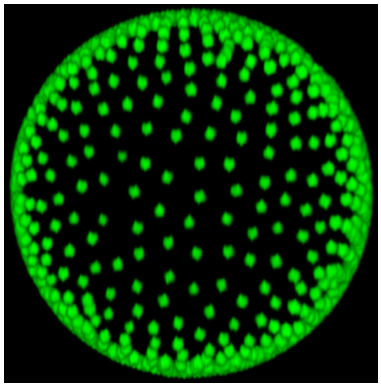
Anisotropic Colloid Ordering On Liquid-Liquid Interfaces



THESIS

submitted in partial fulfillment of the
requirements for the degree of

BACHELOR OF SCIENCE
in
PHYSICS

Author :	J. Hockx
Student ID :	1300598
Supervisors :	D. Kraft,
	V. Meester
2^{nd} corrector :	M. van Hecke

Leiden, The Netherlands, July 30, 2015

Anisotropic Colloid Ordering On Liquid-Liquid Interfaces

J. Hockx

Huygens-Kamerlingh Onnes Laboratory, Leiden University
P.O. Box 9500, 2300 RA Leiden, The Netherlands

July 30, 2015

Abstract

There has been tried to develop a suitable experimental setup to study the ordering of anisotropic colloidal clusters at liquid-liquid interfaces. PMMA colloids and glycerol - CHB-decalin interfaces have been used for this system. Different configurations of interfaces have been made and analyzed. A suitable tracking code has been developed to analyze the ordering of anisotropic colloidal clusters as well. This code can track colloidal clusters in two and three dimensional space.

Contents

Chapter 1

Introduction

Since the dawn of Darwins Theory of Evolution, human kind was given an explanation for the origin of the unbelievable complexity of mother nature [1]. With the advent of medical sciences Darwin's theory became even better understood. Scientists found that the Selection, Variation, Reproduction (SVR) algorithm [1] was even more apparent than they expected. Even for viruses which are not qualified as living organisms [2], the SVR algorithm can be applied. To quote Zandi et al. [3]: *"Although the synthesis of artificial protein cages is a rapidly developing area of materials science, the design criteria for self-assembled shells that can reproduce the remarkable properties of viral capsids are only beginning to be understood."*. Since the building blocks for virus capsids consist of proteins, it is difficult to build an artificial virus capsid. The problem with proteins is that they are generally very complex. They are relatively small in size, can have all different kinds of shapes and are difficult to image.

To understand ordering of proteins in virus capsids a model system is used to get direct in vitro information. Instead of proteins, colloids could be used as a model system to build an artificial virus. Colloids are particles with spatial dimensions of a few nanometer up to a few micrometer and can consist of various of materials. Due to their size colloids can be easily studied with optical microscopy. They are big enough to image and small enough to be influenced by thermal fluctuations. Thermal fluctuations causes the colloids to experience Brownian motion. Due to interparticle interactions colloids can self-assemble into larger structures [4] [5]. Some of these structures are shown in figure 1.1.

Colloids can be attached to spherical liquid-liquid interfaces to form a col-

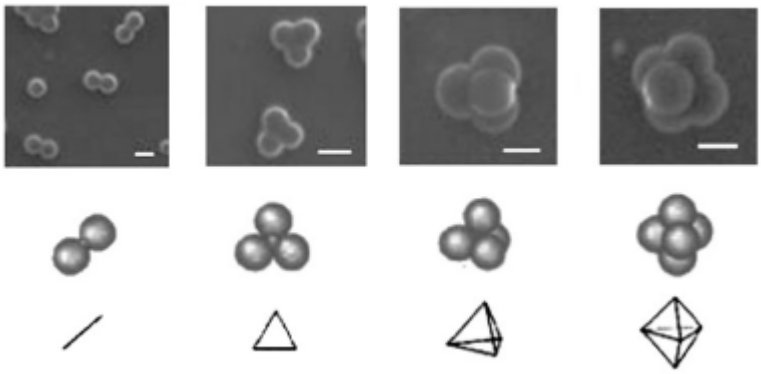


Figure 1.1: *Scanning Electron Microscope (SEM) images of self-assembled large colloidal structures are shown, increasing in size from left to right. The colloids were made of silica. The scale bars are 200 nm [4].*

loidal shell. These shells are formed spontaneously, since the free energy is reduced when the surface area is reduced by the attachment of colloids. Colloids order into lattices on the interface. The ordering of colloids on liquid-liquid interfaces has been proved curvature and density dependent [6] [7].

The ordering of anisotropic colloids on curved interfaces have not been studied yet. This is mainly due to the fact that the analysis of anisotropic colloids at liquid-liquid interfaces in three dimensions is difficult. The goal of this research is to develop a suitable experimental setup to study the ordering of anisotropic colloidal clusters at liquid-liquid interfaces. And a suitable tracking code needs to be developed as well to analyze the ordering.

1.1 Outline Of This Research

An experimental setup was developed to build large structures that consist of colloids self-assemled onto liquid-liquid interfaces. It is shown that colloids can order into lattices on the liquid-liquid interfaces.

The ordering of the particles is analysed using a program language called Python. The computer programs used are Spyder and IPython notebook that both come as a package in Anaconda. Currently there is already tracking code available for Python for spherical colloids called Trackpy [8], but

a code for identifying clusters and within those clusters the right spatial coordinates (per colloid) did not yet exist for Python.

Chapter 2

Theory

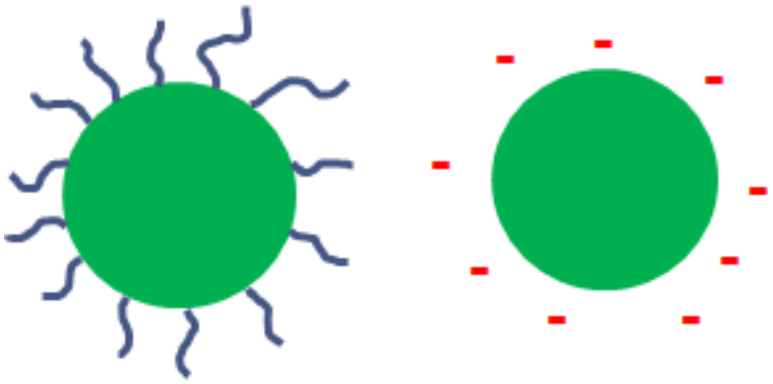
2.1 Colloids

Colloids are particles with length scales of nanometers to micrometers that exhibit Brownian motion [9]. When colloids are suspended in a liquid medium, they aggregate due to Van der Waals forces unless they are stabilized. Van der Waals forces are short ranged; to counteract the van der Waals forces, long range repulsive forces are needed. Stabilization can be achieved in two ways; either steric or electrostatic (figure 2.1). When colloids are sterically stabilized, polymers are absorbed to the surface of the colloids. From thermodynamics it is known that the entropy in a system wants to maximize and by doing so the energy is minimized. When the polymers of two neighboring colloids intertwine, the possible configurations of positions in space of the polymers reduces and therefore the entropy reduces. To maximize the entropy and therefore the energy, the colloids experience a repulsive force. When colloids are electrostatic/charge stabilized, the interaction range can be determined by the Debye screening length.

The Debye screening length, λ, is defined as

$$\lambda = \left(\frac{\epsilon_0 \epsilon_r k_b T}{\sum_{i=1}^{N} z_i^2 e^2 n_i^\infty} \right)^{1/2} \tag{2.1}$$

Where ϵ_0 is the permittivity of the vacuum and ϵ_r is the relative permittivity. k_b is the Boltzman's constant, T the (absolute) temperature, z_i the valence, e the elementary charge and n_i^∞ the bulk concentration. In the derivation of the Debye length the approximation of a small surface po-



(a) *Sterically Stabilized* ***(b)*** *Charge Stabilized*

Figure 2.1: *On the left an illustration of a sterically stabilized colloid with polymers absorbed on the surfaces is shown. On the right an illustration of a charge stabilized colloid is shown. The particles that are charged are attached to the surface of the colloid.*

tential is used. From equation 2.1 it can be seen that the interaction range is influenced by the ion concentration in the suspension. [9].

2.2 Physics Of Liquid-liquid Interfaces

When two liquids are in contact various phenomena play a role. In liquids there is an interplay of cohesive and adhesive forces on a molecular scale. When a liquid is in contact with another liquid, gas or solid a surface tension is present at the interface. In figure 2.2 an illustration of a water-air interface is shown. Inter-molecular forces between water molecules in the bulk are cohesive, but the forces between water surface molecules and air molecules are adhesive. The cohesive forces between water molecules are stronger than the adhesive forces between water and air molecules, because water molecules in the bulk form hydrogen bonds. For each water molecule at the water-air interface, the amount of neighboring water molecules is less than in the bulk. The water molecules on the surface can therefore make less hydrogen bonds and the molecules thus have a higher energy. To reduce the energy in the system, the amount of water molecules on the surface is minimized. This in turn causes a minimization of the surface area. The phenomena of minimizing the surface is called surface tension. [9].

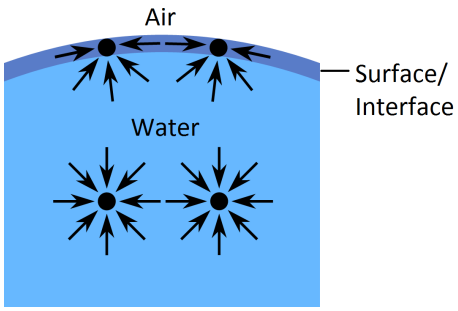


Figure 2.2: *Illustration of part of a water droplet. The molecules are showed as black dots. By cohesive forces they attract each other, this is shown with black arrows. The molecules at the surface can forme less hydrogen bonds with their neighbors than the molecules in the bulk, therefore there is a resulting surface tension [10].*

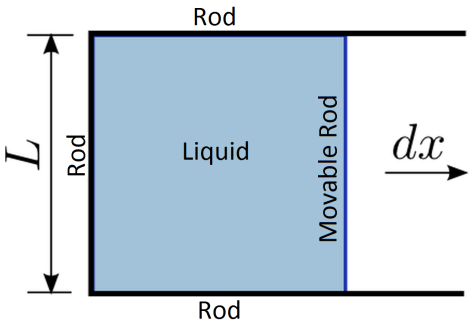


Figure 2.3: *Illustration of a thin film of a liquid. There are three unmovable rods and one rod of length L that can be moved to the right [10].*

For a thin liquid film, the surface tension can be easily derived. In figure 2.3 an illustration of a thin film stuck between three unmovable rods and one rod that can slide to the right is shown. The sliding rod has a length L and is moved by the amount dx. The force required to hold the rod in place, F, is inversely proportional to the length of the rod. The surface tension γ can be defined as

$$\gamma = \frac{1}{2}\frac{F}{L} \tag{2.2}$$

which is a force, F, per unit length, L. The factor of a half comes from the fact that in a thin film two surfaces are present.

A definition for the surface tension can also be derived using statistical thermodynamics. From the first law of thermodynamics it follows that:

$$dE = TdS - PdV - dW_{\text{non-PV}} \tag{2.3}$$

Where E is the energy, T is the temperature, S is the entropy, P is the pressure, V is the volume and $W_{\text{non-PV}}$ is the work done not by pressure or volume. The Gibbs free energy, G, can be written as

$$dG = dE + PdV + VdP - TdS - SdT \tag{2.4}$$

By inserting equation (2.3) into equation (2.4), the definition of dG becomes

$$\begin{aligned} dG &= TdS - PdV - dW_{\text{non-PV}} + PdV + VdP - TdS - SdT \\ &= -dW_{\text{non-PV}} + VdP - SdT \end{aligned} \tag{2.5}$$

If the pressure and temperature are held constant, the Gibbs fee energy becomes

$$dG = -dW_{\text{non-PV}} \tag{2.6}$$

The work done in this equation can be substituded by rewriting 2.2 to

$$W = Fdx = \gamma 2Ldx = \gamma dA \tag{2.7}$$

By inserting equation 2.7 into 2.6, the surface tension becomes the surface derivative of the Gibbs free energy.

$$\gamma = \left(\frac{\partial G}{\partial A}\right)_{T,P} \tag{2.8}$$

Here the surface tension is defined as energy per unit area. From equation (2.8) it becomes apparent why the surface is reduced to a minimum. To reduce the Gibbs free energy of the system, a system with interfaces will always try to reduce the surface area [9].

Another important relation of interfaces is the relation between the pressure and the curvature. The pressure difference between two liquids is related to surface tension and the principle radii of curvature by the La Place equation

$$\Delta P = \gamma \left(\frac{1}{R_1} + \frac{1}{R_2}\right) \tag{2.9}$$

Here ΔP is the pressure difference between the two liquids, and R_1 and R_2 are the principle radii of curvature of the surface [9].

The contact angle is also related to the surface tension. The shape of a liquid is among other things defined by the contact angle. In the image below there are three phases; a gas (G), a liquid (L) and a solid phase (S).

The contact angle (θ_c) is the angle between the liquid-gas and solid-liquid surface. The angle lies at the boundary point with all three phases. This angle is related to all the surface energies shown in figure 2.4 by Young's

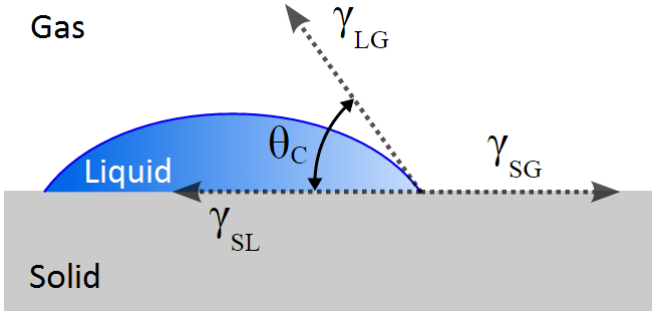


Figure 2.4: *An illustration of a dome of a liquid lies on top of a solid with a gas surrounding both. There are three surface energies to be seen, all indicated with subscript; G - Gas, L - Liquid, S - Solid. The contact angle (θ_c) is the angle between the liquid-gas and solid-liquid surface. The angle lies at the boundary point with all three phases [11].*

equation (2.10). The difference in surface energies of solid-liquid and solid-gas divided by the cosine of the contact angle, will give the liquid-gas surface tension/energy [9] [12] [13].

$$\gamma_{LG} \cos\theta_c = \gamma_{SG} - \gamma_{SL} \tag{2.10}$$

2.3 Crystal Ordering Of Colloids On Liquid-Liquid Interfaces

The surface area in a liquid-liquid system can be reduced by the adsorption of particles to the interface [9]. An example of this spontaneous adsorption is the emulsification of oil droplets in a suspension of particles, forming a pickering emulsion [14]. Dinsmore et al. [15] showed that these shell-like structures are promising candidates for artificial capsules since the size, permeability and strength of these colloidosomes can be controlled.

When colloids adsorb on liquid-liquid interfaces, one can often observe a certain order. The role of curvature on the ordering of colloids on liquid-liquid interfaces was studied by Ershov et al. [6]. Here the ordering of 1.2 µm sized colloids with a fluorescent polystyrene core and a shell of poly(N-isopropyl acrylamide-comethacrylic acid) on oil-in-water interfaces was studied. Droplets with anisotropic curvature were prepared on substrates that were made with soft photolithography.

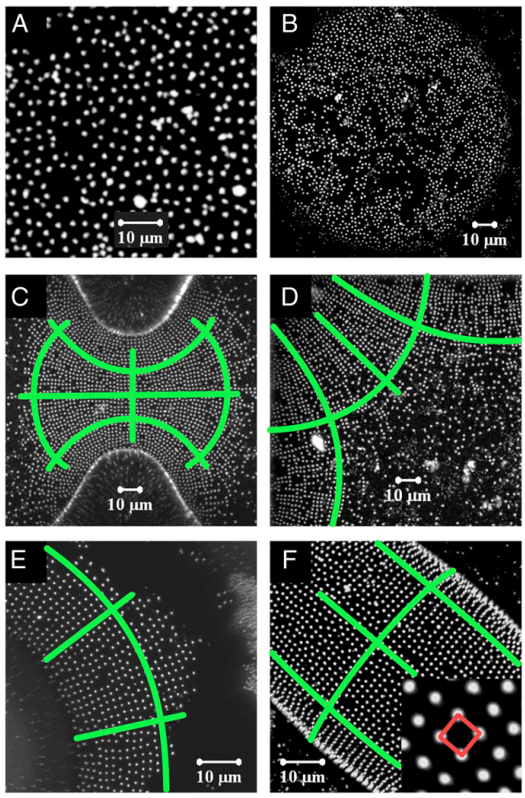


Figure 2.5: *Confocal microscopy images of colloids attached to oil-in-water interfaces of different curvatures are shown. They are maximum intensity projection images along the z-axis. The green lines show the directions of the principle curvatures. The colloids appear to order randomly for the flat interface (figure 2.1A) and for the spherical interface (figure 2.1B). For anisotropic interfaces (figure 2.1C-F), the colloids order in a square-like pattern (figure 2.1F inset) [6].*

Figure 2.5 shows confocal microscopy images of colloids attached to liquid-liquid interfaces of different curvatures. For a flat interface (figure 2.5A) and a spherical interface (2.5B) the ordering appears to be random. On the contrary, when the curvature is anisotropic (2.5C, D, E, F), the colloids order in a square-like pattern. This result was unexpected because on spheres hexagonal packing is the highest packing density. Their explanation for these findings was that the cappilary attraction balanced the long-ranged repulsive interaction because of the charges of the colloids.

To analyze the positional and translational order of the colloids, the radial distribution function, $g(r)$ was used. It can be determined by using the

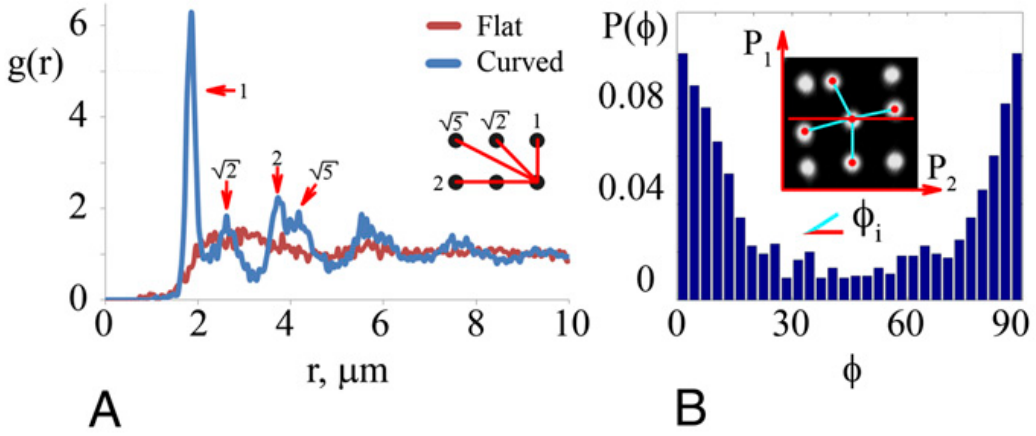


Figure 2.6: *On the left the radial distribution function $g(r)$ is shown for the flat interface and for the dumbbell shaped interface. For the dumbbell the characteristic peaks can be seen. On the right the probability distribution of the angles ϕ is shown. ϕ is the angle between the principal curvature axis and interparticle bonds as shown in the inset. [6].*

coordinates of the center of the particles. The radial distribution function describes the possibility to find a neighboring particle at a particular distance from the center particle. The radial distribution function is defined as

$$g(r) = \rho^{-2} \left(\sum_i \sum_{j \neq i} \delta\left(\overrightarrow{r_i}\right) \delta\left(\overrightarrow{r_j} - \overrightarrow{r_i}\right) \right) \tag{2.11}$$

where ρ is the average density of particles, r_i is the center particle and r_j the most probable location for surrounding particles of r_i. The summations run over all values of j and i, meaning all particles are examined.

In figure 2.5A the $g(r)$ found for a flat and a dumbbell shaped interface are plotted. Whereas no structures characteristics are observed for the flat interface, peaks at characteristic particle distances for a square lattice are found on the dumbbell shaped interface. The probability distribution is shown in 2.5B. The colloids order mostly along the local principle curvature axes.

Besides curvature, density influences the order in the system as well. In figure 2.7A and B a dumbbell shaped droplet with a low particle density and a high particle density are shown respectively. The ordering was ana-

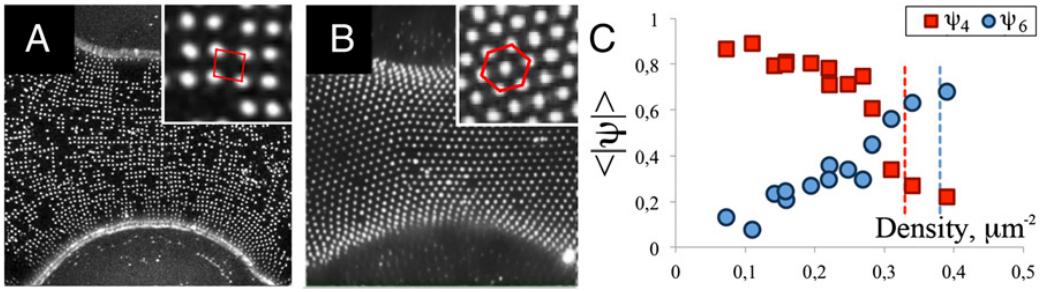


Figure 2.7: *Polystyrene colloids on a curved interface of water-oil. Figure A shows a density of polystyrene of* 0.16 μm^{-2} *with a close up of the square ordering in the inset. Figure B shows a density of polystyrene of* 0.5 μm^{-2} *with a close up of the hexagonal ordering in the inset. Figure C shows the transition by plotting the order parameter (see equation* (2.12)*) against the density [6].*

lyzed by determining the order parameters ψ_n.

$$\psi_n = \frac{1}{N_j} \sum_j e^{in\theta_j} \tag{2.12}$$

Here, N_j is the number of neighbors of a colloid, θ_j is the angle between the bond with a neighboring colloid j and a arbitrary reference axis. The n denotes the symmetry of the axis. ψ_n lies between zero and one and indicates how large the proportion of symmetry is.

With increasing density of the colloids the crystal order changed from square-like to hexagonal-like. In figure 2.7C the transistion between the two patterns can be seen. The authors argue that the transition at higher densities is seen because the hexagonal packing becomes more energetically favourable at higher densities than the square packing.

The order in a different colloidal system was studied by William T. M. Irvine and Chaikin [7]. In this study also defects in ordering of colloids on curved interfaces were studied. The system consisted of PHSA functionalized PMMA colloids at an glycerol - CHB-dodecane interface. To optimize the system the CHB-dodecane was index matched with glycerol and density matched with the PMMA colloids. Interfaces of different curvature were studied, including spheres, domes and dumbbell-like structures as shown in figure 2.8.

With capillary bridges the interface could be stretched out to produce negative curvature. They discovered a series of transitions in the ordering

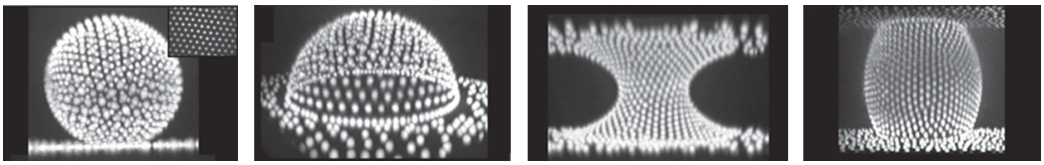


Figure 2.8: *PMMA colloids on different kind of curved glycerol - CHB-dodecane interfaces are shown. The colloids organized into crystals and showed high hexagonal ordering [7].*

of the colloids on the curved liquid-liquid interface: *"From no defects to isolated dislocations, which subsequently proliferated and organized into pleats; finally scars and isolated heptagons (previously unseen) appear"* [7]. Defects are a disruption of the topological ordering. In figure 2.9a-d a confocal image of a capillary bridge with increasing curvature is shown. In figure 2.9h-k the position and nature of the defects in figure 2.9a-d respectively are shown. It can be seen that the number of defects increase when the lattice is stretched, leading to larger interparticle distances. Pleats are seen that stretch out from the boundary to the proximity of the neck [7].

2.4 Imaging Colloids In Suspension

To study colloids in suspension optical microscopy can be used, but since in bright field background scattering interferes with the intensity signal of the colloids, confocal microscopy is often used (see figure 2.10. With this technique a sample is illuminated with a point laser. When the wavelength of the laser is in the absorption range of one of the sample components, emitted signal can be detected. In confocal microscopy a pinhole is used. The pinhole ensures that the signal of only one confocal 2D plane is detected. This prevents out of focus planes to be imaged and therefore the signal-to-noise ratio is much higher than with a bright field microscope. When the point source is moved at high speed along the focal plane covering the whole area that needs to be imaged, a two dimensional image of the focal plane can be formed. From the 2D images a 3D reconstruction can be made. This gives the possibility to track large objects in space over time [16].

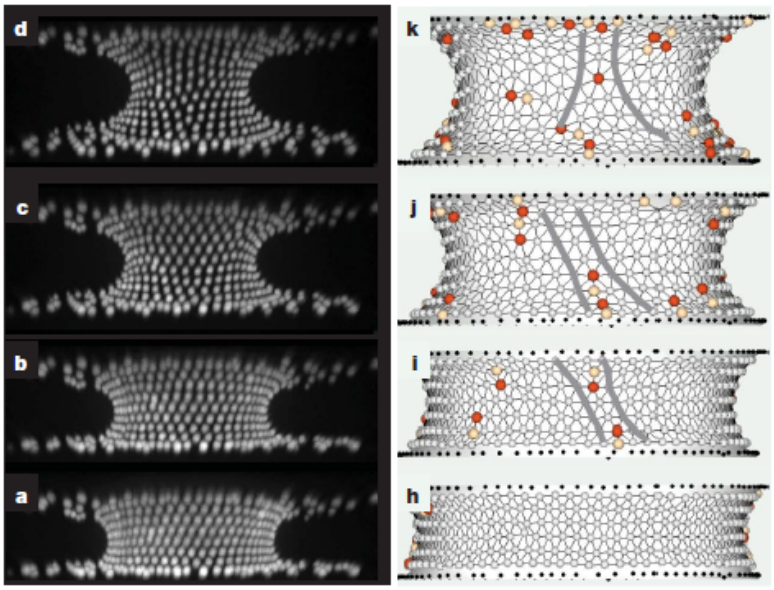


Figure 2.9: *A capillary bridge that was stretched in time. The glycerol - CHB-dodecane interface was stretched from a cylindrical shape to a highly curved shape (2.9a-d). The defects that appear over time are shown on the right; red and white are seven and five-fold defects respectively. In figure 2.1i-j pleats can be seen that begin at the boundary and vanish in the proximity of the neck [7]*

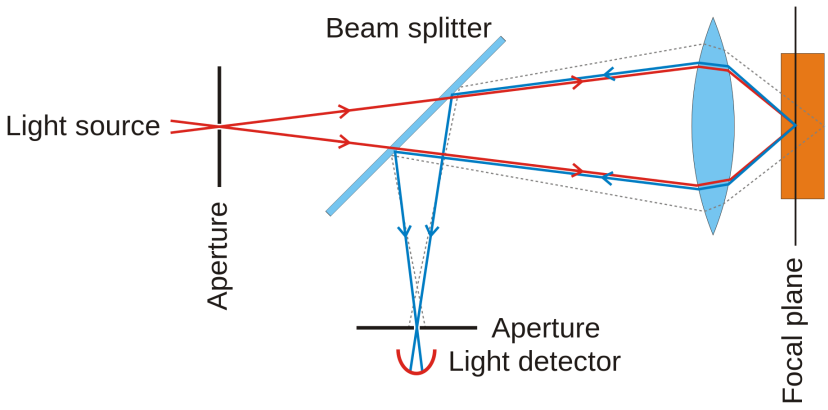


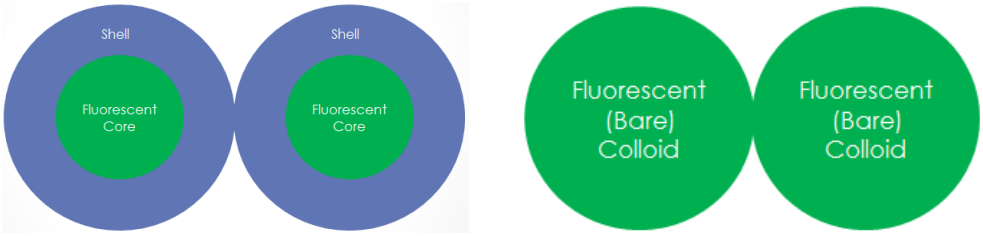
Figure 2.10: *The basic principle behind a confocal microscope. A laser as light source creates a point illumination in the focal plane. The re-emitted light needs to pass the pinhole before it reaches the detector [16].*

2.5 Imaging Colloids At Liquid-Liquid Interfaces

To study interactions between colloids at interfaces other forces, such as drift and gravity, should be counteracted. By density matching, sedimentation is prevented and overall drift caused by sedimentation removed. The core-shell makes them easier to image. This principle is shown in an illustration in figure 2.11. With a core-shell particle, only the core will get a fluorescent dye. The laser of the confocal microscope illuminates the dye in the core. Photons from the laser are absorbed by the electrons and get into an excited state. Emission from the electron falls on the detector of the confocal microscope. Without shell the colloids are in contact and their fluorescent images will partially overlap during image acquisition. This will make it harder to detect single colloids with Trackpy; a Python module that locates individual colloids by analyzing the intensities of colloids in confocal microscopy images (for more information see the next section).

2.6 Particle Analysis Using Python

To quantitatively and qualitatively study structures formed by colloids, experimental data needs to be analyzed. This can be done with a module for the programming language Python. This tracking module is called Trackpy[8] and is used in this study. It can load in confocal microscopy images and localize colloids by determining its coordinates according to the intensities in the image. Tracking in time and in 3D space is possible. When positions in time of colloids have been obtained, the overall drift



(a) *Core-shell (green-blue)* ***(b)*** *No core-shell*

Figure 2.11: *In figure 2.11a an illustration of a core-shell colloid is shown and in figure 2.11b an illustration of a colloid without core-shell is shown. With a core-shell particle, only the core will get a fluorescent dye (green) while the shell (blue) won't be seen in the image (2.11a). If there would be no shell, then there is a higher chance that the colloids lie against each other and that they have overlapping intensities in the image (2.11b)*

can be excluded, the diffusion constant of the sample can be measured and the 3D reconstruction over time can be plotted.

Trackpy refines located positions according to methods described by Crocker and Grier [17]. First an image is refined before locating particles. To subtract the background image, a convolution with a boxcar average is made. The boxcar average has a region of extent of $2w + 1$ pixels, where w is an integer larger than the radius of a particle, but smaller than interparticle distance. Noise caused by digitization in the CCD camera and by the frame grabber can be reduced by a convolution with a Gaussian surface of revolution without blurring the image. The correlation length is in general approximately 1 pixel. Therefore the Gaussian surface used in the convolution has a width of half a pixel.

The method described by Crocker and Grier [17] locates particles by using the intensity in an image. The peaks of intensity in the image are identified as potential particles only if no other pixel within a distance w is brighter. Next, only the upper 30th percentile is taken as a candidate for a particle. Then in an area of w a brightness weighted centroid is taken. If, compared to the original candidate pixel, the centroid lies more than half a pixel off the candidate location can be moved accordingly. After this the refinement can be recalculated until no more refinement can be done [17].

To store the located particles positions and to find them back efficiently, Trackpy and the code made in this study make use of an algorithm called

the k-d tree. With this algorithm, a binary tree of all the positions of located particles is made by using a decision tree in k-dimenions (hence the name k-d tree). The way this algorithm stores points in space is illustrated with an example in two dimensions as shown in figure 2.12. The algorithm picks a position from the set at random and makes this the first node in the tree. From this position it takes the number for the x-axis and compares the next position's x-axis value to it. If it is lower, a node to the left with the position is made, if it is higher the value is stored on a node to the right. Now there are two levels in the k-d tree. At the second level the process is repeated in the same way as at the first level, only now the decision to store a particle on the right or the left in the tree is based on the next axis. This algorithm is used throughout the whole tree until a level is reached where all the axis are used. If this is happens, the algorithm starts at the first axis again, cycling through all the axis until all the points are stored in the k-d tree. This process can also be illustrated in space as seen in figure 2.12, where the k-d tree divides space in the same manner until it cannot split up any more space. The empty cells that remain after the division of space, are called leaf cells [18].

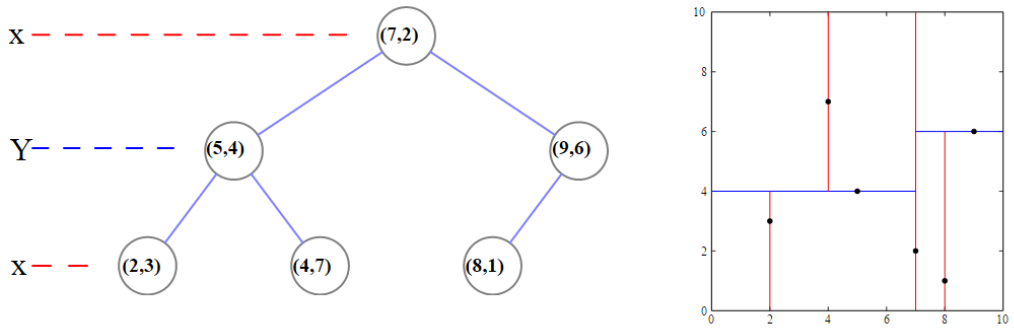


Figure 2.12: *A data set of (2,3), (5,4), (9,6), (4,7), (8,1), (7,2) in 2D space is taken and put into a k-d tree with the k-d tree algorithm. In the illustration on the left the k-d tree is represented as a tree/network, with the coordinates at nodes. In the illustration on the right, the k-d tree is represented in 2D space.*

Chapter 3

Methods

In this section the materials and methods used for this study will be described. First, the experimental part will be explained followed by an extended description of the particle analysis, for which a programming language called Python is used.

3.1 Materials

Name	Short name	Formula	Supplier
Activated Alumina		Al_2O_3	Innovative Technology, Inc.
Cyclohexyl bromide	CHB	$C_6H_{11}Br$	Sigma Aldrich
Decahydronaphthalene	Decalin	$C_{10}H_{18}$	Sigma Aldrich
Propane-1,2,3-triol	Glycerol	$C_3H_8O_3$	Sigma Aldrich
Methylbenzene	Toluene	C_7H_8	
Sodium lauryl sulfate	SDS	$NaC_{12}H_{25}SO_4$	

Table 3.1: *A list of all the materials used in this research*

3.2 The Colloidal System

3.2.1 The Colloids

In this study poly(methyl methacrylate) PMMA colloids functionalized with a fluorescent core (NBD) were used. The colloids were charge and sterically stabilized. They were provided by Dr. D.J. Kraft and were synthesized according to the method described by Elsesser and Hollingsworth [19] and Antl et al. [20]. A SEM image of these particles is shown in figure 3.1. The core was 1.3 µm in diameter and the total diameter of the core-shell was 1.7 µm. The colloids had a cross link density of 1% by weight.

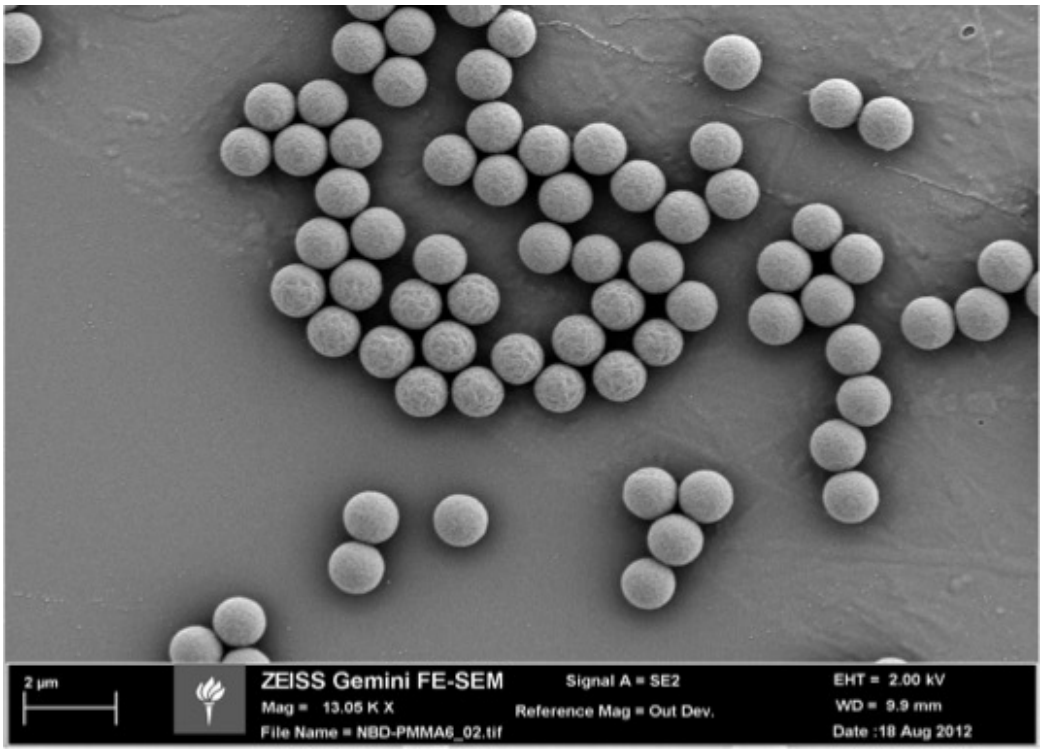


Figure 3.1: *A SEM image of the PMMA NBD core-shell colloids. Drying effects during sample preparation and the influence of the electron beam have caused some colloids to crumble*

3.2.2 Experimental Setup

For this study an interface of propane-1,2,3-triol (glycerol) and a mixture of Cyclohexyl bromide (CHB) and decahydronaphthalene (*cis*-Decalin) (CHB-

decalin) is used. Glycerol has a high viscosity compared to the other substances (see table 3.3). I will refer to this mixture as CHB-D. In this study CHB-D and glycerol are used, because glycerol nearly matches the density of the colloids (see table 3.3) and CHB-D can be *adjusted* to match the colloids their density. To density match the CHB-D with the colloids the concentration of decalin in the CHB-D mixture is 27.2% by weight, or 2.67 : 1 CHB-decalin by weight [21] [22] [23].

The medium CHB-D was density matched with the PMMA colloids. Three stock suspensions were prepared with the concentration of decalin in CHB and the concentration of colloids in CHB-D listed in table 3.2.

	Decalin In CHB (% wt)	**PMMA Colloids In CHB-D (% wt)**
Stock 1	27.2%	7.5%
Stock 2	27.7%	7.5%
Stock 3	27.4%	7%

Table 3.2: *Concentrations of decalin in CHB and PMMA colloids in CHB-D for all the stock suspensions.*

While no sign of sedimentation was observed by eye after 15 minutes of centrifugation at 1500 rpm immediately after density matching a sediment of colloids was observed between several days and several weeks after preparation.

The refractive indexes of all the substance are nearly matched and thus scattering effects were brought to a minimum. This provided for clean, images with only small distortions from the confocal microscope. The PMMA colloids were suspended in CHB-D, before CHB-D is brought in contact with glycerol. The stock suspension used for experiments were 2 months old at most.

When PMMA colloids were suspended in CHB-D the colloids need time to swell (take in the solvent). To speed up this process the colloids in the CHB-D suspension were put into an oven for one hour at 50 °C directly after making the stock solutions. When the colloids where suspended in CHB-D, they could also aggregate. To prevent this the colloids in suspension were put in a sonic bath for 30 minutes before a sample was used. These procedures were not done for every sample prepared in this study, therefore it will be mentioned in the results if the procedure was executed.

Material	**Density** (g/mL)	**Refractive Index**	**Viscosity** $(mPa \cdot s)$
PMMA	1.19	1.49	
Glycerol	1.26	1.47	1.4×10^3
CHB	1.33	1.49	2.3
Decalin	0.89	1.48	3.4
CHB-decalin	1.17	1.49	2.2

Table 3.3: *Physical variables of the used substances [21] [24].*

The liquid CHB experiences the following equilibrium reaction.

$$C_6H_{11}Br \leftrightarrows C_6H_{10} + HBr \tag{3.1}$$

In water HBr will split up into H^+ and B^- and affect the Debye length in the system [21]. To increase the Debye screening length in the system and therefore the interparticle distance, the CHB was filtered with activated alumina as described by Leunissen [21]. Glass wool was put at the bottom of a burette and on top of the glass wool small activated alumina spheres were placed. Next the CHB was poured from an Erlenmeyer flask into the burette until the activated alumina was covered. After 2.5 hours the CHB was collected and the cycle was repeated twice, but now the CHB was left for 1 hour in the activated alumina (figure 3.2). Note that approximately only one fifth of the initial volume is left after this procedure.

3.2.3 Sample Preparation Procedure

To form a curved liquid-liquid interface, a dome of glycerol was made with a glass pipette on a glass round coverslip in a round sample holder. The diameter of the round coverslip was 25 millimeter with a thickness of 0.13 to 0.17 millimeter and provided by VWR International. The sample holder was custom made by the FMD (fine mechanical service) of the University of Leiden and had an outer diameter of 3.6 ± 0.1 centimeter and an inner diameter of 1.8 ± 0.1 centimeter. Next, a suspension of PMMA colloids in CHB-D, was poured over the glycerol domes until they were fully covered with CHB-D. Finally, an extra cover-slip was placed on top of the sample holder for any sample, to prevent most of the evaporation of CHB-D. An illustration of the sample holder is shown in figure 3.3.

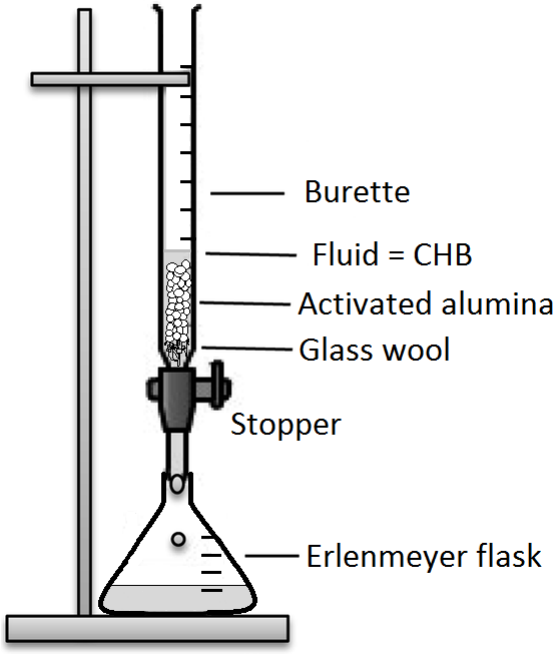


Figure 3.2: *Filtering of CHB. Glass wool lies on the bottom of the burette with the activated alumina on top. The CHB is cycled through the burette for 2.5, 1 and 1 hour respectively with use of an Erlenmeyer flask [25].*

Different colloid concentrations were used in the experiments. The percentages by weight of the colloids in CHB-D used were 0.44%, 1.7%, 7.5%. For some experiments the liquids were reversed. Here, CHB-D was put on the glass and covered with glycerol. This was done to look if electrostatic interactions could be better visualized, because on the inside of the domes colloids would become trapped. In an equilibrium state the separation distance should be much larger than colloids that are only sterically stabilized and this should become apparent inside of the dome where there is no influence of the interface [22] [23].

3.2.4 Fabrication Of Clusters

In this study clusters of spherical PMMA colloids were fabricated according to the method described by Manoharan et al. [26]. Dispersing spherical PMMA NBD6 colloids in methylbenzene (toluene), adding water and then mixing, creates small oil-in-water emulsion droplets. By evaporating all the toluene the colloids get pressed against each other, and when deswelling the van der Waals attraction increases, letting the particles stick.

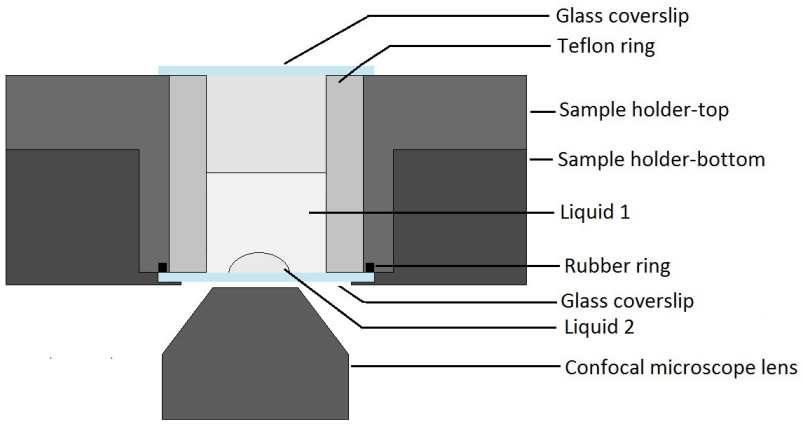


Figure 3.3: *The experimental setup as used throughout this study. A section of the round sample holder is shown with all its components. In the bottom of the sample holder a coverslip with a diameter of 25 millimeter was placed. The top of the sample holder was then screwed on until the rubber ring of the sample holder made contact with the glass coverslip. On the inside of the top of the sample holder a Teflon ring was placed. Finally, a glass coverslip with a diameter of 25 millimeter was placed on top of the sample holder.*

For this method an emulsion of PMMA colloids in toluene in water was prepared, by mixing a 2 mL suspension of 5% wt PMMA NBD6 colloids in toluene with 3 mL of 1% wt Sodium lauryl sulfate (SDS) (aq) and 3 mL millipore filtered water in a 50 mL graduated cylinder. The mixture was stirred for one minute at 8000 rpm using an emulsifier.

The mixture was then placed in a in a preheated oilbath (60 °C) to evaporate the toluene. The solution was heated for 1 hour at 60 °C followed by 2 hours and 15 minutes at 95 °C. Millipore filtered water was added when a significant proportion of the water was evaporated. To make sure the toluene was evaporated, the sample was checked regularly with a bright field microscope.

3.3 The Confocal Microscope

The experimental setup described in section 3.2.3 was placed in a Nikon Eclipse Ti microscope. All the images were obtained by the use of the same 100x oil objective. In addition an extra 1.5 objective or digital magnification could be used. Almost all the images were made by use of the

confocal microscopy technique, but on occasion bright field images were shot. A green laser with a wavelength of 488 nm was used. The detector wavelength for the green laser was 500 - 550nm. The physical length per pixel was identical in the x and y-direction. On the contrary, the physical length per pixel in the z-direction was different.

3.4 Python Analysis

3.4.1 Tracking Spherical Particles

To track particles in time and space, there has been made use of a Python program called Trackpy. To locate particles, a function called `tp.locate` within Trackpy has been used throughout this research. This function takes a raw image (passed through in the arguments of the function) produced by a microscope and can preprocesses the image with a band-pass filter to filter out noise. To do this the criteria for a high and low pass filter need to be given in the arguments, called `"smoothing_size"` and `"noise_size"` respectively. If left empty the image is not preprocessed. After preprocessing `tp.locate` takes the adjusted image and locates all the peaks of brightness. It characterizes the neighborhoods of the peaks and only those peaks that have a higher total brightness (”mass”) than the threshold (`"minmass"`) given in the arguments are identified as features. Depending on the particle's size, the program can also exclude located features that have a diameter larger than the diameter of the particle. The diameter given as argument can be a tuple, which is sometimes necessary because of the difference in physical length per pixel in the z-direction produced by the confocal microscope. The raw image and the diameter are the only *required* arguments (`"raw_image"` and `"diameter"` respectively), the rest of the arguments have either a default or are not taken into account when nothing is passed through. The values of the arguments passed rely on the properties of each image and the size of the colloids used. Therefore, all the settings used in this study are presented in the results. A typical call of the function would look like
`tp.locate(image, diameter, minmass, noise_size, smoothing_size).`

3.4.2 Tracking Clusters

Bias Problems
Trackpy can locate individual particles in an image by refining the coordinates according to the method described by Crocker and Grier [17] in

section 2.6. The method causes problems when used for locating the centroids of individual particles within a cluster. When a particle has overlapping intensity with a neighboring particle, the mass of the particle is higher on the side where the intensities overlap. Therefore, the refinement causes the centroid to lie more in the direction of the neighboring particle, from here on out called the 'close range bias' (see figure 3.4). Moreover, Trackpy cannot identify clusters at all, but only locates individual particles. The new code made in this study locates clusters and solves the bias generated by Trackpy (see appendix A).

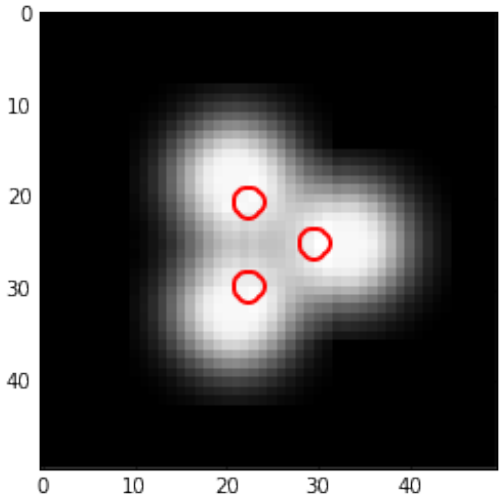


Figure 3.4: *Three artificially produced individual Gaussian-like shaped particles that overlap in intensity are shown. The red circles are the centroid positions in space located by Trackpy and are biased toward the centroid of the cluster.*

Solution For The Bias Because colloids often have a Gaussian-like intensity distribution, solving the close range bias is done by fitting Gaussians functions to the individual particles. Each particle can still be *approximately* located by tp.locate. With the approximate coordinates a Gaussian intensity profile is made over all the available axis (not in the image itself), with the mean at the centroid location of the particle. This is done for all the particles and the overlapping Gaussians are combined into one pattern.

The Gaussian function in this research is defined as

$$f(x) = A \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right) \tag{3.2}$$

Version of July 30, 2015– Created July 30, 2015 - 09:24

Here A is the height of the peak, x is the position along x-axis, μ is the mean and σ the standard deviation [27]. The parameters for each particle's Gaussian intensity profile are taken from the return values of tp.locate as follows

$$\begin{aligned} A &= \frac{mass}{\pi\, size^2} \\ \mu &= cx \\ \sigma &= size \end{aligned} \tag{3.3}$$

Where `mass` is the total integrated brightness, `size` the radius of gyration of its Gaussian-like profile and `cx` the centroid of the feature as determined by tp.locate. Depending on how many axis there are, the Gaussian function in equation (3.2) is multiplied with more Gaussian exponents, only with the exponents being dependent on the other axis that are involved. The total Gaussian intensity profile is then given by the sum over all the individual Gaussian intensity profiles. Fitting the total Gaussian intensity profile to a clusters intensity profile in the image by using the least square method [28], the new refined coordinates are determined.

3.4.3 Tracking Multiple Clusters

If a sample contains a single cluster, the close range bias can be solved. However, if one is to have a sample that contains multiple clusters or spheres, the code is going to make and fit a Gaussian profile for the *whole* image. By making a Gaussian profile per particle and summing the Gaussian profiles over all the particles (as described in the previous subsection), the complete image with all the particles is fit to the total Gaussian profile. The more particles are located, the more calculation time is needed for the fitting. Moreover, spherical particles do not need to be fit, for they are already located in a good manner by Trackpy. To solve this, a code was written to find and identify the clusters. To determine when particles form a cluster and when not, we made use of a function in the KDTree module (see section 2.6). A function from the KDTree lists all the pairs of points within the KDTree. This is determined by looking if another point in the KDTree lies within a distance r, the only required argument for this function. The list of all the pairs is then run through a piece of code to see which pairs are connected to each other. A list is generated containing lists of all the individual particles of the connected pairs. These lists are all the found clusters and their coordinates can be found back in the KDTree

data. For a cluster of N particles, it takes the Cluster Code N+1 steps with a for loop to determine this is cluster of N particles.

To only fit the shape of the found cluster, a code was written to select the part of the image needed. This was done by using another function in Trackpy called `tp.masks.binary_mask`. The newly made function works much like the way the total Gaussian profile is generated from individual Gaussian profiles. The function makes a spherical (3D) or circular (2D) binary mask for an individual particle with the diameter taken by what is passed in the arguments as diameter within. For each particle in a cluster a binary mask was made, masking all the area outside the mask. The resulting total mask is made by taking the sum over all the individual masks. A small rectangular area or volume that has the same dimensions as the total mask is taken and this rectangular area is then masked with the total mask. Then the fit is applied for only this masked part of the image. Now only the areas with clusters are fitted with the Gaussian functions.

3.4.4 The Cluster Code

To summarize; the newly made 'Cluster Code' processes particles in an image described as follows. First it uses Trackpy to approximately locate the centroids of individual particles. Next the Cluster Code determines which particles form a cluster using a function from the k-d tree module to couple the pairs that form up clusters. At this point the number of particles within a cluster and their (approximate) positions is known. To perform a fit on only the particles within one cluster, a for loop runs over all clusters and process the clusters as follows. The particles outside a single cluster are masked by setting the image intensity outside the mask to zero. Next, a Gaussian function profile is made according to the number of particles within the single cluster. Finally the Gaussian function profile is fit with the least squares method. The output of the Cluster Code is a list of all the fitted coordinates in an image. The output can be adjusted to also provide the k-d tree, the number of particles per cluster, the locations per cluster and the centroids of all the clusters.

Chapter 4

Results & Discussion

The characteristics of PMMA colloids attached to a glycerol - CHB-D interface have been studied. In this section the experimental results and the analyzes will be described.

4.1 Experimental

4.1.1 Liquid-Liquid Interface

The glycerol - CHB-D interface could be clearly observed using bright field microscopy. A bright field microscopy image of a typical interface is shown in figure 4.1. While the contrast of the interface is high, the colloids are barely visible. This indicates that the refractive indexes of the PMMA colloids and the CHB-D were almost completely matched.

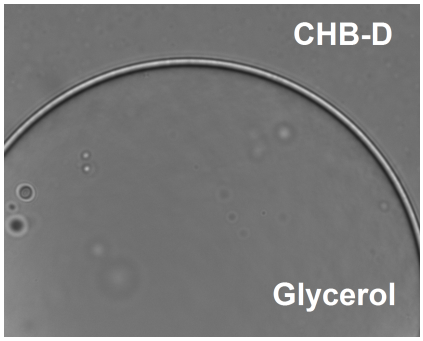


Figure 4.1: *Optical microscopy image of an glycerol - CHB-D interface. The interface can be clearly seen. The colloids (circular shaped features) are much harder to identify with the bright field microscope.*

4.1.2 Colloids In CHB-D

A typical confocal microscopy image of a movie of PMMA colloids in CHB-D (0.44% wt) stock is shown in figure 4.2. Only single particles were observed, indicating that the colloids were well-stabilized over time. Moreover, the interparticle distances in this sample were much larger than the diameter of the colloids and over time the colloids remained in the proximity of their initial position, indicating that the colloids were *charge* stabilized.

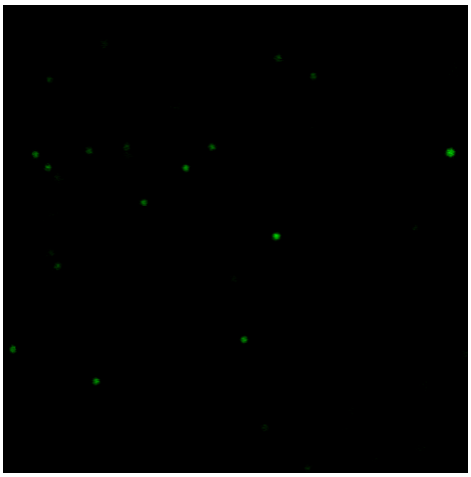


Figure 4.2: *A sample of the PMMA colloids (0.44 % wt) in CHB-D is shown. The interparticle distance is much larger than the diameter of the colloids.*

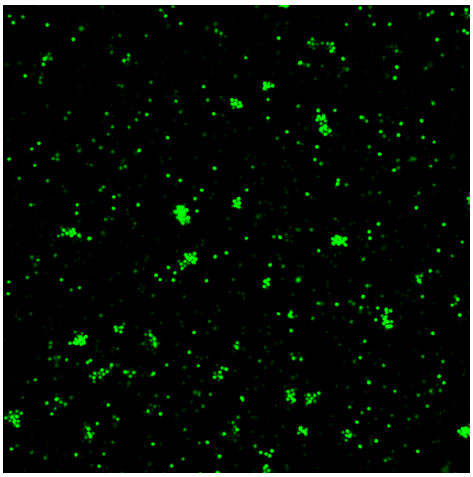


Figure 4.3: *A sample of the PMMA colloids (1.75 % wt) in CHB-D is shown. The interparticle distance is much larger than the diameter of the colloids.*

The PMMA colloids used in this research had the tendency to aggregate. To prevent this, the colloids were put in a oven and a sonic bath as described in the methods. In figure 4.3 an image is shown of the aggregated colloids; here the procedure was not done.

4.1.3 Colloids On Interfaces

In this section results of PMMA colloids attached to glycerol - CHB-D interfaces will be discussed. Two different situations have been studied. An illustration of the two situations is shown in figure 4.4. In situation 1 (figure 4.4a) glycerol droplets are covered with a suspension of colloids in CHB-D. In situation 2 (figure 4.4b) droplets of the suspension are covered

with glycerol. Situation 1 will be discussed first, followed by situation 2.

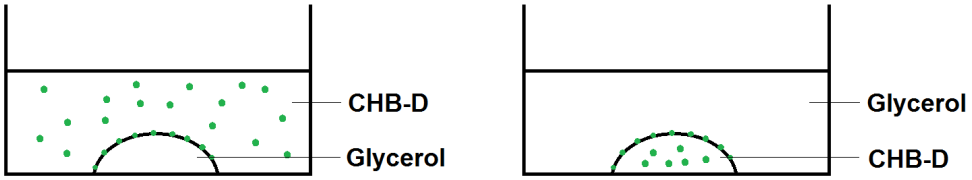


Figure 4.4: *An illustration of the two glycerol - CHB-D interface situations. In situation 1 a glycerol droplet is placed on the glass coverslip and is covered by CHB-D. In situation 2, the two liquids are reversed.*

Situation 1a; Glycerol droplets in CHB-D
In situation 1 it was observed that colloids attached to the glycerol - CHB-D interface. Typical confocal images of situation 1 are shown in figure 4.5. It can be seen that size and the shape of the interface differed throughout the sample. While spherical (figure 4.5a) and ellipsoidal (figure 4.5b) interfaces were observed as well, mainly domes were found (figure 4.5c, d). In real time it was observed that most colloids at the interface were mobile, although some seemed to be stuck as well. By blocking the line of sight the colloids in the CHB-D bulk prevented to see the colloids on the interface clearly, especially for high concentrations of colloids. In the CHB-D bulk outside of the droplets there was a drift in most samples. This drift was caused by evaporation of CHB-D and was partly prevented by the second glass coverslip that is placed on top of the sample holder 3.2.2

The ordering of the colloids on the interface was not always the same throughout the samples and within a single sample. In figure 4.6a the side of a large dome can be seen. Here, the interparticle distance and the ordering of the colloids on the interface is relatively high. On some interfaces there were closely packed patches on the interface (figure 4.6b), where the interparticle distance was smaller than its surroundings. This might be caused by the fact that the colloids were not completely dispersed in the CHB-D before the interface was made. Finally, figure 4.6c shows a ellipsoidal dome where the colloids have a relatively high separation distance. However, the interparticle distances vary between the colloids on the interface. On some interfaces a low quantity of colloids got 'stuck' on the interface and showed no Brownian motion. Here, it might be possible that a mixture of stuck and mobile colloids caused the irregular separation dis-

tances.

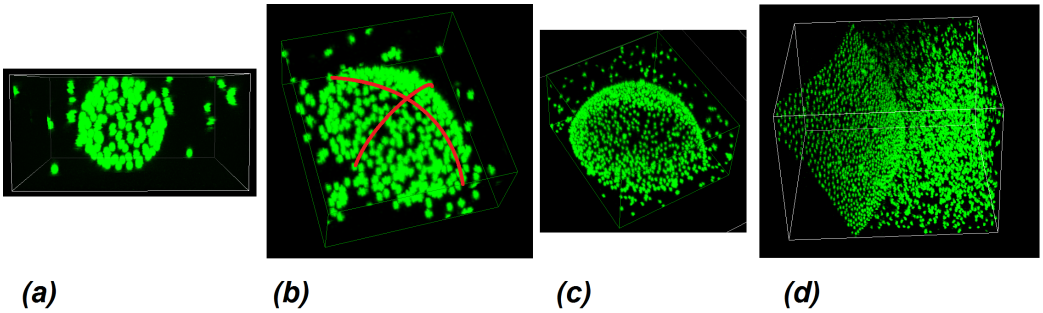


Figure 4.5: *Confocal images of interfaces of a sphere, an ellipse shaped dome, a dome and the side of a very large dome are shown in figure 4.5a-d respectively. The colloids in the CHB-D bulk can be seen, sometimes preventing a clear image of the interface (figure 4.5d). The red lines in figure 4.5b are guides to the eye. The concentration of colloids was 0.44% wt in figure 4.5a-c and 7.5% wt in figure 4.5d*

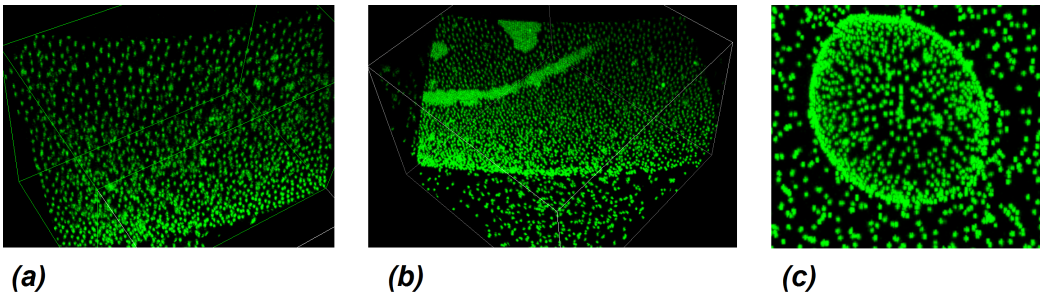


Figure 4.6: *A side view of interfaces of two very large domes with colloids (0.44% wt) attached is shown in figure 4.8a, b. They show relatively high colloidal ordering in the crystal lattice. The dome in figure 4.8b has closely packed colloidal patches on the interfaces. In figure 4.8c a dome with different interparticle distances on the interface is shown (0.44% colloids).*

Time Dependence

The colloids were functionalized with a green fluorescent core and a non-fluorescent shell. In order to check if the dye (NBD) did not bleach, two movies of 10 minutes were made. In figure 4.7 a spherical interface with colloids (0.44% wt) is shown in time. Figure 4.7a shows the colloids at the beginning of the first movie and figure 4.7b at the end of the first movie. Figure 4.7c shows the end of the second movie. The end of the first movie and the start of the second movie were separated by 53 minutes. The images in figure 4.7 show that the intensity of the colloids only decreased significantly when imaged for a long time, therefore bleaching effects did not have to be taken into account. In figure 4.7 it can also be seen that over long periods of time, the ordering of the colloids on the interface does not improve qualitatively and that the structure remains the same. It is also shown that the colloids are mobile, this could be seen within the order of seconds.

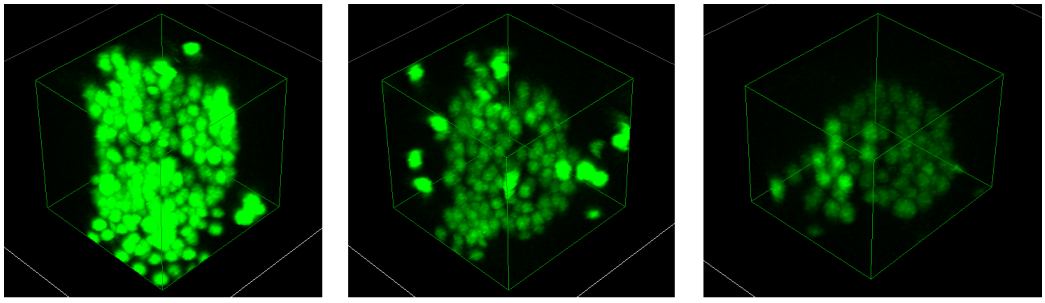


Figure 4.7: *Confocal images of a spherical interface with colloids (0.44 % wt) attached is shown over time. Figure 4.7a and b is at the start and end of a 10 minute movie respectively. Figure 4.7c is a confocal image at the end of a second movie that started 53 minutes after the initial movie. The structure of the spherical droplet remains the same over a time of 73 minutes. Over time the intensity in the images drop.*

Concentration Dependence

In droplets of glycerol in CHB-D, most interfaces showed good interparticle separation distances between the colloids but low ordering. In figure 4.8 and 4.9 examples are shown. In figure 4.8 a large dome is shown, where a high concentration of colloids has been used (7.5% wt). In figure 4.9 a ellipsoidal dome is shown, where the concentration was much lower (0.44% wt). The hexagonal ordering as shown in both figures, was approximately the same. However, a difference in interparticle distance of colloids on the

interface can be seen. This is probably caused by the high respectively low concentrations in the bulk.

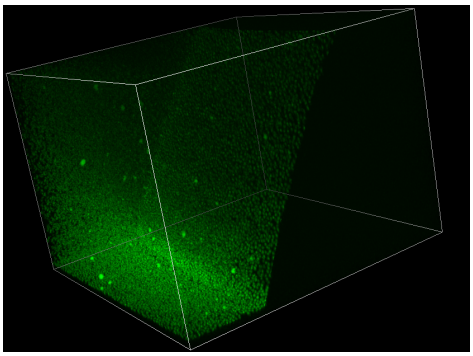


Figure 4.8: *A large dome shaped interface. Colloids (7.5% wt) are attached to the interface.*

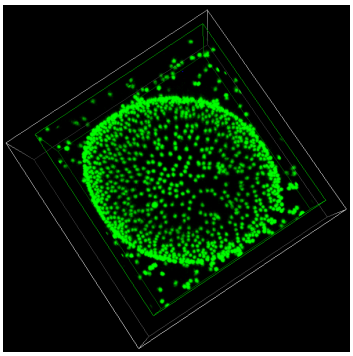
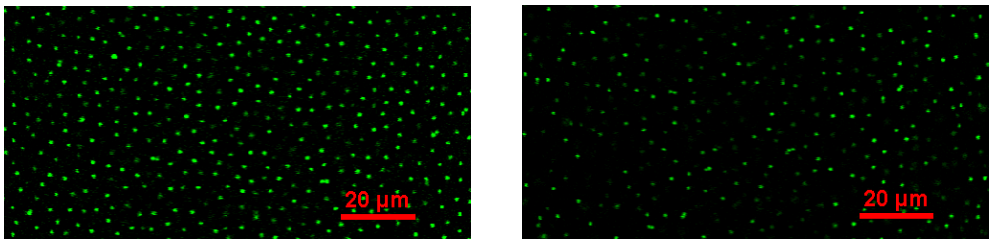


Figure 4.9: *A ellipse shaped interface. Colloids (0.44 % wt) are attached to the interface.*

Situation 1b; Glycerol droplets in filtered CHB-D

To increase the Debye screening length and therefore the interparticle distance of the colloids at the interface, filtered CHB was used. Microscopy images of PMMA colloids in filtered CHB-D (mixture directly made after CHB filtering) and unfiltered CHB-D are shown in figure 4.10. It seems that the interparticle distance is larger when the CHB is filtered prior to making the CHB-decalin mixture than when this is not done, although this can be speculated. A qualitative comparison of the Debye length of the colloids in filtered and unfiltered CHB was unfortunately impossible since no conductivity meter was available to measure such low currents.



(a) *Before filtering* ***(b)*** *After filtering*

Figure 4.10: *Two confocal images of colloids (1.75 % wt) in CHB-D before (left) and after (right) filtering the CHB. The interparticle distance is higher after filtering the CHB.*

Situation 2; CHB-decalin droplets in glycerol
When the two liquids were reversed, meaning CHB-D droplets were covered with glycerol, the PMMA colloids still attached to the interface and mobility of the colloids on the interface was observed. The shapes formed in situation 2 were comparable to situation 1. An example of the attachment of colloids to the interface is shown by a confocal image in figure 4.11. Here, a dome of CHB-D (27.2 % wt decalin) with colloids (7.5% wt) is covered with glycerol. A highly ordered crystalline structure of colloids can be observed. The colloids are hexagonally packed and defects can be recognized.

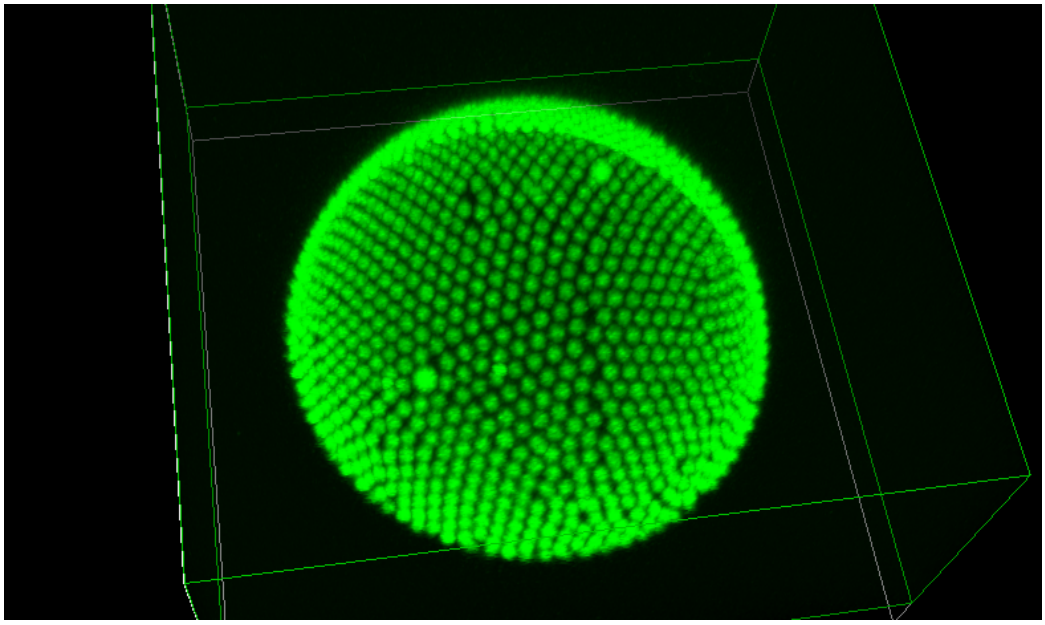


Figure 4.11: *A confocal image of a dome shaped CHB-D - glycerol interface. The colloids (7.5% wt) are closely packed, form a hexagonal lattice on the interface and show several defects.*

In the same sample other ordering of the colloids on the interface was seen. In figure 4.12 another dome is shown from the top, only with a less dense packing of colloids on the interface. It was probably by coincidence that this droplet contained less colloids when put on the glass. Although the interparticle distance observed in figure 4.12 is larger compared to 4.11, the colloids still order on a hexagonal-like lattice. This indicates that the repulsive forces are long ranged.

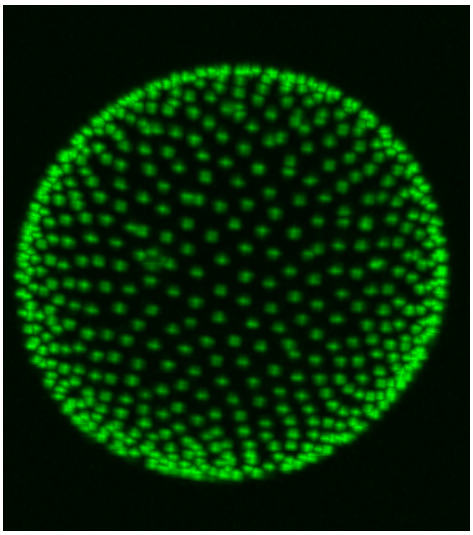


Figure 4.12: *A confocal image of a dome shaped CHB-D - glycerol interface. The colloids (7.5% wt) have a higher separation distance than figure 4.11, form a hexagonal lattice on the interface and show several defects.*

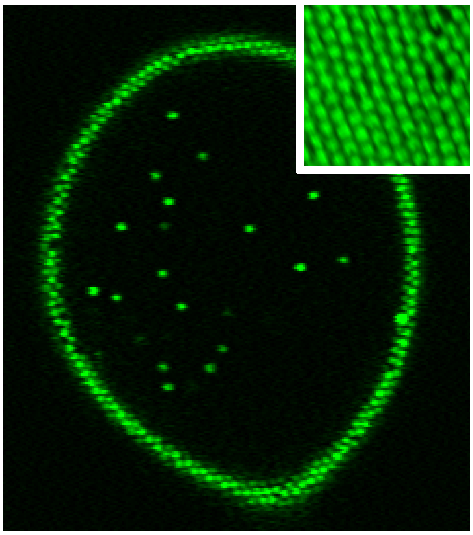


Figure 4.13: *A confocal image of dome shaped CHB-D - glycerol interface. The colloids (7.5% wt) are closely packed on the interface (inset) and form a hexagonal lattice on the interface. In the bulk of the dome colloids can be seen.*

Some droplets of CHB-D contained so many colloids that not the complete surface could be filled with colloids and colloids thus remained in the bulk of the CHB-D. An example of this is shown in figure 4.13, were a cross section of a droplet is shown. Here it is suggested that comparing figure 4.11, 4.12 and 4.13 shows that the interface is first completely filled with colloids before the colloids remaining in the bulk. By first filling up the complete interface, the interfacial energy is brought to a minimum. This agrees with the theory (section 2.2).

In general, the domes and droplets made with CHB-D in glycerol produced good crystal lattices. However, this setup has some disadvantages as well. First of all, the majority of the CHB-D droplets were smeared out instead of producing nice domes or spherical droplets. An example of this is shown in figure 4.14. An explanation for this phenomenon might be the fact that the contact angle of CHB-D with the substrate is different from that of glycerol. Secondly, the colloids were much less mobile on the interface when droplets of CHB-D were taken than when droplets of glycerol were taken. For CHB-D droplets Brownian motion was only vis-

ible in the order of minutes, where for glycerol droplets this would take only seconds (determined bye eye). This is probably due to the fact that glycerol is a highly viscous material (table 3.3) that was now placed on the outside. Finally, the CHB-D in glycerol droplets sometimes contained colloids in the bulk that might cause other ordering on the surface when in (electrostatic) equilibrium with the colloids in the bulk.

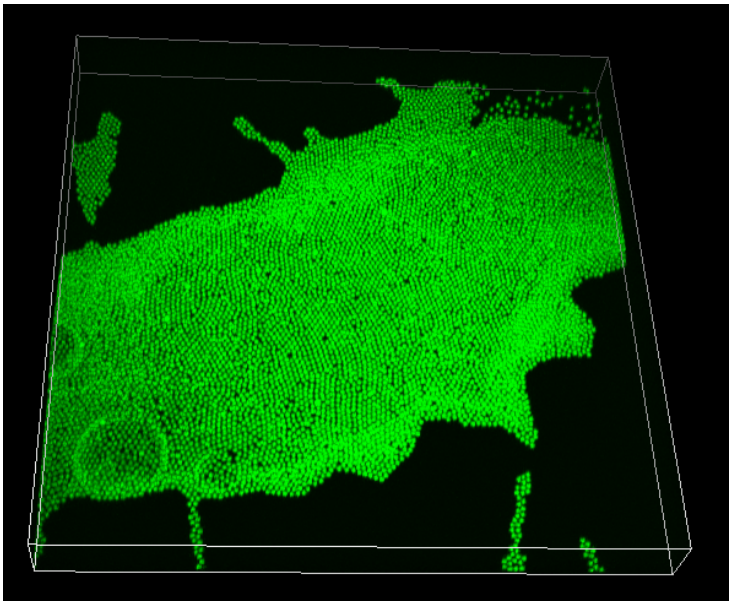
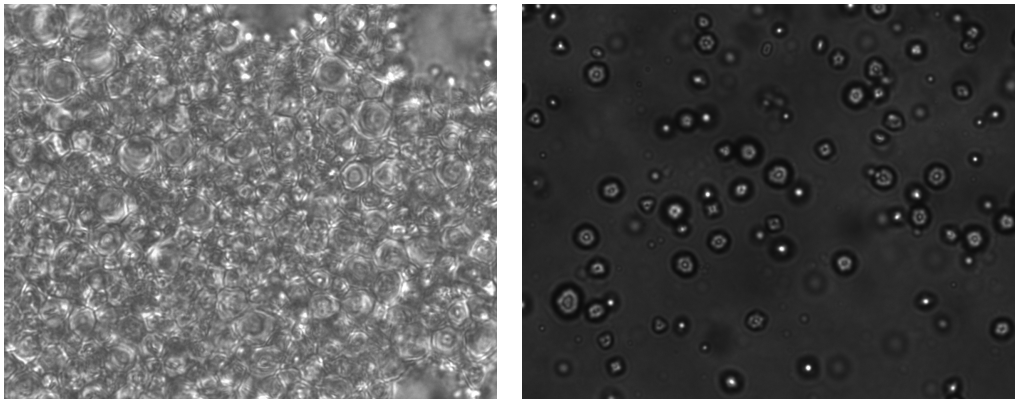


Figure 4.14: *A confocal image of colloids (7.5 % wt) on a smeared out CHB-D - glycerol interface.*

4.2 Colloidal Clusters

Colloidal clusters of spherical PMMA colloids were made according to the method described in section 3.2.4. To obtain colloidal clusters of compact shapes in water, all toluene had to be evaporated. Therefore, optical microscopy images were taken during the evaporation process. In figure 4.15a a bright field image after 1 hour of heating at 60 °C is shown. Large droplets of toluene with colloids could be recognized. In figure 4.15b a bright field image of the clusters in toluene is shown after heating an extra hour at 95 °C. The shape of the clusters can be recognized and no toluene-water interface is observed. The individual colloids within a cluster could be even better seen with the confocal microscope as shown in figure 4.16. Most clusters consisted of 6 to 10 colloids.



(a) ***(b)***

Figure 4.15: *Two optical microscopy images of colloids in toluene in water are shown. In the image on the left the toluene can clearly be seen. In the image on the right, the toluene has evaporated and individual colloids can be seen within the clusters.*

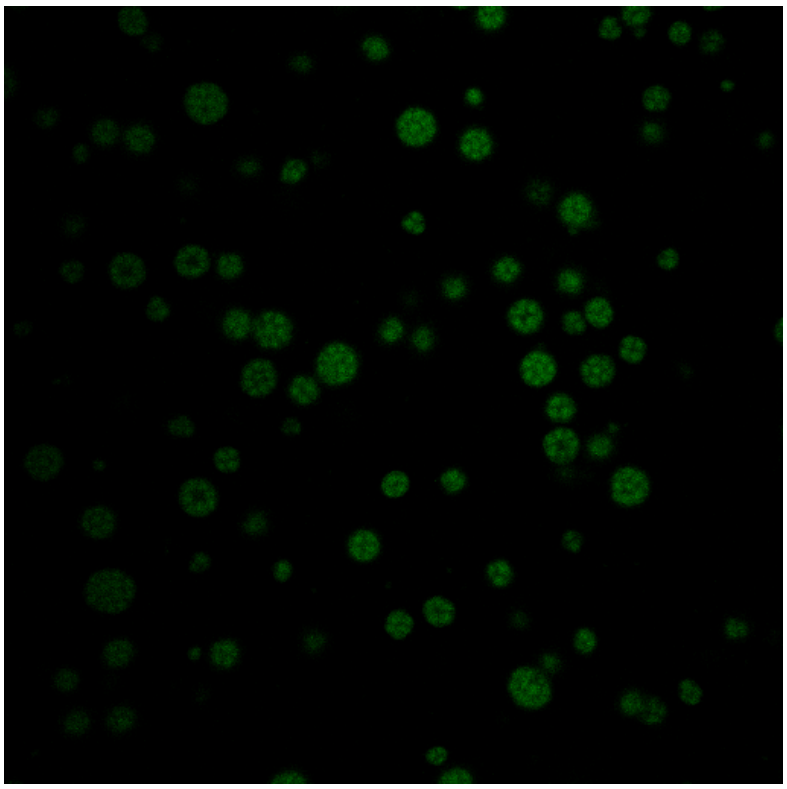


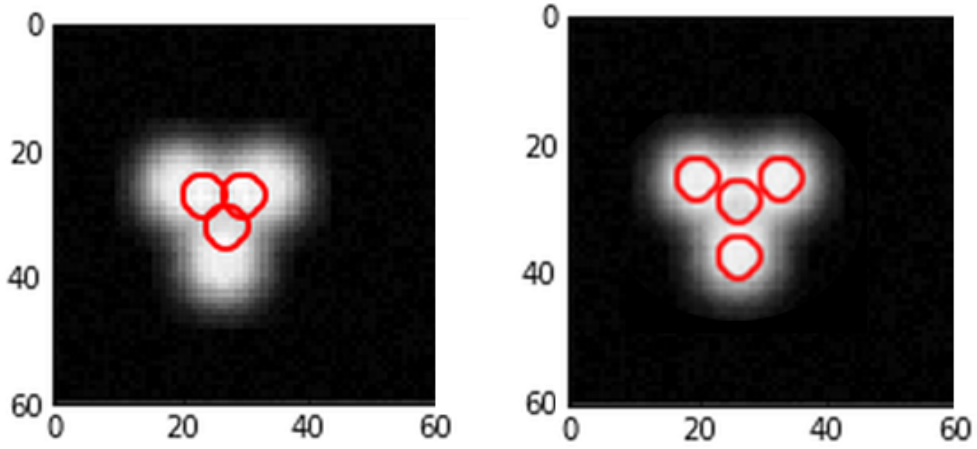
Figure 4.16: *A confocal microscopy image of clusters. Within a single cluster, individual colloids can be seen.*

4.3 Python Analysis Results

A structure analysis of artificial and experimental data of colloidal clusters has been performed using Python. This section is split up into an analysis of the artificial and experimental data.

4.3.1 Artificial Data

To test and debug the code, artificial clusters consisting of particles with a diameter of 20 pixels and with known positions were created by making use of the Trackpy module. A typical analysis of an artificial cluster consisting of three spheres is shown in figure 4.17. Figure 4.17a shows the cluster and the biased tracking when Trackpy was used. The red circles show the positions of the particles found by Trackpy. In figure 4.17b is shown how the Cluster Code solved the bias of Trackpy. The red circles show the particles positions found by the Cluster Code. The red circle in the center however, is not a tracked particle but the centre of mass of the cluster.



(a) ***(b)***

Figure 4.17: *Artificially produced particles with a diameter of 20 pixels overlap in intensity. In the image on the left the centroids located by Trackpy are shown as red circles that are biased towards the centroid of the cluster. In the image on the right the centroids as located by the Cluster Code are shown as red circles. The red circle in the middle is the centroid of the particles located centroids.*

The Cluster Code can identify individual clusters within a sample. In figure 4.18 the result of an analysis of two clusters is shown. It can be seen that the code recognizes which particles belong to the same cluster and distinguishes between the different clusters. The blue and green circles represent how large the value is to pass as argument to the function of the k-d tree to find pairs. If this parameter is large enough, the particle pairs are all coupled together by the Cluster Code. This is plotted in figure 4.18b. If this value is small however, clusters of two and three particles were found. This is plotted in figure 4.18c.

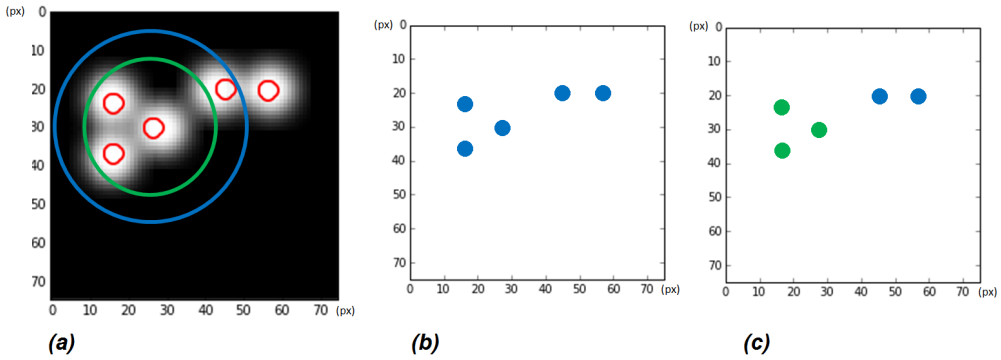
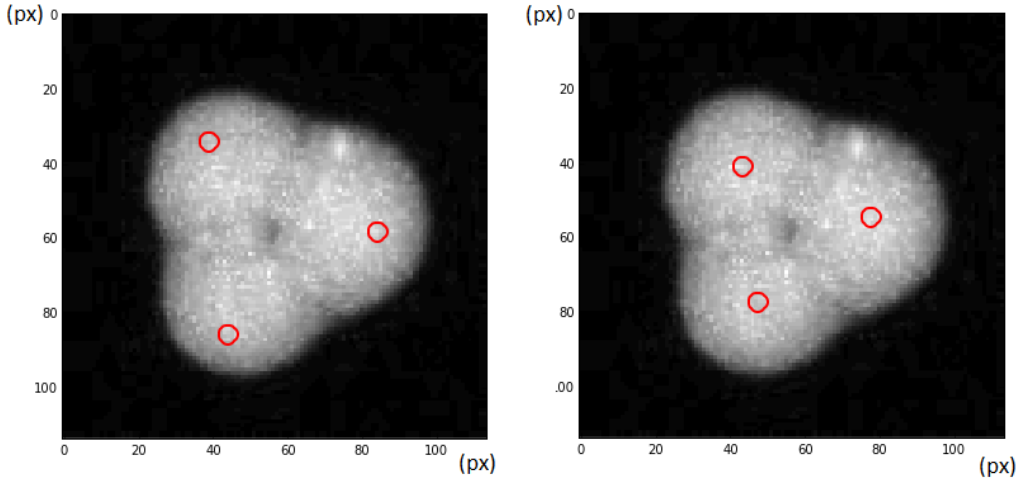


Figure 4.18: *Figure 4.18a shows five artificially produced particles with a diameter of 20 pixels and overlapping intensities. The red circles show the positions Trackpy has located. The blue and green circles represent how large the value is to pass as argument to the function of the k-d tree to find pairs. In figure 4.18b the value is large enough to identify all the particles as one single cluster. In figure 4.18c the value is smaller and two clusters are identified. After the Cluster Code has identified the clusters, the clusters are fitted with Gaussians functions by the Cluster Code; this is not shown here.*

4.3.2 Experimental Data

A confocal image of a colloidal cluster of micron-sized PMMA shperes, provided by V. Meester, MSc, is shown in figure picture 4.19. The colloids do not have a core-shell and are marked with the fluorescent dye NBD. They are micronsized and stabilized with PHSA. Figure 4.19a shows the colloidal cluster before analysis and figure 4.19b shows the colloidal cluster after the analysis with the Cluster Code. In 4.19a the centroids found by Trackpy have a bias towards the outside of the cluster. By letting the new code fit Gaussian functions, the bias is solved. In this image there was only one cluster.



(a) ***(b)***

Figure 4.19: *A confocal image of a PMMA colloidal cluster is shown twice. The centroids (indicated as red circles) of the colloidal cluster located by Trackpy and the Cluster Code are shown in figure 4.19a and 4.19b respectively. The centroids located by Trackpy are biased towards the outside of the cluster.*

Chapter 5

Conclusion

An experimental setup has been developed to successfully attach PMMA colloids to a glycerol - CHB-decalin interface. However, the results show that the system is not completely viable yet. Some improvements are required before it is a suitable setup. In this research two situations have been studied. In situation 1 the interface was made by putting glycerol droplets on a glass coverslip and covering the glycerol with CHB-D, in situation 2 the two liquids were reversed.

In this study interfaces with different shapes and sizes were produced where the PMMA colloids could attach to. The PMMA colloids on the glycerol in CHB-D droplets (situation 1) interfaces were mobile. They showed hexagonal-like ordering on some of the interfaces. However, the amount of ordering was not the same throughout the samples. The interparticle distances differed throughout the samples and sometimes it differed on a single interface. As a product of the decomposition of CHB, HBr in turn could also decompose into ions by metal compounds. The ions increased the Debye length and therefore the interparticle distance in the samples. The interparticle distance was partly determined by the concentration of PMMA in the bulk as well.

Taking CHB-D in glycerol droplets (situation 2), provided in most cases for ordering in the colloidal structure. However, the colloids were less mobile than colloids on a glycerol in CHB-D interface and most droplets were smeared out over the glass. Droplets could also contain colloids in the bulk that might interact with colloids on the interface. An advantage of this containment was that the colloids on the interface became much more visible than glycerol in CHB-D droplets. A solution for making the

domes more visible with glycerol in CHB-D droplets is also presented in the literature, by way of removal of the excess colloids on the outside of the structure [6] [7].

Situation 2 is not preferred for making interfaces, because situation 1 has advantages over situation 2. Because situation 1 does not show high hexagonal ordering and because the curvature and shapes of the interfaces are not completely controlled, the setup is promising but not yet suitable.

To study anisotropic colloids on interfaces a method for the creation of colloidal clusters has been tested. The clusters fabricated as described in section 3.2.4 were well defined with the toluene completely evaporated and contained quite a large number of colloids per cluster. To separate the clusters, density gradient centrifugation can be employed.

Finally, in order to track anisotropic particles on liquid-liquid interface a new cluster code was written. The analysis done with the new Cluster Code solves the bias that is created by the Python module Trackpy. It can identify clusters, store its individual coordinates and works in all three dimensions of space. However, an error in finding clusters did present itself when analyzing a sample. Masking the clusters produced a bug caused by a rounding error. This part of the code needs to be debugged. So far it is not possible to track the clusters in time. By using pieces of code from Trackpy it is possible to rewrite this and to track clusters in time. To analyze ordering of anisotropic colloids on interfaces, including the calculation of order parameters into the code is recommended.

Version of July 30, 2015– Created July 30, 2015 - 09:24

Chapter 6

Outlook

Density matching was achieved by using literature values, but the system was not completely density matched. With a day or more the PMMA formed sedimentation on the bottom of sample. To make a system completely density matched, the density values of CHB, decalin and PMMA should be measured. If all the values correspond to the literature values, the literature value can be (exactly) taken. If the values are off, the CHB-D should be density matched by 'hand' until the density of the CHB-D of the PMMA is matched. This can be achieved by using centrifugation and checking regularly if the colloids form sedimentation. If the colloids do not sediment over a long period of time by centrifugation, the colloids are properly density matched.

The structures made varied throughout the study and depended on how the droplets were placed on to the glass coverslip. Because the placement did not provide the same structures for every sample, it is recommended to use structures that are made in a controlled and reproducible way. This can for example be done with capillary channels [7] or with soft photolitography.

To filter out ions in the system, the CHB is filtered through activated alumina. The difference in conductivity between post- and pre-filtered CHB should be measured using a conductivity meter if one is to conduct new experiments with CHB.

References

[1] Chris Buskes. *Evolutionair denken: De invloed van Darwin op ons wereldbeeld*. Uitgeverij Nieuwezijds, Amsterdam, 2006. ISBN 9789057121807.

[2] James Trefil and Robert M. Hazen. *Sciences: An Integrated Approach*. John Wiley & Sons (Asia) Pte Ltd, 111 River Street, Hoboken, NJ, 2010. ISBN 9780470505816.

[3] Roya Zandi, David Reguera, Robijn F. Bruinsma, William M. Gelbart, and Joseph Rudnick. Origin of icosahedral symmetry in viruses. *Proceedings of the National Academy of Sciences of the United States of America*, 101(44):15556–15560, 2004. doi: 10.1073/pnas.0405844101. URL http://www.pnas.org/content/101/44/15556.abstract.

[4] Stefano Sacanna and David J. Pine. Shape-anisotropic colloids: Building blocks for complex assemblies. *Current Opinion in Colloid & Interface Science*, 16(2):96 – 105, 2011. ISSN 1359-0294. doi: http://dx.doi.org/10.1016/j.cocis.2011.01.003. URL http://www.sciencedirect.com/science/article/pii/S1359029411000069.

[5] Sharon C. Glotzer and Michael J. Solomon. Anisotropy of building blocks and their assembly into complex structures. *Nat Mater*, 6(7): 557–562, Aug 2007. ISSN 1476-1122. doi: 10.1038/nmat1949. URL http://dx.doi.org/10.1038/nmat1949.

[6] Dmitry Ershov, Joris Sprakel, Jeroen Appel, Martien A. Cohen Stuart, and Jasper van der Gucht. Capillarity-induced ordering of spherical colloids on an interface with anisotropic curvature. *Proceedings of the National Academy of Sciences*, 110(23):9220–9224, 2013. doi: 10.1073/pnas.1222196110. URL http://www.pnas.org/content/110/23/9220.abstract.

[7] Vincenzo Vitelli William T. M. Irvine and Paul M. Chaikin. Pleats in crystals on curved surfaces. *Nature*, 468:974–951, Dec 2010. doi: 10.1038/nature09620. URL http://dx.doi.org/10.1038/nature09620.

[8] Dan Allan, Thomas A Caswell, Nathan Keim, François Boulogne, Rebecca W Perry, and Leonardo Uieda. trackpy: Trackpy v0.2.4, October 2014. URL http://dx.doi.org/10.5281/zenodo.12255.

[9] P.C. Hiemenz and R. Rajagopalan. *Principles of Colloid and Surface Chemistry, Third Edition, Revised and Expanded*. Undergraduate Chemistry: A Series of Textbooks. Taylor & Francis, 1997. ISBN 9780824793975. URL https://books.google.nl/books?id=CBvrS8rfPlYC.

[10] Surface tension — Wikipedia, the free encyclopedia, . URL https://en.wikipedia.org/wiki/Surface_tension. Online; accessed 20-July-2015.

[11] Surface tension — Wikipedia, the free encyclopedia, . URL https://en.wikipedia.org/wiki/Contact_angle. Online; accessed 20-July-2015.

[12] P. G. de Gennes. Wetting: statics and dynamics. *Rev. Mod. Phys.*, 57:827–863, Jul 1985. doi: 10.1103/RevModPhys.57.827. URL http://link.aps.org/doi/10.1103/RevModPhys.57.827.

[13] Todd M. Squires. Drops on soft surfaces learn the hard way. *Proceedings of the National Academy of Sciences*, 110(31):12505–12506, 2013. doi: 10.1073/pnas.1310672110. URL http://www.pnas.org/content/110/31/12505.short.

[14] Ming F. Hsu, Michael G. Nikolaides, Anthony D. Dinsmore, Andreas R. Bausch, Vernita D. Gordon, Xi Chen, John W. Hutchinson, David A. Weitz, and Manuel Marquez. Self-assembled shells composed of colloidal particles fabrication and characterization. *Langmuir*, 21(7):2963–2970, 2005. doi: 10.1021/la0472394. URL http://dx.doi.org/10.1021/la0472394. PMID: 15779972.

[15] A. D. Dinsmore, Ming F. Hsu, M. G. Nikolaides, Manuel Marquez, A. R. Bausch, and D. A. Weitz. Colloidosomes: Selectively permeable capsules composed of colloidal particles. *Science*, 298(5595):1006–1009, 2002. doi: 10.1126/science.1074868. URL http://www.sciencemag.org/content/298/5595/1006.abstract.

[16] Confocal microscopy — Wikipedia, the free encyclopedia, . URL https://en.wikipedia.org/wiki/Confocal_microscopy. Online; accessed 20-July-2015.

[17] John C. Crocker and David G. Grier. Methods of digital video microscopy for colloidal studies. *Journal of Colloid and Interface Science*, 179(1):298 – 310, 1996. ISSN 0021-9797. doi: http://dx.doi.org/10.1006/jcis.1996.0217. URL http://www.sciencedirect.com/science/article/pii/S0021979796902179.

[18] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, September 1975. ISSN 0001-0782. doi: 10.1145/361002.361007. URL http://doi.acm.org/10.1145/361002.361007.

[19] Mark T. Elsesser and Andrew D. Hollingsworth. Revisiting the synthesis of a well-known comb-graft copolymer stabilizer and its application to the dispersion polymerization of poly(methyl methacrylate) in organic media. *Langmuir*, 26(23):17989–17996, 2010. doi: 10.1021/la1034917. URL http://dx.doi.org/10.1021/la1034917. PMID: 21053983.

[20] L. Antl, J.W. Goodwin, R.D. Hill, R.H. Ottewill, S.M. Owens, S. Papworth, and J.A. Waters. The preparation of poly(methyl methacrylate) latices in non-aqueous media. *Colloids and Surfaces*, 17(1):67–78, 1986. ISSN 0166-6622. doi: http://dx.doi.org/10.1016/0166-6622(86)80187-1. URL http://www.sciencedirect.com/science/article/pii/0166662286801871.

[21] M.E. Leunissen. *Manipulating colloids with charges and electric fields*. PhD thesis, Utrecht University, 2007.

[22] Mirjam E. Leunissen, Alfons van Blaaderen, Andrew D. Hollingsworth, Matthew T. Sullivan, and Paul M. Chaikin. Electrostatics at the oil–water interface, stability, and order in emulsions and colloids. *Proceedings of the National Academy of Sciences*, 104(8):2585–2590, 2007. doi: 10.1073/pnas.0610589104. URL http://www.pnas.org/content/104/8/2585.abstract.

[23] Mirjam E. Leunissen, Jos Zwanikken, Rene van Roij, Paul M. Chaikin, and Alfons van Blaaderen. Ion partitioning at the oil-water interface as a source of tunable electrostatic effects in emulsions with colloids. *Phys. Chem. Chem. Phys.*, 9:6405–6414, 2007. doi: 10.1039/B711300E. URL http://dx.doi.org/10.1039/B711300E.

[24] Glycerol — Wikipedia, the free encyclopedia, . URL https://en.wikipedia.org/wiki/Glycerol. Online; accessed 20-July-2015.

[25] K. Muthamizhi, P. Kalaichelvi, Shubhangi Tukaram Powar, and R. Jaishree. Investigation and modelling of surface tension of power-

law fluids. *RSC Adv.*, 4:9772, 2014. doi: 10.1039/C3RA46555A. URL http://dx.doi.org/10.1039/C3RA46555A. Figure 1 is adjusted to describe own (filtering) setup.

[26] Vinothan N. Manoharan, Mark T. Elsesser, and David J. Pine. Dense packing and symmetry in small clusters of microspheres. *Science*, 301 (5632):483–487, 2003. doi: 10.1126/science.1086189. URL http://www.sciencemag.org/content/301/5632/483.abstract.

[27] Hongwei Guo. A simple algorithm for fitting a gaussian function. *Streamlining Digital Signal Processing: A Tricks of the Trade Guidebook, Second Edition*, pages 297–305, 2012.

[28] A. Charnes, E. L. Frome, and P. L. Yu. The equivalence of generalized least squares and maximum likelihood estimates in the exponential family. *Journal of the American Statistical Association*, 71(353):pp. 169–171, 1976. ISSN 01621459. URL http://www.jstor.org/stable/2285762.

Appendices

Appendix A

The 'Cluster Code'

```
from __future__ import division
import numpy as np
import pandas as pd
import trackpy as tp
import pims
from scipy.optimize import leastsq
from scipy.spatial import cKDTree

from trackpy.utils import validate_tuple


def find_kdtree(features, ndim, maxDist=20):
    """ This function returns all kdtree related
        ↪ values

    Parameters
    ----------
    features : all the features found by trackpy.
        ↪ locate
    maxDist  : max separation distance from feature to
        ↪  feature.
                This variable will define two features
                    ↪ to be pairs,
                if the separation distance will be <
                    ↪ than the maxDist
                (within the KDTree)
```

```
    Returns
    ---------
    kdt, kdtPairsSort

    """
    if ndim == 3:
        coords = np.array(features[['z', 'y', 'x']])
    if ndim == 2:
        coords = np.array(features[['y', 'x']])
    if ndim == 1:
        coords = np.array(features[['x']])

    # kdt setup
    kdt = cKDTree(coords)
    kdtPairsSet = kdt.query_pairs(maxDist)
    kdtPairsSort = np.array(sorted(list(kdtPairsSet)),
    ↪   dtype=np.uint16)

    return kdt, kdtPairsSort


def find_clusters(features, ndim, maxDist=20):
    """ This function finds all clusters

    If there are N particles in a clusters, separating
    ↪   takes N+1 steps
    (if I'm correct) for a cluster

    Parameters
    ---------
    features : all the features found by trackpy.
    ↪   locate
    maxDist  : max separation distance from feature to
    ↪   feature.
               This variable will define two features
               ↪ to be pairs,
               if the separation distance will be <
               ↪ than the maxDist
               (within the KDTree)

    Returns
```

```
clusterListFinal, kdt

"""

# kdt setup
kdt, kdtPairsSort = find_kdtree(features, ndim,
    ↪ maxDist)

# Put all the clusters in this list
clusterListFinal = []

# This code runs until all clusters are found
while (kdtPairsSort.size > 0):

    # Setup constants and arrays
    tempArray = np.empty([0], dtype=int)
    # pairs that are being used for finding the
        ↪ cluster
    pairArray = np.empty([0, 2], dtype=int)
    # all the particles that belong to this
        ↪ current cluster
    clusterArray = np.empty([0], dtype=int)

    # First check, must be done before the start
        ↪ of the while loop
    # previous minimum within kdtPairsSort
    # (EQUALS kdtPairsSort[0, 0] ??)
    previousMinimum = np.amin(kdtPairsSort)
    # find all indices where equal to prev. min.
    whereEq = np.where(kdtPairsSort ==
        ↪ previousMinimum)
    # stack these actual values
    pairArray = np.vstack([pairArray, kdtPairsSort
        ↪ [whereEq[0]]])
    # append all the used indices
    tempArray = np.append(tempArray, whereEq[0])

    # minimum within kdtPairsSort, the extra one
        ↪ is for masking
    minimum = np.amin(pairArray) + 1
```

```
# masked pairArray for filtering what is
    ↪ already used
mPairArray = np.ma.masked_less(pairArray,
    ↪ minimum, copy=False)
# mPairArray = pairArray[pairArray >= minimum]

# All subsequent checks
while (previousMinimum != minimum):
    whereEq = np.where(kdtPairsSort == minimum
        ↪ - 1)
    pairArray = np.vstack([pairArray,
        ↪ kdtPairsSort[whereEq[0]]])
    previousMinimum = minimum

    mPairArray = np.ma.masked_less(pairArray,
        ↪ minimum, copy=False)
    clusterArray = np.append(clusterArray,
        ↪ minimum - 1)
    minimum = np.amin(mPairArray) + 1
    tempArray = np.append(tempArray, whereEq
        ↪ [0])

    # It is possible to skip, this prevents
        ↪ skipping
    if minimum - previousMinimum > 1:
        whereEq = np.where(kdtPairsSort ==
            ↪ minimum - 1)
        pairArray = np.vstack([pairArray,
            ↪ kdtPairsSort[whereEq[0]]])
        for i in range(1, minimum -
            ↪ previousMinimum):
            if (i + 1) in pairArray:
                clusterArray = np.append(
                    ↪ clusterArray, i + 1)

clusterListFinal = clusterListFinal + [
    ↪ clusterArray]
tempArray = np.unique(tempArray)
kdtPairsSort = np.delete(kdtPairsSort,
    ↪ tempArray, axis=0)
```

```
    return clusterListFinal, kdt


def gauss(mu, sigma, x):
        """
        (not normalized) gaussian, for fitting
           ↪ purposes.
        A * sigma * sqrt(2 pi) gives area
        """
        return np.exp(-(x-mu)**2/(2.*sigma**2))


def generate_residual(N, ndim):
    """ This function generates the Gaussian profile

    Parameters
    ----------
    N                : Number of particles within one
       ↪ cluster
    ndim             : Number of image dimensions


    Returns
    ----------
    residual         : the Gaussian profile

    """
    string = ""
    string2 = ""
    string3 = ""
    for n in range(1, N+1):
        if n != 1:
            string += ", "
            string2 += " + "
        string += "A" + str(n)
        string2 += "A" + str(n)
        if ndim == 3:
            string += ", cz" + str(n)
            string2 += "*gauss(cz" + str(n) + ", s, z)
               ↪ "
        if ndim >= 2:
```

```
            string += ", cy" + str(n)
            string2 += "*gauss(cy" + str(n) + ", s, y)
                ↪ "
        if ndim >= 1:
            string += ", cx" + str(n)
            string2 += "*gauss(cx" + str(n) + ", s, x)
                ↪ "
        if n == N:
            string += ", s = params"
    if ndim == 3:
        string3 = "z, y, x"
    elif ndim == 2:
        string3 = "y, x"
    elif ndim == 1:
        string3 = "x"
    else:
        # RAISE AN ERROR!
        pass

    exec("""def residual(params, im, """ + string3 +
        ↪ """):
        """ + string + """
        return (""" + string2 + """-im)""")
    return residual


def generate_residual_anisotropic(N, ndim):
    """ This function generates the anisotropic
        ↪ Gaussian profile

    Parameters
    ----------
    N               : Number of particles within one
        ↪ cluster
    ndim            : Number of image dimensions


    Returns
    ----------
    residual        : the anisotropic Gaussian
        ↪ profile
```

```
    """
    string = ""
    string2 = ""
    string3 = ""
    for n in range(1, N+1):
        if n != 1:
            string += ", "
            string2 += " + "
        string += "A" + str(n)
        string2 += "A" + str(n)
        if ndim == 3:
            string += ", cz" + str(n)
            string2 += "*gauss(cz" + str(n) + ", sz, z
               ↪ )"
        if ndim >= 2:
            string += ", cy" + str(n)
            string2 += "*gauss(cy" + str(n) + ", sy, y
               ↪ )"
        if ndim >= 1:
            string += ", cx" + str(n)
            string2 += "*gauss(cx" + str(n) + ", sx, x
               ↪ )"
        if n == N:
            if ndim == 3:
                string += ", sz, sy, sx = params"
            elif ndim == 2:
                string += ", sy, sx = params"
    if ndim == 3:
        string3 = "z, y, x"
    elif ndim == 2:
        string3 = "y, x"
    elif ndim == 1:
        string3 = "x"
    else:
        # RAISE AN ERROR!
        pass

    exec("""def residual(params, im, """ + string3 +
       ↪ """):
        """ + string + """
```

```
        return ("""+string2+"""-im)""")
    return residual


def create_params(f, clf, ndim, zmin, ymin, xmin):  #
    ↪ clf=clusterListFinal
    """ This function creates the parameters for
        ↪ fitting

    Parameters
    ----------
    f                   : Found features by Trackpy
    clf                 : ClusterListFinal, contains all
        ↪ the clusters
    ndim                : Number of image dimensions
    zmin, ymin, xmin : Minimum coordinates of the
        ↪ rectangle/box within masking function


    Returns
    ----------
    p0                  : Fitting parameters

    """
    p0 = []
    for i in clf:
        string = ""
        for n in range(1, ndim+1):
            if n == 1:
                string += "f.loc[i]['x'] - xmin]"
            elif n == 2:
                string = "f.loc[i]['y'] - ymin, " +
                    ↪ string
            elif n == 3:
                string = "f.loc[i]['z'] - zmin, " +
                    ↪ string
        exec("""p0 += [f.loc[i]['mass']/ \
        (np.pi*f.loc[i]['size']**ndim),""" + string)
    else:
        p0 += [f.loc[i]['size']]
    return p0
```

```
# clf=clusterListFinal
def create_params_anisotropic(f, clf, ndim, zmin, ymin
    ↪ , xmin):
    """ This function creates the parameters for (
        ↪ anisotropic) fitting

    Parameters
    ----------
    f                   : Found features by Trackpy
    clf                 : ClusterListFinal, contains all
        ↪ the clusters
    ndim                : Number of image dimensions
    zmin, ymin, xmin : Minimum coordinates of the
        ↪ rectangle/box within masking function


    Returns
    ----------
    p0                  : Fitting parameters

    """
    p0 = []
    for i in clf:
        string = ""
        string2 = ""
        for n in range(1, ndim+1):
            if n == 1:
                string += "f.loc[i]['x'] - xmin]"
            elif n == 2:
                string = "f.loc[i]['y'] - ymin, " +
                    ↪ string
                string2 = "p0 += [f.loc[i]['mass']/ \
                    ↪ (np.pi*f.loc[i]['size_y']*np.pi*f.loc[
                    ↪ i]['size_x']),"
            elif n == 3:
                string = "f.loc[i]['z'] - zmin, " +
                    ↪ string
                string2 = "p0 += [f.loc[i]['mass']/ \
```

```
                    (np.pi*f.loc[i]['size_z']*np.pi*f.loc[
    ↪ i]['size_y'] \
                    *np.pi*f.loc[i]['size_x']),"
            exec(string2 + string)
        else:
            if ndim == 3:
                p0 += [f.loc[i]['size_z'],
                       f.loc[i]['size_y'],
                       f.loc[i]['size_x']]
            elif ndim == 2:
                p0 += [f.loc[i]['size_y'],
                       f.loc[i]['size_x']]
        return p0


def generate_masked_image(clf, kdt, image, radius):
        """ This function creates a masked image by
        ↪ selecting a small rectangular region in the
        ↪ image

        Parameters
        ----------
        clf                 : ClusterListFinal, contains all
        ↪ the clusters
        kdt                 : k-d tree
        image
        radius              : Radius of a single particle

        Returns
        ----------
        neighborhood                : Selected rectangular
        ↪ region of the image (masking area)
        zminNew, yminNew, xminNew   : Minimum coordinates
        ↪ of the rectangle/box made within this
        ↪ function

        """
        ndim = image.ndim

        mask = np.asarray(tp.masks.binary_mask(radius,
        ↪ ndim), dtype=int)
```

```
    coords = np.array(kdt.data[clf], dtype=float)
    N = coords.shape[0]
    slices = np.array([[[c - rad, c + rad + 1]
                        for c, rad in zip(coord, radius)
                            ↪ ]
                        for coord in coords])
    if ndim == 3:
        zmaxImage, ymaxImage, xmaxImage = int(image.
            ↪ shape[0]), int(image.shape[1]), int(
            ↪ image.shape[2])
        zmax, zmin = slices[:, 0, 1].max(), slices[:,
            ↪ 0, 0].min()
        ymax, ymin = slices[:, 1, 1].max(), slices[:,
            ↪ 1, 0].min()
        xmax, xmin = slices[:, 2, 1].max(), slices[:,
            ↪ 2, 0].min()
        rect = [slice(np.floor(zmin), np.ceil(zmax)),
                slice(np.floor(ymin), np.ceil(ymax)),
                slice(np.floor(xmin), np.ceil(xmax))]
        coords_rel = coords - np.array([zmin, ymin,
            ↪ xmin])
    elif ndim == 2:
        zmaxImage, ymaxImage, xmaxImage = 0, image.
            ↪ shape[0], image.shape[1]
        zmax, zmin = 0, 0
        ymax, ymin = slices[:, 0, 1].max(), slices[:,
            ↪ 0, 0].min()
        xmax, xmin = slices[:, 1, 1].max(), slices[:,
            ↪ 1, 0].min()
        rect = [slice(np.floor(ymin), np.ceil(ymax)),
                slice(np.floor(xmin), np.ceil(xmax))]
        coords_rel = coords - np.array([ymin, xmin])

    mask_cluster = np.zeros([r.stop - r.start for r in
        ↪  rect], dtype=bool)

    for feat in range(N):
        tempMask = np.zeros([r.stop - r.start for r in
            ↪  rect], dtype=bool)
        coord = coords_rel[feat]
        if ndim == 3:
```

```
            tempMask[coord[0]-radius[0]:coord[0]+
                ↪ radius[0]+1,
                    coord[1]-radius[1]:coord[1]+
                        ↪ radius[1]+1,
                    coord[2]-radius[2]:coord[2]+
                        ↪ radius[2]+1] = mask
        elif ndim == 2:
            tempMask[coord[0]-radius[0]:coord[0]+
                ↪ radius[0]+1,
                    coord[1]-radius[1]:coord[1]+
                        ↪ radius[1]+1] = mask
        mask_cluster = mask_cluster | tempMask

    zmaxNew, zminNew = zmax, zmin
    ymaxNew, yminNew = ymax, ymin
    xmaxNew, xminNew = xmax, xmin

    if zmin < 0:
        zminNew = 0
        mask_cluster = np.delete(mask_cluster, slice
            ↪ (0, abs(int(np.floor(zmin)))), 0)
    if zmax > zmaxImage:
        zmaxNew = zmaxImage
        mask_cluster = np.delete(mask_cluster, slice(
            ↪ int(np.ceil(zmaxNew)), int(np.ceil(zmax)
            ↪ )), 0)
    if ymin < 0:
        yminNew = 0
        mask_cluster = np.delete(mask_cluster, slice
            ↪ (0, abs(int(np.floor(ymin)))), 1)
    if ymax > ymaxImage:
        ymaxNew = ymaxImage
        mask_cluster = np.delete(mask_cluster, slice(
            ↪ int(np.ceil(ymaxNew)), int(np.ceil(ymax)
            ↪ )), 1)
    if xmin < 0:
        xminNew = 0
        mask_cluster = np.delete(mask_cluster, slice
            ↪ (0, abs(int(np.floor(xmin)))), 2)
    if xmax > xmaxImage:
        xmaxNew = xmaxImage
```

```
        mask_cluster = np.delete(mask_cluster, slice(
            ↪ int(np.ceil(xmaxNew)), int(np.ceil(xmax)
            ↪ )), 2)

    if ndim == 3:
        rectNew = [slice(np.floor(zminNew), np.ceil(
            ↪ zmaxNew)),
                   slice(np.floor(yminNew), np.ceil(
                       ↪ ymaxNew)),
                   slice(np.floor(xminNew), np.ceil(
                       ↪ xmaxNew))]
    elif ndim == 2:
        rectNew = [slice(np.floor(yminNew), np.ceil(
            ↪ ymaxNew)),
                   slice(np.floor(xminNew), np.ceil(
                       ↪ xmaxNew))]

    neighborhood = mask_cluster*image[rectNew]

    return neighborhood, zminNew, yminNew, xminNew


def create_new_coords(new_coords, ndim, fit, zmin,
    ↪ ymin, xmin):
    """ This function puts all the single cluster
        ↪ coordinates into one list of all fitted
        ↪ coordinates

    Parameters
    ----------
    new_coords        : All the fitted values (empty at
        ↪  start)
    ndim              : Number of image dimensions
    fit               : Fitted coordinates for one
        ↪ cluster
    zmin, ymin, xmin : Minimum coordinates of the
        ↪ rectangle/box within masking function

    Returns
    ----------
```

```
    new_coords : all the corrected coordinates for all
        ↪ the clusters (at this point)

    """
    string = ""
    tempString = ""

    if ndim < 4:
        for n in range(1, ndim+1):
            tempString = "fit[0][(ndim+1)*i+" + str(
                ↪ ndim+1 - n) + "] + "
            if n == 1:
                tempString += "xmin"
            elif n == 2:
                tempString += "ymin, "
            elif n == 3:
                tempString += "zmin, "
            string = tempString + string
        for i in range(int(len(fit[0]) // (ndim+1))):
            exec("""new_coords += [[""" + string + """
                ↪ ]]""")
    else:
        # RAISE AN ERROR!
        pass
    return new_coords


def fit_clusters(features, image, partDiameter,
    ↪ maxDist=20):
    """ This function first finds all clusters by
        ↪ using find_clusters()
    When all the clusters are found, gaussians are
        ↪ fitted to each cluster

    Parameters
    ----------
    features : all the features found by trackpy.
        ↪ locate
    image

    Returns
```

```
    ----------
    new_coords : all the corrected coordinates for all
        ↪ the clusters

    """
    image = pims.Frame(image)
    ndim = image.ndim
    partDiameter = validate_tuple(partDiameter, image.
        ↪ ndim)
    radius = tuple([x//2 for x in partDiameter])

    f = features
    clusterListFinal, kdt = find_clusters(f, ndim,
        ↪ maxDist)
    new_coords = []

    j = 0
    for cluster in clusterListFinal:
        neighborhood, zmin, \
            ymin, xmin = generate_masked_image(
                ↪ clusterListFinal[j], kdt, image,
                ↪ radius)

        if 'size' in f:
            residual = generate_residual(len(cluster),
                ↪ ndim)
            params = create_params(f, cluster, ndim,
                ↪ zmin, ymin, xmin)
        if 'size_z' in f or 'size_y' in f or 'size_x'
            ↪ in f:
            residual = generate_residual_anisotropic(
                ↪ len(cluster), ndim)
            params = create_params_anisotropic(f,
                ↪ cluster, ndim, zmin, ymin, xmin)

        if ndim == 3:
            z, y, x = np.indices(neighborhood.shape)
            fit = leastsq(residual, params, args=(
                ↪ neighborhood.ravel(), z.ravel(), y.
                ↪ ravel(), x.ravel()))
        elif ndim == 2:
```

```
            y, x = np.indices(neighborhood.shape)
            fit = leastsq(residual, params, args=(
               ↪ neighborhood.ravel(), y.ravel(), x.
               ↪ ravel()))
        elif ndim == 1:
            x = np.indices(neighborhood.shape)
            fit = leastsq(residual, params, args=(
               ↪ neighborhood.ravel(), x.ravel()))
        else:
            # RAISE AN ERROR!
            pass

        j += 1
        new_coords = create_new_coords(new_coords,
           ↪ ndim, fit, zmin, ymin, xmin)

    return new_coords
```