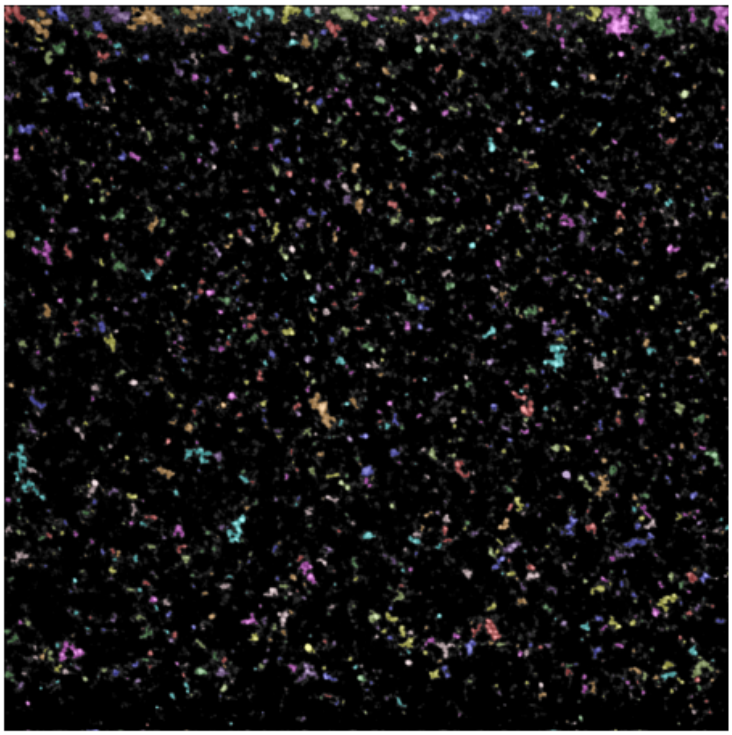
UNIVERSITY OF COPENHAGEN
FACULTY OF SCIENCE



BSc Thesis

Relational shape measures of coagulating milk proteins

Martin Marchioro, Casper Dorph-Jensen,
Aske Nord Raahauge

Vejleder: Jon Sporring

August 29, 2023

Abstract

This paper introduces a novel approach to the study of casein micelle coagulation during milk curdling through the application of spatial analysis methods. Building on Ripley's K-function, we propose a method that computes relational shape measures to analyze the size and spatial distribution of protein clusters at different stages of coagulation. Our method builds on the theory of intersecting expanding contour lines from a reference object to observed objects, quantifying these as summary statistics graphs. We then analyze the evolution of these patterns over a series of images. Our theory suggests that the movement of the proteins when clustering resembles that of Brownian motion, and we try to emulate this by generating synthetic data. By initializing an image with i.i.d. noise, smoothing it with a Gaussian filter, and setting a threshold value, we create synthetic data that capture the behavior of these protein clusters during different stages of coagulation.

To refine our model we use gradient descent with graduated non-convexity to find the parameters $\{\alpha, \tau\}$, that minimize the loss between the synthesized data and the observations. Applying the optimal parameters per timeframe, our model effectively emulates real observations, offering promising potential for further investigation into the interplay between our model, and the biological process it represents.

The project is contained in the repository found here.

Contents

1 Introduction

Image analysis methods play a crucial role in numerous fields, such as medicine and biology. They allow us to unlock and quantify valuable information from visual data, which would otherwise remain unexplored. This paper seeks to provide insight into the structure and behavior of coagulating casein proteins in milk, by exploring different image analysis tools. For this purpose, we developed a method for calculating different relational shape measures of objects classified in microscopy images. This method extends upon Ripley's K-function by considering the shapes and sizes of objects, resulting in different relational shape measures describing the clustering patterns within the input image. To maximize the quality of these results, we also discuss the need for different image processing strategies, to make sure that the methods are applied on enhanced and non-distorted images.

The method explores the shape relations of individual images, however, since coagulation as a process depends on time, we examine the evolution of these patterns, as they evolve through a set of images. We examine that the movement of protein clusters seemed to closely resemble the one caused by Brownian motion. With this in mind, we develop an approach that involves generating synthetic data in the form of a smoothed Gaussian noise, which we believe is an accurate model for capturing the overall behavior of the protein clusters. By fine-tuning the parameters of the model using different parameter optimizing techniques, we are able to accurately synthesize the microscopy image at a given timestamp. We can then examine these optimized parameters at different time-frames, in order to better understand the how the spacial properties change, as the proteins aggregate

Finally, we attempt to interpret the results obtained by applying our methods to the microscopy images. We discuss the effectiveness of using the relational shape measures to understand casein coagulation and the limitations associated with our approach.

2 Background

This thesis is a collaborative effort between the Image Processing Analysis department at KU and the Food Science department at SDU (hereafter referred to as "FOOD"). The partnership with FOOD involves exploring ways to extract geometric and spatial properties from their data, specifically, microscopic imagery of milk proteins. The core objective of the project is to devise methods that can be applied to these images, generating quantitative information that FOOD can utilize as a tool in their analysis work.

In addition to the core objective, we plan to broaden the scope of this project by investigating potential methods to simulate these microscopic images. We hypothesize a correlation between the accumulation pattern of casein proteins over time and the stochastic process known as a random walk. While a detailed exploration of this process lies beyond the scope of this work, we aim to leverage its concepts as an inspiration. Specifically, our approach involves emulating these images through the application of a Gaussian filter to a normal distribution of identical, uniformly distributed pixel values, followed by the application of a threshold.

2.1 Description of data

The data provided consist of a sample time-lapse video of microscopical imagery capturing the aggregation of casein micelles in milk. In said video, the liquid milk has been presented with a rennet enzyme, and placed in an incubator at 37°C which starts a reaction where the casein micelles coagulate turning into curds. The proteins have been visualized using a fluorescent dye that binds to all the different types of proteins present in milk. Milk is comprised of numerous different proteins, and in this case, the fluorescent staining applies to all of the ones present. All of these visible proteins are suspended within the liquid whey, which acts as a water-based gel containing the proteins and lipids. This gel-like substance acts as the medium in which the aggregation process occurs.

The videos are treated as a sequence of greyscale images capturing this process, as such, each of them depicts a fixed time interval in the transition from milk to cheese. Because of the fluorescent dye, the density of proteins at a given point is represented by the hue, such that brighter spots represent areas with a high density of milk proteins, and darker spots represent the medium in which the proteins and lipids are suspended.

2.1.1 Irregularities in casein distribution

Upon visual examination of the microscopy images obtained from the video, we observe distinct patterns in the distribution of protein clusters over the course of the video. We notice a higher density of clusters at the top and a noticeable scarcity of them at the bottom. When examining sample frames throughout the video, it becomes evident that these irregularities persist as the clusters coagulate. An example of three sample frames

from the beginning, middle, and end of coagulation can be seen in Figure 1, where the irregularities at the top and bottom of the frames become apparent. Observing the figure, it is evident that the top and bottom regions stand out in contrast to the remaining images, which display relatively uniform protein distributions with evenly spaced clusters.

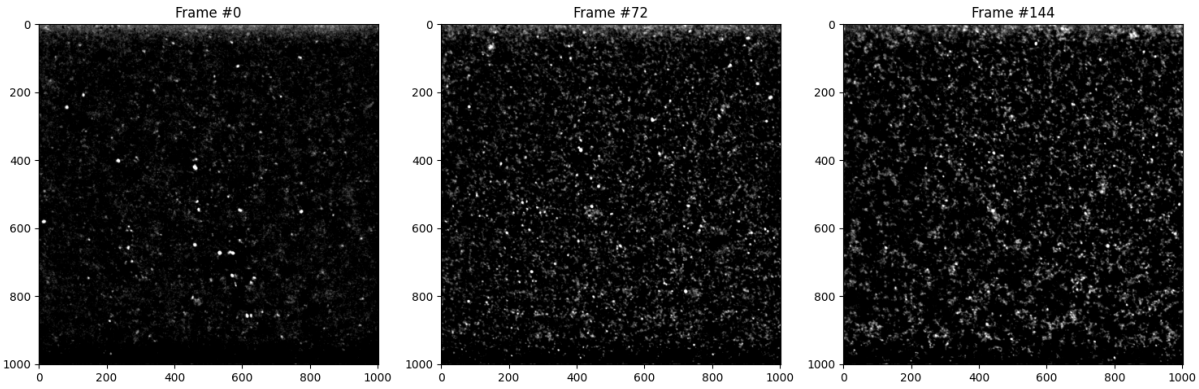


Figure 1: Protein cluster distributions throughout coagulation of unprocessed video

In order to determine whether these phenomena are simply statistical outliers and natural occurrences or the result of an error, we further examine the last frame of coagulation. To better visualize any discrepancies, we examine the total number of pixels above a given threshold for each row and column. This visualization for frame #144 of the video is presented in Figure 2, which displays two graphs. The left graph illustrates the quantity of non-zero pixels (representing protein clusters) for each row, while the right graph depicts the same information but for each column instead.

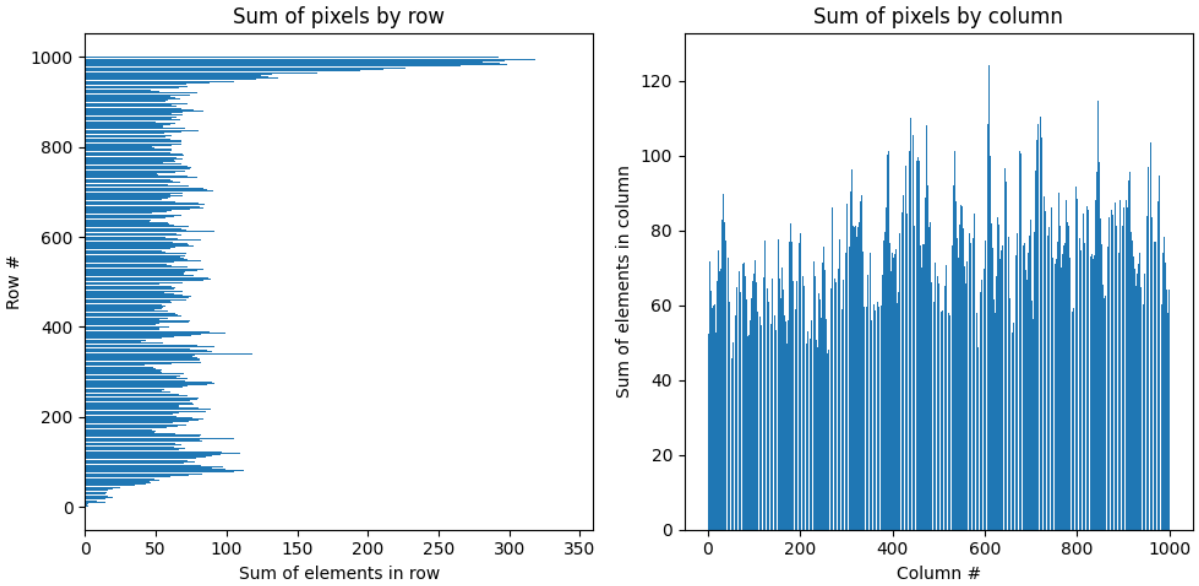


Figure 2: Pixel value distributions of frame #144 by row and column

Looking at these distributions, we observe that the average number of non-zero pixels

for both rows and columns varies between $100 - 200$. It becomes apparent that this distribution remains consistent across the entire range of columns. However, in the case of rows, there is a significant spike of approximately 800 at the top and a lack thereof at the bottom. Due to these irregularities in distribution being so significant, it appears unlikely that these phenomena can be explained as being statistical outliers. Instead, they are likely the result of errors in the experiment or in how the video was originally processed before we received it.

2.1.2 Fluorescent dyeing of proteins in milk

The microscopy imagery analysis crucially relies on the utilization of a fluorescent dye for visualization purposes, however because of the attributes of the dye used it binds to all of the proteins present in the milk. This results in other unwanted proteins (such as whey proteins) being stained as well by this method. However, because the fluorescence intensifies when the proteins aggregate, the contrast between the casein micelles and the unwanted proteins in the background is very high. Furthermore, the unwanted proteins are mobile in the liquid phase of milk, and as such, move too fast for the imaging systems to reliably capture. This difference in intensity between the proteins, combined with the size and mobility of smaller proteins should ensure that the casein structures stand out prominently in the images. However on the off chance that something gets trapped in the structures formed by the caseins, then their fluorescence would contribute to the structures captured by the imaging systems.

While we did not have the resources to examine this, if one were to expand the scope of the project it might be relevant to explore to what degree this combination of data and visualization creates any unwanted noise. However for the purposes of this project we assume that the possibility of unwanted proteins interfering with the casein micelles, in a magnitude great enough to be picked up by the microscope is negligible, and as such can be ignored.

2.2 Molecular structure of milk

When trying to model the coagulation of milk we need to understand the molecular structure of this complex biological fluid. The body of milk in the liquid phase contains fatty lipids, casein micelles, and a mix of lactose, whey proteins, and some minerals dissolved in water. This water-based gel comprises the largest proportion of milk and accounts for approximately 85% of the composition, it acts as a solvent, containing the different nutrients. Milk also contains lipids, which are primarily present in the form of triglycerides, which consist of fatty acids esterified to a glycerol backbone. These lipids serve as one of the main energy sources and appear in the form of big globules in the milk.

The last major component in the molecular structure of milk is the caseins which are a group of proteins found in milk that constitute a significant proportion of its protein

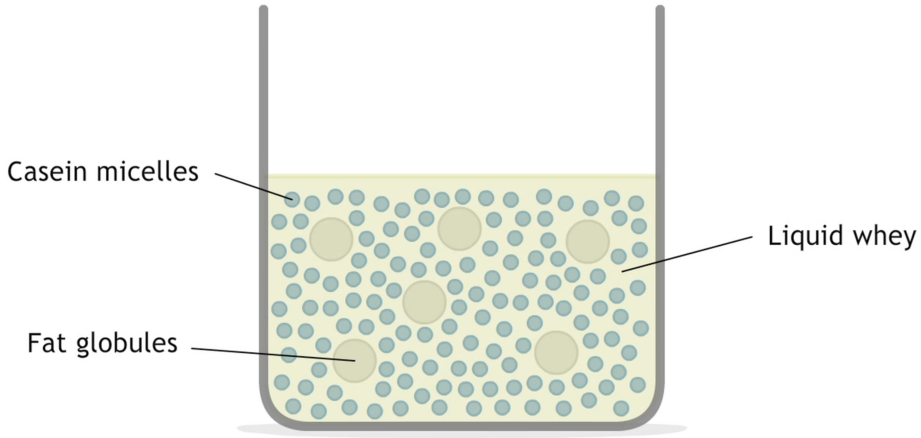


Figure 3: A molecular view of cheese[1]

content. Casein is a collective term encompassing several individual proteins such as α-casein, β-casein, and κ-casein, among others. These proteins aggregate to form casein micelles, which are colloidal structures that are dispersed in the milk. Casein micelles are composed of thousands of individual casein proteins held together by calcium phosphate complexes. This arrangement imparts stability to the micelles and ensures their colloidal suspension in milk.[1]

2.2.1 Transition to cheese

During the cheese-making process, coagulation occurs through the aggregation of casein micelles. This natural process is facilitated by various factors, such as pH and temperature changes, as well as the addition of rennet enzymes. The rennet enzymes interact with the κ-casein component of the casein micelles, causing the micelles to undergo structural changes and aggregate together. The coagulation process observed in the video results from the initiation of this enzymatic reaction. It leads to the coagulation of casein micelles, forming a network within the milk that traps water, fat, and other components. This network, combined with the expulsion of whey, contributes to the formation of curds. The curds then undergo further processing, including pressing and aging, ultimately resulting in the production of cheese.

2.3 Image processing- and analysis pipeline

The process of analyzing and synthesizing data, which is described throughout this report, can be compressed down to four general steps, those being: preprocessing, calculating the relational shape measure, generating synthetic images, and optimizing parameters for the model. This pipeline is visualized in Figure 4, and works as follows.

1. The initial step of the pipeline involves preprocessing each image in the video. This preprocessing stage includes noise reduction, thresholding, bias correction, and labeling, with the objective of reducing noise and enhancing the overall image quality.

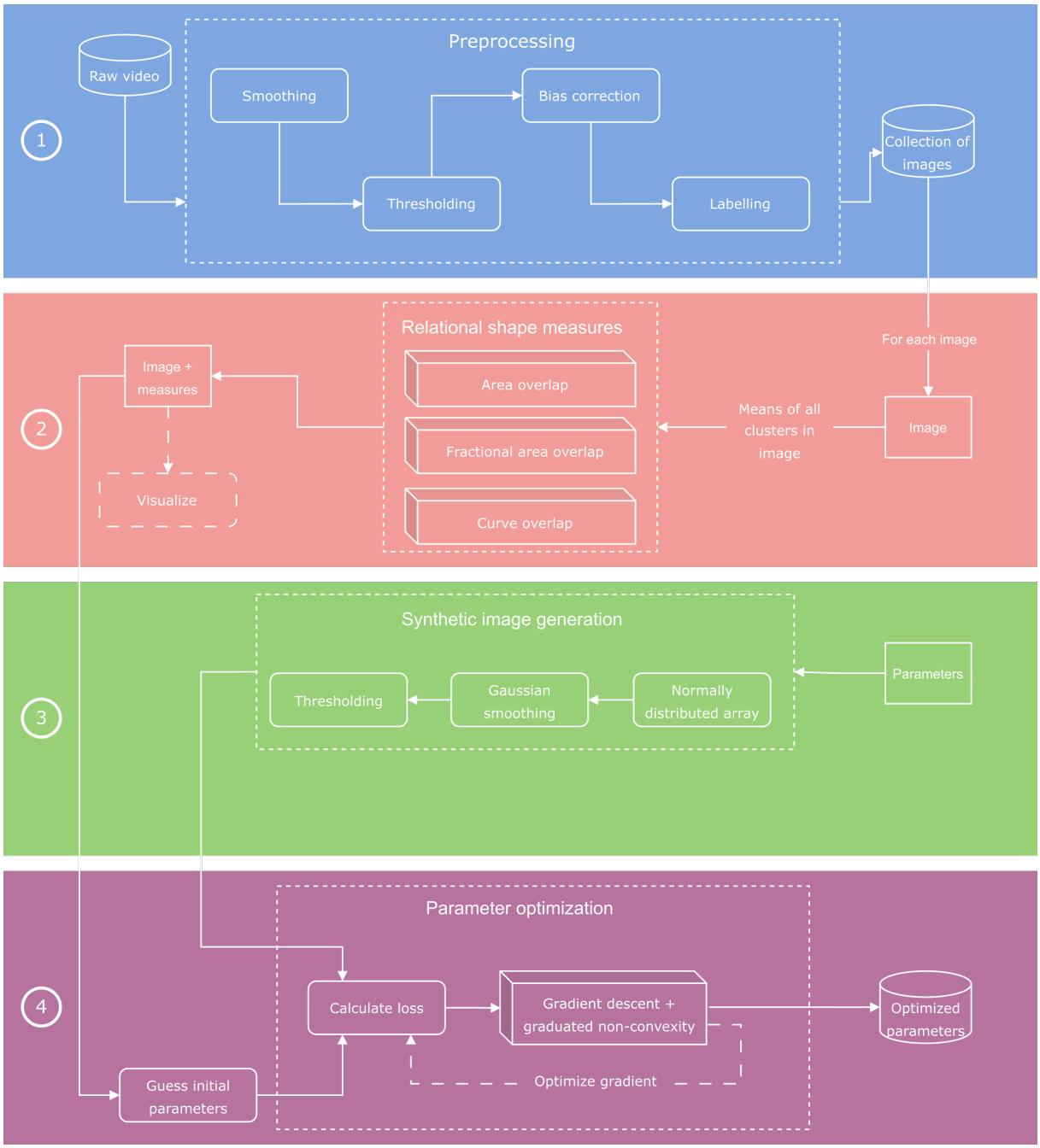


Figure 4: Image processing- and analysis pipeline

These operations are implemented to guarantee that subsequent analysis steps are conducted on high-quality, standardized images.

2. The relational shape measures capture the spatial characteristics and patterns of protein clusters in the images. This measurement quantifies the spatial relationships between protein clusters, providing insights into their distribution and aggregation. By calculating the relational shape measure, we can perform a quantitative assessment of the protein coagulation process, which can be visualized or utilized in the final step of the pipeline.

3. Once the relational shape measure is computed, the next stage is to generate synthetic images, by simulating the protein coagulation process. This is done using a 2D filled with independent and identically distributed (i.i.d.) normally distributed noise. The noise is then smoothed using a Gaussian kernel and subsequently thresholded. The synthetic images can then be adjusted to vary cluster size and amount by modifying the parameters α and τ, which determine the standard deviation of the smoothing, and the threshold respectively. By systematically varying these parameters, a diverse range of synthetic images is generated to represent different scenarios of milk protein coagulation.

4. The final step of the pipeline involves parameter optimization to determine the optimal set of parameters that yield synthetic images closely resembling the real images. This optimization process employs gradient descent to calculate the mean squared error between the fractional area overlap of the real image and the model. In this case, the model refers to the relational shape measure of the synthetic data, with parameters that closely resemble the real image. The optimization employs the graduated non-convexity algorithm in order to combat the optimization locating sub-optimal local minimum. The end result is an optimized set of parameters $\{\alpha, \tau\}$ for each frame of the video, resulting in relational shape measures that closely resemble those of the real data.

3 Methodology

3.1 Image processing

Image processing involves applying a series of transformations and techniques to improve the quality and enhance the information contained within an image, before feeding it into an algorithm or model. Therefore, the objective of image preprocessing is to enhance the accuracy and robustness of subsequent image analysis.

To process the provided video, we start by converting them into a sequence of individual images, with each frame from the video represented as a separate image. Subsequently, we apply several image processing techniques to address any distortions present in the microscopy images, potentially caused by external factors like uneven light distribution. This means that after preprocessing the images, we should be left with a clearer and more accurate representation of the protein structures, which we can use to identify the borders of each cluster of milk proteins more precisely. However, this process can be demanding due to the challenges of distinguishing milk proteins from other substances in the image. These challenges arise from the similarity in shades of grey and the presence of light reflections from the microscope.

Once most of the imperfections are removed and the protein clusters are identified, the statistical methods can be applied to extract quantifiable information or descriptors from the images that can aid in the analysis of protein samples. In the following sections, we will go over various preprocessing techniques relevant to this project in further detail.

3.1.1 Noise reduction

Every image has some sort of noise associated with it. The term noise refers to the unwanted random variations or disturbances that occur in the pixel values of an image. Noise is usually caused by factors such as environmental conditions, sensor limitations, or other electronic components used in image-capturing devices. This is particularly relevant in microscopy images since image acquisition through a microscope requires an extensive amount of magnification, which introduces a lot of uncertainties and imperfections, making it harder to analyze or interpret. To handle the issue of noise in microscopy images, several techniques can be employed, such as noise filtering, image averaging, etc. Among the most common noise filtering methods are median and Gaussian smoothing. As the name suggests, median smoothing is simply where the value of a pixel is determined by the median of the neighboring pixels. Gaussian smoothing, on the other hand, determines the value of a pixel based on a weighted average of the surrounding pixels. In a two-dimensional space, these weights are determined by a bivariate normal distribution, which roughly means that the closer a pixel, the higher the impact.

Median filtering can be seen as a more robust method since outliers don't affect the result

as easily compared to Gaussian filtering. As an example, if the value of a pixel is wrongfully set to infinity, Gaussian filtering sets the value of all surrounding pixels to infinity as well, while median filtering remains unaffected. Although this is a general concern, this isn't an issue in relation to our data, since pixel values are represented with floating point values, bounded by 0-1. On this basis, we have chosen to perform Gaussian smoothing to remove noise in the microscopy data, to stay consistent with the Gaussian filtered synthetic data introduced later in this report. Choosing the right level of smoothing can be a challenging task due to the trade-off between noise suppression and preservation of important image details. Additionally, the provided microscopy images don't seem to contain a lot of noise, which could indicate that the images already have been through some sort of noise reduction process. Based on this, we assessed that applying a low level of smoothing is sufficient for our data.

3.1.2 Thresholding

In the context of studying the coagulation of milk proteins turning into cheese, thresholding enables the extraction of protein regions from the background, enabling us to perform our spatial analysis. When thresholding an image, we simply take all pixels with values below a certain threshold and set them to 0, setting the values of all other pixels to 1. If we instead wanted to perform spatial analysis on the background as its own class, instead of the proteins, we would simply choose to nullify all pixels with values above the threshold instead. Choosing an ideal threshold for image segmentation can be challenging, especially when relying solely on visual inspection. In the case of analyzing microscopy images of milk protein coagulation, the difficulty lies in the absence of well-defined intensity boundaries between the proteins and the background. Unlike scenarios where there are distinct intensity peaks or clear separability, the coagulated proteins exhibit varying intensities, shades, and textures that blend with the surrounding regions.

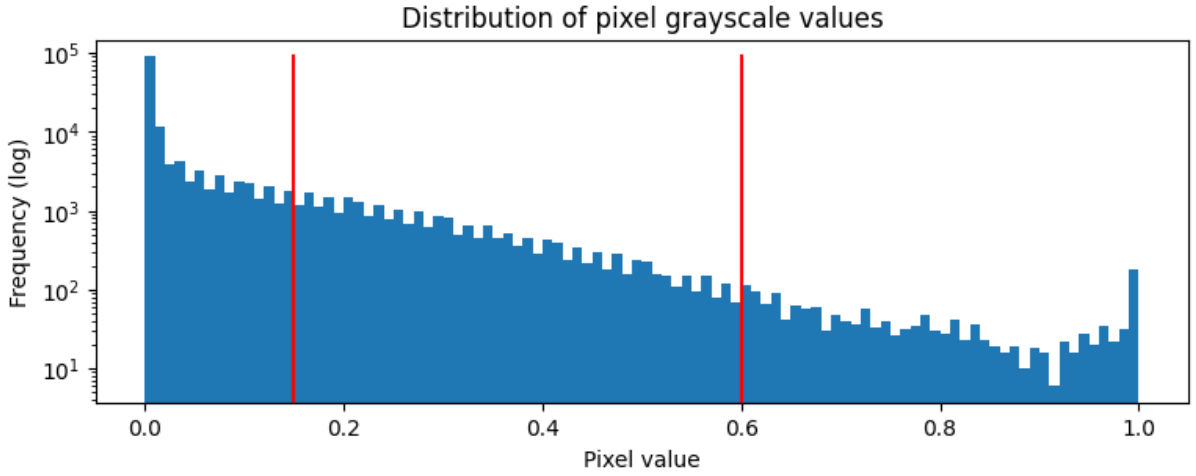


Figure 5: Histogram of pixel values in image with thresholds

In Figure 5, a histogram over the varying pixel values of the image can be seen. Here it can be noted that there is no clear cutoff point that signals the difference between what is a protein and what is not. Because of the gradient transition from what would be classified as a protein and what would be classified as a background, we chose to segment the images into three distinct classes. One represents the background, one represents what is surely a protein, and a class for that which cannot with certainty be categorized as either a protein or the background. Because of this division into three classes rather than a binary divide, we need to specify two thresholds, instead of one. However, simply choosing these ourselves can be problematic, since the subjective nature of visual assessment introduces inherent biases and inconsistencies, i.e., different individuals may perceive boundaries differently, leading to inconsistent threshold selections. This lack of objectivity hampers the reproducibility and reliability of the analysis.

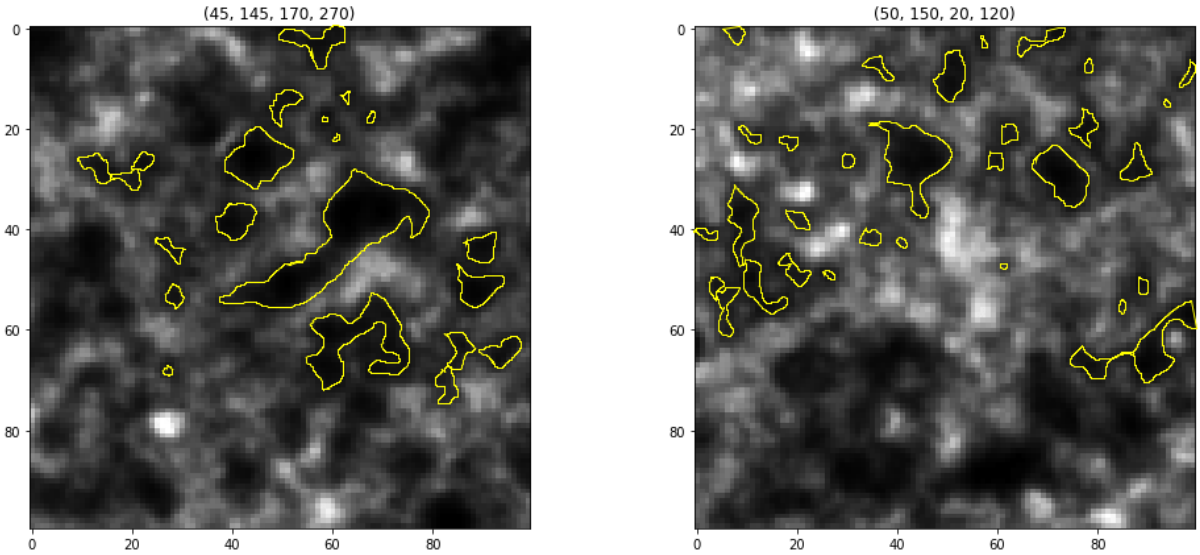


Figure 6: Examples of annotations of thresholds

Instead, we obtained some examples of expert segmentation of a sample image, manually thresholded by FOOD, to which we could then compare different numerical thresholds in order to find fitting values. Looking at Figure 6, we see two examples of annotated thresholds of different images. From these images, we determined the two thresholds which can be seen in Figure 5, and an example of a segmented image can be seen in Figure 7.

Because we are only interested in what can be definitively classified as a protein cluster, and in order to minimize the error created by external factors, we choose to focus solely on this first class of pixels, which clearly classify proteins. This approach results in a large class of 'potential proteins' that is not included in the further spatial analysis due to the uncertainty surrounding their classification. As shown in Figure 5, the number of pixels that represent protein clusters is significantly outnumbered by the number of pixels that could not be classified. The size of this class of uncategorized pixels could be attributed to various factors, such as potential variations in lighting conditions, noise, or other image

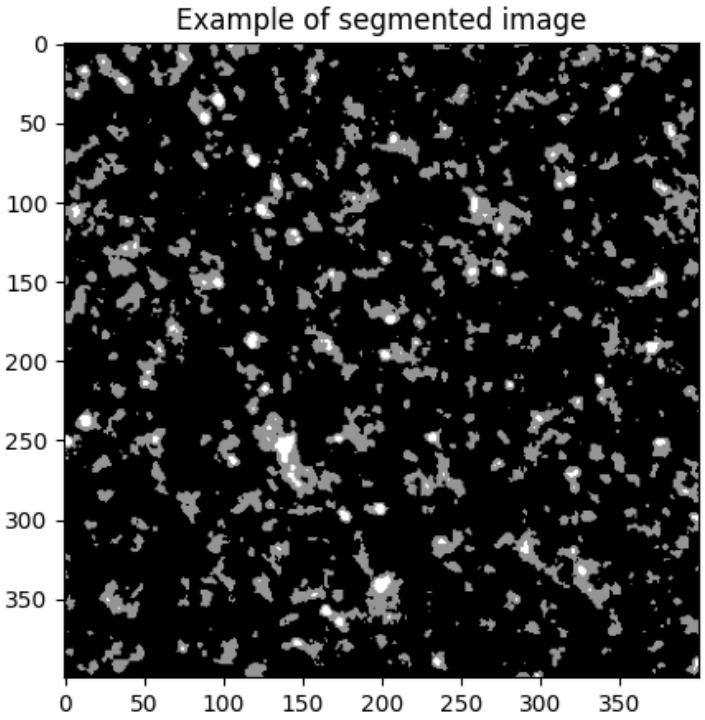


Figure 7: Visualization of extracted clusters with multiple thresholding

artifacts that can compromise the overall image quality.

3.1.3 Bias correction

Due to the nature of the microscopic images, non-uniform light distribution may cause some parts of the image to appear darker than others, making it difficult to distinguish clusters of milk proteins accurately. To overcome this challenge, we apply bias correction by using a polynomial function to balance the brightness levels in the image, giving us more accurate and reliable results.

In practice, to apply bias correction we use our image as input, and a mask based on a 2nd-order polynomial function. The mask consists of the position of all pixels which meet our set threshold, as we are only interested in computing a bias field based on pixel values we believe actually show milk proteins, and not noise or background. The polynomial function we want to fit, should be slowly increasing, in order to capture the pixel intensity level of an area, otherwise known as an *intensity based* method for bias correction.[5] We use a Vandermonde matrix for this purpose, as it provides a structured way to set up the system of linear equations that can then be solved to find the coefficients of the polynomial. The process of fitting a polynomial function to our image starts by creating a Vandermonde Matrix, which is a matrix where each row contains powers of the input vector. In this case, the input vector is $(\mathbf{r}, \mathbf{c})$ where r, c are the rows and columns in our mask.

$$r, c = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_m \end{bmatrix}, \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{bmatrix}, \quad \text{Vandermonde} = \begin{bmatrix} 1 & r_1 & c_1 & r_1^2 & r_1 \cdot c_1 & c_1^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & r_m & c_m & r_m^2 & r_m \cdot c_m & c_m^2 \end{bmatrix} \tag{1}$$

The way we apply the matrix is by storing the intensities of the pixels corresponding to the mask in another matrix **Values** and applying a least squares method to the equation

$$\text{Vandermonde} \cdot \text{Coefficient} = \text{Values}$$

From this, we derive a 2nd-order polynomial function, which can model the bias field, which is an array of the overall pixel intensities at different areas of our image.

$$f(r, c) = a_0 + (a_1 \cdot r) + (a_2 \cdot c) + (a_3 \cdot r^2) + (a_4 \cdot r \cdot c) + (a_5 \cdot c^2)$$

Having obtained the coefficients vector, we can apply the polynomial function to the entire image, constructing a similar Vandermonde matrix for all pixels, and multiplying it by the coefficients. The result of this operation is image J, which represents the estimated bias field across the entire image.

This bias field is then used to correct the original image, subtracting the original by the bias field and adding the mean of the bias field to retain the intensities in the image. The resulting corrected image has balanced brightness across all areas, with brighter regions dimmed and darker regions enhanced. By applying this bias field correction, we make the milk proteins more distinguishable and consequently improve the quality and reliability of our analysis.

3.1.4 Labeling of clusters

Labeling refers to the process of assigning unique identifiers to each connected region in a binary image. Connected regions consist of a set of adjacent pixels, which denote the objects we wish to analyze. Labeling is usually done by iterating through all pixels in the image and assigning a label based on the previously labeled pixels. If no previous pixels are within the given connectivity, the pixel is given a new identifier, meaning that it doesn't belong to any of the previously encountered objects. In a two-dimensional space, one has to choose between one-connectivity or two-connectivity. In one-connectivity, pixels are neighbors to every pixel that touches one of their edges, while two-connectivity also includes diagonal pixels i.e. pixels that touch one of their corners, which is what we have opted for.

In practice, one iteration through all pixels often isn't enough. Connected regions come in all kinds of different shapes, which means that adjacent pixels may end up being assigned different labels after the first pass. Thus a second pass is usually required to ensure that

all pixels belonging to the same connected component have the same label. After the labeling process has been applied, useful properties can easily be computed, such as their area, centroid, or bounding box.

3.2 Spatial image analysis

Spatial analysis methods describe a variety of techniques that are used to quantify and interpret spatial patterns and their relationships. These methods serve to give an insight into the distribution, clustering, and interactions of objects or events. One commonly used spatial analysis method, which our implementation was heavily influenced, is Ripley's K-function.

3.2.1 Ripley's K-function

Ripley's K is a widely used statistical method in spatial analysis that serves to examine the interactions between events as point patterns. Typically, it is utilized in the analysis of point patterns to assess whether they are randomly distributed or if they exhibit some sort of clustering or dispersion.

Calculation Ripley's K-function can be expressed as

$$K(r) = \frac{1}{\lambda} E\left[N(r)\right]$$

Here, r is the distance parameter, and $N(r)$ is the number of points within radius r from a given point. The intensity of points within the observed area is denoted by λ, and is used to normalize the number of points to account for their density. This normalization ensures that the clustering or dispersal tendency of the data points remains unaffected, regardless of their quantity. For stationary data, λ can be calculated by dividing the number of points N by the study area A such that

$$\lambda = \frac{N}{A}$$

Furthermore, $E\left[N(r)\right]$ represents the expected number of points within radius r from a randomly chosen point. If edge corrections are not taken into account, $E[N(r)]$ can be calculated by averaging the pairwise number of points within r distance from each other

$$E[N(r)] = \frac{1}{N} \sum_{j} \sum_{i} I(d_{ij} < r)$$

Here, d_{ij} represents the distance measure between points i and j (usually the Euclidean distance). The function $I(x)$ is an indicator function that evaluates to 1 if x is true and 0 otherwise, which in this case corresponds to including point j if it is within a distance of r from point i.

Interpretation of Ripley's K-function The Ripley's K-function is a cumulative function when plotted, since an increase in the distance parameter r guarantees at least the same number of points to be within radius r from any arbitrary point. A comparison of this plotted graph with the expected value of Ripley's K-function under complete spatial randomness, which follows a Poisson distribution and is given by $K(r) = \pi r^2$, can reveal clustering or dispersion of the points. Function values above this curve imply clustering, while function values below imply dispersed data points.

However, classifying data as either clustered or dispersed based on a hard threshold is not optimal, since randomly distributed points will also end up being labeled as one of the two. Additionally, noise in the data can cause inaccuracies, which can lead to misclassifications. As a result, a confidence envelope is used in practice to represent a certain degree of uncertainty in our analysis. Function values that fall inside this confidence envelope are considered inconclusive, meaning that we cannot confirm or deny the clustering of points [3, pp 1796–1803].

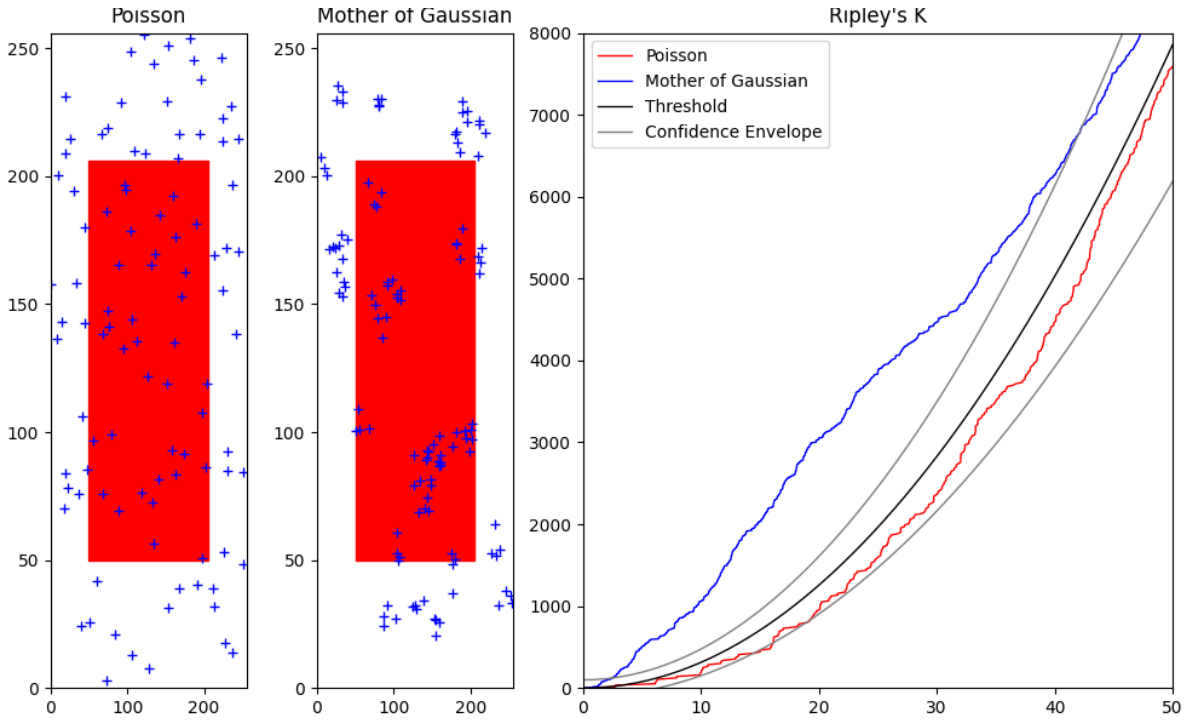


Figure 8: An example of two datasets being classified by Ripley's K-function

Figure 8 provides an example of calculating Ripley's K-function for two simple datasets. Here the two distributions can be seen on the left, where Ripley's K-function is calculated for only the points inside the red squares. As r increases, we see the cumulative functions increase in value on the right. To determine whether the two datasets are clustered or dispersed, one would check if the K-functions are above or below the threshold given by $K(r) = \pi r^2$. In this example, the dataset created from the Mother of Gaussian would be classified as being clustered. Whereas the Poisson data would either be classified as dispersed, or as inconclusive, depending on whether the confidence envelope is used or not.

Limitations Even though Ripley's K gives valuable insights into the clustering or dispersion patterns, it also has some limitations that need to be considered when interpreting the results. The study area of the spatial pattern is often determined by artificial boundaries. As a result, points near the bounding edges of the study area will on average have fewer points within r distance from them, since points outside the bounding edges are not taken into consideration. However, Ripley's K function proposed above doesn't take these edge effects into account. There exist different approaches to mitigate the bias near the boundaries, with the most common involving a weight term that is multiplied inside the double summations of $E[N(r)]$.

Another important limitation arises from the fact that Ripley's K-function is strictly a point process analysis tool, meaning that it is not suitable for analyzing geometric objects. One could try to apply Ripley's on the centroid of each object in an image, but this would completely ignore valuable information, such as the shape and size of each object.

3.2.2 Relational Shape Measures

Due to the aforementioned limitations of Ripley's K-function, which disregard the relational shapes of objects, we deemed it as not fully fitting the goals of the project. Instead, we have developed a new method, a modification of Ripley's K-function, which incorporates the shape measures of objects as well, adding to its complexity. Our algorithm operates much like Ripley's K, but instead of calculating the number of points within a given distance from an arbitrary point, our method labels collections of adjacent pixels as objects, and then defines different measures from the reference objects $Y \subseteq \mathbb{R}^2$ to the remaining observed objects $X \subseteq \mathbb{R}^2$ at a specified distance.

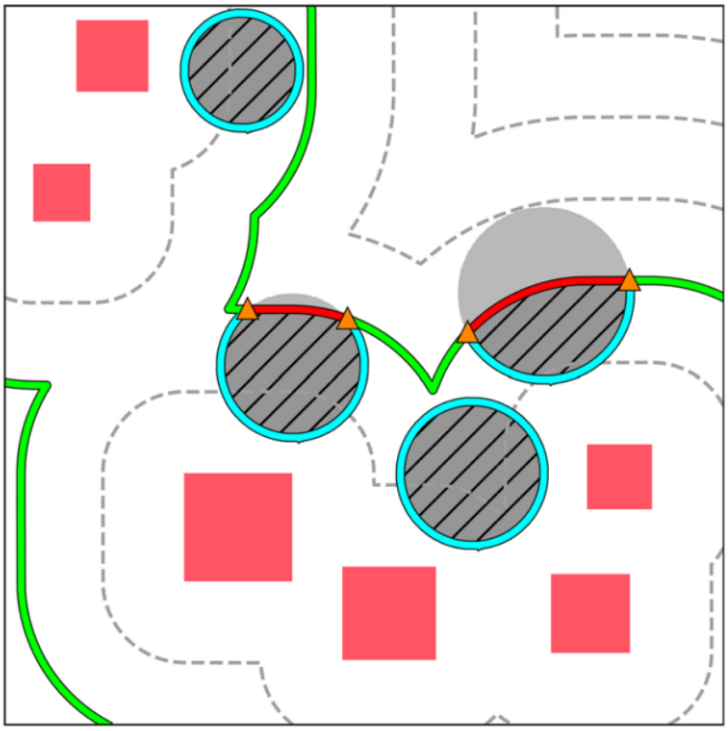


Figure 9: Visualization of the different shape measures.
Area overlap (striped grey area), Curve overlap (red lines), Radius (green lines)

Figure 9 visualizes these different measures, using basic two-dimensional geometric shapes such as circles and squares for the sake of simplicity. The reference objects are depicted as red squares with equidistant curves in the form of contour lines emerging from them (see dotted and green curves). The Figure shows how these distance curves interact with the observed objects, which are depicted as grey circles.
The first measure our method addresses are the area overlap. More formally, it can be expressed as the area overlaps between the observed objects and the set of points within the bounds of the equidistant curve. This can be written more concisely as $X \cap Y^r$, where Y^r denotes the set of all points within distance r from Y, which is given by

$$Y^r = \{\alpha \in \mathbb{R}^2 \mid \inf_{y \in Y} d(\alpha, y) \leq r\}$$

Here $d(\alpha, y)$ is the Euclidean distance between α and y. This is shown by the striped grey area within the circles. The other measure is given by the curve overlap between a contour line and the observed objects, which in Figure 9 is shown by the red line, i.e. the intersection of observed objects and the green line. This can be summarized in set notation as $X \cap \partial Y^r$, where ∂Y^r represents the boundary of Y [6]. We considered these measures as being sufficient for the purpose of our analysis, although this method can easily be extended to deal with other Hausdorff measures as well, such as the circumference of the observed objects within the contour lines (the turquoise line) or the number of intersection points (the orange triangles). Moreover, this method can easily be extended to deal with objects of higher dimensions.

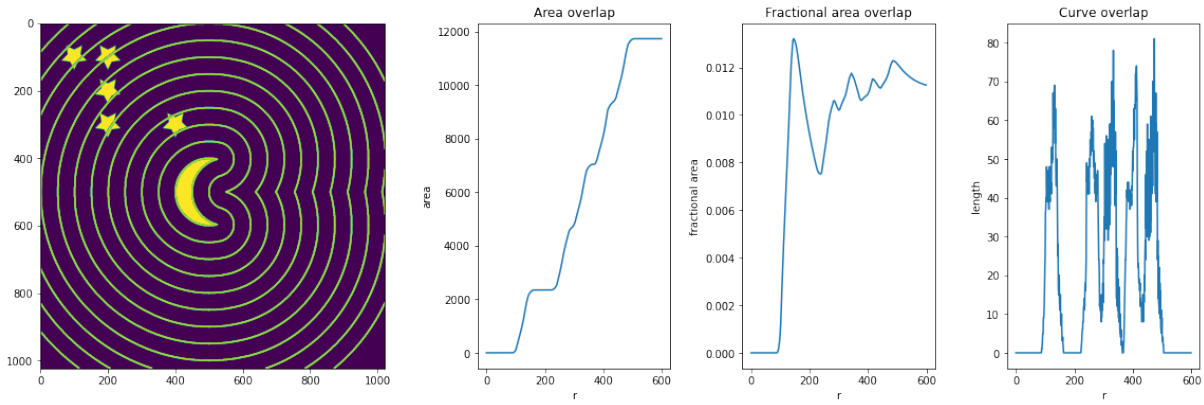


Figure 10: Example of quantified relational shape measures on synthetic data

To quantify these measures across multiple equidistant curves, we chose to plot them as summary statistics graphs, with the x-axis denoting the distance r to the nearest reference object. An example of these quantified shape measures on synthetic data is seen in Figure 10. The left-most image shows the coordinate space, on which the three graphs to the right are based. The moon and stars denote the reference and observed objects respectively, with contour lines shown at every 50th pixel from the reference object. Each graph represents the relation between the moon and the stars according to different shape measurements. The first graph denotes the area overlap, which is non-decreasing since it is impossible for the area of the objects within the contour lines to decrease when r is increased. I.e. we have: $r_1 < r_2 \implies f(r_1) \leq f(r_2)$. The second graph is the fractional area overlap, and is simply computed by dividing the area overlap by the total area within each contour line, which means that: $0 \leq f(r) \leq 1$. Lastly, the third graph represents the curve overlap, which can be thought as the instantaneous rate of change of the area overlap. This means that it can simply be found by computing the derivative of the area overlap. By quickly scanning the graphs, it becomes obvious that they are closely correlated to the moon and stars in the coordinate space. As an example, all three graphs seem to increase drastically at $r \approx 100$, which represents the distance to the closest star.

Algorithm 1 shows the code for deriving these graphs in pseudo form. The algorithm is somewhat self-explanatory. We start with an array of labeled objects, which is determined by the image processing phase, and a max_r value, which is set to a suitable number based on the specific analysis and its goals. Then we create masks for both references and observed objects, which are used to construct a Euclidean distance map to the reference objects. We then calculate the area overlap based on the intersection between the mask for the observed objects and the distances less than r and save those results to the corresponding indices of *area_ overlap*. Finally, we utilize *area_ overlap* to calculate the two remaining graphs, as previously described.

Algorithm 1 Relational shape measures algorithm

1: *label_image* = labelled image array
2: *max_r* = maximum distance to consider from reference objects
3: *area_overlap* = initialize array for area overlap at distance r
4: *area_total* = initialize array for total area at distance r
5: *ref_mask* = create mask of reference objects in *label_image*
6: *obs_mask* = create mask of observed objects in *label_image*
7: D = calculate Euclidean distance map from the reference objects
8: **for** $r = 0$ to *max_r* **do**
9: *dist_mask* = create mask of all distances less than r
10: *area_overlap*[r] += count entries which are TRUE in both *dist_mask* and *obs_mask*
11: *area_total*[r] += count TRUE entries in *dist_mask*
12: $frac\_area\_overlap$ = *area_overlap* divided by *area_total*
13: *curve_overlap* = calculate derivative of *area_overlap*

3.2.3 Addressing edge effects

Edge effects are an essential problem encountered in spatial analysis. It arises from the fact that observations close to the edges of the image have fewer neighboring points since points outside the boundary are not considered, resulting in analysis outcomes near the edge that differ from the interior. By factoring in edge correction, we can mitigate the bias caused by the border effects, thereby providing more accurate results. Various methods have been developed to tackle this issue, and they vary a lot in complexity. One approach is to mimic the observed point pattern outside the boundary, i.e. simulate artificial data that behaves the same way as the observed data. Another approach is to gradually decrease the influence of observations toward the edges of the study area. Arguably the simplest way to address edge effects is to create another bounding polygon or buffer within the study area and focus only on observations within this polygon, while interactions with points outside the polygon are still taken into account. In this project, we will be working with the last-mentioned approach, mainly due to its simplicity, but also because the images are large enough, that the entirety of them are not needed for reliable and consistent results.

3.2.4 Applying method to microscopy data

In the context of the project, which aims to explore the process of casein micelle aggregation, the imagery consists of only one class of data. This means that both reference and observed objects refer to these clusters of proteins, which are classified during image preprocessing. Figure 11 shows the shape measures, where a randomly chosen cluster acts as the reference object, and the remaining clusters as the observed objects. Note that a bounding box (the red square) has been added to account for the bias at the edges.

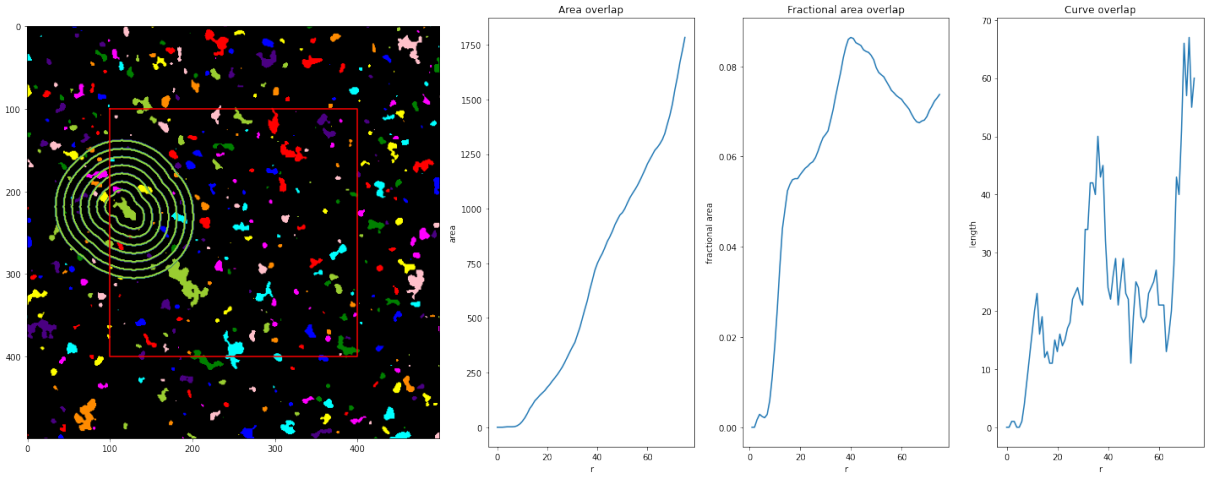


Figure 11: Example of quantified relational measures on a random protein cluster

Selecting a random protein cluster and calculating its shape measurements does not reveal much about the underlying distribution of the entire image, since there is a high degree of randomness associated with the environment of a single cluster. Therefore, we opted for a more comprehensive approach that involves calculating the average shape measures of all clusters within the buffer. By doing so, we obtain more smooth curves, that better capture the overall clustering pattern (See Figure 33). The interpretation of these results will be examined and explained further in the subsequent sections.

The extended algorithm, which incorporates these changes, is presented as pseudocode in Algorithm 2. Besides the labeled image and max_r, the algorithm now also takes in a variable L, which determines the distance from the edges to the inner buffer. From this, a new array of labeled clusters is created, which simply ignores all clusters fully or partly outside the bounding box. Then the average area overlap measure of the clusters is computed, by summing up the area overlap for each cluster, and dividing by the total number of clusters within the buffer. The code for deriving the last two graphs remains the same as for the genetic algorithm.

Even though the area overlap curve is the one that best resembles the graph from Ripley's K, we assess that the fractional area overlap curve is the most effective at describing the clustering patterns in the image. Therefore, we have chosen to mainly focus on said curve in the remainder of the analysis.

Algorithm 2 Extended relational shape measures algorithm

label_image = preprocessed image
max_r = maximum distance to consider from reference objects
L = lower bound for bounding box
M = calculate upper bound for bounding box
bounded_labels = empty array for clusters within bounding box
for *cluster* in *label_image* **do**
    if *cluster* is fully within bounding box **then**
        Add *cluster* to *bounded_labels*
area_overlap = initialize array for area overlap at distance r
area_total = initialize array for total area at distance r
for *cluster* in *bounded_labels* **do**
    ref_mask = create mask of reference *cluster*
    rem_mask = create mask of remaining clusters
    D = calculate Euclidean distance map from the reference *cluster*
    for $r = 0$ to *max_r* **do**
        dist_mask = create mask of all distances less than r
        area_overlap[r] += count entries which are TRUE in both *dist_mask* and *rem_mask*
        area_total[r] += count TRUE entries in *dist_mask*
area_overlap = *area_overlap* divided by the total number of clusters in *bounded_labels*
area_total = *area_total* divided by the total number of clusters in *bounded_labels*
$frac\_area\_overlap$ = *area_overlap* divided by *area_total*
curve_overlap = calculate derivative of *area_overlap*

3.3 Synthetic data generation

Synthetic data generation refers to the process of creating artificial data that simulates real-world data. It is useful in many fields including machine learning, artificial intelligence, and data analysis. It can be used for many purposes, such as training and testing data, improving models, and mitigating bias. It can also be used, as previously mentioned, to combat edge effects in spatial analysis, or when dealing with limited or sparse data. Lastly, synthetic data can also be used to gain insight into the distribution and properties of the data, which is the primary motivation for generating synthetic data in our project.

Choosing the appropriate model for mimicking the provided microscopy images can be quite challenging, due to a trade-off between simplicity and accuracy. Generally, when selecting a model, it is preferable to opt for one that is able to closely imitate the behavior of the desired image, while at the same time being relatively simple, such that valuable information can be extracted more effortlessly. With that in mind, we chose to create our synthetic data based on smoothed Gaussian noise. Gaussian noise is a powerful and fairly simple technique used in various fields such as computer graphics and image processing. By smoothing this noise, we are able to imitate the clustering tendencies of our microscopy data. This is due to the presence of spatial correlation which occurs within closely situated pixels after being smoothed. Smoothing acts as a form of spatial averaging, where the new value of a pixel is influenced by the values of its surrounding pixels, which results in bundles or clusters of pixels with similar values.

However, the clustering behavior of the smoothed Gaussian noise depends on the parameters used by the model, such as the smoothing degree. Therefore, in order to find the ideal parameters and thereby create an accurate model for the microscopy data at different timestamps, we make use of different parameter optimization techniques based on the shape measures of each image. These optimization techniques are elaborated further in the subsequent sections.

3.3.1 Synthesizing Gaussian distributed noise

We generate synthetic data via a four-step process, which is depicted in Figure 12 and concisely outlined in the Image Processing Section 3.1):

1. We first populate an image with random noise, following a normal distribution $X \sim \mathcal{N}(0, 1)$.

2. A Gaussian filter is then applied to smooth the image, where the degree of smoothing is controlled by standard deviation α.

3. Next, we use a thresholding technique to convert the smoothed image into a binary matrix, where all values below a certain threshold τ are set to zero, and all exceeding values are set to one.

4. Finally, we label the binary image, assigning each group of adjacent pixels a unique integer to define distinct synthetic clusters.

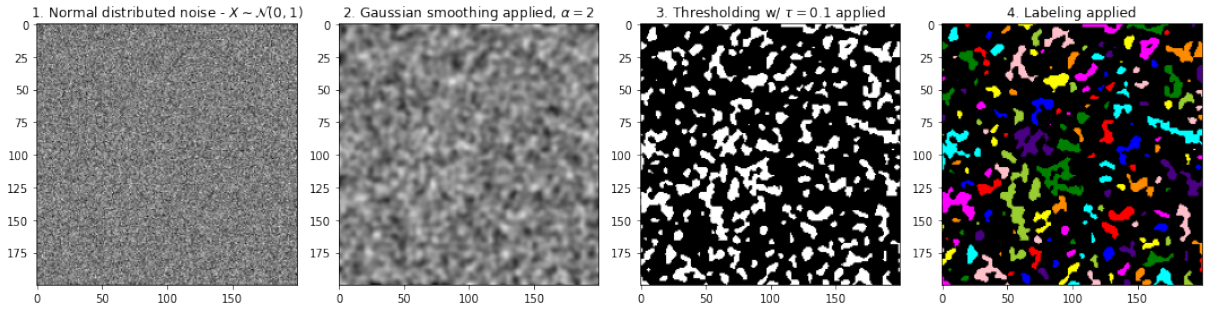


Figure 12: 4-step process for generating synthetic data

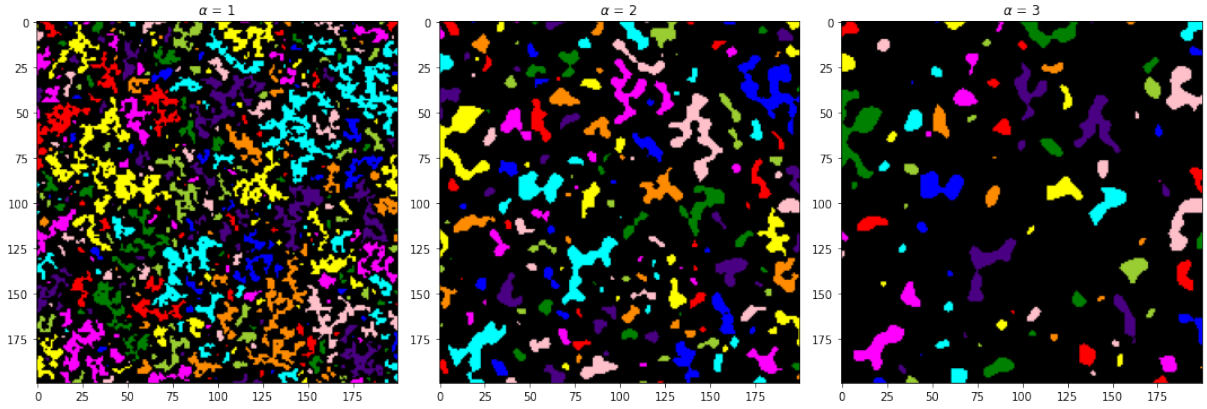
These four steps cumulatively produce a synthetic image that, while rooted in randomness, presents discernible structure and clustering. In the following sections, we will delve into a deeper exploration of the parameters $\{\alpha, \tau\}$, illustrating their roles in influencing the formation of clusters in synthetic data.

3.3.2 Impact of smoothing parameter

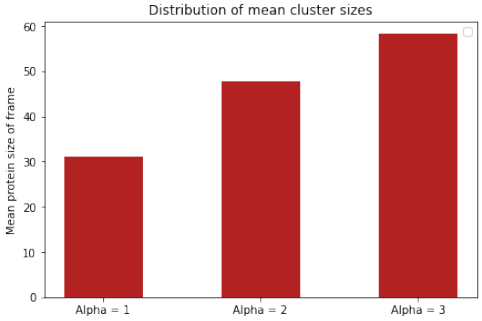
The Gaussian filter works by replacing each pixel in the image with a weighted average of its neighboring pixels. The weights are determined by the standard deviation α, which ultimately decides the weight distribution of each pixel in relation to how close it is to the center of the kernel. In Figure 13a we have applied a Gaussian filter to the same random noise image for three different values of α (1,2,3), afterward they have all been threshold by $\tau = 0.1$ and labeled. The goal of the three images is to create an intuition as to the characteristics of the clusters which form by adjusting the smoothing parameter.

Observing the images in Figure 13a, we can see that the variation of the standard deviation from the Gaussian filtering, even when the threshold is constant, results in very different images. Looking at the images left to right, we see that when $\alpha = 1$, the smoothing effect is localized, primarily influenced by the directly neighboring pixels. This scenario allows both large and small clusters to form. Small clusters emerge when a group of closely situated pixels has high positive values, effectively resisting the pull towards zero from less adjacent, negatively valued pixels. As a result, these pixel groups stay above the threshold of 0.1. Hence, a small α value, with a constant threshold, creates a large frequency of both large and small clusters.

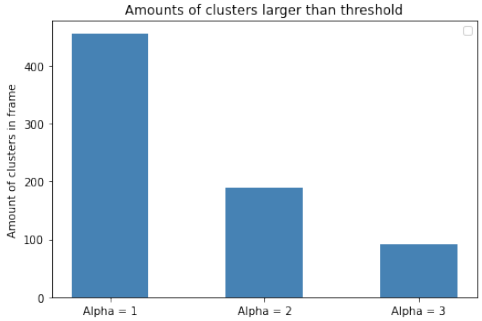
As we progress to images where α is set to 2 and 3, the smoothing effect greatens, incorporating a larger neighborhood of pixels in its calculation. This more global influence results in fewer, generally larger clusters that are more uniformly shaped. The emergence of larger clusters is due to the increased blending of pixel values across a wider area, smoothing out



(a) Varying standard deviation α, with constant threshold $\tau = 0.1$



(b) Mean cluster-size for varying values of α



(c) Amount of clusters above the threshold

Figure 13: Figures showing the relation between clusters and smoothing

local fluctuations and revealing more expansive structures.

Looking at Figures 13b and 13c, it becomes evident that there is a positive trend between the average cluster sizes and the parameter α. This implies that as the value of α increases, the average cluster sizes also increase. On the other hand, there is a negative linear trend between the number of clusters and α. This means that as α increases, the number of clusters decreases. This makes sense, as when a small standard deviation is used, the smoothing operation is essentially limited to a small neighborhood around each pixel. This leads to the less aggressive blending of pixel values, which results in smaller, more defined clusters in the final image. These clusters are typically more representative of the local variations present in the original image.

Conversely, when a large sigma value is used, the smoothing operation takes into account a wider neighborhood around each pixel. This results in a greater blending of pixel values across a larger area, which leads to the creation of larger, more diffuse clusters in the image. These clusters embody more of the larger-scale structure within the image, at the expense of losing some of the finer details.

3.3.3 Interrelation of smoothing parameter and thresholding

In the previous section, we looked at the effect of varying degrees of smoothing while maintaining the threshold at $\tau = 0.1$. This is a partial study of the effects of smoothing as it disregards the fact that by holding τ constant and increasing α, we are excluding a larger percentage of the data. The Gaussian filter has a direct effect on the standard deviation of the pixel values (the broadness of the bell curve which the pixels make in a histogram). When we raise the smoothing parameter α, values will contract towards the mean, as larger values are offset by neighboring large values of the opposite sign. So when we increase α we are cutting off more data when a constant threshold is applied.
This point is illustrated in Figure 14, which includes histograms of all three plots from 13a in the previous section. In the title of each histogram, the percentile corresponding to $\tau = 0.1$ is also displayed, clearly showing how increasing α while holding τ constant, progressively cuts off a larger portion of the data.

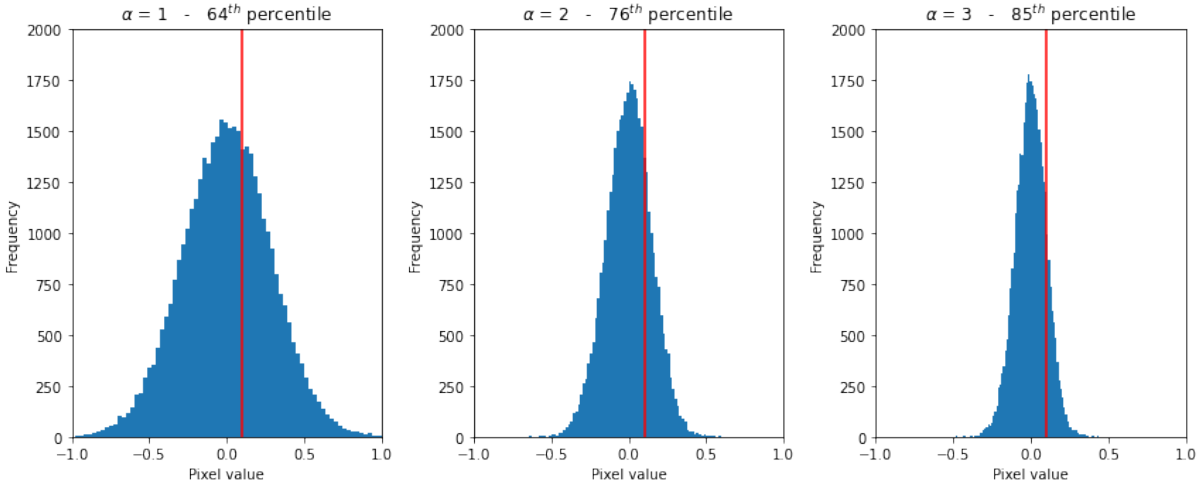


Figure 14: Pixel distributions for varying standard deviations, illustrating the cut-off effect due to a constant threshold

Figure 14 highlights the issue of disregarding the interrelation between α and τ. Consequently, a more informative approach for comparing different α values is to ensure that the same percentile of data is preserved for each α. This requires determining a linear regression relation between α and τ.

To start off we create Gaussian-filtered images with smoothing parameter values ranging from 1 to 3 in increments of 0.1, and for each α value, we compute the τ value that represents the specific percentile of interest in the Gaussian-filtered image. This is achieved through the simple function **generate_poly**, provided in the Appendix 8 . It operates by generating a series of Gaussian-filtered images based on normally distributed noise and then calculating the given percentile for each image based on the α value. These percentile values serve as our τ values for the corresponding α values, with respect to what percentile we are interested in. We end up fitting a 4th-order-polynomial of the form

$p(x) = p_0 \cdot x^4 + ... + p_4$, where our coefficients $p = p_0, p_1, ..., p_4$ are estimated.

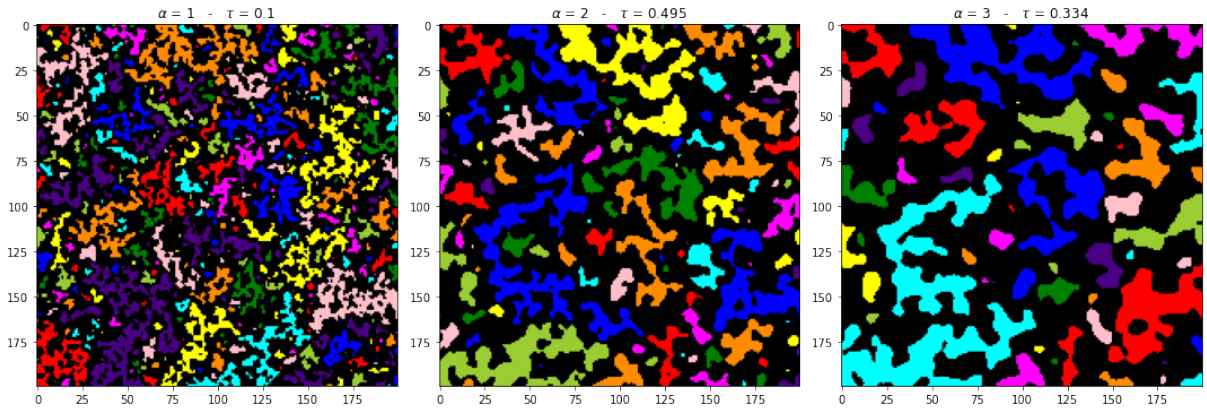


Figure 15: Increasing the smoothness parameter α, with the corresponding τ which would represent the 63^{rd} percentile.

Using the polynomial model that depicts α-τ relation at a given percentile, we can better examine the clustering properties of the smoothing parameter α. Observing Figure 15, the leftmost plot with $\alpha = 1$ and $\tau = 0.1$, we see τ approximates the 63rd percentile, hence retaining 37% of pixels post-thresholding. To consistently maintain this pixel retention rate across various α values, we apply our polynomial function, mapping α to a corresponding τ. This allows comparisons across different α values while preserving the same data ratio. We can clearly see in the other two plots (middle and left) how the average size of clusters increases, making it even more clear that α has an influence on the size of clusters.

3.4 Parameter optimization

In order to find an approximate model for the dataset at different stages of coagulation, we need to fine-tune the parameters from which the synthetic data is derived from. More specifically, this is done by optimizing the parameters responsible for smoothing and thresholding, i.e. the parameter set $\{\alpha, \tau\}$ in the smoothed Gaussian noise model. However, before being able to optimize a set of parameters, one needs to develop a method for comparing the synthetic images to the actual data. Simply defining a loss function that examines the values of each pixel in contrast to the reference image might seem tempting, but unfortunately, it's not particularly indicative of the underlying clustering behavior. For instance, applying the aforementioned loss function to two identical images, where one is slightly shifted in an arbitrary direction, would yield a significant loss. In other words, the clustering properties are not determined by the location of clusters, but rather by the relational measures they exhibit. Therefore we have opted for a loss function, which measures the similarity of two images by the relational shape measures described in the previous sections. More specifically, the loss function assigns a cost based on the fractional area measure curves for each image, by computing the squared error between the two.

Since we are dealing with a stochastic model, we can't derive the exact parameters that lead to the smallest loss each time. Instead, our approach attempts to maximize the probability of the model with unknown parameters α and τ, given some observed data. That is, we try to find the parameters for the model, such that the expected relational shape measure curves, generated by this model, differ the least from the curves obtained by the observed data. This is done by maximizing the posterior probability distribution, also known as maximum a posteriori estimation.

The objective is to minimize the given loss function. Under typical circumstances, this would involve computing the derivative of the loss function. However, since we're dealing with a stochastic process, that approach isn't feasible. Thus, we employ the gradient descent algorithm, a prevalent method used when the loss function isn't differentiable. Gradient descent adopts an iterative technique, adjusting our parameters at each step—also referred to as an epoch—based on whether the computed loss is larger or smaller than that of the previous parameter values. However, since the loss function is not necessarily convex, the local optimum $\{\alpha_i, \tau_i\}$ found by gradient descent might not necessarily be a good approximation of the global minimum $\{\alpha_g, \tau_g\}$, i.e. there might exist another local minimum $\{\alpha_j, \tau_j\}$ with a lower loss than that of $\{\alpha_i, \tau_i\}$. More specifically,

$$\mathcal{L}(\alpha_g, \tau_g) < \mathcal{L}(\alpha_j, \tau_j) < \mathcal{L}(\alpha_i, \tau_i)$$

In order to create a more robust method, that is less likely to be stuck within a suboptimal 'valley' in the loss landscape, we extend gradient descent to make use of the graduated non-convexity algorithm, in order to reduce the probability of finding a non-optimal local minimum.

3.4.1 Bayesian inference

As mentioned, the aim is to estimate the optimal values of $\{\alpha, \tau\}$ that closely resemble our observed data points. Two common approaches for parameter estimation are Maximum Likelihood Estimation (MLE) and Maximum a Posteriori (MAP) estimation. Each method offers distinct advantages depending on the available information and modeling assumptions.

Maximum Likelihood Estimation is widely used when prior information about the parameters is unavailable or when we want to focus solely on the observed data. MLE seeks to find parameter values that maximize the likelihood function, which measures the probability of observing the given data under a specific parameter setting.[7, p. 944] However, a key consideration of MLE is that it does not incorporate prior knowledge or assumptions about the parameters. Without incorporating a prior, MLE may produce estimates that are overly sensitive to outliers. Maximum Posteriori estimation, on the other hand, provides a powerful framework to incorporate prior knowledge about the parameters into the estimation process.

MAP estimation is rooted in Bayes' theorem, as we approximate a *Posterior Distribution* and maximize it. The posterior distribution is a fundamental concept in probability theory and statistics. It provides a mathematical relationship between conditional probabilities, allowing us to update our beliefs or knowledge about an event or hypothesis based on new evidence or observations. The theorem in terms of our data is stated as follows

$$P(\alpha, \tau|X) = \frac{P(X|\alpha, \tau)P(\alpha, \tau)}{P(X)}$$

Here it is given that the terms mean the following

- $P(\alpha, \tau|X)$, also known as the *Posterior*
  Description: Given samples $x_i \in X$, what is the possibility of them being computed by model M with parameters $\{\alpha, \tau\}$.

- $P(X|\alpha, \tau)$, also known as the *Likelihood*
  Description: How likely are we of observing all values in X, given a specific set of values α, τ.

- $P(\alpha, \tau)$, also known as the *Prior*
  Description: Provides a means to convey our pre-existing convictions about the values (α, τ) before seeing any data.

- $P(X)$, also known as the *Model evidence*
  Description: Quantifies the overall goodness-of-fit of a statistical model to the observed data.

We can safely disregard the denominator of the theorem, P(X) for our purposes, since we are primarily concerned with identifying the maximum value. Since $P(X)$ acts as a normalizing constant, it does not impact the location of the global maximum, meaning we can say the *Posterior* is proportional to the *Likelihood* times the *Prior*, i.e.

$$P(\alpha, \tau|X) \propto P(X|\alpha, \tau)P(\alpha, \tau)$$

$$\text{Posterior} \propto \text{Likelihood} \cdot \text{Prior}$$

Now, our objective is to maximize the numerator of Bayes' theorem, also called MAP estimation, which is given by the product of the *Likelihood* and the *Prior*. We express this as $M^*_{\alpha,\tau}$ and it is stated as

$$M^*_{\alpha,\tau} = \underset{\alpha,\tau}{\text{argmax}} \left(P(X|\alpha, \tau)P(\alpha, \tau) \right)$$

We now need to approximate the function expressions of $P(X|\alpha, \tau)$ and $P(\alpha, \tau)$.

$P(X|\alpha,\tau)$ We say the likelihood of observing all samples is equal to the product of the individual likelihoods. We assume that each observation, which is the fractional area overlap of a microscopical image at radius $= r_i$, is independent and identically distributed (i.i.d.).

With this assumption, we can assume our likelihood function of a single observation is given by a Gaussian distribution parameterized by our variables $\{\alpha, \tau\}$. I.e. we consider the residuals from the curve derived from the model $f_{\alpha,\tau}(r_i)$ to the observations y_i to follow a Gaussian distribution.

$$G_\sigma(f_{\alpha,\tau}(r_i), y_i) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(y_i - f_{\alpha,\tau}(r_i))^2}{2\sigma^2}\right)$$

This is depicted in Figure 16, where Gaussian distributions have been drawn at arbitrary r_i with $\mu = f_{\alpha,\tau}(r_i)$. As mentioned, since each observation is i.i.d., we can take the product of all likelihoods, to get a likelihood of observing all samples given parameters (α, τ):

$$\begin{aligned} P(X|\alpha,\tau) &= \prod_{i=1}^{N} \tfrac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\tfrac{-(y_i - f_{\alpha,\tau}(r_i))^2}{2\sigma^2}\right) \\ -\log P(X|\alpha,\tau) &= -\log(\tfrac{1}{\sqrt{2\pi\sigma^2}})^N - \log\exp\tfrac{-\sum(y_i - f_{\alpha,\tau}(r_i))^2}{2\sigma^2} \\ -\log(P(X|\alpha,\tau)) &\simeq a\sum(y_i - f_{\alpha,\tau}(r_i))^2 + b \end{aligned}$$

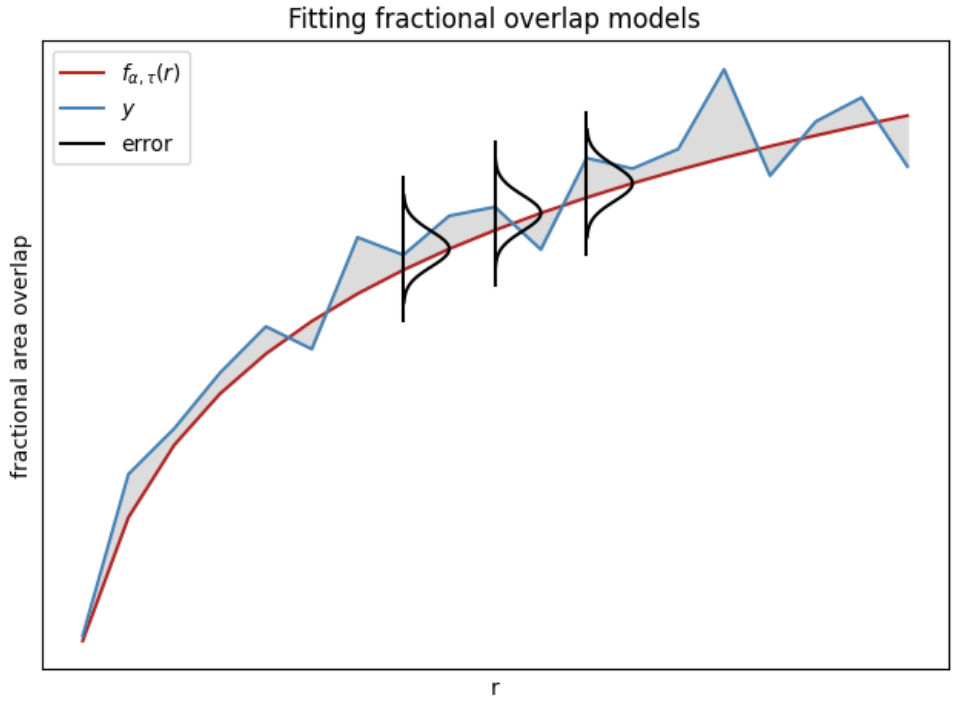


Figure 16: Calculating loss of fractional overlap curves

$P(\alpha,\tau)$ As mentioned, the prior term represents our prior knowledge or assumptions about the parameters we are trying to estimate. Including the prior allows us to balance the

influence of the observed data and our prior beliefs, leading to more robust estimates. The prior is especially useful if the amount of data is limited, or if obtaining the data is computationally expensive. Another advantage of including the prior term is that it enables us to iteratively update our beliefs as new data becomes available, or in our case, as the new frames get introduced. Now comes the challenging part - how do we choose the appropriate prior distribution? In other words, what do we know about $\{\alpha, \tau\}$? If we have absolutely no prior knowledge about these parameters, then a uniform distribution should be used, since we do not want to affect the likelihood term.

We have chosen to approximate the prior distribution as a bivariate Gaussian distribution, centered around $\mu = [\tilde{\alpha}, \tilde{\tau}]$. I.e. a joint distribution of $G_{\sigma_1}(\tilde{\alpha}, \alpha)$ and $G_{\sigma_2}(\tilde{\tau}, \tau)$.

$$\begin{aligned} P(M) = &\quad G_{\sigma_1}(\tilde{\alpha}, \alpha) \cdot G_{\sigma_2}(\tilde{\tau}, \tau) \\ = &\quad \tfrac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\tfrac{-(\alpha-\tilde{\alpha})^2}{2\sigma^2}\right) \cdot \tfrac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\tfrac{-(\tau-\tilde{\tau})^2}{2\sigma^2}\right) \\ -\log P(M) \simeq &\quad a(\alpha-\tilde{\alpha})^2 + b(\tau-\tilde{\tau})^2 + c \\ P(\alpha, \tau) \simeq &\quad a(\alpha-\tilde{\alpha})^2 + b(\tau-\tilde{\tau})^2 + c \end{aligned}$$

Where $(\tilde{\alpha}, \tilde{\tau})$ denote the prior parameters. The prior parameters represent our best guess about the optimal values of α and τ prior to the analysis. We estimated that a good strategy for deriving at a reasonable guess for the prior parameters was to compare the number of clusters and the average cluster sizes between a series of synthetic data and the actual data. This procedure is explained in more detail later.

Now that we have approximated our Likelihood and Prior distributions, we can derive at the expression for the parameters we want to optimize

$$\begin{aligned} M^*_{\alpha,\tau} = &\quad \underset{\alpha,\tau}{\text{argmax}} \left(P(X|\alpha, \tau) P(\alpha, \tau) \right) \\ = &\quad \underset{\alpha,\tau}{\text{argmin}} \left(-\log P(X|M) - \log P(M) \right) \\ \simeq &\quad \underset{\alpha,\tau}{\text{argmin}} \left(\textstyle\sum_i (y_i - f_{\alpha,\tau}(r_i))^2 - \log P(M) \right) \\ \simeq &\quad \underset{\alpha,\tau}{\text{argmin}} \left(\textstyle\sum_i (y_i - f_{\alpha,\tau}(r_i))^2 + a(\alpha-\tilde{\alpha})^2 + b(\tau-\tilde{\tau})^2 + c \right) \end{aligned}$$

Lastly, all we have to do is choose adequate values for the constants a, b and c. We can safely ignore c, since adding a constant term doesn't affect the location of the optimum. Unfortunately, there is no straightforward approach for deriving a and b. As we will later show, the specific values of a and b are not important, since we will be including a scaling factor in front of the prior term when applying gradient non-convexity. Thus what we need to determine is their appropriate values in relation to each other. Since the range of valid values of α is approximately double the range of τ values, we deemed that setting $a = 0.5$ and $b = 1$ was a reasonable choice. Thus our final expression is given by:

$$\text{MAP}_{\alpha,\tau} \simeq \underset{\alpha,\tau}{\text{argmin}} \left(\sum_i (y_i - f_{\alpha,\tau}(r_i))^2 + 0.5(\alpha - \tilde{\alpha})^2 + (\tau - \tilde{\tau})^2 \right)$$

3.4.2 Gradient descent

As aforementioned, in the context of deriving a model for generating synthetic data that best describes the microscopy images of milk protein coagulation, we use gradient descent. That is, we try to find the optimal values for the parameters $\{\alpha, \tau\}$ by iteratively minimizing the loss between the measured data y and the synthetic data $f_{\alpha,\tau}(r)$ generated using different parameter values. However, since $f_{\alpha,\tau}(r)$ is derived from Gaussian noise, it is characterized by a lot of randomness. Because of this, calculating the loss using $f_{\alpha,\tau}(r)$ might lead to inaccurate results. To mitigate this randomness, we have opted for using the expected value of $f_{\alpha,\tau}(r)$ instead, denoted by $\mathbb{E}(f_{\alpha,\tau}(r))$, which provides a more representative loss estimate for the parameters. We estimate the expected value by taking the average of multiple $f_{\alpha,\tau}(r)$ curves since the expected value of a Gaussian distribution is simply its mean μ. In reality, we are not able to obtain the true expected curve, since it would require calculating the mean of an infinite number of samples

$$\mu = \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} x_i$$

Finding the ideal number of curves to average out proves to be quite challenging, due to the trade-off between accuracy and efficiency. We would like to take the mean of as many curves as possible in order to obtain the best possible estimate, and thus produce the most reliable results. However computing each curve is a very complex process, which requires a lot of computational effort. There is no correct answer to this problem, and it is further explored in Section 5.3.1 where a suitable compromise is found.
In the previous section, we derived the expression that we want to minimize in order to determine the optimal parameters. This simply corresponds to the loss function:

$$\mathcal{L} = \sum_i (y_i - \mathbb{E}(f_{\alpha,\tau}(r_i)))^2 + 0.5(\alpha - \tilde{\alpha})^2 + (\tau - \tilde{\tau})^2$$

As seen, the loss function represents the squared error between the data points from each curve, with a regulating term added to it, which denotes our prior beliefs. We want to find parameters (α, τ) such that

$$(\alpha, \tau) = \underset{\alpha,\tau}{\text{argmin}}\, \mathcal{L}$$

The basic idea behind gradient descent is to start with an initial set of parameter values (α^0, τ^0), and then update them iteratively by taking steps proportional to the negative gradient of the loss function, where the length of each step is determined by a learning rate value γ. I.e. for each iteration of the gradient descent algorithm, the next parameter set is given by

$$(\alpha^{s+1}, \tau^{s+1}) = (\alpha^s, \tau^s) - \gamma \left(\frac{\partial \mathcal{L}}{\partial \alpha}\bigg|_s \frac{\partial \mathcal{L}}{\partial \tau}\bigg|_s \right), \quad s \geq 0$$

Where $\frac{\partial \mathcal{L}}{\partial \alpha}$ and $\frac{\partial \mathcal{L}}{\partial \tau}$ are the partial derivatives of $\mathcal{L}$ with respect to α and τ, which together form the gradient. Calculating an accurate gradient is important since it represents the direction toward the optimum. How to derive an appropriate gradient is further explained in the following sections. Finding an appropriate learning rate is another crucial aspect of using gradient descent effectively. A smaller learning rate leads to slower convergence but ensures stability, while a larger learning rate can speed up convergence but may cause overshooting. Through various testing, we determined that $\gamma = 0.01$ is a suitable learning rate for our analysis.

The gradient descent algorithm terminates if a certain stopping criterion is met. The specific stopping criterion varies depending on the context and the problem being solved. Usually, this criterion is either if the maximum number of iterations has been reached, or if a given convergence condition is satisfied. The convergence condition can be related to the loss function itself or the parameters which we want to optimize. I.e. the algorithm might terminate if the loss reaches a sufficiently low and acceptable level, or if the change in the parameters between iterations falls below a certain threshold. Since we do not use the true expected value of $f_{\alpha,\tau}(r_i))$, but instead an approximation, there is some uncertainty associated with the loss function. As a result, $\mathcal{L}$ might not yield the same loss if computed multiple times using the same parameter values. Therefore, incorporating a stopping criterion based on a convergence condition might be a risky approach in our case, because of the likelihood of the algorithm prematurely terminating due to this randomness. For that reason, we have chosen to ignore the convergence criterion and only terminate the procedure when the maximum number of iterations has been reached.

Algorithm 3 shows the pseudo-code for our implementation of gradient descent in a simplified form. We start by specifying the learning rate and the number of iterations. Then, in each iteration step, we calculate the gradient and update the parameters (α, τ) as specified above.

Algorithm 3 Gradient descent algorithm

1: max_iter = number of iterations
2: γ = learning rate
3: **for** $i = 0$ to max_iter **do**
4: $\quad \alpha\_grad$ = calculate partial derivative of $\mathcal{L}$ with respect to α
5: $\quad \tau\_grad$ = calculate partial derivative of $\mathcal{L}$ with respect to τ
6: $\quad \alpha \mathrel{-}= \gamma \cdot \alpha\_grad$
7: $\quad \tau \mathrel{-}= \gamma \cdot \tau\_grad$

Prior estimation During the implementation of gradient descent, the initial guess for the optimal parameter set can play a significant role in determining the convergence and eventual outcome of the optimization process. Because the loss landscape (represented by the relationship between the parameters and the squared error), is unknown, we have to assume that selecting an appropriate initial guess is crucial for finding a good local minimum.

There is no apparent best strategy for choosing the initial parameters, so in our case, we did so by the following methods. By generating synthetic data using different parameter values, we can analyze the resulting images and compare the average protein cluster sizes. The parameters that produce a synthetic image with an average cluster size closest to that of the measured image, are considered plausible starting values. This approach uses the visual assessment of protein sizes as the criterion for selecting suitable initial parameter values. As such, by starting with initial parameter values that align closely with the observed protein sizes, we implicitly assume that there is some correlation between the average protein size and the result of the analysis.

The above-explained method for estimating an appropriate prior is used only for performing gradient descent on the first image in the series. For every subsequent image from the video, we chose to instead simply use the optimized parameter set from the preceding image, as the prior for the next. This method of calculating the prior for the gradient descent algorithm makes the assumption that a good local minimum, found for any image, is also a relatively good local minimum for the subsequent couple of images. In turn, this assumption relies on there being little change in the optimal set of parameters, that best synthesize the relational shape measure of images close to each other.

Gradient approximation Calculating the gradient of the cost function is an essential part of gradient descent since it represents the direction of the greatest change at a given point in each iteration. This means that by moving in the direction, or the opposite direction, of the gradient, we should eventually reach an optimum (if one exists). However, deriving the exact gradient of the loss function with respect to α and τ is not possible. At least not within the scope of this project, due to the complexity of our model and the randomness in producing the Gaussian noise images. Therefore, we instead opted for estimating the gradient, i.e. approximating the partial derivatives of the cost function with respect to each parameter.

There exist various techniques for gradient approximation, with possibly the most prevalent being the finite differences method. The finite difference is a numerical approach that involves calculating the difference in function values at nearby points and dividing by the distance between those points. Generally, it provides a reasonable estimate of the gradient, but its accuracy depends on the choice of step size and how the function behaves. We can derive these numerical approximations by exploiting the Taylor series. Taylor series

are used to approximate a function by a polynomial expression, and it is given by the following summation, centered at $x = a$

$$\begin{aligned} f(x) &= \sum_{i=0}^{\infty} \frac{f^{(i)}(a)}{i!}(x-a)^i \\ &= f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \ldots + \frac{f^{(n)}(a)}{n!}(x-a)^n + \ldots \end{aligned}$$

Central differences are a type of finite difference, which estimates the derivative using two points: one ahead of the current point and one behind it. We can derive the central differences function using the Taylor series of $f(x \pm h)$ centered at $x = a$:

$$f(x+h) \simeq f(x) + hf'(x) + \frac{h^2}{2}f''(x) + \mathcal{O}(h^3)$$
$$f(x-h) \simeq f(x) - hf'(x) + \frac{h^2}{2}f''(x) - \mathcal{O}(h^3)$$

Subtracting the above expressions we get:

$$f(x+h) - f(x-h) = 2hf'(x) + \mathcal{O}(h^3)$$
$$2hf'(x) = f(x+h) - f(x-h) - \mathcal{O}(h^3)$$
$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} - \mathcal{O}(h^2)$$

The same reasoning can be applied to obtain the function for forward differences, which approximates the derivative using the current point and a nearby point ahead of it:

$$f(x+h) - f(x) = hf'(x) + \mathcal{O}(h^2)$$
$$hf'(x) = f(x+h) - f(x) - \mathcal{O}(h^2)$$
$$f'(x) = \frac{f(x+h) - f(x)}{h} - \mathcal{O}(h)$$

The terms $\mathcal{O}(h)$ and $\mathcal{O}(h^2)$ represent the error terms. The error term for forward differences is proportional to h, indicating that the error decreases linearly with the step size, while the error term in central differences is proportional to h^2, meaning that the error decreases quadratically with the step size. From this, we can conclude, that central differences are generally more accurate than forward differences. On the other hand, the central differences method requires evaluating the function on both sides of x, which often results in additional computations compared to forward difference.

This is especially an issue when approximating partial derivatives with respect to multiple variables. Central differences require two function evaluations for each partial derivative, in contrast, to forward differences, which only need an extra function evaluation for each

new partial derivative. This trade-off between accuracy and computational efficiency means that choosing between these approaches depends on the specific analysis and its goals. Due to the complexity related to computing our model, we have chosen to prioritize the computational efficiency aspect and thus opted for forward differences. This allows us to evaluate the model with different τ and α parameters only thrice instead of five times at each iteration of gradient descent, which significantly reduces the run-time of the algorithm.

3.4.3 Graduated non-convexity

Gradient descent is a widely used optimization algorithm for minimizing convex loss functions. However, when running gradient descent with the loss function $\mathcal{L}$, we are left with the problem that $\mathcal{L}$ might be non-convex. In this case, since the gradient descent algorithm only finds a local minimum, there is no guarantee that it will find a global minimum. And depending on the initial point in which GD starts, the local minimum found, might not even be a good approximation of the global minimum. To address these challenges, we chose to integrate the Graduated Non-Convexity Algorithm (GNCA), which uses the concept of approximating a non-convex loss function with a sequence of convex functions. This sequence of approximations is meant to aid our optimization technique by computing starting weights for $\{\alpha, \tau\}$ at which our $\mathcal{L}$ can reach a global minimum [2, p.125].

When applying GNC a few modifications are required, as our loss function $\mathcal{L}$ is non-differentiable, which is due to the nature of our data being a stochastic process. Instead of approximating the loss to a convex function, we incorporate a convex elliptic paraboloid, based on the prior distribution into the optimization process. When we compute MAP, we initially multiply this paraboloid by a large constant factor λ, which gives the prior significant influence on the posterior distribution. This also smoothes out the loss landscape, approximating a paraboloid shape centered on the prior. However as the iterations pass and λ is gradually reduced, the loss landscape will be more and more dominated by the squared error term, until $\lambda = 0$. This new version of the loss function can be described as stated below

$$\mathcal{L} = \sum_i (y_i - \mathbb{E}(f_{\alpha,\tau}(r_i)))^2 + \lambda\Big(0.5(\alpha - \tilde{\alpha})^2 + (\tau - \tilde{\tau})^2\Big), \quad \lambda \geq 0$$

In practice, this modification of the graduated non-convexity algorithm initially results in a convex paraboloid, centered over the prior. As λ decreases, the loss landscape more and more assumes the actual values of $\mathcal{L}$. An example of how such an implementation of GNC looks in practice can be seen in Figure 17, where the different algorithm blue lines represent the algorithm being run with different values of λ, until it finally becomes 0.

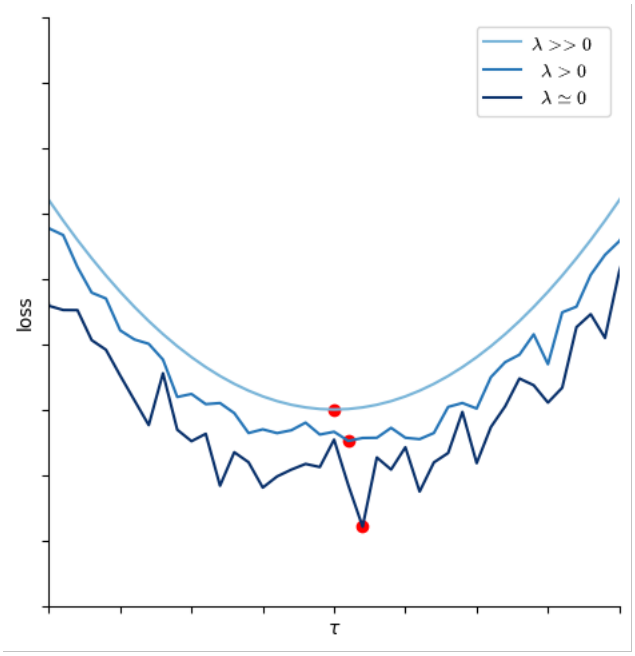


Figure 17: Implementation of GNC on sample loss space

4 Results

4.1 Interpretation of our Method

4.1.1 Expected results

We have applied the algorithm described in section 3.2.4 regarding Relational Shape Measures to a set of 15 microscopical images in sequential order from the video. The purpose was to analyze the casein aggregation and observe the changes in the area overlap, fractional area overlap, and curve overlap. We also want to evaluate which measures can provide the best foundation for drawing conclusions on our images.

By observing the images presented in Figure 18 (top row), the visual presence of aggregated casein proteins forming light-colored clusters becomes evident. However, it is important to note that our analysis considers only those clusters that surpass a specified light-intensity threshold, ensuring their measurability. The corresponding processed images are displayed in the bottom row of Figure 18, where clusters have been identified and labeled. A threshold of light intensity is required, as we are not interested in the overall amount of proteins in an image, but rather how much these casein protein structures cluster together.

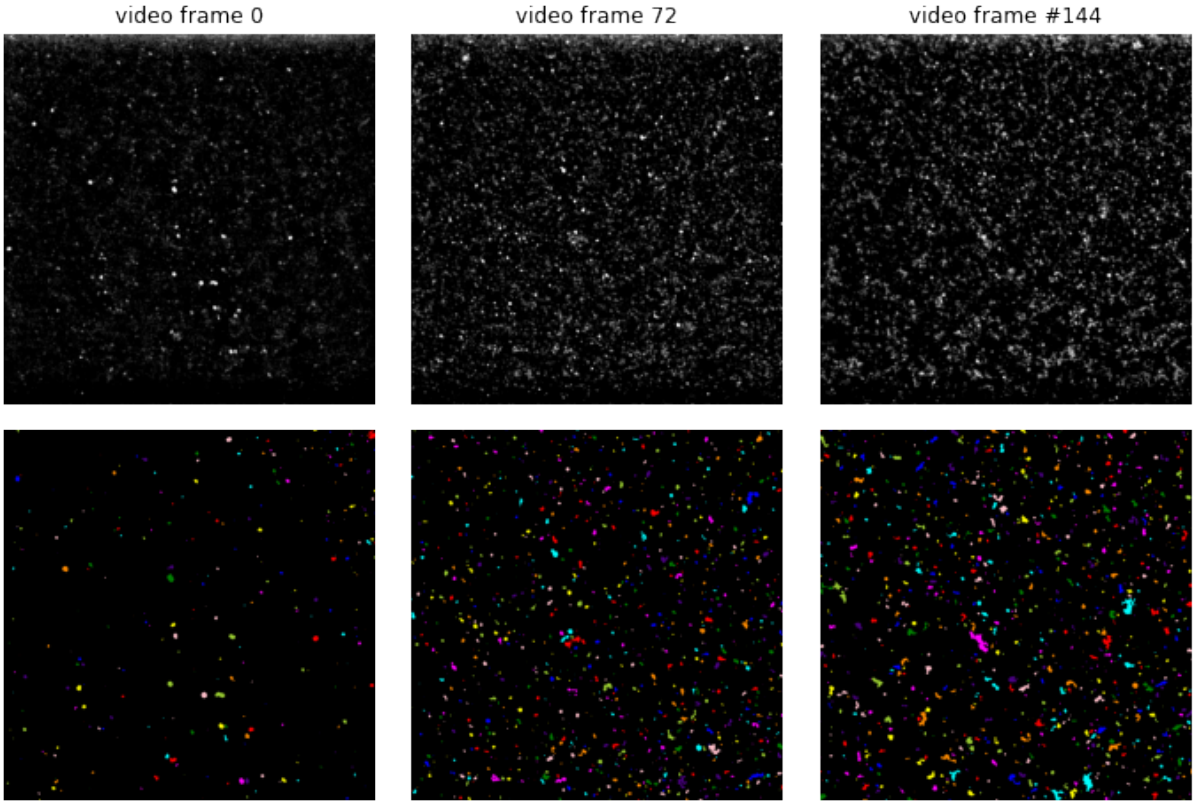


Figure 18: Top: Unprocessed video frames
Bottom: Processed with clusters and a threshold cutoff

Based on the observations made from the bottom row of images, it is anticipated that the three measures—area overlap, fractional area overlap, and curve overlap will exhibit

a noticeable increase with the introduction of each new frame. It becomes evident that more structures within the microscopical images surpass the specified threshold with each frame. This implies that a larger proportion of the observed objects will intersect with the equidistant curves, leading to an expected rise in both the area overlap and fractional area overlap measures.

4.1.2 Interpretation of graphs

Applying our developed relational shape measures algorithm to the set of sequential microscopic images led to a deeper understanding of casein aggregation. As depicted in the area overlap, fractional area overlap, and curve overlap graphs in Figure 19, we can interpret the development of casein aggregation across the 15 frames.

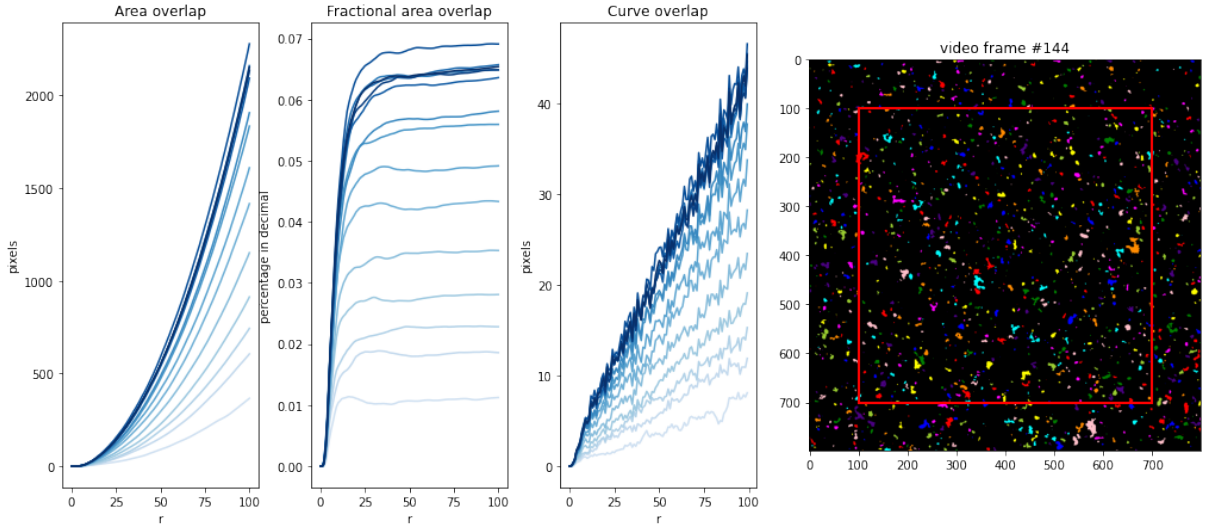


Figure 19: Plotted results of 15 frames

Area overlap The area overlap graph shows a steady increase frame by frame. The difference in area overlap between each of the frames becomes less pronounced in the last four frames, which suggests that the aggregation process approaches a plateau. This observed increase aligns with our hypothesis based on the visible increase in colored clusters in the processed images.

Frac area overlap The fractional area overlap graph similarly shows an increase for the first 10 frames, increasing around 0.5 percentage points per frame when measured at a radius > 20. This consistent increase suggests that as the casein proteins aggregate, they take up a larger proportional area within the defined threshold. The convergence of the last four frames indicates that the rate of change in fractional area overlap decreases as the aggregation progresses.

Curve overlap Finally, the curve overlap graph demonstrates a steady increase that slowly converges towards the end, mirroring the trend seen in the area overlap graph. This convergence

validates our method as the curve overlap is expected to reflect the rate of change of the area overlap

These findings collectively suggest that the three measures increase and eventually plateau, likely indicating the progression and stabilization of casein aggregation over time. Moreover, our analysis confirms that our relational shape measures algorithm successfully builds upon the principles of Ripley's K-function, incorporating shape measures to provide quantifiable information that was previously qualitative or visually assessed.

4.2 Optimizing parameters with gradient descent

4.2.1 Accuracy of model

Performing shape relation measurement on the frames of the coagulation video, we found that the coagulation seems to stop about 70% through. This is evident by the fact that the curve representing fractional area overlap reaches a peak of 0.06 around this point, and can be seen on the second graph in Figure 19. Because we found that the coagulation of the proteins seems to stop around 3/4th of the video, we split the video into 10 frames and ran gradient descent on the first 7. Running the algorithm we decided on using five different values of λ for the graduated non-convexity and performing gradient descent for each of these. This resulted in us essentially running gradient descent for 100 iterations, with the loss landscape changing every 20 frames. This process is repeated for each picture and the resulting loss values at each iteration can be seen in Figure 20.

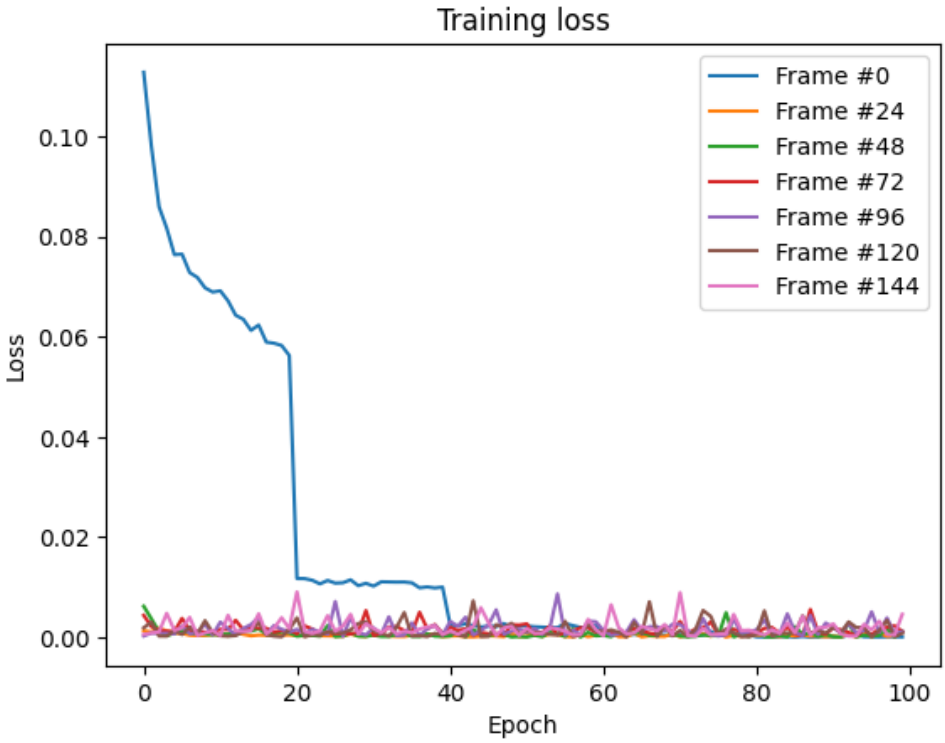


Figure 20: Evolution of loss as a function of training epoch

As can be seen on the figure the loss is initially relatively high in comparison to the rest of the figure. It quickly decreases, until it reaches 20 and 40 iterations where the total loss sharply drops as a result of the updated λ value. The loss stabilizes around 0.01 for the rest of the training apart from a few outliers which likely come as a result of the synthetic data generation being a stochastic process. The low constant loss at the start of each succeeding frame is likely the effect of the prior being a good approximation of the posterior. Since the prior for each frame apart from the first is simply the posterior of the previous iteration, it would suggest that the optimal parameters are only slightly changing throughout the video. In turn, this could indicate a relation between the two we have tried to model in Figure 21.

4.2.2 Parameter relationship

As a result of the gradient descent, we get a set of optimized parameters $\{\alpha, \tau\}$ for each of the seven frames from the video that was analyzed. This relationship between the two parameters over time can then be visualized using a graph, by plotting each parameter set as a point in a 2D system. This system can be seen in Figure 21, where each point corresponds to the optimized parameter set found for the given time t.

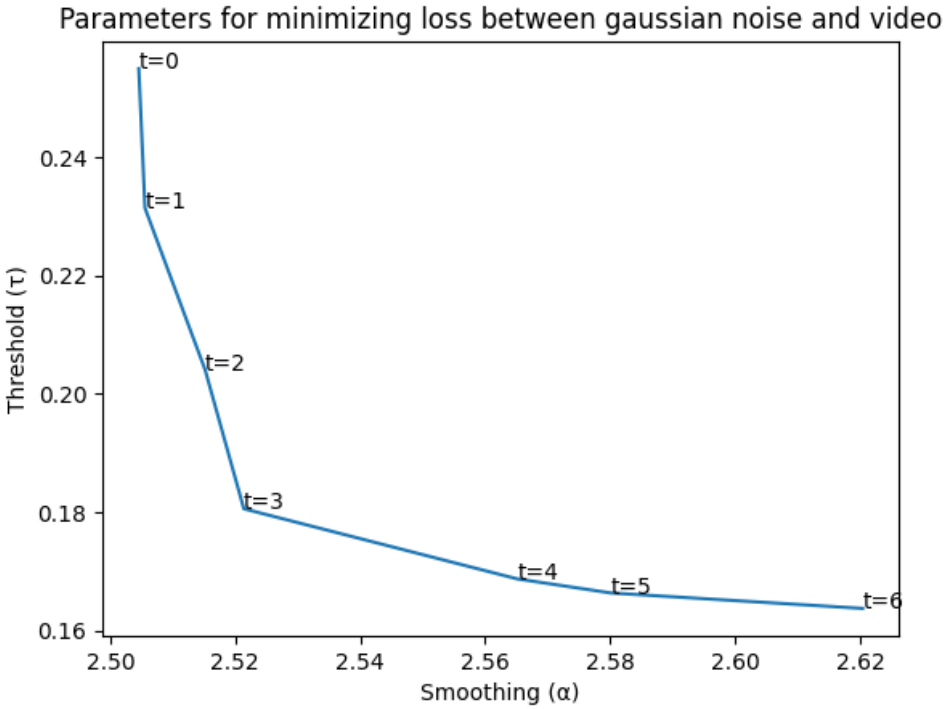


Figure 21: Relationship between α and τ over time

Looking at the graph we see that the optimized threshold value needed to most accurately model the data tends to decrease over time. This decrease in threshold over time implies, that more protein clusters become visible. Since the actual number of proteins in the gel stay constant throughout the entire experiment, this instead implies that as time passes,

more clusters become bigger than the threshold, and therefore become visible.

These implications can be drawn from the generalization discussed in Section 3.3. This found that when synthesizing Gaussian-smoothed data, the threshold seem inversely proportional to the amount of clusters visible, while the smoothing correlated to the general size and uniformity of clusters. In the case of Figure 21 the optimized smoothing found in the parameter optimization increases over time aggregation. This increased smoothing suggests that the general trend as time coagulation occurs is for protein clusters to become bigger, and more uniform.

This relationship between the parameters approximates a reciprocal function, where an inverse relationship exists between the two parameters α and τ, such that when one increases, the other decreases. Such a relationship is not necessarily surprising since the evolution of the protein clusters is a slow gradual change, and when there's a slight increase in one parameter (in this case α), there is a corresponding slight decrease in the other parameter. This is further examined in Section 3.3.3.

4.2.3 Synthesizing data from model

Having obtained the optimized parameters through the application of gradient descent with graduated non-convexity, we proceeded to synthesize data using these parameter values. The synthesis process involved applying these parameter values to the Gaussian model of protein coagulation, resulting in a simulated dataset on which the Ripley's-K implementation is then run.

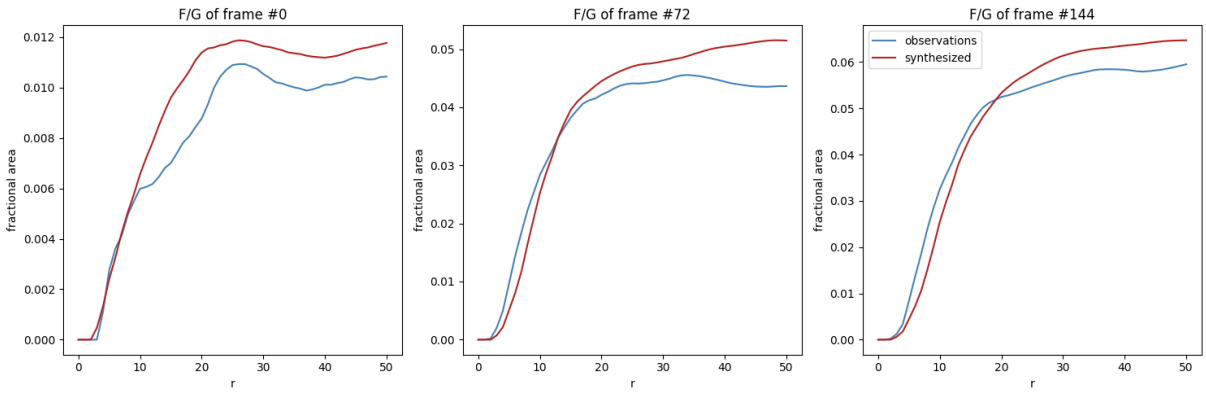


Figure 22: Comparing fractional area overlap of our observations (blue) and synthetic data at three different time frames

In addition to quantitative analysis, we can perform a visual inspection of the expected synthetic data $\mathbb{E}(f_{\alpha,\tau}(r))$ in comparison to the measured data y. The visual examination allows for a more qualitative assessment of the overall shape, patterns, and characteristics of the synthesized dataset and its resemblance to the observed behavior in the microscopy images. Looking at Figure 22 we see such a direct comparison, where the three images

represent the start, middle, and end coagulation. Looking at the data side by side it seems that the model is good at getting the overall shape of the curve correct, however, there are some noticeable differences between the two.

Firstly, the analysis of the synthesized data results in more smoothed-out curves, that do not contain the same imperfections and irregularities that the real data possesses. Another noticeable difference is that as the coagulation process advances we see that the curves become less irregular. Since the synthesized model is more smoothed out than the real data, it looks like it has an easier time fitting the later, more regular, stages of coagulation. The reasoning for the expected synthetic data $\mathbb{E}(f_{\alpha,\tau}(r))$ being more smooth, is that we calculate the expected data as the mean of multiple synthetic datasets $f_{\alpha,\tau}(r)$, as such. This is simply a trade-off between the model having the accuracy of the $\mathbb{E}(f_{\alpha,\tau}(r))$ and the "natural" fluctuations of a randomly generated $f_{\alpha,\tau}(r)$. While there are differences between the synthesized and real data, it is important to recognize that the discrepancies between them are of a small order magnitude, and as such, the model might provide adequate results depending on the margin of error required.

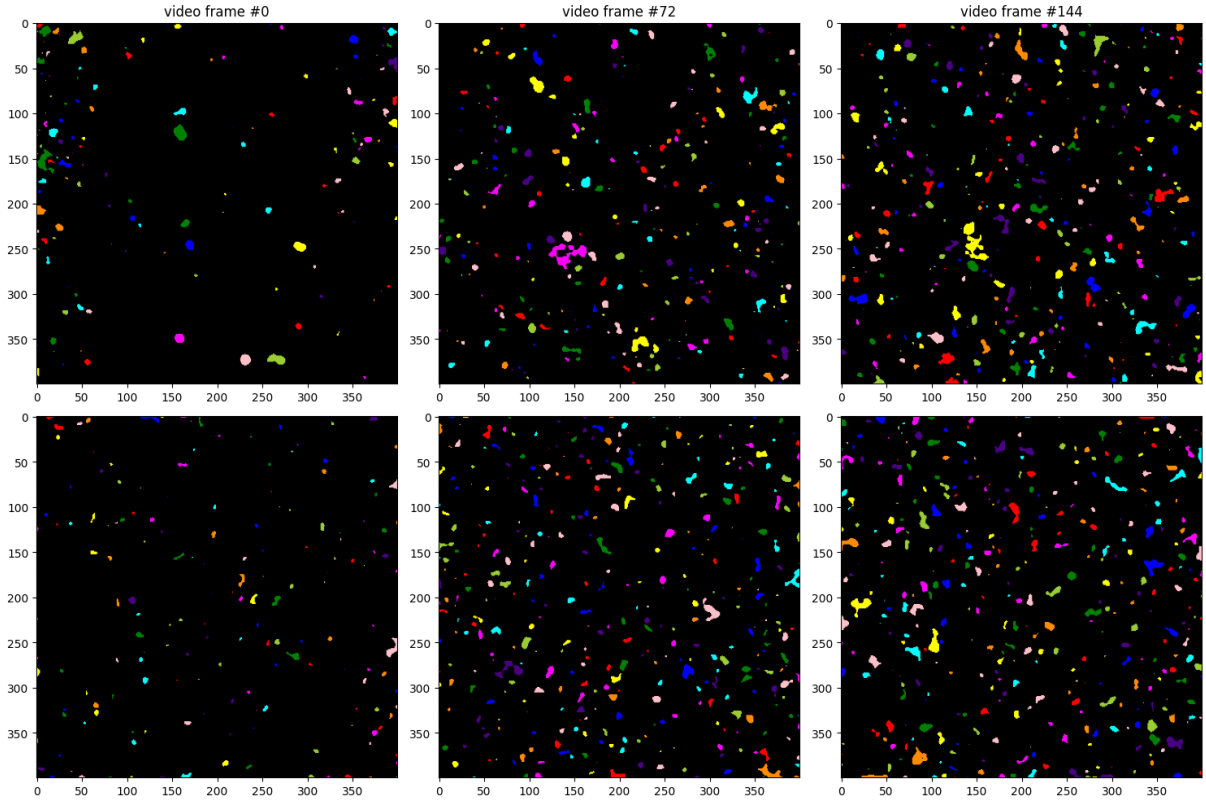


Figure 23: Visual comparison of measured and synthesized data

In addition to analyzing the fractional area overlap curves, we can also perform a visual comparison between the measured data and the data synthesized from the optimized parameters. In Figure 23 such a comparison, where the top three images are from the same three frames from the video, and the corresponding synthesized approximation is at the

bottom. Looking at the evolution of the proteins throughout the coagulation process we see that the number of proteins above the threshold greatly increases with time for both the measured, and synthesized data. Looking at the sizes of the individual protein clusters we see that there is a big variance between the sizes of the bigger and smaller clusters in the real data, in comparison the synthesized data is more uniform in this regard.

Quantifying the mean protein sizes, and total amounts of proteins for each frame as seen in Figure 24 seems to support our observations as well. In doing so it becomes clear that the trend for both real and synthetic data is to increase the mean protein size and amount proportional to time. And while these trends are initially very apparent, the overall growths seem to decelerate towards the end of coagulation. Looking at the graphs, it also becomes apparent that while the mean size of the proteins is larger for the real data, there are also fewer of them in comparison to the synthetic ones.

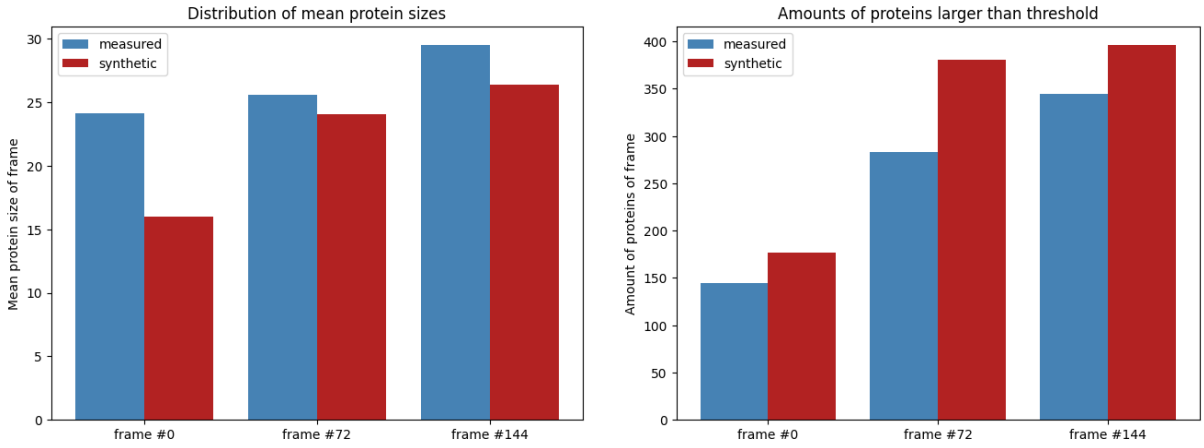


Figure 24: Comparison between samples of measured and synthetic data

5 Discussion

5.1 Modelling protein coagulation as smoothed Gaussian noise

Previously, we vaguely presented some of the rationale for creating synthetic data by smoothing Gaussian noise with a Gaussian kernel. This included different factors, such as the simplicity of the model, which facilitates the extraction of valuable information, and how smoothing introduces spatial correlation, which can be used to imitate clustering behaviour in general.

Although these are valid reasons, they are not very specific in explaining how the smoothed Gaussian noise might be related to the coagulation of milk proteins seen in the microscopy images. It turns out that the process of creating our smoothed Gaussian noise and the process of coagulating milk proteins both share the same underlying characteristics. In fact, both processes can be described by the notion of Brownian motion. When the milk proteins in the liquid medium are exposed to the aforementioned conditions, which include pH and temperature changes, they undergo diffusion. Diffusion refers to the process by which these proteins disperse throughout the milk. This diffusion is mainly controlled by Brownian motion, where proteins experience random movement due to collisions with surrounding molecules. According to the Central Limit Theorem, the sum of all these collisions with surrounding molecules will approximate a Gaussian distribution. I.e. the distribution of the displacement of proteins over a given time interval can be modelled by a Gaussian distribution [4, p. 28].

A similar mechanism occurs in the generation of synthetic data, where the smoothing of Gaussian noise by Gaussian filtering closely resembles the diffusion process. During the convolution of the noise image, the value of each pixel is determined by multiplying the surrounding pixels by the corresponding weight in the Gaussian kernel. This means that high-value pixels will diminish, while also influencing neighboring pixels, dragging their value closer to the value of the reference pixel. This process bears a clear resemblance to the diffusion of particles, which we argued follows a Gaussian distribution. In other words, the Gaussian kernel used in the smoothing process corresponds to the diffusion kernel, which indicates that Smoothed Gaussian noise is a suitable model for capturing the clustering patterns in the coagulating milk proteins.

5.2 Critique of methodology

5.2.1 Thresholding images arbitrarily

When segmenting the video into images to be analysed, we thresholded them based off of values we found suitable, by analysing the manually segmented images brought by FOOD. While those thresholds fit the sample images, having a global threshold on which to segment all images will undoubtedly produce varying uniformity of clusters across the

entire video. When thresholding the data we found the threshold of $\tau = 0.3$ for segmenting the video to be the most accurate, and as such, all of the shape relation measures and synthetic data, was created with this segmentation of the video in mind.

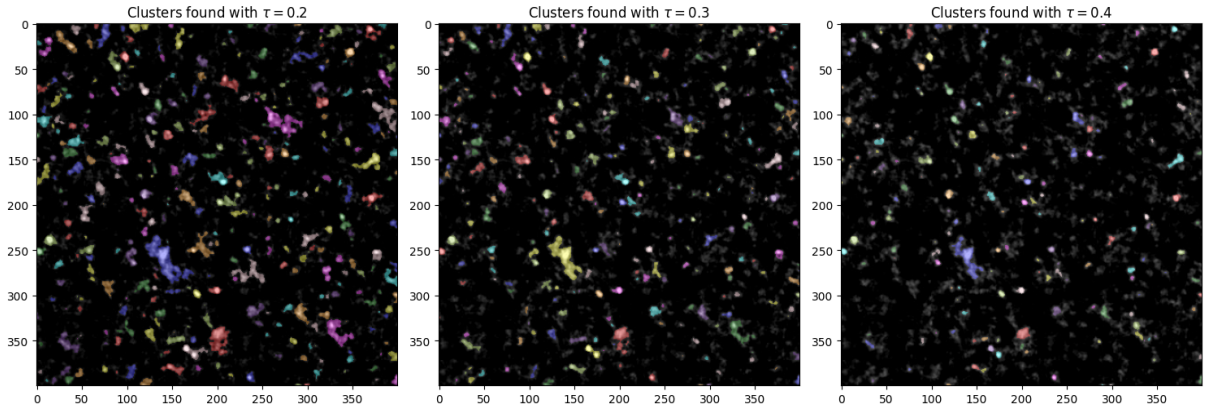


Figure 25: Comparison of clusters found for same image with varying thresholds

When looking at Figure 25, it becomes clear, just how much difference a small change in the threshold can make, especially when not compensated for in terms of the Gaussian kernel's standard deviation, can have a big effect on the resulting shape relation measures. On the figure the last frame from the video can be seen segmented by three different values of threshold, with the middle figure being the value used for the analysis. Behind the images lay a background, which is a greyscale version of the same image to give context to the three images.

As can very clearly be seen on the images, even with a variance of only $\tau \pm 0.1$ the resulting change not only alters the clusters drastically in terms of size, but also in terms of amount and shape as well. To put this into the context of the project, the resulting shape relation measures can be found on Figure 26. Here these qualities can be examined further, and it becomes very clear that there are some fundamental differences between the three images. First and most notably, the sheer difference in quantity of pixels that were classified is very big from the first to last image. This can be seen on the fact that the area overlap of the curve with a high threshold has a much lower maximum value then the two other functions.

Secondly when examining the difference in the fractional area overlap between the three images, besides seeing the numerical difference in overlap, we also see some qualitative differences in the evolution of the curves. We see that as the threshold is increased and the total number of identified clusters decreases, the curves become less uniform and are thus more prone to jaggedness. As such, there seems to be plateauing in the radius where the mean fractional area overlap doesn't increase, meaning that there are ranges of r where the coagulation of proteins seems to stabilize. This seems to be affirmed by the less curve overlap functions being less jagged in this range as well, alluding to the same issue being true.

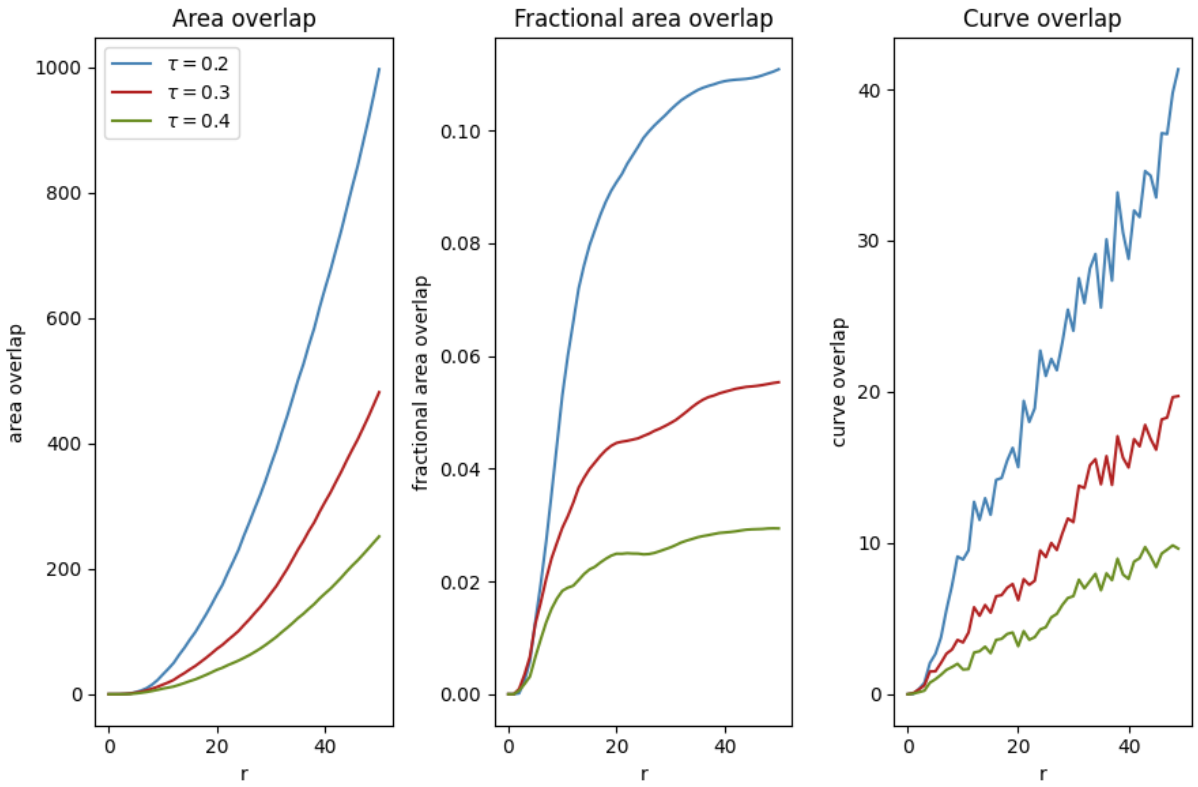


Figure 26: Comparison of shape relation measures of varying thresholds

The consequence of this problem is that while the threshold of $\tau = 0.3$ seems to be best fitting for the images we tested against, it might not be the best fit for all images. And a seemingly small numerical difference can have a huge impact on the resulting image, and therefore the shape relation measures of the said image as well. Our analysis was performed with a constant global threshold throughout the video, and to combat this problem, one could for example instead vary the threshold to accommodate each image individually.

5.2.2 Interpreting spatial and geometric properties with our methods

As we evaluate the utility of our relational shape measure method, we first aim to understand whether it can provide insights into the spatial distributions of objects within an image. The experiment involves images with labeled clusters created using different smoothing parameters, as seen in Figure 15. We want to compare images with the same number of pixels being part of a cluster, but with drastically different average cluster sizes. We want to assess the method's ability to interpret the geometric properties of the clusters in the image.

Looking at Figure 27, we can draw a couple of important conclusions. First, we intentionally used a larger radius than usual. We did this to demonstrate that, when the number of colored pixels in both images is equal, the fractional area overlap levels off at a certain radius.

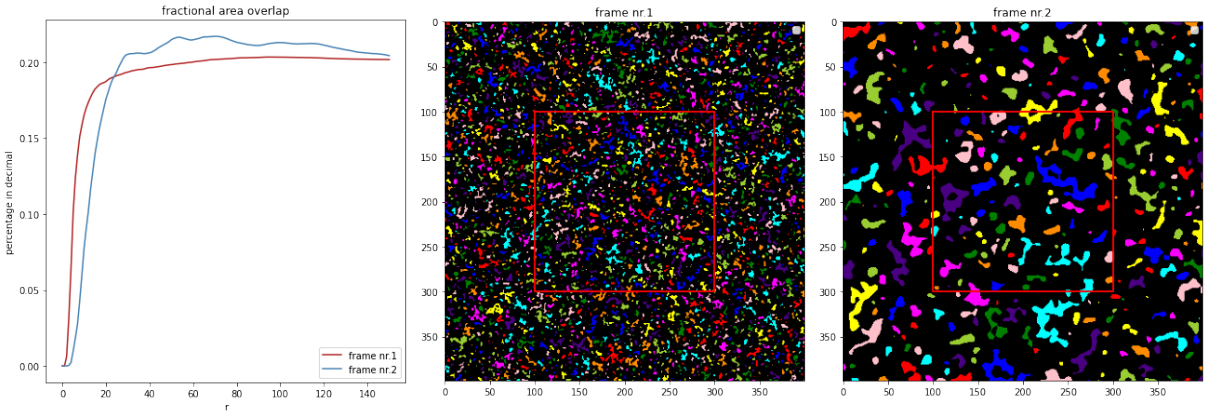


Figure 27: Shape relation measures of synthetic images with different average cluster sizes

In the first frame, where the clusters are smaller, there's a noticeable increase in fractional area overlaps at the beginning, but this levels out when we get to around $r = 40$. This is likely because the smaller clusters are spread more evenly throughout the image, so it makes sense that we'd see clusters form a large fraction of the total area, as they are bound to be more uniformly distributed in all areas of the image on average.

In the second frame, the clusters are larger, which means that there are larger areas with only a few clusters within a short radius. As the radius gets larger, these bigger clusters start to appear.

Next, we can evaluate whether it is possible to discern anything about the shapes of the clusters, by manipulating the pixels of frame 2 in the former Figure to shape all clusters like circles, according to their former size, so as to retain the same amount of colored pixels.

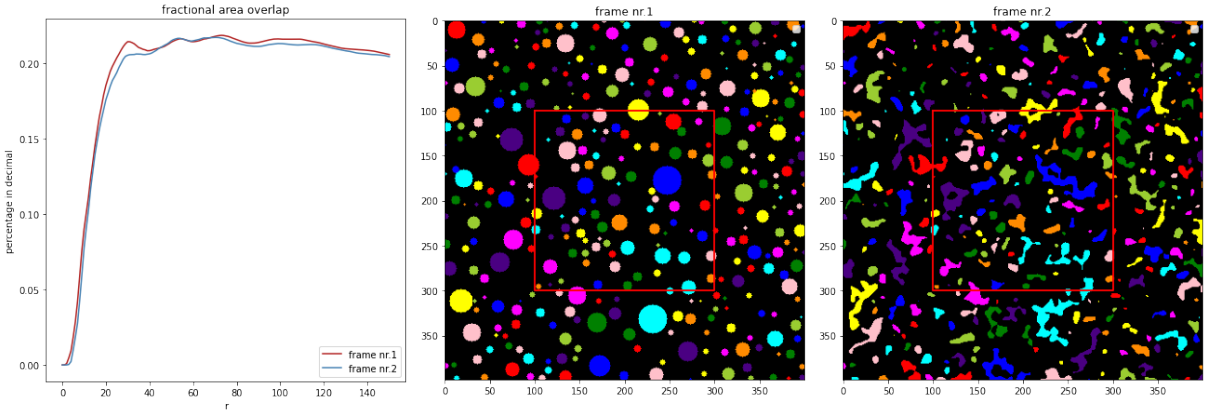


Figure 28: Shape relation measures of synthetic images with different geometric shapes

Viewing the fractional area overlap in Figure 28, frame 1 with round-shaped clusters takes up a slightly larger fractional area overlap at different intervals of r, but it is a minuscule difference, which leads us to believe that our method of averaging the fractional area overlap across all clusters will make any information about the geometric properties of the

clusters less apparent in the graph.

In summary, the relational shape measure method can offer insights when comparing images in this manner, and how early our fractional area overlap measure spikes, can indicate if we have many small and evenly distributed clusters, or a few bigger clusters, creating larger gaps in the image without clusters. It cannot say anything conclusively about the shape of clusters, because we are taking an average across all clusters.

5.2.3 Subset selection for improved performance

The analysis of microscopy images in our study presents a unique challenge due to the large size of the images captured in the video. Notably, the time complexity of certain algorithms, particularly the implementation of Ripley's K function, becomes significant when processing the entire image. This is because the implementation has to measure the distance from every protein cluster, to every other cluster in the frame, and as such the time complexity of the algorithm is lower bounded by $O(N^2)$ where N in this case is the amount of clusters in the frame. When accounting for the gradient descent algorithm the image size because an even greater concern. This is because the implementation of gradient descent with graduated non-convexity uses the shape relation measure of the synthetic image as its model, on which to minimize the loss. As a result, the time complexity of the parameter optimization algorithm is lower bounded by $O(w \cdot e \cdot 3N^2)$ where w is the length of the array of weights from the GNCA, and e is the number of epochs for which to iterate. Because of this reason, a big motivation of ours has been to identify the minimally required subset of the image that can yield reliable and accurate results while improving algorithm efficiency.

When selecting a suitable subset for running the algorithms, we must consider different criteria such as both the size, but also from where in the original image the subset is taken. Because of the inconsistencies with regard to the protein cluster distribution, as previously mentioned in Section 2.1.1, the top and bottom of the images need to be confronted somehow. One potential way to combat these problems could be to employ an image correction method and try to predict a more accurate cluster distribution in these regions. However, because of the sheer size of the images, do not require the entire image in order to perform an adequate measure of the shape relations.

To counter the irregularities we instead simply choose the center portion of the image as it strikes a balance between capturing the spatial information of the clusters while ignoring the impact of the biased regions in the top and bottom. As such, we believe this region provides a representative sample of the protein cluster distribution, enabling meaningful analysis while reducing computational complexity. To ensure consistency and reduce errors resulting from local fluctuations while still maintaining a reasonable computational time for the algorithms, we simply ran the algorithms for different image sizes and chose the largest size for which the runtime was reasonable.

5.2.4 Assumptions of the prior distributions

During the implementation of the gradient descent algorithm, the initial guess for the optimal parameter set can play a significant role in determining the convergence and eventual outcome of the optimization process. Because the nature of the loss landscape is unknown, we have to assume that selecting an appropriate initial guess is crucial for finding a favorable local minimum. Initially, we chose a set of parameters based on which generated a synthetic image, in which the mean protein size most closely resembled that of the real image. This method of choosing a set of parameters for running gradient descent relies on the assumption that there is a correlation between the mean size of the clusters and the corresponding analysis.

This is a very crude method for initializing GD because while it may generally seem like the two should correlate, there are certainly cases where they do not. The most obvious example is when the synthesized image contains many more or fewer clusters than the real image. In this case, the two might have a similar mean size of proteins, however, the resulting shape relation measures would look very different.

When deciding the prior for GD for every preceding image in the video, we simply chose the optimized set of parameters found for the image before. This method is similarly based on the assumption that there is a correlation between optimized sets of parameters for images that are close to each other, and thus that their relational shape measures are similar.

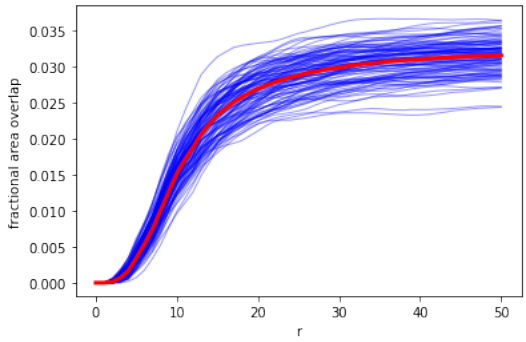
This assumption seems very plausible, especially if the gradient descent algorithm is run for adjacent frames of the video. However, this is not the case for our experiment because of the time complexity of the algorithms being run. Likewise, if there is only a limited amount of differences in the shapes and amounts of clusters from each frame to the next, it seems that the assumption intuitively holds. To test this, one would have to find out how different subsequent frames can be from each other before using the previously optimized set of parameters stops being a good prior. Conclusively, the reason as to why we have chosen these two approaches is because of their simplicity, and because they work as intended within the scope of the project.

5.3 Evaluating models

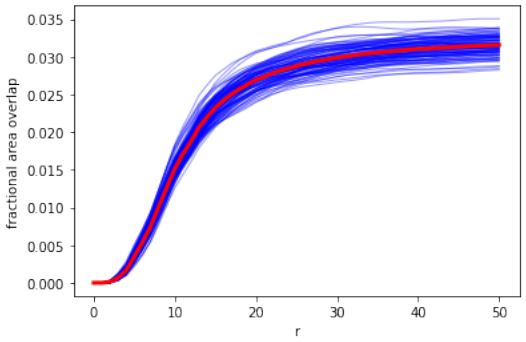
5.3.1 Approximating the expected fractional area overlap

In Section 3.4.2 we introduced the challenge of approximating the expected fractional area overlap curve generated by the synthetic data. The problem arises from the fact that computing each curve is very computationally expensive, and thus avoiding having to calculate unnecessary curves is highly beneficial. We want to find a reasonable balance between the number of averaged curves which is an accurate representation of the expected curve while keeping the number of computations relatively low. The distribution of these

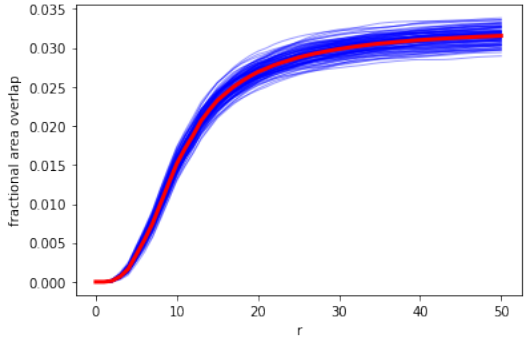
approximations using different numbers of averaged curves is depicted in Figure 29. The thick red curve represents the true expected value, or the mean of the sample curves, which we derived by calculating the average of 1000 fractional area overlap curves. As expected, the variance of each distribution decreases as the number of curves used to calculate the average increases. In other words, the accuracy or robustness of the model increases as we use more curves to approximate the expected curve.



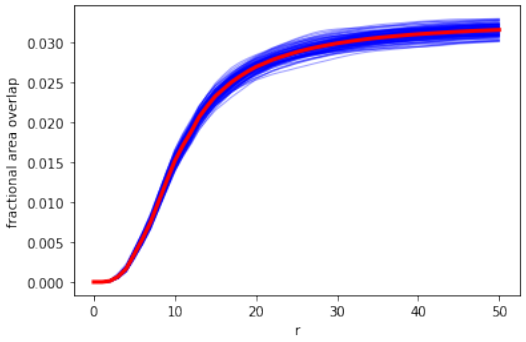
(a) Distribution of fractional area overlap curves



(b) Distribution of mean of 3 fractional area overlap curves



(c) Distribution of mean of 5 fractional area overlap curves



(d) Distribution of mean of 10 fractional area overlap curves

Figure 29: Variance of expected fractional area curves. The red curve represents the true expected mean. $(\alpha, \tau) = (2.6, 0.2)$

Ideally, we want our variance to be 0, since this would eliminate the randomness and uncertainty of the model, but unfortunately, this isn't realistic, since it would require an average of an infinite number of curves. This raises the following question - at which threshold is the variance, and thus the accuracy of the model, good enough for our analyses? A reasonable strategy for helping us answer this question is to plot the relationship between the number of averaged curves and the average variance in a graph, as seen in Figure 30. The variance of each point is calculated by the following equation:

$$Var(X) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}$$

Where $\bar{x}$ represents the mean, which is the corresponding value of the red curve, and n denotes the number of graphs. This means that the average variance of each curve is given

by:

$$\frac{\sum_{i=1}^{N} Var(X_i)}{N}$$

Here N is the number of points in our graph, which is 50 in our case.

It turns out that this decreasing tendency follows a reciprocal function. As the number of averaged curves increases, the rate of change of the average variance decreases. This indicates that increasing the amount of averaged curves has diminishing variance significance. Hence, one could argue that there exists a threshold where increasing the number of curves doesn't provide any noticeable accuracy benefits. By visually assessing the graph we determined that 5 is a suitable threshold since the curve seems to significantly stagnate after that. Thus, we chose that the ideal approximation of the expected fractional overlap curve, both in relation to accuracy and computability, is achieved by calculating the average of 5 fractional overlap curves.

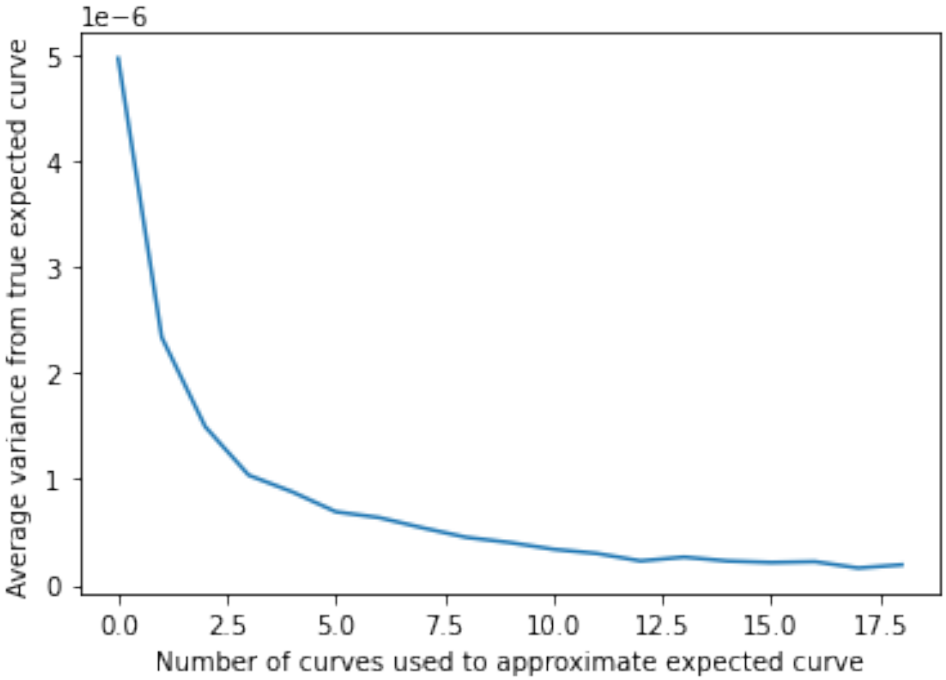


Figure 30: Variance of expected fractional area curve. $(\alpha, \tau) = (2.6, 0.2)$

5.3.2 Efficacy of graduated non-convexity

In previous sections of this thesis, we introduced the Graduated Non-Convexity Algorithm (GNCA) as a potential aid to GD in the context of non-convex loss functions. The focus of this section, therefore, will be a comparative study: we will scrutinize the performance of standard GD, and then we'll delve into how this performance evolves when GNCA becomes part of the optimization mix. Our primary aim here is to apply different evaluation techniques, to discern if and when the incorporation of GNCA proves advantageous for

the optimization process.

The initial exploratory step in comparing these two optimization methods is to look at the progression of loss throughout 100 epochs of GD, without and in combination with GNCA. In its initial implementation, GNCA carries a large constant factor to amplify the influence of the prior distribution on the posterior. This method is intended to direct the optimization process toward parts of the parameter space where we have a better chance of finding a global minimum.

By watching how the loss values change through these 100 epochs, we can learn about how quickly and in what way each model gets better. As we reach the last iteration of GD with GNCA, when the constant factor is nullified, we will investigate if the introduction of GNCA resulted in real improvements.

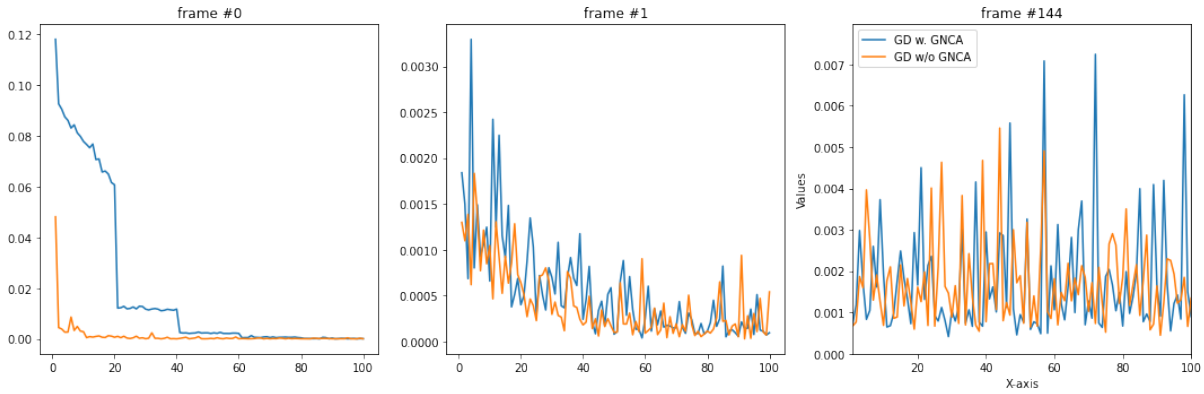


Figure 31: Loss development 100 epochs, GD with and w/o GNCA

In Figure 31, a visualization of the results of Gradient Descent (GD) implemented both with and without the Graduated Non-Convexity Algorithm (GNCA) is presented. This comparison allows us to examine the loss landscape over three different frames. Starting with frame #0, the loss curve of GD combined with GNCA shows distinct drops at 20-epoch intervals. This behavior can be tied back to the additional loss originating from the prior distribution. If this distribution contains a large constant factor, it nudges the gradients towards the prior distributions of our weights (α, τ). As we move to frame #1, the trend we identified earlier starts to fade. This shift can be attributed to GNCA's structure, in which the optimal parameters from one frame serve as starting values for the next frame. Consequently, the path of loss now seems to align more closely with the iterations rather than the previously noted pattern. The final frame we examine, #144, presents an interesting development. Here, GD combined with GNCA seems to perform slightly worse and shows more variability compared to standard GD.

However, drawing concrete conclusions from these limited observations about the benefits of GNCA would be premature. While our results do not point towards any significant performance improvements with GNCA, they do indicate the need for a more thorough

and systematic testing approach. Such a method would enable us to capture any potential performance patterns in the complex loss landscape.

To systematically assess the performance of GD w/GNCA, we consider an approach that involves doing the optimization run on a single frame but conducted multiple times with random (α, τ) values. We want to examine if the starting parameters have any effect, and if "bad" guesses affect the optimization process differently depending on the model. As explained in Section 3.3.3, the smoothing and thresholding parameters are interconnected. A higher smoothing parameter results in a more condensed distribution of values prior to thresholding being applied. This necessitates a random selection of our smoothing parameter, while the threshold parameter must correspond to values within defined percentiles. Otherwise, we risk setting a threshold that either excludes all pixel values or includes them all. If that were to happen, the optimization process will most likely crash. For instance, an abundance of pixels could lead to a large loss and consequently drastically alter the gradients. This, in turn, would result in all pixels being eliminated by the threshold in the ensuing iteration, causing there to be zero clusters in the plot.

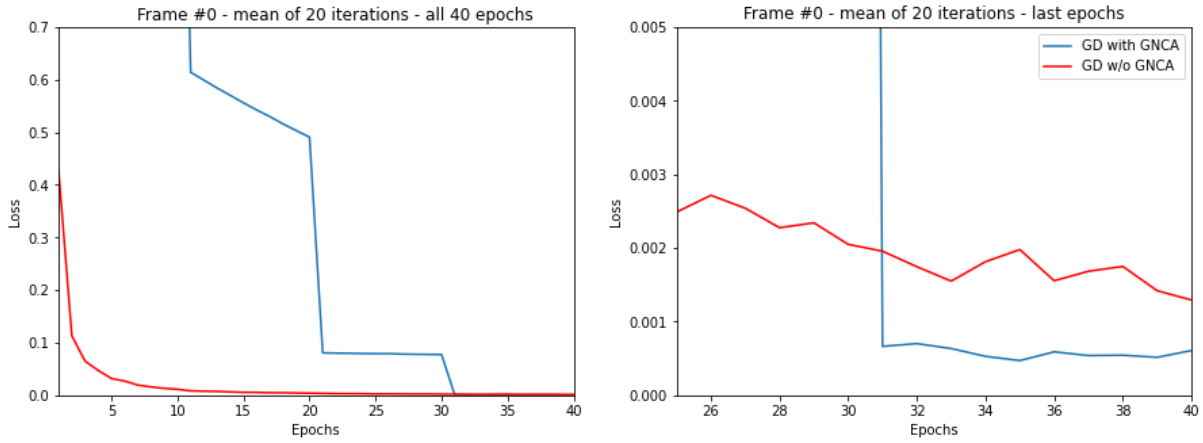


Figure 32: Loss development 40 epochs, mean of 20 iterations with random start parameters

In Figure 32 above, we present two plots side by side of our results. Both plots display the loss output at different epochs. The blue curve represents the mean of 20 iterations of optimizing the gradients, all with different random starting values, and the red curve represents the same, just using normal gradient descent. The plot on the left shows the whole optimization process, which follows a trend that is not surprising and mimics what we have observed before GNCA steadily converges towards the same level of loss as the non-GNCA curve. This pattern demonstrates that the GNCA method is definitely not showing signs of being a worse optimizer, than normal GD.

The right plot provides a closer look at the final epochs. In this section, the GNCA curve is noticeably below the non-GNCA curve. This divergence, while not dramatic, indicates our method's advantage in performance. Our implementation of GD w/GNCA appears to

reach parameter weights that give a lower loss score once the λ constant is zero, which happens in the final 10 epochs.

In a broader context, GNCA appears to improve performance when the initial parameters deviate from the optimal ones, and a suitable measure for the prior distribution has been established.

5.3.3 Limitations to measuring performance

Our analysis indicates that our model shows improvement when running parameter optimizations with graduated non-convexity compared to without. These improvements are measured quantitatively by looking at our defined loss function, and seeing how it decreases throughout training.
However, trying to conclude the actual effectiveness of our model from these loss values alone is not really possible. While we've seen indications of improvement in our model's ability to fit the training data, this doesn't necessarily mean that our model is better in a broader sense. For example, we have throughout the project only had access to one set of observations on which the model was both trained and evaluated. This may very well introduce bias, which we are simply not aware of, and as such can't be corrected for.

6 Conclusion

In this paper, we've presented a quantitative method for spatial analysis, to be used for analyzing the coagulation of casein micelles during the formation of curds in milk. This method builds upon the theoretical foundation of Ripley's K-function, and calculates a set of shape relational measures, based on the intersections between the expanding contour lines of each reference object and the observed objects. The method quantifies these measures as summary statistics graphs, and through carefully chosen testing scenarios, we believe these measures can provide insights that relate to the size and spatial distribution of protein clusters in different stages of protein coagulation. This method allows for an easy understanding of the average spatial distribution in an image by computing the mean measures, hence interpretability is predominantly limited to general trends within an image. This means that the geometric properties of our clusters, not directly related to their size, may not be easily discernible from these measures.

To gain further insight into the behavior of these coagulating proteins, we developed a method for synthetically replicating these images. Because the movement of the casein proteins is believed to be somewhat equivalent to the random walk occurring during Brownian motion, we can use this to synthesize artificial data. This is achieved by initializing an image with normally distributed i.i.d. noise, smoothing said image with a Gaussian filter, and then thresholding it to a certain value. The Gaussian filter simulates the diffusion process occurring during the random walk, providing images that can then be adjusted by tuning the standard deviation of the filter, and the threshold.
By analyzing the interrelation between smoothing and thresholding, we were able to identify a positive trend between the smoothing parameter and cluster size. Discovering an appropriate polynomial fit for this relationship enabled us to adjust the average cluster size by fine-tuning parameters, which is crucial to find a model which can emulate microscopical images at different time frames.
These parameters were subsequently optimized with gradient descent and the incorporation of graduated non-convexity to minimize the loss between the shape relation measures of the real and expected synthetic image. This was done to maximize the probability that the synthesized images resemble the real ones, letting us emulate the spatial patterns at different stages of the coagulation process.

Applying the optimal parameters relative to each time frame, we observe encouraging results; the shape measures effectively emulate the real observations to a satisfactory degree. Despite this, we recognize there are areas in our study yet to be explored that could improve our model. The inability to test our model with new data, and the effectiveness of our chosen methods, such as Graduated Non-Convexity, being limited by the accuracy of our estimated Prior distribution, point to areas that need additional investigation. However, this can provide a solid foundation for FOOD and other specialists to investigate further the possible links between our model and the corresponding biological processes.

7 Bibliography

References

[1] The science of cheese. *Science Learning Hub*, 2020.

[2] Andrew Blake and Andrew Zisserman. *Visual Reconstruction*. The MIT Press, 09 1987.

[3] Philip M. Dixon. *Encyclopedia of Environmetrics*. John Wiley Sons, Ltd, Chichester, 2002.

[4] Ken-Iti Sato. *Lévy processes and infinitely divisible distributions*. Cambridge studies in advanced mathematics. Cambridge University Press, Cambridge, England, November 1999.

[5] Shuang Song, Yuanjie Zheng, and Yunlong He. A review of methods for bias correction in medical images. *Biomedical Engineering Review*, 1(1), 2017.

[6] Hans J. Stephensen, Anne Marie Svane, Carlos B. Villanueva, Steven A. Goldman, and Jon Sporring. Measuring shape relations using r-parallel sets. *Journal of Mathematical Imaging and Vision*, 63(8):1069–1083, 2021.

[7] Richard Szeliski. *Computer vision algorithms and applications*. Springer, London; New York, 2 edition, 2022.

8 Appendix

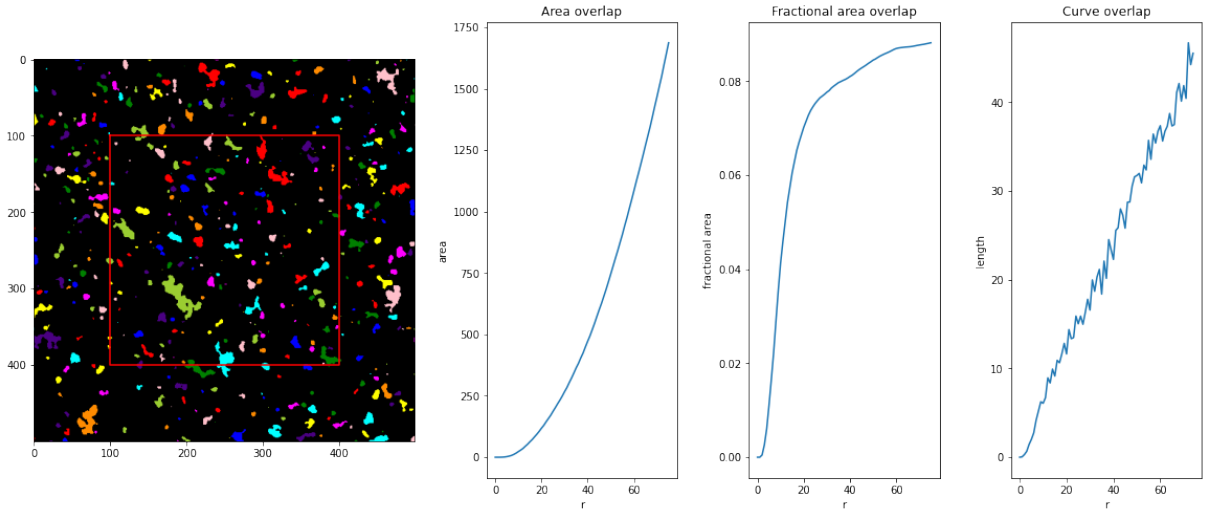


Figure 33: Example of average shape measures of all clusters within bounding polygon

Return to section 3.2.4

```
def generate_poly(size=800, percentile=80, degree=4):
    im = np.random.normal(size=(size, size))
    tau_values=[]

    for i in np.linspace(1,3,30):
        imFilt = filters.gaussian(im, i)
        tau_values.append(np.percentile(imFilt,(percentile)))

    x=np.linspace(1,3,30)
    y=tau_values
    coefficients_lower = np.polyfit(x, y, degree)
    polynomial_lower = np.poly1d(coefficients_lower)

    return polynomial_lower
```

Return to section 3.3.3