mentioned before, there are two variants of it: feature extraction and feature selection. Because of the high complexity of our classification problem and the lack of many positive samples, our classification pipeline should not contain unnecessary complexity (see Section 8.3.1.1). Therefore, we have opted not to use feature extraction, which would introduce additional features to our classification problem. Instead, we use feature selection to reduce the classifier complexity. A subset $\theta_\phi \subseteq \{1, \ldots, n_\phi\}$ of feature indices is selected and only the corresponding features are visible to the classifier. The selection itself is performed by our automatic model selection method, which we will explain in the next section.

Feature scaling rescales the feature values in order to guarantee that the impact of different features on the learned classifier is approximately equal. If, for example, the values of a feature are very small compared to other features, they have to be multiplied by a large factor in logistic regression in order to have an impact. However, the regularization would prevent a large factor. This way, the impact of this feature is smaller than the influence of features with large value ranges. To eliminate this effect, our system normalizes the features so that their values have a mean of 0 and a variance of 1 based on the feature values in the training set.

As mentioned in Section 8.3.1.1, we assume that there are fewer CPGs containing engineered nanoparticles than ones consisting of background particles in the training set. This may lead to the fact that the learned classifier is more likely to classify samples as belonging to the negative class than the positive class. This behavior can result in a poor classification performance, especially for very skewed class distributions in the training set. The automatic model selection algorithm can use two methods to reduce this effect. The first is weighting. The weight function $w_{\text{cls}} : Y \to \mathbb{R}_{>0}$ used in the training of logistic regression classifiers (see Equation (8.2)) and SVMs usually returns 1 for each class. However, returning 1 for Y_- and $\theta_w \in \mathbb{R}_{>0}, \theta_w > 1$ for Y_+ can cause the learned classifier to be more likely to classify samples as engineered nanoparticles. The automatic model selection method can choose the following values for θ_w: 1, 3, 10, 30 and 100. 1 corresponds to no weighting and 100 is slightly higher than the highest ratio of background CPGs to positive samples in our dataset.

Another good solution is to resample the training data for the classifier [Sun et al., 2009]. Forman and Cohen [2004] say that "performance can be greatly improved by non-random sampling that somewhat favors the minority class on tasks with skewed class distributions". It works by removing samples of the majority class (undersampling) from and adding artificial samples of the minority class (oversampling) to the training set. We use the synthetic minority oversampling technique (SMOTE) by Chawla et al. [2002] to oversample the positive class because, according to He and Garcia [2009], it "is a powerful method that has shown a great deal of success in various applications". In addition, we randomly undersample the negative class. The resampling method is explained in Algorithm 8.1. Note that we use 5 nearest neighbors in the SMOTE algorithm as proposed in the original paper. The automatic model selection method can choose the values 0 %, 30 %, 100 %, 300 % and 1000 % for θ_N and 1, 3, 10, 30 and ∞ for θ_{MS}. Here, $\theta_N = 0$ and $\theta_{MS} = \infty$ would mean that neither over- nor undersampling is performed. The maximum oversampling percentage of $\theta_N = 1000$ would mean that 11 times as many positive samples are used for the classifier training compared to the original count, which is a considerable amount. A maximum spread of $\theta_{MS} = 1$ means that an equal number of positive and negative samples are used for the training. We have (apart from ∞) not included values larger than 30 because, in our dataset, none of the ratios between positive and negative samples exceeds 100.

The whole pipeline including preprocessing and classifiers can be thought of as a single classifier $l_{\text{full},\theta} : \mathcal{D} \to \mathcal{Y}^{\mathcal{X}}$ where the parameter θ contains all parameters of the preprocessing steps and classifiers. The parameter has the structure $(\theta_\phi, \theta_w, \theta_N, \theta_{MS}, \theta_l)$, where θ_l can have one of the following structures:

- (l_{LR}, θ_λ).
- $(l_{SVM}, (K_{\text{lin}}), \theta_C)$.
- $(l_{SVM}, (K_{RBF}, \theta_\gamma), \theta_C)$.

Algorithm 8.2 shows how $l_{\text{full},\theta}(T)(\vec{X})$ is computed.

Algorithm 8.1 : *Resample*$(T, \theta_N, \theta_{MS})$

```
Input : A trainig set $T := \{(\vec{X}_1, Y_1), \dots, (\vec{X}_{n_{\vec{X}}}, Y_{n_{\vec{X}}})\} \sqsubseteq \mathcal{X} \times \mathcal{Y}$, the
        oversampling percentage $\theta_N \in \mathbb{R}_{\geqslant 0}$ in percent and the
        maximum spread of the classes $\theta_{MS} \in [1, \infty]$.
Output : The resampled training set $T'$.
// Split the samples into positive and negative ones:
1  $\mathcal{X}_+ := \bigcup_{\substack{i=1,\dots,n_{\vec{X}} \\ Y_i = Y_+}} \{\vec{X}_i\}$.
2  $\mathcal{X}_- := \bigcup_{\substack{i=1,\dots,n_{\vec{X}} \\ Y_i = Y_-}} \{\vec{X}_i\}$.
3  $n_{\vec{X},\mathrm{art}} := \lfloor \frac{\theta_N |\mathcal{X}_+|}{100} \rfloor$. // The number of artificial positive
     samples to be created.
4  $\mathcal{X}_{\mathrm{art}} := \emptyset$. // The multiset of artificial positive samples.
5  $\mathcal{X}'_+ := \mathcal{X}_+$. // The positive samples which have not yet been
     used to create an artificial sample.
// Create some artificial positive samples:
6  while $n_{\vec{X},art} > 0$ do
7      if $\mathcal{X}'_+ = \emptyset$ then
8          $\mathcal{X}'_+ := \mathcal{X}_+$.
9      end
10     Randomly choose $\vec{X}$ from $\mathcal{X}'_+$ with a uniform distribution.
11     Randomly choose $\vec{X}'$ from the 5 nearest neighbors of $\vec{X}$ in
       $\mathcal{X}_+ \setminus \{\vec{X}\}$ with a uniform distribution.
12     Randomly choose $\tau$ from the interval $[0, 1]$ with a uniform
       distribution.
13     $\vec{X}_{\mathrm{art}} := \vec{X} + \tau(\vec{X}' - \vec{X})$. // A new artificial sample between
          the two original samples in feature space.
14     $\mathcal{X}_{\mathrm{art}} := \mathcal{X}_{\mathrm{art}} \cup \{\vec{X}_{\mathrm{art}}\}$.
15     $n_{\vec{X},\mathrm{art}} := n_{\vec{X},\mathrm{art}} - 1$.
16 end
17 $\mathcal{X}_+ := \mathcal{X}_+ \cup \mathcal{X}_{\mathrm{art}}$.
// Undersample the negative class:
18 $n_{\vec{X},\max} := \lfloor \theta_{MS} |\mathcal{X}_+| \rfloor$. // The maximum number of negative
     samples.
19 while $|\mathcal{X}_-| > n_{\vec{X},max}$ do
20     Randomly choose $\vec{X}$ from $\mathcal{X}_-$ with a uniform distribution.
21     $\mathcal{X}_- := \mathcal{X}_- \setminus \{\vec{X}\}$.
22 end
23 return $T' := \{(\vec{X}, Y_+) \mid \vec{X} \in \mathcal{X}_+\} \cup \{(\vec{X}, Y_-) \mid \vec{X} \in \mathcal{X}_-\}$.
```

Algorithm 8.2 : *ClassificationPipeline*$(\theta, T, \vec{X})$

```
   Input : The parameter values $\theta$, a trainig set
           $T := \{(\vec{X}_1, Y_1), \dots, (\vec{X}_{n_{\vec{X}}}, Y_{n_{\vec{X}}})\} \sqsubseteq \mathcal{X} \times \mathcal{Y}$ and a sample
           $\vec{X} \in \mathcal{X}$ to be classified.
   Output : The class $l_{\text{full},\theta}(T)(\vec{X})$ of $\vec{X}$ predicted by the learned
            classifier $l_{\text{full},\theta}(T)$.
1  $(\theta_\phi, \theta_w, \theta_N, \theta_{MS}, \theta_l) := \theta$.
   // DIMENSIONALITY REDUCTION:
   // Only retain the features whose indices are in $\theta_\phi$:
2  for $i = 1, \dots, n_{\vec{X}}$ do
3      $\vec{X}_i := \vec{X}_i(\theta_\phi)$, where $\vec{X}_i(\theta_\phi) \in \mathbb{R}^{|\theta_\phi|}$ is defined as a vector
       containing only the entries of $\vec{X}_i$ with the indices in $\theta_\phi$.
4  end
5  $\vec{X} := \vec{X}(\theta_\phi)$.
   // FEATURE SCALING:
6  for $j = 1, \dots, |\theta_\phi|$ do
7      $\bar{X} := \frac{\sum_{i=1}^{n_{\vec{X}}} \vec{X}_i(j)}{n_{\vec{X}}}$. // Mean value of the feature with index j.
8      $\sigma_X := \sqrt{\frac{\sum_{i=1}^{n_{\vec{X}}} (\vec{X}_i(j) - \bar{X})^2}{n_{\vec{X}} - 1}}$. // Standard deviation of the
          feature with index j.
9      $\vec{X}_i(j) := \frac{\vec{X}_i(j) - \bar{X}}{\sigma_X}, \quad \forall i = 1, \dots, n_{\vec{X}}$.
10     $\vec{X}(j) := \frac{\vec{X}(j) - \bar{X}}{\sigma_X}$.
11 end
   // RESAMPLING:
12 $T' :=$ Resample$(\{(\vec{X}_1, Y_1), \dots, (\vec{X}_{n_{\vec{X}}}, Y_{n_{\vec{X}}})\}, \theta_N, \theta_{MS})$. // See
      Algorithm 8.1.
   // WEIGHTING:
13 $w_{\text{cls}}(Y) := \begin{cases} \theta_w, & \text{if } Y = Y_+, \\ 1, & \text{else,} \end{cases} \quad \forall Y \in \mathcal{Y}$. // The weight function
      passed to the classifier.
   // TRAINING AND CLASSIFICATION:
   // Train and classify using the classifier, classifier
      parameters and, in the case of an SVM, kernel from $\theta_l$
      and the weight function $w_{\text{cls}}$ defined above:
14 return $l_{\mathit{full},\theta}(T)(\vec{X}) := l_{\theta_l, w_{cls}}(T')(\vec{X})$.
```

8.3.2 *Automatic Model Selection*

In the previous sections, we have presented our classification pipeline. However, this pipeline has several parameters including classifiers, kernels and numerical parameters which have to be chosen. In addition, an appropriate subset of features has to be found each time our system is trained. Since our goal is to build a fully automatic solution, the user cannot be responsible for making these decisions. Therefore, we have to find automatic solutions to perform the following two tasks for a given classification problem:

FEATURE SELECTION Choose the subset of features that is passed to the classifier.

MODEL SELECTION Determine a good classifier-parameter combination.

Forman and Cohen [2004] recommend that "feature selection should not be decoupled from the model selection task". The reason is that a certain classifier-parameter combination may work well with a given feature subset while others may not. This means that we have to find a method that selects a feature subset and a classifier-parameter combination at the same time instead of performing these steps consecutively.

We have chosen to use an evolutionary algorithm to perform both feature selection and model selection at the same time. As explained in Section 8.2.3, such an approach is superior to grid-search and random search because it makes use of information about the previously evaluated classifier-parameter combinations. We have opted against Bayesian optimization because it tries to model the function from parameter values to the classifier error by making assumptions about its distribution [Thornton et al., 2013]. However, we know too little about its distribution to make such assumptions.

Friedrichs and Igel [2005] apply an evolution strategy to find optimal parameters for SVMs. However, their approach can only handle real-valued parameters. In particular, it cannot deal with categorical parameters such as which features are used.

Several genetic algorithms have been successfully applied to feature selection [Siedlecki and Sklansky, 1989; Kuncheva, 1993; Yang and Honavar, 1998; Handels et al., 1999; Zheng et al., 1999; Raymer et al., 2000; Cantú-Paz, 2002; Fröhlich et al., 2002; Miller et al., 2003; Jong et al., 2004]. We use such an approach to not only perform feature selection but also to do model selection at the same time. Genetic algorithms are modeled after natural selection and evolution. We will not elaborate on their theory more than is necessary to explain our implementation. More details on the topic can be found in the book by Sivanandam and Deepa [2008].

8.3.2.1 *The Model Selection Algorithm*

Our implementation is able to work with hierarchical dependencies where the parameters to be used depend on the choice of classifier or kernel. Algorithm 8.3 shows its steps. The algorithm works by first randomly generating $n_\theta \in \mathbb{N}_{\geqslant 2}$ classifier-parameter combinations and estimating their classification performances to form an initial population. Then, repeatedly, two configurations are randomly picked from this population and recombined to form a new classifier-parameter combination, which is then randomly mutated. The new configuration is introduced into the population and the worst one is removed.

We have made most of the design decisions of our particular implementation with respect to the fact that the performance estimation takes relatively long and that the user shall be able to specify a running time the algorithm may take. Therefore, in each generation, only one configuration is added to the population. Otherwise, the running time could be exceeded by the time it takes to estimate the classification performances of multiple configurations. In addition, we have chosen a relatively small population size, namely 30, which shows the best performance in the experiments by Grefenstette [1986] for the case of time-constrained applications. While large populations sizes are generally favorable if the running time is not limited, the disadvantage of such a genetic algorithm is that it takes considerable time to converge to a good solution [Grefenstette, 1986]. A small population size, on the other hand, is able to produce a relatively good solution in a time-constrained setup.

In order to prevent overfitting, the parameter search is stopped before the time has been used up if the performances of the parameter population are too similar. This suggests that the results are in the range of the best achievable performance and that any further improvements are likely due to chance. We have implemented the notion of population similarity in the form of a function ι, which takes a parameter population Θ and returns a number in the interval $[0, 1]$:

$$\iota(\Theta) := \frac{\min\limits_{(\theta,\rho)\in\Theta} \rho}{\max\limits_{(\theta,\rho)\in\Theta} \rho}. \tag{8.6}$$

It simply calculates the ratio of the worst and the best classification performances in the population. The parameter search is stopped if the function value is greater than the maximum population similarity parameter $\iota_{\max} := 0.99$.

8.3.2.2 *Parent Selection*

Algorithm 8.4 describes how the random parent selection is done. Configurations with a high estimated classification performance are

Algorithm 8.3 : *ParameterSelection*$(T, \omega, t, n_\theta := 30, \iota_{max} := 0.99)$

Input : A trainig set $T \sqsubseteq \mathcal{X} \times \mathcal{Y}$, the evaluation metric $\omega : \mathbb{N}^4 \to \mathbb{R}$ to be used, the time t that this algorithm may take to run, the desired size of the population $n_\theta \in \mathbb{N}_{\geqslant 2}$ and the maximum population similarity $\iota_{max} \in]0, 1]$. The default values of n_θ and ι_{max} are 30 and 0.99, respectively.

Output : Good parameter values θ_{best}, which can be used as an input for *ClassificationPipeline*() (see Algorithm 8.2).

```
// Create an initially empty population of parameter
   configurations θ and their estimated classification
   performances ρ ∈ ℝ:
1  $\Theta := \emptyset$
2  repeat
      // Generate new parameters $\theta_{new}$:
3     if $|\Theta| < n_\theta$ then
         // The population has not yet reached its desired
            size. Randomly create new parameters:
4        $\theta_{new} :=$ GenerateRandomParameters(). // See
            Algorithm 8.5.
5     else
         // Randomly select two parents, which are
            recombined to form new parameters:
6        $(\theta_1, \rho_1) :=$ RandomlySelectParent($\Theta$). // See
            Algorithm 8.4.
7        $(\theta_2, \rho_2) :=$ RandomlySelectParent($\Theta \setminus \{(\theta_1, \rho_1)\}$).
         // Generate new parameters $\theta_{new}$ by random
            recombination and mutation of $\theta_1$ and $\theta_2$ and add
            them to $\Theta$:
8        $\theta_{new} :=$ RandomlyRecombineParameters($\theta_1, \theta_2$). // See
            Algorithm 8.6.
9        $\theta_{new} :=$ RandomlyMutateParameters($\theta_{new}$). // See
            Algorithm 8.8.
10       $\rho_{new} :=$ EstimateClassificationPerformance($\theta_{new}, T, \omega$).
            // See Algorithm 8.9 on Page 160.
11       $\Theta := \Theta \cup \{(\theta_{new}, \rho_{new})\}$
12    end
13    while $|\Theta| > n_\theta$ do
         // Remove the element from Θ with the lowest
            classification performance:
14       $\Theta := \Theta \setminus \left\{ \arg\min_{(\theta,\rho) \in \Theta} \rho \right\}$.
15    end
16 until $|\Theta| = n_\theta \wedge \iota(\Theta) > \iota_{max}$ or the processing time t has been used up.
   // Let $(\theta_{best}, \rho_{best})$ be the element of Θ with the highest
      classification performance:
17 $(\theta_{best}, \rho_{best}) := \arg\max_{(\theta,\rho) \in \Theta} \rho$.
18 return $\theta_{best}$.
```

Algorithm 8.4 : *RandomlySelectParent*$(\Theta, \zeta := 0.2)$

```
   Input : A set $\Theta = \{(\theta_1, \rho_1), \dots, (\theta_{n_\theta}, \rho_{n_\theta})\}$ of parameter
           configurations $\theta_i$ and their estimated classification
           performances $\rho_i$ and the minimum selection probability
           $\zeta \in [0, 1]$, measured relative to the selection probability of
           the configuration with the highest classification
           performance. The default value of $\zeta$ is 0.2.
   Output : A random element $(\theta, \rho) \in \Theta$.
   // The minimum and maximum estimated classification
      performances:
 1 $\rho_{\min} := \min_{i:=1,\dots,n_\theta} \rho_i$.
 2 $\rho_{\max} := \max_{i:=1,\dots,n_\theta} \rho_i$.
   // Compute the probabilities of the elements to be
      chosen:
 3 if $\rho_{min} = \rho_{max}$ then
 4     for $i := 1, \dots, n_\theta$ do
 5         $\rho'_i := 1$.
 6     end
 7 else
       // Rescale the classification performance measures in
          such a way that the worst one has a value $\zeta$ times
          that of the best one:
 8     for $i := 1, \dots, n_\theta$ do
 9         $\rho'_i := \zeta + (1 - \zeta)\frac{\rho_i - \rho_{\min}}{\rho_{\max} - \rho_{\min}}$.
10     end
11 end
   // Rescale the values so that their sum is 1:
12 for $i := 1, \dots, n_\theta$ do
13     $\rho''_i := \frac{\rho'_i}{\sum_{j=1}^{n_\theta} \rho'_j}$.
14 end
   // Randomly pick a configuration $\theta_i$ with the selection
      probabilities being equal to the $\rho''_i$ and return it:
15 Randomly choose $\tau$ from the interval $[0, 1]$ with a uniform
   distribution.
16 $i := 1$.
17 while $\tau > \rho''_i$ do
18     $\tau := \tau - \rho''_i$.
19     $i := i + 1$
20 end
21 return $(\theta_i, \rho_i)$.
```

more likely to be chosen as a parent. The probabilities depend linearly on the estimated classification performance measures and the probability of the worst configuration is ζ as high as that of the best one. We set $\zeta := 0.2$. In a case where only one configuration in the population of 30 has a classification performance larger than 0, this means that this classifier-parameter combination is selected as a parent of about every fourth newly tested configuration on average. This increases the chance that another configuration with a performance measure larger than 0 is found. Such a scenario seems unlikely. However, in the case of high class imbalance, which we have discussed in Sections 8.2.3 and 8.3.1.1, it is relatively likely that a classifier assigns the majority class to every single sample in the test set.

8.3.2.3 *Random Parameter Generation*

The generation of random parameters is explained in Algorithm 8.5. The configuration θ returned by the algorithm has the same structure as explained in Section 8.3.1.3. At first, the indices of the features to be used for the classification are chosen. Since we do not know the optimal number of features for the given problem, the algorithm first randomly chooses the probability τ of a given feature to be included. In the next step, each feature index is added to θ_ϕ with a probability of τ. This means that for a given τ, on average $\tau \cdot n_\phi$ feature indices are added to θ_ϕ.

Because of the relatively high number of parameters that need to be optimized at the same time, the search space of the genetic algorithm is very large. Therefore, we have decided to use our knowledge of the classification problem to choose some sensible values for each parameter beforehand. Thus, the algorithm can choose among these value instead of all possible values for a given feature. This reduces the size of the search space, while the algorithm is still able to test a range of values for each feature. The values listed in Algorithm 8.5 coincide with those given in the text where the corresponding parameters have been introduced.

Finally, the algorithm chooses with equal probabilities between logistic regression and an SVM and, if an SVM has been chosen, between a linear kernel and an RBF kernel.

8.3.2.4 *Recombination*

After the population has its desired size, new configurations are generated by recombining two classifier-parameter combinations using Algorithm 8.6. It uses a helper algorithm called `RandomlyChoose`(), which returns its first argument with a probability of q_{θ_1} and otherwise its second argument. It is listed in Algorithm 8.7.

`RandomlyRecombineParameters` generates a new configuration by randomly choosing for each parameter to either use the parameter

Algorithm 8.5 : *GenerateRandomParameters*()

```
Output : Random parameter values $\theta$ for
         ClassificationPipeline() (see Algorithm 8.2).
1  $\theta_\phi := \emptyset$. // The indices of the selected features.
2  Randomly choose $\tau$ from $[0, 1]$ with a uniform distribution.
      // The probability of a given feature index being
      added to $\theta_\phi$.
3  for $i := 1, \ldots, n_\phi$ do
4      Randomly choose $\tau_i$ from $[0, 1]$ with a uniform distribution.
5      $\theta_\phi := \begin{cases} \theta_\phi \cup \{i\} & \text{if } \tau_i < \tau, \\ \theta_\phi, & \text{else.} \end{cases}$
6  end
7  Randomly choose $\theta_w$ from $\{1, 3, 10, 30, 100\}$ with a uniform
   distribution.
8  Randomly choose $\theta_N$ from $\{0, 30, 100, 300, 1000\}$ with a uniform
   distribution.
9  Randomly choose $\theta_{MS}$ from $\{1, 3, 10, 30, \infty\}$ with a uniform
   distribution.
   // Choose a classifier and its parameters:
10 Randomly choose $\tau_l$ from $[0, 1]$ with a uniform distribution.
11 if $\tau_l < 0.5$ then
       // Logistic regression.
12     Randomly choose $\theta_\lambda$ from $\{0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1\}$
       with a uniform distribution.
13     $\theta_l := (l_{LR}, \theta_\lambda)$.
14 else
       // SVM.
15     Randomly choose $\theta_C$ from $\{1, 3, 10, 30, 100, 300, 1000\}$ with a
       uniform distribution.
       // Choose a kernel:
16     Randomly choose $\tau_K$ from $[0, 1]$ with a uniform distribution.
17     if $\tau_K < 0.5$ then
           // Linear kernel.
18         $\theta_l := (l_{SVM}, (K_{lin}), \theta_C)$.
19     else
           // RBF kernel.
20         Randomly choose $\theta_\gamma$ from $\{0.001, 0.003, 0.01, 0.03, 0.1\}$ with
           a uniform distribution.
21         $\theta_l := (l_{SVM}, (K_{RBF}, \theta_\gamma), \theta_C)$.
22     end
23 end
24 return $\theta := (\theta_\phi, \theta_w, \theta_N, \theta_{MS}, \theta_l)$.
```

Algorithm 8.6 : *RandomlyRecombineParameters*$(\theta_1, \theta_2, q_{\theta_1} := 0.5)$

```
Input : Two parameter configurations $\theta_1$ and $\theta_2$ and the
        probability $q_{\theta_1} \in [0, 1]$ of choosing a parameter value
        from $\theta_1$ over the one from $\theta_2$. The default value of $q_{\theta_1}$ is
        0.5.
Output : Parameter values $\theta$ that are a combination of $\theta_1$ and $\theta_2$.
1  $(\theta_{\phi,1}, \theta_{w,1}, \theta_{N,1}, \theta_{MS,1}, \theta_{l,1}) := \theta_1$.
2  $(\theta_{\phi,2}, \theta_{w,2}, \theta_{N,2}, \theta_{MS,2}, \theta_{l,2}) := \theta_2$.
3  $\theta_\phi := \emptyset$. // The indices of the selected features.
4  for $i := 1, \dots, n_\phi$ do
5      Randomly choose $\tau_i$ from $[0, 1]$ with a uniform distribution.
6      $\theta_\phi := \begin{cases} \theta_\phi \cup (\theta_{\phi,1} \cap \{i\}), & \text{if } \tau_i < q_{\theta_1}, \\ \theta_\phi \cup (\theta_{\phi,2} \cap \{i\}), & \text{else.} \end{cases}$
7  end
8  $\theta_w := RandomlyChoose(\theta_{w,1}, \theta_{w,2}, q_{\theta_1})$.
9  $\theta_N := RandomlyChoose(\theta_{N,1}, \theta_{N,2}, q_{\theta_1})$.
10 $\theta_{MS} := RandomlyChoose(\theta_{MS,1}, \theta_{MS,2}, q_{\theta_1})$.
   // Extract the used classifiers:
11 $l_1 := \theta_{l,1}(1)$.    $l_2 := \theta_{l,2}(1)$.
12 if $l_1 \neq l_2$ then
       // The configurations use different classifiers.
13     $\theta_l := RandomlyChoose(\theta_{l,1}, \theta_{l,2}, q_{\theta_1})$.
14 else if $l_1 = l_{LR}$ then
       // Both use logistic regression. Extract $\theta_\lambda$:
15     $(l_1, \theta_{\lambda,1}) := \theta_{l,1}$.    $(l_2, \theta_{\lambda,2}) := \theta_{l,2}$.
16     $\theta_\lambda := RandomlyChoose(\theta_{\lambda,1}, \theta_{\lambda,2}, q_{\theta_1})$.
17     $\theta_l := (l_{LR}, \theta_\lambda)$.
18 else
       // Both use SVMs. Extract $\theta_K$ and $\theta_C$:
19     $(l_1, \theta_{K,1}, \theta_{C,1}) := \theta_{l,1}$.    $(l_2, \theta_{K,2}, \theta_{C,2}) := \theta_{l,2}$.
20     $\theta_C := RandomlyChoose(\theta_{C,1}, \theta_{C,2}, q_{\theta_1})$.
       // Extract the used kernels:
21     $K_1 := \theta_{K,1}(1)$.    $K_2 := \theta_{K,2}(1)$.
22     if $K_1 \neq K_2$ then
           // The configurations use different kernels.
23         $\theta_K := RandomlyChoose(\theta_{K,1}, \theta_{K,2}, q_{\theta_1})$.
24     else if $K_1 = K_{lin}$ then
           // Both use the linear kernel.
25         $\theta_K := (K_{lin})$.
26     else
           // Both use an RBF kernel. Extract $\theta_\gamma$:
27         $(K_1, \theta_{\gamma,1}) := \theta_{K,1}$.    $(K_2, \theta_{\gamma,2}) := \theta_{K,2}$.
28         $\theta_\gamma := RandomlyChoose(\theta_{\gamma,1}, \theta_{\gamma,2}, q_{\theta_1})$.
29         $\theta_K := (K_{RBF}, \theta_\gamma)$.
30     end
31     $\theta_l := (l_{SVM}, \theta_K, \theta_C)$.
32 end
33 return $\theta := (\theta_\phi, \theta_w, \theta_N, \theta_{MS}, \theta_l)$.
```

```
Algorithm 8.7 : RandomlyChoose($\theta_1, \theta_2, q_{\theta_1}$)
    Input : Two parameters $\theta_1$ and $\theta_2$ and the probability $q_{\theta_1} \in [0, 1]$
            of choosing $\theta_1$.
    Output : Either $\theta_1$ with a probability of $q_{\theta_1}$ or $\theta_2$ with a
             probability of $1 - q_{\theta_1}$, picked randomly.
1   Randomly choose $\tau$ from $[0, 1]$ with a uniform distribution.
2   $\theta := \begin{cases} \theta_1, & \text{if } \tau < q_{\theta_1}, \\ \theta_2, & \text{else.} \end{cases}$
3   return $\theta$.
```

value from the first or the second parent configuration with equal probabilities. Special cases are the selected features and the used classifiers. Each feature is represented as a binary parameter, which determines if the feature is used in the classification. For each of these, the algorithm randomly picks the decision if this feature shall be used from one of the two configurations.

The classifiers and their specific parameters are recombined as follows: If both configurations use the same classifier, then the default procedure is used to choose the parameter values specific to this classifier. If this is not the case, then the used classifier along with the values of its specific parameters are simply taken from one of the configurations. The same procedure is used for the kernels in the case of SVM classifiers.

8.3.2.5 *Mutation*

After a new configuration is generated by recombining two parents, it is randomly mutated by `RandomlyMutateParameters`(), which is listed in Algorithm 8.8. The mutation is simply done by generating a ran-

```
Algorithm 8.8 : RandomlyMutateParameters($\theta, q_{mut} := 0.01$)
    Input : Parameters $\theta$ and the mutation probability $q_{mut} \in [0, 1]$.
            The default value of $q_{mut}$ is 0.01.
    Output : A mutated version of the parameters $\theta$.
1   $\theta' :=$ GenerateRandomParameters(). // See Algorithm 8.5.
2   $\theta :=$ RandomlyRecombineParameters($\theta, \theta', 1 - q_{mut}$). // See
      Algorithm 8.6.
3   return $\theta$.
```

dom set of parameters and recombining the original configuration with this new one. The major difference to the standard recombination step, however, is that the probability of taking a parameter value from the new configuration is $q_{mut} \in [0, 1]$. This variable is called mutation probability and we use a value of 0.01, which shows the best results for time-constrained applications in the experiments performed by Grefenstette [1986].

We have tried to choose sensible parameter values for our genetic algorithm based on experiments performed by others on data that is not directly comparable to ours. However, evaluating these parameter values on our data is impossible for two reasons:

- To do such an evaluation, we would have to split our dataset so that we could test different parameters on one subset and evaluate the algorithm on the another subset. Otherwise, the obtained results would be biased as explained in Section 8.2.3. This would mean that even fewer samples could be used for the evaluation of the classification pipeline.

- Because of the long time required to do the performance evaluation of a single classifier-parameter combination, a single run of the automatic model selection method takes about 10 hours. This means that the evaluation of a single set of the parameter values for the genetic algorithm using 10-fold cross-validation would take about 100 hours. If we wanted to evaluate only 10 combinations, this would in turn take approximately 1000 hours. This process would then need to be done for each of the three particle types. All of this would take about 125 days. If we wanted to make the evaluation a bit more thorough and test three values for each of the three parameters, the whole procedure would run for almost one year.

These circumstances are not optimal. However, Grefenstette [1986] has found that "while it is possible to optimize GA control parameters, very good performance can be obtained with a range of GA control parameter settings". This means that our automatic model selection method should be able to find good classifier-parameter combinations for all particle types.

8.3.3 *Classification Performance Estimation*

The final part of the automatic parameter selection method that still needs to be explained is the classification performance estimation of a given classifier-parameter combination. We have already devoted the major part of Section 8.2.3 on how this can be done. We have said that three question need to be answered:

- How do we evaluate the quality of a classifier-parameter combination?

- How do we estimate the error of a given classifier-parameter combination in order to calculate the evaluation metric?

- How do we select classifier-parameter combinations to be tested?

The third question has been addressed in the previous section. This leaves the first two still to be answered.

For the error estimation, we have established in Section 8.2.3 that bootstrapping and cross-validation are the only sensible options for our problem because of the limited availability of training samples. Kohavi [1995] compares these two methods and finds that 10-fold cross-validation shows the best results compared to bootstrapping. In addition, he finds that it performs better than cross-validation using a higher number of folds. Therefore, we have decided to use 10-fold cross-validation. We have opted against repeated cross-validation because it would reduce the number of configurations the model selection method would be able to test by a factor equal to the number of repetitions.

Simply using standard cross-validation on our data would likely produce biased results. This has two reasons:

- Two CPGs from the same image may end up in different folds. This means that a classifier trained using one of them is tested on the other CPG. However, all CPGs in an image share the same contrast- and brightness-settings, which could be reflected in features such as the estimated electron count. This gives the classifier an unfair advantage, which it would not have in a real usage scenario.

- As explained in Section 4.4, our dataset usually contains two images of different magnifications for each selected position on the substrate. This means that it may contain multiple images of the same CPG. Again, if these would end up in different folds, the classifier could learn the specific properties of this CPG and apply that knowledge to the version of it in the test set.

This means that, in order to do a fair evaluation, all CPGs from a single position on the substrate need to be grouped into the same fold. This fact not only makes the used cross-validation algorithm slightly more complicated. It also prevents us from using stratified cross-validation, which has the additional requirement that the class proportions of every fold are approximately equal.

Algorithm 8.9 shows the cross-validation process that is tailored to these circumstances. The only difference to standard cross-validation is that all samples of a position on the substrate are assigned to a fold at once instead of one sample at a time.

During the cross-validation, the number of true positives, false positives, true negatives and false negatives are recorded. These terms are defined as follows:

TRUE POSITIVE A sample that is assigned to the positive class by the learned classifier and truly belongs to the positive class.

FALSE POSITIVE A sample that is assigned to the positive class by the learned classifier but in fact belongs to the negative class.

Algorithm 8.9 : *EstimateClassificationPerformance*$(\theta, T, \omega, n_F := 10)$

```
Input : Parameters $\theta$ to be used for the classification, a dataset
        $D \sqsubseteq \mathcal{X} \times \mathcal{Y}$ to test the parameters on, the evaluation
        metric $\omega : \mathbb{N}^4 \to \mathbb{R}$ to be used and the number of folds
        $n_F \in \mathbb{N}_{\geqslant 2}$ to be used for the cross-validation. The default
        value of $n_F$ is 10.
Output : The estimated classification performance $\rho \in \mathbb{R}$ of the
         parameters $\theta$ measured by the evaluation metric $\omega$
         using a $n_F$-fold cross-validation on the dataset $D$.
1  Let $D'_1, \dots, D'_m \sqsubseteq \mathcal{X} \times \mathcal{Y}$ be the elements of $D$ split by their
   substrate position so that $\bigcup_{i=1}^{m} D'_i = D$.
2  $\mathcal{D}' := \{D'_1, \dots, D'_m\}$ as a multiset.
3  $F_0, \dots, F_{n_F-1} := \emptyset$ as multisets. // The folds for the
      cross-validation.
   // Distribute the samples among the folds:
4  $i := 0$.
5  while $|\mathcal{D}'| > 0$ do
6      Randomly choose $D'$ from $\mathcal{D}'$ with a uniform distribution.
7      $F_i := F_i \cup D'$.
8      $\mathcal{D}' := \mathcal{D}' \setminus \{D'\}$.
9      $i := (i+1) \bmod n_F$.
10 end
   // Ensure that $F_0, \dots, F_{n_F-1}$ are not empty:
11 $n_F := \max(\{i \in \{1, \dots, n_F\} \mid F_{i-1} \neq \emptyset\})$.
   // Perform the cross-validation:
12 $n_{TP} := 0$. // True positive count.
13 $n_{FP} := 0$. // False positive count.
14 $n_{TN} := 0$. // True negative count.
15 $n_{FN} := 0$. // False negative count.
16 for $i := 0, \dots, n_F - 1$ do
17     for $(\vec{X}, Y) \in F_i$ do
18         $T := D \setminus F_i$
           // Predict the class of $\vec{X}$ using the parameters $\theta$
              and the training set $T$:
19         $Y' :=$ ClassificationPipeline$(\theta, T, \vec{X})$. // See
              Algorithm 8.2 on Page 149.
20         if $Y' = Y_+$ then
21             if $Y = Y_+$ then $n_{TP} := n_{TP} + 1$.
22             else $n_{FP} := n_{FP} + 1$.
23         else
               // $Y' = Y_-$.
24             if $Y = Y_-$ then $n_{TN} := n_{TN} + 1$.
25             else $n_{FN} := n_{FN} + 1$.
26         end
27     end
28 end
   // Calculate the evaluation metric from the counts and
      return it:
29 return $\rho := \omega(n_{TP}, n_{FP}, n_{TN}, n_{FN})$.
```

TRUE NEGATIVE A sample that is assigned to the negative class by the learned classifier and truly belongs to the negative class.

FALSE NEGATIVE A sample that is assigned to the negative class by the learned classifier but in fact belongs to the positive class.

These are then passed to the evaluation metric, whose value is returned.

This brings us to the final unanswered question of how to evaluate the quality of a given classifier-parameter combination. To answer it, we need to find a suitable evaluation metric for our problem. In Section 8.2.3, we have already established that the accuracy measure is inappropriate in our case. Sun et al. [2009] suggest F-measure and G-mean as appropriate measures in the case of class imbalance. F-measure $\omega_F : \mathbb{N}^4 \to [0,1]$ is defined as follows:

$$\omega_F(n_{TP}, n_{FP}, n_{TN}, n_{FN}) := \frac{2n_{TP}}{2n_{TP} + n_{FN} + n_{FP}}, \quad \forall\, n_{TP}, n_{FP}, n_{TN}, n_{FN} \in \mathbb{N}. \tag{8.7}$$

This corresponds to the harmonic mean of the precision, which is the percentage of positively classified samples that is truly positive, and the recall (or true positive rate), which is the percentage of correctly classified positive samples. G-mean $\omega_G : \mathbb{N}^4 \to [0,1]$ is defined as:

$$\omega_G(n_{TP}, n_{FP}, n_{TN}, n_{FN}) := \sqrt{\frac{n_{TP}}{n_{TP} + n_{FN}} \cdot \frac{n_{TN}}{n_{TN} + n_{FP}}}, \quad \forall\, n_{TP}, n_{FP}, n_{TN}, n_{FN} \in \mathbb{N}. \tag{8.8}$$

This corresponds to the geometric mean of the true positive rate $\omega_{TP} : \mathbb{N}^4 \to [0,1]$ and the true negative rate $\omega_{TN} : \mathbb{N}^4 \to [0,1]$, which is the percentage of correctly classified negative samples:

$$\omega_{TP}(n_{TP}, n_{FP}, n_{TN}, n_{FN}) := \frac{n_{TP}}{n_{TP} + n_{FN}}, \quad \forall\, n_{TP}, n_{FP}, n_{TN}, n_{FN} \in \mathbb{N}. \tag{8.9}$$

$$\omega_{TN}(n_{TP}, n_{FP}, n_{TN}, n_{FN}) := \frac{n_{TN}}{n_{TN} + n_{FP}}, \quad \forall\, n_{TP}, n_{FP}, n_{TN}, n_{FN} \in \mathbb{N}. \tag{8.10}$$

One major difference between F-measure and G-mean is their behavior under changing class distributions. Let us assume that a trained classifier correctly classifies 90 % of the positive samples and, equally, 90 % of the negative samples. If a test set has a class ratio of 1:1, this trained classifier will achieve an average precision, recall (or true positive rate) and true negative rate of 0.9 each. Equally, both F-measure and G-mean will average at 0.9. Now, if we use a test set that consists of 90 % negative samples and 10 % positive samples, the true positive rate and the true negative rate will remain at 0.9. However, the precision will drop to 0.5. This means that the G-mean will still be 0.9 while the F-measure will change to about 0.64.

The problem we have is that our system usually does not know the class distribution to expect in workplace samples when training a classifier for a given particle type. The still limited experience of the BAuA suggests that more background particles than engineered nanoparticles can be expected. However, the true ratio depends on many factors such as the presence of enclosures around the production equipment or ventilation systems. According to Forman and Cohen [2004], an unknown class distribution is a common problem in industrial classification applications. It means that the class distributions in the training set and the workplace samples are disconnected from each other. Moreover, the class distribution of the training set is completely arbitrary because positive and negative training samples are usually created independently of each other as explained in Section 4.3. Using F-measure in the automatic model selection method would therefore mean that these arbitrary changes would change the optimization objective of the classification. This means that having more negative samples in the training set leads to a learned classifier which is more likely to classify samples as negative. This is fine in cases where the class distribution of the training set and target distribution are the same. However, in our case, this is inappropriate for two reasons:

- We do not know the target distribution. Therefore, we cannot adjust the class distribution of the training set accordingly. By using F-measure, we would therefore influence the optimization objective by arbitrarily choosing the class distribution of the training set. This means that the optimization objective in itself would be arbitrary.

- As explained in Section 8.3.1.1, our training sets usually contain more negative samples than positive ones. Using F-measure as the optimization objective would lead to a classifier that is biased towards classifying samples as negative. However, in the case of estimating the concentration of engineered nanoparticles at workplaces, it is better to overestimate this figure than to underestimate it. Transferred to our classification problem, this means that we do not want a trained classifier which is biased towards classifying samples as negative.

We cannot totally remove any bias introduced by the training set class distribution. However, we can reduce the bias by using G-mean, which gives the classification quality of negative and positive samples equal weight, independent of the class distribution of the training set. Therefore, our system uses G-mean to evaluate the parameter configurations in its automatic model selection method.

At this point, we have described all of the main components of our system. The next section will evaluate the parts introduced in this and the previous chapter as well as the system as a whole.

8.4 EVALUATION

In this section, we will evaluate the parts of our system described in this chapter. In addition, we will examine the features introduced in the previous chapter, which we have not yet evaluated due to the lack of prerequisites. Furthermore, we will evaluate the system as a whole. Specifically, this section will take a look at the following aspects:

- The classification performance of our system compared to that of human experts and non-experts
- The contributions of single features to the classification quality
- The time spent by a user to analyze SEM images using the system versus manually performing the task

For the experiments described in this section, we use the classifier implementations of the data mining software Weka [Hall et al., 2009].

For the evaluation, we use the data described in Section 4.4. As explained before, this dataset has been generated in a way to ensure that engineered nanoparticles of a certain type appear on separate images from other types of engineered nanoparticles as well as background particles. Notwithstanding, we have found a small number of background particles in the images of engineered nanoparticles. In addition, we have found depositions of gas residuals as described in Section 4.2.2.5.

As explained in Section 6.1, together with experts from the BAuA, we have made the assumption that agglomerates containing engineered nanoparticles as well as background particles would not occur. However, our dataset contains a few agglomerates that appear to be composed of Ag as well as another particle type.

In light of these circumstances and in order not to give our system an unfair advantage in the evaluation, we have used the following rules for the labels used to evaluate the classification performance:

- Label all CPGs in images of background particle samples (the images listed below the horizontal line in Table 4.1) as background particles.
- Label a CPG in an image of engineered nanoparticle samples (the images listed above the horizontal line in Table 4.1) as the corresponding nanoparticle type except you are sure that it does not contain engineered nanoparticles. In the latter case, do not label the CPG.

Giving a CPG no label corresponds to excluding the CPG from the classifier training in this case. Note that these rules may lead to imaging artifacts and gas residuals being labeled as background particles. However, this has no negative consequences as a classifier that regards such artifacts as background particles does not affect the estimated concentration of engineered nanoparticles.

8.4.1 *Classification*

In this section, we will evaluate the classification performance of our system as a whole on the CPGs found by our segmentation approach. First, however, we want to present a benchmark against which our system can be evaluated, the classification performance of human experts and non-experts.

We have asked some BAuA employees to classify random subsets of the CPGs found by our segmentation approach. Some of these employees are experts in classifying particles in SEM images and others are not. In essence, they have been given the same task our system has to perform after its segmentation part has found the CPGs and has been told which engineered nanoparticle type may occur. In each session, only one engineered nanoparticle type could occur besides different kinds of background particles and the participants have been told this type beforehand. Each classification session had the following steps:

1. Choose an engineered nanoparticle type and tell the participant which type has been chosen.
2. From all particles in the dataset used for this thesis (see Section 4.4) pick all background CPGs and all CPGs containing engineered nanoparticles of the chosen type and use them to create an experiment dataset.
3. Until the participant quits the experiment, repeat:
    a) Choose a random CPG from the experiment dataset and remember if it is positive (contains engineered nanoparticles) or negative (is a background particle).
    b) Extract the CPG plus a margin of 10 pixels from its image and place it onto a uniform image whose pixels have the mean background intensity of the CPG's original image.
    c) Show the extracted CPG and a corresponding scale bar to the participant, ask if he or she thinks it is positive (contains the chosen nanoparticle type) or negative (is a background particle) and remember the decision.

In order for the experiment to be fair, the randomly selected CPG has to be extracted from the image. Otherwise, the participant may have an unfair advantage by gaining contextual information by other CPGs in the image. This is due to the fact that, for the used dataset, background particles have been collected together with other background particles and engineered nanoparticles together with other nanoparticles. However, because the segmentation algorithm may have made an error segmenting the CPG, a margin of 10 pixels is added in order not to cut off part of it. The results of this experiment will be given later together with the classification results of our system (see Table 8.1 on Page 168).

In order to evaluate the classification pipeline of our system, we use 10-fold cross-validation because, as explained in Section 8.3.3, it has proven to yield good results compared to other techniques. The steps of the evaluation are as follows:

1. Choose an engineered nanoparticle type.
2. From all particles in the dataset used for this thesis (see Section 4.4) pick all background CPGs and all CPGs containing engineered nanoparticles of the chosen type and use them to create an experiment dataset.
3. Split these CPGs into 10 folds as described in Lines 1 to 11 of Algorithm 8.9 on Page 160.
4. For each of the folds F, do:
    a) Let T be the training set composed of all folds except F.
    b) Use the automatic model selection method to find good parameters on the training set: $\theta := \mathtt{ParameterSelection}(T, \omega_G, t)$ with $t := 10\,h$ (see Algorithm 8.3 on Page 152).
    c) Train the classification pipeline using the found parameters θ on the training set, classify every sample $\vec{X} \in F$ of the current fold and remember the classification $l_{\text{full},\theta}(T)(\vec{X}) = \mathtt{ClassificationPipeline}(\theta, T, \vec{X})$ (see Algorithm 8.2 on Page 149).
5. Calculate appropriate evaluation metrics from the number of true positives, false positives, true negatives and false negatives.

Here are a few things to note:

- This evaluation uses exactly the same CPGs as the evaluation of the manual classification performance described above.
- As explained in Section 8.3.3, the cross validation needs to group all CPGs of a single position on the substrate into the same fold. Therefore, we use the same cross validation method here as explained in Algorithm 8.9 on Page 160.

- When classifying a sample from a fold, the model selection method as well as the classification pipeline only have access to information of the samples from the nine other folds.
- Passing ω_G to *ParameterSelection*() means that the model selection method chooses the classifier-parameter combination that optimizes the G-mean evaluation metric. In addition, passing $t := 10\,\text{h}$ means that, in each fold, the parameter selection runs for a maximum of ten hours. Thus, the whole evaluation runs for a little over 100 hours.

This evaluation approach can be viewed from two different perspectives:

- Cross-validation of a black-box classifier
- Nested cross-validation in order to guarantee a fair model selection

From the first perspective, we can view the evaluation as standard cross-validation if we treat the combination of model selection algorithm and classification pipeline as a black-block classifier. In fact, the above algorithm is equivalent to Algorithm 8.9 on Page 160 if we replace the call *ClassificationPipeline*$(\theta, T, \vec{X})$ in Line 19 by *ClassificationPipeline*(*ParameterSelection*$(T, \omega_G, t), T, \vec{X}$).

The second perspective requires a little more explanation: There are several methods by which the classification performance of a given classifier-parameter combination can be evaluated such as holdout or cross-validation (see Section 8.2.3). However, when comparing multiple classifier-parameter combinations using such an method, the evaluation result of the best found combination cannot be generalized to unknown data. The reason is that the found classifier-parameter combination is fine-tuned towards the samples used to compare it to the other combination. Therefore, it is important that the model selection does not use the samples used to compute the reported classification performance [Japkowicz and Shah, 2011, pp. 177f.].

Therefore, in the case of cross-validation, Japkowicz and Shah [2011, pp. 177f.] suggest using nested cross-validation in order to report an unbiased classification performance. In the case of n_F-fold cross-validation, this means that for each of the n_F folds, multiple independent cross-validation are performed using the samples of the $n_F - 1$ other folds, which are not used for the evaluation. Each of these inner cross-validation uses a different classifier-parameter combination in order to find the best one. This chosen classifier-parameter combination is then trained on the $n_F - 1$ folds of the outer cross-validation (which have been used to perform the inner cross-validations) and applied to the samples in the remaining fold. This exactly matches the behavior of our evaluation approach listed above because the algorithm *ParameterSelection*() is called inside the outer cross-validation and performs multiple inner cross-validations to find the best classifier-parameter combination.

Note that this approach means that not only one classifier-parameter combination is chosen and evaluated but potentially ten different ones. However, another interpretation more in line with the first perspective is that not the classifier-parameter combinations are evaluated but instead the combination of model selection algorithm and classification pipeline.

In addition to the results of manual classification, we wanted to compare our system to a baseline classifier trained and tested on the same data and using the same features as our classification pipeline. We have chosen the Nearest Neighbor classifier [Aha et al., 1991] for this task because it is one of the most simple classifiers one could think of. When classifying, it searches for the nearest, or in other words most similar, sample in the training set and classifies the new one as belonging to the same class. Thus, it makes no assumption about the data other than that similar samples have similar feature values, which is the basic assumption of machine learning. In addition, it does not apply some sort of advanced intelligence to the classification. In our eyes, this makes it a perfect baseline classifier to compare our classification pipeline with.

We have tested Nearest Neighbor using the same kind of 10-fold cross-validation as we have used to evaluate our classification pipeline. The only kind of preprocessing we have performed is feature scaling as described in Algorithm 8.2 on Page 149, which normalizes the mean and standard deviation of each feature. This shall prevent that changes of one feature overshadow those of another feature as explained in Section 8.3.1.3. Notably, we have also not performed any dimensionality reduction such as feature selection for the Nearest Neighbor test.

A summary of the results of the classification evaluation is listed in Table 8.1. It compares the classification performances of human experts and non-experts, the Nearest Neighbor baseline classifier and the full classification pipeline of our system. Besides the raw true positive, false negative, true negative and true positive counts and the true positive and false negative rates, the values of G-mean and F-measure are also listed. These two evaluation metrics are recommended for cases of class imbalance by Sun et al. [2009] (see Section 8.3.3).

Note that the number of Ag CPGs classified by human experts as well as non-experts is relatively low. This makes the given values of true positive rate G-mean and F-measure for non-experts and experts in the case of Ag unreliable. The low number of classified Ag CPGs has two reasons: Firstly, in our Ag dataset, the ratio of engineered nanoparticle CPGs to background CPGs is about 1:85 compared to approximately 1:11 and 1:5 for TiO_2 and ZnO, respectively.

Table 8.1: The results of the classification performance evaluation. Listed are the results of human experts and non-experts, the Nearest Neighbor baseline classifier and the classification pipeline of our system. All experiments have been performed on the same dataset, except that for the experiments performed by humans, random subsets of the dataset have been used. For each of these categories, the following measures are given: true positive count n_{TP}, false negative count n_{FN}, true negative count n_{TN}, false positive count n_{FP}, true positive rate ω_{TP}, true negative rate ω_{TN}, G-mean ω_G and F-measure ω_F. The best results of each evaluation metric for each particle type are underlined.

		n_{TP}	n_{FN}	n_{TN}	n_{FP}	ω_{TP}	ω_{TN}	ω_G	ω_F
Ag	Non-experts	3	4	683	241	0.429	0.739	0.563	0.024
	Experts	13	1	1070	571	$\underline{0.929}$	0.652	0.778	0.043
	Nearest Neighbor	91	18	9266	13	0.835	$\underline{0.999}$	0.913	$\underline{0.854}$
	Our system	96	13	9129	150	0.881	0.984	$\underline{0.931}$	0.541
TiO_2	Non-experts	53	32	889	202	0.624	0.815	0.713	0.312
	Experts	127	30	1686	633	0.809	0.727	0.767	0.277
	Nearest Neighbor	459	386	8681	598	0.543	$\underline{0.936}$	0.713	0.483
	Our system	685	160	8354	925	$\underline{0.811}$	0.900	$\underline{0.854}$	$\underline{0.558}$
ZnO	Non-experts	79	88	825	170	0.473	0.829	0.626	0.380
	Experts	247	161	1857	406	0.605	0.821	0.705	0.466
	Nearest Neighbor	920	854	8677	602	0.519	$\underline{0.935}$	0.696	0.558
	Our system	1522	252	8015	1264	$\underline{0.858}$	0.864	$\underline{0.861}$	$\underline{0.668}$

The second reason is that we believed for a long time that human experts were able to almost perfectly discriminate between the Ag particles and background particles. However, newer images of Ag nanoparticles have shown agglomerates which look different from those in previous images. In addition, a careful re-evaluation of the existing images has shown that some small CPGs, which were previously assumed to be background particles, could in fact very well be agglomerates of Ag agglomerates. And because of the labeling rules for our test data as described at the beginning of Section 8.4, we have labeled them as Ag nanoparticles. These small particles have a much lower intensity than the larger Ag agglomerates. Due to these reasons, the Ag particles were not part of the early round of human classification testing. Therefore, the total number of particles classified by human experts and non-experts for the Ag experiment is only a little more than two thirds of those of the other two experiments.

Our system has yielded better results than human experts in every case except for the true positive rate in the Ag case. However, since only 14 particles played a part in the determination of the manual true positive rate of 0.929 due to the reasons mentioned above, it falls within the margin of error of our system's true positive rate of 0.881. In addition, the manual true negative rate of 0.652 is considerably lower than the 0.984 of our system. This leads to much better values of G-mean (0.931 compared to 0.778) as well as F-measure (0.541 versus 0.043) for our system. On the other hand, even if the manual true positive rate, which we failed to measure exactly, would be equal to 1, the resulting values of G-mean (0.807) and F-measure (0.047) would still be worse than those of our system. This means that we have achieved our main goal expressed in Chapter 3 of matching and also exceeding the classification performance of human experts. Consequently, our system also produces better results than non-experts.

Compared to the Nearest Neighbor classifier, our classification pipeline has in every case yielded a higher G-mean and in two out of three cases a better F-measure. There are some things to note about this result: The classifiers and their parameters in our pipeline have been chosen from the point of view that it is better to over-estimate the concentration of engineered nanoparticles than it is to under-estimate it. The reason for this is that if the concentration was under-estimated, the worker would be exposed to a higher potential health risk than the estimation would have suggested. This tendency favoring over-estimation leads to a trade-off towards higher true positive rates and lower true negative rates. This effect is reflected in the results of our classification pipeline compared to Nearest Neighbor. Note that the same trend can be seen in the results of the human experts compared to non-experts where the former always have a higher true positive rate and a lower true negative rate than the latter.

In this sense, Nearest Neighbor behaves like a non-expert compared to our system.

In the case of Ag, Nearest Neighbor shows a higher F-measure than the system pipeline. There are some potential reasons for this behavior:

- The parameter selection method explained in Section 8.3.2 chooses the classifier-parameter combination by their G-mean score instead of F-measure.
- The model selection method may perform better with many training particles compared to Nearest Neighbor because of the high number of parameters to be selected. For Ag, there are relatively few training particles available compared to the other engineered nanoparticle types.
- In the case of Ag, the ratio between engineered CPGs and background particles is so low that one can relatively easily achieve a high F-measure value by concentrating on achieving a high true negative rate. However, because our classification pipeline deliberately favors concentration over-estimation as explained above, this behavior would be counterproductive.

In addition, as mentioned in Section 8.3.3, we believe that F-measure is not a good fit for our system. The reason is that it is heavily dependent on the class distribution. However, we do not have reliable data about the class distributions in real-world scenarios. Therefore, basing the performance estimation of a trained classifier on the arbitrary class distribution of our test dataset would, in our opinion, be incorrect.

To demonstrate this, we can calculate the F-measure values that the classifiers would achieve if the class distribution of the test set was different. For the case of Ag, our pipeline achieves an F-measure of about 0.541 on our test set. Nearest Neighbor yields 0.854 on the same data. If the class ratio was 1:1, our pipeline would achieve an F-measure of about 0.929 compared to 0.909 for Nearest Neighbor. We have done this calculation by computing F-measure in terms of precision and recall (which is the same true positive rate), where we leave the recall unchanged and calculate the precision as $\frac{\omega_{TP}}{\omega_{TP}+(1-\omega_{TN})}$ for the class ratio of 1:1.

Another fact that shows that F-measure is a suboptimal performance measure in our case is that the human experts have achieved a lower F-measure value than the non-experts in the case of TiO_2. We have nonetheless provided the F-measure values of our experiments for the sake of completeness.

Note that the relatively good results of Nearest Neighbor also show that the preprocessing and features of our system have been chosen

Table 8.2: The classifiers and parameters chosen by the automatic model selection method on all available CPGs. N/A means that a parameter is not applicable to the chosen classifier.

PARAMETER	Ag	TiO_2	ZnO
Classifier	Logistic regression	SVM	SVM
θ_λ	0.003	N/A	N/A
SVM kernel	N/A	K_{RBF}	K_{RBF}
θ_γ	N/A	0.1	0.001
θ_C	N/A	10	300
θ_w	100	1	3
θ_N	1000 %	100 %	0 %
θ_{MS}	1	30	3
Selected feature count	9	22	48

well. In addition, we want to note that the maximum population similarity parameter ι_{max} (see Algorithm 8.3), which shall prevent overfitting, has only been reached and has led to an early termination of the parameter selection for Ag. We presume this is the case because of the relatively low positive sample count.

Finally, Table 8.2 lists the classifier-parameter combinations that have been selected by the automatic model selection method on the full datasets.

8.4.2 *Features*

In addition to the system and the classification pipeline as a whole, we want to evaluate the quality and impact of each feature described in Section 7.3. To do that, we use two separate approaches:

- Classify each particle type using only a single feature.
- Count how often each feature has been chosen by the feature selection method in our classification experiments.

We will explain these strategies in the following two sections.

8.4.2.1 *Single Feature Classification*

For this experiment, we have classified each of the three datasets containing all background CPGs and one engineered nanoparticle type using only one feature per run. This has been done using basically the same method as for the Nearest Neighbor experiment explained on Page 167. Again, we have used 10-fold cross-validation and the Nearest Neighbor classifier. The only difference is that the classifier has had only access to the value of a single feature plus pixel density of the image per cross-validation. This means that we have performed three times the number of features separate cross-validations because we have three types of engineered nanoparticles in our dataset. Again, we have chosen Nearest Neighbor because it is a very simple classifier, which makes no assumption about the data.

The results of this experiment are given in Table 8.3. We will analyze the results in the following section.

Table 8.3: The classification performances measured in G-mean when using only a single feature. For each feature, the G-means for the three engineered nanoparticle types and the average value are given. The features are sorted by their average classification performance. As the classifier, Nearest Neighbor has been used.

FEATURE	Ag	TiO_2	ZnO	AVG.
Mean electron count	0.913	0.407	0.644	0.654
Maximum electron count	0.882	0.441	0.627	0.650
Haralick cluster shade (-1,1)	0.770	0.438	0.451	0.553
Haralick cluster shade (0,1)	0.770	0.466	0.414	0.550
Haralick cluster shade (1,1)	0.782	0.462	0.400	0.548
Haralick cluster shade (1,0)	0.770	0.430	0.426	0.542
Haralick energy (1,1)	0.640	0.479	0.461	0.527
Haralick energy (0,1)	0.618	0.479	0.457	0.518
Haralick entropy (1,0)	0.641	0.480	0.429	0.516
Haralick entropy (1,1)	0.619	0.465	0.454	0.512
Haralick entropy (0,1)	0.654	0.446	0.429	0.510
Haralick cluster prominence (1,0)	0.547	0.495	0.483	0.508
Haralick cluster prominence (0,1)	0.596	0.484	0.440	0.507
Perimeter	0.611	0.440	0.463	0.505
Haralick entropy (-1,1)	0.604	0.454	0.449	0.502
Haralick cluster prominence (1,1)	0.573	0.501	0.429	0.501
Haralick cluster prominence (-1,1)	0.531	0.534	0.423	0.496
Minimum electron count	0.452	0.430	0.607	0.496
Projected area	0.682	0.434	0.370	0.495
Haralick energy (-1,1)	0.564	0.454	0.453	0.490
Haralick energy (1,0)	0.595	0.450	0.405	0.483
Haralick maximum probability (1,0)	0.586	0.445	0.401	0.477
Mean wavelet response (2048nm)	0.618	0.385	0.415	0.473
Haralick maximum probability (0,1)	0.573	0.441	0.399	0.471

Continued on next page...

Table 8.3: (continued)

FEATURE	Ag	TiO_2	ZnO	AVG.
Haralick maximum probability (-1,1)	0.505	0.475	0.431	0.470
Haralick maximum probability (1,1)	0.548	0.445	0.416	0.470
Perimeter+	0.588	0.385	0.424	0.465
Maximum intensity	0.449	0.354	0.391	0.398
Mean wavelet response (1024nm)	0.415	0.350	0.412	0.392
Mean wavelet response (512nm)	0.316	0.393	0.389	0.366
Normalized histogram (3)	0.357	0.252	0.383	0.331
Normalized histogram (2)	0.357	0.245	0.383	0.328
Normalized histogram (0)	0.233	0.342	0.404	0.327
Mean wavelet response (256nm)	0.270	0.307	0.386	0.321
Normalized histogram (9)	0.213	0.304	0.374	0.297
Normalized histogram (5)	0.252	0.248	0.382	0.294
Normalized histogram (6)	0.286	0.245	0.340	0.290
Haralick local homogeneity (1,1)	0.252	0.262	0.354	0.289
Normalized histogram (1)	0.233	0.264	0.365	0.287
Mean wavelet response (128nm)	0.190	0.262	0.387	0.280
Mean wavelet response (64nm)	0.165	0.284	0.384	0.278
Normalized histogram (7)	0.191	0.268	0.365	0.275
Isoperimetric quotient	0.190	0.250	0.348	0.263
Normalized histogram (8)	0.135	0.264	0.360	0.253
Haralick contrast (1,0)	0.095	0.286	0.377	0.253
Mean wavelet response (8nm)	0.134	0.252	0.366	0.251
Normalized histogram (4)	0.165	0.228	0.347	0.247
Mean wavelet response (16nm)	0.135	0.232	0.367	0.245
Haralick contrast (0,1)	0.000	0.328	0.394	0.241
Haralick local homogeneity (0,1)	0.095	0.257	0.346	0.233
Haralick contrast (-1,1)	0.000	0.290	0.390	0.227
Haralick contrast (1,1)	0.000	0.287	0.391	0.226
Mean wavelet response (4nm)	0.000	0.299	0.361	0.220
Mean wavelet response (32nm)	0.000	0.269	0.370	0.213
Haralick local homogeneity (1,0)	0.000	0.273	0.358	0.211
In-image contour percentage	0.270	0.146	0.100	0.172
Haralick local homogeneity (-1,1)	0.000	0.087	0.357	0.148

8.4.2.2 *Feature Selection Count*

In addition to the single feature classification experiments, the results of the evaluation of our classification pipeline (see Section 8.4.1) can give us information about the quality of our features. As mentioned before, our experiments using 10-fold cross-validation have yielded 10 classifier-parameter combinations per nanoparticle type, adding up to a total of 30. Each of these classifier-parameter combinations contains a list of selected features to be used for the classification. By counting the number of classifier-parameter combinations that contain a given feature, we can get an indication of its quality. The results are listed in Table 8.4.

Table 8.4: The number of feature subsets chosen by the automatic parameter selection method including each feature out of 10 for each nanoparticle type and a total of 30.

FEATURE	Ag	TiO_2	ZnO	TOTAL
Maximum electron count	9	10	8	27
Haralick energy (1,0)	7	10	9	26
Haralick contrast (-1,1)	7	7	10	24
Mean electron count	8	7	9	24
Mean wavelet response (1024nm)	7	9	8	24
Normalized histogram (5)	9	7	8	24
Haralick local homogeneity (0,1)	5	10	8	23
Haralick maximum probability (-1,1)	6	8	9	23
Haralick maximum probability (1,0)	8	7	8	23
Haralick maximum probability (1,1)	5	9	9	23
Haralick cluster prominence (-1,1)	5	7	10	22
Haralick contrast (1,1)	3	9	10	22
Normalized histogram (2)	5	8	9	22
Haralick cluster shade (0,1)	6	6	9	21
Maximum intensity	6	5	10	21
Mean wavelet response (2048nm)	5	7	9	21
Mean wavelet response (256nm)	3	9	9	21
Mean wavelet response (32nm)	6	7	8	21
Haralick local homogeneity (1,0)	6	5	9	20
Mean wavelet response (512nm)	7	7	6	20
Mean wavelet response (8nm)	7	5	8	20
Minimum electron count	4	9	7	20

Continued on next page...

Table 8.4: (continued)

FEATURE	Ag	TiO_2	ZnO	TOTAL
Haralick cluster prominence (0,1)	5	5	9	19
Haralick cluster shade (-1,1)	6	7	6	19
Haralick contrast (0,1)	5	4	10	19
Haralick contrast (1,0)	4	7	8	19
Haralick local homogeneity (-1,1)	6	6	7	19
Mean wavelet response (128nm)	6	6	7	19
Normalized histogram (0)	5	6	8	19
Normalized histogram (1)	2	8	9	19
Normalized histogram (7)	5	6	8	19
Perimeter	5	8	6	19
Perimeter+	5	6	8	19
Projected area	7	8	4	19
Haralick energy (-1,1)	5	5	8	18
Haralick energy (0,1)	5	7	6	18
Haralick entropy (0,1)	5	7	6	18
Haralick entropy (1,0)	5	6	7	18
Haralick entropy (1,1)	5	7	6	18
Haralick local homogeneity (1,1)	4	7	7	18
In-image contour percentage	3	8	7	18
Isoperimetric quotient	4	7	7	18
Mean wavelet response (4nm)	4	6	8	18
Normalized histogram (4)	7	4	7	18
Haralick cluster shade (1,0)	4	5	8	17
Haralick cluster shade (1,1)	4	5	8	17
Haralick energy (1,1)	5	5	7	17
Haralick maximum probability (0,1)	3	6	8	17
Normalized histogram (6)	4	6	7	17
Normalized histogram (8)	6	6	5	17
Normalized histogram (9)	4	7	6	17
Haralick cluster prominence (1,0)	5	6	5	16
Haralick cluster prominence (1,1)	6	4	6	16
Haralick entropy (-1,1)	1	7	8	16
Mean wavelet response (16nm)	5	5	6	16
Mean wavelet response (64nm)	1	5	7	13

Continued on next page...

Table 8.4: (continued)

FEATURE	Ag	TiO_2	ZnO	TOTAL
Normalized histogram (3)	1	5	7	13

We will comment these two sets of results feature by feature:

PROJECTED AREA For both types of evaluation, the projected area achieves an average result. It is used in almost two thirds of the feature subsets. Therefore, removing it from the system would probably decrease its classification performance.

PERIMETER While performing similar to the projected area in the second result, its classification results are among the top 15. This makes it a valuable feature.

PERIMETER INCLUDING INNER CONTOURS (PERIMETER+) Its classification result is worse than the normal perimeter despite having the same selection count. Therefore, it seems that the information about holes in the CPG is not very valuable. Possibly, the recognition quality of these holes has to be increased for this feature to be helpful.

ISOPERIMETRIC QUOTIENT By itself, this feature does not provide much information to discriminate CPGs. However, like almost all features, it has been included in more than half of the feature subsets.

IN-IMAGE CONTOUR PERCENTAGE As expected, this feature is not able to distinguish particles by itself because it is not a property of the CPG itself but rather of the imaging circumstances. However, like all features listed up until now, which are the basic geometric features, it shall provide context information to the classifier. For example, other feature values may change if the CPG is not fully visible. This feature shall give the classifier a chance to notice this change.

MEAN CONTOUR ANGLE WAVELET RESPONSE The performance of these features depend on their wavelengths and the size of the engineered nanoparticles. For larger ones like the Ag particles, long wavelengths such as 2048 nm perform better. Smaller wavelengths seem to work better for small particles. In general, higher values for the wavelength yield better results. This is an interesting result because these wavelengths are several times the size of the nanoparticle diameters. It suggests that the feature values do not capture the size of the particles themselves but the size of the structures they form.

NORMALIZED RELATIVE HISTOGRAM Here, 10 feature values are calculated, one for each bin of the histogram. By themselves, these provide an average classification performance. However, by the number of feature subsets, bin 5 is on the shared third place while bin 3 is on the last. However, note that bin 3 is included in 7 of 10 feature subsets for ZnO and in only 1 for Ag. This suggests that the typical intensity distribution highly depends on the type of particles.

HARALICK FEATURES The performance of the different Haralick features varies. Cluster Shade, energy, entropy and cluster prominence yield good results while local homogeneity and contrast perform poorly. It seems that the offset direction is not very important because of the random orientation of the CPGs on the substrate. This can explain the relatively low subset count in Table 8.4 of features such as the cluster shade. These four feature values contain much redundant information. Therefore, the classifier does not need all of them.

ELECTRON COUNTS The estimated electron counts developed by us perform very well. The mean and maximum achieve much better G-means for Ag and ZnO than all other features. In fact, the estimated mean electron count alone achieves a classification performance (0.913) for Ag that outperforms human experts (0.778) and is comparable to our system (0.931). In addition, the maximum electron count is included in 90 % of all feature subsets. Compared to the maximum intensity of a CPG, which we have included as a comparison, the maximum electron count performs much better. This proves our assumption from Sections 4.2.2.4 and 7.3 that one cannot rely on absolute intensities and that our electron count estimation method is effective.

8.4.3 *Time Consumption*

One goal of our system has been to reduce the time a human spends analyzing SEM images of airborne particles of a workspace. Therefore, in this section, we will compare the time spent in an analysis with the system and without it, respectively. Because the workflow in these two cases is quite different, Table 8.5 gives an overview over the required steps and if the computer or the human performs them.

Note that the topic of this thesis are only the Detection and Classification substeps of both steps. However, we want to give an overview over the time consumption of the whole process. In addition, we have left out smaller steps such as the sample and image management. These are difficult to capture systematically and usually take less time than the listed steps. Next, we will explain these steps in more detail. Because we have laid out the basics of the process in Chapters 2 and 4,

Table 8.5: The steps of sample analysis with and without our system, respectively. For each step and case, it is listed if the process is manual or automatic. N/A is listed if the step does not occur when our system is not used. If a step can potentially be done automatically, pot. automatic is listed. The asterisks are explained in the text.

STEP	SUBSTEP	WITHOUT SYSTEM	WITH SYSTEM
Training	Sampling	Manual*	Manual
	Imaging	Pot. automatic*	Pot. automatic
	Detection	Manual*	Automatic
	Classification	N/A	Manual*
Analysis	Sampling	Manual	Manual
	Imaging	Pot. automatic	Pot. automatic
	Detection	Manual	Automatic
	Classification	Manual	Automatic

we will not repeat them here. We will not consider times when the system is working because the user is free to walk away from the computer and spent the time on other things.

The first step is training the recognition ability with known particles. Usually, this phase is only associated with automatic systems such as classifiers. This is why we have added asterisks to these steps for the process without the system. However, humans performing manual classification also have to learn the appearance of the particles to be recognized. In addition, they also need background particle samples to see if these may look similar to the particles of interest. Furthermore, we have experienced that certain particles may look different on distinct images. Also, we have seen in Section 8.4.1 that the manual classification performance has been worse than the one of our system even though the human classifiers had access to the same training images. Therefore, we will assume that the training phase is also applied in the case without our system but potentially with fewer images and particles.

As a first step of the training, particles of known types have to be gathered using a precipitator. This has to be done once per particle type or sampling location. A person operating the precipitator has to be on site. The time consumption of this step varies based on the used precipitator type and the circumstances such as the particle concentration to be expected. According to the BAuA, typical gathering times are 8 h for the thermal precipitator and 2 h to 4 h for the electrostatic precipitator.

Next, SEM images of the gathered particles have to be taken. In case of the BAuA, this process is done by hand. However, SEMs can be operated by a computer, which can be instructed to take images at random or systematic locations in a certain area of the sample. According to the BAuA, an operator needs about 2 min per image using a modern SEM.

Next, the CPGs in the training images have to be found. When using our system, this step is performed automatically. However, without it, they have to be found manually. The time for this depends on several factors such as the number of CPGs in an image and their contrast compared to the background. In our experiments, we have found that this process takes a human on average about 13 s per CPG. There will typically also be time spent on keeping track of the CPGs that have been counted and avoiding counting them multiple times. We do not consider this time because it is difficult to capture systematically.

For the case without our system, the training phase is complete. With the system, it would theoretically suffice to tell it which images contain background particles and which show engineered nanoparticles. However, as mentioned at the beginning of Section 8.4, the samples containing engineered nanoparticles may include some background particles. Therefore, in this step, an expert has to classify the CPGs in the images that contain engineered nanoparticles. For the images showing background particles, this step is not necessary because it is very unlikely that these contain engineered nanoparticles. In the experiment on manual classification performance, the average time of a human expert classifying a CPG has been approximately 5.5 s.

After the system or a human expert have been trained to recognize a certain type of engineered nanoparticles, workplace samples can be analyzed. Such an analysis comprises the same substeps as the training phase. The first three of these are very similar except that the samples are taken at a workplace. The classification step, however, is different. Here, each CPG has to be classified either as engineered nanoparticles or as background particles. With our system, this is done automatically. Without it, however, an human expert has to do this. As mentioned above, this takes approximately 5.5 s per CPG.

Because the cases with and without our system are quite different, it is difficult to compare the time consumption in these cases objectively. However, in order to establish some kind of comparability, we will estimate the manual time consumption of these two cases in a concrete scenario:

- The system or the expert, respectively, shall learn to recognize a new type of engineered nanoparticles.
- Both have access to enough previously recorded images of background particles, which have been used to learn recognizing other particle types.

Table 8.6: The time consumption of a sample analysis scenario with and without our system, respectively.

STEP	SUBSTEP	WITHOUT SYSTEM	WITH SYSTEM
Training	Sampling	$3\,\mathrm{h}$	$3\,\mathrm{h}$
	Imaging	$22 \cdot 2\,\mathrm{min} = 44\,\mathrm{min}$	$44 \cdot 2\,\mathrm{min} = 88\,\mathrm{min}$
	Detection	$422 \cdot 13\,\mathrm{s} = 91.43\,\mathrm{min}$	$0\,\mathrm{min}$
	Classification	$0\,\mathrm{min}$	$845 \cdot 5.5\,\mathrm{s} = 77.46\,\mathrm{min}$
Analysis	Sampling	$3\,\mathrm{h}$	$3\,\mathrm{h}$
	Imaging	$50 \cdot 2\,\mathrm{min} = 100\,\mathrm{min}$	$50 \cdot 2\,\mathrm{min} = 100\,\mathrm{min}$
	Detection	$700 \cdot 13\,\mathrm{s} = 151.67\,\mathrm{min}$	$0\,\mathrm{min}$
	Classification	$700 \cdot 5.5\,\mathrm{s} = 64.17\,\mathrm{min}$	$0\,\mathrm{min}$
Total		$811.27\,\mathrm{min}$	$625.46\,\mathrm{min}$

- We assume that an electrostatic precipitator gathers particles for $3\,\mathrm{h}$ for the training sample and workplace sample, respectively.
- For the training of the system, we assume that 44 images containing 845 CPGs are taken by a human operator. This coincides with the number of TiO_2 images and CPGs used in this thesis. For the case without the system, we assume that half of the images and CPGs are taken and analyzed for the training of the expert.
- For the workplace sample, 50 images containing 700 CPGs are taken and analyzed.

The time consumption of this scenario is listed in Table 8.6. With our system, the training phase is slightly longer with $5\,\mathrm{h}$ and $45\,\mathrm{min}$ compared to $5\,\mathrm{h}$ and $15\,\mathrm{min}$ without it. However, the analysis phase takes much less manual time with our system. Here, the time consumption is $4\,\mathrm{h}$ and $40\,\mathrm{min}$ compared to $8\,\mathrm{h}$ and $16\,\mathrm{min}$. All in all, our system saves $3\,\mathrm{h}$ and $6\,\mathrm{min}$ with a total time of $10\,\mathrm{h}$ and $25\,\mathrm{min}$ compared to $13\,\mathrm{h}$ and $31\,\mathrm{min}$.

Note that if an automatic SEM is used, the time saving of the training phase without our system will vanish. In addition, in a usual scenario, the system will be trained once for a given nanoparticle type and used to analyze multiple workplace samples. There, the time savings will be even higher.

In this chapter, we have provided details on the last part of the system workflow and presented the performance results of the system as a whole. In the next chapter, we will present an addition to our system, which is able to give the user information how additional training samples will affect the classification performance to be expected.

9 PERFORMANCE PREDICTION

In this chapter, we will propose an algorithm that enables our system to predict its classification performance under the assumption that additional training data is available. In Section 9.1, we will outline the goals for our performance prediction approach. Section 9.2 will present an overview of related literature. Our method will be described in Section 9.3 and in Section 9.4, we will present an evaluation of the proposed algorithm.

9.1 GOALS

As explained in Section 8.1.2, it is important to have as many training samples as possible. However, gathering particles and producing SEM images is time-consuming and potentially expensive. Therefore, a user training the system has an interest in producing neither too few nor too many training samples.

We want our system to be able to predict the influence of the addition of a certain number of samples of a specific class to the training set on the classification performance. In particular, we want the system to be able to answer the following questions:

- How will the classification performance change if n CPGs composed of engineered nanoparticles are added to the training set?
- How will the classification performance change if n CPGs composed of background particles are added to the training set?
- How will the classification performance change if n CPGs with the same class distribution as the current training set are added to the training set?

In particular, we want to be able to predict performance measures such as G-mean instead of only the classification accuracy because they are better suited for situations such as class imbalance (see Section 8.3.3). For that, it is sufficient to predict the false positive and false negative rates because the other measures can be derived from them.

If the system makes predictions like these, the user is able to make a well-informed decision if additional training data is needed and what type these new samples should have. This can have two effects:

- The user saves time and money by avoiding to generate additional training data that does not sufficiently improve the classification performance.

- The classification performance is optimized because the user is able to generate the right amount of training data of the proper class.

9.2 RELATED WORK

We have found two approaches to predict the expected performance of a given classifier after the addition of training samples to the training set. Both are only able to predict the misclassification rate, which is defined as the percentage of samples that is incorrectly classified by the learned classifier. Thus, it is identical to one minus the accuracy. However, in Section 8.2.3, we have already demonstrated that the accuracy is an unsuitable performance metric in our case. Both approaches work in two steps:

1. Estimate the misclassification rate or components of it for training set sizes smaller than the available dataset.
2. Predict the expected misclassification rate given larger training set sizes.

The first method is proposed by Mukherjee et al. [2003]. It uses a repeated holdout approach to estimate the misclassification rates (see Section 8.2.3). For the prediction step, a power-law function (which we will describe in Section 9.3) is fit to these estimates. It is used to predict the misclassification rates for larger training sets. In addition, Mukherjee et al. [2003] fit such functions to the 25th and 75th percentiles of the error estimates, which are then used to give bounds on the predicted error. We find that the estimation procedure of the method has several drawbacks. For a given training set size $n_{\vec{X}}$, it repeatedly and randomly chooses $n_{\vec{X}}$ samples from the dataset, trains the classifier using these and uses the remaining samples to estimate the expected misclassification rate. This can lead to the following problems:

- Some samples will more often end up in the test set than the training set. For others, the opposite will be true. This may introduce a bias in the estimation of the misclassification rate.
- The size of the test sets is equal to the size of the whole dataset minus $n_{\vec{X}}$. This means that the size of the test set will vary from 10 to the size of the whole dataset minus 10 if we use the parameters recommended by the authors. This means that the variance of the error estimation strongly depends on $n_{\vec{X}}$. In other words, estimations for small values of $n_{\vec{X}}$ are much more exact than those for larger values.
- Similarly, the calculation of the 25th and 75th percentiles are biased to deviate stronger for large values of $n_{\vec{X}}$. This means that

the predicted error bounds are further apart than they would be using an unbiased estimation method, especially for large values of $n_{\vec{X}}$.

Smith et al. [2012] propose another method to predict the classification performance. Contrary to Mukherjee et al. [2003], they decompose the misclassification rate into a bias and a variance component. For this, they use the decomposition method by Kohavi and Wolpert [1996]. The assumption behind it is that the responses of these two components to changes of the training set size can better be predicted individually than that of the misclassification rate as a whole. Instead of a holdout approach, Smith et al. [2012] use a cross-validation technique proposed by Webb and Conilione [2003]. Therefore, their method does not have the disadvantages of the estimation phase of the approach by Mukherjee et al. [2003].

However, their prediction method makes some disputable assumptions. In order to predict the misclassification rate for a training set size of $n_{\vec{X}}'$, their method only needs estimates of the bias and variance for a training set size of $n_{\vec{X}}$ where $n_{\vec{X}}$ is a fixed value. The predicted misclassification rate for a training set size of $n_{\vec{X}}'$ is then calculated from these estimates using a simple linear model. However, Smith et al. [2012] argue that the parameter values of this model only depend on the value of $n_{\vec{X}}$. This means that these values do not depend on the following factors:

- The used classifier
- The classification data
- The value of $n_{\vec{X}}'$

We find the last point especially concerning. It means that their method will predict the same misclassification rate no matter whether 10 samples are added to the training set or $1\,000\,000$.

As mentioned before, both approaches are only able to predict the misclassification rate or, equivalently, the accuracy of a classifier-parameter combination. However, we need a method which is able to predict other performance metrics such as the G-mean. In the next section, we will therefore propose a method which is able to do this.

9.3 METHOD

As explained in Section 8.1.1, a trained classifier $l(T) : \mathcal{X} \to \mathcal{Y}$ tries to assign a class $Y \in \mathcal{Y}$ to a given sample $\vec{X} \in \mathcal{X}$. For the derivation of our method, we will assume that the function $L : \mathcal{X} \to \mathcal{Y}$ returns the true class of a given sample. This means that for a perfectly learned classifier, $l(T)(\vec{X}) = L(\vec{X}), \forall \vec{X} \in \mathcal{X}$ would be true. In addition, we will assume that L is deterministic. Otherwise, the choice of classification features would be insufficient in order to distinguish the classes.

Let $\tilde{X} : \Omega \to \mathcal{X}$ be a random variable representing the probability distribution of generating a sample when gathering particles in a realistic scenario. In addition, let $\tilde{T} : \Omega \to \mathcal{X} \times \mathcal{Y}$ be a random multiset representing a randomly generated training set. With these definitions, we can write the misclassification rate given a training set size of $n_{\vec{X}} \in \mathbb{N}_{>0}$ as $\mathbb{P}(l(\tilde{T})(\tilde{X}) \neq L(\tilde{X}) \mid |\tilde{T}| = n_{\vec{X}})$. The methods by Mukherjee et al. [2003] and Smith et al. [2012] both try to predict this quantity. However, we cannot calculate the G-mean from the misclassification rate. To do that, we need to predict the true positive rate $\mathbb{P}(l(\tilde{T})(\tilde{X}) = Y_+ \mid L(\tilde{X}) = Y_+, |\tilde{T}| = n_{\vec{X}})$ and the true negative rate $\mathbb{P}(l(\tilde{T})(\tilde{X}) = Y_- \mid L(\tilde{X}) = Y_-, |\tilde{T}| = n_{\vec{X}})$ given a training set size of $n_{\vec{X}}$. These can be expressed as

$$\begin{aligned}\mathbb{P}(l(\tilde{T})(\tilde{X}) = Y_+ \mid L(\tilde{X}) = Y_+, |\tilde{T}| = n_{\vec{X}}) = \\ 1 - \mathbb{P}(l(\tilde{T})(\tilde{X}) \neq L(\tilde{X}) \mid L(\tilde{X}) = Y_+, |\tilde{T}| = n_{\vec{X}}), \\ \forall n_{\vec{X}} \in \mathbb{N}_{>0}, \quad (9.1)\end{aligned}$$

and

$$\begin{aligned}\mathbb{P}(l(\tilde{T})(\tilde{X}) = Y_- \mid L(\tilde{X}) = Y_-, |\tilde{T}| = n_{\vec{X}}) = \\ 1 - \mathbb{P}(l(\tilde{T})(\tilde{X}) \neq L(\tilde{X}) \mid L(\tilde{X}) = Y_-, |\tilde{T}| = n_{\vec{X}}), \\ \forall n_{\vec{X}} \in \mathbb{N}_{>0}. \quad (9.2)\end{aligned}$$

Therefore, it will be sufficient if our method is able to predict the probabilities $\mathbb{P}(l(\tilde{T})(\tilde{X}) \neq L(\tilde{X}) \mid L(\tilde{X}) = Y, |\tilde{T}| = n_{\vec{X}})$ for $Y \in \{Y_+, Y_-\}$. However, this will only be sufficient to answer how the classification performance changes if samples of both classes are added to the training set. To predict the influence of additional samples of a single class, we need to define the random variables $\tilde{n}_+ : \Omega \to \mathbb{N}$ and $\tilde{n}_- : \Omega \to \mathbb{N}$, which stand for the number of positive and negative samples in the training set $\tilde{T}$, respectively:

$$\tilde{n}_+ := |\tilde{T} \cap (\mathcal{X} \times \{Y_+\})|, \quad (9.3)$$

$$\tilde{n}_- := |\tilde{T} \cap (\mathcal{X} \times \{Y_-\})|. \quad (9.4)$$

To answer the remaining questions, our method shall be able to predict the false negative rates $E_{FN} : \mathbb{N}^2 \to [0, 1]$:

$$\begin{aligned}E_{FN}(n_+, n_-) := \\ \mathbb{P}(l(\tilde{T})(\tilde{X}) \neq L(\tilde{X}) \mid L(\tilde{X}) = Y_+, \tilde{n}_+ = n_+, \tilde{n}_- = n_-), \\ \forall n_+, n_- \in \mathbb{N}_{>0}, \quad (9.5)\end{aligned}$$

and the false positive rates $E_{FP} : \mathbb{N}^2 \to [0, 1]$:

$$\begin{aligned}E_{FP}(n_+, n_-) := \\ \mathbb{P}(l(\tilde{T})(\tilde{X}) \neq L(\tilde{X}) \mid L(\tilde{X}) = Y_-, \tilde{n}_+ = n_+, \tilde{n}_- = n_-), \\ \forall n_+, n_- \in \mathbb{N}_{>0}, \quad (9.6)\end{aligned}$$

for certain values of $n_+, n_- \in \mathbb{N}_{>0}$.

Similar to the two existing methods, our prediction approach consists of the following two phases:

1. *Estimation phase.* We use the same cross-validation approach as Smith et al. [2012]. However, instead of the misclassification rates for a single class distribution, we estimate the false negative rates and false positive rates for several class distributions.

2. *Prediction phase.* To predict future values, we use a power-law function as proposed by Mukherjee et al. [2003]. Similar to the first phase, we predict the values of the false negative rates and false positive rates for varying class distributions instead of the misclassification rate for a single one. We have also tested to decompose the estimates into bias and variance components and predicting their values individually (see our paper on this topic [Kockentiedt et al., 2014]). However, directly predicting the probabilities has shown to produce better results.

The estimation of the false negative and false positive rates given multiple classification results works as follows: The probability $\mathbb{P}\big(l(\tilde{T})(\vec{X}) \neq L(\vec{X}) \,\big|\, \tilde{n}_+ = n_+, \tilde{n}_- = n_-\big)$ for a given $\vec{X} \in \mathcal{X}$ can be estimated by $\hat{\mathbb{P}}\big(l(\tilde{T})(\vec{X}) \neq L(\vec{X}) \,\big|\, \tilde{n}_+ = n_+, \tilde{n}_- = n_-\big)$. Here, $\hat{\mathbb{P}}(\cdot)$ is the estimate of $\mathbb{P}(\cdot)$ obtained by calculating the observed frequency of the event in question over multiple classifications using different training sets. An estimate of $\mathbb{P}\big(l(\tilde{T})(\tilde{X}) \neq L(\tilde{X}) \,\big|\, L(\tilde{X}) = Y, \tilde{n}_+ = n_+, \tilde{n}_- = n_-\big)$ for a given class $Y \in \mathcal{Y}$ can then be obtained by taking the mean of $\hat{\mathbb{P}}\big(l(\tilde{T})(\vec{X}) \neq L(\vec{X}) \,\big|\, \tilde{n}_+ = n_+, \tilde{n}_- = n_-\big)$ over all samples $\vec{X}$ in the dataset belonging to class Y. By doing this, we assume that the samples of the dataset belonging to a particular class $Y \in \mathcal{Y}$ reflect the distribution of $\mathbb{P}\big(\tilde{X} = \vec{X} \,\big|\, L(\tilde{X}) = Y\big)$. Note, however, that we do not assume that the class distribution of the dataset reflects a particular distribution.

In order to obtain multiple classification results using different training sets, we use a variation of the sub-sampled cross-validation method used by Smith et al. [2012], which has originally been proposed by Webb and Conilione [2003]. As mentioned in Section 9.2, it does not share the disadvantages of the holdout used by Mukherjee et al. [2003]. Every sample in the dataset is classified by the same number of trained classifiers. We have adapted the algorithm so that we are able to specify the number of samples per class in each training set instead of only the total number of samples. In addition, we have altered the method so that all samples from the same position on the substrate are grouped into the same fold, similar to the changes to the cross-validation method described in Section 8.3.3. The algorithm is listed in Algorithm 9.1.

In Lines 4 to 5, the number of folds $n_F \in \mathbb{N}_{\geqslant 2}$ for the cross-validation is calculated. n_F has to be high enough so that at least

Algorithm 9.1 : *EstimateErrorRates*(l, D, n_+, n_-, n_T)

```
Input : A classifier $l : \mathcal{D} \to \mathcal{Y}^{\mathcal{X}}$, a dataset $D \in \mathcal{D}$, the numbers
        $n_+, n_- \in \mathbb{N}_{>0}$ of positive and negative samples to be
        used in the training sets and the number $n_T \in \mathbb{N}_{>0}$ of
        training sets used to classify each sample.
Output : The estimated false negative rate $\hat{E}_{FN}(n_+, n_-)$ and false
         positive rate $\hat{E}_{FP}(n_+, n_-)$ of the classifier $l$ trained using
         $n_+$ positive and $n_-$ negative samples.
1  $D_+ := D \cap (\mathcal{X} \times \{Y_+\})$,   $D_- := D \cap (\mathcal{X} \times \{Y_-\})$.
2  Let $D'_1, \ldots, D'_m \sqsubseteq \mathcal{X} \times \mathcal{Y}$ be the elements of $D$ split by their
   substrate position so that $\bigcup_{i=1}^{m} D'_i = D$.
3  $\mathcal{D}' := \{D'_1, \ldots, D'_m\}$ as a multiset.
4  Let $i_F \in \mathbb{N}_{>0}, i_F < m$, be the highest number so that any $i_F$
      elements of $\mathcal{D}'$ contain at most $|D_+| - n_+$ positive and
      $|D_-| - n_-$ negative samples. // Max. number of positions
      per fold.
5  $n_F := \lceil \frac{m}{i_F} \rceil$. // The number of folds of the cross-validation.
6  $\hat{\mathbb{P}}_{\vec{X}} := 0 \,\forall\, (\vec{X}, Y) \in D$.
      // Stands for $\hat{\mathbb{P}}\left(l(\tilde{T})(\vec{X}) \neq L(\vec{X}) \mid \tilde{n}_+ = n_+, \tilde{n}_- = n_-\right)$.
7  for $i := 1, \ldots, n_T$ do
8     $F_0, \ldots, F_{n_F - 1} := \emptyset$ as multisets. // The CV folds.
9     $j := 0$.
10    while $|\mathcal{D}'| > 0$ do
11       Randomly choose $D'$ from $\mathcal{D}'$ with a uniform distribution.
12       $F_j := F_j \cup D'$.
13       $\mathcal{D}' := \mathcal{D}' \setminus \{D'\}$.
14       $j := (j + 1) \bmod n_F$.
15    end
      // Perform the cross-validation:
16    for $j := 0, \ldots, n_F - 1$ do
17       $T' := D \setminus F_j$.
         // Resample the training set:
18       Let $T_+$ be a random subset of $T' \cap D_+$ with size $n_+$.
19       Let $T_-$ be a random subset of $T' \cap D_-$ with size $n_-$.
20       $T := T_+ \cup T_-$.
21       for $(\vec{X}, Y) \in F_j$ do
            // Predict the class of $\vec{X}$ using the parameters
               $\theta$ and the training set $T$:
22          $Y' := l(T)(\vec{X})$.
23          if $Y \neq Y'$ then $\hat{\mathbb{P}}_{\vec{X}} := \hat{\mathbb{P}}_{\vec{X}} + \frac{1}{n_T}$.
24       end
25    end
26 end
27 $\hat{E}_{FN}(n_+, n_-) := \frac{1}{|D_+|} \sum_{(\vec{X}, Y) \in D_+} \hat{\mathbb{P}}_{\vec{X}}$.
28 $\hat{E}_{FP}(n_+, n_-) := \frac{1}{|D_-|} \sum_{(\vec{X}, Y) \in D_-} \hat{\mathbb{P}}_{\vec{X}}$.
29 return $\left(\hat{E}_{FN}(n_+, n_-), \hat{E}_{FP}(n_+, n_-)\right)$.
```

n_+ positive and n_- negative samples are present in every possible combination of $n_F - 1$ folds. On the other hand, it shall be as small as possible in order to reduce computation time. At the same time, we must consider that all samples of a single substrate position must be grouped into the same fold (see Section 8.3.3). Therefore, i_F stores the maximum number of substrate positions a single fold may contain so that it is guaranteed to contain at most $|D_+| - n_+$ positive and $|D_-| - n_-$ negative samples. This means that the number of folds n_F must be at least as high as the substrate position count m divided by the maximum number of positions per fold i_F. This is calculated in Line 5. Note that this limits the parameter n_+ to the number of positive samples in the dataset minus the maximum number of positive samples at a single substrate position. Similarly, n_- is limited to the number of negative samples in the dataset minus the maximum number of negative samples at a single substrate position.

After that, n_T cross-validations are performed. For each fold of every iteration, a training set containing n_+ positive and n_- negative samples is randomly assembled from the samples of the remaining folds. This is used to train a classifier, which is then used to classify all samples of the given fold. This means that every sample of the dataset is classified by n_T learned classifiers trained on training sets each containing n_+ positive and n_- negative samples. From the obtained classification results, the estimated false negative rate $\hat{E}_{FN}(n_+, n_-)$ and false positive rate $\hat{E}_{FP}(n_+, n_-)$ are calculated and returned.

As for the prediction phase, we essentially want to be able to choose any positive and negative sample counts $n_+, n_- \in \mathbb{N}_{>0}$ and predict the false negative and false positive rates $E_{FN}(n_+, n_-)$ and $E_{FP}(n_+, n_-)$ of a classifier trained on a set containing the chosen number of positive and negative samples.

As mentioned before, our prediction approach is based on power-law functions because theoretical and empirical evidence suggests that the error rate can be described as a function of this kind [Mukherjee et al., 2003; Smith et al., 2012]. These functions have the following form:

$$E(n_{\vec{X}}) \approx Q n_{\vec{X}}^{-\beta} + B. \tag{9.7}$$

Its parameters are the learning rate $Q \in \mathbb{R}_{\geqslant 0}$, the decay rate $\beta \in \mathbb{R}_{\geqslant 0}$ and the Bayes error $B \in [0, 1]$, which is the minimum achievable error rate [Mukherjee et al., 2003]. Their values are specific to a combination of classifier, parameters and underlying sample distribution. Furthermore, $n_{\vec{X}} \in \mathbb{N}_{>0}$ stands for the number of samples in the training set. In the classical interpretation, $E(n_{\vec{X}})$ stands for the misclassification rate of a classifier trained on a set with $n_{\vec{X}}$ samples. However, our research shows that this kind of function is also appropriate to predict the false negative and false positive rates of a classifier [Kockentiedt et al., 2014].

Algorithm 9.2 : *PredictErrorRates*$(l, D, n_+, n_-, n_T, n_{\hat{E}}, \tau_n)$

```
Input : A classifier $l : \mathcal{D} \to \mathcal{Y}^{\mathcal{X}}$, a dataset $D \in \mathcal{D}$, the numbers
        $n_+, n_- \in \mathbb{N}_{>0}$ of positive and negative samples of the
        predicted scenario and parameters controlling the
        prediction quality and time consumption of the algorithm:
        the number $n_T \in \mathbb{N}_{>0}$ of training sets used for each error
        estimation, the number $n_{\hat{E}} \in \mathbb{N}_{>0}$ of error estimations to
        perform and the fraction $\tau_n \in (0, 1]$ of the maximum
        possible number of samples to be used for the estimation.
Output : The predicted false negative rate $\hat{E}_{FN}(n_+, n_-)$ and false
         positive rate $\hat{E}_{FP}(n_+, n_-)$ of the classifier $l$ trained using
         $n_+$ positive and $n_-$ negative samples taken from the
         same sample space as $D$.
1  $D_+ := D \cap (\mathcal{X} \times \{Y_+\}), \quad D_- := D \cap (\mathcal{X} \times \{Y_-\})$.
2  Let $D'_1, \ldots, D'_m \sqsubseteq \mathcal{X} \times \mathcal{Y}$ be the elements of $D$ split by their
   substrate position so that $\bigcup_{i=1}^{m} D'_i = D$.
   // The maximum possible parameter values that can be
      passed to EstimateErrorRates():
3  $n_{+,\max} := |D_+| - \max_{i=1,\ldots,m} |D'_i \cap D_+|$.
4  $n_{-,\max} := |D_-| - \max_{i=1,\ldots,m} |D'_i \cap D_-|$.
5  $\tau := \tau_n \cdot \min\left(\frac{n_{+,\max}}{n_+}, \frac{n_{-,\max}}{n_-}\right)$.
6  for $i = 1, \ldots, n_{\hat{E}}$ do
7      $n_{+,i} := \mathrm{round}\left(\frac{i}{n_{\hat{E}}} \tau \cdot n_+\right)$.
8      $n_{-,i} := \mathrm{round}\left(\frac{i}{n_{\hat{E}}} \tau \cdot n_-\right)$.
9      $n_{\bar{X},i} := n_{+,i} + n_{-,i}$.
10     $\left(\hat{E}_{FN,i}, \hat{E}_{FP,i}\right) :=$ EstimateErrorRates$(l, D, n_{+,i}, n_{-,i}, n_T)$.
11 end
   // Fit two power-law functions of the form $E = Q n_{\bar{X}}^{-\beta} + B$
      to predict the error rates $E_{FN}$ and $E_{FP}$ for varying
      sample counts $n_{\bar{X}}$. This can be done using an
      appropriate non-linear least squares optimization
      algorithm:
12 $(Q_{FN}, \beta_{FN}, B_{FN}) := \underset{(Q_{FN}, \beta_{FN}, B_{FN})}{\arg\min} \sum_{i=1}^{n_{\hat{E}}} \left(Q_{FN} n_{\bar{X},i}^{-\beta_{FN}} + B_{FN} - \hat{E}_{FN,i}\right)^2$.
13 $(Q_{FP}, \beta_{FP}, B_{FP}) := \underset{(Q_{FP}, \beta_{FP}, B_{FP})}{\arg\min} \sum_{i=1}^{n_{\hat{E}}} \left(Q_{FP} n_{\bar{X},i}^{-\beta_{FP}} + B_{FP} - \hat{E}_{FP,i}\right)^2$.
   // Predict the error rates for the positive and negative
      sample counts $n_+$ and $n_-$:
14 $n_{\bar{X}} := n_+ + n_-$.
15 $\hat{E}_{FN}(n_+, n_-) := Q_{FN} n_{\bar{X}}^{-\beta_{FN}} + B_{FN}$.
16 $\hat{E}_{FP}(n_+, n_-) := Q_{FP} n_{\bar{X}}^{-\beta_{FP}} + B_{FP}$.
17 return $\left(\hat{E}_{FN}(n_+, n_-), \hat{E}_{FP}(n_+, n_-)\right)$.
```

The steps of our prediction approach are listed in Algorithm 9.2. Among others, the algorithm takes the parameters l, D, n_+ and n_-. It returns a prediction of the false negative rate and false positive rate of the classifier l trained on a set containing n_+ positive and n_- negative samples taken from the same sample space as the dataset D. As a first step, the method estimates the error rates of training sets having the same ratio of positive to negative samples as the prediction target. However, before that, the maximum values of n_+ and n_- that can be passed to *EstimateErrorRates* are calculated given that a single substrate position shall not be split among multiple folds of the cross validation. Note that we do only use τ_n times the maximum values for the estimation, where $\tau_n \in (0, 1]$. The reason is that if the maximum values were used, the number of folds of the cross validation would be equal to the number of substrate positions, which would lead to long computation times. In addition, there would be no inter-training-set variability in the estimation for some of the samples, which could likely introduce bias into the estimation. In other words, in *EstimateErrorRates*, some of the samples would be classified n_T times by classifiers trained on the exact same training set. Therefore, τ_n reduces the computation time and introduces inter-training-set variability.

The training set sizes for the estimation are then spaced equidistantly between 0 (exclusively) and τ_n times the maximum possible value (inclusively). The parameter $n_{\hat{E}} \in \mathbb{N}_{>0}$ determines the number of different training set sizes to be used and $n_T \in \mathbb{N}_{>0}$ defines the number of training sets to be used of each size.

Essentially, the parameters n_T, $n_{\hat{E}}$ and, to a certain degree, τ_n control a trade-off between shorter computation time and estimation accuracy where higher values tend to increase the computation time as well as the estimation quality. We have found that parameter values of $n_T = 100$, $n_{\hat{E}} = 5$ and $\tau_n = 0.5$ provide a good balance between computation time, estimation quality and inter-training-set variability and therefore use these values in our system. Note that $n_T = 100$ corresponds to twice the value used by Webb and Conilione [2003]. We have chosen this value in order to increase the estimation quality. The duration of a run of the algorithm of course depends on the used classifier-parameter combination. It is in the order of 6 hours for our test setup.

As the second step of the algorithm, two power-law functions are fitted to the estimates, one for each of false negative rate and false positive rate. We use the software library dlib by King [2009] to perform the non-linear least squares fitting using the Levenberg–Marquardt algorithm [Marquardt, 1963]. Finally, the power-law functions are used to calculate the predicted error rates, which are returned from the algorithm.

We can also predict various performance measures by calculating them from the predicted false positive and false negative rates. For example, G-mean can be calculated as described in Section 8.3.3.

9.4 EVALUATION

In this section, we want to evaluate the method proposed in the previous section. We want to assess how well it is able to predict the classification performance of a classifier when more samples are added to its training set.

In order to optimally evaluate the algorithm described in the previous section, one would need a large dataset with many CPGs. The reason for this is the following: In a real-world scenario, where this method would be applied, there would be an existing dataset, which we will call A. Then, the prediction method would estimate the classification error using at most 50 % of the CPGs of dataset A at a time in order to prevent bias. After that, it would predict the classification error of a training set B, which in turn would be larger than dataset A. In order to evaluate this prediction, one would need an even larger dataset C, which would need about twice as much CPGs as dataset B, again, in order to prevent bias.

If we would perform the evaluation in this way and would use our dataset as dataset C and we would assume that training set B is 25 % larger than dataset A, then the maximum number of Ag CPGs in an error estimation of our algorithm would be about 18. Moreover, the minimum number of Ag CPGs in an estimation would be about 3. We therefore think that our dataset is too small for this kind of evaluation.

Instead, we have decided to take the following approach:

1. For each of the three engineered nanoparticle types, do:
   a) From all particles in the dataset used for this thesis (see Section 4.4) pick all background CPGs and all CPGs containing engineered nanoparticles of the chosen type and use them to create an experiment dataset D.
   b) For $(n_+, n_-) \in \{(0.3 \cdot n_{+,\max}, 0.6 \cdot n_{-,\max}), (0.6 \cdot n_{+,\max}, 0.3 \cdot n_{-,\max}), (0.6 \cdot n_{+,\max}, 0.6 \cdot n_{-,\max})\}$ with $n_{+,\max}$ and $n_{-,\max}$ defined as in Lines 3 and 4 of Algorithm 9.2 on Page 190, do:
      i. Compute predictions $\left(\hat{E}_{FN}(n_+, n_-), \hat{E}_{FP}(n_+, n_-)\right) := \mathtt{PredictErrorRates}(l, D, n_+, n_-, n_T, n_{\hat{E}}, \tau_n)$ with the maximum particle fraction $\tau_n := 0.3$, the other parameter values as provided at the end of Section 9.3 and the classifier l that has been found to be best for D by our automatic model selection method (see Table 8.2 on Page 171).

ii. Compute estimates $(E_{FN}(n_+, n_-), E_{FP}(n_+, n_-)) :=$ `EstimateErrorRates`(l, D, n_+, n_-, n_T).

iii. Compare the predictions $\hat{E}_{FN}(n_+, n_-)$ and $\hat{E}_{FP}(n_+, n_-)$ to the estimates $E_{FN}(n_+, n_-)$ and $E_{FP}(n_+, n_-)$.

With this approach, the prediction algorithm is run almost unchanged on the whole dataset. The size of the training set whose classification performance shall be predicted, however, is smaller than the dataset. The number of positive and negative particles, respectively, is at most 0.6 times the maximum possible number given the current dataset. This is done in order not to get a biased estimation due to a too low training set variability. To prevent that this choice leads to the training set size whose behavior shall be predicted only being slightly larger than the one used for the predictions, we have changed τ_n from 0.5 to 0.3 for the experiment. This ensures that the training set size used for the predictions is at most half the size of the one whose behavior shall be predicted, which more closely resembles the ratios in a realistic scenario.

In some runs, we set the number of particles of one class to 0.3 times the maximum possible number in order to test the algorithm's behavior under different class ratios. In addition, we use `EstimateErrorRates()` to compute the ground truth, with which we can compare the predictions computed by `PredictErrorRates()`.

Note that we do not use our automatic model selection method inside the cross validations performed by the error estimation and prediction algorithms. Doing this would simply take too much time. Instead, we use the classifier-parameter combinations found to be best for the corresponding engineered nanoparticle types. They are listed in Table 8.2 on Page 171. This leads to classification performance estimations that are higher than the ones found in the evaluation in the previous chapter because the classifier-parameter combinations are tuned for the datasets they are used on. This, contrary to the previous chapter, poses no problem because we are not interested in the classification performance itself. Instead, we are interested in the prediction of its change while varying the training set size.

We have compared our algorithms to two baseline algorithms. The first one works exactly like our prediction method except that it fits a linear function instead of a power-law function to predict the classification error. The second one also uses a linear function. However, it only uses the two estimates with the highest CPG count to fit it. Figure 9.1 shows the results of this comparison. It displays the medians and lower and upper quartiles of the prediction error of the three algorithms among all runs of our evaluation. The data is split by the type of predicted error rate. The prediction error in our experiments has on average been lower when predicting the false positive rate, that is the percentage of negative samples being labeled as positive

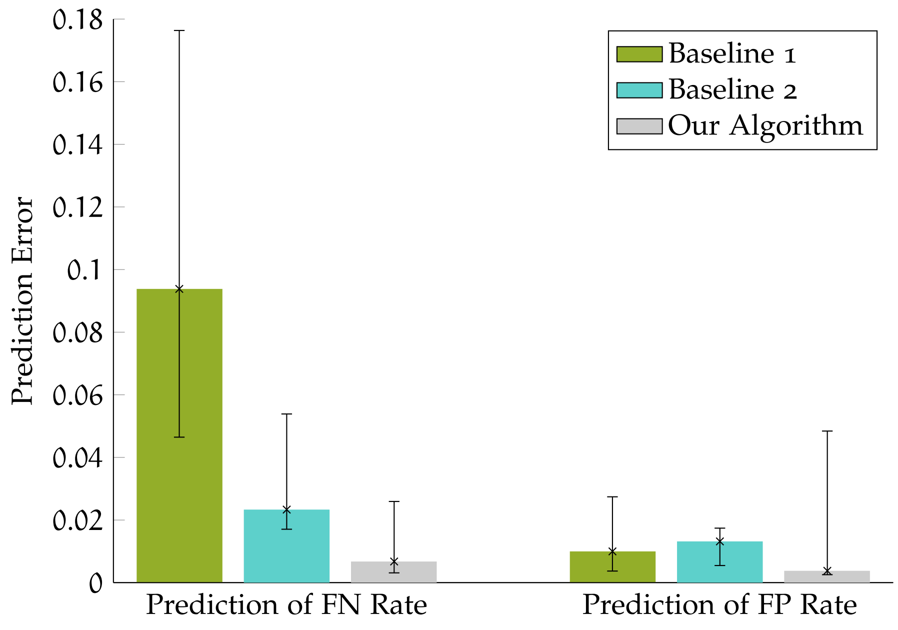


Figure 9.1: The prediction errors defined as the absolute difference between the prediction and the estimate ($|\hat{E}_{FN}(n_+, n_-) - E_{FN}(n_+, n_-)|$ or $|\hat{E}_{FP}(n_+, n_-) - E_{FP}(n_+, n_-)|$, respectively) of our algorithm compared to the two baseline algorithms. The bars show the median prediction errors while the whiskers show the lower and upper quartiles, respectively.

by the classifier. The reason for this is that, for our data, the false positive rate itself is generally lower than the false negative rate.

Our method performs better than the baseline algorithms for both error rates. For the false negative rate, the median prediction error of the first baseline (0.0938) is more than 13 times that of our algorithm (0.0068) while that of the second baseline (0.0233) is over 3 times as high. When predicting the false positive rate, the first baseline median (0.0100) and the second baseline median (0.0132) are over 2.5 and 3 times as high as the median prediction error of our algorithm (0.0038), respectively.

While these results show that our method generally performs better than the baselines, the upper quartile of the prediction errors for the false positive rate of our algorithm is an exception. It is worse than those of the baselines and even worse than that of our algorithm when predicting the false negative rate. The reason is that in some runs of our experiment, the false positive rate does not resemble a power-law function at all. An example of this behavior can be seen in Figure 9.2. Here, the relation between CPG count of the training set and false positive rate of the trained classifier seems unpredictable. As a comparison, Figure 9.3 shows the relationship between CPG count of the training set and the false negative rate for another run of our evaluation. The line fitted to all data points shows very well

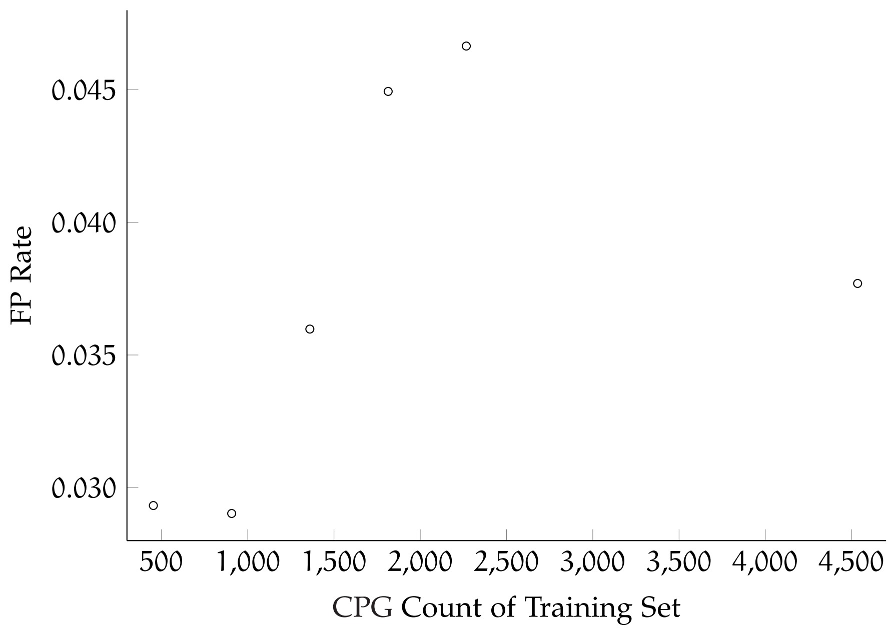


Figure 9.2: False positive rate estimates for a run of our evaluation for Ag nanoparticles. Each point represents the estimated false positive rate of a classifier trained with the given number of CPGs.

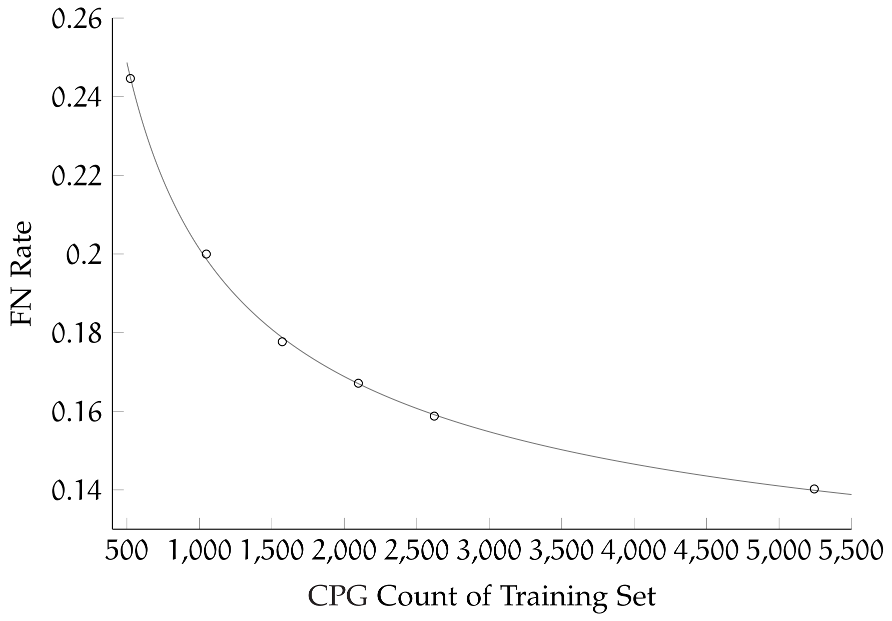


Figure 9.3: False negative rate estimates for a run of our evaluation for ZnO nanoparticles. Each point represents the estimated false negative rate of a classifier trained with the given number of CPGs. The line represents a power-law function that has been fitted to all data points.

that they closely follow a power-law function, which is typical for the false negative rate in our experiment.

We believe that a part of the reason for the unpredictable behavior of the false positive rate in some examples lies in the small overall variation of the rate. It is usually much smaller than that of the false negative rate for the data used in this thesis. For example, the false positive rates of all points in Figure 9.2 lie within a span of 0.02 of each other while the false negative rates in Figure 9.3 span a range of over 0.1. This increases the influence of the prediction error for the false positive rate, which makes a prediction more difficult.

However, the small variation of the false positive rate compared to the false negative rate also reduces its influence on an overall classification performance measure such as G-mean. This can be seen when we combine the predictions of the two error rates and calculate a G-mean prediction from them. Over all runs of our evaluation, the median error of the G-mean prediction relative to the change of the estimated G-Means from the last value available to the prediction algorithm to the value to predict is 0.392 for our method compared to 4.7340 and 1.0612 for the baselines. In other words, half of the time, the prediction of our method differs from the G-mean to predict less than 0.4 times as much as the last G-Mean available to the algorithm.

In this chapter, we have proposed an algorithm to predict the classification performance of our system given more training samples. In the next, the final chapter, we will look at all parts of our system and make some concluding remarks.

10 CONCLUSIONS

In Chapters 1 to 4, we have motivated our work, described the circumstances it shall be used in and laid out our goals. Then, Chapter 5 has introduced related literature and, based on that, proposed the basic architecture of our system. In Chapters 6 to 9, we have presented the related work, algorithms and evaluation of each of the steps of our system, where each step has been presented in its own chapter. We have chosen this layout so that the literature discussion and algorithm description are still fresh in the readers mind when reading an evaluation section.

This chapter will revisit all parts of our system by providing a conclusion to this thesis. In particular, we will address the following points:

- Shortly summarize the contributions of this thesis.
- Assess which of the goals we have set ourselves have been met.
- Look at possible future directions of research related to the topics of this thesis.

We will begin with our contributions in the following section. The other points will be covered in the two sections after that.

10.1 CONTRIBUTIONS OF THIS THESIS

To the best of our knowledge, the work described in this thesis is the first automatic solution to image-based identification of engineered nanoparticles in a realistic scenario. Therefore, much thought and many decisions have been necessary to build this system. In particular, we have had to make decisions about the overall approach of the system as well as which algorithm categories and implementations to use. In addition, we have developed several algorithms ourselves appropriate for the characteristics of the task and the targeted nanoparticles. In the order of appearance in this thesis, this section will shortly name our main contributions.

We have compiled and reviewed related literature on automatic image-based particle analysis in Chapter 5 and the related work sections of Chapters 6 to 8. Only little of that work could be directly applied to our problem because the scenarios of the examined literature are diverse and considerably different from ours. Therefore, we have made the effort to analyze and describe in detail which approaches of the literature can and cannot be used for our problem and why.

In Section 6.3.1, we have proposed a completely new noise parameter estimation algorithm specifically targeted at SEM images. It is derived using statistical means from a realistic image generation model. As far as we know, it is the first such algorithm targeted at SEM. On simulated images, it shows very good results on its own (see Section 6.4.1) as well as combined with a noise removal method (see Section 6.4.2).

We have selected suitable feature categories and features for the unique properties of nanoparticle agglomerates in Chapter 7. We could not directly adopt the features used in the literature because, for agglomerates, the global form and size is very variable. For the same reason, we have created a new set of features called Mean Contour Angle Wavelet Response (see Section 7.3.2.1), which is able to extract the absolute sizes of local contour features.

In Section 7.3.3.3, we have introduced a new method to estimate the number of electrons detected by the detector for each pixel of an SEM image. It uses the same derivation as the noise parameter estimation mentioned above and is the only such algorithm we are aware of. It performs so well that two of the three features derived from it have been found to be the most valuable features used by us in our evaluation (see Section 8.4.2).

For the classification step of our system, we have selected appropriate classifiers, parameters, performance measures and preprocessing steps such as dimensionality reduction, weighting and resampling (see Chapter 8). For this, we have had to consider the characteristics of the classification problem such as small sample sizes, class imbalance and unknown target class ratios. In addition, we have compiled an automatic model selection method from existing evolutionary algorithms in order to make the system adaptable to new kinds of nanoparticles without the need for a classification expert (see Section 8.3.2).

Finally, in Chapter 9, we have created a new method to predict the classification performance of a classifier trained on a dataset of different size and composition. For that, we have combined two existing methods and extended them to be able to predict the false positive and false negative rates as well as take the class ratio into account.

10.2 GOAL REVIEW

At the beginning of this thesis, we have stated that the main goal of our system is to save time the user spends in front of the computer compared to a manual analysis of the images. A prerequisite for this is that it meets the minimum goals stated in Chapter 3:

DETECTION Locate the engineered nanoparticles and (optionally) the background particles in every image.

IDENTIFICATION Decide for every found particle if it either consists of engineered nanoparticles of a specific type or is a background particle.

We have evaluated the detection goal in Section 6.4.5. There, we have found that the segmentation pipeline can match human performance regarding the number of found CPGs. In fact, it finds all CPGs containing engineered nanoparticles in the randomly chosen images we have looked at in our evaluation. Thus, it meets the first requirement.

The particle identification has been evaluated in Section 8.4.1. Again, this has shown that the system can match the performance of human experts. This means that our system fulfills the minimum goals defined in this thesis.

Having established this, we can look at the time that can be saved using our system. We have investigated this in Section 8.4.3. For the examined scenario, taking sampling, imaging, training of experts or the system, respectively, and classification into account, we have found that the system can save 3 h of the user's time. This accounts for savings of 22.9 %. Assuming that the system has been trained before for the targeted nanoparticle type, the savings rise to 43.5 %. When only regarding the work after the images have been created, the user only has to provide the images to the system instead of spending 3 h and 36 min detecting and classifying particles.

In addition to our main goal of saving time for the user, we have defined further goals. One of them is future viability. This means that our system shall be able to adapt to new particle types without the need for a classification expert. Therefore, we have proposed a fully automatic model selection method in Section 8.3.2. It has proven its merit in Section 8.4.1 where it has been used to select the classifier-parameter combinations used to evaluate the classification performance of our system. We will give more detail on the quality of these classification results below.

Until now, we have established that our system is able to save time while matching human performance. Next, we will assess whether it can provide insights that exceed those provided by human experts. First, we want to look at the particle detection and identification quality. In our segmentation evaluation, our system has been able to find all CPGs containing engineered nanoparticles while only 91 % have been found manually (see Section 6.4.5). In the classification evaluation, our system has yielded a G-mean of 0.931 for Ag compared to 0.778 achieved by human experts. For TiO_2, the result has been 0.854 versus 0.767 and for ZnO, the system has achieved 0.861 compared to 0.705 (see Section 8.4.1).

In Chapter 9, we have shown that our system is able to predict the improvement in classification performance if it is given more training data. This helps the user to decide whether the additional work needed to produce such sample images is worth the added classification precision. It is much more difficult for human experts to assess and predict their classification ability given longer training using more sample images with a reasonable precision. While it is conceivable to devise a procedure to make the same predictions for a human expert, it would certainly involve a considerable amount of time spent classifying particles for evaluation purposes. And because a human is not able to easily unlearn previously gained knowledge, it would probably require more training images than to predict the systems performance. Thus, such a procedure would be infeasible.

Finally, while a human can count the number of CPGs containing engineered nanoparticles in an image, our system captures the exact form and size of the image area covered by these CPGs. The former allows to estimate the number of CPGs per volume of air, the so-called number concentration. Additionally, the image area recorded by our system allows for the estimation of other forms of concentration:

- The number of engineered primary particles per volume of air.
- The volume of engineered nanoparticles per volume of air (volume concentration).
- The surface of engineered nanoparticles per volume of air (surface concentration).

While the estimation of these numbers is beyond the scope of this thesis, especially the surface concentration seems to be very important in predicting the potential toxicity of nanoparticles [Crosera et al., 2009].

In conclusion, we have proposed a system that is able to significantly reduce the manual work needed to analyze airborne nanoparticles while maintaining or even improving the quality of the analysis.

10.3 FUTURE WORK

Since our system is the first of its kind, it is a proof of concept. By definition, this means that there may be better algorithms for some of its parts. In this section, we will outline some areas where we think future research may be able to improve the system.

On Page 49, we have briefly mentioned CCSEM, where a computer controls the SEM and automatically takes particle images. If future generations of such tools would be able to detect particles as small as the nanoparticles to be found, they could be a valuable tool for a system like ours. In our opinion, it could have the following benefits:

- Having more training data would probably improve the classification quality of our system.
- It would reduce the manual work needed for sample analysis even further.

- It could perform an EDX analysis for every found particle providing valuable information about its composition.

While our system has been able to locate all CPGs with engineered nanoparticles in our evaluation in Section 6.4.5, it has overlooked some small and faint background particles in proximity to a large bright particle. We think that the possible solutions we have considered such as dynamic thresholding would have difficulties to cope with the highly variable size of the agglomerates. Nevertheless, it would be beneficial to find a segmentation method that is able to also find such small and faint particles.

As described in Section 6.1, we have assumed that engineered nanoparticles and background particles never appear in the same agglomerate. However, as said at the beginning of Section 8.4, we have observed a few agglomerates which contain both types of particles. Future research could look into how such cases can be handled better.

Our noise and electron count estimation algorithms described in Sections 6.3.1 and 7.3.3.3 work very well as seen in Sections 6.4.1 and 8.4.2. However, as described on Page 74, intensity clipping produces outliers in the parameter fitting. The noise model by Foi et al. [2008] is able to model this clipping. It would be nice if this could be incorporated in the algorithm proposed by us.

The features we have derived from the estimated electron count (minimum, maximum and mean) work very well in our evaluation (see Section 8.4.2). It would be interesting to see if more complicated features such as a histogram or Haralick features computed from the electron count estimates of all pixels would perform equally well.

We think that directly estimating the primary particle distribution of an agglomerate could be another valuable goal. This data could be used to aid the classification or to directly estimate different concentrations indicative of a worker's health risk.

APPENDIX A: ADDITIONAL MATH

This appendix collects formulas that would disturb the flow of the main text.

Using the definitions from Section 6.3.1 and any function $\Gamma : \mathbb{N} \times \mathbb{N} \to \mathbb{R}$, we can show:

$$\begin{aligned}
&\mathbb{E}(\Gamma(\tilde{c}_1, \tilde{c}_2) \,|\, \tilde{c}_1 + \tilde{c}_2 = c) \\
&= \sum_{\substack{i,j \in \mathbb{N} \\ i+j=c}} \Gamma(i,j)\, \mathbb{P}(\tilde{c}_1 = i, \tilde{c}_2 = j \,|\, \tilde{c}_1 + \tilde{c}_2 = c) \\
&= \sum_{\substack{i,j \in \mathbb{N} \\ i+j=c}} \Gamma(i,j) \frac{\mathbb{P}(\tilde{c}_1 = i, \tilde{c}_2 = j)}{\mathbb{P}(\tilde{c}_1 + \tilde{c}_2 = c)} \\
&= \sum_{\substack{i,j \in \mathbb{N} \\ i+j=c}} \Gamma(i,j) \frac{\mathbb{P}(\tilde{c}_1 = i)\, \mathbb{P}(\tilde{c}_2 = j)}{\mathbb{P}(\tilde{c}_1 + \tilde{c}_2 = c)} \\
&= \sum_{\substack{i,j \in \mathbb{N} \\ i+j=c}} \Gamma(i,j) \frac{\frac{\mu^i}{i!} e^{-\mu} \frac{\mu^j}{j!} e^{-\mu}}{\frac{(\mu+\mu)^c}{c!} e^{-(\mu+\mu)}} \\
&= \sum_{\substack{i,j \in \mathbb{N} \\ i+j=c}} \Gamma(i,j) \frac{\mu^i \mu^j c!}{(\mu+\mu)^c i! j!} \\
&= \sum_{i=0}^{c} \Gamma(i, c-i) \frac{\mu^i \mu^{c-i} c!}{2^c \mu^c i! (c-i)!} \\
&= \frac{1}{2^c} \sum_{i=0}^{c} \Gamma(i, c-i) \binom{c}{i}, \quad \forall c \in \mathbb{N}.
\end{aligned} \tag{A.1}$$

Here, we have used the fact that the sum of two independent Poisson distributed random variables is itself Poisson distributed.

B

APPENDIX B: ADDITIONAL ALGORITHMS

This appendix presents algorithms that do not fit in the main text but may be of interest to the reader.

Algorithm B.1 shows the iteratively reweighted least squares algorithm [Maronna et al., 2006, p. 105].

Algorithm B.1 : *IterativelyReweightedLeastSquares*($\overset{\square}{s}, \vec{v}, \vec{\alpha}_0$)

```
Input : A matrix $\overset{\square}{s} \in \mathbb{R}^{l \times m}$, a vector $\vec{v} \in \mathbb{R}^l$ and an initial estimate
        $\vec{\alpha}_0 \in \mathbb{R}^m$, where $l, m \in \mathbb{N}_{>0}, l \gg m$.
Output : An estimate $\vec{\alpha} \in \mathbb{R}^m$ as a robust approximate solution
         to the equation $\overset{\square}{s}\vec{\alpha} = \vec{v}$.
$\vec{r}_0 := \vec{v} - \overset{\square}{s}\vec{\alpha}_0$.
$k := 0$
repeat
    $\hat{\sigma}_k := \underset{i=1,\dots,l}{\mathrm{med}} (|\vec{r}_k(i)|)/0.675$. // med denotes the median.
    $\overset{\square}{W}_k := \underset{i=1,\dots,l}{\mathrm{diag}} \left( \begin{cases} \left(1 - \left(\frac{\vec{r}_k(i)}{4.685\hat{\sigma}_k}\right)^2\right)^2, & \text{if } \left|\frac{\vec{r}_k(i)}{\hat{\sigma}_k}\right| \leqslant 4.685 \\ 0, & \text{if } \left|\frac{\vec{r}_k(i)}{\hat{\sigma}_k}\right| > 4.685 \end{cases} \right)$.
      // diag generates a diagonal matrix with the given
      diagonal entries.
    $\vec{\alpha}_{k+1} = \left(\overset{\square}{s}^T \overset{\square}{W}_k \overset{\square}{s}\right)^{-1} \overset{\square}{s}^T \overset{\square}{W}_k \vec{v}$.
    $\vec{r}_{k+1} := \vec{v} - \overset{\square}{s}\vec{\alpha}_{k+1}$.
    $k := k + 1$
until $\max_i (|\vec{r}_k(i) - \vec{r}_{k-1}(i)|)/\hat{\sigma}_{k-1} < \varepsilon$.
// $\varepsilon \in \mathbb{R}_{>0}$ is a small constant.
return $\vec{\alpha} := \vec{\alpha}_k$.
```

BIBLIOGRAPHY

D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66, 1991. (Cited on page 167.)

F. J. Anscombe. The transformation of Poisson, binomial and negative-binomial data. *Biometrika*, 35(3/4):246–254, 1948. (Cited on pages 80 and 81.)

I. ap Gwynn and C. Wilson. Characterizing fretting particles by analysis of SEM images. *European Cells & Materials*, 1:1–11, 2001. (Cited on pages 48, 50, 53, 59, 108, and 109.)

N. Azong-Wara, C. Asbach, B. Stahlmecke, H. Fissan, H. Kaminski, S. Plitzko, D. Bathen, and T. a. J. Kuhlbusch. Design and experimental evaluation of a new nanoparticle thermophoretic personal sampler. *Journal of Nanoparticle Research*, 15(4):1530, 2013. (Cited on page 31.)

J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(1):281–305, 2012. (Cited on page 139.)

G. Bhabra, A. Sood, B. Fisher, L. Cartwright, M. Saunders, W. H. Evans, A. Surprenant, G. Lopez-Castejon, S. Mann, S. a. Davis, L. a. Hails, E. Ingham, P. Verkade, J. Lane, K. Heesom, R. Newson, and C. P. Case. Nanoparticles can cause DNA damage across a cellular barrier. *Nature Nanotechnology*, 4(12):876–83, 2009. (Cited on page 2.)

C. M. Bishop. *Pattern recognition and machine learning*. Springer, 2006. (Cited on pages 135, 144, and 145.)

A. Brink. Grey-level thresholding of images using a correlation criterion. *Pattern Recognition Letters*, 9(5):335–341, 1989. (Cited on page 60.)

A. Buades, B. Coll, and J. M. Morel. A review of image denoising algorithms, with a new one. *SIAM Journal on Multiscale Modeling and Simulation*, 4(2):490–530, 2005a. (Cited on page 80.)

A. Buades, B. Coll, and J.-M. Morel. A non-local algorithm for image denoising. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 60–65. IEEE, 2005b. (Cited on pages 80, 95, and 96.)

Bundesanstalt für Arbeitsschutz und Arbeitsmedizin (BAuA). Stichprobenplanung für die Auswertung von Messungen mit dem Thermalpräzipitator. Unpublished report, 2012. (Cited on pages 10 and 18.)

J. Calderon De Anda, X. Wang, X. Lai, and K. Roberts. Classifying organic crystals via in-process image analysis and the use of monitoring charts to follow polymorphic and morphological changes. *Journal of Process Control*, 15(7):785–797, 2005. (Cited on pages 48, 50, 58, 110, and 134.)

E. Cantú-Paz. Feature subset selection by estimation of distribution algorithms. In *Genetic and Evolutionary Computation Conference*, 2002. (Cited on page 150.)

K. Cha, H.-W. Hong, Y.-G. Choi, M. J. Lee, J. H. Park, H.-K. Chae, G. Ryu, and H. Myung. Comparison of acute responses of mice livers to short-term exposure to nano-sized or micro-sized silver particles. *Biotechnology Letters*, 30(11):1893–9, 2008. (Cited on page 2.)

N. V. Chawla, K. W. Bowyer, and L. O. Hall. SMOTE: Synthetic minority over-sampling technique. *Journal Of Artificial Intelligence Research*, 16:321–357, 2002. (Cited on page 147.)

M. Crosera, M. Bovenzi, G. Maina, G. Adami, C. Zanette, C. Florio, and F. Filon Larese. Nanoparticle dermal absorption and toxicity: A review of the literature. *International Archives of Occupational and Environmental Health*, 82(9):1043–1055, 2009. (Cited on pages 1, 10, and 200.)

P. Dokas, L. Ertoz, V. Kumar, A. Lazarevic, J. Srivastava, and P.-N. Tan. Data mining for network intrusion detection. In *National Science Foundation Workshop on Next Generation Data Mining*, pages 21–30, 2002. (Cited on page 136.)

K. Donaldson and A. Seaton. A short history of the toxicology of inhaled particles. *Particle and Fibre Toxicology*, 9(1):13, 2012. (Cited on page 13.)

E. Donskoi, S. Suthers, S. Fradd, J. Young, J. Campbell, T. Raynlyn, and J. Clout. Utilization of optical image analysis and automatic texture classification for iron ore particle characterisation. *Minerals Engineering*, 20(5):461–471, 2007. (Cited on page 47.)

H. Drolon, F. Druaux, and A. Faure. Particles shape analysis and classification using the wavelet transform. *Pattern Recognition Letters*, 21(6-7):473–482, 2000. (Cited on pages 48, 50, 109, 111, and 118.)

R. Fisker, J. Carstensen, M. Hansen, F. Bødker, and S. Mørup. Estimation of nanoparticle size distributions by image analysis. *Journal of*

Nanoparticle Research, 2(3):267–277, 2000. (Cited on pages 47, 48, 50, 53, 58, 108, and 109.)

A. B. Flores, L. A. Robles, M. O. Arias, and J. A. Ascencio. Small metal nanoparticle recognition using digital image analysis and high resolution electron microscopy. *Micron*, 34(2):109–118, 2003. (Cited on pages 47, 48, 49, 50, 61, 111, and 126.)

A. Foi. Clipped noisy images: Heteroskedastic modeling and practical denoising. *Signal Processing*, 89(12):2609–2629, 2009. (Cited on page 79.)

A. Foi, M. Trimeche, V. Katkovnik, and K. Egiazarian. Practical Poissonian-Gaussian noise modeling and fitting for single-image raw-data. *IEEE Transactions on Image Processing*, 17(10):1737–54, 2008. (Cited on pages 64 and 201.)

G. Forman and I. Cohen. Learning from little: Comparison of classifiers given little training. In *Knowledge Discovery in Databases: PKDD 2004*, pages 161–172, 2004. (Cited on pages 147, 150, and 162.)

F. Friedrichs and C. Igel. Evolutionary tuning of multiple SVM parameters. *Neurocomputing*, 64:107–117, 2005. (Cited on pages 140 and 150.)

H. Fröhlich, O. Chapelle, and B. Schölkopf. Feature selection for support vector machines by means of genetic algorithm. In *15th IEEE International Conference on Tools with Artificial Intelligence*, number December, pages 142–148. IEEE Comput. Soc, 2002. (Cited on page 150.)

Future Markets Inc. The world market for nanoparticle titanium dioxide. Report, 2011. URL `http://www.researchandmarkets.com/reports/1651709/the_world_market_for_nanoparticle_titanium`. (Cited on page 1.)

A. Genga, F. Baglivi, M. Siciliano, T. Siciliano, M. Tepore, G. Micocci, C. Tortorella, and D. Aiello. SEM-EDS investigation on PM10 data collected in Central Italy: Principal component analysis and hierarchical cluster analysis. *Chemistry Central Journal*, 6(Suppl 2):S3, 2012. (Cited on pages 48, 50, 53, 107, 108, and 109.)

M. S. Germani and P. R. Buseck. Automated scanning electron microscopy for atmospheric particle analysis. *Analytical Chemistry*, 63(20): 2232–2237, 1991. (Cited on pages 48, 50, 53, 59, 106, 107, and 108.)

J. Goldstein, D. E. Newbury, D. C. Joy, C. E. Lyman, P. Echlin, E. Lifshin, L. Sawyer, and J. R. Michael. *Scanning electron microscopy and X-ray microanalysis*. Springer, third edition, 2003. (Cited on pages 18, 33, 34, 36, 38, 40, 41, and 128.)

L. Gonzalez, D. Lison, and M. Kirsch-Volders. Genotoxicity of engineered nanomaterials: A critical review. *Nanotoxicology*, 2(4):252–273, 2008. (Cited on page 2.)

A. Greco and A. Maffezzoli. Powder-shape analysis and sintering behavior of high-density polyethylene powders for rotational molding. *Journal of Applied Polymer Science*, 92(1):449–460, 2004. (Cited on pages 48, 50, 53, 108, 109, 110, and 111.)

J. J. Grefenstette. Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics*, 16(1): 122–128, 1986. (Cited on pages 151 and 158.)

Grimm Aerosol Technik. Scanning mobility particle sizer (SMPS+C). Datasheet, 2012. URL `http://wiki.grimm-aerosol.de/images/6/6c/GrimmAerosolTechnik_Nano_SMPS_C.pdf`. (Cited on page 16.)

M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: An update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009. (Cited on page 163.)

H. Handels, T. Ross, J. Kreusch, H. H. Wolff, and S. J. Pöppl. Feature selection for optimized skin tumor recognition using genetic algorithms. *Artificial Intelligence in Medicine*, 16(3):283–297, 1999. (Cited on page 150.)

C. Hans, F. A. Merchant, and S. K. Shah. Decision fusion for urine particle classification in multispectral images. In *Proceedings of the Seventh Indian Conference on Computer Vision, Graphics and Image Processing - ICVGIP '10*, pages 419–426, New York, New York, USA, 2010. ACM Press. (Cited on pages 48, 50, 61, 62, and 112.)

R. Haralick, K. Shanmugan, and I. Dinstein. Textural features for image classification. *IEEE Transactions on Systems, Man and Cybernetics*, 3(6):610–621, 1973. (Cited on page 126.)

H. He and E. A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009. (Cited on page 147.)

G. Healey and R. Kondepudy. Radiometric CCD camera calibration and noise estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(3):267–276, 1994. (Cited on pages 64 and 71.)

P. K. Hopke and X.-H. Song. Classification of single particles by neural networks based on the computer-controlled scanning electron microscopy data. *Analytica Chimica Acta*, 348(1-3):375–388, 1997. (Cited on pages 48, 50, and 107.)

C.-W. Hsu, C.-C. Chang, and C.-J. Lin. A practical guide to support vector classification. Technical report, National Taiwan University, Taipei, 2010. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf`. (Cited on page 139.)

H. Ito, K. Kamimura, N. Tsumura, T. Nakaguchi, H. Motomura, and Y. Miyake. Noise estimation from a single image taken by specific digital camera using a priori information. In *Proceedings of SPIE*, volume 6808. SPIE, 2008. (Cited on page 64.)

A. K. Jain, R. P. Duin, and J. Mao. Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–37, 2000. (Cited on pages 52, 53, 132, 133, 135, 136, 137, 143, and 144.)

N. Japkowicz and M. Shah. *Evaluating learning algorithms: A classification perspective*. Cambridge University Press, 2011. (Cited on pages 137, 138, and 166.)

H. J. Johnston, G. Hutchison, F. M. Christensen, S. Peters, S. Hankin, and V. Stone. A review of the in vivo and in vitro toxicity of silver and gold particulates: Particle attributes and biological mechanisms responsible for the observed toxicity. *Critical Reviews in Toxicology*, 40(4):328–46, 2010. (Cited on page 2.)

K. Jong, E. Marchiori, and A. van der Vaart. Analysis of proteomic pattern data for cancer detection. In *Applications of Evolutionary Computing*, pages 41–51, 2004. (Cited on page 150.)

D. C. Joy. Noise and its effects on the low-voltage SEM. In H. Schatten and J. B. Pawley, editors, *Biological low-voltage scanning electron microscopy*, chapter 4, pages 129–144. Springer, 2008. (Cited on pages 64 and 65.)

V. V. Kindratenko and P. J. M. Van Espen. Classification of irregularly shaped micro-objects using complex Fourier descriptors. In *Proceedings of 13th International Conference on Pattern Recognition (Volume 2)*, pages 285–289. IEEE, 1996. (Cited on pages 48, 50, 110, and 134.)

V. V. Kindratenko, P. J. M. Van Espen, B. A. Treiger, and R. E. Van Grieken. Fractal dimensional classification of aerosol particles by computer-controlled scanning electron microscopy. *Environmental Science & Technology*, 28(12):2197–202, 1994. (Cited on pages 48, 50, 53, 59, 106, and 109.)

V. V. Kindratenko, P. J. M. Van Espen, and B. A. Treiger. Characterisation of the shape of microparticles via fractal and Fourier analyses of scanning electron microscope images. *Mikrochimica Acta*, suppl. 13:355–361, 1996. (Cited on pages 48, 50, 53, 107, 109, and 110.)

V. V. Kindratenko, B. A. Treiger, and P. J. M. Van Espen. Classification of silver halide microcrystals via K-NN clustering of their shape descriptors. *Journal of Chemometrics*, 11(2):131–139, 1997. (Cited on pages 48, 50, 109, 110, and 134.)

D. E. King. Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10:1755–1758, 2009. URL `http://jmlr.csail.mit.edu/papers/v10/king09a.html`. (Cited on page 191.)

J. Kittler and J. Illingworth. Minimum error thresholding. *Pattern Recognition*, 19(1):41–47, 1986. (Cited on pages 86, 98, and 99.)

S. Kockentiedt, U. Gernert, K. Tönnies, and E. Gierke. A new level-dependent noise reduction method applied to high resolution SEM images. In R. Rachel, editor, *Proceedings Microscopy Conference 2013, Part 2*, pages 604–605, Regensburg, 2013. (Cited on page 80.)

S. Kockentiedt, K. Tönnies, and E. Gierke. Predicting the influence of additional training data on classification performance for imbalanced data. In X. Jiang, J. Hornegger, and R. Koch, editors, *Proceedings of the 36th German Conference on Pattern Recognition, GCPR 2014*, volume 8753 of *Lecture Notes in Computer Science*, pages 377–387. Springer International Publishing, 2014. (Cited on pages 187 and 189.)

R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *International Joint Conference on Artificial Intelligence*, volume 14, pages 1137–1143, 1995. (Cited on page 159.)

R. Kohavi and D. H. Wolpert. Bias plus variance decomposition for zero-one loss functions. In *Proceedings of the 13th International Conference on Machine Learning*, pages 275–283, 1996. (Cited on page 185.)

H. Kohl and L. Reimer. *Transmission electron microscopy: Physics of image formation*. Springer, New York, fifth edition, 2008. (Cited on page 49.)

A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 25, pages 1097–1105, 2012. (Cited on page 136.)

D.-J. Kroon. Fast non-local means 1D, 2D color and 3D, 2010. URL `http://www.mathworks.com/matlabcentral/fileexchange/27395-fast-non-local-means-1d--2d-color-and-3d`. (Cited on page 82.)

L. Kuncheva. Genetic algorithm for feature selection for parallel classifiers. *Information Processing Letters*, 46(4):163–168, 1993. (Cited on page 150.)

M. S. Laghari. Recognition of texture types of wear particles. *Neural Computing & Applications*, 12(1):18–25, 2003. (Cited on pages 48, 50, 107, 108, 109, 111, 126, and 134.)

M. Langford, G. Taylor, and J. Flenley. Computerized identification of pollen grains by texture analysis. *Review of Palaeobotany and Palynology*, 64(1-4):197–203, 1990. (Cited on pages 48, 50, 53, 111, 126, and 134.)

S. Le Cessie and J. C. Van Houwelingen. Ridge estimators in logistic regression. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 41(1):191–201, 1992. (Cited on page 144.)

C. Liu, R. Szeliski, S. Bing Kang, C. L. Zitnick, and W. T. Freeman. Automatic estimation and removal of noise from a single image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2): 299–314, 2008. (Cited on page 64.)

T. Luo, K. Kramer, D. Goldgof, L. Hall, S. Samson, A. Remsen, and T. Hopkins. Recognizing plankton images from the shadow image particle profiling evaluation recorder. *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, 34(4):1753–1762, 2004. (Cited on pages 48, 50, 107, and 134.)

M. Mäkitalo and A. Foi. A closed-form approximation of the exact unbiased inverse of the Anscombe variance-stabilizing transformation. *IEEE Transactions on Image Processing*, 20(9):2697–2698, 2011a. (Cited on pages 80 and 81.)

M. Mäkitalo and A. Foi. Optimal inversion of the Anscombe transformation in low-count Poisson image denoising. *IEEE Transactions on Image Processing*, 20(1):99–109, 2011b. (Cited on page 80.)

R. A. Maronna, R. D. Martin, and V. J. Yohai. *Robust statistics*. Wiley Series in Probability and Statistics. John Wiley & Sons, Ltd, 2006. (Cited on pages 74, 78, 79, and 205.)

D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441, 1963. (Cited on page 191.)

H. G. Merkus. *Particle size measurements: Fundamentals, practice, quality*. Springer, 2009. (Cited on page 108.)

M. Methner, L. Hodson, and C. Geraci. Nanoparticle emission assessment technique (NEAT) for the identification and measurement of potential inhalation exposure to engineered nanomaterials: Part A. *Journal of Occupational and Environmental Hygiene*, 7(3):127–132, 2009. (Cited on pages 2, 3, and 9.)

M. T. Miller, A. K. Jerebko, J. D. Malley, and R. M. Summers. Feature selection for computer-aided polyp detection using genetic algorithms. In *Proc. SPIE 5031, Medical Imaging 2003: Physiology and Function: Methods, Systems, and Applications*, volume 5031, pages 102–110, 2003. (Cited on page 150.)

S. Mukherjee, P. Tamayo, S. Rogers, R. Rifkin, A. Engle, C. Campbell, T. R. Golub, and J. P. Mesirov. Estimating dataset size requirements for classifying DNA microarray data. *Journal of Computational Biology*, 10(2):119–142, 2003. (Cited on pages 184, 185, 186, 187, and 189.)

National Institute for Occupational Safety and Health (NIOSH). *Approaches to safe nanotechnology: Managing the health and safety concerns associated with engineered nanomaterials*. 2009. (Cited on page 3.)

D. E. Newland. Harmonic wavelet analysis. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 443(1917): 203–225, 1993. (Cited on page 111.)

A. Ng. CS229 lecture notes, part V, support vector machines, 2014. URL `http://cs229.stanford.edu/notes/cs229-notes3.pdf`. (Cited on page 145.)

G. Oberdörster, V. Stone, and K. Donaldson. Toxicology of nanoparticles: A historical perspective. *Nanotoxicology*, 1(1):2–25, 2007. (Cited on page 1.)

V. P. Oleshko, V. V. Kindratenko, R. H. Gijbels, P. J. M. Van Espen, and W. A. Jacob. Study of quasi-fractal many-particle-systems and percolation networks by zero-loss spectroscopic imaging, electron energy-loss spectroscopy and digital image analysis. *Mikrochimica Acta Supplement*, 13:443–451, 1996. (Cited on pages 47, 48, 49, 50, 53, 60, 107, and 109.)

S. Olsen. Estimation of noise in images: An evaluation. *CVGIP: Graphical Models and Image Processing*, 55(4):319–323, 1993. (Cited on page 64.)

J. D. Orford and W. B. Whalley. The use of the fractal dimension to quantify the morphology of irregular-shaped particles. *Sedimentology*, 30(5):655–668, 1983. (Cited on pages 48, 53, and 109.)

Organisation for Economic Co-operation and Development (OECD). *List of manufactured nanomaterials and list of endpoints for phase one of the sponsorship programme for the testing of manufactured nanomaterials: Revision*. 2010. (Cited on pages 4 and 44.)

T. Oster. *Erkennung von Nanofaseragglomeraten in REM-Bildern auf Basis von Texturinformationen*. Bachelor thesis, Otto-von-Guericke University of Magdeburg, 2010. URL `http://wwwisg.cs.uni-magdeburg.`

de/bv/?page_id=124. (Cited on pages 48, 49, 50, 60, 62, 85, 108, 111, 113, and 134.)

A. D. Ostrowski, T. Martin, J. Conti, I. Hurt, and B. Herr Harthorn. Nanotoxicology: Characterizing the scientific literature, 2000-2007. *Journal of Nanoparticle Research*, 11(2):251–257, 2009. (Cited on page 2.)

N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man and Cybernetics*, 9(1):62–66, 1979. (Cited on page 61.)

K. R. Parker. Why an electrostatic precipitator? In K. R. Parker, editor, *Applied electrostatic precipitation*, chapter 1, pages 1–9. Blackie Academic & Professional, first edition, 1997. (Cited on page 32.)

Z. Peng and T. Kirk. Wear particle classification in a fuzzy grey system. *Wear*, 225-229:1238–1247, 1999. (Cited on pages 48, 50, 107, 108, and 109.)

T. M. Peters, S. Elzey, R. Johnson, H. Park, V. H. Grassian, T. Maher, and P. O'Shaughnessy. Airborne monitoring to distinguish engineered nanomaterials from incidental particles for environmental health and safety. *Journal of Occupational and Environmental Hygiene*, 6(2):73–81, 2009. (Cited on page 9.)

J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in kernel methods: Support vector machines*, chapter 12. MIT Press, Cambridge, 1998. (Cited on page 146.)

S. Plitzko, C. Thim, and V. Bachmann. Zweite Fragebogenaktion zu Aspekten des Arbeitsschutzes bei der Herstellung und bei Tätigkeiten mit Nanomaterialien in Deutschland. *Gefahrstoffe - Reinhaltung der Luft*, 73(1-2):7–13, 2013. (Cited on page 14.)

P. Podsiadlo and G. W. Stachowiak. Development of advanced quantitative analysis methods for wear particle characterization and classification to aid tribological system diagnosis. *Tribology International*, 38(10):887–897, 2005. (Cited on page 53.)

P. Poelt, M. Schmied, I. Obernberger, T. Brunner, and J. Dahl. Automated analysis of submicron particles by computer-controlled scanning electron microscopy. *Scanning*, 24(2):92–100, 2002. (Cited on page 49.)

S. Raadnui. Wear particle analysis—utilization of quantitative computer image analysis: A review. *Tribology International*, 38(10):871–878, 2005. (Cited on pages 48, 108, 109, 110, 112, and 134.)

K. Rank, M. Lendl, and R. Unbehauen. Estimation of image noise variance. In *IEE Proceedings Vision Image and Signal Processing*, volume 146, pages 80–84, 1999. (Cited on page 64.)

M. Ranzato, P. Taylor, J. House, R. Flagan, Y. LeCun, and P. Perona. Automatic recognition of biological particles in microscopic images. *Pattern Recognition Letters*, 28(1):31–39, 2007. (Cited on pages 48, 50, and 58.)

M. Raymer, W. Punch, E. Goodman, L. Kuhn, and a.K. Jain. Dimensionality reduction using genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 4(2):164–171, 2000. (Cited on page 150.)

M. T. Reetz, M. Maase, T. Schilling, and B. Tesche. Computer image processing of transmission electron micrograph pictures as a fast and reliable tool to analyze the size of nanoparticles. *The Journal of Physical Chemistry*, 2000. (Cited on page 47.)

E. Ribeiro and M. Shah. Computer vision for nanoscale imaging. *Machine Vision and Applications*, 17(3):147–162, 2006. (Cited on pages 20, 37, 41, and 71.)

M. Rodriguez-Damian, E. Cernadas, A. Formella, and M. Fernandez-Delgado. Automatic detection and classification of grains of pollen based on shape and texture. *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, 36(4):531–542, 2006. (Cited on pages 48, 50, 58, 107, 108, 110, 111, 126, and 134.)

P. L. Rosin. Unimodal thresholding. *Pattern Recognition*, 34(11):2083–2096, 2001. (Cited on page 85.)

K. Savolainen, H. Alenius, H. Norppa, L. Pylkkänen, T. Tuomi, and G. Kasper. Risk assessment of engineered nanomaterials and nanotechnologies: A review. *Toxicology*, 269(2-3):92–104, 2010. (Cited on pages 2, 3, 10, and 13.)

M. Sezgin and B. Sankur. Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic Imaging*, 13(1):146–165, 2004. (Cited on page 86.)

L. G. Shapiro and G. C. Stockman. *Computer vision*. Pearson, 2001. (Cited on page 89.)

W. Siedlecki and J. Sklansky. A note on genetic algorithms for large-scale feature selection. *Pattern Recognition Letters*, 10(5):335–347, 1989. (Cited on page 150.)

S. N. Sivanandam and S. N. Deepa. *Introduction to genetic algorithms*. Springer Berlin Heidelberg, 2008. (Cited on page 150.)

J. E. Smith, M. A. Tahir, D. Sannen, and H. Van Brussel. Making early predictions of the accuracy of machine learning classifiers. In M. Sayed-Mouchaweh and E. Lughofer, editors, *Learning in non-stationary environments*, chapter 6, pages 125–151. Springer New York, 2012. (Cited on pages 185, 186, 187, and 189.)

K. Spurny, C. Boose, and D. Hochrainer. Zur Zerstäubung von Asbestfasern in einem Fließbett-Aerosolgenerator. *Staub, Reinhaltung der Luft*, 35(12):440–445, 1975. (Cited on page 44.)

G. P. Stachowiak, P. Podsiadlo, and G. W. Stachowiak. A comparison of texture feature extraction methods for machine condition monitoring and failure analysis. *Tribology Letters*, 20(2):133–147, 2005. (Cited on page 126.)

G. P. Stachowiak, G. W. Stachowiak, and P. Podsiadlo. Automated classification of wear particles based on their surface texture and shape features. *Tribology International*, 41(1):34–43, 2008. (Cited on pages 48, 50, 59, 107, 108, 109, 111, 112, 126, and 134.)

Y. Sun, A. K. Wong, and M. S. Kamel. Classification of imbalanced data: A review. *International Journal of Pattern Recognition and Artificial Intelligence*, 23(4):687–719, 2009. (Cited on pages 143, 144, 147, 161, and 167.)

J. H. Sung, J. H. Ji, J. D. Park, J. U. Yoon, D. S. Kim, K. S. Jeon, M. Y. Song, J. Jeong, B. S. Han, J. H. Han, Y. H. Chung, H. K. Chang, J. H. Lee, M. H. Cho, B. J. Kelman, and I. J. Yu. Subchronic inhalation toxicity of silver nanoparticles. *Toxicological Sciences: an Official Journal of the Society of Toxicology*, 108(2):452–461, 2009. (Cited on page 1.)

The Royal Society & The Royal Academy of Engineering. *Nanoscience and nanotechnologies: Opportunities and uncertainties*. London, UK, 2004. (Cited on pages 1 and 2.)

M. C. Thomas, R. J. Wiltshire, and A. T. Williams. The use of Fourier descriptors in the classification of particle shape. *Sedimentology*, 42 (4):635–645, 1995. (Cited on pages 48, 50, 110, and 134.)

C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '13*, pages 847–855, 2013. (Cited on pages 139 and 150.)

TSI. Hand-held condensation particle counter model 3007. Datasheet, 2012. URL `http://www.tsi.com/uploadedFiles/_Site_Root/Products/Literature/Spec_Sheets/3007_1930032.pdf`. (Cited on page 16.)

TSI. Nanometer aerosol sampler model 3089. Datasheet, 2013. URL `http://www.tsi.com/uploadedFiles/_Site_Root/Products/Literature/Spec_Sheets/3089.pdf`. (Cited on pages 32 and 44.)

V. N. Vapnik and A. Lerner. Pattern recognition using generalized portrait method. *Automation and Remote Control*, 24(6):774–780, 1963. (Cited on page 145.)

E. L. Vezey and J. J. Skvarla. Computerized feature analysis of exine sculpture patterns. *Review of Palaeobotany and Palynology*, 64(1-4): 187–196, 1990. (Cited on page 47.)

S. Wagner, S. Münzer, P. Behrens, T. Scheper, D. Bahnemann, and C. Kasper. Cytotoxicity of titanium and silicon dioxide nanoparticles. *Journal of Physics: Conference Series*, 170(1):012022, 2009. (Cited on page 1.)

G. I. Webb and P. Conilione. Estimating bias and variance from data. Technical report, Monash University, Melbourne, 2003. URL `http://i.giwebb.com/wp-content/papercite-data/pdf/WebbConilione04.pdf`. (Cited on pages 185, 187, and 191.)

D. Wienke, Y. Xie, and P. K. Hopke. Classification of airborne particles by analytical scanning electron microscopy imaging and a modified Kohonen neural network (3MAP). *Analytica Chimica Acta*, 310(1):1–14, 1995. (Cited on pages 48, 50, 107, and 134.)

K. Xu, A. Luxmoore, L. Jones, and F. Deravi. Integration of neural networks and expert systems for microscopic wear particle analysis. *Knowledge-Based Systems*, 11(3-4):213–227, 1998. (Cited on pages 48, 50, 107, 108, 110, and 134.)

J. Yang and V. Honavar. Feature subset selection using a genetic algorithm. In *Feature Extraction, Construction and Selection*, pages 117–136. Springer US, Boston, MA, 1998. (Cited on page 150.)

Z. Q. Yu, P. S. Chow, and R. B. Tan. Quantification of particle morphology by boundary Fourier transform and generic Fourier transform. *Chemical Engineering Science*, 62(14):3777–3786, 2007. (Cited on pages 48, 50, 58, 110, and 112.)

G. W. Zack, W. E. Rogers, and S. A. Latt. Automatic measurement of sister chromatid exchange frequency. *Journal of Histochemistry & Cytochemistry*, 25(7):741–753, 1977. (Cited on pages 60, 85, 86, 87, 98, and 99.)

C. T. Zahn. Two-dimensional pattern description and recognition via curvaturepoints. Technical report, Stanford Linear Accelerator Center, Stanford University, Stanford, California, 1966. (Cited on page 117.)

C. T. Zahn and R. Z. Roskies. Fourier descriptors for plane closed curves. *IEEE Transactions on Computers*, C-21(3):269–281, 1972. (Cited on page 120.)

G. P. Zhang. Neural networks for classification: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 30(4):451–462, 2000. (Cited on page 143.)

B. Zheng, Y. H. Chang, X. H. Wang, W. F. Good, and D. Gur. Feature selection for computerized mass detection in digitized mammograms by using a genetic algorithm. *Academic Radiology*, 6(6): 327–332, 1999. (Cited on page 150.)

EHRENERKLÄRUNG

Ich versichere hiermit, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; verwendete fremde und eigene Quellen sind als solche kenntlich gemacht. Insbesondere habe ich nicht die Hilfe eines kommerziellen Promotionsberaters in Anspruch genommen. Dritte haben von mir weder unmittelbar noch mittelbar geldwerte Leistungen für Arbeiten erhalten, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen. Ich habe insbesondere nicht wissentlich:

- Ergebnisse erfunden oder widersprüchliche Ergebnisse verschwiegen,
- statistische Verfahren absichtlich missbraucht, um Daten in ungerechtfertigter Weise zu interpretieren,
- fremde Ergebnisse oder Veröffentlichungen plagiiert,
- fremde Forschungsergebnisse verzerrt wiedergegeben.

Mir ist bekannt, dass Verstöße gegen das Urheberrecht Unterlassungs- und Schadensersatzansprüche des Urhebers sowie eine strafrechtliche Ahndung durch die Strafverfolgungsbehörden begründen kann. Die Arbeit wurde bisher weder im Inland noch im Ausland in gleicher oder ähnlicher Form als Dissertation eingereicht und ist als Ganzes auch noch nicht veröffentlicht.

Magdeburg, den 07.09.2017

Stephen Kockentiedt