Bayesian object classification in nanoimages

by

Andries Stefan Haywood

Submitted in partial fulfillment of the requirements for the degree

Magister Scientiae

In the Department of Statistics

In the Faculty of Natural and Agricultural Sciences

University of Pretoria

September 2017

I, *Andries Stefan Haywood,* declare that this mini-dissertation (100 credits), which I hereby submit in partial fulfillment for the degree Magister Scientiae in Mathematical Statistics at the University of Pretoria, is my own work and has not previously been submitted by me for a degree at this or any other tertiary institution.

Signature: 

Date: September 2017

Summary

In this mini-dissertation the importance of having an automated object classification procedure for classifying nanoparticles in nanoscale images (or referred to as nanoimages in this mini-dissertation) is discussed, and a detailed overview of such a procedure, proposed by [70] is provided, with emphasis on applying the procedure to nanoimages of gold nanoparticles. In the process a simplified approach to classifying occluded objects when dealing with homogeneously shaped objects is introduced. Nanotechnology is a technology that deals with measurements obtained in nano-scale (one billionth of a metre), and for ease of reference these images will henceforth be referred to as nanoimages. The focus is restricted to nanoimages, obtained using a Transmission Electron Microscope (TEM). A common phenomenon that occurs during the image capturing is occlusion of objects in the image. This occlusion leads to some unwanted results during the image analysis phase, making the use of a more sophisticated classification algorithm necessary. An automated classification algorithm that successfully deals with occluded objects in nanoimages [70] is discussed and a detailed discussion on the implementation of this algorithm is provided. The techniques used in the algorithm involve a combination of several Bayesian techniques to classify the objects in the nanoimage. Markov Chain Monte Carlo (MCMC) sampling techniques are used to simulate the unknown posterior, with samplers ranging from the Metropolis-Hastings and Reversable Jumps MCMC samplers to Monte Carlo Metropolis Hastings samplers used in obtaining the simulated posterior. Since one of the main objectives of this investigation will be the processing of images, a discussion on the most widely used image processing techniques is provided, with specific focus on how these techniques are used to extract objects of interest from the image. An overview of nanotechnology and its applications is provided, along with a variability study for the capturing of nanoimages using TEM. The aim of the study is to introduce controlled variability in the sampling through imposing specific sampling conditions, in order to determine if imposing these conditions significantly affects the measurements obtained. This variability study, according to our knowledge, is the first performed at this level of detail, and provides very useful considerations when performing a nanoimage study.

Acknowledgements

I would like to thank Dr. Inger Fabris-Rotelli for her endless patience, advice and valuable insights throughout this immense journey of learning. Her knowledge and motivation made even the mightiest of tasks seem possible, not to mention her "If you haven't figured it out yet you simply aren't trying hard enough." email signature. Thank you to my co-supervisors Dr. Sonali Das (Advanced Mathematical Modelling, CSIR Modelling and Digital Science) and Dr. James Wesley-Smith (DST/CSIR National Centre for Nanostructured Materials, CSIR) for their valuable contributions during the completion of this mini-dissertation. Their passion for learning and transferring of knowledge can motivate anyone to try harder.

I would also like to thank the Department of Statistics at the University of Pretoria, the NRF (under CSUR grant 90315), and the CSIR for allowing me the opportunity to not only complete this mini-dissertation, but also to attend various conferences around the country.

A special thank you to Prof. Alex Konomi for the valuable feedback on several pivotal questions, as well as the time spent on taking my questions over Skype®.

Thank you to my family and friends for their support during the completion of this mini-dissertation. Your words of encouragement and support have meant more to me than you think.

To my partner, Gerhard, I cannot thank you enough for the love and support you have provided me. Without you I would not have been able to come this far. You are my rock and my daily inspiration.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Nanotechnology is a fast growing research field with its main applications in medical and material sciences. This technological field relates to the manipulation of matter at nanoscale (one billionth of a meter), and is an interdisciplinary research field with opportunities to develop smarter devices and more precise solutions. Image analysis in Nanotechnology has important applications, with potential to employ nanoparticles as biomarkers, sensors, and drug targeting agents [97], and enables the quantification of physical properties in nanoimages. Opportunities and risks associated with the development of nanoengineered products must be understood at nanoscale and bulk scale, from synthesis to implementation. Nanoparticles have increased surface to volume ratio compared to their bulk form, making them more reactive and useful in material manipulation studies [115].

In applications, size and size distribution of nanoparticles is of primary concern. However, particle occlusion is most often an unwanted phenomenon occurring during the image analysis. Occlusion occurs when the three-dimensional sampling material (in the case of Fig. 1.1 the sampling material is colloidal gold) is captured onto a two-

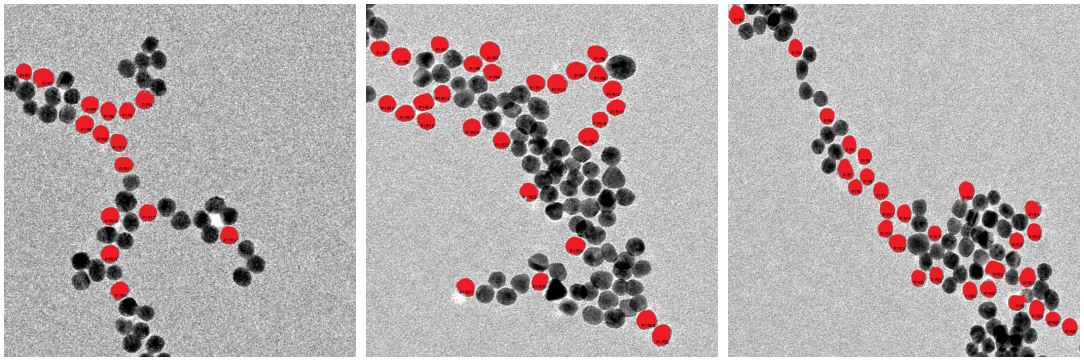


Figure 1.1: Gold nanoparticle images with occluded particles, obtained using TEM. Note: particles classified using ImagePro® software are highlighted in red.

dimensional image. When this is done, distinct particles appear as one (bigger or oddly shaped) particle due to the collapsing of the third dimension. Think of it as having two cars parked next to one another: when viewing the cars from the side, it may appear as one car, but when moving to the front or the back (the third dimension), the second car becomes clear. An example of occlusion during image analysis is shown in Figure 1.1, where it can be seen that the automated image analysis software, ImagePro®, neglects some nanoparticles (those not highlighted in red) in the Transmission Electron Microscopy (TEM) nanoimage. The nanoparticles that are not classified are often occluded, and the software identifies these particles as being 'too big' to be a nanoparticle, and excludes them from the classification process. This shortcoming can potentially lead to unwanted results for particle size measurements and subsequent particle size distributions, and analysis and inferences thereof.

To ensure consistent results with regards to sampled images, measurements obtained from a variety of images sampled under varying imaging conditions are compared. Analysing the variability in measurements from sampling schemes under varying conditions, such as different levels of magnification or time delays between measurements, is a crucial aspect in experimental design, and such analyses help in the identification of the sources of variability and possibly control for them. It has been shown that nanoparticles exhibit fractal properties [90], which enables the modeling of bulk scale[1] behaviour using the fractal approach. The fractal approach is a useful modeling technique when similar patterns or structures occur at progressively smaller or bigger scales, with the statistical information remaining in tact [90]. However, since this approach is heavily dependent on accurate nanoparticle measurements, it is crucial that the measurements obtained are consistent, so that accurate conclusions can be drawn from samples obtained under various sampling conditions. It is therefore important to understand the source of variability (if any) at the nanoscale, before attempting to draw conclusions at the bulk scale. Even though there is widespread interest towards nanoresearch in literature, there is a paucity of studies dealing with sampling scheme stability and accuracy of image measurements obtained. A preliminary analysis on variability in measurements made using TEM under varying imaging conditions commonly used is conducted, which, to the best of our knowledge, is the first of its kind being reported in this level of detail.

In this mini-dissertation a semi-automated Bayesian technique [70] to combat the problems faced with occlusion in images obtained using TEM is investigated. The technique uses several statistical techniques that are powerful in dealing with classification problems where the number of objects that need to be classified are unknown a priori,

[1]Bulk scale as referred to in this mini-dissertation refers to the end product stage.

and where the dimensionality between objects is potentially different from object to object. That is, the number of known and/or unknown parameters needed to define the object are potentially different between objects (for example, a triangle has more parameters than a circle, and as such needs to be modelled in different dimensions). Using a Bayesian framework, and due to the nature of the sampling scene and the assumptions made, the posterior distribution used for inference has an unknown normalising constant. The technique, as proposed by [70], can be seen as a two-stage sampler, where a Markov Chain Monte Carlo (MCMC) setting is used to sample the parameters from the pseudo posterior distribution, with an additional Monte Carlo Metropolis Hastings (MCMH) step to account for the unknown normalising constant. The MCMC steps use a variety of samplers to sample the parameters needed to characterise each object, with an additional sampler for the number of objects. The samplers used include Metropolis-Hastings-within-Gibbs, Independence and Reversible-Jumps MCMC samplers, each with its own set of complications and considerations. As a result, the techniques can be quite difficult to grasp at first sight. A version of this technique for dealing with homogeneously shaped gold nanoparticles is discussed, and some notes relating to the implementation steps for understanding of the algorithm when applied to a more general Bayesian object classification problem for nanoscale images is also provided. Though other parametrizations to this problem exists (see for example [62] and [119]), the specific aim of this mini-dissertation is to make the algorithm proposed by [70] easier to understand for a first time reader. A successful implementation of this algorithm in an image analysis software package, such as ImagePro®, may lead to great gains in better classification of occluded nanoparticles, and subsequently more accurate size measurements and size the three main sampling conditions distributions can be obtained.

As part of the investigation, an introductory discussion on the basics of image processing techniques is provided, along with a broad overview of the field of nanotechnology and its applications. The remainder of this mini-dissertation is set out as follows: Chapter 2 provides some necessary background to image processing, Chapter 3 gives an overview of nanotechnology and its applications. This chapter also gives the results of variability study, with Chapter 4 providing a discussion of the Bayesian object classification algorithm considered for this investigation. Chapter 5 provides some concluding remarks and further recommendations.

Chapter 2

Image Processing Basics

2.1 Introduction

The first techniques used for processing digital two-dimensional image data were mainly signal processing operators, generalized to two-dimensional data. Since then, the range of applications of image analysis techniques have extended to almost all engineering and scientific fields such as visual inspection, quality and security control, document imaging, remote sensing, microscopy, biology and medical image processing [13, 2, 111]. These widespread fields of application have led to numerous image analysis challenges, which is in turn the origin of many approaches that have been (and are still being) developed for analysing image data. A common technique used in image processing is mathematical morphology. Mathematical morphology is used for the analysis of spatial structures, and in image processing, is used to investigate the interaction between an image and a certain chosen structuring element (structuring elements are discussed in Section 2.5.1). An image, as referred to in this text, is defined as a two-dimensional function $f(x, y)$, where x and y are spatial coordinates, and the amplitude of f at any pair of coordinates (x, y) is called the intensity of the image at that point [46]. In practice, images are defined over a rectangular frame, referred to as the definition domain of the image or the image plane. However, when acquiring an image (a process referred to as sensing[1]), the input is on a continuous scale. It is therefore necessary to convert the continuous sensed data into digital (discrete) form. This is done through sampling and quantization, which is discussed in detail below.

2.2 Sampling and quantization

The input for a sensed image is continuous with respect to the x- and y- coordinates, as well as in amplitude. To create a digital image, the function is sampled in both

[1] More details on image sensing can be found in [46, 95, 2]

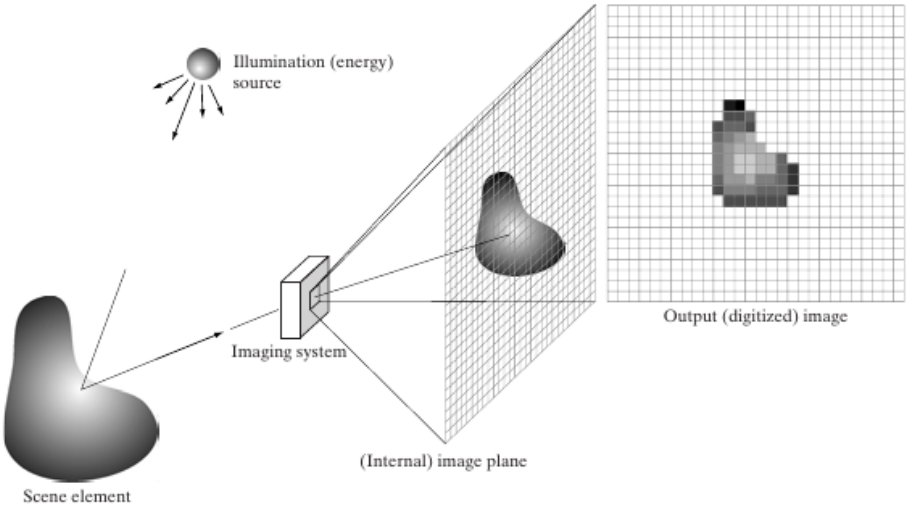


Figure 2.1: Image sensing example - obtaining a discretized image from a continuous image scene [46].

coordinates and in amplitude. Digitizing the coordinate values is called sampling and digitizing the amplitude values is called quantization [46]. Gonzalez [46] gives the following example of image sensing and corresponding digitization:

Figure 2.1 shows how an image is acquired in general, while Figure 2.2 shows the result of digitization of a sensed image[2]. Sampling and quantization allow us to represent digitized images as M × N matrices [123, 95]:

$$f = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,\mathrm{N}-1) \\ f(1,0) & f(1,1) & \cdots & f(1,\mathrm{N}-1) \\ \vdots & \vdots & & \vdots \\ f(\mathrm{M}-1,0) & f(\mathrm{M}-1,1) & \cdots & f(\mathrm{M}-1,\mathrm{N}-1) \end{bmatrix}. \tag{2.1}$$

The representation in (2.1) defines a digital image, where each matrix element is referred to as a picture element (or pixel). Note that the only requirement for M and N are that they must be nonnegative integers.

2.3 Background to morphology in discrete space

Even though the field of mathematical morphology was designed for Euclidean spaces, most image analysis technologies rely on the processing of discrete spatial data, since digital images are defined in discrete space. Morphology was therefore simplified from the Euclidean space, $\mathbb{R}^2$, to the discrete space, $\mathbb{Z}^2$, by sampling $\mathbb{R}^2$. Because prior knowledge about the sampled object is seldom available, a network of points evenly

[2]Both images were sourced from Gonzalez [46].

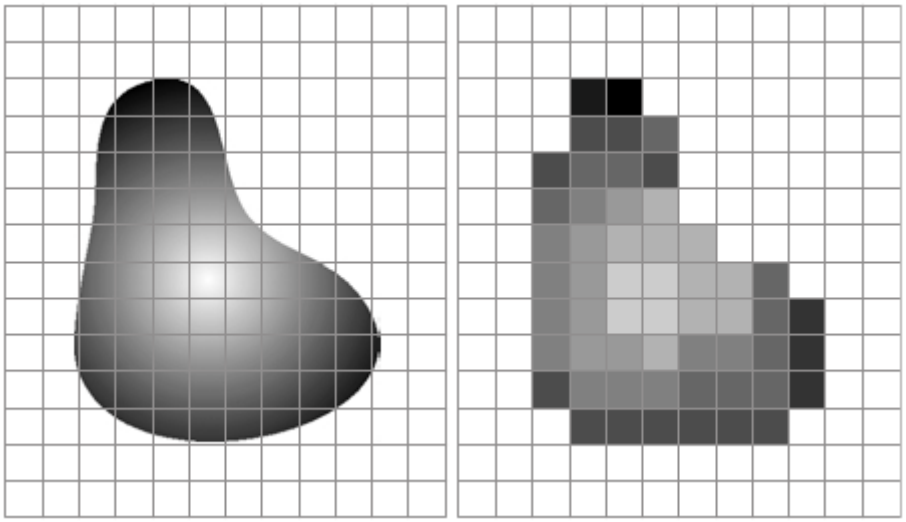


Figure 2.2: Image digitization, an example of how a digitization network can be set up to sample from the continuous image [46].

distributed at the nodes of either a triangular or square grid is considered as sampling scheme [111, 13]. In practice, majority of spatial data is sampled according to the square network. As described above, discrete images are defined by associating a numerical value with each point of the digitization network. Rather than a strict point measurement, each point value usually represents the mean value of the sensed signal averaged over the sampling window, and the sampling points can therefore be seen as the centers of convex polygons, called meshes. Square grids lead to square meshes, and in the context of digital image processing, these meshes are called pixels, as defined in (2.1).

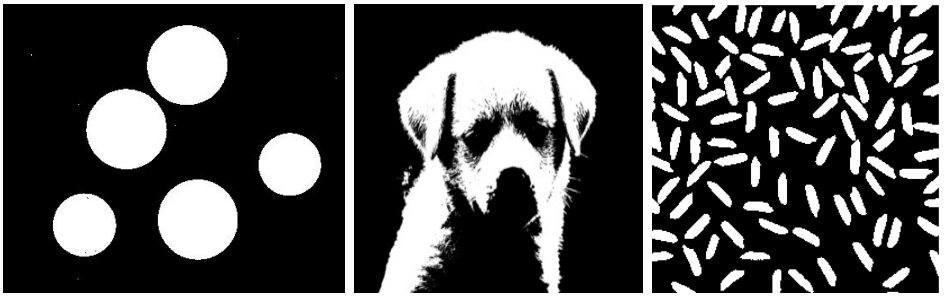
2.3.1 Discrete images and graphs

Binary and greyscale images are distinguished by the range of the values given to the points (pixels) of the digitization network. Both are mono-channel images because a single scalar value is stored for each pixel. Multi-channel images refer to those images where a vector of scalar values is associated with each pixel, such as red, green and blue (RGB) components of a colour image [46].

Binary images

The value of a pixel in a binary image is either 0 or 1, depending on whether the pixel belongs to the image foreground or background [111]. On a white support, foreground images are printed in black, and background in white. The opposite is valid for black support.

A binary image f is a mapping of a subset D_f of $\mathbb{Z}^2$ (called the definition domain



(a) Coins (b) A puppy (c) Rice grains

Figure 2.3: Examples of binary images on a black support, obtained through thresholding of greyscale images

of f) into the set $\{0, 1\}$ [111], i.e:

$$f \ : \ D_f \subset \mathbb{Z}^2 \longrightarrow \{0, 1\} \, .$$

Therefore, for each pixel $p \in D_f$ of the image definition domain, $f(p)$ is either 0 or 1. Note that this definition can be extended to an n-dimensional image: an image whose definition domain is a subset of the n-dimensional discrete space $\mathbb{Z}^n$. In this chapter, as well as in subsequent chapters, n will be restricted to 2. In a binary image the sets of black and white pixels are dual sets, that is, pixels that do not belong to the former necessarily belong to the other and visa versa [111]. That is, if FG denotes the foreground pixels and BG the background pixels, with the full image set S, then $FG = S \setminus BG$ or $FG = BG^c$ and simultaneously, $BG = S \setminus FG$ or $BG = FG^c$, where $A \setminus B$ denotes the relative complement of B in A (i.e. the set of all elements in A that are not in B [111]). Figure 2.3 shows examples of binary images of coins[3], a puppy[4] and rice grains[5] (obtained through thresholding - refer to section 2.4.2 for more information). Note that each pixel in the image is either black ($f(p) = 0$) or white ($f(p) = 1$).

Greyscale images

The range of values for the pixels of a greyscale image is not restricted to $\{0, 1\}$, but is extended to a larger, yet finite, set of non-negative integers. Denote this set by $\mathcal{N}_f$. In formal terms, a greyscale image f is a mapping of a subset D_f of $\mathbb{Z}^2$, (called the definition domain of f) into a bounded set of non-negative integers [111]. That is:

$$f : D_f \subset \mathbb{Z}^2 \longrightarrow \{0, 1, \ldots, t_{max}\} \, ,$$

[3]Original image sourced from http://inperc.com/ on 10 November 2014.
[4]Original image sourced from http://www.photomaskportal.com on 10 November 2014.
[5]Original image sourced from http://blogs.mathworks.com on 10 November 2014.

(a) Coins (b) A puppy (c) Rice grains

Figure 2.4: Examples of greyscale images, where each pixel in the image is a varying tone of grey.

where t_{max} is the maximum value of the data type used for storing the image[6]. In other words, for all pixels p of the image definition domain, the mapping $f(p)$ belongs to the set $\{0, 1, \ldots, t_{max}\}$. For greyscale images t_{max} is commonly set to 255, therefore 256 grey levels. Figure 2.4 shows examples of greyscale images of coins[7], a puppy[8] and rice grains[9]. Note that each pixel of the definition domain is a varying tone of grey, i.e. belongs to the set $\{0, 1, \ldots, 255\}$.

Multi-channel images

A multi-channel image consists of an array of mono-channel images (binary or greyscale) defined over a common definition domain, and hence a vector of scalar values is associated with each pixel in the domain. The number of available channels determines the dimensionality. In other words, if $\mathbf{f}$ denotes a multi-channel image with m channels, then the values of each pixel p in the definition domain of $\mathbf{f}$ define a m-dimensional vector:

$$\mathbf{f}(p) = (f_1(p), f_2(p), \ldots, f_m(p)).$$

A direct set representation of a multi-channel image cannot be obtained, and consequently each channel f_i of a multi-channel image is (usually) processed as a single greyscale image (that is, independently of the other channels). Many types of multi-channel images exist depending on the information that is collected for each image pixel. For example, colour images are multi-channel images containing three channels, one for each primary colour in the RGB colour model.

This text will focus on mono-channel images, specifically on greyscale ($t_{max} = 255$,

[6] $t_{max}=2^K$-1 for values coded on K bits, that is, for an image with a 256 grey level, $K = 8$, or an 8-bit image

[7] Original image sourced from http://inperc.com/ on 10 November 2014.

[8] Original image sourced from http://www.photomaskportal.com on 10 November 2014.

[9] Original image sourced from http://blogs.mathworks.com on 10 November 2014.

Table 2.1: 1-D signal f

x	0	1	2	3	4	5	6	7	8	9	10	11
$f(x)$	0	1	1	2	3	3	2	4	2	1	1	0

with 256 levels) and binary images.

Graphs and sub-graphs

In morphology greyscale images are represented as a topographical relief, by linking each pixel in the image with a height (or elevation) proportional to the intensity of the pixel. In other words, the values of an image are seen as heights of a surface above the image plane. This morphological representation will enable the application of set transformations to greyscale images. That is, greyscale images are considered as sets through their graphs and sub-graphs.

The graph G of an image f is the set of points (x, t) such that the pixel x belongs to the image plane of f and $t = f(x)$ [111]:

$$G(f) = \left\{(x, t) \in \mathbb{Z}^2 \times \mathcal{N}_f \mid t = f(x)\right\}.$$

The sub-graph SG of an image f is the set of points of $\mathbb{Z}^2 \times \mathcal{N}_f$ lying below the graph of the image and over the image plane:

$$SG(f) = \left\{(x, t) \in \mathbb{Z}^2 \times \mathcal{N}_f | \; 0 \leq t \leq f(x)\right\}.$$

Figure 2.5 shows the graph and sub-graph of the 1D image signal given in Table 2.1. This concept is similarly extended to two-dimensional graphics.

2.4 Image operations

2.4.1 Discrete geometry

Graphs

When computing operations involving some neighbourhood relationships, the use of a digitization network is not enough. By using *graphs*, one can define neighbourhood relationships between points of the digitization network in the following manner: a graph $\mathcal{G}$ linked to a digitization network is a pair (V, E) of *vertices* V, and *edges* E [17, 27], where

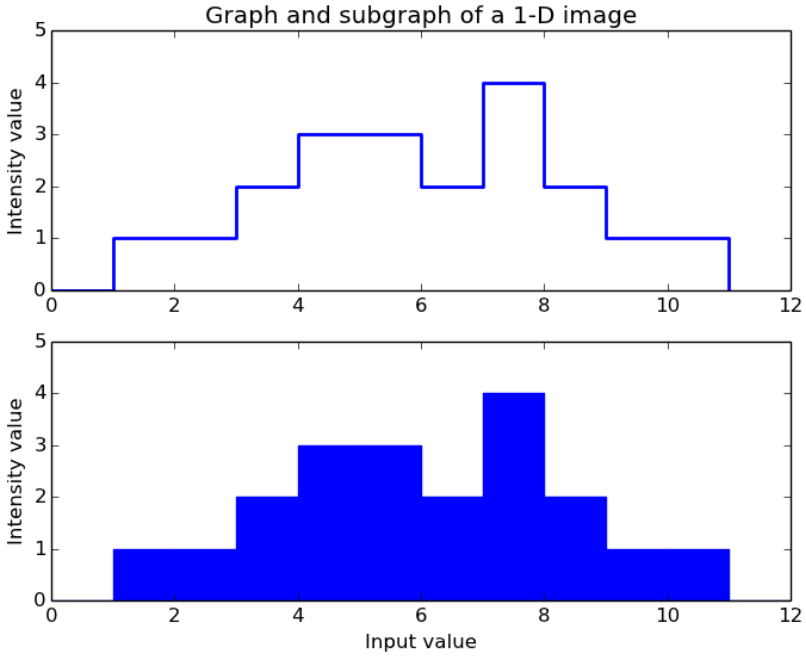


Figure 2.5: Graph and sub-graph of the 1D signal f given in Table 2.1

- $V = \{v_1, v_2, \ldots, v_n\}$ is a set of vertices representing the points of the digitization network,
- $E = \{e_1, e_2, \ldots, e_m\}$ is a group of unordered pairs (v_i, v_j) of vertices represented by edges or *arcs*.

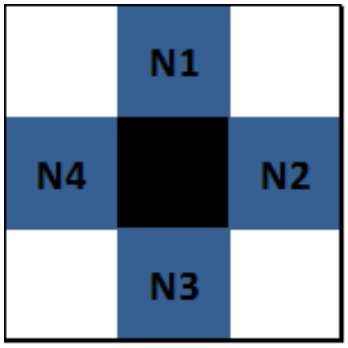
Further to this, if a graph does not contain any loop (i.e. (v_i, v_j) type edge) and if there exists one or less arcs linking any given pair of vertices, it is said to be a *simple* graph. A graph is *planar* if it can be drawn in the plane without intersecting any pair of edges [111]. The neighbours of a vertex v in a graph $\mathcal{G} = (V, E)$, are denoted by

$$\mathcal{N}_{\mathcal{G}}(v) = \{v' \in V \mid (v, v') \in E\} \cup \{v\},$$

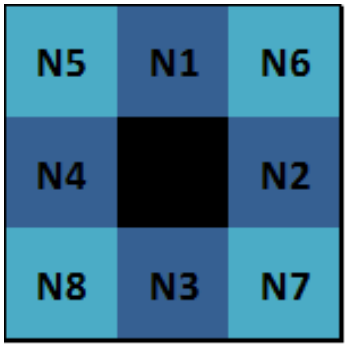
and the vertices v' are said to be $\mathcal{G}-adjacent$ to the vertex v [111]. Similarly the neighbours of a vertex v with a given value h are denoted by $\mathcal{N}^h_{\mathcal{G}}(v) = \{v' \in \mathcal{N}_{\mathcal{G}}(v) \mid f(v') = h\}$. The *foreground neighbourhood* of v, denoted by $\mathcal{N}^{\geq}_{\mathcal{G}}(v)$, is defined as:

$$\mathcal{N}^{\geq}_{\mathcal{G}}(v) = \{v' \in \mathcal{N}_{\mathcal{G}}(v) \mid f(v') \geq f(v)\},$$

i.e. all those neighbours of the vertex v whose function values are greater than or equal to that of v. Consequently, the *backgroundneighbourhood* of a vertex v is defined as the set of neighbours of v having strictly lower values than v, and is denoted by $N^{<}_{\mathcal{G}}(v)$. In addition to this we define the foreground FG of a pixel v of an image f



(a) Example of a 4−connected pixel (in black), with the 4 neighbours labelled N1, N2, N3 and N4 (in blue)



(b) Example of a 8−connected pixel (in black), with the 8 neighbours labelled N1 to N8 (in blue)

Figure 2.6: 4- and 8-connectivity in a discretized image.

as the threshold of f for intensity values greater than $f(v)$: $\mathrm{FG}(v) = T_{t>f(v)}(f)$. A threshold for intensity values lower than $f(v)$ defines the background BG of a pixel v: $\mathrm{BG}(v) = T_{t<f(v)}(f)$.

A sequence $v_0, v_1, ..., v_k$ of vertices of a graph $\mathcal{G}$ is said to be a *path* of length k if v_i and v_{i+1} are neighbours for all $i = 0, 1, ..., k-1$ and $(v_i, v_{i+1}) \in E$ [111]. The length of a path is the number of edges of the considered path. In general we denote by $\pi(v_i \rightsquigarrow v_j)$ a path in the graph $\mathcal{G}$, and $\pi(v_i \rightsquigarrow_h v_j)$ refers to a path whose vertices all have value h.

Grids and connectivity

The *graph connectivity* of a set is defined as follows: if a path can join each pair of points in a set, with all points being in the set under consideration, then the set is a connected set. Two pixels p and q of an image f are $\mathcal{G}^h$-connected if and only if there exists a path $\rightsquigarrow_h$ whose end points are p and q, denoted by $\pi(p \rightsquigarrow_h q)$. The corresponding connected components define the $\mathcal{G}^h$-connected components of pixels of a greyscale image, and is denoted by $\mathrm{CC}_{\mathcal{G}}^h$ (also referred to as flat zones) [111]. In a discretized image, neighbouring pixels are usually connected through either 4− or 8−connectivity [2]. Figure 2.6 shows an example of how (a) 4− and (b) 8−connectivity is defined in a discretized image. As can be seen from Figure 2.6, with 4−connectivity each pixel will have 4 neighbours, N1, N2, N3 and N4, and with 8−connectivity, each pixel has neighbours N1 to N8.

A transformation directly linked to the concept of connectivity is the connected component labelling of a binary image. It is formed by setting each pixel that belongs to a connected component equal to specific grey level value, with different values used for each connected component. The resulting image is referred to as a label image. There exists many algorithms for labelling binary images, see for example [111, 2]. In

(a) Original binary image f

(b) Connected component labelling of binary image f

Figure 2.7: Connected component labelling of a binary image.

Figure 2.7 an example of connected component labelling of an image of the earlier sample image containing three circles.

Discrete lines

Lines and line segments are fundamental geometric elements used by many neighbourhood image analysis operators. In discretized images a line or a line segment is different to lines in the Euclidean space, due to images being defined in discrete space and is therefore a set of pixels connected through 4− or 8−connectivity:

- *Connected discrete lines:* Suppose that S is a digital path, i.e. an 8-connected set of pixels all but two of which have exactly two 8-neighbours in S, with the exceptional two (the endpoints) having only one. It can be shown that a digital 8-connected path L is a digitization of a straight line segment if and only if it satisfies the *chord property*, that is, the line segment joining any two points of L lies everywhere within a distance of 1 of S [111].

- *Angular resolution:* In some applications, an image transformation that is invariant to image rotation is created by computing image transformations along all possible line orientations. In a discrete grid, the angular resolution of a discrete line segment depends on its length, and in a square grid only $2\lambda - 2$ orientations can be defined with a connected line segment of odd length equal to λ pixels, and whose middle and extreme pixels are matched by the Euclidean line of the same orientation [111]. An example of this concept is illustrated in Figure 2.8, where it can be seen that the number of lines that can be drawn in discrete space is limited by the size of the grid.

Figure 2.9 shows examples of a line defined in Euclidean space, and in discrete space through 4− and 8−connectivity.

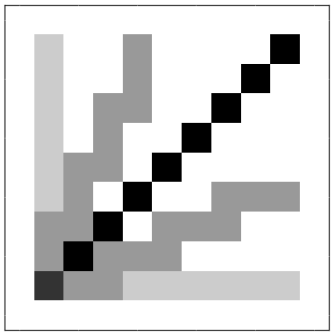
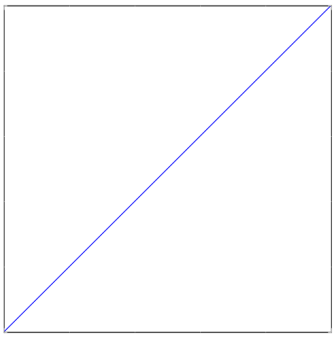
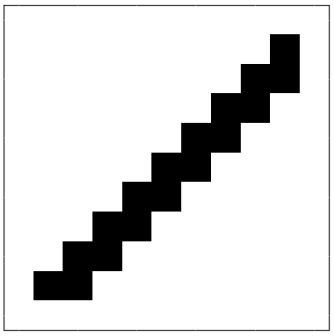


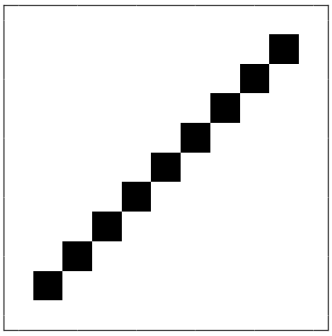
Figure 2.8: Example of the concept of angular resolution in discrete space.



(a) Line drawn in Euclidean space



(b) Line drawn in discrete space, using 4−connectivity



(c) Line drawn in discrete space, using 8−connectivity

Figure 2.9: Lines in Euclidean and discrete space.

2.4.2 Image to image transformations

In morphological image analysis the transformations considered are image to image transformations as the transformed image has the same definition domain as the input, and is a mapping of this definition domain onto the set of non-negative integers. The notation Ψ will be used to denote an image to image transformation. In general, an image to image transformation Ψ can be represented as $g = \Psi(f)$ [123], where

- f is the input image
- g is the output image
- Ψ is the operator whose output depends on the defined neighbours of each pixel (x, y), e.g. $4-$ or $8-$connectivity.

The *identity transformation*, denoted by id is defined as $\forall f$, $id[f(x, y)] = f(x, y)$. The set of images that are not altered by the given transformation is called the *domain of invariance* . For example, since the id transformation does not alter any image, its domain of invariance is the set of all images. When the transformed image is used as input for a second transformation, and so forth, the *iterated image transformation* is denoted by $\Psi^{(n)}$. That is $\Psi^{(n)} = \Psi^{(n-1)}\Psi$, the n^{th} iteration of an image transformation Ψ. Note that $\Psi^{(0)} = id$.

Point image transformations

In the case of point image transformations, the output values of a given pixel $p =(x, y)$ is a function of the input values of the input pixel, without taking into account the values of any other pixels. In other words, the neighbourhood size is set to 1×1 (a single pixel). In mathematical terms, a point image operator Ψ may be represented by $S = \Psi(r)$, where

$$S = g(x, y),$$
$$r = f(x, y).$$

Note that this notation is used to make it clear that a neighbourhood size of a single pixel is used. The basic point image transformations are the *additive image offset* and *multiplicative image scaling* operators [19]. The additive image offset operator may be defined as $S = \Psi_b^+(r) = r + b$, for some scalar b, that is $g(x, y) = \Psi_b[f(x, y)] = f(x, y) + b$. Note that the resulting output must still lie within the image definition domain, i.e. $|r + b| \leq t_{max}$, or alternatively clip those values falling outside of the image range. In other words, if $S < 0$, then $S = 0$, or if $S > t_{max}$ then $S = t_{max}$. The

multiplicative image scaling operator can be represented by:

$$S = \Psi^{\times}{}_b(r) = r \times b,$$

with the scalar b chosen such that the resulting output S lies within the image definition domain, that is $S \in [0, t_{max}]$. A simple way to handle overflows, where $S < 0$ or $S > t_{max}$, is to clip those values falling outside of the allowable range to the endpoint values. That is, if $S < 0$, then $S = 0$, and likewise if $S > t_{max}$ then $S = t_{max}$. An alternative to clipping is to allow any values for the output value S during calculation, and to re-scale the final image so that S lies within the image definition domain. Combining the additive image offset and multiplicative image scaling operators leads to *linear point operators* [19], for example, a linear point operator can be defined as $S = a \times r + b = a \times f + b$, for some scalars a and b. One of the widely used point image operators is the *threshold operator*: The *threshold operator* sets all pixels of the input image lying in a given range of values, equal to 1, with the remaining ones being set to 0 [123, 111]. More formally, the threshold operator T for given range $[t_i, t_j]$, sets all pixels (x, y) of the input image f with values in the interval $[t_i, t_j]$ to 1 and the others to 0:

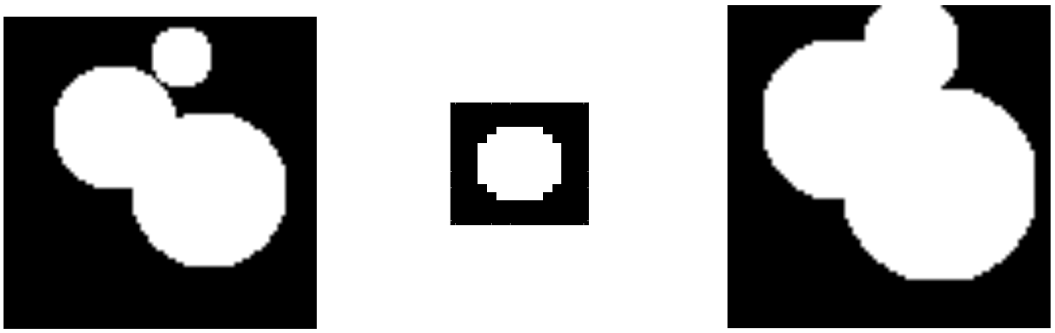
$$\Psi(p) = T_{[t_i,t_j]}[f\,(p)] = \begin{cases} 1 & \text{if } t_i \leq f\,(p) \leq t_j \\ 0 & \text{otherwise.} \end{cases}$$

In other words, thresholding can be seen as dividing pixels in a greyscale image into two classes, C_1 and C_2. For a grey level $f\,(p)$ all the pixels with grey level $t_i \leq f\,(p) \leq t_j$ will form a class C_1 and all the others will form a different class C_2. Figure 2.3 shows examples of binary images of coins, a puppy and rice grains, obtained through thresholding. The threshold operator used in this example is Otsu's threshold, where the threshold t_j is selected by maximising the between class variance [2]. Otsu's proposed algorithm is derived from the viewpoint of discriminant analysis, and is an automatic thresholding operator [92]. Note that Otsu's threshold only specifies a single threshold value, and not an interval.

Neighbourhood image transformations

In contrast to point image transformations, the output value of a neighbourhood image transformation at a given pixel is a function of the values of the pixels falling within a neighbouring region, centered around the input pixel. Some of the most fundamental neighbourhood image transformations include the spatial convolution and the cross-correlation.

1. The *spatial convolution*, denoted by *, involves an input image f together with a second image g, whose origin is usually located at the center of its definition



(a) Image f (b) Image g (c) Output image $f * g$

Figure 2.10: Spatial convolution of f by g: $f * g$.

domain $\mathcal{D}_g$, with the definition domain of g typically smaller than that of f. The output of the convolution at a given pixel p of f is then defined by the weighted sum of the pixels of f falling within $\mathcal{D}_g$ when the origin of $\mathcal{D}_g$ coincides with p, and the weights defined by the values of g [111]:

$$[f * g](p) = \sum_{b \in \mathcal{D}_g} [f(p - b)\, g(b)]. \tag{2.2}$$

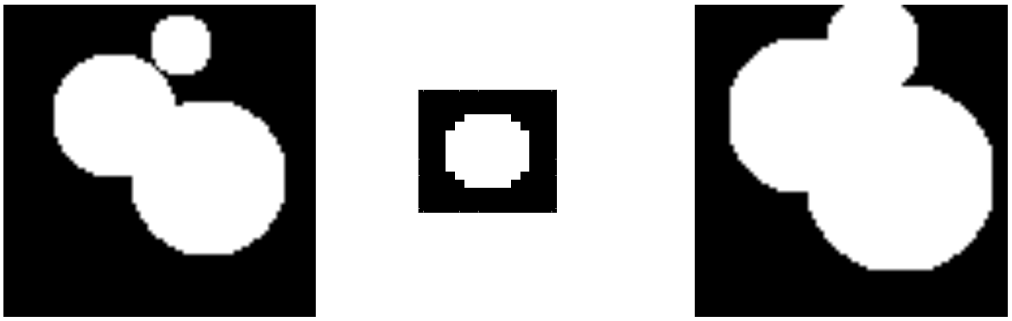
Figure 2.10 shows an example of a spatial convolution of image f, an image with three circles, of which two overlap, by image g, an image with a single circle. As can be seen from Figure 2.10, the result of the spatial convolution is an image with the weighted sum of the pixels taken whenever the origin of image g is placed over a given pixel p. Note that the effect of this is most visible on the border of the circles, where the weighted sum leads to an expansion of the border.

2. The *cross-correlation* C of two images f and g is defined as [111]

$$[C(f, g)](p) = \sum_{b \in \mathcal{D}_g} [f(p + b)\, g(b)].$$

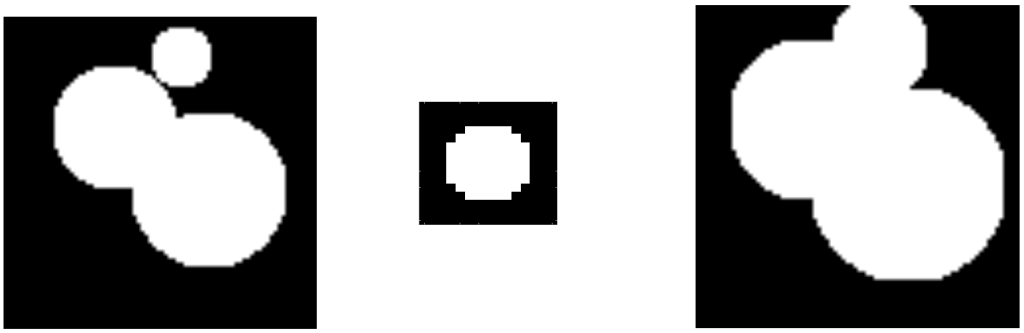
Thus, in the two-dimensional case, the cross-correlation of f with g comes down to the convolution of f with the 180° rotation of g, say g_{180}. Also, if the image g is symmetric in its origin ($g(p) = g(-p)$) then spatial convolution and cross-correlation are identical transformations. Figure 2.11 shows an example of the cross-correlation between image f, and image g, while Figure 2.12 shows the convolution of f with the 180° rotation of g, g_{180}. Note that since the image used for g in the operation is symmetric in its origin, the resulting cross-correlations are identical, with both also identical to the spatial convolution in Figure 2.10.

3. The *translation of an image* f by vector b is denoted by f_{-b} and is defined as:



(a) Image f (b) Image g (c) Output image $C(f, g)$

Figure 2.11: Cross correlation of f and g: $C(f, g)$.



(a) Image f (b) Image g_{180} (c) Output image $f * g_{180}$

Figure 2.12: Spatial convolution of f by g_{180}: $f * g_{180}$.

$f_{-b}(p) = f(p - b)$. The *translation of an image* can only be applied to images defined over an evenly distributed network of points.

2.4.3 Image filtering

In general, an image filter is defined as a neighbourhood image operator. When using this definition, image filters can perform a wide variety of tasks, including such tasks as noise reduction, edge detection, compensation for incorrect focusing, and motion blur. Mathematical morphology provides us with image filters for performing the first two of the above mentioned tasks, and in addition to this, morphological filters are especially suited to the extraction or suppression of image objects. Several filters exist in literature, but we restrict the discussion in this section to smoothing filters (or low-pass filters), specifically that of the median filter and Gaussian filter. For discussions on other filters, the reader is referred to [2, 19, 107, 13].

Median filter

The median filter replaces the value of a pixel $p = (x, y)$ by the median of the grey levels $f(p)$ in the neighbourhood of that pixel, with the original value of the pixel included in the computation of the median [46]. Mathematically, the median filter is defined as

$$\Psi_{med}(f) = median\{f(y_1), \ldots, f(y_n)\},$$

where $\mathcal{N} = \{y_1, \ldots, y_n\} \subseteq \mathbb{Z}^2$ is a suitably chosen neighbourhood [2, 19]. Median filters are effective in dealing with impulse noise and salt-and-pepper noise and as noted by [19], a median filter is a special case of a rank filter, Ψ_q, for which the q^{th} rank is chosen rather than the median.

Gaussian filter

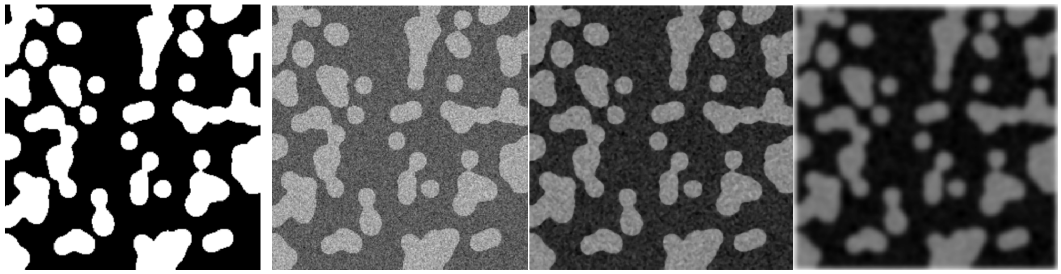
The Gaussian filter in image processing is a smoothing filter based on a convolution operator. Mathematically the filter is defined as [107, 95]

$$\Psi_{Gaussian}(x, y) = [f * g](x, y) = f * \left[\frac{1}{\sqrt{2\pi}\sigma} \exp\left\{-\frac{(x^2 + y^2)}{2\sigma^2}\right\}\right],$$

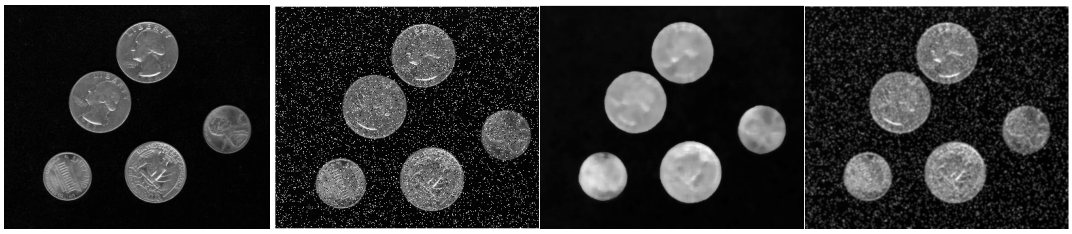
where σ is the standard deviation of the distribution, and the operator $*$ is used as defined in (2.2). In other words, Gaussian filtering is achieved by convolving the Gaussian distribution function, g, with the input image f. In image processing however, a discrete representation is required, since the Gaussian function above is nonzero for $x, y \in (-\infty, \infty)$ and therefore an infinitely large convolution kernel is required [107]. Practically it is reasonable to truncate the filter window since the function decays rapidly. There are several implementation strategies to obtain a discrete Gaussian kernel, for example, one strategy is to sample values from the continuous function at the center point of the pixel, whereas another strategy is to discretise the function itself [95, 107]. Once a suitable kernel is chosen, in this case through truncation of the Gaussian function, the filter is applied using a standard convolution method, as defined earlier.

Figure 2.13 shows examples of noisy images of random blobs[10] containing Gaussian noise, and coins containing salt and pepper noise respectively. The salt and pepper noise was added by thresholding random pixels in the image, using the Numpy [63] package in Python[11], and the Gaussian noise was added using the Numpy [63] and Scipy [64] packages in Python. As can be seen from the images, the median filter is effective in removing salt and pepper noise, while the Gaussian filter is effective in smoothing out Gaussian noise. Note however that since both filters are smoothing

[10] Image obtained using Python code obtained from http://scipy-lectures.github.io on 7 June 2015.
[11] Code used to add noise was obtained from http://stackoverflow.com on 12 June 2015.



(a) Original image (b) Noisy image (c) Median filter applied to image (d) Gaussian filter applied to image



(e) Original image (f) Noisy image (g) Median filter applied to image (h) Gaussian filter applied to image

Figure 2.13: Filtering noisy images: a) Random blobs, and b) Coins.

filters, the resulting images can sometimes be blurred (as a result of oversmoothing). Restoration of blurred images is not in the scope of this chapter, and the reader is referred to [2] for a detailed discussion on this topic.

2.4.4 Image segmentation

Image segmentation consists of dividing an image into different regions, where each region has certain properties. Once an image has been segmented into regions, neighboring relationships can be investigated by performing image measurements, and segmentation is therefore a crucial element in the quantitative interpretation of image data. Mathematically image segmentation can be described as follows [2]: A complete segmentation of an image $R = f(x, y)$ relates to identifying a finite set of regions $\{R_1, \ldots, R_N\}$ such that

1. $R = \cup_{i=1}^{N} R_i$, that is, the combination of all the regions must be the original image,

2. $R_i \cap R_j = \emptyset =; \forall\, i \neq j$, in other words no overlapping regions must exist in the segmentation,

3. $P(R_i) > 0; \forall\, i = 1, \ldots, N$, that is each region must consist of at least one pixel, and

4. $P(R_i \cup R_j) = 0; \forall\, i \neq j$, so that no pixel belongs to more than one region.

(a) Image f (b) Image g (c) Point-wise minimum $[f \wedge g]$ (d) Point-wise maximum $[f \vee g]$

Figure 2.14: Point-wise minimum and maximum of a sample image.

Several possibilities exist for image segmentation, but most segmentation algorithms are based on either pixel discontinuity or pixel similarity. As noted by [2], algorithms that focus on pixel discontinuity are often interested in line and/or edge detection, whereas algorithms that focus on pixel similarity often has the aim of grouping together pixels of homogeneous intensity into the same regions. Examples of image segmentation include thresholding (such as Otsu's threshold), edge detection (such as Canny's algorithm), clustering and the watershed transformation [2, 46]. The latter of these, the watershed transformation, will be discussed later in this chapter.

2.4.5 Image measurements

Image measurements aim at using numerical values to characterise the objects of an image. This includes measurements such as number of objects in an image, average size of an object, mean intensity of an object, compared to mean background intensity, shapes of the objects identified and overlapping of objects, to name a few.

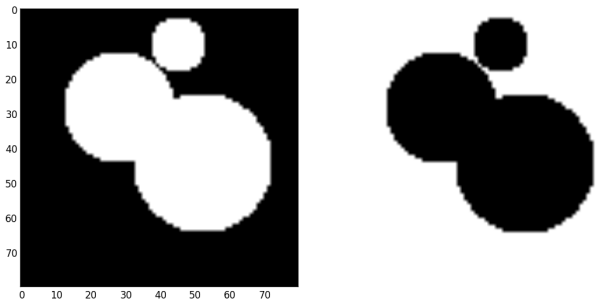
2.4.6 Set operators applied to images

The basic set operators, $\cup$ and $\cap$ becomes the *point-wise maximum* operator and *point-wise minimum* operator respectively, when working with greyscale images. The point-wise maximum $\vee$ and point-wise minimum $\wedge$ between two images f and g (having identical definition domains) are defined as follows for each point $p = (x, y)$ [111]:

$$[f \vee g](p) = \max[f(p), g(p)],$$

$$[f \wedge g](p) = \min[f(p), g(p)].$$

Figure 2.14 shows examples of the point-wise minimum and maximum operators applied to a sample image. Note that these operators can alternatively be represented in terms



(a) Original image f (b) Image complement $\complement[f]$

Figure 2.15: Image complement $\complement[f]$ for a binary image.

of unions and intersections of sub-graphs [111]:

$$\mathrm{SG}[f \vee g](p) = \mathrm{SG}[f(p)] \cup \mathrm{SG}[g(p)],$$

$$\mathrm{SG}[f \wedge g](p) = \mathrm{SG}[f(p)] \cap \mathrm{SG}[g(p)].$$

Starting from two arbitrary transformations Ψ_1 and Ψ_2, one can create a new transformation by applying the point-wise minimum or maximum to these two transformations:

$$[\Psi_1 \vee \Psi_2][f](p) = \Psi_1[f(p)] \vee \Psi_2[f(p)],$$

$$[\Psi_1 \wedge \Psi_2][f](p) = \Psi_1[f(p)] \wedge \Psi_2[f(p)].$$

Another set operator is *complementation*, where the complement $\complement$ of f, denoted by f^c, is defined for each pixel p as the maximum value of the data type used (i.e. t_{max}=255) minus the value of the image f at $p = (x, y)$ [111]:

$$\complement[f](p) = f^c(p) = t_{max} - f\,(p)\,.$$

Figure 2.15 shows an example of the complementation operator applied to a sample binary image. The *set difference* between two sets X and Y, denoted by $X \setminus Y$, is defined as the intersection between X and the complement of Y : $X \setminus Y = X \cap Y^c$, and applies to binary images only, an example of which is shown in Figure 2.16.

2.4.7 Ordering relations

To determine the ordering of two sets, the inclusion relation is used.

1. A *partially ordered set* S is a set which has a relation $\subseteq$ and for all sets $A, B, C \in S$ satisfies[111]:

   - $A \subseteq A$(reflexivity)

(a) Image f (b) Image g (c) Set difference $f \backslash g$

Figure 2.16: Set difference $f \backslash g$ of a binary image.

- $A \subseteq B$ and $B \subseteq A$ if and only if $A = B$ (anti-symmetry)
- if $A \subseteq B$ and $B \subseteq C$ then $A \subseteq C$ (transitivity).

2. A *totally ordered* set S is a partially ordered set which satisfies the following strengthened form of the anti-symmetry property:

   - For any two sets A and B in S, exactly one of $A \subset B, A = B$ or $A \supset B$ is true.

The partial ordering $\leq$ is defined on greyscale images as follows:

$$f \leq g \Leftrightarrow \forall\ p,\ f(p) \leq g(p)\,.$$

Equivalently, for all grey levels t, the cross-section of f at level t is included in the cross-section of g at level t,and so:

$$f \leq g \Leftrightarrow \forall\ p,\ f(p) \leq g(p) \Leftrightarrow \forall\ t, p,\ \mathrm{CS}_t[f](p) \subseteq \mathrm{CS}_t[g](p).$$

In other words, $f \leq g \Leftrightarrow \mathrm{SG}[f(p)] \subseteq \mathrm{SG}[g(p)]\ \forall p$.

2.4.8 Discrete distances and distance functions

The ability to measure how far points are separated in an image is often required in image analysis, and is the reason why the concept of distance is widely used during image analysis. A measurement d for a given space $\mathbb{E}$ is a function that associates with any two points p and q of $\mathbb{E}$ a non-negative real number, while satisfying the following conditions [111]:

1. $d(p,q) \geq 0$ and $d(p,q) = 0 \Leftrightarrow p = q$
2. $d(p,q) = d(q,p)$
3. $d(p,q) \leq d(p,r) + d(r,q)$.

The discrete distance $d_{\mathcal{G}}$ between two pixels p and q in the graph $\mathcal{G}$ is the shortest path $\pi_i(p \rightsquigarrow q)$ linking p to q:

$$d_{\mathcal{G}}(p,q) = \min_i \{\text{card}(\pi_i(p \rightsquigarrow q)) \mid \pi_i(p \rightsquigarrow q) \text{ path linking } p \text{ to } q \text{ in } \mathcal{G}\}.$$

If the underlying graph is 4−connected, the metric is known as the *city-block metric*, denoted by d_4. The city-block metric simplifies to $d_4((x_1,y_1),(x_2,y_2)) = \mid x_2 - x_1 \mid + \mid y_2 - y_1 \mid$, where (x_i, y_i) are the coordinates of pixel p_i. An 8−connected graph defines a *chessboard metric*, denoted by d_8, where d_8 simplifies to $d_8((x_1,y_1),(x_2,y_2)) = \max\{\mid x_2 - x_1 \mid, \mid y_2 - y_1 \mid\}$.

The *distance function* D on a binary image f associates each pixel p of the definition domain $\mathcal{D}_f$ of f with its distance to the nearest zero-valued pixel:

$$[D(f)](p) = \min\{d(p,q) \mid f(q) = 0\},$$

where $d(p,q) = d_4$ or $d(p,q) = d_8$ depending on whether 4−connectivity or 8−connectivity is used.

2.4.9 Image transformation properties

Knowledge of the properties of a transformation will allow us to predict its behavior and assist us in choosing appropriate transformations. Table 2.2 summarises some of the most widely used image transformation properties.

2.5 Erosion and Dilation

2.5.1 Structuring elements (SE)

A structuring element (SE) is a small set used to probe the image under study. Recall that the sub-graph of an n−dimensional image corresponds to an $n+1$−dimensional set, and therefore we can use $n+1$−dimensional SE's to investigate n−dimensional images. For simplicity, n−dimensional SE's are commonly used, that is to use subsets of the image definition domain. Operations using such so-called flat SE's are dimensionless, in other words, the shape of flat SE's does not depend on the scaling of the image grey levels.

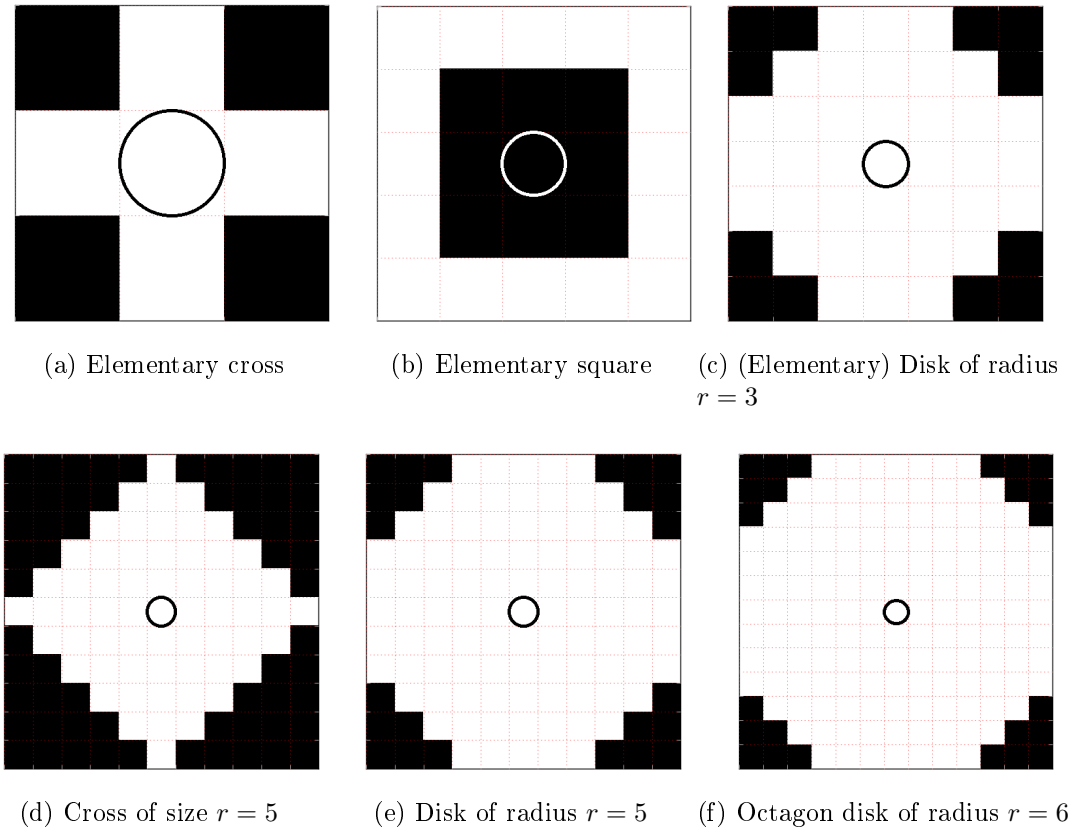
Fundamental morphological operators require that an origin is defined for each SE, as this allows the user to position an SE at a given pixel. An SE placed at a given point p means that the SE's origin matches with p. Most often flat and symmetric SE's are used, with the shape and size of each SE to be adapted according to the geometric properties of the image under consideration and the aim of the outcome. Note that

Table 2.2: Image transformation properties

Property	Definition	Formulation
Invariance to translation	A transformation Ψ is invariant to translations if it commutes with image translation	Ψ is invariant to translation $\Leftrightarrow$ $\forall f, \forall b,\ \Psi(f_b) = [\Psi(f)]_b$
Invariance to rotation	A transformation Ψ is invariant to rotation if it commutes with the image rotation operator Θ	Ψ is invariant to rotations $\Leftrightarrow \Psi\Theta = \Theta\Psi$
Linearity	A transformation Ψ is linear if the transformation of the linear combination of a series of input images is the same as the linear combination of the transformations of the input images	Ψ is linear $\Leftrightarrow \Psi\left(\sum_i a_i f_i\right) = \sum_i a_i \left[\Psi\left(f_i\right)\right]$
Invariance to threshold decomposition	If an image to image transformation Ψ can be expressed as the sum of the transformations of the cross-sections, it is said to be invariant to threshold decomposition. Image transformations that are invariant to threshold decomposition are called *flat operators*.	Ψ is invariant to threshold decomposition $\Leftrightarrow \Psi = \sum_{t=1}^{t_{max}} \Psi \mathrm{CS}_t$
Local knowledge	Let $\mathcal{D}$ be the definition domain of an image, with $\mathcal{D}'$ a subset of $\mathcal{D}$. An image transformation Ψ satisfies the local knowledge property if there exists a subset $\mathcal{D}'$ such that when transforming f restricted to $\mathcal{D}$ and then restricting the result to $\mathcal{D}'$ is the same as applying the transformation to the image defined over the whole space and then restricting the result to $\mathcal{D}'$.	$\Psi\left(f \mid \mathcal{D}\right) \mid \mathcal{D}' = \Psi\left(f\right) \mid \mathcal{D}'$,where $f \mid \mathcal{D}$ means the image f with a definition domain restricted to $\mathcal{D}$.
Idempotence	A transformation Ψ is idempotent if applying the operator Ψ to an image f consecutively results in the same output as when Ψ is applied to f only once.	Ψ is idempotent $\Leftrightarrow \Psi\Psi = \Psi$
Increasingness	A transformation Ψ is increasing if it preserves the ordering relation between images	Ψ is increasing $\Leftrightarrow \forall\ f\ ,\ g$ $f \leq g \Rightarrow \Psi\left(f\right) \leq \Psi\left(g\right)$

Note: Table continued on next page.

Property	Definition	Formulation
Connected operator	A connected operator Ψ is defined as an operator coarsening the partition of any given input image, with the partition Π of an image being taken in the sense of that induced by its flat zones. In other words, any flat zone of the output image corresponds to a union of flat zones already existing in the input image.	Ψ is a connected operator $\Leftrightarrow \Pi(\mathrm{id}) \leq \Pi(\Psi) \Leftrightarrow \forall f, \forall q \in N_{\mathcal{G}}(p),$ $[\Psi(f)](p) \neq [\Psi(f)](q) \Rightarrow f(p) \neq f(q)$
Extensivity	A transformation Ψ is extensive if, for all images f, the transformed image is greater than or equal to the original image, that is, if Ψ is greater than or equal to the identity transformation id	Ψ is extensive $\Leftrightarrow \Psi \geq \mathrm{id}$
Anti-extensivity	A transformation Ψ is anti-extensive if, for all images f, the transformed image is less than or equal to the original image, that is, if Ψ is less than or equal the identity transformation id	Ψ is anti-extensive $\Leftrightarrow \Psi \leq \mathrm{id}$
Duality	Two transformations Ψ and Φ are dual with respect to complementation if applying Ψ to an image is equivalent to applying Φ to the complement of the image, and taking the complement of the result	Ψ and Φ are dual with respect to complementation $\mathsf{C} \Leftrightarrow \Psi = \mathsf{C}\Phi\mathsf{C}$
Self-duality	A transformation Ψ is self-dual with respect to complementation if its dual transformation with respect to the complement is Ψ itself.	Ψ is self-dual with respect to complementation $\mathsf{C} \Leftrightarrow \Psi = \mathsf{C}\Psi\mathsf{C}$
Complementary	Two operators are complementary if and only if applying the first to an image is equivalent to applying the second to the complement of this image.	Two complementary operators Ψ and Φ satisfy $\Phi = \Psi\mathsf{C}$



(a) Elementary cross (b) Elementary square (c) (Elementary) Disk of radius $r = 3$

(d) Cross of size $r = 5$ (e) Disk of radius $r = 5$ (f) Octagon disk of radius $r = 6$

Figure 2.17: Examples of some basic structuring elements, with origins indicated by circled pixels. Individual pixels are denoted by squares.

the size of the SE under consideration will have an influence on the operator applied to the image, for example, for a larger SE, more of the image will be eroded when the erosion operator is applied. Figure 2.17 shows examples of some basic SE's used in image processing, with varying SE sizes.

2.5.2 Erosion

When probing a set with a SE, the question must be asked if the SE fits the set (where the set under study represents either the objects of a binary image or the sub-graph of a greyscale image). The eroded set will be the locus of points where the answer to this question is true. The erosion of a set X by a SE B is denoted by $\varepsilon_B(X)$ and is defined as the locus of points p such that B is included in X when the SE's origin is placed at p:

$$\varepsilon_B(X) = \{p |\ B_p \subseteq X\}. \tag{2.3}$$

Note that (2.3) can be rewritten in terms of an intersection of set translations, with the translations (as defined in Section 2.4.2) being determined by the SE:

$$\varepsilon_B(X) = \bigcap_{b \in B} X_{-b}. \tag{2.4}$$

When this definition is extended to greyscale images, the erosion of an image f by structuring element B is denoted by $\varepsilon_B(f)$ and is defined as the minimum of all translations of f by the vector $-b$ of B:

$$\varepsilon_B(f) = \bigwedge_{b \in B} f_{-b}.$$

Hence, the eroded value at a given pixel p is the minimum value of the image in the window defined by the SE when its origin is placed at p:

$$\varepsilon_B[f(p)] = \min_{b \in B} f(p+b).$$

The Minkowski subtraction, denoted by $\ominus$, of a set B to a set X is the intersection of the translations of X by the vectors of B:

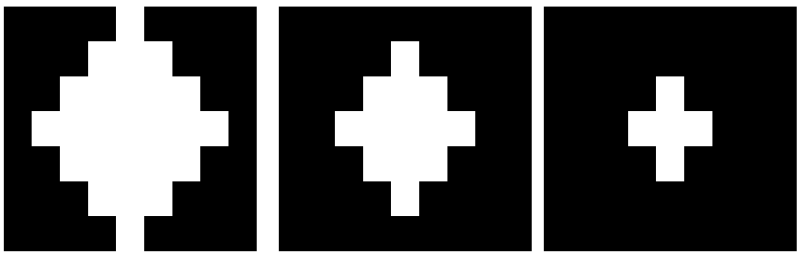
$$X \ominus B = \bigcap_{b \in B} X_b.$$

Note the similarity to the definition of the erosion given in (2.4). From the latter equation, we see that $\varepsilon_B(X) = X \ominus \check{B}$, with $\check{B}$ the complement set of B:

$$\begin{aligned} \varepsilon_B(X) &= \bigcap_{b \in B} X_{-b} \\ &= \bigcap_{-b \in B} X_b \\ &= \bigcap_{b \in \check{B}} X_b \\ &= X \ominus \check{B}. \end{aligned}$$

To illustrate the basic concept of the erosion operator, Figure 2.18 shows an example where a simple binary image is eroded, using some of the elementary structuring elements.

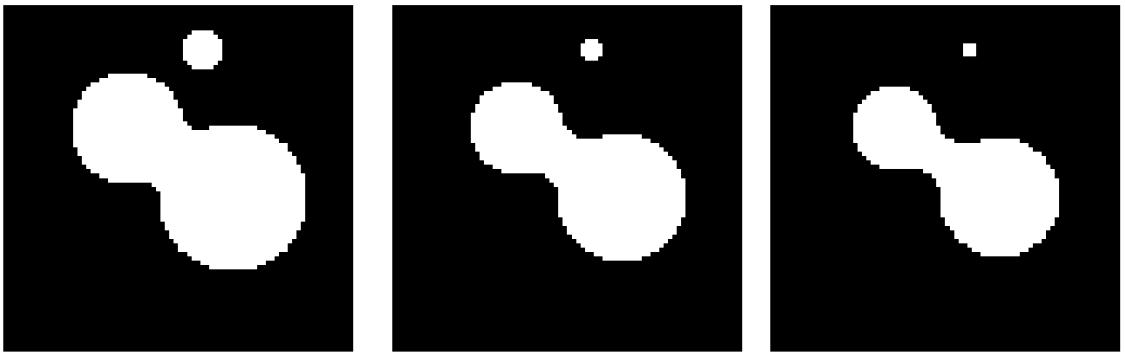
As can be seen from the example, the choice of structuring element is crucial when performing this operation, as the resulting image will differ depending on the choice of structuring element. This is also true for the choice of the size of the structuring element, as can be seen in Figure 2.19. In short, the erosion operator *shrinks* the objects, but *expands* the object's background, and is useful for removing smaller objects that should not be classified as an object in the image.



(a) Original image f to be eroded (b) Erosion of f by elementary cross (c) Erosion of f by elementary square

Figure 2.18: Erosion of a simple binary image.

(a) Original image f to be eroded (b) Erosion of f by elementary cross (c) Erosion of f by elementary square



(d) Erosion of f by elementary disk (e) Erosion of f by disk of radius $r = 5$ (f) Erosion of f by disk of radius $r = 6$

Figure 2.19: Erosion of a sample binary image using different structuring elements and different sizes of structuring element.

2.5.3 Dilation

The dilation is the dual operator of the erosion and is based on the question: "Does the SE hit the set?" The dilated set is the locus of points where the answer to this question is true. In set terms, the dilation of a set X by a SE B, denoted by $\delta_B(X)$, is defined as the locus of points p such that B hits X when its origin is placed at p:

$$\delta_B(X) = \{p \mid B_p \cap X \neq \emptyset\}. \tag{2.5}$$

Note that (2.5) can be rewritten in terms of a union of set translations, with the translations being defined by the SE:

$$\delta_B(X) = \bigcup_{b \in B} X_{-b}. \tag{2.6}$$

When the definition of set dilation is extended to greyscale images; the dilation of an image f by SE B, denoted by $\delta_B(f)$, is defined as the maximum of the translation of f by the vectors $-b$ of B:

$$\delta_B(f) = \bigvee_{b \in B} f_{-b}.$$

In other words, the dilated value of a given pixel p is the maximum value of the image in the window defined by the SE when its origin is placed at p:

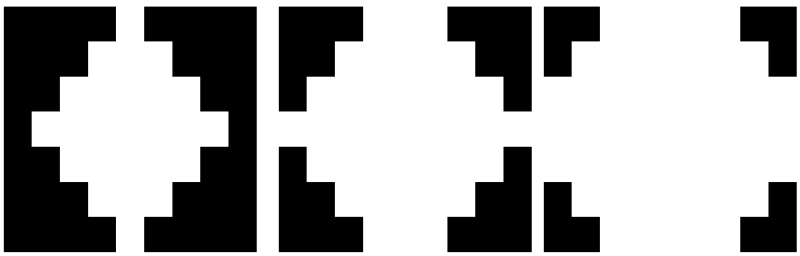
$$\delta_B[f(p)] = \max_{b \in B} f(p+b).$$

The Minkowski addition, denoted by $\oplus$, of a set B to a set X is the union of the translations of X by the vectors of B:

$$X \oplus B = \bigcup_{b \in B} X_b.$$

Note the similarity to the definition of the dilation (2.6). From the latter equation, we see that $\delta_B(X) = X \oplus \check{B}$:

$$\begin{aligned} \delta_B(X) &= \bigcup_{b \in B} X_{-b} \\ &= \bigcup_{-b \in B} X_b \\ &= \bigcup_{b \in \check{B}} X_b \\ &= X \oplus \check{B}. \end{aligned}$$



(a) Original image f to be dilated (b) Dilation of f by elementary cross (c) Dilated of f by elementary square

Figure 2.20: Dilation of a simple binary image.

To illustrate the basic concept of the erosion operator, Figure 2.20 shows an example where a simple binary image is eroded, using some of the elementary structuring elements. As with the erosion operator, the choice of structuring element influences the final image obtained from dilation, as seen in Figure 2.21.

In short, the dilation operator *expands* the objects, but *shrinks* the object's background.

2.5.4 Properties of erosion and dilation

Duality

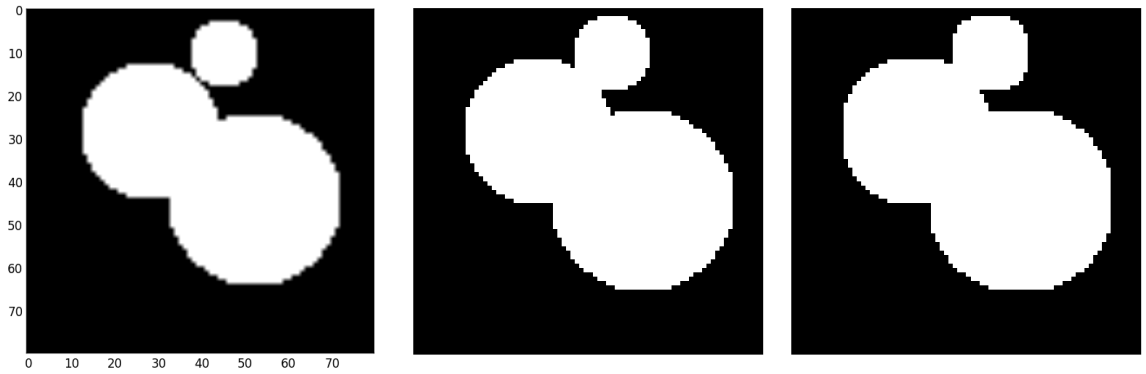
The dilation and the erosion are dual transformations with respect to complementation:

$$\varepsilon_B = \complement \delta_B \complement.$$

This means that any erosion of an image is equivalent to the dilation of the complemented image, and then taking the complement of the resulting image (and visa versa). The duality can be demonstrated as follows:

$$\begin{aligned} \delta_B\left(f^c\right) &= \forall_{b\in B}\left[t_{\max} - f_{-b}\right] \\ &= t_{\max} - \wedge_{b\in B}\left[f_{-b}\right] \\ &= t_{\max} - \varepsilon_B\left(f\right) \\ &= \left[\varepsilon_B\left(f\right)\right]^c. \end{aligned}$$

The result of this duality property shows that erosion and dilation do not transform the objects and their background in the same manner. Erosion *shrinks* the objects, while the opposite happens with the object's background (and through the duality the opposite is valid for dilation). It is also clear from this that erosion and dilation are not self-dual. Moreover, there is no inverse transformation for the dilation and the erosion.



(a) Original image f to be dilated (b) Dilation of f by elementary cross (c) Dilation of f by elementary square

(d) Dilated of f by elementary disk (e) Dilation of f by disk of radius $r = 5$ (f) Dilation of f by disk of radius $r = 6$

Figure 2.21: Dilation of a sample binary image using different structuring elements and different sizes of structuring element.

Increasingness

Erosion and dilation are invariant to translations, and also preserve the ordering relation on images, in other words, they are increasing transformations:

$$f \leq g \Rightarrow \begin{cases} \varepsilon_B(f) \leq \varepsilon_B(g) \\ \delta_B(f) \leq \delta_B(g) \end{cases}.$$

Distributivity

The dilation distributes the point-wise maximum operator $\vee$ and the erosion distributes the point-wise minimum operator $\wedge$:

$$\delta_B(\vee_i f_i) = \vee_i \delta_B(f_i),$$

$$\varepsilon_B(\wedge_i f_i) = \wedge_i \varepsilon_B(f_i).$$

Composition

The following equations concern the composition of dilation and erosion:

$$\delta_{B_2}\delta_{B_1} = \delta_{\left(\delta_{B_2}B_1\right)},$$

$$\varepsilon_{B_2}\varepsilon_{B_1} = \varepsilon_{\left(\varepsilon_{B_2}B_1\right)}.$$

A morphological operation with a large SE can therefore be decomposed into a series of operations each using a smaller SE. For example, eroding/dilating using an SE of size n is the same as eroding/dilating n times with the corresponding SE of size 1: $\delta_{nB} \coloneqq \delta_B^{(n)}$.

Ordering relations

The erosion with a SE is less than or equal to the dilation with the same SE: $\varepsilon_B \leq \delta_B$. Furthermore, if the structuring element B contains its origin, the following holds:

$$\varepsilon_B \leq id \leq \delta_B \Leftrightarrow B \text{ contains its origin.}$$

2.5.5 Morphological gradients

Basic morphological gradients

A common assumption in image analysis is that objects in an image have similar grey levels, that are different to the area surrounding the object. Therefore, object boundaries can be found where there are high grey level variations. Gradient operators use these variations to enhance the objects. When there is noise in the image, it should be filtered before applying a gradient operator to avoid enhancing the noise component. Morphological gradients use an SE to improve variations in pixel intensity in the neighbourhood of the SE. The erosion/dilation for example, returns for each pixel the minimum/maximum value of the image in the neighbourhood of the SE. By combining these elementary operators, variations in pixel intensity can be enhanced . The combinations currently used are:

1. the difference between dilation and erosion;
2. the difference between dilation and the original image
3. the difference between the original image and its erosion.

Only symmetric SE's that contain their origin are considered, as this assures that the arithmetic difference is always non-negative.

The basic morphological gradient is defined as the arithmetic difference between the dilation and erosion by the elementary SE B, of the considered grid. This morphological gradient is denoted by ρ:

$$\rho_B = \delta_B - \varepsilon_B. \tag{2.7}$$

The reader is referred to [111] for more on gradients. Figure 2.22 shows examples of gradients obtained from the erosion and dilation operations performed in Sections 2.5.2 and 2.5.3.

2.6 Opening and Closing

Since the erosion of an image removes all objects that cannot contain the SE, and reduces the size of the rest, an operator that will recover most (or at least some) of the objects removed or reduced by the erosion is required. This leads to the definition of the morphological opening. This operator dilates the resulting image after the erosion, using the same SE. In general, not all objects are recovered in full, as objects

(a) Original image f (b) Gradient using $\delta_B - \varepsilon_B$ (c) Gradient using $\delta_B - f$ (d) Gradient using $f - \varepsilon_B$

Figure 2.22: Morphological gradients.

that are removed by the erosion are not recovered at all. The dual operator for the morphological opening is the morphological closing. Important to note, is that openings filter the foreground regions, while closings filter the background regions. For example, to filter out noisy pixels with high intensity values, an opening should be used. If there is symmetric noise in the image, the closing of an opened image or the opening of a closed image can be performed.

2.6.1 Morphological opening

The opening γ of an image f by SE B, denoted by $\gamma_B(f)$, is defined as the erosion of f by B followed by the dilation with same SE B:

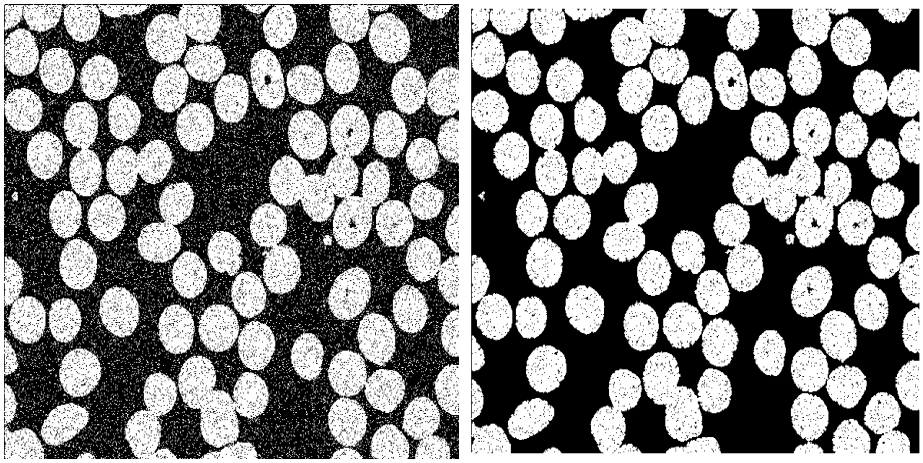
$$\gamma_B(f) = \delta_B[\varepsilon_B(f)], \tag{2.8}$$

i.e. $\gamma_B = \delta_B \varepsilon_B$. In (2.8) it is essential to consider the reflected SE for the dilation, if the chosen SE is not symmetric. In this text the focus will only be on symmetric SE's, and therefore the reflected SE is the same as the original SE. Although the opening is defined in terms of erosion and dilation in (2.8), it possess a geometric formulation in terms of SE fit using the question introduced for erosion: "Does the SE fit the set?" Each time the answer to this question is affirmative, the whole SE must be kept (for the erosion, it is the origin of the SE that is kept). Therefore the opened set is the union of all SE's fitting the set:

$$\gamma_B(X) = \cup_x \{B_x \mid B_x \subseteq X\}. \tag{2.9}$$

Figure 2.23 shows an example of a noisy image[12], with an opening performed using an elementary cross structuring element. Note that the noise in the image has been added by thresholding random pixels in the image, using the Numpy [63] package in

[12]Image obtained from http://cs.wellesley.edu on 10 November 2014.



(a) Noisy image f (b) Opening of f by elementary cross

Figure 2.23: Opening applied to a noisy image.

Python[13] As can be seen from Figure 2.23, the opening is successful in removing the background noise in the image.

2.6.2 Morphological closing

The closing of an image f by SE B, denoted $\phi_B(f)$, is defined as the dilation of f with a SE B followed by the erosion with the same SE B:

$$\phi_B(f) = \varepsilon_B[\delta_B(f)], \tag{2.10}$$

i.e. $\phi_B = \varepsilon_B \delta_B$. Using set formalism, we have the following question for defining a closing: "Does the SE fit the background of the set?" If yes, then all points of the SE belong to the complement of the closing of the set:

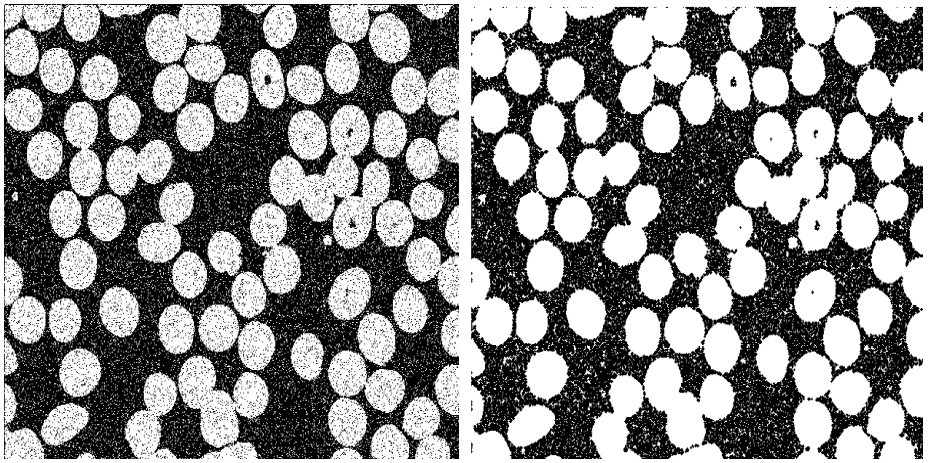
$$\phi_B(X) = [\cup_x \{B_x \mid B_x \subseteq X^c\}]^c. \tag{2.11}$$

An equivalent formulation, using the intersection of all translations of the complement of the SE B such that it contains X, is given by:

$$\phi_B(X) = \cap_x \{B_x^c \mid X \subseteq B_x^c\}. \tag{2.12}$$

Figure 2.24 shows an example of a noisy image, with an closing performed using an elementary cross structuring element. As can be seen from Figure 2.24, the closing is successful in removing the foreground noise in the image.

[13]Code used to add noise was obtained from http://stackoverflow.com on 12 June 2015.



(a) Noisy image f (b) Closing of f by elementary cross

Figure 2.24: Closing applied to a noisy image.

2.6.3 Properties of openings and closings

Duality

When an opening is applied to a set, it is examined from the inside as the SE has to fit the set. More precisely, if an object pixel cannot contain the SE, it is removed by the opening. A closing will do the opposite as it adds background pixels that cannot contain the SE. Hence, opening an image is the same as the closing of the complemented image and then complementing the resulting image. That is, openings and closings are dual transformations with respect to set complementation:

$$\gamma_B = \mathsf{C}\phi_B\mathsf{C}.$$

This can be demonstrated by expanding the right hand term of the above:

$$\begin{aligned}
\mathsf{C}\phi_B\mathsf{C} &= \mathsf{C}\varepsilon_B\delta_B\mathsf{C} \\
&= \mathsf{C}\varepsilon_B\mathsf{C}\varepsilon_B \\
&= \mathsf{CC}\delta_B\varepsilon_B \\
&= \delta_B\varepsilon_B \\
&= \gamma_B.
\end{aligned}$$

Ordering relations

Openings are anti-extensive transformations as they remove object pixels, and closings are extensive transformations as they add objet pixels. These transformations therefore

satisfy the following ordering relationship:

$$\gamma_B \leq id \leq \phi_B.$$

It follows that by simply performing the arithmetic difference between the input image and the opened image (or the arithmetic difference between the closing and the original image), we get an image of the pixels that have been modified by the opening (closing).

Increasingness and idempotence

Morphological openings γ and closings ϕ are both increasing transformations:

$$f \leq g \Rightarrow \begin{cases} \gamma_B(f) \leq \gamma_B(g) \\ \phi_B(f) \leq \phi_B(g) \end{cases}.$$

Openings and closings therefore maintain ordering relations between images, and consecutive applications of openings or closings will not further modify the image. To show this, note from (2.9) that the opening of a set X by an SE B can be written as the union of translations B_i of B for some i: $\gamma_B(X) = \cup_i B_i$. It follows that:

$$\begin{aligned} \gamma_B[\gamma_B(X)] &= \cup_x \{B_x \mid B_x \subseteq \cup_i B_i\} \\ &= \cup_i B_i \\ &= \gamma_B(X). \end{aligned}$$

Similarly for a closing we have that:

$$\begin{aligned} \phi[\phi_B(X)] &= \cap_x \left\{ B_x^{\complement} | B_x \supseteq X \right\} \\ &= \cap_x \left\{ B_x^{\complement} | B_x \supseteq \cap_i B_i \right\} \\ &= \cap_i B_i \\ &= \phi_B(X). \end{aligned}$$

The idempotence property is an important property of a filter as it ensures that the image will not be further modified when repeating the transformation.

2.6.4 Watershed Transformation

The watershed transformation is a powerful tool for segmentation in images. With this algorithm, the grey level image is considered as a topographic relief, with every pixel assigned to a catchment basin of a regional minima [2]. The image intensity is

considered as the altitude in the topographic relief. Every regional minima zone is then used to obtain the watershed lines that separate the regions. In other words, each pixel will flow along a descending path to a local minimum, and when the altitude of "water" gradually increases, two catchment basins will reach at some points, called watershed points [107]. A collection of watershed pixels on the contour is then considered as the watershed line. A common flaw of standard watershed algorithms is over-segmentation [2, 35]. Watershed from markers is an effective way to reduce over segmentation, in which case the watershed is performed as a flooding process where each marker is associated with a colour. The catchment basins are then filled uniformly with water, and when water of distinct colours are about to merge, a wall is put up to prevent these colours merging. The walled of regions are then the catchment basins associated with the markers [35]. The markers used in the algorithm are obtained by getting the regional maxima of a filtered image, with the input image usually being a gradient of the original image. In cases where the objects contain a natural "border" around the foreground objects.

Many algorithms exist for the watershed transformation, and the one discussed here is that of the minimum-cost path for the watershed from markers transform, as defined in [35]. The minimum-cost path between two pixels p and q is given by the minimal cost of all the paths connecting p and q:

$$C^{*}(p,q) = \min_{i} \left\{ C\left(\pi_{i}\left(p \rightsquigarrow q\right)\right) \right\},$$

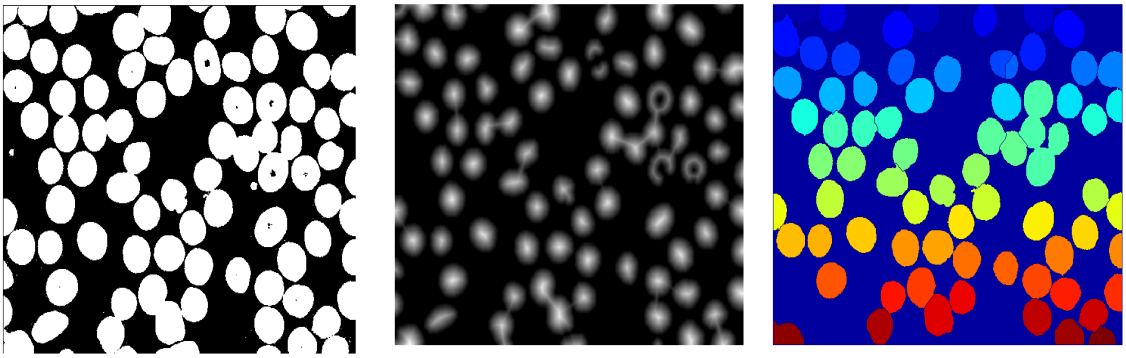
where $\pi_{i}(p \rightsquigarrow q)$ is a path linking p to q in $\mathcal{G}$. The cost of a connected path from p_1 to p_n is given by:

$$\begin{aligned} C\left(p_1,\ldots,p_n\right) &= \left[C^{1}\left(p_n\right), C^{2}\left(p_n\right)\right], \\ C^{1}\left(p_1\right) &= 0, \\ C^{1}\left(p_n\right) &= \max\left\{C^{1}\left(p_1\right), f\left(p_2\right), \ldots, f\left(p_n\right)\right\}, \text{ for } n > 1, \\ C^{2}\left(p_n\right) &= \max_{j=0}^{n-1}\left\{C^{1}\left(p_n\right), C^{1}\left(p_{n-j}\right)\right\}. \end{aligned}$$

where f is the input image or the gradient of the input image, and $C^{1}(p_n)$ is the maximum pixel value $f(p_i)$ and p_i in the path $\pi(p_1 \rightsquigarrow p_n)$. Next, the catchment basin of CB_k associated with the marker L_k is given by the pixels p with less than or equal path cost from this marker than any other marker:

$$CB_{k} = \left\{ p : C^{*}\left(L_{k}, p\right) \leq C^{*}\left(L_{j}, p\right) j \neq k \right\},$$

where the image is modelled as a graph and each pixel is a node. $C^{*}(L,p)$ is the



(a) Overlapping objects (b) Distances (c) Separated objects

Figure 2.25: Watershed transform applied to an image of red blood cells.

minimum-cost path from any pixel of region L to pixel p,

$$C^*(L,p) = \min\{C^*(l,p) : l \in L\}.$$

The algorithm proposed in [35], with input image f and output (and labelled image input) L is a form of a Hierarchical FIFO queue (HFQ), which has two operations:

- $inHFQ(p,v)$: insert pixel p with priority v
- $outHFQ$: remove the pixel wit the lowest priority. For pixels with equal priority, the FIFO policy is used.

The algorithm is as follows:

1. Initialisation: $for\ L(p) \neq 0 : inHFQ(p,0)$
2. Propagation:

   while HFQ is not empty:
   $p \leftarrow outHFQ$
   for each non-labelled q neighbour of p:
   $L(q) \leftarrow L(p)$
   $inHFQ(q, f(q))$

Some alternatives and improvements to this algorithm are discussed in [35]. In Figure 2.25 an example of a watershed is shown. In this example the original image is filtered using a median filter, after which Otsu's threshold is applied. The resulting image (a) is then used as input into the marker based watershed transform. The markers are generated as local maxima of the distance to the background. As can be seen from this example, there is over segmentation in the image due to the "holes" inside the foreground objects in the image, but addressing this is outside the scope of this work. From this example it can also be seen that objects that lie too far over the image border are disregarded in the final transformation.

2.7 Image pre-processing using mathematical morphology

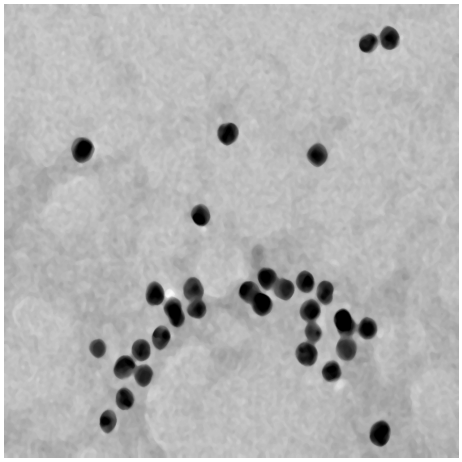
In this section the techniques discussed throughout this chapter are applied towards solving the greater goal of this work. Here mathematical morphology techniques are used to pre-process images in order to obtain input values for the Bayesian object classification algorithm that will be discussed in Chapter 4. The pre-processing will be done on images of gold nanoparticles obtained using TEM, and is discussed in more detail in Chapter 3. The structure of the pre-processing can be set out as follows:

1. Filter the original image to remove Gaussian and impulse noise, by using a Gaussian and/or median filter.
2. Threshold the filtered image to obtain a binary image, using Otsu's threshold.
3. Apply the opening operator to the image to remove unwanted objects in the image.
4. Since the image does not contain natural object borders, obtain the gradient.
5. Use the image gradient to obtain markers for the watershed transform.
6. Apply the watershed transform.
7. Obtain image measurements needed for the starting values Bayesian object classification algorithm, most important of which is the number of objects in the image.

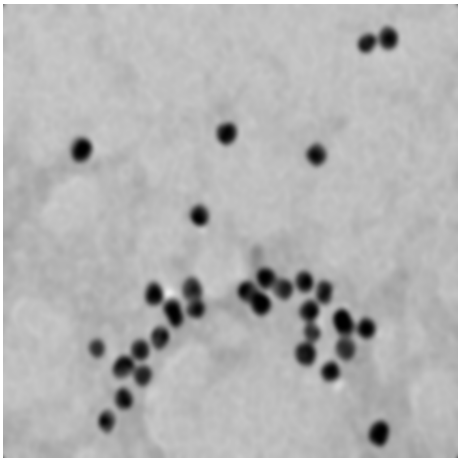
Figure 2.26 shows an example where pre-processing was done on an image of gold nanoparticles. For this example, the watershed transformation segmented the image into 31 objects. As can be seen from Figure 2.26, there is some over-segmentation present, and crucially, some of the occluded objects are classified as a single object. The latter will be addressed in Chapter 4.

2.8 Summary

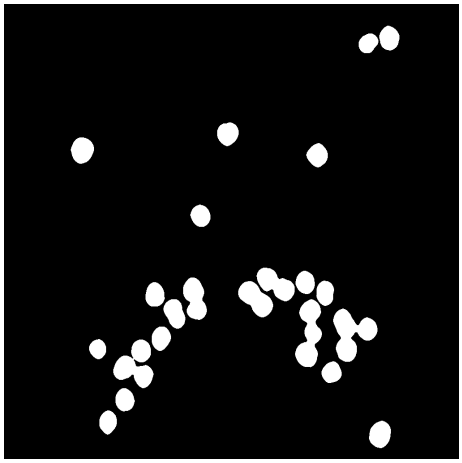
In this chapter a basic introduction to image processing and mathematical morphology is provided. The theory of some of the widely used morphology techniques is discussed and various examples are provided. A combination of these techniques are used to set up a process for the pre-processing of images of gold nanoparticles. The results of this pre-processing will be used Chapter 4 and will serve as input into the Bayesian object classification algorithm. It is also seen from the examples in this chapter that many



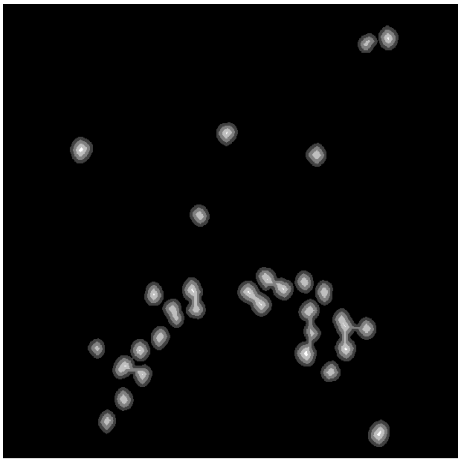
(a) Original image



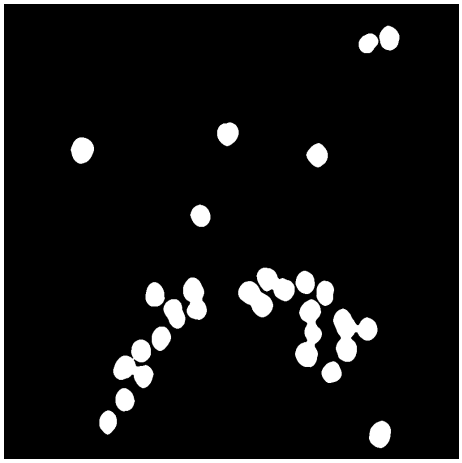
(b) Gaussian filter applied to image



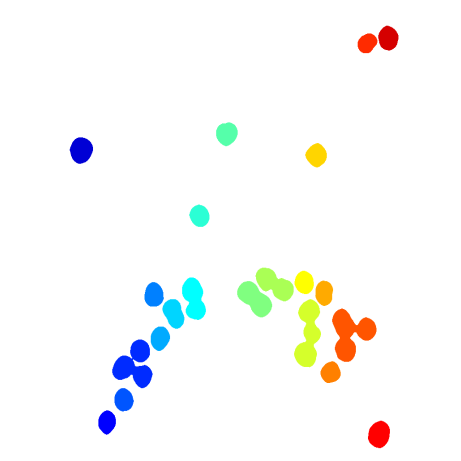
(c) Otsu's threshold and opening applied to image



(d) Gradient of filtered image



(e) Image with small (less than 5^{th} percentile) objects removed



(f) Watershed transformation applied to image, displayed on a white support.

Figure 2.26: Gold nanoparticle images: image pre-processing steps.

manual inputs are required for the morphological operators, and that the results of the pre-processing need some additional work to enable better classification. However, since this will be addressed in more detail in Chapter 4, no further work was done here on improving the pre-processing steps. The next chapter will go into more detail on nanotechnology and the challenges faced when obtaining images of nanoparticles.

Chapter 3

Fractal properties of nanoimage measurements

3.1 Background

In this chapter some of the concepts introduced in Chapter 2 are used in a variability study, where it is tested whether the measurements obtained from the sample images are consistent (discussed in detail later in this chapter). Having already covered some basic mathematical morphology concepts enables the use of these operators to pre-process the images obtained during sampling. Analysing the variability in measurements from sampling schemes under varying conditions is a crucial aspect in experimental design. Such analyses help in the identification of the sources of variability and possibly control for them. These conditions can refer to, amongst others, differences in sampling techniques, differences in instrumentation settings (such as magnification) and differences in sample preparation (such as waiting time). Experimental designs that are reliant on crucial user (manual) input are particularly vulnerable to variability, as different users may approach the problem in different manners. For example, one user may choose to use instrument defaults, where as another user may change the default settings based on experience or prior knowledge. In this study, an important aspect investigated is the difference in particle behavior at the nanoscale where the analysis will be performed, compared to the bulk scale where the results are implemented. Since the samples are dependent on user input, the main focus of this study will be to identify possible sources of variability that may influence results obtained. It has been shown that nanoparticles exhibit fractal properties [90], which enables the modeling of bulk scale behavior using the fractal approach. However, since this approach is heavily dependent on accurate nanoparticle measurements, it is crucial that the measurements obtained are consistent, so that accurate conclusions can be drawn from samples obtained under various sampling conditions. It is therefore important

to understand the source of variability (if any) at the nanoscale, before attempting to draw conclusions at the bulk scale.

Even though there is widespread interest towards nanoresearch in literature, there is a paucity of studies dealing with sampling scheme stability and accuracy of image measurements obtained. The focus of this chapter will be to conduct a preliminary analysis on variability in measurements made using TEM under varying imaging conditions commonly used. In the present study, it was decided to use a relatively homogeneous sample in order to minimise inherent sample variability and thus reveal other sources of variability at play during the analysis, which may likely influence the accuracy of the results obtained. The next section provides a brief overview of nanotechnology, followed by some basics of imaging techniques in nanotechnology, after which the variability analysis is discussed.

3.2 Nanotechnology: A brief overview

Nanotechnology, simply defined as the manipulation of matter at nanoscale (one billionth of a meter), is an emerging interdisciplinary field with vast opportunities for development and the design of smarter devices and more precise solutions to problems faced in some of the applications where this technology is currently used [127, 124, 82]. Applications exist in various fields, including biology [121], medicine [126, 6] and biotechnology [49, 55]. In the biomedical field nanotechnology advancements have led to numerous commercially available nanoproducts [29, 106, 56], with a number of nanobased products under clinical trial [126]. Research and development of nanobased products for the use as potential drug cues in cancer treatment [48, 58, 68], Alzheimer's treatment [105, 103], Parkinson's disease [80], dermatology [34] and several other medical fields [7, 38, 108, 109, 121], are also receiving great interest. Image analysis in nanotechnology has important applications, where there is potential to employ nanoparticles as biomarkers, sensors, and drug targeting agents [124, 120, 114, 59]. With the rapid advancements in nanoresearch [72], nanoengineered products are becoming more and more available to the consumer. It is therefore crucial that both opportunities and risks associated with the development of these products are better understood at both nanoscale as well as bulk scale, from synthesis to implementation.

An important aspect of nanoparticles is the increased surface to volume ratio of nanoparticles compared to their bulk form [115]. The increased surface area makes the particles more reactive and useful in material manipulation studies. Unintended side effects of nanotechnology can thus have a detrimental impact on the future development of the technology. These side effects, such as toxicity, are of great concern in nanoresearch as current knowledge gained from studying particles at nanoscale might present entirely new risks when used in their bulk form [91]. This has led to particular

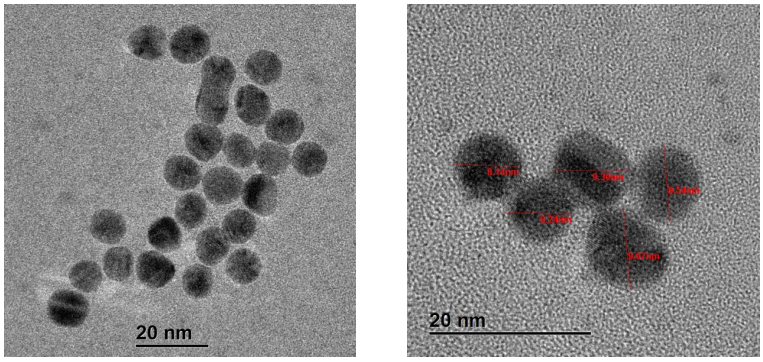
interest in the study of toxicity in nanoparticles, where metal nanoparticles have shown to be tremendously toxic [61].

Due to the ease of preparation and modification [26], the homogeneity of the size of its nanoparticles and unique optical properties [97], and various applications, the material chosen for the present study is commercially-produced colloidal gold. These nanoparticles are being extensively used in drug delivery systems [97, 44], cancer treatment [74, 51, 58], Alzheimer's disease detection and treatment [105, 20], biochemistry [37], tumour therapy [50], and biology [31, 120]. Studies on the toxicity of gold nanoparticles showed that gold nanoparticles can translocate between different organs [33, 73, 125]; and that biodistribution of gold nanoparticles occur after injection [74, 10, 112]. It has also been found that gold nanoparticles can induce oxidative stress [76], and aggravates seizure activity in mice [66]. There is size dependent particle distribution present in gold nanoparticles and the toxicity effect vary with particle size [33, 112, 25].

3.3 Imaging nanoparticles

Particle size, morphology, surface properties, and compositional information of nanoparticles play a vital role in deciding on the interactions and behavior of nanoparticles in their bulk form [102]. As a result, it is important to characterise nanoparticle properties such as diameter, particle size distribution, and surface area - all relevant for subsequent application implementation [52, 69]. Some of the most common methods used in characterisation include Analytical Ultracentrifugation (AUC), UV-Vis Spectroscopy, Scanning Electron Microscopy (SEM), Transmission Electron Microscopy (TEM), and Dynamic Light Scattering (DLS) [26, 30].

AUC is a high speed centrifuge machine that is useful in characterizing nanoparticle sedimentation rates, the occurrence of agglomeration and its resulting size as well as the absorbance and molecular weights of particles [18, 22, 128, 11, 28]. UIV-Vis spectroscopy is a technique that is useful in giving an overview of particle sizes, aggregation of particles and concentration of particles [16, 5, 52]. SEM is used to generate surface information from 'bulk' samples. The electron beam is scanned repeatedly in a raster pattern over a given area and reveals surface information from samples as well as their size, shape and presence of surface defects, amongst others [81, 53, 94, 21]. In cases where the SEM samples for imaging are mono-layered, there should be little to no difference in the results obtained using TEM. However, when this is not the case, SEM is not useful in providing details on individual particles. DLS is commonly used to measure the hydrodynamic diameter of nanoparticles [67, 18], and is useful in giving information on coating of nanoparticles that is often missed by TEM diameter measurements. An important additional advantage is that the number of particles



(a) Magnification of 250 000 times

(b) Magnification of 500 000 times

Figure 3.1: Gold nanoparticle images obtained at instrument magnifications of (a) 250 000, and (b) 500 000 times obtained using TEM. Note: manual measurements indicated on (b).

measured is orders of magnitude greater. However, TEM is perhaps the most used tool in nanoparticle characterisation. It uses energetic electrons to provide morphological information on the material samples by producing high quality two-dimensional images [30, 32, 1]. By scattering beam electrons by atoms in a sample, an image is formed in the TEM. Atoms that have large molecular weights will be more electron-dense, and will scatter electrons by larger angles. As these electrons do not reach the image plane, it creates dark regions in the image. Scattered and unscattered electrons then lead to an image that contains varying degrees of grey levels. Unlike SEM, TEM does not allow the user to determine if a feature is located at the top, middle or bottom of a sample as all this information with be imaged at the same time in two dimensions, and the best results are gained when samples do not exceed $100nm$ thickness. With thicker samples, multiple collisions between illuminating electrons and sample atoms heating damage along with lower image quality can be caused. Another fundamental requirement for TEM is for samples to be completely dry before imaging occurs, as 'wet' samples will degrade the high-vacuum environment inside the column, which can lead to contamination of the microscope.

Figure 3.1 shows examples of gold nanoparticles, visible as dark objects against a grey background, obtained using TEM at two different instrument magnifications. The TEM images used throughout this chapter was sourced by staff from the National Centre for Nanostructured Materials of the CSIR, South Africa. As can be seen from these images, there is some variation in size and shape amongst these, and that there is a trade-off between the ease of manual measurement at higher magnification at the expense of the number of particles visible in the field of view.

TEM provides images of the silhouettes of individual particles, although occluded particles may prove challenging. In contrast, other methods do not provide the re-

searcher with information from individual particles but only measurements of bulk samples instead.

3.4 A variability analysis of nanoscale image measurements

3.4.1 Sample preparation and considerations

The material chosen for this study is colloidal gold, due to the homogeneity of the size of gold nanoparticles [97]. Homogeneity in particle size is an important aspect in this experimental design, since differences in particle size will immediately indicate variability in the samples. Pre-prepared samples in an aqueous suspension are used for TEM, where particles have an average particle size of $10nm$. TEM samples are supported by a $3mm$ grid where a nm thin layer of carbon is laid to act as an electron-transparent support for the nanomaterials during imaging. The gold solution is dispersed on the film by placing a drop of gold colloid on the film, drawing excess solutions using a filter paper before drying, to ensure a mono-dispersed preparation is formed[1]. A JEOL-JEM 2100 TEM instrument (JEOL, Japan[2]) was used to obtain images, and ImagePro® software (Media Cybernetics, USA[3]) was used to analyse these images, as well as to calculate sample statistics from the images obtained. The data used throughout this study was sourced by staff from the National Centre for Nanostructured Materials of the CSIR, South Africa.

3.4.2 TEM operational procedures

Electromagnetic lenses used in electron microscopes suffer from a condition known as hysteresis, where the previous magnetic history of a lens affects the focal length of such lenses, and thus image magnification in the case of the objective lens. Accordingly, even under optimal operational procedures, size variation of a feature in an object of approximately 5% is not uncommon [122]. Various microscope conditions were tested in order to establish the influence – if any – of instrument operation upon the variability of outputs, and these were:

C1: *Three magnification levels* 500 000, 250 000 and 100 000.

C2: *Five minute time delay* The TEM was allowed a five minute stabilization time before the first measurement, and then another five minutes prior to each subsequent measurement in the three repetitions. This setting aimed to reduce

[1]Information obtained on 2014/06/18 at http://ls-ncnsm.csir.co.za/
[2]More information available at http://www.jeol.co.jp
[3]More details and software packages available at http://www.mediacy.com.

hysteresis by allowing microscope lens currents five minutes to stabilize before image recording in triplicate at each of the magnifications used.

C3: *Upwards magnification* This condition tested the presence of hysteresis by immediately recording three images in succession after the magnification was changed from (a) 100 000 to 250 000, and (b) 100 000 to 500 000. (The time delay between magnification changes to image recording did not exceed 30 seconds in each instance.)

C4: *Down to 30 000 and up* Magnification was reduced from 500 000 to 30 000 times and back up to 500 000 before recording three images immediately. The large change in magnification tested the presence of hysteresis, but in a more severe manner than C3 above.

3.4.3 Sampling scheme and statistics

To further reduce variability in this preliminary study, the same 5 particles (as shown in Fig. 3.1(b)) were measured under the various sampling conditions (described in C1-C4 above). The measurements reported (in nanometers) represent the longest diameter of each particle, with each measurement taken in the same place for each particle. Diameters were drawn manually on images using a mouse as opposed to using a thresholding function and automated measurement software.

Conditions C1-C4 were all chosen for the specific purpose of attempting to introduce controlled variability in the sampling, in order to determine if imposing these conditions significantly affects the measurements obtained. Since, in practice, researchers and technicians will not all use the same imaging conditions, it is crucial to examine whether sampling under varying conditions leads to stable measurements, or not. The sample measurements obtained from this scheme are given in Table 3.1, and the corresponding summary statistics are given in Tables 3.2 and 3.3. The key points of the summary statistics are discussed in Table 3.4.

Based on the observations made, along with visual inspection of the distributions, there appears to be no obvious outliers present in the data. These distributions are given in Figure 3.2 for Particle 1, with the rest of the distributions provided in Appendix A. Note that the x-axes in these figures are measured on a continuous scale, with midpoints at intervals of length 0.5 shown on the graphs. Since outliers can impact analysis, leading to incorrect conclusions, outlier testing and removal is an important step in any statistical investigation. As such, the inter-quartile range (IQR) outlier test is used to assess if any of the measurements are outliers. When the results of the IQR outlier tests were considered, there were no significant outliers in the data. The basic conditions for the IQR test are:

Table 3.1: Sample data for sampling conditions C1-C4 for three repetitions for each of 5 nanoparticles as shown in Figure 3.1.

First Sampling Condition (C1)	Second Sampling Condition	Repetition	Particle 1	Particle 2	Particle 3	Particle 4	Particle 5
500	C2	1	8.77	7.89	9.45	9.68	9.52
500	C2	2	8.19	8.4	9.88	9.96	9.36
500	C2	3	8.14	8.24	9.3	9.97	9.54
500	C3(b)	1	8.4	8.83	9.94	9.51	9.3
500	C3(b)	2	8.4	8.35	9.72	9.94	9.2
500	C3(b)	3	8.25	8.19	9.19	9.62	9.41
500	C4	1	8.19	8.93	9.51	9.8	9.53
500	C4	2	7.98	9.46	9.82	9.35	9.25
500	C4	3	8.24	8.35	9.98	9.67	9.3
250	C2	1	9.5	9.27	10.79	10.7	10.44
250	C2	2	9.27	9.44	10.27	10.87	10.68
250	C2	3	9.6	8.88	10.33	10.76	10.79
250	C3(a)	1	9.15	8.92	10.68	9.75	10.09
250	C3(a)	2	9.27	9.85	10.79	10.71	10.33
250	C3(a)	3	9.62	9.5	10.32	9.99	10.33
100	C2	1	9.11	9.04	9.94	10.37	10.18
100	C2	2	9.73	8.62	9.73	10.02	10.29
100	C2	3	9.18	7.51	9.45	9.18	10.01
100	-	1	9.11	9.04	9.94	10.37	10.18
100	-	2	9.73	8.62	9.73	10.02	10.29
100	-	3	9.18	7.51	9.45	9.18	10.01

Table 3.2: TEM sample statistics

Descriptive	Particle	Mean	Median	Skewness	Kurtosis	Std. dev.	Range
Complete sample	1	8.91	9.11	-0.16	-1.54	0.60	1.75
	2	8.71	8.83	-0.26	-0.40	0.64	**2.34**
	3	9.91	9.88	0.52	-0.49	0.47	1.60
	4	9.97	9.96	0.29	-0.68	0.50	1.69
	5	9.91	10.01	0.04	-1.39	0.51	1.59
Repetition 1	1	8.89	9.11	-0.46	-0.84	0.46	1.31
	2	8.85	8.93	**-2.08**	**5.06**	0.44	1.38
	3	10.04	9.94	0.57	-1.04	0.52	1.34
	4	10.03	9.80	0.46	-1.60	0.45	1.19
	5	9.89	10.09	-0.24	-1.89	0.43	1.14
Repetition 2	1	8.94	9.27	-0.23	**-2.12**	0.74	1.75
	2	8.96	8.62	0.47	-1.89	0.60	1.50
	3	9.99	9.82	**1.69**	**2.29**	0.40	1.07
	4	10.12	10.02	0.24	-0.10	0.51	1.52
	5	9.91	10.29	-0.20	**-2.34**	0.62	1.48
Repetition 3	1	8.89	9.18	-0.12	**-2.38**	0.66	1.48
	2	8.31	8.24	0.54	-0.09	0.71	**1.99**
	3	9.72	9.45	0.43	**-2.02**	0.48	1.14
	4	9.77	9.67	0.84	0.92	0.55	1.58
	5	9.91	10.01	0.51	-0.65	0.54	1.49

Table 3.3: TEM sample statistics (continued)

Descriptive	Particle	Mean	Median	Skewness	Kurtosis	Std. dev.	Range
100 000 Magnification	1	9.34	9.18	0.92	-1.88	0.30	0.62
	2	8.39	8.62	-0.67	-1.88	0.71	1.53
	3	9.71	9.73	-0.24	-1.87	0.22	0.49
	4	9.86	10.02	-0.62	-1.88	0.55	1.19
	5	10.16	10.18	-0.35	-1.88	0.13	0.28
250 000 Magnification	1	9.40	9.39	-0.05	-2.31	0.20	0.47
	2	9.31	9.36	0.16	-0.85	0.37	0.97
	3	10.53	10.51	0.07	-3.00	0.25	0.52
	4	10.46	10.71	**-1.01**	-1.20	0.47	1.12
	5	10.44	10.39	0.14	-0.76	0.26	0.70
500 000 Magnification	1	8.28	8.24	**1.23**	**2.51**	0.22	0.79
	2	8.52	8.35	0.95	**0.73**	0.47	1.57
	3	9.64	9.72	-0.39	-1.45	0.29	0.79
	4	9.72	9.68	-0.37	-0.74	0.22	0.62
	5	9.38	9.36	0.12	-1.61	0.13	0.34
Five minute time delay	1	9.05	9.18	-0.71	-0.75	0.58	1.59
	2	8.59	8.62	-0.36	-0.77	0.64	**1.93**
	3	9.90	9.88	0.56	-0.41	0.49	1.49
	4	10.17	10.02	-0.36	-0.52	0.56	1.69
	5	10.09	10.18	-0.16	-1.46	0.52	1.43
Upwards magnification	1	9.01	9.15	-0.29	-1.35	0.54	1.48
	2	8.76	8.83	-0.21	0.21	0.70	**2.34**
	3	9.97	9.94	0.30	-0.79	0.54	1.60
	4	9.90	9.94	0.31	0.20	0.46	1.53
	5	9.90	10.09	-0.72	-1.54	0.47	1.13
Down to 30 000 and up	1	8.14	8.19	**-1.48**	N/A	0.14	0.26
	2	8.91	8.93	-0.13	N/A	0.56	1.11
	3	9.77	9.82	-0.90	N/A	0.24	0.47
	4	9.61	9.67	**-1.14**	N/A	0.23	0.45
	5	9.36	9.30	**1.52**	N/A	0.15	0.28

Table 3.4: Sample statistics summary

Sampling condition	Summary
Repetition samples	• Mean measurements across different repetitions in the sampling scheme seem to be consistent (this is tested in the next section). • The medians being close to the means suggests approximate symmetry in the data. This is confirmed by the skewness measurements, with the exception of particle 2, repetition 1 and particle 3, repetition 2, where it is observed that these particles are negatively and positively skewed, respectively. • The kurtosis measurements vary across both the individual particles as well as the different repetitions. For those particles where the kurtosis is much greater than 0, the distribution will have very sharp peaks, whereas those particles with kurtosis much less than 0 have a multiple peaks, lower than that of a Normal distribution. • Even though there is varying results for the skewness and kurtosis, the standard deviation and range is consistently low, and close to the overall sample results, indicating that the values observed are clustered closely together.
Magnification level samples	• The means and medians across different magnification levels exhibit some variation. The means, however, are consistently close to the medians for each magnification level. • The skewness measurements vary from approximately symmetric to skewed for the different paricles at different magnification levels. • The kurtosis, with the exception of particle 1 and 2 at 500 000 times magnification, is consistently negative, indicating that its central peak is lower and broader, and its tails are shorter and thinner than that of a Normal distribution. For kurtosis measurements much less than zero, there are multiple peaks in the distributions per magnification level. For particle 1 and 2, at 500 000 times magnification, the distributions are flat, as the kurtosis is greater than 0. • Both the standard deviation and range measurements are consistently lower than the overall sample versions. • This, combined with the observations made for the mean and median, gives indication that there are differences in the distributions for the different magnification levels .

Note: Table continued on next page.

Sampling condition	Summary
Imaging condition	• The means for the different conditions appear to be close to the overall sample mean for all particles, with the respective medians close to the means. • The skewness measurements across different conditions mostly indicate moderately skew distributions, with the distributions for the "Down to 30 000 and up" condition being highly skewed. • There is negative kurtosis for majority of the conditions and particles, indicating that the distributions, compared to the Normal distribution, have a central peak that is lower and broader, with shorter and thinner tails. • For all sampling conditions, apart from "Down to 30 000 and up", the standard deviation and ranges are approximately the same as that of the overall sample for each of the particles. • For the "Down to 30 000 and up" condition the smaller sample size impacts the standard deviation and range.

- Compute the inter-quartile range: IQR=Q3-Q1, with Q3 the third quartile, and Q1 the first quartile.
- Compute the upper and lower bounds for testing as: $U = Q3 + 1.5 \times IQR$ and $L = Q1 - 1.5 \times IQR$.
- If a measurement lies below L or above U the measurement is considered an outlier.

3.4.4 Investigating equality in distributions

We next proceed with testing whether the different conditions C1-C4 under which sampling took place have any influence on the results obtained. To that effect, we test for equality of the underlying distribution of the data from different measurement conditions. Since the sample size is small ($n = 5$), and given that normality tests are impacted heavily by sample size, and on visual inspection of the sample distributions, it was decided to use nonparametric tests for the analysis of equality of distribution. Specifically, we considered the Kolmogorov-Smirnov test, the Wilcoxon Rank Sum test and the Ansari-Bradley test. The Wilcoxon rank sum test and the Ansari-Bradley test outperform the Kolmogorov-Smirnov test when the spreads and shapes, and the

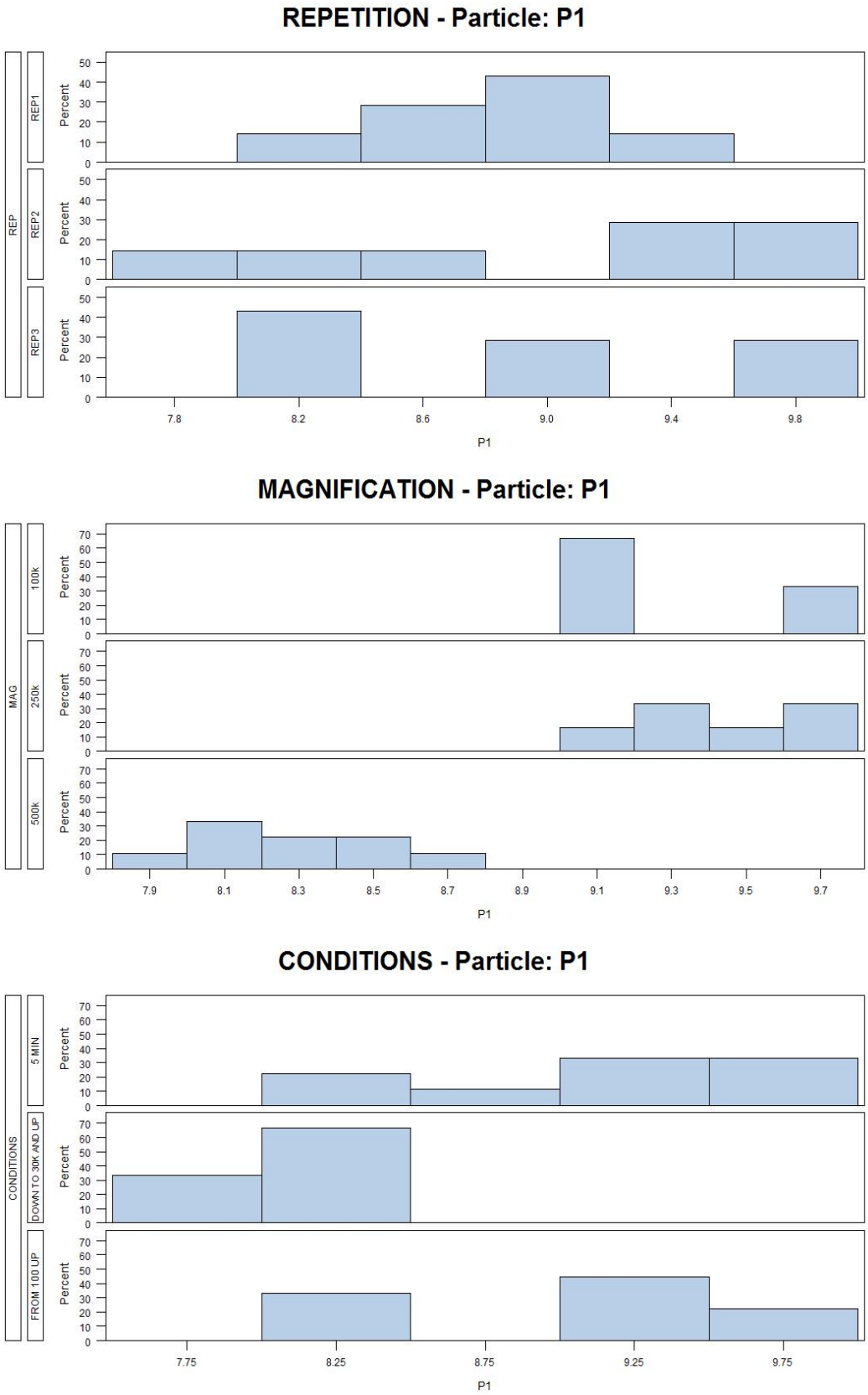


Figure 3.2: Sample distributions (Particle 1). This figure shows the distribution of the intensity values measured, split into the following categories: Repetition, Magnification and Other conditions. Repetition further splits the samples into the different repetitions, magnification into the different magnification levels used and conditions into the remainder of the sampling conditions. For each main group the distributions are split between the different levels, for example between repetitions 1, 2 and 3. Each sub-graph shows on the x-axis the intensity values and on the y-axis the percentage of observations within the range corresponding to the x-axis value.

median and shapes of the two distributions are the same [93]. Since the observations in Table 3.4 and visual inspection of the sample distributions indicate that this is not the case, these tests are not expected to outperform the Kolmogorov-Smirnov test. We therefore use Empirical Distribution Function (EDF) goodness of fit tests with the two-sample Kolmogorov-Smirnov test statistic to test the hypothesis H_0: *Equal distributions* against H_a: *Unequal distributions*. Details on the test is described in [113, 98, 84]. Note that though other techniques, like Q-Q plots, exist, it was decided to use the two-sample Kolmogorov-Smirnov test due to the abovementioned reasons, and not to consider other techniques. The procedure used for this study is as follows:

- For each condition under consideration (C1-C4), divide the data so that only two of the levels are present (for example, keep only Repetition 1 and Repetition 2 data).
- For each pair, test the null hypothesis specified above. Exact, rather than approximate tests, are performed due to small sample sizes.
- Test at a significance level of $\alpha = 0.05$, reject H_0: *Equal distributions* in favour of H_a: *Unequal distributions* if p-value < 0.05.

From the p-values obtained, given in Table 3.5, the following conclusions are made:

- Since all p-values for different repetition comparisons are ≥ 0.05, we cannot reject H_0: *Equal distributions*, and conclude that repeating the sampling does not cause changes in the underlying distribution.
- Since all but one of the p-values for different sampling conditions comparisons are ≥ 0.05, we cannot reject H_0: *Equal distributions*, and conclude that sampling under the different conditions does not cause changes in the underlying distribution.
- For the different magnification levels, there are varying results for the p-values, with majority of the p-values < 0.05. We can conclude that there is sufficient evidence supporting the alternative hypothesis that sampling at different magnification levels causes differences in measurements obtained.

The results of the initial study makes it clear that magnification levels have an impact on the measurement results. It is important to note that greater sample sizes, and therefore more particles and sampling repetitions, can have a significant impact on the results obtained.

Table 3.5: p-values for the Kolmogorov-Smirnov test.

Comparison	Particle 1	Particle 2	Particle 3	Particle 4	Particle 5
Rep1 & 2	0.5412	0.5412	0.5412	0.5412	0.5412
Rep1 & 3	0.5412	0.2032	0.5412	0.5412	0.9375
Rep2 & 3	0.9375	0.2032	0.2032	0.5412	0.9375
Mag 100 & 250	0.4413	0.1389	**0.0050**	0.1389	**0.0310**
Mag 100 & 500	**0.0015**	0.8186	0.9942	0.0815	**0.0015**
Mag 250 & 500	**0.0015**	**0.0257**	**0.0015**	**0.0135**	**0.0015**
5 Min & Down 30	0.1314	0.9639	0.9639	0.1314	0.1314
5 min & 100 up	0.9794	0.9794	1.0000	0.6994	0.6994
Down 30 & 100 up	**0.0222**	0.9639	0.9639	0.4910	0.2700

3.5 Concluding remarks and recommendations

In this preliminary study we have investigated the impact of varying imaging conditions on the measurements obtained using TEM. Commercially-produced aqueous colloidal gold is used for the investigation, where the average particle sizes are 10 nm. Different imaging conditions are considered, with three samples obtained for each condition. Sample statistics gave an indication that there is variation amongst different magnification levels, which was confirmed with EDF tests, using the two-sample Kolmogorov-Smirnov test statistic. The tests also confirmed that repeated sampling under the same conditions does not cause variation amongst the measurements obtained. This was also the case for sampling under different imaging conditions, such as time delay.

With initial results indicating that image measurements at varying magnification levels in TEM do cause variation in the measurements, the recommendation is made that magnification levels be carefully selected prior to obtaining samples using TEM. Given that instrument calibration is done at low and intermediate magnification (less than 200 000), it is possible the variation observed originates from the extrapolation of this relationship to higher magnifications. Conducting an actual calibration at higher magnification (rather than using extrapolated values) would almost certainly address this problem. Following on from the results of the preliminary study, further investigation will be done to explore the accuracy and repeatability of sample measurements obtained when using TEM in combination with image analysis software capable of performing thresholding and automated measurements. The automated measurement approach that will be used to conduct the study can be outlined as follows:

1. Capture images using TEM at 100 000 times magnification. This commonly used magnification level has proven to be a good compromise between particle visibility and image size (and therefore number of particles in the image);

2. Apply a simple thresholding to the image using ImagePro®;

3. Obtain several different images from the same sample image, by selecting a number of non-overlapping areas in the sample image. This allows the user to obtain similar samples from independent regions with a varying number of particles in each new image;

4. For each of the new images obtain a selection of measurements using the built-in capabilities of ImagePro®. These include measurements such as:

    - Mean region diameter,
    - Circularity,
    - Fractal dimension,
    - Roundness,
    - Maximum and minimum diameter,
    - Mean centroid;

5. Convert the RGB image to an 8-bit greyscale image, followed by blob detection is done using 8-connectivity, where areas are restricted to a predetermined minimum and maximum number of pixels;

6. Perform a watershed segmentation to split touching particles, using the default setting in ImagePro®;

7. Identify and remove "oddly" shaped particles using visual inspection.

Performing such a study is necessary to ensure consistent sample measurements are obtained using either manual measurements, or the built-in capabilities of ImagePro®.

Chapter 4

Object classification in nanoimages

4.1 Background to object classification in nanoimages

As discussed in Chapter 3, the characterisation of nanoparticle properties (such a diameter, particle size distribution and surface area) is important for application implementation [102][1]. As such we need to have a reliable method for identifying the nanoparticles in the nanoimage, from which we can then be able to obtain their characteristics. In the TEM analysis, occlusion of particles provide challenges in recognising these particles, and the object classification procedure must therefore be able to correctly identify these occluded particles. The aim of this chapter is to discuss the detail of such a method, and to provide an illustration of the implementation of this method.

To set the scene, a brief overview of some recent work on object classification in nanoimages is provided. Huitink *et al.* [57], in 2010, present a multivariate statistical approach to analyse nanoparticles and their boundaries. The technique used includes a semi-supervised learning procedure, with an embedded Bayesian multiclass logistic regression method to classify possible object shapes (e.g. circles, ellipses, triangles). The procedure in [57] is outlined as follows:

1. An edge detection algorithm detects object boundaries, where edges are identified as all the line segments on which colour differences are significantly high;

2. After noise removal and collecting only those boundaries that form complete closed shapes, the centroids of each of the shapes are found;

3. The identified shapes are classified into groups of similar shapes;

[1] As discussed in Chapter 3, application implementation relates to the development and design of smarter devices and more precise solutions to problems faced in some of the applications where this technology is currently used [127, 124, 82].

4. Finally, the most probable shapes are identified - using a Bayesian multiclass regression method - and overlapping particles are separated into individual shapes.

The procedure in [57] is effective in capturing $85\% - 95\%$ of the shapes present in the images. The results of [57] are fundamental to the work done by Park *et al.* [94] (2011) and Konomi *et al.* [70] (2013) since the results give a finite set of possible shapes that the nanoparticles can take.

Park *et al.* [94], in 2011, present a multi-stage, semi-automated procedure that can be used in morphological analysis of nanoparticles. The procedure makes use of several statistical learning tools, including multidimensional scaling, semi-supervised clustering, multiclass classification, peak detection, and functional principle component analysis (fPCA). The procedure in [94] is outlined as follows:

1. Particle boundaries are extracted using an edge detection algorithm, and a simple thresholding rule is applied to remove unnecessary edges;

2. The extracted boundaries are changed into parametric curves, to ensure invariance under rotation, scaling and translation;

3. By using a rotationally invariant similarity measure on the space of parametric curves and a non-linear projection of these parametric curves to a low dimensional Euclidean space, a rotationally invariant, reduced dimension feature set is created that can be used in shape clustering;

4. Graph-based semi-supervised learning is used, where labeled and unlabeled data are represented as vertices in a connected graph, to label the data;

5. Convexity analysis is used to split a composite boundary into individual boundaries, each of which becomes an incomplete boundary;

6. A $k-$nearest neighbour classifier is used to classify a nanoparticle with incomplete neighbour information;

7. A shape recovery method, using fPCA, is used to estimate the missing part of an incomplete boundary;

8. Finally, summary statistics of the morphology of the nanoparticles are obtained, using

   (a) the size distribution,

   (b) the shape distribution, and

   (c) the distribution of the aspect ratios.

The procedure suggested in [94] is successful in recognising $94 - 100\%$ of particles for images with small number of overlapping particles, and $78 - 95\%$ of particles for images with a multitude of overlapping particles. The drop off in classification success from images with small overlap to those with more overlapping particles makes the procedure less useful in applications where TEM images are used, due to the challenge faced with occlusion of objects in these images.

In 2013, Konomi *et al.* [70] address this issue by presenting an automatic image segmentation and classification procedure, which simultaneously detects the boundaries and classifies the nanoparticles into one of the predetermined shape families. A high level outline of the procedure proposed in [70] is given in Figure 4.1, and can briefly be described as follows:

1. By treating each particle in the image as an object, it is possible to specify each object uniquely using a set of object parameters, such as, but not limited to, scale, location, and rotation. Since these parameters will vary from object to object, the parameters are assumed unknown. More details on the specification of objects are provided in Sections 4.4.1 and 4.4.2;

2. The objects are then modelled as a Markov Point Process (MPP), using the Area Interaction Process Prior (AIPP). Since there is different degree of overlapping from image to image, the parameters of the AIPP are assumed unknown. For now the following may be noted: An MPP is a point process with the property that its distribution possesses a density that is a Markov function [99]. A brief introduction to MPPs and more details on the AIPP specification used in this analysis are discussed in Section 4.2;

3. A Bayesian framework is used to infer the object parameters. This requires the specification of prior distribution(s) for the unknown parameters of interest. In a closed form Bayesian analysis the posterior distribution can be obtained analytically. However, in this instance, the unknown AIPP parameters lead to an intractable normalising constant, which in turn leads to an intractable posterior. The priors used in this analysis are discussed in Section 4.4.5;

4. Owing to the complexity in obtaining the posterior distribution, an alternative approach is needed. A Markov Chain Monte Carlo (MCMC) framework is used to simulate the unknown posterior distribution, and is particularly useful in this analysis due to the presence of an unknown normalising constant.

Since an appropriate theoretical overview of the techniques used in this analysis has not yet been provided in this chapter, we only give a brief overview of the steps in Figure 4.1, with Figure 4.2 providing further detail on step Main 4 in Figure 4.1. After

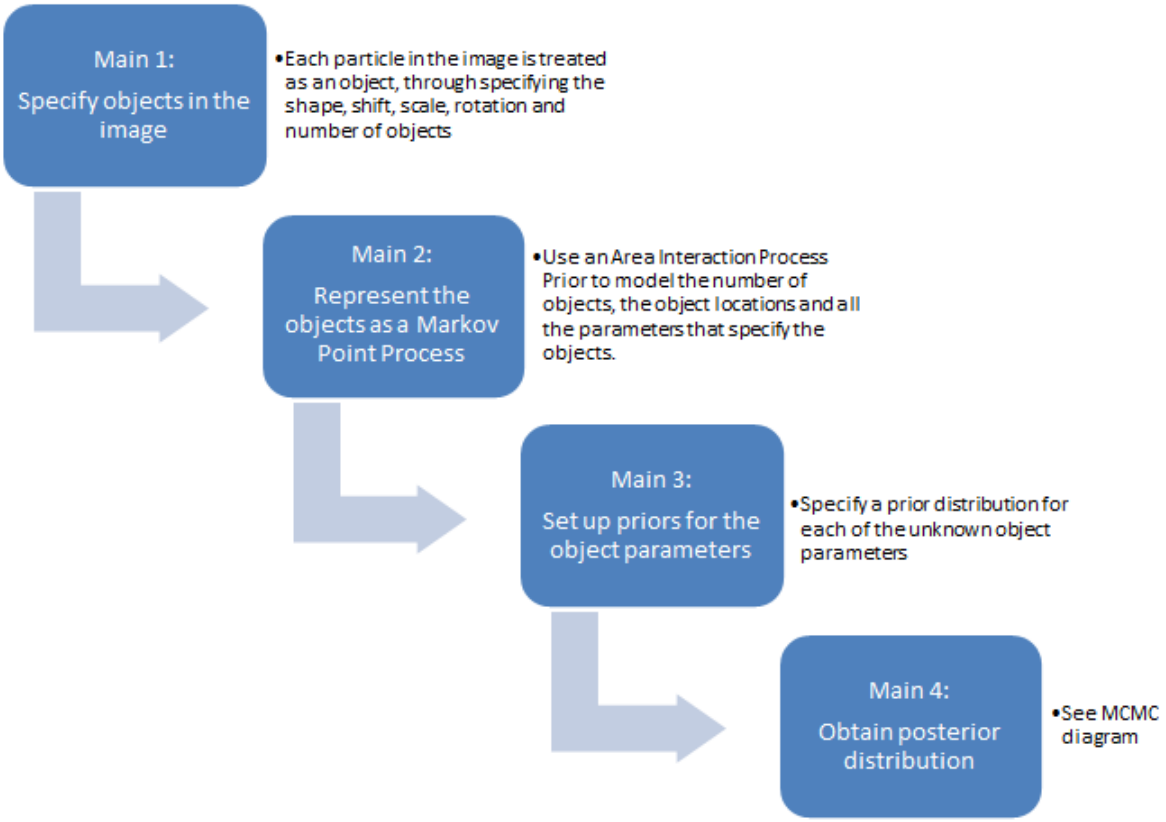


Figure 4.1: Outline of Konomi *et al*'s procedure, showing the main steps (or components) that make up the algorithm. Note that step 'Main 4' is further outlined in Figure 4.2. [70].

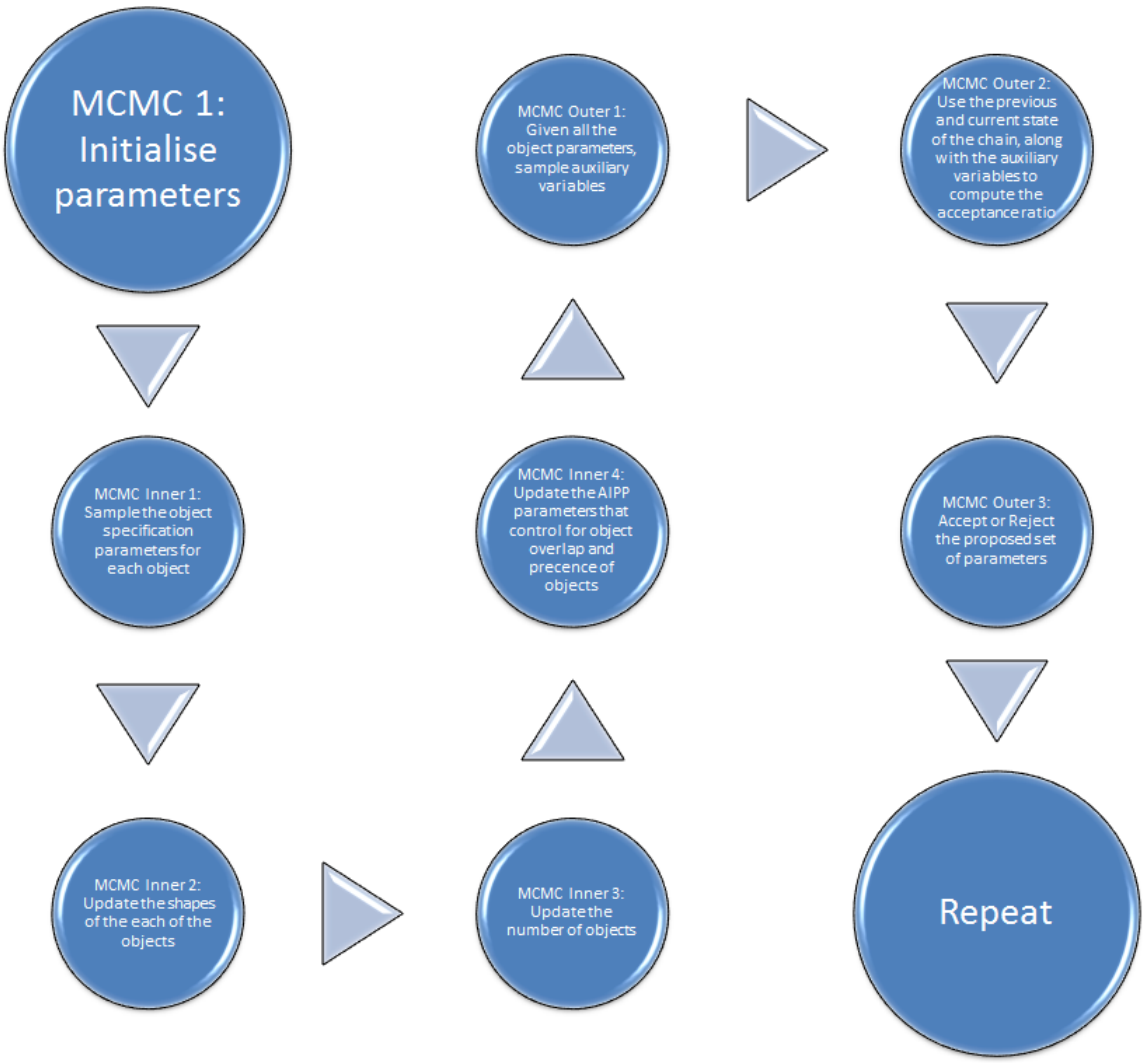


Figure 4.2: MCMC diagram detailing further the steps followed to obtain the posterior distribution, that is, step 'Main 4' in Figure 4.1. The steps in this diagram are divided into 'MCMC Inner' and 'MCMC Outer' steps, where multiple repeats of the 'MCMC Inner' steps can occur in a single 'MCMC Outer' step.

the theoretical background to the techniques have been provided, detailed descriptions of steps Main 1 - Main 4 are provided in Sections 4.4 and 4.5. The aim in this chapter is to use this method to classify the objects in the nanoimage of gold nanoparticles obtained using TEM, with the main motivation being that this method is an automated classification algorithm that can deal with occluded objects. The remainder of this chapter is set out as follows: Section 4.2 gives a brief introduction to MPPs and provides a definition for the AIPP; Section 4.3 provides an overview of MCMC methods; in Section 4.4 we discuss the details of steps Main 1, Main 2 and Main 3 as labeled in Figure 4.1. In Section 4.5 details of step Main 4 in Figure 4.1, or alternatively steps MCMC 1, MCMC Inner 1 - MCMC Inner 4 and MCMC Outer 1 - MCMC Outer 3 as labeled in Figure 4.2 is provided. Section 4.7 provides details of the simulation

results of this algorithm, implemented in Matlab®, with Section 4.9 providing some concluding remarks and recommendations.

4.2 Markov Point Processes (MPPs) and the Area Interaction Process (AIPP)

Since the specification of an AIPP is fundamental to the occlusion algorithm used in this study, we provide here an overview of MPPs and in particular the AIPP. Some preliminary notation and definitions are needed, as provided in [99]:

- $(X, \mathscr{X}, \mu)$ is a finite measure space, with the $\sigma-$field $\mathscr{X}$ containing all singletons[2];
- $(X_e, \mathscr{X}_e, \nu)$ is the corresponding exponential space, as defined in [23], that is, X_e is the union of the classes $X_1, \ldots, X_n \subseteq X$[99];
- ν is the distribution of the Poisson process with mean measure μ;
- if $t_1, t_2, t_3, \ldots$ are independent $EXP(\lambda)$ random variables, with $T_n = t_1 + t_2 + \ldots + t_n$ for $n \geq 1$, and $T_0 = 0$, then $N(s) = \max\{n : T_n \leq s\}$ is a called a Poisson process, and $N(s) \sim POI(\lambda s)$[36] (see [36] for a detailed discussion on Poisson processes);
- an environment $E(A)$ of $A \subset X$ is defined as $E(A) = \{\xi \mid \xi \sim \eta \text{ for some } \eta \in A\}$ and '$\sim$' is a measurable symmetric reflexive relation;
- a function $f : X_e$ is called a Markov function if and only if there is a function $g : X \times X_e$ such that

$$f(x \cup y) = f(x)\, g(y, x \cap E(y)), \text{ for all } x \in X_e, y \in X.$$

Then a point process on X is a measurable map from a probability space[3] to $(X_e, \mathscr{X}_e)$, with its distribution being the probability induced on $\mathscr{X}_e$, and a Markov Point Process (MPP) is a point process whose distribution has a density with respect to ν, which is a Markov function [99]. Based on the theory provided on MPPs by [99], Baddeley

[2]
- A finite measure space is a measurable space with a finite number of nonnegative measures [15],
- a $\sigma-$field (or $\sigma-$algebra) $\mathscr{X}$ is a non-empty collection of subsets of X such that (i) X is in $\mathscr{X}$, (ii) if a set A is in $\mathscr{X}$ then so is the complement of A, and (iii) if A_n is a sequence of elements of $\mathscr{X}$ then the union of all of the elements $\{A_i\}_{i=1}^n$ is in $\mathscr{X}$[15].
- a singleton is a set with exactly one element.

[3] A probability space is a measurable space $(S, \mathscr{S}, P)$, with P a measure defined on $\mathscr{S}$, with $P(S) = 1$ [15].

and van Lieshout in 1995 [9] introduced an MPP called the Area Interaction Process Prior (AIPP). The general case of the AIPP is defined as follows [9]: The AIPP in a compact region $A \subseteq \mathbb{R}^d$ is the process with density

$$p(c) = \alpha\beta^{n(c)}\gamma^{-\nu(U(c))},$$

with respect to the unit rate Poisson process ($\lambda = 1$) on A, where

- $c = \{c_1, \ldots, c_m\}, c_i \in X$;
- $n(c) = m$ is the total number of points;
- β, γ, r are the parameters;
- α is the normalising constant;
- ν is a finite Borel regular measure, that is for each set $X \subseteq \mathbb{R}^d$ there exists a Borel set $B \supset X$ (an element of the $\sigma-$field generated by the open sets), such that $\nu(B) = \nu(X)$; and
- $U(c) = \cup_{i=1}^{m} Z(c_i)$ with Z a myopically continuous function that assigns to each $c_i \in X$ a compact set $Z(c_i) \subseteq X$.

For the study at hand, consider the case where we wish to model the objects in an observed image $y = \{y_t : t \in T\}$, where the image space T is a finite set, and in the case of a greyscale image $t \in T$ ranges over the set $\{0, 1, \ldots, 255\}$ (as discussed in Chapter 2). Denote by X the (finite) set of possible parameter vector values that can be used to represent the objects, so that a point $c \in X$ represents an object $R(c) \subseteq T$, then an object configuration is an unordered list of objects $c = \{c_1, \ldots, c_m\}, c_i \in X$ [8]. Further denote by $S(c) = \cup_{i=1}^{m} R(c_i)$ the silhouette formed by taking the union of all the objects in the configuration. Using this setting, an overlapping object model can be specified using the AIPP, defined as

$$p(c) = \alpha\beta^{n(c)}\gamma^{|S(c)|},$$

with parameters $\beta > 0$, $\delta \geq 1$, and $|S(c)|$ is the area, or pixel count, of the silhouette [8]. We discuss the specification of objects in the image in Sections 4.4.1 and 4.4.2, with details on the AIPP used in the occlusion algorithm provided in Section 4.4.3.

4.3 Markov Chain Monte Carlo methods

Real world statistical problems involve observed data x that we wish to use in determining some unknown quantity (or vector of quantities) θ of interest. In the Bayesian framework [79, 45], data analysis amounts to

1. specifying a sampling model X for the observed data x, conditional on the unknown parameter θ,

$$X \sim f(x \mid \theta),\ x \in \mathcal{X}; \theta \in \Theta,$$

where $f(\cdot)$ is the probability density function (pdf) or probability mass function (pmf) depending on the application at hand, $\mathcal{X} \subseteq \mathbb{R}^n$ is the sampling space and Θ is the parameter space ;

2. specifying a marginal distribution $\pi(\theta)$ for θ, the prior distribution,

$$\theta \sim \pi(\theta),\ \theta \in \Theta;$$

3. using Bayes' theorem to compute the posterior distribution,

$$\pi(\theta \mid x) = \frac{\pi(\theta) L(\theta \mid x)}{\int \pi(\theta) L(\theta \mid x) d\theta},\ \theta \in \Theta, \tag{4.1}$$

where $L(\theta \mid x) \propto f(x \mid \theta)$ is referred to as the likelihood of θ given x.

Since we are interested in the characteristics of the posterior distribution (such as the mean and variance), we must be able to draw samples from $\pi(\theta \mid x)$. This can be done using direct sampling techniques (for closed form case), or Monte Carlo methods (for more complicated cases where a closed form is not attainable). One of the most well-known direct sampling techniques, the Inverse CDF method, relies on getting the inverse distribution function to perform sampling. The inverse CDF method [100] generates a random variable X with a cumulative distribution function F by first sampling a random variable $U \sim UNI(0,1)$ from a standard uniform distribution, then applying the inverse transformation to obtain the desired distribution $X = F^{-1}(U) = \min\{x \mid F(x) \geq U\}$. It is a common occurrence that an analytical form for F is unknown or it is impractical to obtain the inverse form of the distribution function F. In such instances, Monte Carlo methods are popular to generate samples from the target distribution [42]. With Monte Carlo methods, the aim is to generate samples from a given probability distribution and to use these samples to estimate expectations of functions under this distribution (such as the mean or variance). Monte Carlo methods rely on the fact that $\pi(\theta) f(x \mid \theta)$ is proportional to a density [100]. With these methods, given some measurable function $h(\theta)$, the Law of Large Numbers is used to calculate $\int h(\theta) \pi(\theta) f(x \mid \theta) d\theta$, since, given that we are able to generate random variables $\{\theta_i\}_{i=1}^K$ from $\pi(\theta)$, the average $\frac{1}{K} \sum_{i=1}^K h(\theta_i) f(x \mid \theta_i)$ almost surely converges to $\int h(\theta) \pi(\theta) f(x \mid \theta) d\theta$ as $K \longrightarrow \infty$ [100]. In addition, if an i.i.d (independent and identically distributed) sample of θ's can be obtained from the posterior

$\pi(\theta \mid x)$, then [100]

$$\lim_{K \longrightarrow \infty} \left(\frac{1}{K} \sum_{i=1}^{K} h(\theta_i) \right) = \mathbb{E}[h(\theta) \mid x] = \frac{\int h(\theta) \pi(\theta) f(x \mid \theta) d\theta}{\int \pi(\theta) f(x \mid \theta) d\theta}.$$

In one of the commonly used Monte Carlo methods, an importance function $g(\theta)$, with the support of $g(\theta)$ including the support of $\mid h(\theta) \mid \pi(\theta) f(x \mid \theta)$, is used to generate random variables $\{\theta_i\}_{i=1}^{K}$, and $\frac{1}{K} \sum_{i=1}^{K} h(\theta_i) \omega_i(\theta_i)$ is used to approximate $\int h(\theta) \pi(\theta) f(x \mid \theta) d\theta$, with $\omega_i = f(x \mid \theta_i) \pi(\theta_i) / g(\theta_i)$ [100]. The reader is referred to [42] for a detailed overview of this method known as the Importance algorithm and other Monte Carlo methods.

Alternatives to Monte Carlo methods have been proposed to address the problem of intractable normalising constants. One of the approaches, Markov Chain Monte Carlo (MCMC), has proven to be particularly useful and popular, and in some cases the only feasible approach [110]. A Markov chain is a sequence of random variables $\left\{X^{(i)} : i = 0, 1, 2, \ldots\right\}$ with the Markov property:

$$\begin{aligned} p(x, y) &= \mathrm{P}\left(X^{(t+1)} \in A \mid X^{(0)} = x_0, X^{(1)} = x_1, \ldots, X^{(t)} = x_t\right) \\ &= \mathrm{P}\left(X^{(t+1)} \in A \mid X^{(t)} = x_t\right), \end{aligned}$$

for $t = 0, 1, \ldots$, with x, y two states, where $A \in \mathcal{X}$ is some measurable set [79] and let $P = \{p(x, y)\}$ be the transition matrix. In other words, given the present state, the next state is dependent only on the current state and independent of past states. The basic idea behind MCMC is as follows: Suppose we want to generate samples from a distribution $f(x)$, with $x \in \mathcal{X} \subseteq \mathbb{R}^n$ but cannot do this directly. However, if we were able to construct a Markov chain with state space $\mathcal{X}$ (from which we are able to simulate) with $f(x)$ the invariant distribution of the Markov chain, we could run the chain for a sufficiently long period to obtain simulated values from which we can study features of $f(x)$ [110]. In order for f to be the invariant distribution it must satisfy $f(\cdot) \times P(\cdot) = f(\cdot)$, but it is also sufficient for the transition matrix to satisfy the detailed balance condition [36]

$$f(x) p(x, y) = f(y) p(y, x), \text{ for all } x \neq y. \tag{4.2}$$

To visualise this, think of the invariant as having $f(x)$ litres of some liquid at the starting position in container x (and $f(y)$ litres in container y), and when a transition is made in the Markov chain, a fraction, $p(x, y)$, of the liquid is moved from x to y. This transition is repeated a number of times, until all the transitions have been

made to reach an equilibrium state - for example when the weight of the containers are appropriately spread for transportation. If at the end of all the transitions the distribution of liquid between x and y has remained the same then $f(x)$ is an invariant distribution. That is, for each fraction $p(x,y)$ of liquid transferred from the $f(x)$ litres of liquid (transferred from x to y), the same amount of liquid is transferred back from y to x, expressed as a fraction of liquid, $p(y,x)$, from the starting $f(y)$ litres of liquid. For the detailed balance condition, the amount of liquid that is transferred from x to y in one transition is exactly balanced by the amount of liquid being transferred back from y to x.

The discussion which follows will concentrate on key ideas and concepts of MCMC, with focus on implementation strategies specifically in relation to object classification in nanoimages. For a detailed overview of MCMC methods that includes appropriate theoretical frameworks, the reader is referred to [79, 14, 116, 45]. In order to implement a MCMC strategy, some algorithms are needed that allows the construction of chains with specified equilibrium distributions. In sections 4.3.1 to 4.3.5 some of the key algorithms applicable to this investigation are discussed.

4.3.1 The Gibbs algorithm

The Gibbs algorithm, first introduced by Geman and Geman in 1984 [41], was popularised by Gelfand and Smith in a 1990 paper [40], where they demonstrated the value of using the Gibbs algorithm in the Bayesian analysis framework. The Gibbs algorithm is an iterative sampling scheme, in which sampling is done as follows: Suppose samples need to be drawn from a target density $f(x)$, $x \in \mathcal{X} \subseteq \mathbb{R}^K$, with $x = (x_1, \ldots, x_K)'$ and we denote by

$$f_k(x_k \mid x_1, \ldots, x_{k-1}, x_{k+1}, \ldots, x_K), \qquad k = 1, \ldots, K \tag{4.3}$$

the set of conditional distributions, from which samples can be generated.

1. Given an arbitrary starting point $x^{(0)} = \left(x_1^{(0)}, \ldots, x_K^{(0)}\right) \in \mathcal{X}$ from $f\left(x^{(0)}\right) > 0$, iterate for $t = 1, 2, \ldots$;

2. For $k = 1, \ldots, K$, generate

   1 $\quad x_1^{(t)} \sim f_1\left(x_1 \mid x_2^{(t-1)}, \ldots, x_K^{(t-1)}\right)$,

   $\vdots$

   k $\quad x_k^{(t)} \sim f_k\left(x_k \mid x_1^{(t)}, \ldots, x_{k-1}^{(t)}, x_{k+1}^{(t-1)}, \ldots, x_K^{(t-1)}\right)$,

   $\vdots$

$$K \qquad x_K^{(t)} \sim f_K\left(x_K \mid x_1^{(t)}, \ldots, x_{K-1}^{(t)}\right).$$

It has been shown that the distribution of $x^{(t)} = \left(x_1^{(t)}, \ldots, x_K^{(t)}\right)'$, denoted by $f_t(x)$, will converge to $f(x)$ [79, 101]. Note that for iteration k, $x_i^{(t)}$ rather than $x_i^{(t-1)}$ is used for $i = 1, \ldots, k-1$, since the updated $x_i^{(t)}$ will already have been generated. The Gibbs algorithm will therefore be useful when the conditional distributions $f^{(t)}$ are more tractable, i.e. can be sampled from, compared to the target density $f(x)$.

4.3.2 Metropolis-Hastings Algorithm

Metropolis *et al.* [85] introduced the Metropolis algorithm, which was later generalised to the Metropolis-Hastings algorithm by Hastings [54]. The latter is useful in cases where the conditional distributions of some (or all) of the components in $x = (x_1, \ldots, x_K)'$ are not standard. Considering the target distribution $f(x)$ on the sample space $\mathcal{X}$, the basic idea is to create a Markov chain with some transition kernel P, with invariant distribution $f(x)$. In addition, to simplify the strategy, the Markov chain is restricted to be reversible (i.e to satisfy the detailed balanced condition), with $p\left(x^{(t-1)}, y\right)$ the transition probability as defined at the beginning of this section. Given $x^{(t-1)}$, the state of the Markov chain at time $t-1$, a two-step approach is used to construct the transition kernel by specifying a symmetric proposal distribution with pdf $q\left(y \mid x^{(t-1)}\right) = q\left(x^{(t-1)} \mid y\right)$ and adjusting random draws from $q\left(y \mid x^{(t-1)}\right)$ using an accept-reject rule. Note that throughout this chapter $q(\cdot)$ will be used to refer to a proposal distribution, where the specific distribution used is dependent on the application at hand. In many applications, the proposal distribution is chosen to be a uniform distribution. The Metropolis algorithm is defined as follows:

Metropolis algorithm

1. Draw y from $q\left(y \mid x^{(t-1)}\right)$;
2. Compute the acceptance ratio as $\alpha = \alpha\left(x^{(t-1)}, y\right) = \min\left\{1, \frac{f(y)}{f\left(x^{(t-1)}\right)}\right\}$;
3. Set $x^{(t)} = y$ with probability α and set $x^{(t)} = x^{(t-1)}$ with probability $1 - \alpha$.

The acceptance ratio can intuitively be interpreted as follows: If the target density at the proposed point y is greater than the target density in the point $x^{(t-1)}$, the current state, in other words it is more likely to observe the proposed state, then the ratio $\frac{f(y)}{f\left(x^{(t-1)}\right)} > 1$, and the acceptance ratio will be set to $\alpha = 1$. We therefore set $x^{(t)} = y$ with probability 1, and use the proposed state as the next state. The algorithm above was generalised by Hastings by allowing the proposal distributions to be asymmetric

[54, 79]. In mathematical terms, the condition $q\left(y \mid x^{(t-1)}\right) = q\left(x^{(t-1)} \mid y\right)$ is relaxed to allow for $q\left(y \mid x^{(t-1)}\right) \neq q\left(x^{(t-1)} \mid y\right)$:

Metropolis-Hastings (MH) algorithm

1. Draw y from $q\left(y \mid x^{(t-1)}\right)$;
2. Compute the acceptance ratio as $\alpha = \alpha\left(x^{(t-1)}, y\right) = \min\left\{1, \frac{f(y)q\left(x^{(t-1)}|y\right)}{f\left(x^{(t-1)}\right)q\left(y|x^{(t-1)}\right)}\right\} = \min\left\{1, \left(\frac{f(y)}{q\left(y|x^{(t-1)}\right)} / \frac{f\left(x^{(t-1)}\right)}{q\left(x^{(t-1)}|y\right)}\right)\right\}$;
3. Set $x^{(t)} = y$ with probability α and set $x^{(t)} = x^{(t-1)}$ with probability $1 - \alpha$.

Compared to the acceptance ratio for the Metropolis algorithm, and considering that $q\left(y \mid x^{(t-1)}\right) = q\left(x^{(t-1)} \mid y\right)$, the acceptance ratio for the Metropolis algorithm can therefore be expressed as $\alpha = \min\left\{1, \frac{f(y)}{f\left(x^{(t-1)}\right)}\right\} = \min\left\{1, \frac{f(y)q\left(x^{(t-1)}|y\right)}{f\left(x^{(t-1)}\right)q\left(y|x^{(t-1)}\right)}\right\}$, it is easy to see that the MH algorithm is defined in exactly the same manner. As explained in [36], the MH algorithm can be seen as starting with a Markov chain $q\left(y \mid x^{(t-1)}\right)$ (the proposed jump distribution) and a move is accepted with probability

$$\alpha\left(x^{(t-1)}, y\right) = \min\left\{1, \frac{f(y)\, q\left(x^{(t-1)} \mid y\right)}{f\left(x^{(t-1)}\right) q\left(y \mid x^{(t-1)}\right)}\right\},$$

so that the transition probability is given by $p\left(x^{(t-1)}, y\right) = q\left(y \mid x^{(t-1)}\right) \alpha\left(x^{(t-1)}, y\right)$. As noted in [36], this transition probability satisfies the detailed balance condition. Note that this statement is also valid for the Metropolis algorithm, since $q\left(y \mid x^{(t-1)}\right) = q\left(x^{(t-1)} \mid y\right)$.

The Independence algorithm

In this case we draw y independently of the current state of the Markov chain , i.e. $q\left(y \mid x_t\right) = q\left(y\right)$. In this case, the ratio in step 2 above becomes

$$\alpha = \alpha\left(x_t, y\right) = \min\left\{1, \frac{w\left(y\right)}{w\left(x_t\right)}\right\} = \min\left\{1, r\left(x_t, y\right)\right\},$$

with $w\left(y\right) = \frac{f(y)}{q(y)}$ and $w\left(x_t\right) = \frac{f(x_t)}{q(x_t)}$. The acceptance ratio again has the same formulation as for the Metropolis and MH algorithms, with the difference being that the proposal distributions have no conditioning, as they are drawn independently of the current state, and therefore this acceptance ratio leads to a transition probability that satisfies the detailed balance condition.

It can be seen that in all of the algorithms above the acceptance ratios involves comparing the value of the target density at the proposed point to the value of the

proposed density at the current point (with some variations for asymmetric proposals and independent proposals), with the proposed values being accepted with probability of 1 if the value at the proposed point is greater than the value of the target density at the current point (i.e. if the proposed point is more likely).

4.3.3 Metropolis-Hastings-within-Gibbs algorithm (MHWG)

In certain cases, some (or all) of the components/conditional distributions used in the Gibbs algorithm cannot be easily simulated. A compromised algorithm is suggested, by replacing a step in the Gibbs algorithm where it is difficult to simulate from the conditional distribution $f_k\left(x_k \mid x_1, \ldots, x_{k-1}, x_{k+1}, \ldots, x_K\right)$ with a single MH step for each iteration [88, 87]. The algorithm has the following form:

For each iteration $k = 1, \ldots, K$ and given $\left(x_1^{(t)}, \ldots, x_{k-1}^{(t)}, x_{k+1}^{(t-1)}, \ldots, x_K^{(t-1)}\right)$, perform a single MH step as follows:

1. Draw y_k from $q_k\left(x_k \mid x_1^{(t)}, \ldots, x_{k-1}^{(t)}, x_{k+1}^{(t-1)}, \ldots, x_K^{(t-1)}\right)$;

2. Compute the acceptance ratio as $r = \frac{A(y_k)}{A^*\left(x_k^{(t-1)}\right)}$, with

$$A\left(y_k\right) = \frac{f_k\left(y_k | x_1^{(t)}, \ldots, x_{k-1}^{(t)}, x_{k+1}^{(t-1)}, \ldots, x_K^{(t-1)}\right)}{q_k\left(y_k | x_1^{(t)}, \ldots, x_{k-1}^{(t)}, x_k^{(t-1)}, x_{k+1}^{(t-1)}, \ldots, x_K^{(t-1)}\right)},$$

$$A^*\left(x_k^{(t-1)}\right) = \frac{f_k\left(x_k^{(t-1)} | x_1^{(t)}, \ldots, x_{k-1}^{(t)}, x_{k+1}^{(t-1)}, \ldots, x_K^{(t-1)}\right)}{q_k\left(x_k^{(t-1)} | x_1^{(t)}, \ldots, x_{k-1}^{(t)}, y_k, x_{k+1}^{(t-1)}, \ldots, x_K^{(t-1)}\right)},$$

   and $\alpha = \alpha\left(x_k^{(t-1)}, y_k\right) = \min\{1, r\}$. Note that in each of the ratios $A\left(y_k\right)$ and $A^*\left(x_k^{(t-1)}\right)$, the k^{th} component is not present in the conditional target density (as we are conditioning on this component), and the proposal densities are conditioned as follows: For y_k the proposal is conditioned on

   (a) all of the $x_i^{(t)}$ for which the updated components are available, that is for $i = 1, \ldots, k-1$,

   (b) and on all of the $x_j^{(t-1)}$ for which the updated components at state t are **not** available, that is for $j = k, k+1, \ldots, K$.

   For $x_k^{(t-1)}$ the proposal is conditioned in the same manner as for y_k, except for the k^{th} component, which is replaced by y_k, the proposed value for the k^{th} component, since in this case an "updated" value for the k^{th} component is available;

3. Set $x_k^{(t)} = y_k$ with probability α and set $x_k^{(t)} = x_k^{(t-1)}$ with probability $1 - \alpha$.

Recall from Section 4.3.2 that the acceptance ratio for the MH algorithm is given by the ratio $\alpha = \min\left\{1, \left(\frac{f(y)}{q\left(y|x^{(t-1)}\right)} / \frac{f\left(x^{(t-1)}\right)}{q\left(x^{(t-1)}|y\right)}\right)\right\}$. In the case of the MHWG algorithm, the ratio is expressed in a similar manner, but the conditional target and proposal distributions are used in stead, since these are the distributions of interest in this algorithm. As with the MH algorithm this choice of α leads to a transition probability $p\left(x_k^{(t-1)}, y_k\right) = q\left(y_k \mid x_k^{(t-1)}\right) \times \alpha\left(x_k^{(t-1)}, y_k\right)$.

4.3.4 The Monte Carlo Metropolis-Hastings (MCMH) algorithm

When sampling from distributions with an intractable normalising constant, the acceptance ratio contains unknown constants. As mentioned in Section 4.1, in this study the normalising constant is intractable due to the unknown AIPP parameters. Several algorithms have been proposed to deal with this problem [77, 86, 89, 24, 14, 43]. In this section we discuss a Monte Carlo version of the Metropolis-Hastings algorithm presented by Liang (2013) [78].
Suppose we have a data set x generated from a distribution with likelihood function

$$f(x \mid \theta) = \frac{1}{\kappa(\theta)} g(x;\theta), \qquad x \in \chi,\ \theta \in \Theta, \tag{4.4}$$

where θ is the parameter and $\kappa(\theta)$ is the normalising constant that depends on the unknown parameter θ. Further denote by $\pi(\theta)$ the prior density of θ, then the posterior density is given by

$$\pi(\theta \mid x) \propto \frac{1}{\kappa(\theta)} g(x;\theta)\, \pi(\theta). \tag{4.5}$$

In this instance, the MH algorithm described in Section 4.3.2 will have an unknown ratio $\frac{\kappa(\theta^*)}{\kappa\left(\theta^{(t-1)}\right)}$ in the acceptance ratio α, with θ^* the proposed value of θ and $\theta^{(t-1)}$ the current draw of θ. With the MCMH algorithm proposed by Liang in 2013 [78], the unknown ratio $\frac{\kappa(\theta^*)}{\kappa\left(\theta^{(t-1)}\right)}$ is replaced by a Monte Carlo estimate at each iteration.

Suppose we want to sample from the distribution given in (4.5), the algorithm works by iterating over the following steps:

1. Draw θ^* from a proposal distribution $q\left(\theta^* \mid \theta^{(t-1)}\right)$;
2. Draw auxiliary samples $y^{(t-1)} = \left\{y_1^{(t-1)}, \ldots, y_K^{(t-1)}\right\}$ from $f\left(y^{(t-1)} \mid \theta^{(t-1)}\right)$ using either an MCMC algorithm or an exact algorithm. These auxiliary samples are simulated from the most recently updated likelihood function, and will be used to estimate the normalising constant;
3. Compute the MCMH acceptance ratio

(a) Estimate the normalising constant ratio $R\left(\theta^*, \theta^{(t-1)}\right) = \frac{\kappa(\theta^*)}{\kappa\left(\theta^{(t-1)}\right)}$ by

$$\hat{R}\left(\theta^{(t-1)}, y^{(t-1)}, \theta^*\right) = \frac{1}{K}\sum_{k=1}^{K}\frac{g\left(y_k^{(t-1)}, \theta^*\right)}{g\left(y_k^{(t-1)}, \theta^{(t-1)}\right)}.$$

Given the form of the likelihood function from which the auxiliary samples were simulated as given in (4.4), the above estimate is intuitively the ratio of the unknown normalising constant at the proposed state to the unknown normalising constant at the current state.

(b) Calculate

$$\begin{aligned}\hat{r} &= \hat{r}\left(\theta^{(t-1)}, y^{(t-1)}, \theta^*\right) \\ &= \frac{1}{\hat{R}\left(\theta^{(t-1)}, y^{(t-1)}, \theta^*\right)}\frac{g\left(x, \theta^*\right)\pi\left(\theta^*\right)q\left(\theta^{(t-1)} \mid \theta^*\right)}{g\left(x, \theta^{(t-1)}\right)\pi\left(\theta^{(t-1)}\right)q\left(\theta^* \mid \theta^{(t-1)}\right)} \\ &= \frac{1}{\text{“constant”}}\frac{\hat{f}\left(\theta^*\right)q\left(\theta^{(t-1)} \mid \theta^*\right)}{\hat{f}\left(\theta^{(t-1)}\right)q\left(\theta^* \mid \theta^{(t-1)}\right)}\end{aligned} \tag{4.6}$$

and estimate the acceptance ratio using $\tilde{\alpha} = \alpha\left(\theta^{(t-1)}, y^{(t-1)}, \theta^*\right) = \min\left\{1, \hat{r}\right\}$;

4. Set $\theta^{(t)} = \theta^*$ with probability $\tilde{\alpha}$ and set $\theta^{(t)} = \theta^{(t-1)}$ with probability $1 - \tilde{\alpha}$.

Note the similarity between the MH acceptance ratio $\alpha = \min\left\{1, \frac{f(y)q\left(x^{(t-1)}|y\right)}{f\left(x^{(t-1)}\right)q\left(y|x^{(t-1)}\right)}\right\}$ and the MCMH acceptance ratio $\tilde{\alpha} = \min\left\{1, \frac{1}{\text{“constant”}}\frac{\hat{f}(\theta^*|x)q\left(\theta^{(t-1)}|\theta^*\right)}{\hat{f}\left(\theta^{(t-1)}|x\right)q\left(\theta^*|\theta^{(t-1)}\right)}\right\}$. In the latter, the “constant”, which is an estimate for $\frac{\kappa(\theta^*)}{\kappa\left(\theta^{(t-1)}\right)}$ can be distributed over both $\hat{f}\left(\theta^* \mid x\right)$ and $\hat{f}\left(\theta^{(t-1)} \mid x\right)$, the proportional target distributions defined at the proposed and current states in this case, and hence we have an acceptance probability with the same form as that of the MH algorithm. The reader is referred to [78, 79] for further discussion on the MCMH algorithm, including convergence results and variations to the algorithm.

4.3.5 Reversible-Jumps MCMC (RJ-MCMC)

In some statistical problems, the dimension of the unknown parameter(s) is not fixed. There are many examples of such studies, where broadly the problem relates to the selection between a set of models, with the parameter vector having a different dimension depending on the model chosen. Examples include, but are not limited to, Bayesian choice between models with different number of parameters, boundary detection, image segmentation and object recognition using a marked point process [47, 3, 104, 96]. This section provides a basic overview of the MCMC algorithm introduced by Green in 1995

[47] that jumps between parameter subspaces of differing dimensionality, making the algorithm very useful for tackling the above mentioned problem. Some preliminaries needed for the algorithm are:

- Suppose we have to make a model selection out of a collection of candidate models $\{T_k;\, k \in \mathcal{K}\}$, where model T_k has a vector $\theta_k = \{\theta_1, \ldots, \theta_{n_k}\}$ of unknown parameters, and n_k the number of parameters needed to specify model T_k;
- Each T_k has its own parameter space $\Theta_k \subseteq \mathbb{R}^{n_k}$;
- We wish to fit the candidate model T_k to observed data y;
- The full Bayesian model can be written as $f(k, \theta_k \mid y) \propto \pi(k)\, \pi(\theta_k \mid k)\, p(y \mid k, \theta_k)$, where $\pi(k)$ is the prior probability imposed on model T_k, $\pi(\theta_k \mid k)$ is the prior specified for the parameter θ_k and $p(y \mid k, \theta_k)$ is the sampling model for the observed data y. Note that the sampling model is conditional on both k and θ_k;
- If we let $x = (k, \theta_k)$ and $\mathcal{X}^k = \mathcal{K} \times \Theta_k$ we can view the Markov Chain $\left\{x^{(t)}\right\}$ as jumping between models changing over the space $\mathcal{X} = \cup_{k \in \mathcal{K}} \mathcal{X}_k$;
- Let $x^{(t-1)} = \left(k^{(t-1)}, \theta_k^{(t-1)}\right)$ denote the current state, and $x^* = (k^*, \theta_k^*)$ the proposed state for $x^{(t)}$. Also denote the transition probability by $p\left(x^{(t-1)}, x^*\right)$.

The idea behind RJ-MCMC is to match dimensions with the resulting chain such that $f(k, \theta_k \mid y) \propto \pi(k)\, \pi(\theta_k \mid k)\, p(y \mid k, \theta_k)$ is preserved as the invariant distribution. In other words, the chain must satisfy the detailed balance condition, as defined in (4.2):

$$f(k, \theta_k \mid y)\, p\left(x^{(t-1)}, x^*\right) = f(y \mid k, \theta_k)\, p\left(x^*, x^{(t-1)}\right).$$

An intuitive explanation of what the RJ-MCMC algorithm attempts to do is as follows:

- We wish to be able to move between model choices of (potentially) different dimension;
- In order to do so, we must ensure that we are able to move from the current state to the next state, and back again. We therefore require auxiliary random variables generated from a proposed distribution that will not lead to slow convergence or low acceptance probabilities. The s such variables are generated for the forward move, with s^* variables generated for the reverse move, with the specific condition that the dimensions of the proposed state and the current state will be equal if these auxiliary variables are added into the mix. That is $s + n_{k^{(t-1)}} = s^* + n_{k^*}$, with $n_{k^{(t-1)}}$ the number of parameters needed to specify model $T_{k^{(t-1)}}$ and n_{k^*} the number of parameters needed to specify model T_{k^*}, the proposed model;

- The new state of the chain is then proposed using a bijection T^*. Along with the dimension matching properties of the auxiliary variables this proposal is reversible;

- The new state is accepted with probability $\alpha = \min\{1, r\}$,with an adjustment made to the ratio r to allow for the dimension matching condition and bijection.

The RJ-MCMC algorithm:

1. Generate model T_{k^*} from the proposal distribution;

2. Generate s random variables $u_1, \ldots, u_s \sim \psi_{k^{(t-1)} \to k^*}$, where $\psi_{k^{(t-1)} \to k^*}$ is a known proposed density, and the subscript $k^{(t-1)} \to k^*$ indicates moving from the current state $k^{(t-1)}$ to the proposed state k^*. The subscript is required to differentiate between the different proposal distributions at the various update times, as the proposal distributions may differ in dimensionality. Note that there is no theoretical restriction on the form of the proposed density, but care has to be taken that the choice of the proposed density does not lead to moves that cause a slow conversion or lead to low acceptance probabilities [39]. The choice of $\psi_{k^{(t-1)} \to k^*}$ is therefore dependent on the application at hand, or may simply be chosen as $UNI(0, 1)$ distribution;

3. Construct the proposed new state θ_k^* as $(\theta_k^*, u^*) = T^*\left(\theta_k^{(t-1)}, u\right)$, where

   (a) T^* (not related to T_{k^*}) denotes some deterministic bijection[4]. The choice of T^* will dictate in which manner the parameters are updated in the RJ-MCMC algorithm, and this 'mapping' is often chosen to be some type of move with a parameter update. The choice of bijection T^* is however dependent on the application at hand; and in this chapter the choice is restricted to swap, birth-death and split-merge moves. The moves used in this analysis, i.e. choices for T^*, are described in detail in Section 4.5.2,

   (b) $u^* = (u_1, \ldots, u_{s^*})$ are the s^* random numbers from a known joint density $\psi_{k^* \to k^{(t-1)}}$ that are required for the reverse move from θ_k^* to $\theta_k^{(t-1)}$, using the inverse function of T^*,

   (c) s and s^* satisfy the dimension-matching condition $s + n_{k^{(t-1)}} = s^* + n_{k^*}$,

   (d) If $n_{k^{(t-1)}} \leq n_{k^*}$ then $s^* = 0$, and if $n_{k^{(t-1)}} > n_{k^*}$ then $s = 0$;

[4] A bijection is a function that is both one-to-one and onto [4], in other words, if T^* is a bijection defined on a set A and takes values in set B, then for each $a \in A$ and $b \in B$, if $a = b$ then $T^*(a) = T^*(b)$ (one-to-one [4]) and for any $b \in B$, there exists an $a \in A$ such that $b = T^*(a)$ (onto [4]) .

4. Compute the RJ-MCMC acceptance ratio as

$$r = \frac{f\left(k^*, \theta_k^* \mid y\right) q\left(k^*, k^{(t-1)}\right) \psi_{k^* \to k^{(t-1)}}\left(u^*\right)}{f\left(k^{(t-1)}, \theta_k^{(t-1)} \mid y\right) q\left(k^{(t-1)}, k^*\right) \psi_{k^{(t-1)} \to k^*}\left(u\right)} \left| \frac{\partial\left(\theta_k^*, u^*\right)}{\partial\left(\theta_k^{(t-1)}, u\right)} \right|,$$

where $\left| \frac{\partial\left(\theta_k^*, u^*\right)}{\partial\left(\theta_k^{(t-1)}, u\right)} \right|$ is the Jacobian of the transformation T^*. Note that the Jacobian can be reduced to 1 if T^* is chosen to be the identity transformation. This occurs in instances where samples in the new space $\mathcal{X}^{k^*}$ are proposed directly, for example a birth-death or split-merge move [79];

5. Set $x^{(t)} = x^* = (k^*, \theta_k^*)$ with probability $\alpha = \min\{1, r\}$ and set $x^{(t)} = x^{(t-1)}$ with probability $1 - \alpha$. Note the similarity between the RJ-MCMC acceptance ratio and the MH acceptance ratio defined in Section 4.3.2, where an adjustment is made to the acceptance ratio to allow for the dimension matching condition and bijection. That is the addition of $\frac{\psi_{k^* \to k^{(t-1)}}(u^*)}{\psi_{k^{(t-1)} \to k^*}(u)} \left| \frac{\partial\left(\theta_k^*, u^*\right)}{\partial\left(\theta_k^{(t-1)}, u\right)} \right|$ in the ratio r.

The RJ-MCMC algorithm is therefore a special case of the MH algorithm described in Section 4.3.2, with the difference being that the proposal distribution includes auxiliary variables to enable dimension matching.

4.4 Occlusion algorithm: Prior and Likelihood specification

Recall from Section 4.1 that the procedure proposed in [70] can be described as a multilevel procedure that can be used to classify occluded objects in an image. In the following sections we elaborate on the steps given in Figures 4.1 and 4.2.

4.4.1 Object specification

This section provides more information on step 'Main 1', as described in Figure 4.1. Before the complete prior distribution can be specified, it is necessary to know how the nanoparticles are treated in the procedure. Konomi *et al.* in 2013 [70], along the same lines as [8, 104, 83], make use of objects with parametrised shapes to analyse the image shapes. This is done by specifying

1. Shape template;
2. Shift, scale and rotation parameters, and;
3. Object multiplicity.

A **shape template**, T, is used to model the object shape by using a set of parameters, $g_T^0 = \{g_T^0(1), \ldots, g_T^0(q)\}$, where q is the number of parameters needed to specify the object. The g_T^0 parameters are distinguished into random parameters g^r, and constant parameters g^{co}. That is, $g_T^0 = \{g^r, g^{co}\}$. It is important to note that the parameters are chosen in such a manner that the template will have an area equal to that of the unit circle (π square units), since any template, once defined, can be shifted, scaled and rotated and still belong to the same family of shapes. Konomi *et al* [70] also specify landmarks $l^0 = \{l^0(1), \ldots, l^0(M)\}$, as the M equally spaced boundary points of a given template. For completeness, note that the superscript 0 in the notation is used to indicate that the parameters of interest are related to the shape of the object only - and not the shifted, rotated, scaled object. As noted in [70], these landmarks can be determined if one knows the g_T^0 parameters, and is represented in polar coordinates as $l^0(k) = c^0 + S^0(k) \begin{bmatrix} cos(\theta(k)) \\ sin(\theta(k)) \end{bmatrix}$, where $S^0(k)$ is the distance of the k^{th} landmark from the center c^0 and $\theta(k)$ is the rotation with respect to the baseline of the k^{th} landmark.

After the template is defined (i.e. the shape of the object is defined), the shift, scale and rotation parameters need to be defined in order to represent the actual object. A shape with **shift** $c = (c_x, c_y)$, **scale** s and **rotation** θ is given by the landmarks $l = \{l(1), \ldots, l(M)\}$, with each landmark having polar coordinates

$$l(k) = c + c^0 + sS^0(k) \begin{bmatrix} cos(\theta(k) + \theta) \\ sin(\theta(k) + \theta) \end{bmatrix}.$$

For the purposes of the analysis in [70], $M = 90$ landmark points are used for each of the shapes. That is, for each shape 90 boundary points are used to represent the shape in the image. Note the similarity in this specification to that of the unshifted, unscaled, unrotated objects with landmarks $l^0(k)$. As noted in [83], this specification gives a boundary outline of the object, represented by a piecewise linear path connecting the M vertices. The use of landmarks to represent the shape of the object is a coding choice - that is, the choice is made based on how the algorithm will be coded - and for the purposes of the application in this chapter, the simulated shape rather than a landmark representation is used to represent the shape in the image.

An image consists of **multiple objects**, each with a potentially different shape. As mentioned earlier, the number of objects are assumed unknown, and modelled through a Markov Point Process (MPP). For the application at hand the process used is that of the Area Interaction Process Prior (AIPP) as defined in [8] and [83]. This is discussed in more detail in section 4.4.3.

4.4.2 Templates used

Konomi *et al.* [70] consider a fixed number of different shapes: *circle, ellipse, square, rectangle, triangle.* This was done using knowledge gained from a prior study done by Huitink *et al.* [57], where it was found that gold nanoparicles have the aforementioned possible shapes [70]. As the focus of this study is restricted to gold nanoparticles, with the additional advantage that the nanoscale images have been specifically prepared for this study, the shapes used in the algorithm for the current study will be restricted to that of circles and ellipses.

4.4.2.1 Template specification for a circle

A circle is specified by

$$(u - c_x)^2 + (v - c_y)^2 = s^2,$$

where $(c_x, c_y) = (0, 0)$ is the center and $s = 1$ is the radius. The constant parameters for a circle is therefore given by $g^0(1) = c_x = 0$; $g^0(2) = c_y = 0$; $g^0(3) = s = 1$, that is

$$\begin{aligned} g^{co} &= \left\{g^0(1), g^0(2), g^0(3)\right\} \\ &= \{0, 0, 1\} \end{aligned}$$

with no g^r parameters, so that $g_T^0 = \{g^r, g^{co}\} = \{0, 0, 1\}$. For the landmarks we have that $l^0(k) = c^0 + S^0(k) \begin{bmatrix} cos(\theta(k)) \\ sin(\theta(k)) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} + 1 \times \begin{bmatrix} cos(\theta(k)) \\ sin(\theta(k)) \end{bmatrix}$. For $\theta(k)$ we follow the same specification as [83] and let $\theta(k) = 2\pi(k-1)/n$, for $k = 1, \ldots, 90$.

4.4.2.2 Template specification for an ellipse

An ellipse is specified by

$$\left[\frac{1}{E_1}\left((u - c_x)\cos(\theta) - (v - c_y)\sin(\theta)\right)\right]^2 + \left[\frac{1}{E_2}\left((u - c_x)\sin(\theta) - (v - c_y)\cos(\theta)\right)\right]^2 = 1,$$

where $(c_x, c_y) = (0, 0)$ is the center, $\theta = 0$ is the object rotation, so that the major axis is on the $x-$axis, $E_1 \in (1.12, 1.4)$ the largest distance and $E_2 = \frac{1}{E_1}$ the shortest. The range for E_1, as noted in [70], is chosen specifically to distinguish the ellipse from the circle templates, since values of E_1 and E_2 close to 1 imply that the ellipse is closely related to a circle. The random specification of E_1 allow us to deform the pure shape to allow for differently shaped ellipses to be specified, with the threshold values and E_2 chosen such that the area of the ellipse is π [70]. The parameters for an ellipse is therefore given by $g^0(1) = c_x = 0$; $g^0(2) = c_y = 0$; $g^0(3) = E_1 = g^r$; $g^0(4) = E_2$. In

other words

$$\begin{aligned} g^{co} &= \left\{g^0(1), g^0(2), g^0(4)\right\} \\ &= \{0, 0, E_2\} \end{aligned}$$

and

$$\begin{aligned} g^r &= \left\{g^0(3)\right\} \\ &= \{E_1\}. \end{aligned}$$

We therefore have that $g_T^0 = \{g^r, g^{co}\} = \left\{E_1, 0, 0, \frac{1}{E_1}\right\}$. If a landmark representation is used, then $l^0(k) = c^0 + S^0(k) \begin{bmatrix} cos(\theta(k)) \\ sin(\theta(k)) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} + S^0(k) \cdot \begin{bmatrix} cos(\theta(k)) \\ sin(\theta(k)) \end{bmatrix}$. The reader is referred to [71][5] for a detailed discussion on the other shape templates.

4.4.3 Area Interaction Process Prior (AIPP) specification

In this section we discuss step 'Main 2' as highlighted in Figure 4.1. Recall the definition of a MPP and the AIPP from Section 4.2, where the location parameters in the MPP is represented by $c = \{c_1, \ldots, c_m\}$, with m the number of objects in the image. Note that m should not be confused with M, the number of boundary points used for each of the landmarks. These parameters, along with the points in the MPP representation are then modelled using the AIPP [70]:

$$\pi(c, m \mid g^r, s, \theta, T, \gamma_1, \gamma_2) = \frac{1}{A^*} \exp\left(-\gamma_1 m - \gamma_2 S(\eta)\right) = \frac{1}{A^*} \exp\left(-\gamma_1 m\right) \exp\left(-\gamma_2 S(\eta)\right),$$

with

- $S(\eta)$ is the silhouette formed by taking the union of the objects in the specification. That is, $S(\eta)$ is the set of pixels in the image that are occupied by an object;
- $\eta_i = \{c_i, g_i^r, s_i, \theta_i, T_i\}$ the collection of parameters that represents the i^{th} object, and $\eta = \{\eta_i\}_{i=1}^m = \{c_i, g_i^r, s_i, \theta_i, T_i\}_{i=1}^m = \{c, g^r, s, \theta, T\}$ the set of parameters for all the objects. Note that c_i represents the shift, g_i^r the random parameters, s_i the scale, θ_i the rotation and T_i the template for the i^{th} object;
- $\gamma = \{\gamma_1, \gamma_2\}$ a set of positive unknown parameters controlling the potential for the presence of each object (γ_1) and the overlap potential between neighboring

[5]This supplementary material was obtained online on 2015/01/27 at http://projecteuclid.org

objects (γ_2) in the image [83]. Note that $\gamma_2 = 0$ does not penalise overlapping, with $\gamma_2 = \infty$ not allowing any overlapping;

- A^* an unknown normalising constant that depends on all the parameters above, that is

$$A^* = A^*(\eta, m, \gamma) = A^*(c, g^r, s, \theta, T, m, \gamma_1, \gamma_2).$$

In terms of the notation used in Section 4.2, where $p(c) = \alpha\beta^{n(c)}\gamma^{|S(c)|}$, we have the following:

- $c = \{c_1, \ldots, c_m\}$ is the object configuration (object locations);
- $\alpha = \frac{1}{A^*}$, the unknown normalising constant;
- $\beta = e^{-\gamma_1}$, with γ_1 the parameter controlling the presence of objects in the image;
- $n(c) = m$, with m the number of objects;
- $\gamma = e^{-\gamma_2}$, with γ_2 the parameter controlling the overlap between neigbouring objects.

Note that the AIPP is defined conditional on $g^r, s, \theta, T, \gamma_1, \gamma_2$ (i.e. these parameters are assumed to be known).

4.4.4 Likelihood specification

As discussed in Chapter 3, the mean intensity of the background is greater than the mean intensity of the region of pixels occupied by an object in the TEM image. Let $\mu = (\mu_0, \ldots, \mu_m)$ be the mean vector and $\sigma^2 = \sigma_m^2 = (\sigma_0^2, \ldots, \sigma_m^2)$ the variance vector for the background and objects intensity. In this representation μ_0 is the mean background intensity, and σ_0^2 the variance of the intensities of the background pixels, with μ_i and σ_i^2, $i = 1, \ldots, m$, the mean pixel intensity and variance of the pixel intensities of the region of pixels that make up the i^{th} object. In addition, let

$$\Theta = \left\{\eta, m, \mu, \sigma^2\right\} = \left\{c, g^r, s, \theta, T, m, \mu, \sigma^2\right\},$$

the set of parameters of all m objects. The likelihood function of the observed values $y = \{x_p, y_p\}_{p=1}^N$, where N is the number of pixels in the image, given the object parameters Θ is then specified using an independent Gaussian model, with [70]

$$f(y|\Theta, \gamma) \propto \prod_{p=1}^{N} \exp\left\{-\frac{1}{2\phi(x_p)}(y_p - \delta(x_p))^2\right\}, \text{ where} \tag{4.7}$$

- x_p is the p^{th} pixel position, i.e. the image domain;

- y_p is the intensity of the p^{th} pixel;
- $\delta(x_p)$ is the mean intensity of the object that covers the neighbourhood of the p^{th} pixel;
- $\phi(x_p)$ is the variance of the pixel intensities of the object that covers the neighbourhood of the p^{th} pixel.

In other words, if the p^{th} pixel is covered by object $i \in (1, m)$, where object i contains, say, n_i pixels, then $\delta(x_p) = \frac{1}{n_i}\Sigma_{k=1}^{n_i} y_k$, where y_k, $k = 1, \ldots, n_i$ are the intensities of the n_i pixels that make up the i^{th} object. Similarly $\phi(x_p)$ is the variance of the intensities of the n_i pixels that make up the i^{th} object. If the p^{th} pixel is not covered by any object (in other words form part of the background pixels) then $\delta(x_p)$ and $\phi(x_p)$ is the mean background intensity and the variance of the intensities of the background pixels, respectively. When a pixel is covered by more than one object (in other words covered by an occluded object), the minimum mean intensity of the objects is used for $\delta(x_p)$ with $\phi(x_p)$ the corresponding variance of that object [70].

Also note that whenever an object in the image is updated, through either adding or removing pixels from the object, the mean intensity and corresponding variance of that object will change, and therefore so will the likelihood calculated using (4.7).

4.4.5 Hierarchical Prior Specification

This section provides a discussion on step 'Main 3' as given in Figure 4.1. The following assumptions are made in the prior specification:

A1: the mean and variance pairs (μ_i, σ_i^2) of the background and the objects intensity are assumed independent;

A2: all the object parameters $\eta = \{c, g^r, s, \theta, T\}$, except the location parameters (c), are assumed independent from object to object;

A3: the scale (s), shift (c), rotation (θ) and template (T) parameters are assumed independent of the other parameters $\{g^r, m, \mu, \sigma^2\}$;

A4: the random template parameters, g_i^r, are assumed to be closely related to the template T_i, since the g_i^r are different from template to template.

Given these assumptions, the prior is elicited hierarchically as follows:

$$\begin{aligned}\pi(\Theta, \gamma) &= \pi(\Theta \mid \gamma)\,\pi(\gamma) \\ &= \pi\left(\mu, \sigma^2, \eta, m \mid \gamma\right)\pi(\gamma) \\ &= \pi\left(\mu, \sigma^2\right)\pi(\eta, m \mid \gamma)\,\pi(\gamma),\end{aligned}$$

since the mean and variance pairs do not depend on γ and μ,σ are independent from η, m,

$$\begin{aligned}\pi(\Theta,\gamma) &= \pi\left(\mu,\sigma^2\right)\pi\left(c,g^r,s,\theta,T,m \mid \gamma\right)\pi(\gamma)\\ &= \pi\left(\mu,\sigma^2\right)\pi\left(c,m \mid \gamma,g^r,s,\theta,T\right)\pi\left(g^r,s,\theta,T\right)\pi(\gamma).\end{aligned} \tag{4.8}$$

Prior for the mean and variances: $\pi\left(\mu,\sigma^2\right)$

Based on the assumptions $1-4$ above, with an uninformative prior assigned, the prior for the mean and variances of the background and the object intensities is given by

$$\pi\left(\mu,\sigma^2\right) = \prod_{i=0}^{m}\pi\left(\mu_i,\sigma_i^2\right) \propto \prod_{i=0}^{m}\left(\sigma_i^2\right)^{-1}$$

This is obtained as follows: For ease of reference, denote by $\theta^* = \begin{pmatrix}\mu\\ \sigma^2\end{pmatrix}$, and assume a naive Jeffreys prior [60], given by

$$\pi\left(\mu,\sigma^2\right) = \pi\left(\theta^*\right) = \mid \mathrm{I}\left(\theta^*\right) \mid^{\frac{1}{2}}$$

with Fisher information $\mathrm{I}\left(\theta^*\right)_{i,j} = -\mathrm{E}_{\theta^*_{\mathrm{i,j}}}\left[\frac{\partial^2 \log\{f(\mathrm{X}|\theta^*)\}}{\partial\theta^*_{\mathrm{i}}\partial\theta^*_{\mathrm{j}}}\right]$. For a Gaussian Normal distribution with both μ and σ^2 unknown, we have that

$$\mathrm{I}\left(\theta^*\right) = -\mathrm{E}_{\theta^*}\begin{pmatrix}-\frac{1}{\sigma^2} & -\frac{(\mathrm{X}-\mu)}{(\sigma^2)^2}\\ -\frac{(\mathrm{X}-\mu)}{(\sigma^2)^2} & \frac{1}{2(\sigma^2)^2}-\frac{(X-\mu)^2}{(\sigma^2)^3}\end{pmatrix} = \begin{pmatrix}\frac{1}{\sigma^2} & 0\\ 0 & \frac{1}{2(\sigma^2)^2}\end{pmatrix},$$

since $\mathrm{E}\left(\mathrm{X}-\mu\right)=0$ and $\mathrm{E}\left(\mathrm{X}-\mu\right)^2=\sigma^2$. Therefore,

$$\pi\left(\mu,\sigma^2\right) = \pi\left(\theta^*\right) = \mid \mathrm{I}\left(\theta^*\right) \mid^{\frac{1}{2}} = \left(\frac{1}{2\left(\sigma^2\right)^3}\right)^{\frac{1}{2}} \propto \frac{1}{\sigma^2}.$$

Prior for the location and number of objects: $\pi\left(c,m \mid \gamma,g^r,s,\theta,T\right)$

Recall from Section 4.4.3 that

$$\pi\left(c,m \mid \gamma,g^r,s,\theta,T\right) = \frac{1}{A^*}\exp\left(-\gamma_1 m-\gamma_2 S\left(\eta\right)\right) = \frac{1}{A^*}\pi^*\left(c,m \mid \gamma,g^r,s,\theta,T\right). \tag{4.9}$$

For the model at hand, this conditioning is necessary for the parameters that specify the objects in the object configuration $c=\{c_1,\ldots,c_m\}$, since we need to know the AIPP parameters γ, the random object parameters for object deformation g^r, the scale parameters for each object s, the rotation of each object θ, and the template used for each object T before the object configuration can be modelled.

Prior for the object parameters (excl. the location parameters): $\pi\left(g^r, s, \theta, T\right)$

Using assumptions **A1 - A4** above,

$$\pi\left(g^r, s, \theta, T\right) = \prod_{i=1}^{m} \pi\left(s_i\right) \pi\left(\theta_i\right) \pi\left(g_i^r \mid T_i\right) \pi\left(T_i\right).$$

The scale parameters s_i are assigned a uniform prior proportional to the size of the image, S_{max} (the number of pixels in the image), that is

$$\pi\left(s_i\right) \sim UNI\left(0, S_{max}\right).$$

Unlike in [70], where the rotation parameters θ_i are assigned a prior that favours values close to 0 and π,

$$\pi\left(\theta_i\right) \sim \left\{\mid cos\left(\theta\right) \mid + \pi^{-1}\right\}/3, \tag{4.10}$$

with $\theta \in (0, \pi)$, we assign an alternative prior, $\pi\left(\theta_i\right) \sim UNI\left(0, \pi\right)$. The motivation for this comes from the fact that the nanoparticles considered in this study are either circles or ellipses, and we assume that information favouring certain rotation values of an ellipse above others is not known beforehand (i.e. we assume that all rotations values are equally likely), and logically, a rotation of a circle is an identity transformation. The template number T_i is assigned a discrete uniform prior,

$$T_i \sim Discrete\ UNI\left(1, T_{max}\right),$$

where T_{max} is the number of templates specified. Those templates that contain at least one random parameter (i.e. all the templates considered apart from the circle and square) all have one basic characteristic, where the random parameter is constrained to take values between (a, b). An altered location and scale beta distribution is therefore assigned as prior for the random parameters,

$$\pi\left(g_i^r\right) = \frac{1}{\text{BETA}\left(\alpha, \beta\right)} \frac{\left(g_i^r - a\right)^{\alpha} \left(b - g_i^r\right)^{\beta - 1}}{\left(b - a\right)^{\alpha + \beta - 1}},$$

with $\text{BETA}\left(\alpha, \beta\right) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)} = \frac{(\alpha-1)!(\beta-1)!}{(\alpha+\beta-1)!}$, and α and β chosen randomly from $UNI\left(0, 1\right)$. For the implementation in this chapter, since only circles and ellipses are considered in the implementation (due to the way the nanoparticle samples were prepared), a uniform prior is assigned when the object is an ellipse, $\pi\left(g_i^r\right) \sim UNI(1.12, 1.4)$. These parameters are chosen based on the template specification, as discussed in Section 4.4.2.2.

Prior for $\gamma = (\gamma_1, \gamma_2)$: $\pi(\gamma)$

For the object multiplicity and object overlap parameters, $\gamma = (\gamma_1, \gamma_2)$ Konomi *et al* [70] propose independent log-normal priors with parameters that determine a mean close to 100 and large variance,

$$\begin{aligned} \gamma_1 &\sim LN(\alpha_1, \delta_1), \\ \gamma_2 &\sim LN(\alpha_2, \delta_2). \end{aligned}$$

A large variance is chosen in order to obtain a higher spread in the priors due to lack of knowledge of the true parameter. As noted in [83], the quality of image reconstructions is robust when using a wide range of values for γ_1 and γ_2. Due to this, and since our application in the same manner as [70] apply this algorithm to TEM images of gold nanoparticles, we use the same specification for γ_1 and γ_2 in this study. Note that we will use $\alpha_1 = \alpha_2 = 4$ and $\delta_1 \sim UNI(1, 1.5)$ and $\delta_2 \sim UNI(1, 1.5)$, since these parameters determine a mean close to 100 with large variance for both priors.

4.5 Occlusion algorithm: Posterior distribution

Step 'Main 4' as given in Figures 4.1 and 4.2 is described in this section. Recall from (4.8) and (4.9) that

$$\begin{aligned} \pi(\Theta, \gamma) &= \pi\left(\mu, \sigma^2\right) \pi(c, m \mid \gamma, g^r, s, \theta, T) \pi(g^r, s, \theta, T) \pi(\gamma) \\ &= \frac{1}{A^*} \pi\left(\mu, \sigma^2\right) \pi^*(c, m \mid \gamma, g^r, s, \theta, T) \pi(g^r, s, \theta, T) \pi(\gamma), \end{aligned}$$

and $\Theta = \{\eta, m, \mu, \sigma^2\} = \{c, g^r, s, \theta, T, m, \mu, \sigma^2\}$, with $\eta = \{\eta_i\}_{i=1}^m = \{c_i, g_i^r, s_i, \theta_i, T_i\}_{i=1}^m$ $= \{c, g^r, s, \theta, T\}$ the set of parameters for all the objects, m the number of objects, $\{\mu, \sigma^2\}$ the mean and variance vectors for the background and objects intensity and A^* the unknown normalising constant that depends on all the parameters. The posterior can therefore be represented as follows:

$$\begin{aligned} p(\Theta, \gamma \mid y) &\propto \pi(\Theta, \gamma) f(y \mid \Theta, \gamma) \\ &= \frac{1}{A^*} \pi\left(\mu, \sigma^2\right) \pi^*(c, m \mid \gamma, g^r, s, \theta, T) \pi(g^r, s, \theta, T) \pi(\gamma) f(y \mid \Theta, \gamma) \\ &= \frac{1}{A^*} p^*\left(\eta, \mu, \sigma^2, m, \gamma \mid y\right), \end{aligned} \tag{4.11}$$

with $f(y \mid \Theta, \gamma)$ as defined in (4.7). In this specification, since the normalising constant in unknown due to the dependency on the unknown object parameters (see Section 4.4.3 for more details), and also since it is infeasible to obtain an analytical expression for

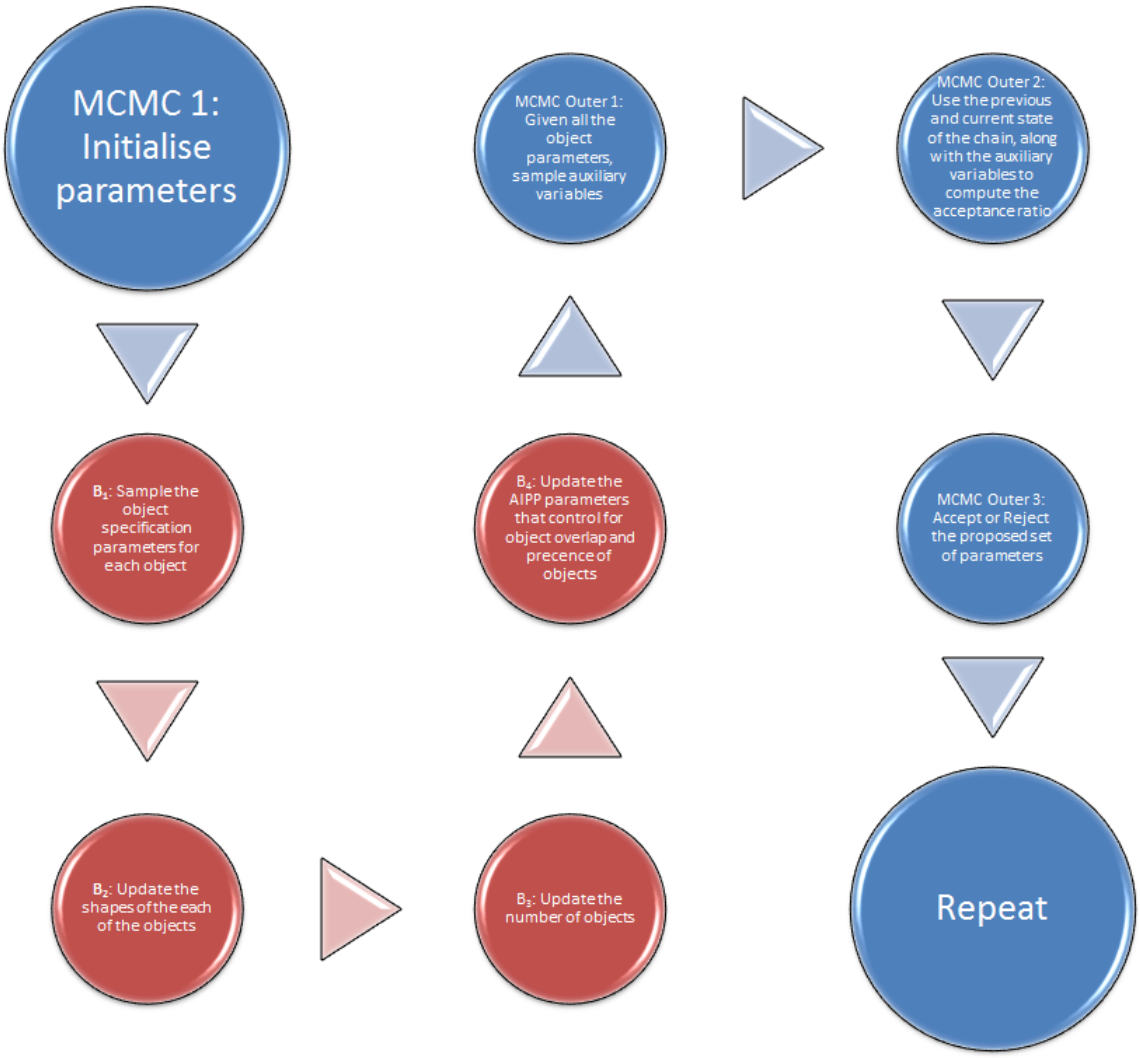


Figure 4.3: MCMC diagram as given in Figure 4.2, with the 'MCMC Inner' steps highlighted for clarity. As noted in Figure 4.2, these steps can be repeated multiple times in a single iteration of the 'MCMC Outer' steps, as these steps deal with the updating of the object parameters.

the posterior, an MCMC approach is proposed to carry out the inference. The MCMC approach suggested by [70] can be described as a 2−stage MH algorithm, containing an inner and outer MH algorithm, as indicated in Figure 4.2. In the inner algorithm, as highlighted in Figure 4.3, the parameters are sampled from $p^*(\eta, \mu, \sigma^2, m, \gamma \mid y)$, and in the outer algorithm, as given in Figure 4.4, the unknown normalising constant A^* is accounted for. The algorithm used in the outer algorithm is the Monte Carlo Metropolis-Hastings algorithm, with a combination of a Metropolis-Hastings-within-Gibbs, RJ-MCMC and Metropolis-Hastings algorithm used for the inner algorithm. This remainder of this section will be set out as follows:

- A high level description of the algorithm proposed by [70] is provided;

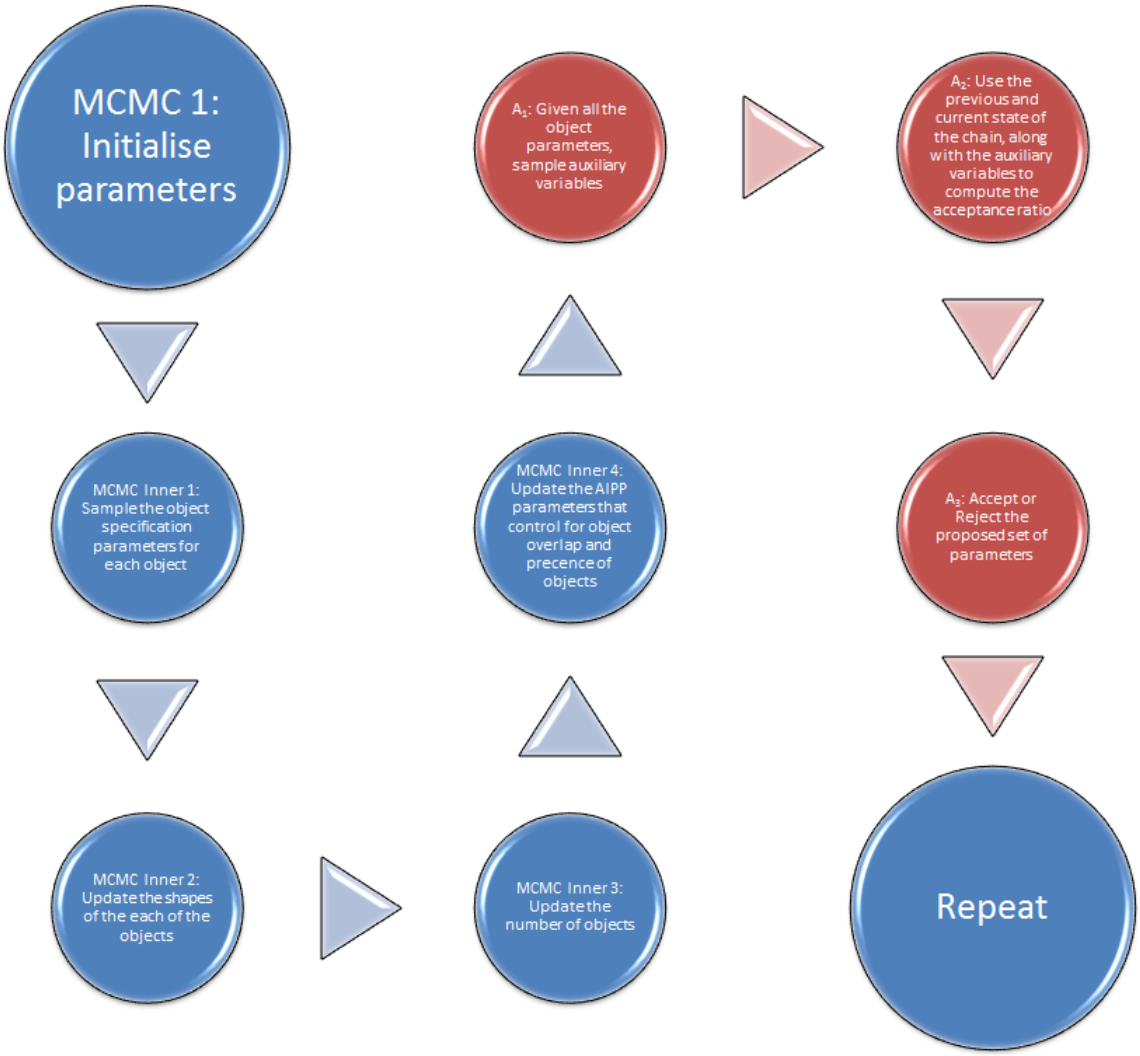


Figure 4.4: MCMC diagram, with the 'MCMC Outer' steps highlighted for clarity. These steps deal with the updating of the unknown normalising constant, given the updated parameters obtained in 'MCMC Inner'.

- A detailed discussion is provided for both the inner (where the parameters are sampled) and outer algorithms (where the unknown normalising constant is approximated);
- Matlab® code for implementation is provided in Appendix B.

For ease of reference the algorithm will be labeled as follows: $\mathbf{A}_i$ will be used to refer to the i^{th} step in the outer (MCMC) algorithm, and $\mathbf{B}_j$ to refer to the j^{th} step in the inner algorithm.

4.5.1 High level description of the algorithm

As indicated in Figure 4.3, the inner algorithm, used to sample all of the unknown parameters, can be outlined as follows:

$\mathbf{B_1}$: Use a Metropolis-Hastings-within-Gibbs algorithm to sample the parameters μ^*, $(\sigma^*)^2$ and η^*, excluding the template parameter T^*;

$\mathbf{B_2}$: Use a RJ-MCMC algorithm with a swap move to sample the template parameter T^*;

$\mathbf{B_3}$: Use a RJ-MCMC algorithm with a birth-death and split-merge move to sample m^*;

$\mathbf{B_4}$: Use a Metropolis-Hastings algorithm to sample the parameter γ^*.

The outer algorithm (see Figure 4.4), are the same steps as specified for the MCMH algorithm described in Section 4.3.4, which can be summarised as follows:

1. A proposed value of the parameter(s) is drawn using a standard algorithm;
2. Auxiliary samples needed to estimate the normalising constant are drawn;
3. The unknown normalising constant ratio $\hat{R}$ (step 3 (a) in Section 4.3.4) is estimated;
4. The MH-rejection ratio $\hat{r}$ is estimated, and the acceptance ratio $\tilde{\alpha}$ (step 3 (b) in Section 4.3.4) is computed;
5. The proposed value of the parameter(s) is either accepted or rejected, using $\tilde{\alpha}$.

The outer algorithm is as follows:

$\mathbf{A_1}$: Given the current state $\Theta^{(t\text{-}1)}$, $\gamma^{(t\text{-}1)}$, draw Θ^*, γ^* from $q(\cdot) = p^*(\cdot)$ using the steps outlined in B_1-B_4, where $p^*(\cdot)$ is the pseudo-posterior distribution;

$\mathbf{A_2}$: Given all the parameters, simulate auxiliary variables $z_1,\dots z_K$ from $f(z;\Theta^*)$. Simulating the auxiliary variables simply requires us to sample from a normal distribution with parameters defined at the proposed state of the algorithm. Recall from Section 4.3.4 that these auxiliary variables are used to estimate the normalising constant;

$\mathbf{A_3}$: Estimate the normalising constant ratio as $\hat{R} = \frac{1}{K}\sum_{i=1}^{K}\frac{f(z;\Theta^*)}{f(z;\Theta^k)}$;

$\mathbf{A_4}$: Estimate the MH-rejection ratio as $\hat{r} = \frac{1}{\hat{R}}$. Note that the remainder of the fraction in (4.6) cancels out since $q(\cdot) = p^*(\cdot)$;

$\mathbf{A_5}$: Accept Θ^*, γ^* with probability $\tilde{\alpha} = \min\{1;\hat{r}\}$.

4.5.2 Detailed description of the algorithm

This section provides a detailed discussion of the sampling steps in the algorithm outlined in $\mathbf{B}_1-\mathbf{B}_4$ above, or alternatively steps Inner 1 - Inner 4 as given in Figure 4.3. All of these steps describe how to sample from the pseudo-posterior $p^*(\eta,\mu,\sigma^2,m,\gamma \mid y)$, by alternating between draws from the conditional priors

- $p^*(\mu,\sigma^2,\eta \mid y,m,\gamma)$ (Steps $\mathbf{B}_1$ and $\mathbf{B}_2$)
- $p^*(m \mid y,\mu,\sigma^2,\eta,\gamma)$ (Step $\mathbf{B}_3$)
- $p^*(\gamma \mid y,\mu,\sigma^2,\eta,m)$ (Step $\mathbf{B}_4$)

$\mathbf{B}_1$ - Updating μ and σ:

The conditional distribution of $p^*\left(\mu_j,\sigma_j^2 \mid \cdot\right)$, with $\cdot$ denoting the remainder of the parameters, is given by

$$p^*\left(\mu_j,\sigma_j^2 \mid \cdot\right) \propto \pi\left(\mu_j,\sigma_j^2\right) f\left(y \mid \Theta\right).$$

A Metropolis-Hastings step is used to draw from the posterior, using

$$q\left(\mu_j,\sigma_j^2\right) = \frac{1}{\sigma_j^2}^{N_j} \exp\left(-\frac{1}{2\sigma_j^2}\left[\left(N_j-1\right)s_j^2 + N\left(\bar{y}_{j\cdot}-\mu_j\right)^2\right]\right),$$

as proposal distribution, where

- $s_j^2 = \frac{1}{N_j-1}\sum_{i=1}^{N_j}(y_i-\bar{y}_j)$,
- N_j is the total number of pixels in the region of the proposed shape,
- $\bar{y}_{j\cdot}$ is the sample mean intensity of the j^{th} object,

- N is the total number of pixels in the image.

To draw from this proposal, samples are drawn sequentially from

- $\sigma_j^2 \mid \cdot \equiv \text{Inv}\chi^2\left(N_j - 1, \sigma^2\right)$, with $\text{Inv}\chi^2\left(\cdot\right)$ denoting the Inverse Chi-Square distribution, and then
- $\mu_j \mid \sigma_{j,\cdot}^2 \equiv N\left(\bar{y}_{j\cdot}, \frac{\sigma_j^2}{N_j}\right)$.

An Inverse Chi-Square distribution is chosen as proposal for the unknown variance as this distribution is commonly used as prior for a scale parameter such as unknown variance for a Gaussian distribution [75, 12]. Konomi *et al* [70] also provides the conditional distribution for the independent pixel case, i.e. when it is assumed that objects in neighbouring pixels are independent of one another.

$\mathbf{B}_1$ - Updating $\eta = \{c, g^r, s, \theta, T\}$ excluding T:

A Metropolis-Hastings step is used to draw proposals for the components of $\eta = \{c, g^r, s, \theta, T\}$ excluding T. The proposal distributions need to be well defined in order for the chain to have good properties. Konomi *et al* [70] suggests making use of pre-processing, and this is also the approach we follow here. Table 4.1 provides details on the proposal distributions used in the MH step within the Gibbs algorithm.

$\mathbf{B}_2$ - Updating T:

With shape selection we are faced with the problem of choosing templates $\{T_k;\, k \in \mathcal{K}\}$, where each template potentially has a different number of parameters that characterise the template. We therefore need to jump between different sub-spaces when moving from template to template, and as discussed in section 4.3.5 RJ-MCMC is useful in problems with dimensionality matching, and is the algorithm that will be used to sample the template T_j. As discussed in [70], the only parameter that will have different meaning from template to template is the random pure parameter g^r. To make it clear that the random pure parameters g^r have different meaning from template to template, two variables are introduced: $u_{T_j} = g^r_{T_j}$, the random pure parameter of the current template, and $v_{T_j} = g^r_{T_j}$ the random pure parameter of the proposed template. In addition we denote by

$$\phi_{T_j^{(t-1)}} = \left\{c_{T_j^{(t-1)}}, s_{T_j^{(t-1)}}, \theta_{T_j^{(t-1)}}, T_j^{(t-1)}, u_{T_j^{(t-1)}}\right\}$$

the current state of the chain, and by

$$\phi_{T_j^*} = \left\{c_{T_j^*}, s_{T_j^*}, \theta_{T_j^*}, T_j^*, u_{T_j^*}\right\}$$

Table 4.1: Proposal distributions for $\eta = (c, g^r, s, \theta, T)$ excluding T.

Component	Proposal	Discussion
Scaling parameter: s_j	$q\left(s_j^*, s_j^{(t-1)}\right) \equiv N\left(s_j^{(t-1)}, \sigma_{s_j}^2\right)$	$\sigma_{s_j}^2$ is derived from the estimated scale s_j^0 obtained during pre-processing. In this paper $\sigma_{s_j}^2 = \frac{1}{10} s_j^0$ is used. The reader is reminded that there is only one scaling parameter per object, and is chosen in such a way that different shapes with the same scaling parameter will have equal area. This is important when moving between templates in the RJ-MCMC algorithm.
Location parameter: c_j	$q\left(c_j^*, c_j^{(t-1)}\right) \equiv N\left(c_j^{(t-1)}, \sigma_{c_j}^2 \mathrm{I}\right)$	Given m number of objects, there are $2m$ location parameters, m in the $x-$coordinates, and m in the $y-$coordinates. In the proposal, $\sigma_{c_j}^2$ is the variance for the (x, y) coordinates.
Rotation parameter: θ_j	$q\left(\theta_j^*, \theta_j^{(t-1)}\right) \equiv UNI\,(0, \pi)$	Note that the circle template does not have a rotation parameter. The prior for the rotation parameter, given in (4.10) is different to the proposal distribution provided here. This is done because we expect the shapes to be rotated randomly throughout the image, and prior favouring values close to 0 or π will only add value to the algorithm if pre-processing reveals that this is the case for objects in the sample image.
Random pure parameter: g_j^r	$q\left(\left(g_j^r\right)^*, \left(g_j^r\right)^{(t-1)}\right) \equiv \pi\left(\left(g_j^r\right)^*\right)$	The sampling is done using an independence algorithm, with the prior distribution as proposal. 1. Generate $\left(g_j^r\right)^*$ from $q\left(\left(g_j^r\right)^{(t-1)}\right)$, 2. Compute the acceptance probability $\alpha = \frac{p^*\left(\left(g_j^r\right)^*, \mu^{(t)}, \left(\sigma^2\right)^{(t)}, s^{(t)}, c^{(t)}, \theta^{(t)}, T^{(t-1)}, \left(g_j^r\right)_{1:(j-1)}^{(t)}, \left(g_j^r\right)_{(j+1):m}^{(t-1)} \mid y\right) q\left(s_j^{(t-1)}, s_j^*\right)}{p^*\left(\left(g_j^r\right)^{(t-1)}, \mu^{(t)}, \left(\sigma^2\right)^{(t)}, s^{(t)}, c^{(t)}, \theta^{(t)}, T^{(t-1)}, \left(g_j^r\right)_{1:(j-1)}^{(t)}, \left(g_j^r\right)_{(j+1):m}^{(t-1)} \mid y\right) q\left(s_j^*, s_j^{(t-1)}\right)}$ 3. Set $\left(g_j^r\right)^{(t)} = \left(g_j^r\right)^*$ with probability $r = min\,\{1, \alpha\}$ and $\left(g_j^r\right)^{(t)} = \left(g_j^r\right)^{(t-1)}$ with the remainder.

the proposed state. The reason for the introduction of the new notation is to make it clear that the parameters are dependent on the template T_j^*.

The RJ-MCMC algorithm used to update the template is as follows:

1. Select model T_j^* with probability $q\left(T_j^{(t-1)} \mid T_j^*\right) = \pi\left(T_j^{(t-1)}\right)$;
2. If $T_j^* \neq T_j^{(t-1)}$, then $v_{T_j^{(t-1)}}$ is unknown. Therefore, generate $v_{T_j^{(t-1)}}$ from $\pi\left(v_{T_j}\right)$;
3. Consider the bijection

$$\left\{c_{T_j^*}, s_{T_j^*}, \theta_{T_j^*}, u_{T_j^*}, v_{T_j^*}\right\} = \left\{c_{T_j^{(t-1)}}, s_{T_j^{(t-1)}}, \theta_{T_j^{(t-1)}}, u_{T_j^{(t-1)}}, v_{T_j^{(t-1)}}\right\};$$

4. Compute the MH ratio as

$$r = \frac{p^*\left(c_{T_j^*}, s_{T_j^*}, \theta_{T_j^*}, v_{T_j^*} \mid y\right) \pi\left(T_j^{(t-1)}\right) \pi\left(u_{T_j^*}\right)}{p^*\left(c_{T_j^{(t-1)}}, s_{T_j^{(t-1)}}, \theta_{T_j^{(t-1)}}, u_{T_j^{(t-1)}} \mid y\right) \pi\left(T_j^*\right) \pi\left(v_{T_j^*}\right)} |J|,$$

   where $|J|$ is the Jacobian of the transformation T^*. Note that the Jacobian is reduced to 1 since the bijection is chosen to be the identity transformation;

5. Set $\phi_{T_j^{(t)}} = \phi_{T_j^*}$ with probability $\alpha = min\{1, r\}$ and set $\phi_{T_j^{(t)}} = \phi_{T_j^{(t-1)}}$ with probability $1 - \alpha$.

$\mathbf{B}_3$ - Updating m:

To update the number of objects in the image, m, two types of update moves are considered: birth-death and split-merge.

1. Birth-death: In the death step, a randomly chosen object is deleted, and in contrast, in the birth step, an object with parameters generated from the priors is added to the image;
2. Split-merge: Instead of generating a randomly chosen object as in the birth step, the split move uses an existing object and splits it into two new objects. With the merge move, two existing objects are merged into one.

Before a move is performed, we need to know which move will be proposed, and subsequently accepted/rejected. It is therefore required to specify probabilities of selecting each of the moves. Denote by $P(\text{birth})$, $P(\text{death})$, $P(\text{split})$, $P(\text{merge})$ the probabilities of proposing a birth, death, split or merge move. For the algorithm proposed the following probabilities are used:

$$\begin{aligned} P(\text{birth}) = P(\text{death}) = P(\text{split}) = P(\text{merge}) = 1/4, &\quad \text{for } m \geq 2; \\ P(\text{birth}) = P(\text{split}) = P(\text{death}) = 1/3;\ P(\text{merge}) = 0, &\quad \text{for } m = 1; \\ P(\text{birth}) = 1;\ P(\text{split}) = P(\text{death}) = P(\text{merge}) = 0, &\quad \text{for } m = 0. \end{aligned}$$

Birth-Death move:

The RJ-MCMC algorithm used in the **birth** step is as follows:

1. Given the current state $\phi^{(t-1)} = \{\eta_m, \mu_m, \sigma_m^2\}$, with $\eta_m, \mu_m, \sigma_m^2$ the current parameters for an image with m objects, select model "birth move" with probability $q(m \to (m+1)) = P(\text{birth})$, $m \to (m+1)$ denoting a move from m to $m+1$ objects;

2. Generate a new object η_{m+1} with a randomly assigned center. In this step the dimension of the parameters are increased by $Q_{m+1} = \dim\left(\eta_{m+1}, \mu_{m+1}, \sigma_{m+1}^2\right)$;

3. Sample all the parameters that describe the new object, $\left\{\eta_{m+1}, \mu_{m+1}, \sigma_{m+1}^2\right\}$ from the prior distributions of the parameters and set $\phi^* = \left\{\eta_{m+1}, \mu_{m+1}, \sigma_{m+1}^2, \eta_m, \mu_m, \sigma_m^2\right\}$;

4. Compute the MH ratio as

$$r_b = \frac{p^*\left(\eta_{m+1}, \mu_{m+1}, \sigma_{m+1}^2, \eta_m, \mu_m, \sigma_m^2 \mid y\right) q((m+1) \to m)}{p^*(\eta_m, \mu_m, \sigma_m^2 \mid y)\, \pi\left(\eta_{m+1}, \mu_{m+1}, \sigma_{m+1}^2\right) q(m \to (m+1))} |J|,$$

   where $|J| = 1$ is the Jacobian of the transformation, $q(m \to (m+1))$ is the birth proposed probability, and $q((m+1) \to m)$ is the death proposed probability. As noted by [70], the Jacobian is reduced to 1 for this the bijection where auxiliary variables are introduced;

5. Set $\phi^{(t)} = \phi^*$ with probability $\alpha_b = \min\{1, r_b\}$ and set $\phi^{(t)} = \phi^{(t-1)}$ with probability $1 - \alpha_b$.

The RJ-MCMC algorithm used in the **death** step is as follows:

1. Given the current state $\phi^{(t-1)} = \{\eta_m, \mu_m, \sigma_m^2\}$, with $\eta_m, \mu_m, \sigma_m^2$ the current parameters for an image with m objects, select model "death move" with probability $q((m+1) \to m) = P(\text{death})$;

2. Choose a random object η_j and remove it from the configuration. In this step the dimension of the parameters are decreased by $Q_j = \dim\left(\eta_j, \mu_j, \sigma_j^2\right)$;

3. Set $\phi^* = \phi_{-j}^{(t-1)}$ where $\phi_{-j}^{(t-1)}$ is the current state with the parameters of object η_j removed;

4. Compute the MH ratio as
$$r_d = \frac{1}{r_b},$$
to ensure that the detailed balance condition is met [70, 3];

5. Set $\phi^{(t)} = \phi^*$ with probability $\alpha_d = min\{1, r_d\}$ and set $\phi^{(t)} = \phi^{(t-1)}$ with probability $1 - \alpha_d$.

Split-Merge move:

For the split and merge steps, note the following:

- the algorithm is restricted to the case where only neighboring objects are split or a single object is merged into two objects that are neighbours;
- when moving from one state to another, the proposed object must have equal area as the existing;
- the Markov chain must be reversible, as given in (4.2).

The **merge** step is set out as follows:

- Suppose there are two objects with parameters given by $\{\eta_i, \mu_i, \sigma_i^2, \eta_j, \mu_j, \sigma_j^2\}$;
- We propose to move to a new object $\{\eta_h, \mu_h, \sigma_h^2\} = \{x_h, y_h, s_h, \theta_h, T_h, g_h^r, \mu_h, \sigma_h^2\}$;
- The new objects are set up using:
  - $s_h = \sqrt{s_i + s_j}$,
  - $(x_h, y_h) = \left(\frac{s_i x_i + s_j x_j}{s_i + s_j}, \frac{s_i y_i + s_j y_j}{s_i + s_j}\right)$,
  - $\{\theta_h, T_h, g_h^r, \mu_h, \sigma_h^2\}$ are from one of the parent objects, or at random. In this text the choice is restricted to a random choice between one of the two parent objects;
- In order to match the dimensions, auxiliary variables $\{u_1, u_2, u_3, u_4, u_5, u_6\}$ are introduced
  - $u_1 = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$, i.e. u_1 expresses the distance between the centers of the two objects,
  - $u_2 = \arctan\left(\frac{y_j - y_i}{\sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}}\right)$, the angle created from the union of the two (c_1, c_2),
  - $u_3 = \frac{s_i^2 - s_j^2}{s_i^2 + s_j^2}$,
  - $u_4 = \theta_2$,

  - $u_5 = T_2$,
  - $u_6 = g_2^r$;

- The MH-ratio is computed as

$$r_m = \frac{p^*\left(\eta_h, \mu_h, \sigma_h^2, \eta_{-(i,j)}, \mu_{-(i,j)}, \sigma^2_{-(i,j)} \mid y\right) q\left(1 \to 2\right) \prod_{i=1}^{6} \pi\left(u_i\right)}{p^*\left(\eta_{(i,j)}, \mu_{(i,j)}, \sigma^2_{(i,j)}, \eta_{-(i,j)}, \mu_{-(i,j)}, \sigma^2_{-(i,j)} \mid y\right) q\left(2 \to 1\right)} \left|J\right|,$$

  where $|J| = 1$ is the Jacobian of the transformation, $q\left(1 \to 2\right)$ is the split proposed probability, and $q\left(2 \to 1\right)$ is the merge proposed probability;

- The proposal is accepted with probability $\alpha_m = \min\left\{1, r_m\right\}$ and rejected with probability $1 - \alpha_m$.

The **split** step is set out as follows:

- A move from a single object $\{x_h, y_h, s_h, \theta_h, T_h, g_h^r, u_1, u_2, u_3, u_4, u_5, u_6\}$ to two objects $\{x_1, x_2, y_1, y_2, s_1, s_2, \theta_1, \theta_2, T_1, T_2, g_1^r, g_2^r\}$ is proposed;

- In order to facilitate this move, auxiliary proposal distributions as introduced for $u_1 - u_6$:

  - $\frac{u_1}{2} \sim \pi\left(s\right) \sim UNI\left(0, S_{max}\right)$ the prior for the scale parameter, with S_{max} the number of pixels in the image,
  - $u_2 \sim \pi\left(\theta\right) \sim UNI\left(0, \pi\right)$,
  - $u_3 \sim UNI\left(-1, 1\right)$,
  - $u_4 \sim \pi\left(\theta\right) \sim UNI\left(0, \pi\right)$,
  - $u_5 \sim \pi\left(T\right) \sim Discrete\ UNI\left(1, T_{max}\right)$, with T_{max} the number of templates specified,
  - $u_6 \sim \pi\left(g^r\right) \sim UNI(1.12, 1.4)$;

- To make the move reversible the same transform is used as with the merge step;

- The MH-ratio is computed as $r_s = \frac{1}{r_m}$ to ensure that the de
tailed balance condition is met [70, 3];

- The proposal is accepted with probability $\alpha_s = \min\left\{1, r_s\right\}$ and rejected with probability $1 - \alpha_s$.

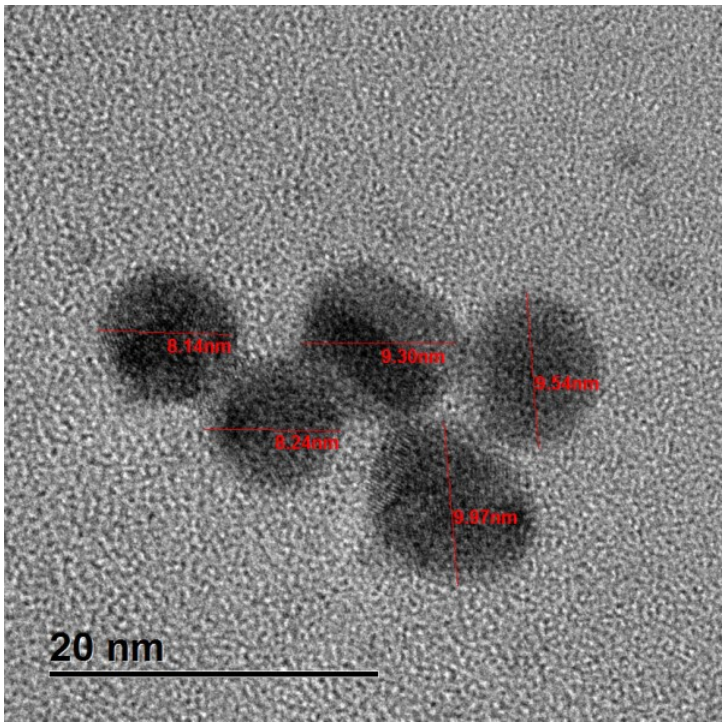


Figure 4.5: Example image for a simplistic case: 5 gold nanoparticles with manual measurements

$\mathbf{B}_4$ - Updating γ :

A Metropolis-Hastings step is used to draw $\gamma = (\gamma_1, \gamma_2)$ from $p^*(\gamma \mid y, \mu, \sigma^2, \eta, m)$, with a random walk log-normal proposal, that is $q\left(\gamma_j^*, \gamma_j^{(t-1)}\right) \equiv LN\left(\alpha_j^{(t-1)}, \delta_j^{(t-1)}\right)$.

Once these steps have been used to update the object parameters, an MCMH algorithm is used to estimate the unknown normalising constant. These steps are outlined in Figure 4.3, with the steps for the inner algorithm (highlighted in red) as discussed in Section 4.5.2.

4.6 A hypothetical example

Prior to providing the algorithm implementation details, a simplistic hypothetical example is provided in this section in order to explain the algorithm for a simple case. Consider the image provided in Figure 4.5. It is easily seen that the image contains 5 objects.

Assuming that the image-processing has already been performed, the algorithm will determine the parameters as follows:

1. Using the output from the pre-processing as the starting state of $\Theta = \{\eta, m, \mu, \sigma^2\} = \{c, g^r, s, \theta, T, m, \mu, \sigma^2\}$, proceed as follows:

(a) Start at Object 1, use the Metropolis-Hastings algorithm to sample μ_1 and σ_1^2, as well as the acceptance probability, say α. Accept the proposed values for μ and σ^2 with probability α. If the proposal is accepted, the values of μ_1 and σ_1^2 are updated at the same time, otherwise both values remain as is;

(b) Staying with Object 1, update $\eta_1 = \{c_1, g_1^r, s_1, \theta_1, T_1\}$ excluding T_1 using the Metropolis-Hastings algorithm, as well as the acceptance probability for each individual parameter $c_1, g_1^r, s_1, \theta_1$. Calculate the overall acceptance probability for this step as the product of the individual acceptance probabilities, say β. Accept the proposed values with probability β. If the proposal is accepted, the values of $\eta_1 = \{c_1, g_1^r, s_1, \theta_1, T_1\}$ excluding T_1 are updated at the same time, otherwise all values remain as is;

(c) Randomly select the model template to use. If the proposed template differs from the current one, the parameters needed to represent the object are unknown. We therefore proceed to sample these parameters from the proposed distributions. Compute the MH ratio as specified, and accept the proposal with the calculated acceptance probability and update the value of T_1;

(d) Update the value of $\gamma_1 = (\gamma_{1,1}, \gamma_{1,2})$ using the Metropolis-Hastings algorithm, as well as the acceptance probability, say δ. Accept the proposed value for γ with probability δ;

(e) All parameter values for Object 1 have now been updated. Repeat steps (a) to (d) for the remainder of the current number of objects (i.e. for the remainder of the 5 objects);

(f) To update m, we first need to know which move type to use, that is, we increase, decrease or keep m as is by either adding a random new object (birth), deleting a random object (death), combining two random objects (merge) or splitting a random object into two new object (split). This is done by selecting the move type using the probabilities provided.

   i. If a Birth move is selected, generate a random new object by sampling all the object parameters, calculate the acceptance ratio as specified, and accept the proposed parameters with probability equal to the acceptance ratio. If accepted, the object parameters are added to the sample and the number of objects are updated;

   ii. If a Death move is selected, randomly choose an object to delete, calculate the acceptance ratio as specified, and accept the proposed parameters with probability equal to the acceptance ratio. If accepted, the object parameters are removed and the number of objects are updated;

iii. If a Merge move is selected, randomly choose an object to merge, generate the merged object as specified, calculate the acceptance ratio as specified, and accept the proposed parameters with probability equal to the acceptance ratio. If accepted, the object parameters are removed and replaced with the updated parameter values and the number of objects are updated;

iv. If a Split move is selected, randomly choose an object to split, generate the split object as specified, calculate the acceptance ratio as specified, and accept the proposed parameters with probability equal to the acceptance ratio. If accepted, the object parameters are removed and replaced with the updated parameter values and the number of objects are updated;

(g) Now that the parameters for each of the current objects are updated, all the proposed parameters for each of the objects are considered together in a MCMH step. To do this, auxiliary variables are sampled from a normal distribution with parameters defined at the proposed state, and the MH-rejection ratio estimated as indicated. The full set of proposed parameters at the current state is the accepted or rejected.

2. Using the updated values of $\Theta = \{\eta, m, \mu, \sigma^2\} = \{c, g^r, s, \theta, T, m, \mu, \sigma^2\}$ as the new state, redo step 1;

3. Repeat for a specified number of samples, including a burn-in sample. Discard the burn-in sample values and estimate the final parameter values as the average of the rest of the sample's values.

In the next section the full algorithm implementation details are provided, along with coding considerations.

4.7 Algorithm implementation details

This section provides some considerations and recommendations for implementing this algorithm in a suitably chosen coding language. In this study Matlab® is chosen as the coding language, due to, amongst others, the flexibility of being widely used and a well-known language for scientific programming, as well as the capability of Matlab® to deal with both image processing, as illustrated in Chapter 2, as well as MCMC sampling. Other coding languages were considered, but were however not used due to certain shortcomings, such as the simplistic image processing capabilities in R, where more advanced techniques were needed, and the lack of sample code for most of the

MCMC algorithms in Python®, where only basic MCMC algorithms were available. The approach taken in implementing this algorithm can be outlined as follows:

- The image pre-processing steps, as discussed in Section 2.7, are coded as a standalone codeset, with the code provided in Section B.1;
- The inner and outer algorithm are coded as a standalone codeset, with the aim of being able to call the simulation as a function. The code used is provided in Section B.2.

In addition to the code examples, some additional comments are provided next, with the aim of providing guidelines for coding this algorithm from scratch.

4.7.1 Coding considerations

Owing to the fact that the algorithm is very detailed and complex, a first time reader might find (like myself) the task of coding this algorithm very daunting. The following are some considerations that might be helpful when coding, and are all the things that came up during my own journey of learning this algorithm:

- Instead of thinking of the bigger picture, and how it all fits together, I found it helpful to break up the components of the algorithm into easier to understand blocks;
- It is encouraged that the authors and contributors of the papers being researched be approached for help and guidance. Several times during this research project, the feedback and input from the authors of [70] proved to be valuable beyond words. The conversations guided me through the most intricate parts of the algorithm, and provided a clear understanding of the authors' thought processes when specifying the algorithm;
- Taking it one step at a time. By coding the algorithm from the inside out (from the most detailed part of the algorithm up to the outermost element), I managed to gain a deeper understanding of how the components fit together;
- Through a couple of detailed internet searches I was able to gather a multitude of coding examples, used in various settings, with detailed explanations. These examples were effective in guiding my understanding of the coding language Matlab®, a language that I had taught myself through trial and error, and multiple internet searches;
- Probably one of the most important lessons learned while coding was to take some time to understand the coding language structure. More specifically, once

I had spent some time to understand how data structures are interpreted and stored in the various Matlab® packages, it became easier to code the algorithm steps;

- When coding the MCMC steps, it is recommended that the steps be updated one object at a time. Therefore, the steps need to be repeated for each of the object in the image;
- The object parameters are defined for multiple object, therefore, when coding, it should be kept in mind that each of the parameters represents a vector of values.
- Storage of simulated values should be carefully considered. The reader is reminded that multiple parameters are required to be stored at each iteration of the algorithm. Further to this, that the number of values required to be stored at each iteration may differ, as the number of objects change. Therefore, sufficient pre-allocated space should be made available prior to executing the code. In the implementation, a null-valued matrix (for each parameter that needs to be simulated) is pre-allocated with sufficient space to store a large number of values at each iteration.
- When testing the code for the first time, it's advisable to run the code in small increments, ensuring that each step is working as it should. In addition, when code has multiple iteration (for example a for-loop), rather manually enter the loop counter/ iteration value to see if the code inside the loop is working correctly.

Appendix B contains the code sets used in the implementation of the occlusion algorithm. The full code set is also available on GitHub by accessing the following link: GitHub - Bayesian object classification in nanoimages. As discussed in the summary of this mini-dissertation, the aim here is not to improve on the existing algorithm, given that the results obtained by Konomi *et al* [70] are very good. The implementation here rather focusses on explaining the detail of the algorithm in a simplified manner to enable a wider audience to understand and replicate the algorithm. As discussed in Chapter 1, there is great importance in having an automated occlusion algorithm widely available, and by implementing Konomi *et al*'s algorithm using an open source language, such as Matlab®. Snippets of Matlab® code that can be used to implement this algorithm are discussed in Appendix B.

4.8 Algorithm implementation results

This section provides some of the simulation results from the algorithm implementation as discussed in the previous sections.

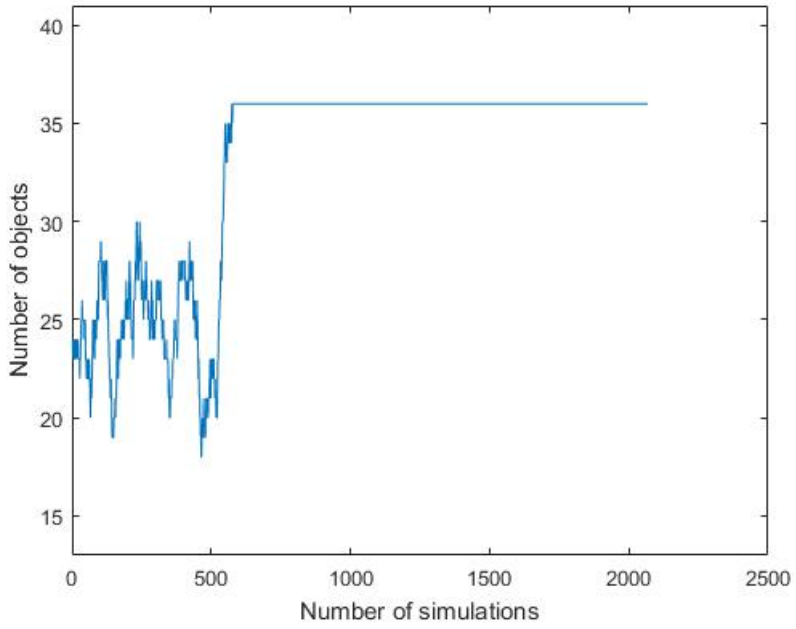


Figure 4.6: Simulation results: Number of simulated objects over time

4.8.1 Number of objects identified

The number of objects simulated at each iteration (including the burn-in period of 500 samples) is shown in Figure 4.6, with the sample distribution given in Figure 4.7, from which it can also be seen that the final number of simulated objects is $m = 36$. When comparing these results to the initial parameter estimates from the image pre-processing, it can be seen that the number of objects identified increases from $m = 24$ to $m = 36$. From visual inspection of the image, it can be seen that there are a total of $m = 44$. The reasons for the algorithm not identifying the remaining objects is discussed in Section 4.9. Also note the increasing trend over time in Figure 4.6, showing that the algorithm is successful in identifying more and more objects over time. The locations of the identified objects are plotted in Figure 4.8. The reader is reminded that all parameters that make up the image, including the location, is simulated, and therefore may not be the exact result from the original image. The simulation results for the object parameters are given in the next section.

4.8.2 Parameter results

The simulated parameter values at each iteration for five (5) randomly selected objects (including the burn-in period of 500 samples) are shown in Figures 4.9 to 4.12. As can be seen from these figures, there is consistent volatility for each of the parameters during the burn-in period, after which the simulation results starts stabilising around the final simulation values. It is observed from the images that all parameter results stabilise

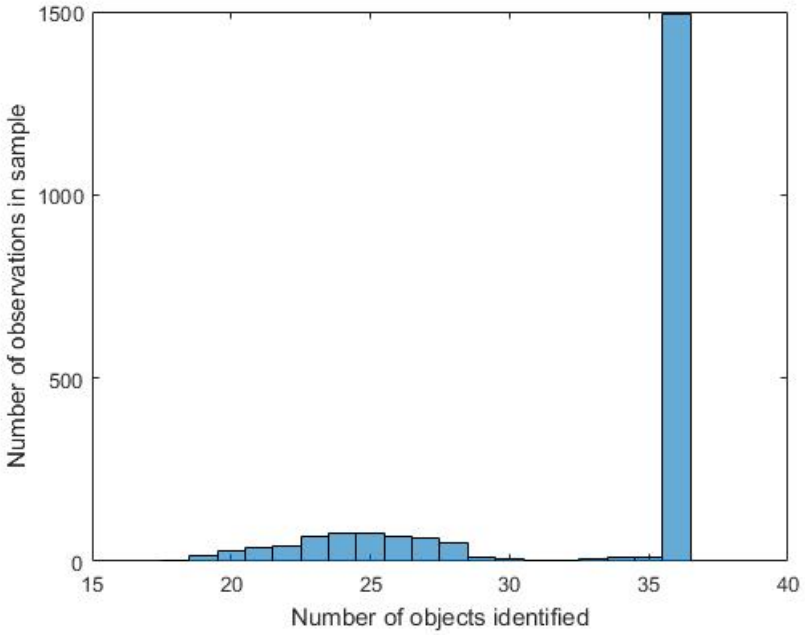


Figure 4.7: Simulation results: Sample distribution of number of simulated objects over time

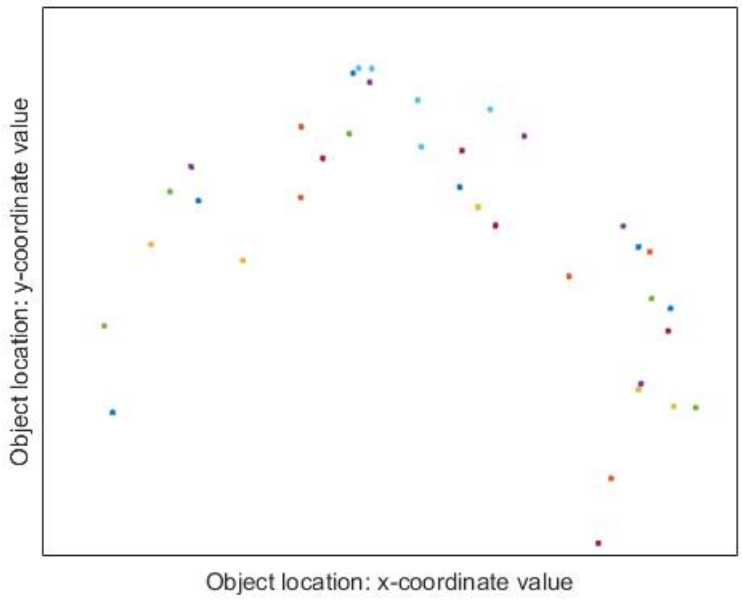


Figure 4.8: Simulation results: Final object locations of simulated objects

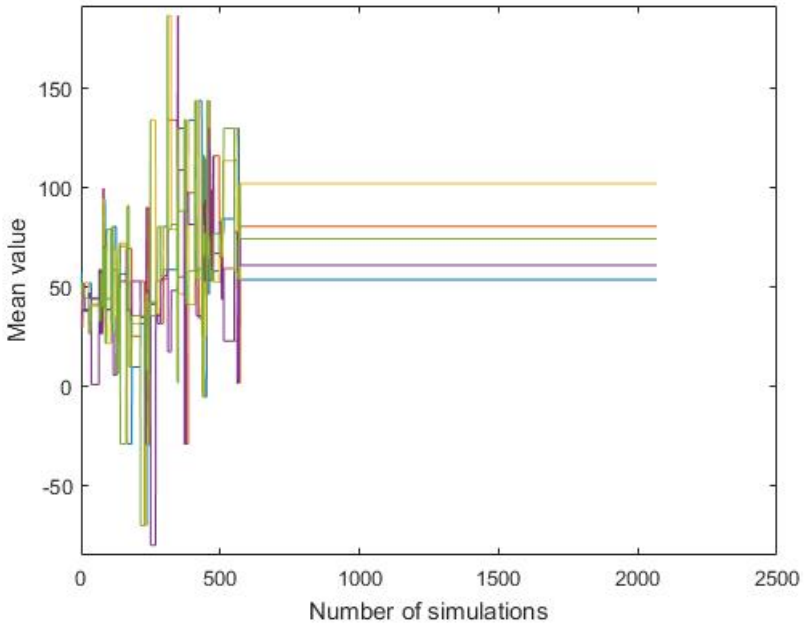


Figure 4.9: Simulation results (5 randomly selected objects): Object mean (μ) over time.

in close succession to one another. This is due to the structure of the algorithm (as discussed in Section 4.5.2), where the in the inner algorithm the parameters results are simulated and accepted/rejected at an individual level, with the full set of parameters then accepted/rejected in the outer algorithm. Therefore, if an individual parameter change is small in comparison to the overall change required to accept the full set of proposed parameters, it is unlikely that the individual parameter change would be accepted in the outer algorithm.

In Figure 4.9, the mean value, μ, for the five randomly selected objects is plotted over time (number of simulations). Note the volatility observed in the image, as discussed in the preceding paragraph, as well as the stabilisation of parameter results in close succession to one another.

Similar to Figure 4.9, Figure 4.10 shows the simulation results for the object variance, σ^2, of the five randomly selected objects, plotted over time (number of simulations). Note the volatility observed in the image, as discussed in the preceding paragraph, as well as the stabilisation of parameter results in close succession to one another.

Figures 4.11 and 4.12, shows the simulation results for the rotation (θ) and object template (T), respectively, where $T = 1$ denotes an ellipse, and $T = 2$ a circle. Note that the parameter values for the template for each of the random objects is given in Table 4.2, to provide clarity on which object was assigned to which template.

The results from the simulation for two other images of gold nanoparticles are given in Appendix B.

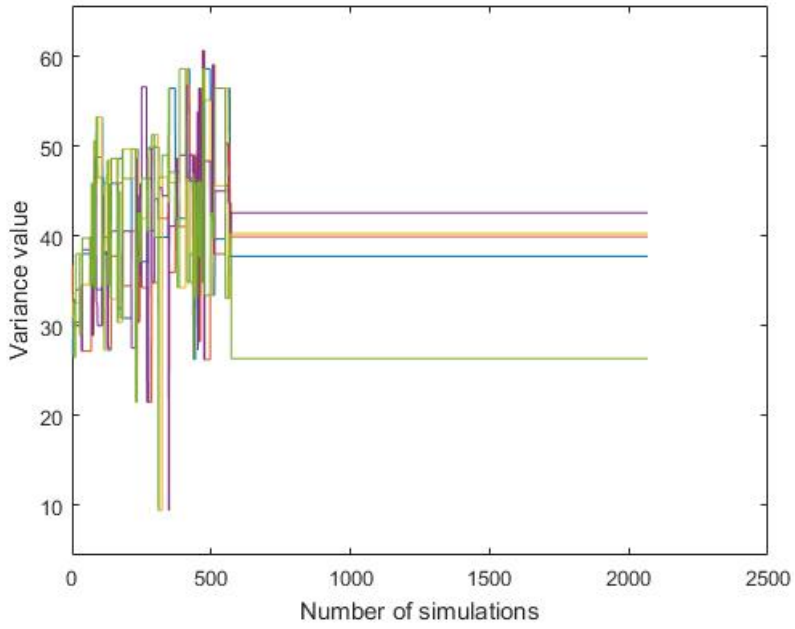


Figure 4.10: Simulation results (5 randomly selected objects): Object variance (σ^2) over time.

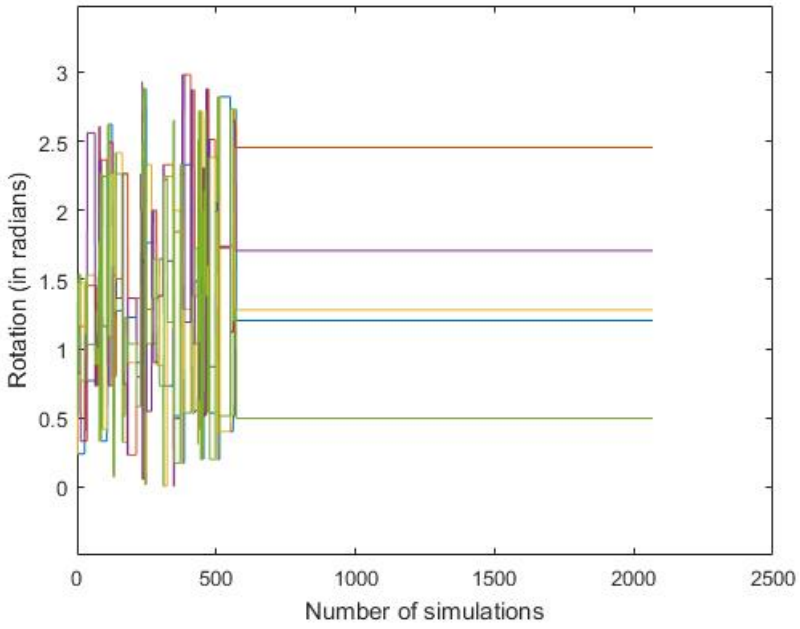


Figure 4.11: Simulation results (5 randomly selected objects): Object rotation (θ) over time.

Table 4.2: Simulation results: Template assigned to the randomly chosen objects

Object	1	2	3	4	5
Template	2	1	2	2	1

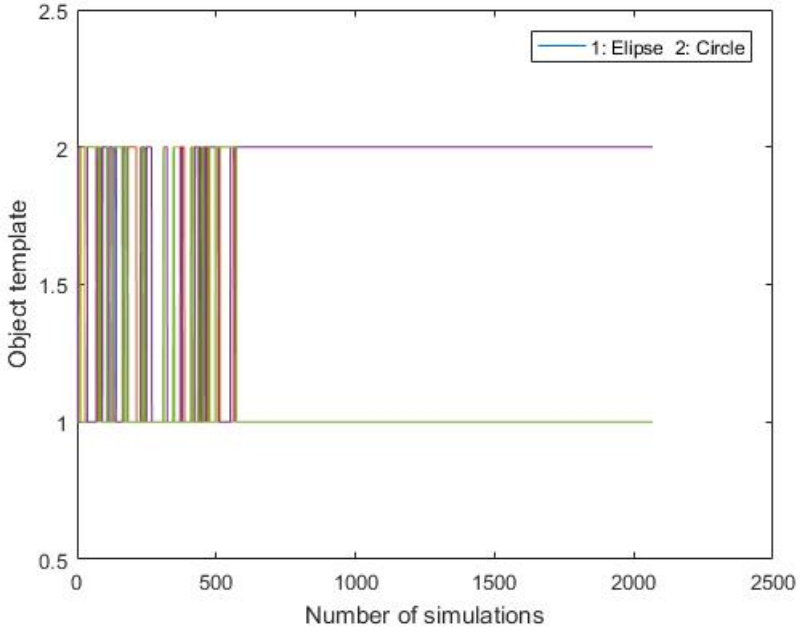


Figure 4.12: Simulation results (5 randomly selected objects): Object template over time

4.9 Discussion

In this chapter an in-depth review of some of the widely used MCMC techniques were discussed, with the aim of making the occlusion algorithm, proposed by [70], easier to follow and understand for the first time reader. In addition to the theoretical overview of these techniques, a detailed breakdown of the components that make up the occlusion algorithm is provided. Flow diagrams are also provided to aid in visualising the process flow in constructing this intricate and detailed algorithm. Code snippets are provided to aid in replicating this algorithm in Matlab®. The simulation results are provided, from where it can be seen that the algorithm is successful in identifying additional object not identified during image pre-processing. However, some objects are still not identified, and this can be attributed to several possible reasons, including the limitation that only circles and ellipses are considered as templates, initial parameter estimates, and the possibility that the algorithm gets "stuck" in a low acceptance probability cycle, which is discussed next.

Although the success of the algorithm is not in question, the expensive computing time is [70]. Several options exist to tackle this problem, of which the most successful is likely to be better acceptance probabilities. As discussed in [47, 14, 24, 39], the acceptance probabilities in MH and RJ-MCMC updates are dependent on having well defined proposals that mix well with the posterior distributions, or as in this case, the pseudo-posterior distributions. In addition to this, not having well defined starting states for the algorithm can have a detrimental impact on the convergence of the

MCMC updates, as is the main reason for having the image pre-processing component in the algorithm. Any improvements made to the image pre-processing can have a significant impact on the algorithm convergence time. This includes improvements such as better initial classification of objects, using more sophisticated segmentation algorithms, leading to a better initial estimate for the number of objects in the image. An almost certain improvement in final classification can be obtained by including further templates for the objects, such as the templates provided in [70].

Chapter 5

Conclusion and Recommendations

In this mini-dissertation the importance of having an automated object classification procedure for classifying nanoparticles in nanoscale images (or nanoimages) is discussed, and a detailed overview of such a procedure, proposed by [70] is provided, with emphasis on applying the procedure to nanoimages of gold nanoparticles. The study aims to address the common phenomenon that occurs during image capturing, namely occlusion of objects in the image. This occlusion leads to some unwanted results during the image analysis phase, making the use of a more sophisticated classification algorithm necessary. A necessary and detailed overview of the theoretical framework required for this procedure is provided. In the process of investigating this procedure, a simplified approach to classifying occluded objects when dealing with homogeneously shaped objects is introduced.

The techniques used in the algorithm involve a combination of several Bayesian techniques to classify the objects in the nanoimage. Markov Chain Monte Carlo (MCMC) sampling techniques are used to simulate the unknown posterior, with samplers ranging from the Metropolis-Hastings and Reversable Jumps MCMC samplers to Monte Carlo Metropolis Hastings samplers used in obtaining the simulated posterior. Since one of the main objectives of this investigation is the processing of images, a discussion on the most widely used image processing techniques is provided, with specific focus on how these techniques are used to extract objects of interest from the image. A short overview of nanotechnology and its applications is provided, along with a variability study for the capturing of nanoimages using a Transmission Electron Microscope. This variability study, which investigates the impact of varying imaging conditions on the measurements obtained using TEM, according to our knowledge, is the first of its kind to be reported in this level of detail, and provides very useful considerations when performing a nanoscale study. Commercially-produced aqueous colloidal gold is used for the investigation, where the average particle sizes are $10nm$. Various microscope conditions were tested in order to establish the influence – if any – of instrument oper-

ation upon the variability of outputs, with three samples obtained for each condition, and these were:

C1: *Three magnification levels* 500 000, 250 000 and 100 000.

C2: *Five minute time delay* The TEM was allowed a five minute stabilisation time before the first measurement, and then another five minutes prior to each subsequent measurement in the three repetitions. This setting aimed to reduce hysteresis by allowing microscope lens currents five minutes to stabilise before image recording in triplicate at each of the magnifications used.

C3: *Upwards magnification* This condition tested the presence of hysteresis by immediately recording three images in succession after the magnification was changed from (a) 100 000 to 250 000, and (b) 100 000 to 500 000. (The time delay between magnification changes to image recording did not exceed 30 seconds in each instance.)

C4: *Down to 30 000 and up* Magnification was reduced from 500 000 to 30 000 times and back up to 500 000 before recording three images immediately. The large change in magnification tested the presence of hysteresis, but in a more severe manner than C3 above.

With the results indicating image measurements at varying magnification levels in TEM causes variation in the measurements, the recommendation is made that magnification levels be carefully selected prior to obtaining samples using TEM. Given that instrument calibration is done at low and intermediate magnification (less than 200 000), it is possible the variation observed originates from the extrapolation of this relationship to higher magnifications. Conducting an actual calibration at higher magnification (rather than using extrapolated values) would almost certainly address this problem. Future studies will explore the accuracy and repeatability obtained when using TEM in combination with image analysis software capable of performing thresholding and automated measurements.

In addition to the theoretical overview of the MCMC techniques, a detailed breakdown of the components that make up the occlusion algorithm is provided. Some flow diagrams are also provided to aid in visualising the process flow in constructing this intricate and detailed algorithm. This detailed breakdown, along with the flow diagrams serve to further explain to the novice user the intricate details of the algorithm, while at the same time making provision for a simplified object shape recognition algorithm. Code snippets are provided to aid in replicating this algorithm in Matlab®. The simulation results are provided, from where it can be seen that the algorithm is successful

in identifying additional objects not identified during image pre-processing. The implementation in this mini-dissertation is therefore not only successful in providing a simplified implementation for novice users, but (similar to the implementation in [70]) is also successful in increasing the classification rate compared to the results obtained using ImagePro® software. However, some objects are still not identified, and this can be attributed to several possible reasons, including the limitation that only circles and ellipses are considered as templates, initial parameter estimates, and the possibility that the algorithm gets "stuck" in a low acceptance probability cycle.

Although the success of the algorithm is not in question, the expensive computing time is [70]. Several options exist to tackle this problem, of which the most successful is likely to be better acceptance probabilities. As discussed in [47, 14, 24, 39], the acceptance probabilities in MH and RJ-MCMC updates are dependent on having well defined proposals that mix well with the posterior distributions, or as in this case, the pseudo-posterior distributions. In addition to this, not having well defined starting states for the algorithm can have a detrimental impact on the convergence of the MCMC updates, as is the main reason for having the image pre-processing component in the algorithm. Any improvements made to the image pre-processing can have a significant impact on the algorithm convergence time. This includes improvements such as better initial classification of objects, using more sophisticated segmentation algorithms, leading to a better initial estimate for the number of objects in the image.

An almost certain improvement in final classification can be obtained by including further templates for the objects, such as the templates provided in [70]. The convergence rate of the algorithm is also of concern when needing a fast computing algorithm. This can be addressed by improving the starting states of the algorithm (in other words by improving the image pre-processing), or alternatively by improving the acceptance probabilities. This can be done by defining proposal distributions that mix well with the pseudo-posterior distribution.

In conclusion, we believe that the work provided in this mini-dissertation will be useful in the analysis of nanoimages, given the improved object classification, simplified approach for classification of occluded, homogeneously shaped objects, detailed breakdown of the components that make up the occlusion algorithm, flow diagrams to aid in visualising the process flow, and Matlab® code provided in this mini-dissertation.

Bibliography

[1] P. Abdulkin, T. L. Precht, B. R. Knappett, H. E. Skelton, D. A. Jefferson, and A. E. H. Wheatley. Systematic control of size and morphology in the synthesis of gold nanoparticles. *Particle & Particle Systems Characterization*, 31(5):571–579, 2014.

[2] T. Acharya and A. K. Ray. *Image Processing: Principles and Applications*. Wiley, 2005.

[3] F. Al-Awadhi, C. Jennison, and M. Hurn. Statistical image analysis for a confocal microscopy two-dimensional section of cartilage growth. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 53(1):31–49, 2004.

[4] H. Amann, J. Escher, and G. Brookfield. *Analysis*, volume 3. Springer, 2005.

[5] V. Amendola and M. Meneghetti. Size evaluation of gold nanoparticles by UV-Vis spectroscopy. *The Journal of Physical Chemistry C*, 113(11):4277–4285, 2009.

[6] M. M. Amiji. *Nanotechnology for Cancer Therapy*. CRC Press, 2006.

[7] M. Arruebo, R. Fernandez-Pacheco, M. R. Ibarra, and J. Santamara. Magnetic nanoparticles for drug delivery. *Nano Today*, 2:22 – 32, 2007.

[8] A. J. Baddeley and M. N. M. Van Lieshout. Stochastic geometry models in high-level vision. *Journal of Applied Statistics*, 20(5-6):231–256, 1993.

[9] A. J. Baddeley and M. N. M. Van Lieshout. Area-interaction point processes. *Annals of the Institute of Statistical Mathematics*, 47(4):601–619, 1995.

[10] S. K. Balasubramanian, J. Jittiwat, J. Manikandan, C. Ong, L. E. Yu, and W. Ong. Biodistribution of gold nanoparticles and gene expression changes in the liver and spleen after intravenous administration in rats. *Biomaterials*, 31:2034 – 2042, 2010.

[11] S. A. Berkowitz. Role of analytical ultracentrifugation in assessing the aggregation of protein biopharmaceuticals. *The AAPS Journal*, 8:E590–E605, 2006.

[12] J. M. Bernardo and A. F. M. Smith. *Bayesian Theory*. IOP Publishing, 2001.

[13] J. Bernd. *Digital Image Processing*. Springer, 2010.

[14] J. Besag and P. J. Green. Spatial statistics and Bayesian computation. *Journal of the Royal Statistical Society. Series B (Methodological)*, 55(1):25–37, 1993.

[15] P. Billingsley. *Probability and Measure*. John Wiley & Sons, 2008.

[16] Y. V. Bogachev, J. S. Chernenco, K. G. Gareev, I. E. Kononova, L. B. Matyushkin, V. A. Moshnikov, and S. S. Nalimova. The study of aggregation processes in colloidal solutions of magnetite-silica nanoparticles by NMR relaxometry, AFM, and UV-Vis spectroscopy. *Applied Magnetic Resonance*, 45(4):329–337, 2014.

[17] J. A. Bondy and U. S. R. Murty. *Graph Theory*. Springer, 2008.

[18] A. Bootz, V. Vogel, D. Schubert, and J. Kreuter. Comparison of scanning electron microscopy, dynamic light scattering and analytical ultracentrifugation for the sizing of poly(butyl cyanoacrylate) nanoparticles. *European Journal of Pharmaceutics and Biopharmaceutics*, 57:369 – 375, 2004.

[19] A. C. Bovik. *The Essential Guide to Image Processing*. Academic Press, 2009.

[20] D. Brambilla, B. LeDroumaguet, J. Nicolas, S. H. Hashemi, L. Wu, S. M. Moghimi, P. Couvreur, and K. Andrieux. Nanotechnologies for Alzheimer's disease: diagnosis, therapy, and safety issues. *Nanomedicine: Nanotechnology, Biology and Medicine*, 7:521 – 540, 2011.

[21] E. Buhr, N. Senftleben, T. Klein, D. Bergmann, D. Gnieser, C. G. Frase, and H. Bosse. Characterization of nanoparticles by scanning electron microscopy in transmission mode. *Measurement Science and Technology*, 20(8):084025, 2009.

[22] R. P. Carney, J. Y. Kim, H. Qian, R. Jin, H. Mehenni, F. Stellacci, and O. M. Bakr. Determination of nanoparticle size distribution together with density or molecular weight by 2D analytical ultracentrifugation. *Nature Communications*, 2:335, 2011.

[23] D. S. Carter and P. M. Prenter. Exponential spaces and counting processes. *Probability Theory and Related Fields*, 21(1):1–19, 1972.

[24] M. Chen and Q. Shao. Monte Carlo methods for Bayesian analysis of constrained parameter problems. *Biometrika*, 85(1):73–87, 1998.

[25] W. Cho, M. Cho, J. Jeong, M. Choi, H. Cho, B. S. Han, S. H. Kim, H. O. Kim, Y. T. Lim, B. H. Chung, and J. Jeong. Acute toxicity and pharmacokinetics of 13nm-sized peg-coated gold nanoparticles. *Toxicology and Applied Pharmacology*, 236:16 – 24, 2009.

[26] W. Cho, M. Cho, J. Jeong, M. Choi, B. S. Han, H. Shin, J. Hong, B. H. Chung, J. Jeong, and M. Cho. Size-dependent tissue kinetics of peg-coated gold nanoparticles. *Toxicology and Applied Pharmacology*, 245:116 – 123, 2010.

[27] J. Clark and D. A. Holton. *A First Look at Graph Theory.* World Scientific, 2005.

[28] H. Cölfen. Analytical ultracentrifugation of nanoparticles. *Polymer News*, 29(4):101–116, 2004.

[29] J. Corbett, P. A. McKeown, G. N. Peggs, and R. Whatmore. Nanotechnology: international developments and emerging products. *CIRP Annals-Manufacturing Technology*, 49(2):523–545, 2000.

[30] Y. Cui, L. J. Lauhon, M. S. Gudiksen, J. Wang, and C. M. Lieber. Diameter-controlled synthesis of single-crystal silicon nanowires. *Applied Physics Letters*, 78:2214–2216, 2001.

[31] M. Daniel and D. Astruc. Gold nanoparticles: Assembly, supramolecular chemistry, quantum-size-related properties, and applications toward biology, catalysis, and nanotechnology. *Chemical Reviews*, 104:293–346, 2004.

[32] P. de Temmerman, E. Verleysen, J. Lammertyn, and J. Mast. Semi-automatic size measurement of primary particles in aggregated nanomaterials by transmission electron microscopy. *Powder Technology*, 261(0):191 – 200, 2014.

[33] W. H. DeJong, W. I. Hagens, P. Krystek, M. C. Burger, A. J. A. M. Sips, and R. E. Geertsma. Particle size-dependent organ distribution of gold nanoparticles after intravenous administration. *Biomaterials*, 29:1912 – 1919, 2008.

[34] L. A. DeLouise. Applications of nanotechnology in dermatology. *The Journal of Investigative Dermatology*, 132:964–975, 2012.

[35] E. R. Dougherty and R. A. Lotufo. *Hands-on Morphological Image Processing.* SPIE, 2003.

[36] R. Durrett. *Essentials of Stochastic Processes.* Springer Science & Business Media, 2012.

[37] L. A. Dykman and V. A. Bogatyrev. Gold nanoparticles: preparation, functionalisation and applications in biochemistry and immunochemistry. *Russian Chemical Reviews*, 76(2):181, 2007.

[38] A. H. Faraji and P. Wipf. Nanoparticles in cellular drug delivery. *Bioorganic & Medicinal Chemistry*, 17:2950 – 2962, 2009.

[39] D. Gamerman and H. F. Lopes. *Markov Chain Monte Carlo: Stochastic Simulation for Bayesian Inference*. CRC Press, 2006.

[40] A. E. Gelfand, A. F. M. Smith, and T. Lee. Bayesian analysis of constrained parameter and truncated data problems using Gibbs sampling. *Journal of the American Statistical Association*, 87(418):pp. 523–532, 1992.

[41] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):721–741, 1984.

[42] J. E. Gentle. *Random Number Generation and Monte Carlo Methods*. Springer, second edition, 2005.

[43] C. J. Geyer and J. Møller. Simulation procedures and likelihood inference for spatial point processes. *Scandinavian Journal of Statistics*, pages 359–373, 1994.

[44] P. Ghosh, G. Han, M. De, C. K. Kim, and V. M. Rotello. Gold nanoparticles in delivery applications. *Advanced Drug Delivery Reviews*, 60:1307 – 1315, 2008.

[45] W. R. Gilks. *Markov Chain Monte Carlo*. Wiley Online Library, 2005.

[46] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Prentice-Hall, 2002.

[47] P. J. Green. Reversible jump Markov Chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82(4):pp. 711–732, 1995.

[48] F. X. Gu, R. Karnik, A. Z. Wang, F. Alexis, E. Levy-Nissenbaum, S. Hong, R. S. Langer, and O. C. Farokhzad. Targeted nanoparticles for cancer therapy. *Nano Today*, 2:14 – 21, 2007.

[49] A. K. Gupta and M. Gupta. Synthesis and surface engineering of iron oxide nanoparticles for biomedical applications. *Biomaterials*, 26:3995 – 4021, 2005.

[50] J. F. Hainfeld, D. N. Slatkin, and H. M. Smilowitz. The use of gold nanoparticles to enhance radiotherapy in mice. *Physics, Medicine & Biology*, 49, 2004.

[51] J. F. Hainfeld, H. M. Smilowitz, M. J. O'Connor, F. A. Dilmanian, and D. N. Slatkin. Gold nanoparticle imaging and radiotherapy of brain tumors in mice. *Nanomedicine*, 8(10):1601–1609, 2013.

[52] W. Haiss, N. T. K. Thanh, J. Aveyard, and D. G. Fernig. Determination of size and concentration of gold nanoparticles from UV-Vis spectra. *Analytical Chemistry*, 79:4215–4221, 2007.

[53] L. Hartsuiker, P. Van Es, W. Petersen, T. G. Van Leeuwen, L. W. M. M. Terstappen, and C. Otto. Quantitative detection of gold nanoparticles on individual, unstained cancer cells by scanning electron microscopy. *Journal of Microscopy*, 244(2):187–193, 2011.

[54] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[55] R. He, X. You, J. Shao, F. Gao, B. Pan, and D. Cui. Core/shell fluorescent magnetic silica-coated composite nanoparticles for bioconjugation. *Nanotechnology*, 18:315601, 2007.

[56] D. W. Hobson. Commercialization of nanotechnology. *Wiley Interdisciplinary Reviews. Nanomedicine and nanobiotechnology*, 1(2):189–202, 2009.

[57] D. Huitink, S. Kundu, C. Park, B. Mallick, J. Z. Huang, and H. Liang. Nanoparticle shape evolution identified through multivariate statistics. *The Journal of Physical Chemistry A*, 114(17):5596–5600, 2010.

[58] P. K. Jain, I. H. El-Sayed, and M. A. El-Sayed. Au nanoparticles target cancer. *Nano Today*, 2:18 – 29, 2007.

[59] T. Jamieson, R. Bakhshi, D. Petrova, R. Pocock, M. Imani, and A. M. Seifalian. Biological applications of quantum dots. *Biomaterials*, 28:4717 – 4732, 2007.

[60] H. Jeffreys. An invariant form for the prior probability in estimation problems. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 186(1007):453–461, 1946.

[61] H. A. Jeng and J. Swanson. Toxicity of metal oxide nanoparticles in mammalian cells. *Journal of Environmental Science and Health, Part A*, 41:2699–2711, 2006.

[62] D. E. Jones, H. Ghandehari, and J. C. Facelli. A review of the applications of data mining and machine learning for the prediction of biomedical properties of nanoparticles. *Computer methods and programs in biomedicine*, 132:93–103, 2016.

[63] E. Jones, T. Oliphant, P. Peterson, et al. NumPy: Open source scientific tools for Python, 2001–. [Online; accessed 2015-06-13].

[64] E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. [Online; accessed 2015-06-13].

[65] E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. [Online; accessed 2014-11-08].

[66] S. Jung, M. Bang, B. S. Kim, S. Lee, N. A. Kotov, B. Kim, and D. Jeon. Intracellular gold nanoparticles increase neuronal excitability and aggravate seizure activity in the mouse brain. *PLoS ONE*, 9:e91360, 2014.

[67] H. Kato, A. Nakamura, and N. Noda. Determination of size distribution of silica nanoparticles: A comparison of scanning electron microscopy, dynamic light scattering, and flow field-flow fractionation with multiangle light scattering methods. *Materials Express*, 4(2):144–152, 2014.

[68] E. S. Kawasaki and A. Player. Nanotechnology, nanomedicine, and the development of new, effective therapies for cancer. *Nanomedicine: Nanotechnology, Biology and Medicine*, 1:101 – 109, 2005.

[69] T. Kim, K. Lee, M. Gong, and S. Joo. Control of gold nanoparticle aggregates by manipulation of interparticle interaction. *Langmuir*, 21(21):9524–9528, 2005.

[70] B. A. Konomi, S. S. Dhavala, J. Z. Huang, S. Kundu, D. Huitink, H. Liang, Y. Ding, and B. K. Mallick. Bayesian object classification of gold nanoparticles. *The Annals of Applied Statistics*, 7(2):640–668, 2013.

[71] B. A. Konomi, S. S. Dhavala, J. Z. Huang, S. Kundu, D. Huitink, H. Liang, Y. Ding, and B. K. Mallick. Bayesian object classification of gold nanoparticles: Online supplementary material. *The Annals of Applied Statistics*, 7(2):640–668, 2013.

[72] R. N. Kostoff, R. G. Koytcheff, and C. G. Y. Lau. Global nanotechnology research literature overview. *Technological Forecasting and Social Change*, 74(9):1733–1747, 2007.

[73] C. Lasagna-Reeves, D. Gonzalez-Romero, M.A. Barria, I. Olmedo, A. Clos, V.M. Sadagopa Ramanujam, A. Urayama, L. Vergara, M.J. Kogan, and C. Soto. Bioaccumulation and toxicity of gold nanoparticles after repeated administration in mice. *Biochemical and Biophysical Research Communications*, 393:649 – 655, 2010.

[74] C. Lee, S. Syu, Y. Chen, S. M. Hussain, A. A. Onischuk, W. L. Chen, and G. S. Huang. Gold nanoparticles regulate the blimp1/pax5 pathway and enhance antibody-secretion in b-cells. *Nanotechnology*, 25:125103, 2014.

[75] P. M. Lee. *Bayesian Statistics: An Introduction*. John Wiley & Sons, 2012.

[76] J. J. Li, L. Zou, D. Hartono, C. N. Ong, B. H. Bay, and L. Y. Lanry Yung. Gold nanoparticles induce oxidative damage in lung fibroblasts in vitro. *Advanced Materials*, 20:138–142, 2008.

[77] F. Liang. A double metropolis–hastings sampler for spatial models with intractable normalizing constants. *Journal of Statistical Computation and Simulation*, 80(9):1007–1022, 2010.

[78] F. Liang and I. H. Jin. A Monte Carlo Metropolis-Hastings algorithm for sampling from distributions with intractable normalizing constants. *Neural Computation*, 25(8):2199 – 2234, 2013.

[79] F. Liang, C. Liu, and R. Carroll. *Advanced Markov Chain Monte Carlo Methods: Learning From Past Samples*, volume 714. John Wiley & Sons, 2011.

[80] G. Linazasoro. Potential applications of nanotechnologies to Parkinson's disease therapy. *Parkinsonism & Related Disorders*, 14:383 – 392, 2008.

[81] J. Liu. Scanning transmission electron microscopy and its application to the study of nanoparticles and nanoparticle systems. *Journal of Electron Microscopy*, 54(3):251–278, 2005.

[82] X. Luo, A. Morrin, A. J. Killard, and M. R. Smyth. Application of nanoparticles in electrochemical sensors and biosensors. *Electroanalysis*, 18:319–326, 2006.

[83] K. V. Mardia, W. Qian, D. Shah, and K. M. A. de Souza. Deformable template recognition of multiple occluded objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(9):1035–1042, 1997.

[84] G.V. Martynov. Statistical tests based on empirical processes and related questions. *Journal of Soviet Mathematics*, 61(4):2195–2271, 1992.

[85] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[86] J. Müller, A. N. Pettitt, R. Reeves, and K. K. Berthelsen. An efficient Markov chain Monte Carlo method for distributions with intractable normalising constants. *Biometrika*, 93(2):451–458, 2006.

[87] P. Müller. A Generic Approach to Posterior Integration and Gibbs Sampling. Technical report, Purdue University, Department of Statistics, 1991.

[88] P. Müller. Alternatives to the Gibbs Sampling Scheme, 1992.

[89] I. Murray, Z. Ghahramani, and D. MacKay. MCMC for doubly-intractable distributions. *arXiv preprint arXiv:1206.6848*, 2012.

[90] S. M. Musa. *Nanoscale Flow: Advances, Modeling, and Applications*. CRC Press, 2014.

[91] A. Nel, T. Xia, L. Madler, and N. Li. Toxic potential of materials at the nanolevel. *Science*, 311:622–627, 2006.

[92] N. Otsu. A threshold selection method from gray-level histograms. *Automatica*, Vol 11:23–27, 1975.

[93] P. A. Pappas and V. DePuy. An overview of non-parametric tests in SAS: when, why, and how. *Paper TU04. Duke Clinical Research Institute, Durham*, pages 1–5, 2004.

[94] C. Park, J. Z. Huang, D. Huitink, S. Kundu, B. K. Mallick, H. Liang, and Y. Ding. A multistage, semi-automated procedure for analyzing the morphology of nanoparticles. *IIE Transactions*, 44:507–522, 2011.

[95] M. Petrou and C. Petrou. *Image Processing: The Fundamentals, 2nd Edition*. Wiley, 2010.

[96] A. Pievatolo and Peter J. Green. Boundary detection through dynamic polygons. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 60(3):pp. 609–626, 1998.

[97] D. Pissuwan, T. Niidome, and M. B. Cortie. The forthcoming applications of gold nanoparticles in drug and gene delivery systems. *Journal of Controlled Release*, 149:65 – 71, 2011.

[98] N. M. Razali and Y. B. Wah. Power comparisons of Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors and Anderson-Darling tests. *Journal of Statistical Modeling and Analytics*, 2(1):21–33, 2011.

[99] B. D. Ripley and F. P. Kelly. Markov point processes. *Journal of the London Mathematical Society*, 2(1):188–192, 1977.

[100] C. P. Robert. Bayesian computational methods. In J. E. Gentle, W. K. Härdle, and Y. Mori, editors, *Handbook of Computational Statistics*, Springer Handbooks of Computational Statistics, pages 751–805. Springer Berlin Heidelberg, 2012.

[101] G. O. Roberts and A. F. M. Smith. Simple conditions for the convergence of the Gibbs sampler and Metropolis-Hastings algorithms. *Stochastic Processes and Their Applications*, 49(2):207–216, 1994.

[102] E. Roduner. Size matters: why nanomaterials are different. *Chemical Society Reviews*, 35:583–592, 2006.

[103] C. Roney, P. Kulkarni, V. Arora, P. Antich, F. Bonte, A. Wu, N.N. Mallikarjuana, S. Manohar, H. Liang, A. R. Kulkarni, H. Sung, M. Sairam, and T. M. Aminabhavi. Targeted nanoparticles for drug delivery through the blood-brain barrier for Alzheimer's disease. *Journal of Controlled Release*, 108:193 – 214, 2005.

[104] H. Rue and M. A. Hurn. Bayesian object identification. *Biometrika*, 86(3):649–660, 1999.

[105] J. K. Sahni, S. Doggui, J. Ali, S. Baboota, L. Dao, and C. Ramassamy. Neurotherapeutic applications of nanoparticles in Alzheimer's disease. *Journal of Controlled Release*, 152:208 – 231, 2011.

[106] S.K. Sahoo, S. Parveen, and J.J. Panda. The present and future of nanotechnology in human health care. *Nanomedicine: Nanotechnology, Biology and Medicine*, 3:20 – 31, 2007.

[107] F. Y. Shih. *Image Processing and Mathematical Morphology: Fundamentals and Applications*. CRC Press, 2009.

[108] V. I. Shubayev, T. R. Pisanic II, and S. Jin. Magnetic nanoparticles for theragnostics. *Advanced Drug Delivery Reviews*, 61:467 – 477, 2009.

[109] R. Singh and J. W. Lillard. Nanoparticle-based targeted drug delivery. *Experimental and Molecular Pathology*, 86:215 – 223, 2009.

[110] A. F. M. Smith and G. O. Roberts. Bayesian computation via the Gibbs sampler and related Markov Chain Monte Carlo methods. *Journal of the Royal Statistical Society. Series B (Methodological)*, 55(1):pp. 3–23, 1993.

[111] P. Soille. *Morphological Image Analysis: Principles and Applications*. Springer, 1999.

[112] G. Sonavane, K. Tomoda, and K. Makino. Biodistribution of colloidal gold nanoparticles after intravenous administration: Effect of particle size. *Colloids and Surfaces B: Biointerfaces*, 66:274 – 280, 2008.

[113] M. A. Stephens. Statistics for goodness of fit and some comparisons. *Journal of the American Statistical Association*, 69(347):730–737, 1974.

[114] C. Sun, J. S. H. Lee, and M. Zhang. Magnetic nanoparticles in MR imaging and drug delivery. *Advanced Drug Delivery Reviews*, 60:1252 – 1265, 2008.

[115] K. Tiede, A. B. A. Boxall, S. P. Tear, J. Lewis, H. David, and M. Hassellov. Detection and characterization of engineered nanoparticles in food and the environment. *Food Additives and Contaminants: Part A*, 25:795–821, 2008.

[116] L. Tierney. Markov chains for exploring posterior distributions. *The Annals of Statistics*, pages 1701–1728, 1994.

[117] S. van der Walt, S. C. Colbert, and G. Varoquaux. The numpy array: A structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30, 2011.

[118] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, G. Gouillart, T. Yu, and the Scikit-image contributors. Scikit-image: Image processing in Python. *PeerJ*, 2:e453, 6 2014.

[119] T. Wagner, A. Kroll, C. R. Haramagatti, H. G. Lipinski, and M. Wiemann. Classification and segmentation of nanoparticle diffusion trajectories in cellular micro environments. *PLOS ONE*, 12(1):e0170165, 2017.

[120] F. Wang, D. Banerjee, Y. Liu, X. Chen, and X. Liu. Upconversion nanoparticles in biological labeling, imaging, and therapy. *Analyst*, 135:1839–1854, 2010.

[121] S. A. Wickline, A. M. Neubauer, P. Winter, S. Caruthers, and G. Lanza. Applications of nanotechnology to atherosclerosis, thrombosis, and vascular biology. *Arteriosclerosis, Thrombosis, and Vascular Biology*, 26:435–441, 2006.

[122] D. Williams and C. Carter. *Transmission Electron Microscopy - A Textbook for Material Science.* Springer Science & Business Media, 2009.

[123] A. Yadav and P. Yadav. *Digital Image Processing.* University Science Press, 2009.

[124] T.C. Yih and M. Al-Fandi. Engineered nanoparticles as precise drug delivery systems. *Journal of Cellular Biochemistry*, 97:1184–1190, 2006.

[125] L. E. Yu, L. L. Yung, C. Ong, Y. Tan, K. S. Balasubramaniam, D. Hartono, G. Shui, M. R. Wenk, and W. Ong. Translocation and effects of gold nanoparticles after inhalation exposure in rats. *Nanotoxicology*, 3:235–42, 2007.

[126] L. Zhang, F. X. Gu, J. M. Chan, A. Z. Wang, R. S. Langer, and O. C. Farokhzad. Nanoparticles in medicine: Therapeutic applications and developments. *Clinical Pharmacology Therapies*, 83:761–769, 2007.

[127] X. Zhu, I. Yuri, X. Gan, I. Suzuki, and G. Li. Electrochemical study of the effect of nano-zinc oxide on microperoxidase and its application to more sensitive hydrogen peroxide biosensor preparation. *Biosensors and Bioelectronics*, 22:1600 – 1604, 2007.

[128] J. M. Zook, V. Rastogi, R. I. MacCuspie, A. M. Keene, and J. Fagan. Measuring agglomerate size distribution and dependence of localized surface plasmon resonance absorbance on gold nanoparticle agglomerate size using analytical ultracentrifugation. *ACS Nano*, 5(10):8070–8079, 2011.

Appendix A

Sample distributions: Particles 2-5

This appendix provides the remainder of the sample distributions for Particle 2 (given in Figure A.1), Particle 3 (given in Figure A.2), Particle 4 (given in Figure A.3), and Particle 5 (given in Figure A.4).

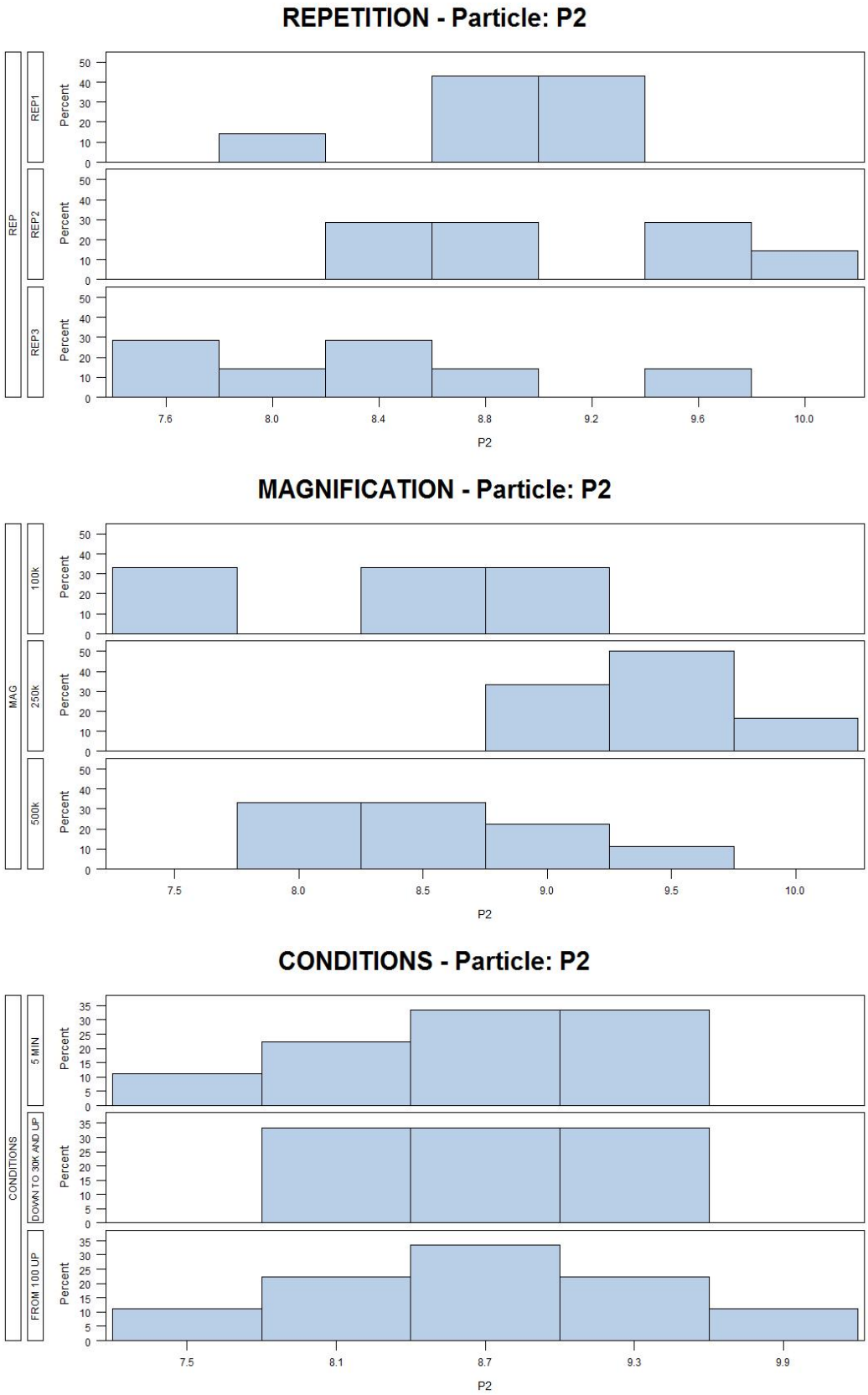


Figure A.1: Sample distributions (Particle 2). This figure shows the distribution of the intensity values measured, split into the following categories: Repetition, Magnification and Other conditions. Repetition further splits the samples into the different repetitions, magnification into the different magnification levels used and conditions into the remainder of the sampling conditions. For each main group the distributions are split between the different levels, for example between repetitions 1, 2 and 3. Each sub-graph shows on the x-axis the intensity values and on the y-axis the percentage of observations within the range corrensponding to the x-axis value.

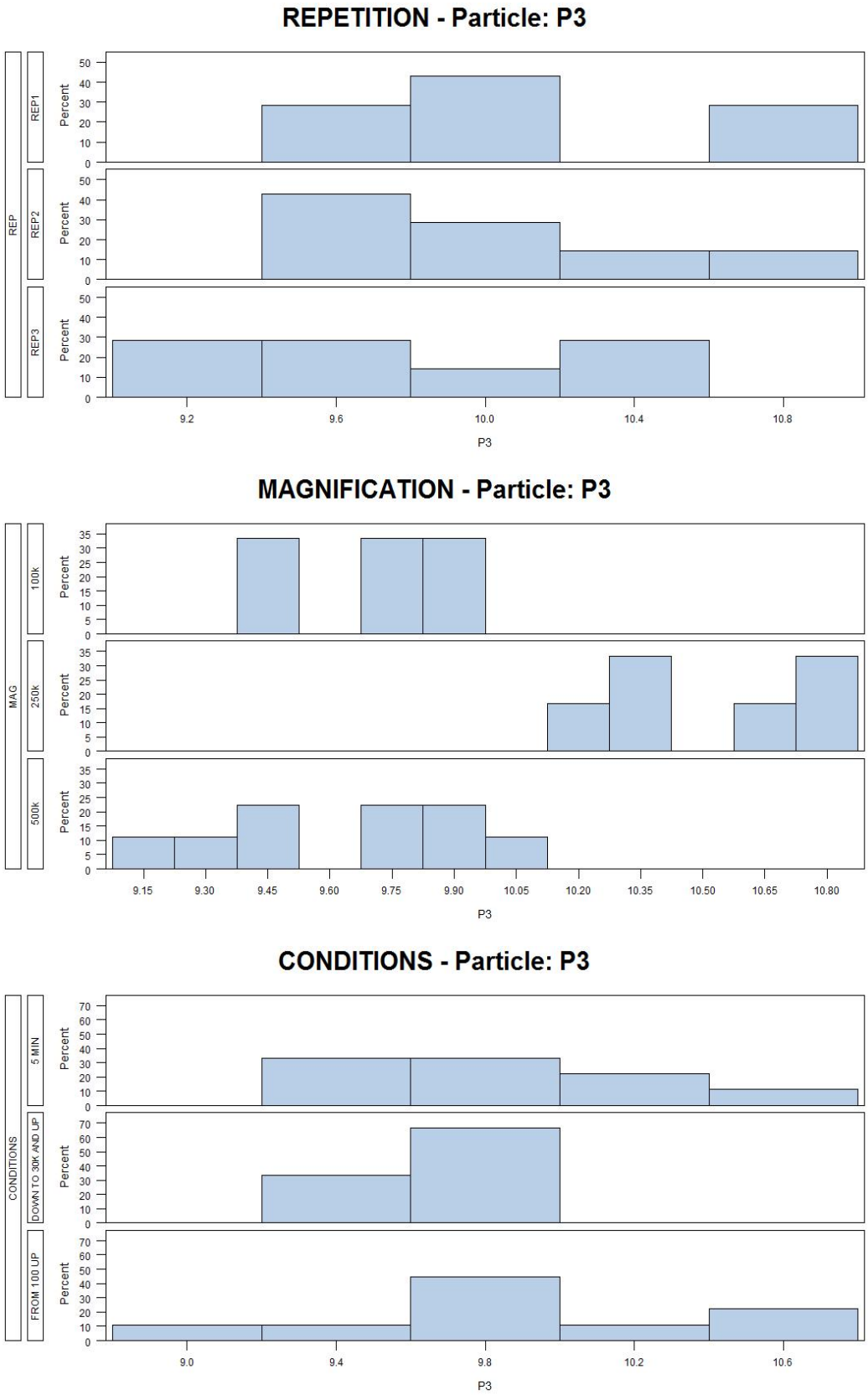


Figure A.2: Sample distributions (Particle 3). This figure shows the distribution of the intensity values measured, split into the following categories: Repetition, Magnification and Other conditions. Repetition further splits the samples into the different repetitions, magnification into the different magnification levels used and conditions into the remainder of the sampling conditions. For each main group the distributions are split between the different levels, for example between repetitions 1, 2 and 3. Each sub-graph shows on the x-axis the intensity values and on the y-axis the percentage of observations within the range correnspondingto the x-axis value.

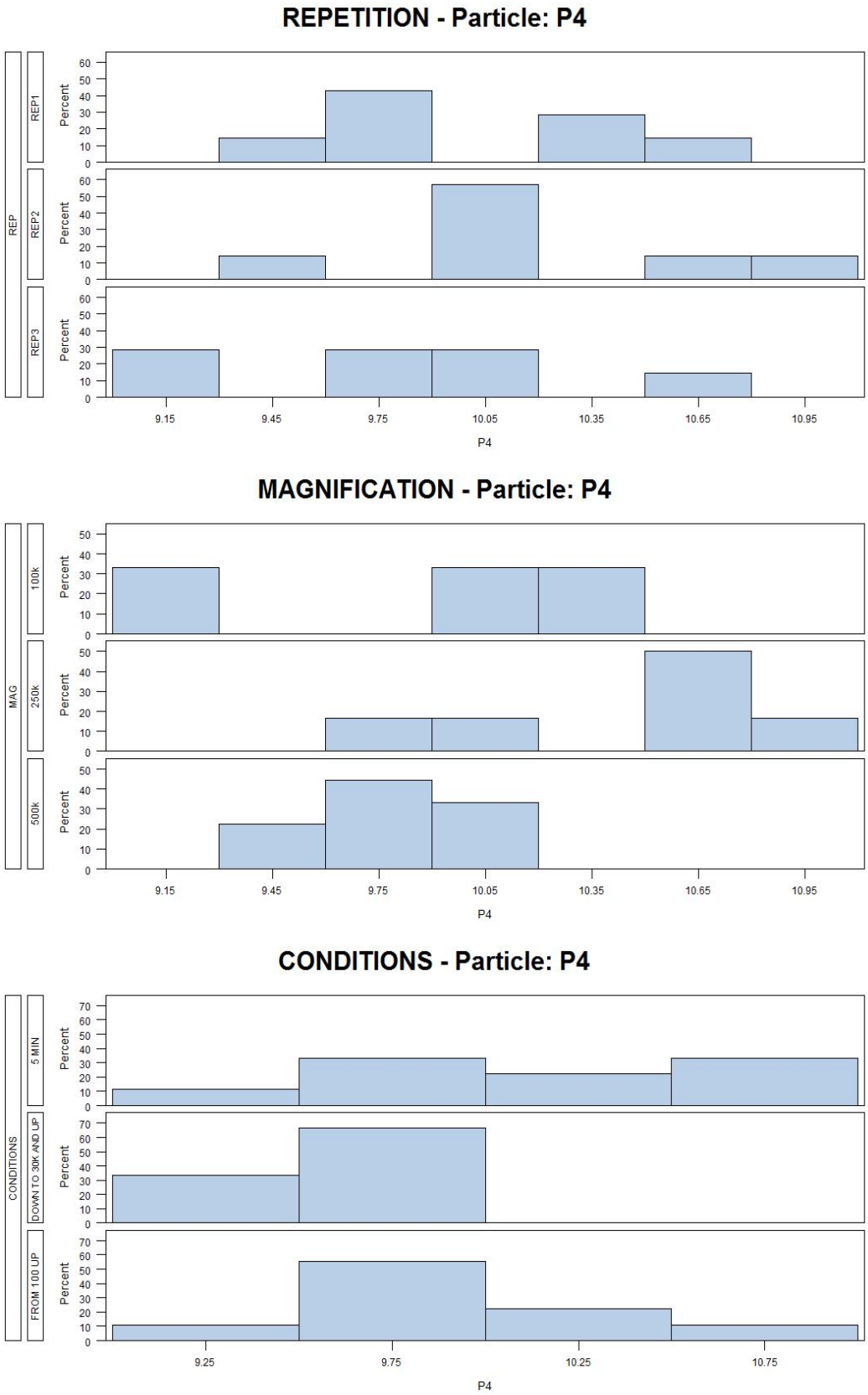


Figure A.3: Sample distributions (Particle 4). This figure shows the distribution of the intensity values measured, split into the following categories: Repetition, Magnification and Other conditions. Repetition further splits the samples into the different repetitions, magnification into the different magnification levels used and conditions into the remainder of the sampling conditions. For each main group the distributions are split between the different levels, for example between repetitions 1, 2 and 3. Each sub-graph shows on the x-axis the intensity values and on the y-axis the percentage of observations within the range corrensponding to the x-axis value.

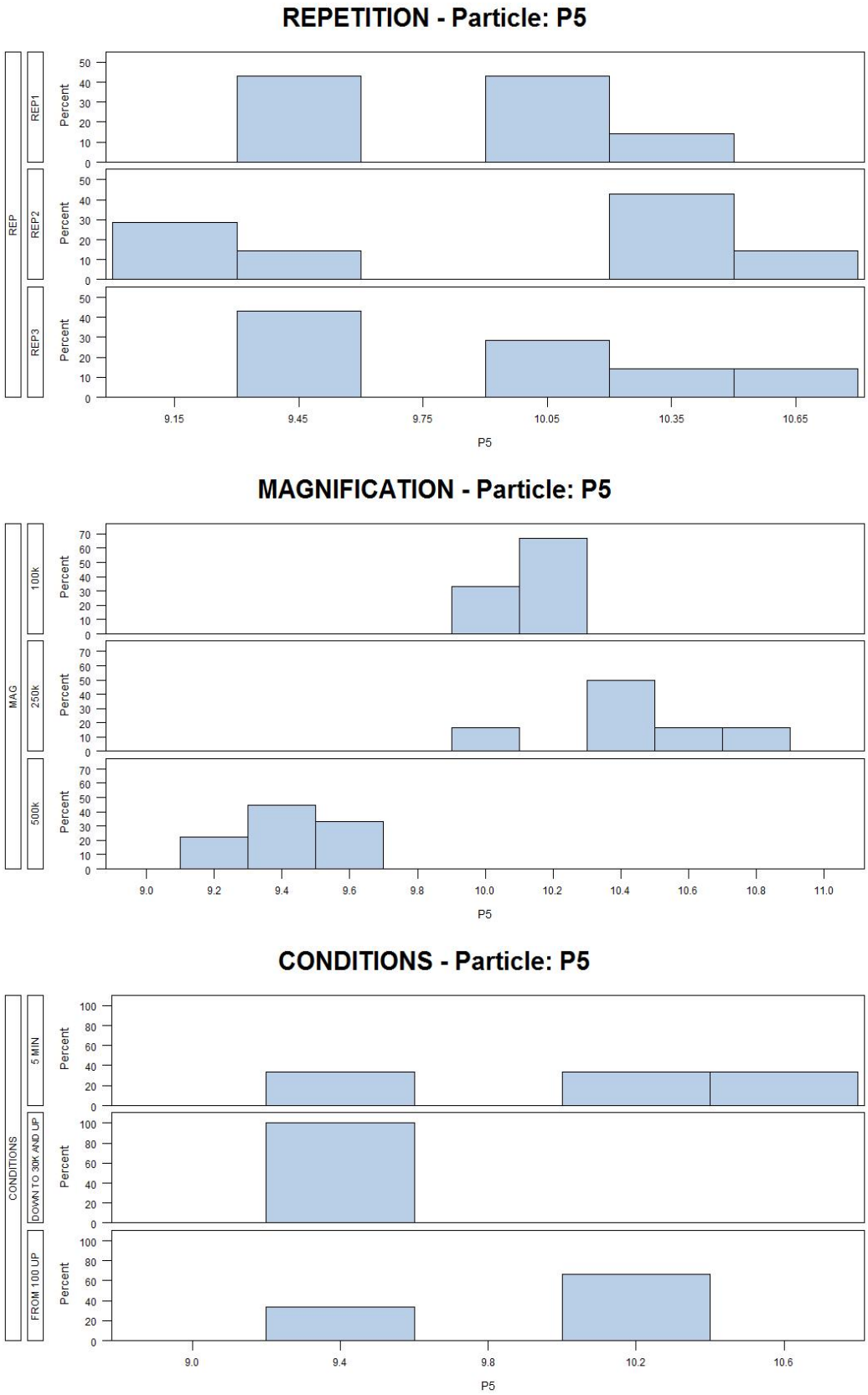


Figure A.4: Sample distributions (Particle 5). This figure shows the distribution of the intensity values measured, split into the following categories: Repetition, Magnification and Other conditions. Repetition further splits the samples into the different repetitions, magnification into the different magnification levels used and conditions into the remainder of the sampling conditions. For each main group the distributions are split between the different levels, for example between repetitions 1, 2 and 3. Each sub-graph shows on the x-axis the intensity values and on the y-axis the percentage of observations within the range correnspondingto the x-axis value.

Appendix B

Python® and Matlab® implementation code

This chapter provides snippets of code used in this mini-dissertation. The success of the algorithm proposed by Konomi *et al* has already been proven in [70], and as such the aim of this chapter is not to reproduce those results, but to rather provide some useful code examples that can be used when implementing this algoirithm, such as Python® and Matlab®. Throughout this chapter various Python® packages are utilised, including Numpy [117], a scientific computing package, which includes, amongst others, array processing and image processing capabilities; Scipy [65], which is considered to be one of the core packages for scientific computing, and provides many useful numerical routines; and Scikit-image [118], which contains several algortihms for image processing. Useful documentation and examples for these packages can be found on the following websites:

- Numpy: www.numpy.org;
- Scipy: www.scipy.org;
- Scikit-image scikit-image.org.

B.1 Image pre-processing code

This section provides highlights of the code used during to perform the image pre-processing described in Section 2.7. The code used for image pre-processing is given in Program Listing B.1.

Listing B.1: Image pre-processing: Python® code.

```
import numpy as np
import cv2
```

```
import matplotlib.pyplot as plt

import pymorph
from scipy import ndimage
from skimage.morphology import opening, erosion, dilation
from scipy import misc

dna=misc.imread('goldnp3.png')
dna2=cv2.imread('goldnp3.png')

#Using Gaussian filter on the image before we threshold
med=ndimage.median_filter(dna,5)
med=ndimage.gaussian_filter(med,5)

#Threshold the image using Otsu's Threshold
_, T=
    cv2.threshold(med,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
T=255-T

# Noise removal
kernel=pymorph.sedisk(r=5,dim=2,metric='euclidean',
                      flat='true',h=0).astype(np.uint8)
opening = cv2.morphologyEx(T, cv2.MORPH_OPEN, kernel,
    iterations = 2)

# Finding sure background area
sure_bg = cv2.dilate(opening,kernel,iterations=3)

# Finding sure foreground area
dist_transform =
    cv2.distanceTransform(opening,cv2.cv.CV_DIST_L2,5)
ret, sure_fg =
    cv2.threshold(dist_transform,0.7*dist_transform.max(),255,0)

# Finding unknown region
sure_fg = np.uint8(sure_fg)
unknown = cv2.subtract(sure_bg,sure_fg)

# Marker labelling
```

```
from skimage import measure
markers = measure.label(sure_fg)
# Add one to all labels so that sure background is not 0, but
    1
markers = markers+1
# Now, mark the region of unknown with zero
markers[unknown==255] = 0

markers=markers.astype(np.int32)
nr_objects=np.max(markers)
print "Number of components:", nr_objects

#Apply watershed
markers_old=np.copy(markers)
cv2.watershed(dna2,markers)
markers=markers.astype(np.int32)
dna2[markers == -1] = [255,0,0]

#Save the image
plt.imshow(markers, cmap='jet', interpolation='nearest')
plt.axis('off')
plt.savefig('goldnp3_markers_w.png',
                        bbox_inches='tight',frameon=False)
```

Alternatively the Matlab® code in Program Listing B.2 can be used to perform the image processing. The image pre-processing done in Chapter 2 made use of this code.

Listing B.2: Image pre-processing: Matlab® code.

```
% function [size1,parameters] = image_pre_processing(f)
%Threshold the image using Otsu's threshold level
fmed = medfilt2(f, [5 5]);
fgauss=imgaussfilt(fmed,5);
mmshow(fgauss);
% mmwrite(fgauss,'filepath');
level = graythresh(fgauss);
fthresh =  mmthreshad (f, 255*level);

%Clean up the image:
```

```
    % Erosion, Opening and obtain the gradient image in order
        to label
fero = mmero(fthresh);
fopen=mmopen(fero);
fopen=mmneg(fopen);
mmshow(fopen);
%mmwrite(fopen,'filepath');
fdist=mmdist(fopen,mmsebox(5));
mmshow(fdist);
% mmwrite(fdist,'filepath');
%fgrad = mmgradm(fdist);
%mmshow(fgrad);
flabel=bwlabel(fdist);

%Find the 20th percentile and remove objects smaller that this
s=regionprops(fdist,'basic');
fscale=regionprops(flabel,'area');
fscalearray=transpose(double([fscale.Area]));
fquart=quantile(fscalearray,0.05);
idx=find([fscale.Area]>fquart);
f2=ismember(flabel,idx);
imshow(f2);
% mmwrite(f2,'filepath');



%Perform Opening Top Hat to remove holes and further noise
% f3=mmopenth(f2);
% imshow(f3);

% Label the image and obtain watershed
flabel=bwlabel(f2);
g = mmcwatershed( f2, flabel, mmsebox);
mmshow(f2, g, flabel);

%For plotting only, obtain the line segments surrounding the
    image
mmshow(f2,g);
h = mmintersec(f2,mmneg(g));
 mmshow(h);
```

```
%Removing the objects on the edges
%i = mmedgeoff(h);
%mmshow(i);
%j=mmgradm(h);
%mmshow(f,j);


%Show the labelled image in colour
flabel=bwlabel(f2);
mmlblshow(flabel);
rgb=label2rgb(flabel);

figure, imshow(rgb,'Colormap',jet(255));
% mmwrite(rgb,'filepath');
% Find and plot the centroids
fcentroid=regionprops(flabel,'centroid');
fcentroid2=cat(1,fcentroid.Centroid);
imshow(rgb)
hold on
plot(fcentroid2(:,1),fcentroid2(:,2),'black*')
hold off;
%  mmwrite(rgb,'filepath');


% Calculate the starting values for the paramters:
    %Scale, shift, rotation and mean intensity
fscale=regionprops(flabel,'area');
feccentric=regionprops(flabel,'eccentricity');
%fextrema=regionprops(flabel,'extrema');
fmajoraxis=regionprops(flabel,'majoraxislength');
fminoraxis=regionprops(flabel,'minoraxislength');
frotation=regionprops(flabel,'orientation');
fmean=regionprops(flabel,f,'MeanIntensity');
fscale2=cat(1,fscale.Area);
frotation2=cat(1,frotation.Orientation);
feccentric2=cat(1,feccentric.Eccentricity);
fmean2=cat(1,fmean.MeanIntensity);
size1=size(fscale,1); %Number of initial objects
```

```
% Assign a template to each object based on the eccentricity
%(E1=1.2 => e=0.6)
ftemplate=zeros(size1,1);
for i=1:size1
 if feccentric2(i)<0.6
     ftemplate(i)=1;
     % 0's are ellipses and 1's are circles
 end
end
% Obtain initial parameter matrix, columns 2&3 are the x,y
   coordinates

parameters=[fscale2,fcentroid2,frotation2,ftemplate,fmean2];
```

B.2 Occlussion algorithm code

In this section detailed code sets are provided that describes the different aspects of the occlusion algortihm as discussed in Chapter 4. This set of code follows on from that given in Program listing B.2. Note that the code in Program Listing B.2 already obtains the mean, standard deviation and size (in number of pixels) for each of the objects obtained during image processing. These values are used as input values into the code that follows.

Since the proposal distributions will be used in different steps during the algorithm, the proposal distributions are defined as callable functions in Matlab®.

Examples of this is given in Program Listing B.3.

Listing B.3: Occlussion algorithm Matlab® code.

```
%Input MH WG steps here
%Define proposals: (mu,sigma)
% qsig=@(nu,x)
   (((2)^(-nu/2)/gamma(nu/2))*x^(-nu/2-1)*exp(-1/(2*x)));
qsig=@(nu,x) chi2rnd(x);
qmu=@(mu,sigma) normrnd(mu,sigma);

%Define proposals: ETA={c, g_r, s, theta, T} EXCLUDING T

%Updating scaling paramter s: q=N(sj_prev, sigma_sj)
qs=@(sj_n_1,sigma_sj) normrnd(sj_n_1,sigma_sj);
qs_X = @(X,mu,sigma) normpdf(X,mu,sigma);
```

```
%Updating location paramter c: q=N(cj_prev, sigma_cjI)
qc=@(cj,sigma_sj) mvnrnd(cj,sigma_sj);
qc_X = @(X,mu,sigma) mvnpdf(X,mu,sigma);

%Updating rotation paramter theta: q=UNI(0,PI)
%qtheta=unifrnd(0,pi);
qtheta_X=@(X) unifpdf(X,0,pi);

%Updating random pure paramter g_r: If T=circle(2): No action
    required
%                                   If T=elipse(1):
    q=E1=UNI(1.005;1.8)= (a,b)
g_r_X=@(X) unifpdf(X,1.005,1.8);

%Updating template parameter T: T=DiscreteUNI(1,2)
q_T=@() unidrnd(2);
q_T_X=@(X) unidpdf(X,2);

%Updating gamma: q=LogNormal(alpha_i,delta_i);
q_gamma=@(mu,sigma) lognrnd(mu,sigma);
q_gamma_X=@(X,mu,sigma) lognpdf(X,mu,sigma);

%Initialise sampling constants

% burn=500;
%Use struct

%Initialise sampler
size1=10*size0; %From image pre-processing
fscale3=transpose(parameters(:,1)); % Tranpose to 1 by m
    (#objects) (row,column)
fmean3=transpose(parameters(:,6)); % Tranpose to 1 by m
    (#objects)
fvar3=(transpose(parameters(:,7))); % Tranpose to 1 by m
    (#objects)
floc3=transpose(parameters(:,2:3)); % Tranpose to 2 by m
    (#objects)
ftemplate3=transpose(parameters(:,5)); % Tranpose to 1 by m
```

```
    (#objects) (row, column)
frotation3=transpose(parameters(:,4)); % Tranpose to 1 by m
    (#objects)
feccentric3=transpose(parameters(:,8)); %Transpose to 1 by m
sigmasqr=1;

nSamples=5000;
% Initialise all sample vectors :mu, sigma, s, c, T, theta,
    g_r, gamma
% (alpha, delta), Nj, m
zz=size(fmean3,2);
muSample=zeros(nSamples,size1);
muSample(1,1:zz)=fmean3(1,:);
sigmaSample=zeros(nSamples,size1);
sigmaSample(1,1:zz)=fvar3(1,:);

sSample=zeros(nSamples,size1);
sSample(1,1:zz)=fscale3(1,:);
sigma_sj0=sSample(1,:);

c_xSample= zeros(nSamples,size1);
c_xSample(1,1:zz)=floc3(1,:);
c_ySample= zeros(nSamples,size1);
c_ySample(1,1:zz)=floc3(2,:);

% cSample=struct('C',floc3);
% cj_sample=zeros(2,size1);



TSample=ones(nSamples,size1);
TSample(1,1:zz)=ftemplate3(1,:);

thetaSample=ones(nSamples,size1);
thetaSample(1,1:zz)=abs((pi/180)*frotation3(1,:)); %Convert
    to radians

g_rSample=zeros(nSamples,size1);
for i=1:zz
    g_rSample(1,i)=1/(1-feccentric3(1,i)^2)^(1/4);
```

```
end



alpha1_sample=ones(nSamples,1);
alpha2_sample=ones(nSamples,1);
delta1_sample=ones(nSamples,1);
delta2_sample=ones(nSamples,1);
alpha1_sample(1,1)=unifrnd(3,5);
alpha2_sample(1,1)=unifrnd(3,5);
delta1_sample(1,1)=unifrnd(1,1.5);
delta2_sample(1,1)=unifrnd(1,1.5);

gamma_sample=ones(nSamples,2);
gamma_sample(1,1)=q_gamma(alpha1_sample(1,1),delta1_sample(1,1));
gamma_sample(1,2)=q_gamma(alpha2_sample(1,1),delta2_sample(1,1));

NjSample=zeros(nSamples,size1);
NjSample(1,1:zz)=fscale3(1,:);
n=sum(NjSample(1,:));
t=1;
m_sample=ones(nSamples,1);
m_sample(1,1)=size0;

m_updated_ind=0;

while t<nSamples+1;
    if m_updated_ind==1
        t=t+1;
        latest_m=m_sample(t-1,1);
    end
    if m_updated_ind==0 && t>1
        latest_m=m_sample(t-1,1);
    end
    if t==1
        t=t+1;
        latest_m=size0;
    end

    if latest_m>0
```

```
        for j=1:latest_m %For every object at time t
            [f_n,f_m]=size(f);
            sMax=f_n*f_m;
            sum_f=0;
            sum_f_star=0;
            %Gibbs steps
            %B1: Updating mu and sigma
            sigmaStar=qsig(NjSample(t-1,j)-1,sigmaSample(t-1,j));
                %Draw proposal sigma from I/CHI2(Nj-1,sigma2)
            muStar=qmu(muSample(t-1,j),sigmaStar/sSample(t-1,j));
                %Draw proposal mu from N(yj_bar, sigmaj/Nj)

            %B1: Updating ETA={c, g_r, s, theta, T} EXCLUDING
                T

            %Updating scaling paramter s:
                q=N(sj_prev,sigma_sj)
%                   ,sigma_sj=1/10sj_0, sj_0=estimated scale
    for each object
            sStar=qs(sSample(t-1,j),sigma_sj0(1,j)/10);

            %Updating location paramter c: q=N(cj_prev,
                sigma_cjI)
            x_std=std(floc3(:,1))^2;
            y_std=std(floc3(:,2))^2;
            sigma_cj0=[x_std, 0;0, y_std];
            %end1=size(cSample);
            latest_c=[c_xSample(t-1,j);c_ySample(t-1,j)];
            cStar=qc(latest_c,sigma_cj0/10);

            %Updating rotation paramter theta: q=UNI(0,PI)
            thetaStar=unifrnd(0,pi);

            %Updating random pure paramter g_r: If T=circle:
                SKIP
            %                                   If T=elipse:
                q=E1=UNI(1.005; 1.8)
            Tcurrent=TSample(t-1,j);
            % 1's are ellipses and 2's are circles
```

```
        if Tcurrent==1
            g_rStar=unifrnd(1.005,1.8);
        else
            g_rStar=0;
        end
        %Calculate accpetance probabilities
        for f_i=1:f_n
            for f_j=1:f_m
                item1=1/(2*sigmaSample(t-1,j));
                item2=f(f_i,f_j);
                item3=muSample(t-1,j);
                sum_f=sum_f+(item1*(item2-item3));
                item1star=1/(2*sigmaStar);
                item2star=f(f_i,f_j);
                item3star=muStar;
                sum_f_star=sum_f_star+...
                    (item1star*(item2star-item3star));
            end
        end
        log_like_curr=sum_f;
        log_like_hats=sum_f_star;
%       Steps for B1 Acceptance probability;
%       Acceptance probabilities for MU and SIGMA
        s_j=sigmaSample(1,j);
        item1=(1/sigmaSample(t-1,j));
        item2=1/sSample(t-1,j);
        item3=-1/(2*sigmaSample(t-1,j));
        item4=(sSample(t-1,j)-1);
        item5=muSample(t-1,j)-mean(muSample(1:t-1,j));

        qcurr=item1^item2*...
            exp(item3*...
            (item4*s_j+n*(item5)^2));

        item1star=(1/sigmaStar);
        item2star=1/sStar;
        item3star=-1/(2*sigmaStar);
        item4star=(sStar-1);
        item5star=muStar-mean(muSample(1:t-1,j));
```

```
qhats=item1star^item2star*...
    exp(item3star*...
    (item4star*s_j*(item5star)^2));
pi1=prod(sigmaSample(t-1,1:latest_m));
pi2=sigmaSample(t-1,j)*sigmaStar;
pi_mu_sighats=1/(pi1/pi2);
pi3=prod(sigmaSample(t-1,1:latest_m));
pi_mu_sigcurr=1/(pi3);
if qcurr==0
    qcurr=normrnd(0,0.00005);
end
if qhats==0
    qhats=normrnd(0,0.00005);
end
i1=qcurr*log_like_hats*pi_mu_sighats;
i2=qhats*log_like_curr*pi_mu_sigcurr;
acceptance_B1=(i1/i2);

%Acceptance probabilities for Eta excluding T
qhats_s=qs_X(sStar,sStar,sigma_sj0(1,j)/10);
holder_s=unifpdf(sSample(t-1,1:latest_m),0,sMax);
pi_s_hats=unifpdf(sStar,0,sMax)*...
    prod(holder_s)/holder_s(j);
qcurr_s=qs_X(sSample(t-1,j),sSample(t-1,j),...
    sigma_sj0(1,j)/10);
pi_s=prod(holder_s);

s_eta_hats=sqrt(sum(sSample(t-1,1:latest_m))-...
    sSample(t-1,j)+sStar);
s_eta=sum(sSample(t-1,1:latest_m));
qhats_c=qc_X(cStar,cStar,sigma_cj0/10);

chats1=-gamma_sample(t-1,1);
chats2=gamma_sample(t-1,2);
ch1=chats1*latest_m;
ch2=chats2*s_eta_hats;
pi_c_hats=exp(ch1-ch2);
qcurr_c=qc_X(latest_c,latest_c,sigma_cj0/10);
```

```
pi_c=exp(-gamma_sample(t-1,1)*...
    latest_m-gamma_sample(t-1,2)*s_eta);

qhats_theta=qtheta_X(thetaStar);
holder_theta=...
    unifpdf(thetaSample(t-1,1:latest_m),0,pi);
pi_theta_hats=unifpdf(thetaStar,0,pi)*...
    prod(holder_theta)/holder_theta(j);
qcurr_theta=qtheta_X(thetaSample(t-1,j));
pi_theta=prod(holder_theta);

if g_rStar==0
    qhats_gr=1;
else
    qhats_gr=g_r_X(g_rStar);
end
holder_gr=ones(1,latest_m);
for i= 1:latest_m
    if g_rSample(t-1,i)<1.005 %(e<0.15)
        holder_gr(i)=1;
    else
        holder_gr(i)=...
            unifpdf(g_rSample(t-1,i),1.005,1.8);
    end
end
pi_gr_hats=unifpdf(g_rStar,1.005,1.8)*...
    prod(holder_gr)/holder_gr(j);
qcurr_gr=g_r_X(g_rSample(t-1,j));
pi_gr=prod(holder_gr);

acceptance_s=(qcurr_s*pi_s_hats/qhats_s*pi_s);
acceptance_c=(qcurr_c*pi_c_hats/qhats_c*pi_c);
acceptance_theta=...
(qcurr_theta*pi_theta_hats/qhats_theta*pi_theta);
acceptance_gr=(qcurr_gr*pi_gr_hats/qhats_gr*pi_gr);
acceptance_all=...
    acceptance_s*acceptance_c*...
    acceptance_theta*acceptance_gr;
```

```
u_b1=unifrnd(0,1);
u_b2=unifrnd(0,1);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Update paramters mu, sigma, eta (excl. T)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


if acceptance_B1 > u_b1
    muSample(t,j)=muStar;
    sigmaSample(t,j)=sigmaStar;
else
    muSample(t,j)=muSample(t-1,j);
    sigmaSample(t,j)=sigmaSample(t-1,j);
end
if acceptance_all >= u_b2
    sSample(t,j)=sStar;
    NjSample(t,j)=sStar;
    c_xSample(t,j)=cStar(1);
    c_ySample(t,j)=cStar(2);
    thetaSample(t,j)=thetaStar;
    g_rSample(t,j)=g_rStar;
else
    sSample(t,j)=sSample(t-1,j);
    NjSample(t,j)=NjSample(t-1,j);
    c_xSample(t,j)=c_xSample(t-1,j);
    c_ySample(t,j)=c_ySample(t-1,j);
    thetaSample(t,j)=thetaSample(t-1,j);
    g_rSample(t,j)=g_rSample(t-1,j);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%B2: Updating T
TStar=q_T();
if TStar == TSample(t-1,j)
```

```
          u_T=g_rSample(t,j);
          v_T=u_T;
     else
          u_T=g_rSample(t,j);
          if TStar == 1
               v_T=unifrnd(1.005,1.8);
          else
               v_T=0;
          end
     end

     if v_T<1.005
          pstar_THat=(pi_gr)/holder_gr(j)*...
               1*pi_s*pi_c*pi_theta;
     else
          pstar_THat=(pi_gr)/holder_gr(j)*...
               unifpdf(v_T,1.005,1.8)*pi_s*...
               pi_c*pi_theta;
     end;
     if u_T<1.005
          pstar_Tcurr=(pi_gr)/holder_gr(j)*1;
     else
          pstar_Tcurr=(pi_gr)/holder_gr(j)*...
               unifpdf(u_T,1.005,1.8);
     end;
     r_A=pstar_THat*q_T_X(TSample(t-1,j))*...
          g_r_X(u_T);
     r_B=pstar_Tcurr*q_T_X(TStar)*g_r_X(v_T);
     acceptance_T=r_A/r_B;

     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     %Update T
     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

     u_T=unifrnd(0,1);
     if acceptance_T >= u_T
          TSample(t,j)=TStar;
     else
          TSample(t,j)=TSample(t-1,j);
```

```
        end

        %B4: Updating Gamma=LN(alpha_i, delta_i) with
        % alpha_1=alpha_2=4 and delta_i ~ UNI(1,1.5)

        alpha1=unifrnd(3,5);
        alpha2=unifrnd(3,5);
        delta1=unifrnd(1,1.5);
        delta2=unifrnd(1,1.5);

        gamma1Star=q_gamma(alpha1,delta1);
        gamma2Star=q_gamma(alpha2,delta2);

%       Calculate acceptance probability;
        qhats_gamma1=...
            q_gamma_X(gamma1Star,alpha1,delta1);
        qhats_gamma2=...
            q_gamma_X(gamma2Star,alpha2,delta2);
        qcurr_gamma1=...
          q_gamma_X(gamma_sample(t-1,1),alpha1,delta1);
        qcurr_gamma2=...
          q_gamma_X(gamma_sample(t-1,2),alpha2,delta2);
        pi_gamma=...
            lognpdf(gamma_sample(t-1,1),alpha1,delta1)...
            *lognpdf(gamma_sample(t-1,2),alpha2,delta2);
        pi_gamma_hats=lognpdf(gamma1Star,alpha1,delta1)*...
            lognpdf(gamma2Star,alpha2,delta2);
        acceptance_gamma1=...
        (qcurr_gamma1*pi_gamma_hats/qhats_gamma1*pi_gamma);
        acceptance_gamma2=...
        (qcurr_gamma2*pi_gamma_hats/qhats_gamma2*pi_gamma);


        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %Update gamma
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        u_gam1=unifrnd(0,1);
        u_gam2=unifrnd(0,1);
```

```
            if acceptance_gamma1 >= u_gam1
                gamma_sample(t,1)=gamma1Star;
            else
                gamma_sample(t,1)=gamma_sample(t-1,1);
            end
            if acceptance_gamma2 >= u_gam2
                gamma_sample(t,2)=gamma2Star;
            else
                gamma_sample(t,2)=gamma_sample(t-1,2);
            end




        %END OF UPDATES B1, B2, B4
        %(loop over number of objects);
        end
    end



    %B3: Updating m;
    %RJMCMC steps
    m_updated_ind=0;

    [f_n,f_m]=size(f);
%Determine move type; birth=1/death=2/split=3/merge=4
    if latest_m>1 %p_birth=p_death=p_split=p_merge=0.25;
        p_birth=0.25;
        p_death=0.25;
        p_split=0.25;
        p_merge=0.25;
        move=unifrnd(0,1);
        if move<= p_birth
            move_type=1;
        end
        if (p_birth<move)&&(move<=p_birth+p_death)
            move_type=2;
```

```
        end
        if (p_birth+p_death<move)&&...
                    (move<=p_birth+p_death+p_split)
            move_type=3;
        end
        if p_birth+p_death+p_split<move
            move_type=4;
        end
    end
    if latest_m==1
        %p_birth=p_death=p_split=0.333333;p_merge=0.0;
        p_birth=0.333333;
        p_death=p_birth+0.333333;
        p_split=p_death+0.333333;
        p_merge=0.00;
        move=unifrnd(0,1);
        if move<= p_birth
            move_type=1;
        end
        if (p_birth<move)&&(move<=p_death)
            move_type=2;
        end
        if p_split<move
            move_type=3;
        end
    end
    if latest_m==0
        %p_birth=1.0; p_death=p_split=p_merge=0;
        move_type=1;
        p_birth=1.00;
        p_death=0.00;
        p_split=0.00;
        p_merge=0.00;
    end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%% BIRTH MOVE %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
if move_type==1 %BIRTH MOVE
    %Get current state eta=; mu, sigma
    new_m=latest_m+1;
    cur_mu=muSample(t-1,1:latest_m);
    cur_sigma=sigmaSample(t-1,1:latest_m);
    cur_s=sSample(t-1,1:latest_m);
    %end1=size(cSample);
    cxx=c_xSample(t-1,1:latest_m);
    cyy=c_ySample(t-1,1:latest_m);
    latest_c=[cxx; cyy];
    latest_theta=thetaSample(t-1,1:latest_m);
    latest_gr=g_rSample(t-1,1:latest_m);
    latest_T=TSample(t-1,1:latest_m);

    %Birth move

    %Generate random object
    rn_sigma=qsig(mean(NjSample(t-1,1:latest_m)) ...
        -1,mean(sigmaSample(t-1,1:latest_m)));
    rn_mu=qmu(mean(muSample(t-1,1:latest_m)),rn_sigma);
    rn_s=qs(mean(sSample(t-1,1:latest_m)) ,...
        mean(sigma_sj0(1,1:latest_m))/10);
    kk=unidrnd(new_m);
    rn_c=qc(latest_c(:,kk),sigma_cj0/10);
    rn_theta=unifrnd(0,pi);
    rn_T=q_T();
    rn_gr=unifrnd(1.005,1.8);

    %Calculate pstar (m+1 and m)
    pi_mu_sigbirth=1/...
        (prod(sigmaSample(t-1,1:latest_m))*rn_sigma);
    pi_mu_sigremain=1/...
        (prod(sigmaSample(t-1,1:latest_m)));
    pi_sbirth=...
        prod(unifpdf(sSample(t-1,1:latest_m),0,sMax))...
        *unifpdf(rn_s,0,sMax);
    pi_sremain=...
        prod(unifpdf(sSample(t-1,1:latest_m),0,sMax));
    s_eta_birth=sum(sSample(t-1,1:latest_m));
```

```
        bb1=-gamma_sample(t-1,1)*new_m;
        bb2=gamma_sample(t-1,2)*s_eta_birth;
        pi_cbirth=exp(bb1-bb2);
        pi_cremain=exp(-gamma_sample(t-1,1)*...
            latest_m-gamma_sample(t-1,2)*s_eta_birth);
        pi_thetabirth=...
            prod(unifpdf(thetaSample(t-1,1:latest_m),0,pi))...
            *unifpdf(rn_theta,0,pi);
        pi_thetaremain=...
            prod(unifpdf(thetaSample(t-1,1:latest_m),0,pi));
        pi_grbirth=...
            prod(unifpdf(g_rSample(t-1,1:latest_m),1.005,1.8))...
            *unifpdf(rn_gr,1.005,1.8);
        pi_grremain=...
            prod(unifpdf(g_rSample(t-1,1:latest_m),1.005,1.8));
        pi_Tbirth=prod(unidpdf(TSample(t-1,1:latest_m),2))*...
            unidpdf(rn_T,2);
        pi_Tremain=prod(unidpdf(TSample(t-1,1:latest_m),2));
    %No likelihood needed as birth and remain likelihood
        cancels out

        pstar_birth=pi_mu_sigbirth*pi_sbirth*pi_cbirth*...
            pi_thetabirth*pi_grbirth*pi_Tbirth;
        pstar_remain=pi_mu_sigremain*pi_sremain*...
            pi_cremain*pi_thetaremain*pi_grremain*pi_Tremain;

        pi_birth=(1/rn_sigma)*unifpdf(rn_s,0,sMax)*...
            exp(-gamma_sample(t-1,1)*new_m-...
            gamma_sample(t-1,2)*s_eta_birth)...
            *unifpdf(rn_theta,0,pi)*...
            unifpdf(rn_gr,1.005,1.8)*unidpdf(rn_T,2);
        Jacobian=1;
        r_b1=(pstar_birth*p_death);
        r_b2=(pstar_remain*pi_birth*p_birth);
        if r_b2==0
            r_b2=0.000001;
        end
        r_b=r_b1/r_b2*Jacobian;
        if r_b>1
```

```
            r_b=1;
        end
        %Update m
        u_m=unifrnd(0,1);

        if r_b < u_m
            m_sample(t,1)=new_m;
            muSample(t,new_m)=rn_mu;
            sigmaSample(t,new_m)=rn_sigma;
            sSample(t,new_m)=rn_s;
            NjSample(t,new_m)=rn_s;
            c_xSample(t,new_m)=rn_c(1);
            c_ySample(t,new_m)=rn_c(2);
            thetaSample(t,new_m)=rn_theta;
            g_rSample(t,new_m)=rn_gr;
            TSample(t,new_m)=rn_T;
            m_updated_ind=1;
        else
            m_sample(t,1)=m_sample(t-1,1);
            m_updated_ind=1;
        end
    end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%% DEATH MOVE %%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    if move_type==2 %DEATH MOVE
        cur_mu=muSample(t,1:latest_m);
        cur_sigma=sigmaSample(t,1:latest_m);
        cur_s=sSample(t,1:latest_m);
        %end1=size(cSample);
        latest_c=[c_xSample(t,1:latest_m); ...
            c_ySample(t,1:latest_m)];
        latest_theta=thetaSample(t,1:latest_m);
        latest_gr=g_rSample(t,1:latest_m);
        latest_T=TSample(t,1:latest_m);

        %Choose random object to remove
```

```
        kk=unidrnd(latest_m);

        %Death move
        new_m=latest_m-1;
        %Choose random object
        rn_sigma=cur_sigma(1,kk);
        rn_mu=cur_mu(1,kk);
        rn_s=cur_s(1,kk);
        rn_c=latest_c(:,kk);
        rn_theta=latest_theta(1,kk);
        rn_T=latest_T(1,kk);
        rn_gr=latest_gr(1,kk);


        pi_mu_sigdeath=1/...
            (prod(sigmaSample(t,1:latest_m))/rn_sigma);
        pi_mu_sigremain=1/(prod(sigmaSample(t,1:latest_m)));
        pi_sdeath=...
            prod(unifpdf(sSample(t,1:latest_m),0,sMax))...
            /unifpdf(rn_s,0,sMax);
        pi_sremain=prod(unifpdf(sSample(t,1:latest_m),0,sMax));
        s_eta_death=sum(sSample(t,1:latest_m));
        pi_cdeath=exp(-gamma_sample(t,1)*...
            new_m-gamma_sample(t,2)*s_eta_death);
        pi_cremain=exp(-gamma_sample(t,1)*latest_m-...
            gamma_sample(t,2)*s_eta_death);
        pi_thetadeath=...
            prod(unifpdf(thetaSample(t,1:latest_m),0,pi))/...
            unifpdf(rn_theta,0,pi);
        pi_thetaremain=...
            prod(unifpdf(thetaSample(t,1:latest_m),0,pi));
        pi_grdeath=prod(unifpdf(g_rSample(t,1:latest_m),...
            1.005,1.8))/unifpdf(rn_gr,1.005,1.8);
        pi_grremain=prod(unifpdf(g_rSample(t,1:latest_m),...
            1.005,1.8));
        pi_Tdeath=prod(unidpdf(TSample(t,1:latest_m),2))...
            /unidpdf(rn_T,2);
        pi_Tremain=prod(unidpdf(TSample(t,1:latest_m),2));
%No likelihood needed as death and remain likelihood cancels
```

```
out

    pstar_death=pi_mu_sigdeath*pi_sdeath*pi_cdeath*...
        pi_thetadeath*pi_grdeath*pi_Tdeath;
    pstar_remain=pi_mu_sigremain*pi_sremain*pi_cremain...
        *pi_thetaremain*pi_grremain*pi_Tremain;

    pi_death=(1/rn_sigma)*unifpdf(rn_s,0,sMax)*...
        exp(-gamma_sample(t,1)*new_m-...
        gamma_sample(t,2)*s_eta_death)...
        *unifpdf(rn_theta,0,pi)*...
        unifpdf(rn_gr,1.005,1.8)*unidpdf(rn_T,2);
    Jacobian=1;
    r_b1=(pstar_death*p_death);
    r_b2=(pstar_remain*pi_death*p_death);
    if r_b2==0
        r_b2=0.000001;
    end
    r_b=r_b1/r_b2*Jacobian;
    if r_b==0
        r_b=0.000001;
    end
    if r_b>1
        r_b=1;
    end
    r_d=1/r_b;

    %Update m
    u_m=unifrnd(0,1);

    if r_d < u_m
        m_sample(t,1)=new_m;

        muSample(t,kk:new_m)=muSample(t,kk+1:latest_m);
        muSample(t,latest_m)=0;

        sigmaSample(t,kk:new_m)=...
            sigmaSample(t,kk+1:latest_m);
        sigmaSample(t,latest_m)=0;
```

```
            sSample(t,kk:new_m)=sSample(t,kk+1:latest_m);
            sSample(t,latest_m)=0;

            NjSample(t,kk:new_m)=NjSample(t,kk+1:latest_m);
            NjSample(t,latest_m)=0;

            c_xSample(t,kk:new_m)=...
                c_xSample(t,kk+1:latest_m);
            c_xSample(t,latest_m)=0;

            c_ySample(t,kk:new_m)=...
                c_ySample(t,kk+1:latest_m);
            c_ySample(t,latest_m)=0;

            g_rSample(t,kk:new_m)=...
                g_rSample(t,kk+1:latest_m);
            g_rSample(t,latest_m)=0;

            thetaSample(t,kk:new_m)=...
                thetaSample(t,kk+1:latest_m);
            thetaSample(t,latest_m)=0;

            TSample(t,kk:new_m)=TSample(t,kk+1:latest_m);
            TSample(t,latest_m)=0;
            m_updated_ind=1;
        else
            m_sample(t,1)=m_sample(t-1,1);
            m_updated_ind=1;
        end
    end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% MERGE MOVE %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    if move_type==4 %MERGE MOVE

        cur_mu=muSample(t,1:latest_m);
```

```
cur_sigma=sigmaSample(t,1:latest_m);
cur_s=sSample(t,1:latest_m);
%end1=size(cSample);
latest_c=[c_xSample(t,1:latest_m); ...
    c_ySample(t,1:latest_m)];
latest_theta=thetaSample(t,1:latest_m);
latest_gr=g_rSample(t,1:latest_m);
latest_T=TSample(t,1:latest_m);

%Choose random object to merge
kk=unidrnd(latest_m);

%Merge move
new_m=latest_m-1;

%Generate merged object


if kk < latest_m
    kk1=kk+1;
else
    kk1=kk-1;
end

si=cur_s(1,kk);
sj=cur_s(1,kk1);
xi=latest_c(1,kk);
xj=latest_c(1,kk1);
yi=latest_c(2,kk);
yj=latest_c(2,kk1);
rn_s=sqrt(si+sj);
rn_cx=(si*xi+sj*xj)/(si+sj);
rn_cy=(si*yi+sj*yj)/(si+sj);
rn_c=[rn_cx;rn_cy];
rn_kk=unifrnd(0,1);
if rn_kk<0.5
    kk2=kk;
else
    kk2=kk1;
```

```
end
rn_theta=latest_theta(kk2);
rn_T=latest_T(kk2);
rn_gr=latest_gr(kk2);
rn_sigma=cur_sigma(kk2);
rn_mu=cur_mu(kk2);
u1=sqrt((xi-xj)^2+(yi-yj)^2);
u2=atan((yj-yi)/(u1));
u3=(si^2-sj^2)/(si^2+sj^2);
u4=latest_theta(kk1);
u5=latest_T(kk1);
u6=latest_gr(kk1);

pi_mu_sigmerge=1/(prod(sigmaSample(t,1:latest_m))...
    *rn_sigma/(sigmaSample(t,kk)*sigmaSample(t,kk1)));
pi_mu_sigremain=1/(prod(sigmaSample(t,1:latest_m)));
pi_smerge=prod(unifpdf(sSample(t,1:latest_m),0,sMax))...
    *unifpdf(rn_s,0,sMax)...
    /(unifpdf(sSample(t,kk),0,sMax)*...
    unifpdf(sSample(t,kk1),0,sMax));
pi_sremain=prod(unifpdf(sSample(t,1:latest_m),0,sMax));
s_eta_merge=sum(sSample(t,1:latest_m));
pi_cmerge=exp(-gamma_sample(t,1)*...
    new_m-gamma_sample(t,2)*s_eta_merge);
pi_cremain=exp(-gamma_sample(t,1)*latest_m-...
    gamma_sample(t,2)*s_eta_merge);
pi_thetamerge=...
    prod(unifpdf(thetaSample(t,1:latest_m),0,pi))...
    *unifpdf(rn_theta,0,pi)...
    /(unifpdf(thetaSample(t,kk),0,pi)*...
    unifpdf(thetaSample(t,kk1),0,pi));
pi_thetaremain=prod(unifpdf...
    (thetaSample(t,1:latest_m),0,pi));
pi_grmerge=prod(unifpdf(g_rSample(t,1:latest_m)...
    ,1.005,1.8))*unifpdf(rn_gr,1.005,1.8)...
    /(unifpdf(g_rSample(t,kk),1.005,1.8)*...
    unifpdf(g_rSample(t,kk1),1.005,1.8));
pi_grremain=prod(unifpdf(g_rSample(t,1:latest_m)...
,1.005,1.8));
```

```
        pi_Tmerge=prod(unidpdf(TSample(t,1:latest_m),2))...
            *unidpdf(rn_T,2)...
            /(unidpdf(latest_T(kk),2)*...
            unidpdf(latest_T(kk1),2));
        pi_Tremain=prod(unidpdf(TSample(t,1:latest_m),2));
%No likelihood needed as death and remain likelihood cancels
    out

         pstar_merge=pi_mu_sigmerge*pi_smerge*pi_cmerge...
             *pi_thetamerge*pi_grmerge*pi_Tmerge;
         pstar_remain=pi_mu_sigremain*pi_sremain*...
           pi_cremain*pi_thetaremain*pi_grremain*pi_Tremain;

         pi_merge=unifpdf(u1,0,sMax)*...
             unifpdf(u2,1.005,1.8)*unifpdf(u3,-1,1)...
             *unifpdf(u4,1.005,1.8)*unidpdf(u5,2)*...
             unifpdf(u6,1.005,1.8);
         Jacobian=1;
         r_m1=(pstar_merge*p_split*pi_merge);
         r_m2=(pstar_remain*p_merge);
         if r_m2==0
             r_m2=0.00001;
         end
         r_m=(r_m1/r_m2)*Jacobian;
         if r_m>1
             r_m=1;
         end
         %Update m
         u_m=unifrnd(0,1);

         if r_m < u_m && kk<latest_m
             m_sample(t,1)=new_m;

             muSample(t,kk:latest_m-2)=...
                 muSample(t,kk+2:latest_m);
             muSample(t,latest_m-1)=rn_mu;
             muSample(t,latest_m)=0;

             sigmaSample(t,kk:latest_m-2)=...
```

```
        sigmaSample(t,kk+2:latest_m);
    sigmaSample(t,latest_m-1)=rn_sigma;
    sigmaSample(t,latest_m)=0;

    sSample(t,kk:latest_m-2)=...
        sSample(t,kk+2:latest_m);
    sSample(t,latest_m-1)=rn_s;
    sSample(t,latest_m)=0;

    NjSample(t,kk:latest_m-2)=...
        NjSample(t,kk+2:latest_m);
    NjSample(t,latest_m-1)=rn_s;
    NjSample(t,latest_m)=0;

    c_xSample(t,kk:latest_m-2)=...
        c_xSample(t,kk+2:latest_m);
    c_xSample(t,latest_m-1)=rn_cx;
    c_xSample(t,latest_m)=0;

    c_ySample(t,kk:latest_m-2)=...
        c_ySample(t,kk+2:latest_m);
    c_ySample(t,latest_m-1)=rn_cy;
    c_ySample(t,latest_m)=0;

    g_rSample(t,kk:latest_m-2)=...
        g_rSample(t,kk+2:latest_m);
    g_rSample(t,latest_m-1)=rn_gr;
    g_rSample(t,latest_m)=0;

    thetaSample(t,kk:latest_m-2)=...
        thetaSample(t,kk+2:latest_m);
    thetaSample(t,latest_m-1)=rn_theta;
    thetaSample(t,latest_m)=0;

    TSample(t,kk:latest_m-2)=...
        TSample(t,kk+2:latest_m);
    TSample(t,latest_m-1)=rn_T;
    TSample(t,latest_m)=0;
    m_updated_ind=1;
```

```
        end
        if r_m < u_m && kk>=latest_m
            m_sample(t,1)=new_m;

            muSample(t,kk-1:latest_m-2)=...
                muSample(t,kk+1:latest_m);
            muSample(t,latest_m-1)=rn_mu;
            muSample(t,latest_m)=0;

            sigmaSample(t,kk-1:latest_m-2)=...
                sigmaSample(t,kk+1:latest_m);
            sigmaSample(t,latest_m-1)=rn_sigma;
            sigmaSample(t,latest_m)=0;

            sSample(t,kk-1:latest_m-2)=...
                sSample(t,kk+1:latest_m);
            sSample(t,latest_m-1)=rn_s;
            sSample(t,latest_m)=0;

            NjSample(t,kk-1:latest_m-2)=...
                NjSample(t,kk+1:latest_m);
            NjSample(t,latest_m-1)=rn_s;
            NjSample(t,latest_m)=0;

            c_xSample(t,kk-1:latest_m-2)=...
                c_xSample(t,kk+1:latest_m);
            c_xSample(t,latest_m-1)=rn_cx;
            c_xSample(t,latest_m)=0;

            c_ySample(t,kk-1:latest_m-2)=...
                c_ySample(t,kk+1:latest_m);
            c_ySample(t,latest_m-1)=rn_cy;
            c_ySample(t,latest_m)=0;

            g_rSample(t,kk-1:latest_m-2)=...
                g_rSample(t,kk+1:latest_m);
            g_rSample(t,latest_m-1)=rn_gr;
            g_rSample(t,latest_m)=0;
```

```
            thetaSample(t,kk-1:latest_m-2)=...
                thetaSample(t,kk+1:latest_m);
            thetaSample(t,latest_m-1)=rn_theta;
            thetaSample(t,latest_m)=0;

            TSample(t,kk-1:latest_m-2)=...
                TSample(t,kk+1:latest_m);
            TSample(t,latest_m-1)=rn_T;
            TSample(t,latest_m)=0;
            m_updated_ind=1;
        end
        if r_m >= u_m
            m_sample(t,1)=m_sample(t-1,1);
            m_updated_ind=1;
        end

    end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%% SPLIT MOVE %%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    if move_type==3 %SPLIT MOVE

        cur_mu=muSample(t,1:latest_m);
        cur_sigma=sigmaSample(t,1:latest_m);
        cur_s=sSample(t,1:latest_m);
        %end1=size(cSample);
        latest_c=[c_xSample(t,1:latest_m); ...
            c_ySample(t,1:latest_m)];
        latest_theta=thetaSample(t,1:latest_m);
        latest_gr=g_rSample(t,1:latest_m);
        latest_T=TSample(t,1:latest_m);

        %Choose random object to SPLIT
        kk=unidrnd(latest_m);

        %Split move
        new_m=latest_m+1;
```

```
%Generate split object

if kk < latest_m
    kk1=kk+1;
else
    kk1=kk-1;
end

si=cur_s(1,kk);
sj=cur_s(1,kk1);
xi=latest_c(1,kk);
xj=latest_c(1,kk1);
yi=latest_c(2,kk);
yj=latest_c(2,kk1);
rn_s=sqrt(si+sj);
rn_cx=(si*xi+sj*xj)/(si+sj);
rn_cy=(si*yi+sj*yj)/(si+sj);
rn_c=[rn_cx;rn_cy];
rn_kk=unifrnd(0,1);
if rn_kk<0.5
    kk2=kk;
else
    kk2=kk1;
end
rn_theta=latest_theta(kk2);
rn_T=latest_T(kk2);
rn_gr=latest_gr(kk2);
rn_sigma=cur_sigma(kk2);
rn_mu=cur_mu(kk2);
u1=sqrt((xi-xj)^2+(yi-yj)^2);
u2=atan((yj-yi)/(u1));
u3=(si^2-sj^2)/(si^2+sj^2);
u4=latest_theta(kk1);
u5=latest_T(kk1);
u6=latest_gr(kk1);

pi_mu_sigmerge=1/...
    (prod(sigmaSample(t,1:latest_m))*...
```

```
            rn_sigma/(sigmaSample(t,kk)*sigmaSample(t,kk1)));
        pi_mu_sigremain=1/(prod(sigmaSample(t,1:latest_m)));
        pi_smerge=...
            prod(unifpdf(sSample(t,1:latest_m),0,sMax))...
            *unifpdf(rn_s,0,sMax)/(unifpdf(sSample(t,kk),0,sMax)...
            *unifpdf(sSample(t,kk1),0,sMax));
        pi_sremain=prod(unifpdf(sSample(t,1:latest_m),0,sMax));
        s_eta_merge=sum(sSample(t,1:latest_m));
        pi_cmerge=exp(-gamma_sample(t,1)*...
            new_m-gamma_sample(t,2)*s_eta_merge);
        pi_cremain=exp(-gamma_sample(t,1)*latest_m-...
            gamma_sample(t,2)*s_eta_merge);
        pi_thetamerge=prod(unifpdf(thetaSample(...
            t,1:latest_m),0,pi))*unifpdf(rn_theta,0,pi)...
            /(unifpdf(thetaSample(t,kk),0,pi)*...
            unifpdf(thetaSample(t,kk1),0,pi));
        pi_thetaremain=...
            prod(unifpdf(thetaSample(t,1:latest_m),0,pi));
        pi_grmerge=prod(unifpdf(g_rSample(t,1:latest_m)...
            ,1.005,1.8))*unifpdf(rn_gr,1.005,1.8)...
            /(unifpdf(g_rSample(t,kk),1.005,1.8)*...
            unifpdf(g_rSample(t,kk1),1.005,1.8));
        pi_grremain=prod(unifpdf(g_rSample(...
            t,1:latest_m),1.005,1.8));
        pi_Tmerge=prod(unidpdf(TSample(t,1:latest_m),2))...
            *unidpdf(rn_T,2)/(unidpdf(latest_T(kk),2)...
            *unidpdf(latest_T(kk1),2));
        pi_Tremain=prod(unidpdf(TSample(t,1:latest_m),2));
%No likelihood needed as death and remain likelihood cancels
    out

        pstar_merge=pi_mu_sigmerge*pi_smerge*pi_cmerge*...
            pi_thetamerge*pi_grmerge*pi_Tmerge;
        pstar_remain=pi_mu_sigremain*pi_sremain*...
            pi_cremain*pi_thetaremain*pi_grremain*pi_Tremain;

        pi_merge=unifpdf(u1,0,sMax)*...
            unifpdf(u2,1.005,1.8)*unifpdf(u3,-1,1)...
            *unifpdf(u4,1.005,1.8)*unidpdf(u5,2)*...
```

```
        unifpdf(u6,1.005,1.8);
    Jacobian=1;
    r_m1=(pstar_merge*p_split*pi_merge);
    r_m2=(pstar_remain*p_merge);
    if r_m2==0
        r_m2=0.000001;
    end
    r_m=r_m1/r_m2*Jacobian;
    if r_m==0
        r_m=0.000001;
    end
    if r_m>1
        r_m=1;
    end
    r_s=1/r_m;
    if r_s>1
        r_s=1;
    end
    %Update m
    u_m=unifrnd(0,1);

    if r_s < u_m
        m_sample(t,1)=new_m;

        muSample(t,kk+1)=rn_mu;
        muSample(t,kk+2:new_m) =...
            muSample(t,kk+1:latest_m);

        sigmaSample(t,kk+1)=rn_sigma;
        sigmaSample(t,kk+2:new_m) =...
            sigmaSample(t,kk+1:latest_m);

        sSample(t,kk+1)=rn_s;
        sSample(t,kk+2:new_m)=sSample(t,kk+1:latest_m);

        NjSample(t,kk+1)=rn_s;
        NjSample(t,kk+2:new_m) =...
            NjSample(t,kk+1:latest_m);
```

```
            c_xSample(t,kk+1)=rn_cx;
            c_xSample(t,kk+2:new_m)=...
                c_xSample(t,kk+1:latest_m);

            c_ySample(t,kk+1)=rn_cy;
            c_ySample(t,kk+2:new_m)=...
                c_ySample(t,kk+1:latest_m);

            g_rSample(t,kk+1)=rn_gr;
            g_rSample(t,kk+2:new_m)=...
                g_rSample(t,kk+1:latest_m);

            thetaSample(t,kk+1)=rn_theta;
            thetaSample(t,kk+2:new_m)=...
                thetaSample(t,kk+1:latest_m);

            TSample(t,kk+1)=rn_T;
            TSample(t,kk+2:new_m)=TSample(t,kk+1:latest_m);
            m_updated_ind=1;
        end

        if r_s >= u_m
            m_sample(t,1)=m_sample(t-1,1);
            m_updated_ind=1;
        end

    end


    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%STEP A2: DRAW AUXILIARY VARIABLES FROM THE LIKELIHOOD
    FUNCTION
    K = m_sample(t,1)*8+2+1;
    %parameters: mu, sigma, s, cx, cy, theta,
    %g_r, T + gamma1, gamma2 + m
    mm=m_sample(t,1);
```

```
latest_parms=[m_sample(t,1), muSample(t,1:mm) ,...
    sigmaSample(t,1:mm)...
    ,sSample(t,1:mm),c_xSample(t,1:mm),c_ySample(t,1:mm)...
    ,thetaSample(t,1:mm),g_rSample(t,1:mm),TSample(t,1:mm)...
    ,gamma_sample(t,1:2)];
auxil_parms=normrnd(mean(muSample(t,1:mm)) ,...
    mean(sigmaSample(t,1:mm)),1,K);

%STEP A3: COMPUTE MCMH ACCEPTANCE RATIO
f_zStar=normpdf(auxil_parms,mean(muSample(t,1:mm))...
    ,mean(sigmaSample(t,1:mm)));
f_zCur=normpdf(latest_parms,mean(muSample(t,1:mm))...
    ,mean(sigmaSample(t,1:mm)));

ratio_z=zeros(1,K);
for i=1:K
    ratio_z(1,i)=f_zStar(1,i)/f_zCur(1,i);
end
R_hat=(sum(ratio_z,2))/K;
if R_hat==0
    R_hat=0.000001;
end
if R_hat>1
    R_hat=1;
end
r_hat=1/R_hat;

%STEP A4: ACCEPT/REJECT PROPOSAL
u_z=unifrnd(0,1);

if r_hat < u_z
    %Reject proposal
    m_sample(t,:)=m_sample(t-1,:);
    muSample(t,:)=muSample(t-1,:);
    sigmaSample(t,:)=sigmaSample(t-1,:);
    sSample(t,:)=sSample(t-1,:);
    c_xSample(t,:)=c_xSample(t-1,:);
    c_ySample(t,:)=c_ySample(t-1,:);
    thetaSample(t,:)=thetaSample(t-1,:);
```

```
        g_rSample(t,:)=g_rSample(t-1,:);
        TSample(t,:)=TSample(t-1,:);
        gamma_sample(t,:)=gamma_sample(t-1,:);
    end









end
% Final parameters:
% nSamples=t;
nSamples=t-1;
final_m0=m_sample(1:nSamples,1);
final_m=m_sample(nSamples,1);
final_mu=muSample(1:nSamples,1:final_m);
final_sigma=sigmaSample(1:nSamples,1:final_m);
final_s=sSample(1:nSamples,1:final_m);
final_cx=c_xSample(1:nSamples,1:final_m);
final_cy=c_ySample(1:nSamples,1:final_m);
final_theta=thetaSample(1:nSamples,1:final_m);
final_gr=g_rSample(1:nSamples,1:final_m);
final_T=TSample(1:nSamples,1:final_m);
final_gamma=gamma_sample(1:nSamples,:);

% input ellipse parameters
n=90;
theta_grid = zeros(1,n);
for k = 1: n
        theta_grid(k) = 2 *pi*(k - 1)/n;
end

phi = transpose(final_theta(nSamples,1:final_m));
X0=transpose(final_cx(nSamples,1:final_m));
Y0=transpose(final_cy(nSamples,1:final_m));
S0=transpose(final_s(nSamples,1:final_m));
a=transpose(final_gr(nSamples,1:final_m));
b=zeros(final_m,1);
```

```
shape_x_r=zeros(final_m,n);
shape_y_r=zeros(final_m,n);



for k=1:final_m
    b(k)=1/final_gr(k);
    shape_x_r(k,:)= X0(k) + a(k)*cos( theta_grid );
    shape_y_r(k,:)= Y0(k) + b(k)*sin( theta_grid );
    % the ellipse in x and y coordinates
    R =[cos(phi(k)) sin(phi(k));-sin(phi(k)) cos(phi(k))];

    %Define a rotation matrix
    %let's rotate the ellipse to some angle phii
    rotated_shape = R * [shape_x_r(k,:);shape_y_r(k,:)];
    plot(rotated_shape(1,:),-1*rotated_shape(2,:),'.');
    hold on

end

 hold off

 plot(final_m0)
 ylim([min(final_m0)-5,max(final_m0)+5])
 title('Number of simulated objects over time')
 xlabel('Number of simulations')
 ylabel('Number of objects')

    kk=1

    plot(final_mu(:,kk))
    ylim([min(final_mu(:,kk))-5,max(final_mu(:,kk))+5])
    title(['Object ' num2str(kk) ' mean over time'])
    xlabel('Number of simulations')
    ylabel('Mean value')

    plot(final_sigma(:,kk))
    ylim([min(final_sigma(:,kk))-5,max(final_sigma(:,kk))+5])
    title(['Object ' num2str(kk) ' variance over time'])
    xlabel('Number of simulations')
```

```
ylabel('Variance value')

plot(final_theta(:,kk))
ylim([min(final_theta(:,kk))-5,max(final_theta(:,kk))+5])
title(['Object ' num2str(kk) ' rotation over time'])
xlabel('Number of simulations')
ylabel('Rotation (in radians)')

plot(final_T(:,kk))
ylim([min(final_T(:,kk))-5,max(final_T(:,kk))+5])
title(['Object ' num2str(kk) ' template over time'])
xlabel('Number of simulations')
ylabel('Object template')
legend('1: Elipse  2: Circle')
```

- All the other MH steps are defined in the same manner as the MH updates used in Algorithm B.2, using the information given in Section 4.5.2;
- The RJ-MCMC steps are coded in exactly the same manner as discussed in Section 4.3.5;
- The combined MH and RJ-MCMC steps then form part of a single update in the MHWG algorithm, or alternatively are all one step in the outer algorithm, as discussed in Section 4.5.1;
- The updates are run until no significant change in the parameters are observed, with a burn-in of 500 samples.

B.3 Occlussion algorithm simulation results

In this section, the parameter results of an additional two images of gold nanoparticles are given. The simulated object parameters at each iteration for 5 randomly selected objects (including the burn-in period of 500 samples) is shown in the figures below.

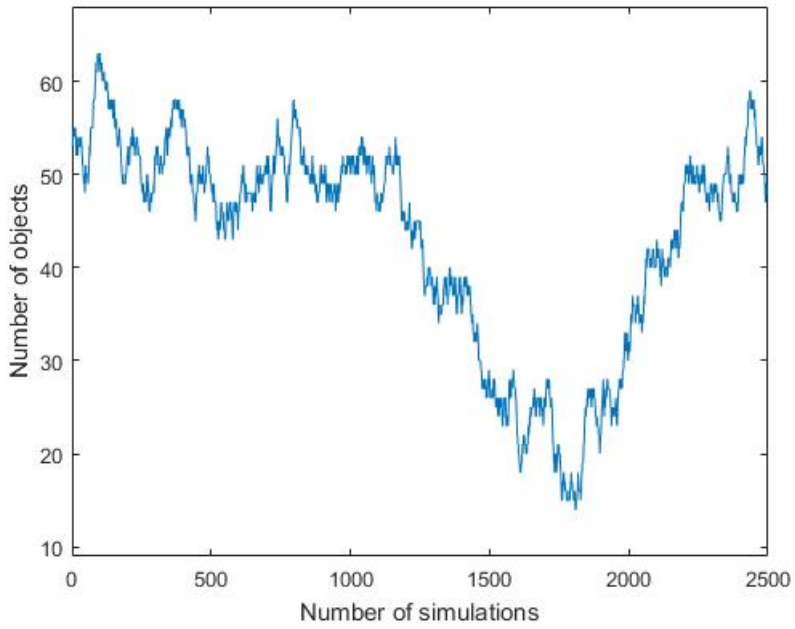


Figure B.1: Simulation results: Image 2 - Number of simulated objects over time.

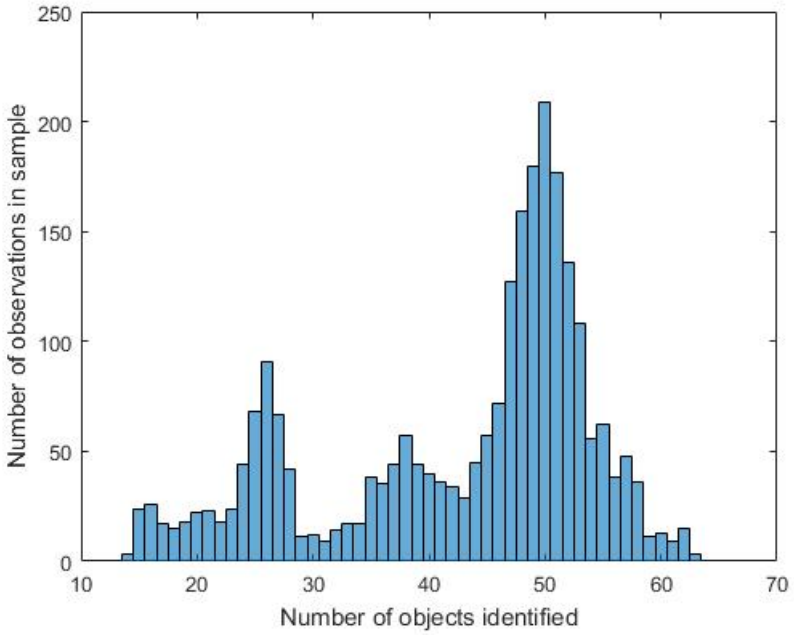


Figure B.2: Simulation results: Image 2 - Sample distribution for the number of simulated objects. Note that $m = 50$ objects are identified in the final result.

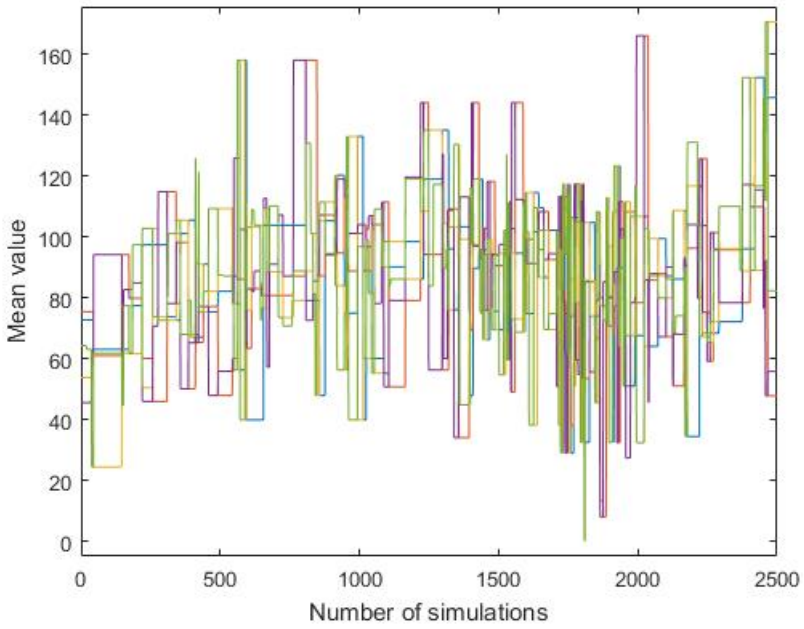


Figure B.3: Simulation results: Image 2 - Mean (μ) value over time.

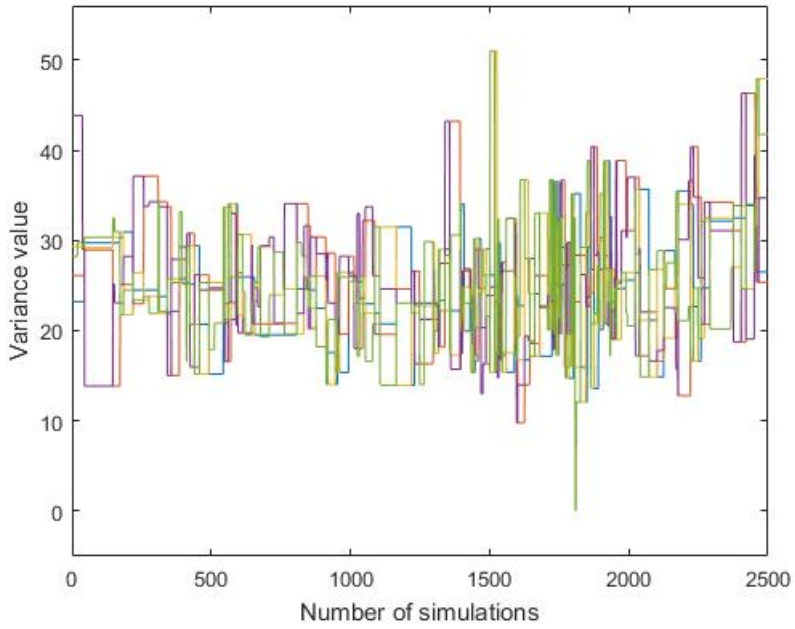


Figure B.4: Simulation results: Image 2 - Variance (σ^2) over time.

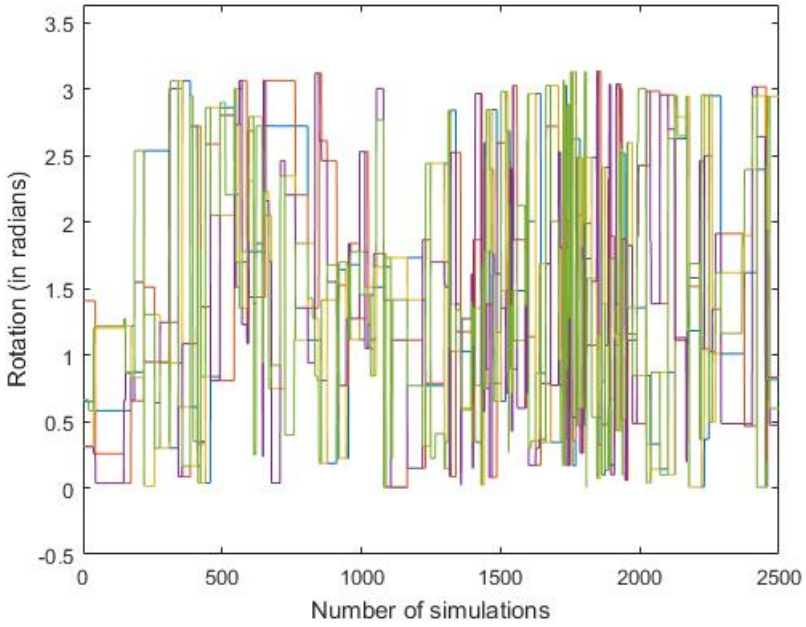


Figure B.5: Simulation results: Image 2 - Rotation (θ) over time.

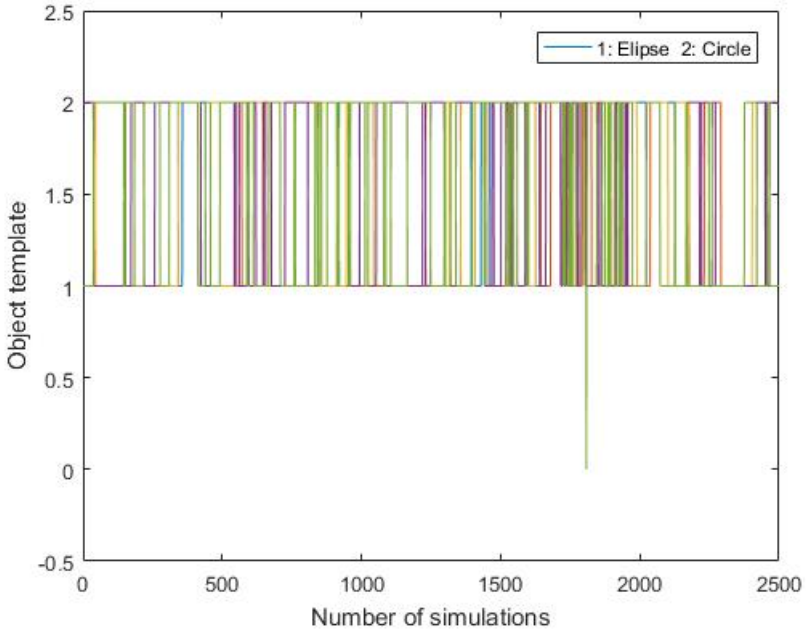


Figure B.6: Simulation results: Image 2 - Template (T) over time.

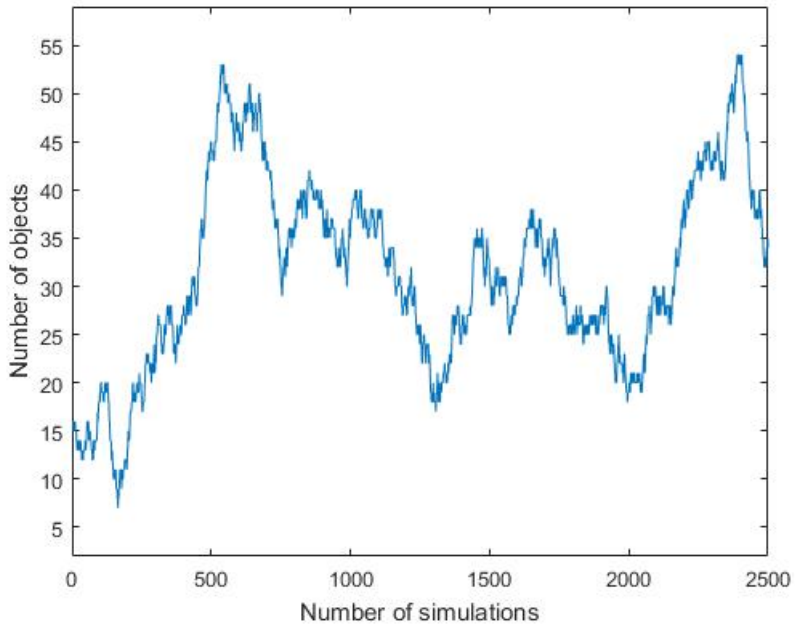


Figure B.7: Simulation results: Image 3 - Number of simulated objects over time.

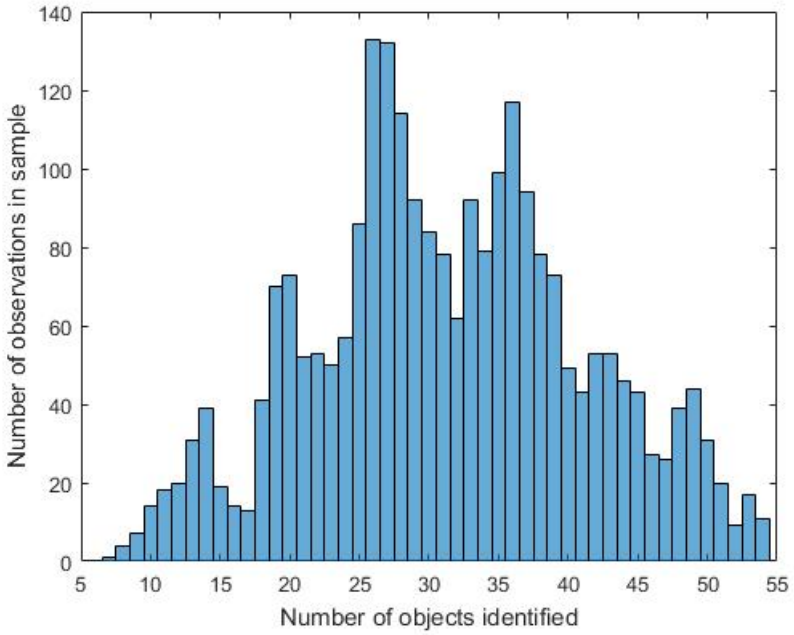


Figure B.8: Simulation results: Image 3 - Sample distribution for the number of simulated objects. Note that $m = 26$ objects are identified in the final result.

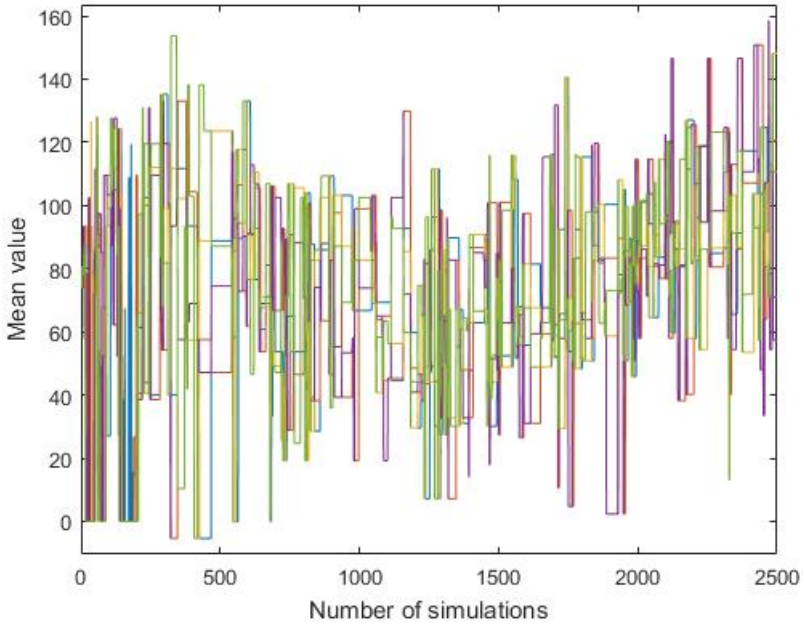


Figure B.9: Simulation results: Image 3 - Mean (μ) over time.

Figure B.10: Simulation results: Image 3 - Variance (σ^2) over time.

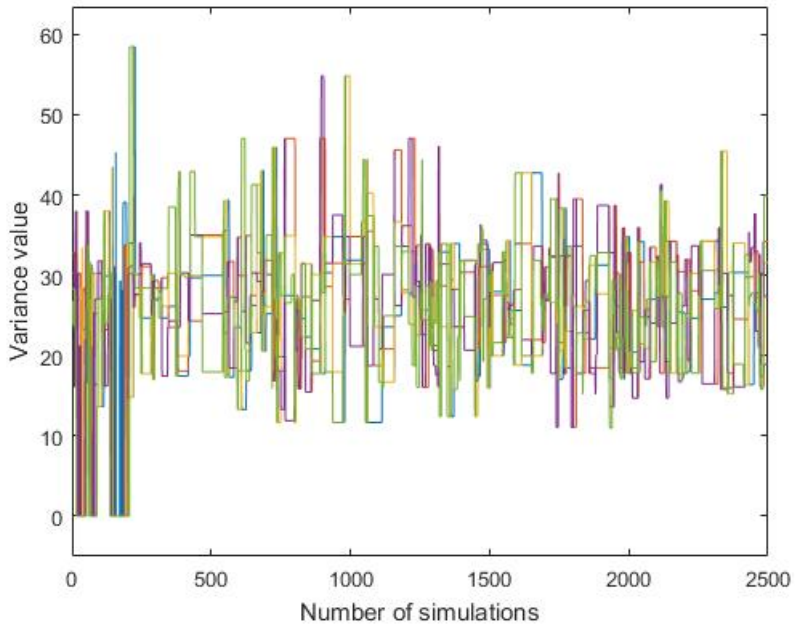


Figure B.11: Simulation results: Image 3 - Rotation (θ) over time.

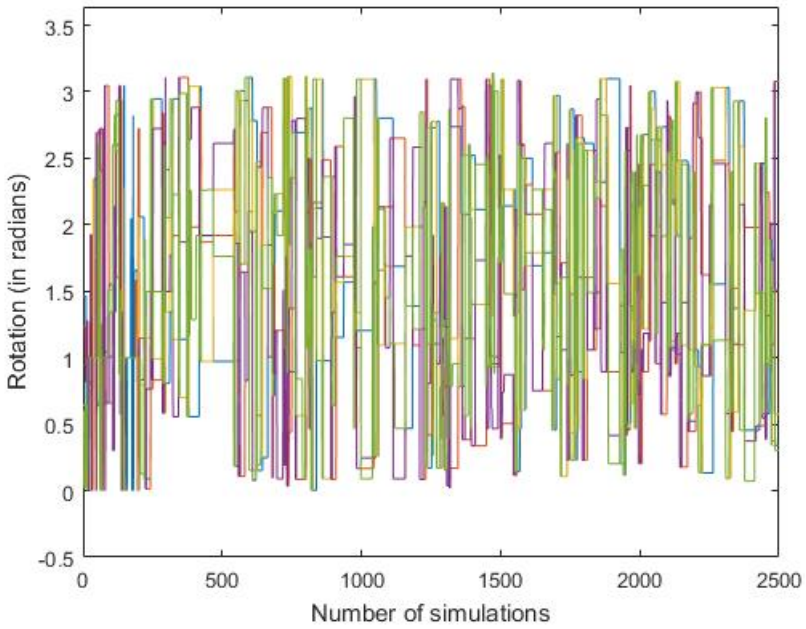


Figure B.12: Simulation results: Image 3 - Template (T) assigned over time.