Gelation and Phase Separation of Attractive Colloids

A dissertation presented
by

Peter James Lu

to
The Department of Physics
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy
in the subject of

Physics

Harvard University
Cambridge, Massachusetts
May 2008

©2008 - Peter James Lu

All rights reserved.

Thesis advisor Author

David Allan Weitz **Peter James Lu**

Gelation and Phase Separation of Attractive Colloids

Abstract

I present several scientific and technical contributions in this thesis. I demonstrate that the gelation of spherical particles with isotropic, short-range attractive interactions is initiated by spinodal decomposition, a thermodynamic instability that triggers the formation of clusters that span and dynamically arrest to create a gel. This simple, universal gelation picture does not depend on microscopic system-specific details—thus broadly describing any particle system with short-range attractions—and suggests that gelation, often considered a purely kinetic phenomenon, is in fact a direct consequence of equilibrium liquid-gas phase separation. I also demonstrate that spherical particles with isotropic attractive interactions exhibit a stable phase—a fluid of particle clusters—that persists on experimental timescales even in the absence of any long-range Coulombic charge repulsion; this contrasts some expectations based on simulation and theory. I describe a new capability I created by integrating accelerated image processing software that I wrote into a high-speed confocal microscope system that I developed: active target-locking, the ability to follow freely-moving complex objects within a microscope sample, even as they change size, shape, and orientation—in real time. Finally, I report continuous, month-long observations of near-critical spinodal decomposition of colloids with isotropic attractions, aboard the International Space Station. I also include detailed descriptions, with examples and illustrations, of the tools and techniques that I have developed to produce these results.

Contents

List of Figures

Citations to Previously Published Work

Chapter 2 has been published as the first chapter in an edited book:

> Peter J. Lu, "Confocal Scanning Optical Microscopy and Nanotechnology," in *Handbook of Microscopy for Nanotechnology*, Nan Yao and Zhong Lin Wang eds. (New York: Kluwer, 2005), pp. 3-24.

Chapter 4 has been published and patented as:

> Peter J. Lu, Peter A. Sims, Hidekazu Oki, James B. Macarthur and David A. Weitz, "Target-locking acquisition with real-time confocal (TARC) microscopy," *Optics Express* **15**, 8702 (2007).

> Peter J. Lu, "Target-locking Acquisition with Real-Time Confocal (TARC) Microscopy," U.S. Provisional Patent Application, Serial Number 60/932,396, filed on May 31, 2007.

Chapter 9 has been published as:

> Peter J. Lu, Jacinta C. Conrad, Hans M. Wyss, Andrew B. Schofield and David A. Weitz, "Fluids of Clusters in Attractive Colloids," *Physical Review Letters* **96**, 028306 (2006).

Chapter 10 has been published as:

> Peter J. Lu, Emanuela Zaccarelli, Fabio Ciulla, Andrew B. Schofield, Francesco Sciortino and David A. Weitz, "Gelation of particles with short-range attraction," *Nature* **453**, (2008).

Electronic copies of all papers are available online at:

> `http://www.peterlu.org`

Acknowledgments

Obviously, I could not have completed this work without the generous assistance of a large number of individuals. First and foremost are my scientific collaborators, identified in each chapter and listed as co-authors of the corresponding papers cited on the preceding page. Additional contributions, particularly specific technical assistance, are described in a separate acknowledgements section within each chapter, which also include project-specific grant information. More generally, I was supported for my first three years of graduate school at Harvard by the National Defense Science and Engineering Graduate Fellowship, a program of the United States Department of Defense. My subsequent years have been funded primarily by grants from the Microgravity Sciences Program at the National Aeronautics and Space Administration to my advisor David A. Weitz, Mallinckrodt Professor of Physics and of Applied Physics at Harvard University. In addition, I received crucial financial support of a number of individuals and corporations. My parents, Dr. Calvin and Mrs. Florence M. Lu, generously paid for the computer workstations which I used for the vast majority of programming, analysis and rendering. This work simply could not have been completed in any reasonable amount of time with the otherwise available computing resources, which highlights the rather unfortunate requirement of independent financial resources when pursuing a physics PhD these days. Intel, nVidia and Pixar provided crucial pieces of hardware and software to me at low or no cost, via their registered developer and educational grants programs. On a personal note, I thank my former roommate Motohiro Yogo, at present a Visiting Assistant Professor in the Department of Economics and the Bendheim Center for Finance at Princeton University and a Faculty Research Fellow of the National Bureau of Economic Research, on leave from the Wharton School of the University of Pennsylvania, where he is an Assistant Professor of Finance.

Dedicated to my grandfather, Benjamin Chih-Yeu Lu, Professor of Chemical Engineering, emeritus, at the University of Ottawa, who has worked over the last half-century on the equilibrium phase separation of fluids.

Chapter 1

Introduction

This thesis describes my work on phase separation and gelation in attractive colloid-polymer mixtures. In this brief introductory chapter, I provide some basic context for the scientific problems addressed in the thesis, and summarize the major results, at a level intended to address nonspecialist physicists familiar with the basics of undergraduate statistical mechanics.

1.1 Colloids: Why do Physicists care?

Colloids are dispersions of mesoscale particles in a fluid background solvent. Common examples include ink, milk, mayonnaise, paint, and smoke, so their study has myriad practical applications [134]. Here, I focus on colloidal systems comprising solid, micron-scale particles dispersed in a liquid medium. For physicists, these micron-scale colloids provide an important platform for investigating statistical physics, for several main reasons.

1. They are small enough that their thermal energy $k_\mathrm{B}T$ drives their dynamics.

2. They are large enough to interact with visible light, and can therefore be identified and tracked on a single-particle level, even in bulk samples.

3. Their interactions can be precisely manipulated.

To understand the first condition, consider a sphere of polystyrene (with density $\rho_\mathrm{p} = 1.05$ g/mL and diameter $d = 2a = 1\mu$m) free to diffuse in water (with density $\rho_\mathrm{w} = 1.00$ g/mL). The gravitational energy U_g needed to lift the particle its own size is:

$$U_\mathrm{g} = m \cdot g \cdot \Delta h = \frac{4}{3}\pi a^3(\rho_\mathrm{p} - \rho_\mathrm{w}) \cdot g \cdot 2a = 2.6 \times 10^{-23}\ \mathrm{J} \tag{1.1}$$

By contrast, the thermal energy $k_\mathrm{B}T$ at room temperature ($T = 298$K) is 4.1×10^{-21} J. Clearly, $k_\mathrm{B}T \gg U_g$, so the particle will naturally diffuse around in solution, with relatively little influence of gravity on short time scales up to hours or days. This contrasts the granular limit, where gravity essentially pins the particles, and thermal motion is considered irrelevant; for example, if an otherwise identical polystyrene particle were a millimeter in diameter, it would require 12 orders of magnitude more energy—because of the d^4 dependence in the energy—to lift the particle its own size, which is ten orders of magnitude higher than the available thermal energy.

Because particles on the micron scale are larger than the wavelength of visible light (400 to 700 nm), they can be resolved individually with ordinary light. Many convenient techniques are therefore applicable, particularly optical microscopy and light scattering, and investigations do not require the more difficult methods of electron microscopy, X-ray and neutron scattering. From a practical experimental standpoint, this vastly increases the possible techniques and amount of data that can be gathered. But more importantly, the ability to identify each and every particle individually gives these system a significant advantage over molecules, where only under very specific conditions can individuals be observed. In general, studies of molecular systems, particularly as pertaining to statistical mechanics, can only access bulk quantities averaged over relatively large collections of molecules, e.g. on the order of Avogadro's number $N_\mathrm{A} \sim 10^{23}$. With colloids, individual particles can be identified and followed through time, as well as large 3D bulk systems.

Finally, there are a host of possible colloidal interparticle interactions that can be generated with various mechanisms in the laboratory [226, 285]. Particles can be made attractive, repulsive or perfectly neutral, by controlling chemistry, temperature, or other easily-accessed experimental variables, such as the concentration of flexible polymer additives that create the depletion effect [10, 120, 169, 276].

1.2 Hard Spheres

The simplest colloidal system is the hard spheres, in which particles neither attract nor repel over distances greater than their diameter, but cannot interpenetrate. Two hard sphere particles of radius a have an infinite repulsion if their center separation r is less than $2a$, but zero otherwise. This potential is therefore described by [51]:

$$U(r \leq 2a) = \infty \tag{1.2}$$

$$U(r > 2a) = 0 \tag{1.3}$$

The free energy $U - TS$ is then entirely driven by the entropic contribution: the particles simply move to increase their local free volume, which maximizes the number of spatial configuration states (i.e. positions) that they can occupy. The only control parameter that can be adjusted is the overall density of spheres, expressed as a particle volume fraction ϕ

(total volume of the spheres divided by total volume of the system). Because temperature does not play a role in the ultimate equilibrium configuration attained—though it obviously affects kinetics by controlling the rate of diffusion of particles through the solvent—these hard-sphere colloid systems are sometimes described as *athermal*.

1.2.1 Equilibrium Behavior

The equilibrium phase behavior of the hard spheres is well understood, and has been widely published [81]. Because the only control parameter is ϕ, the phase diagram is one dimensional. For $\phi \leq 0.49$, the hard sphere system is a fluid. For $0.54 \leq \phi \leq 0.74$, the equilibrium state of the system is a face-centered cubic (FCC) crystal. Interestingly, for these high volume fractions, an ordered, crystalline state represents the phase with *higher* entropy, relative to a disordered configuration at the same ϕ; a crystal maximizes local volume available to each particle, and therefore increases the total number of spatial configurations available to the system. $\phi = 0.74$ is the maximum ϕ possible, representing the limit of close-packed spheres [11]. For $0.49 \leq \phi \leq 0.54$, the overall free energy is minimized if the system partitions into two coexisting phases: some particles crystallize, while the rest remain a fluid, with constant exchange between the two.

1.2.2 Kinetic Arrest

For $\phi \approx 0.58$, if nucleation of the FCC crystals is avoided, something interesting happens that is not apparent from a consideration of equilibrium thermodynamics: the system arrests to form a solid, but instead of its constituent particles adopting the ordered arrangement of a crystal, they remain disordered—structurally identical to a liquid, in fact—but frozen in place as a *glass* [243, 245]. What distinguishes an arrested glass from an equilibrium liquid is not structure, but dynamics. The particles in a colloidal hard-sphere liquid will traverse all points of a finite volume in a finite time determined by their rate of diffusion, and thus are considered ergodic. By contrast, particles in a glass are bounded, and can only move to nearby points, within a so-called cage created by the crowding of other nearby particles. This confinement is typically quantified by the ensemble-averaged mean-squared displacement:

$$\left\langle \left|\Delta\vec{r}(t)\right|^2 \right\rangle = \left\langle \left|\vec{r}(t)-\vec{r}(0)\right|^2 \right\rangle \tag{1.4}$$

which grows monotonically without bound for particles in a liquid, but plateaus at long times for particles in a glass.

These simple concepts of entropy-driven equilibrium phase coexistence and kinetic arrest are well understood in the hard sphere system. Many types of simulations can easily reproduce the experimentally-observed equilibrium phase diagram, and mode-coupling theory captures some aspects of the experimentally-observed dynamics of the hard-sphere glass [243, 245].

1.3 Hard Spheres with Interactions

1.3.1 Purely Repulsive Spheres

The situation is not so simple, however, when additional interactions are added to the hard-sphere particles. In general, these interactions can be purely repulsive, purely attractive, or a combination of both [290]. The purely repulsive case, as occurs with Coulombic repulsion, is usually (and successfully) treated by renormalizing the sphere radius a to a value—larger than the physical hard-sphere radius of the particle—that instead depends on the amount of charge [303]. Using this renormalized radius (related to the Debye screening length), many of the hard sphere concepts and results can be applied immediately to describe the behavior of the system. These charge-repulsive systems have been exploited in systems with more than one species of colloid to create novel crystal structures, some not observed in atomic or molecular systems [156].

1.3.2 Purely Attractive Spheres

When an attractive interaction is added to hard-sphere particles, the phase behavior becomes significantly more rich and diverse. For example, distinct liquid and gas phases, and a liquid-gas critical point, become possible [6, 1, 184, 198, 247, 248]. In the hard-sphere case, the only parameter that defines phase behavior is the volume fraction ϕ, and the only length scale in the problem is the particle radius a. Purely attractive interparticle potentials introduce several new parameters: the strength of the attraction $U/k_{\mathrm{B}}T$ (expressed in units of the thermal energy $k_{\mathrm{B}}T$), and the range of the attraction ξ (expressed as a fraction of a). These three parameters—ϕ, $U/k_{\mathrm{B}}T$ and ξ—form a three-dimensional phase space [146, 171], where phase boundaries become two-dimensional surfaces (see, for example, figure 9.8). For attractive colloidal systems, a convenient way to break this 3D state diagram down is to consider different regimes in ξ, looking at 2D $U-\phi$ planes perpendicular to the ξ axis. The high-ϕ regions of the phase space are typically dominated by crowding effects, just as in the hard-sphere case; attractive particles unsurprisingly form glasses at high volume fractions [83, 193]. The main interest of this thesis, however, is to look at $\phi \ll 0.5$, where the behavior is far from generally understood.

Long-range attraction

The region where $\xi \sim 1$ (i.e. the attraction range is comparable to the size of the particle) is sometimes termed a "long-range" attraction. This is similar in many ways to intermolecular potentials, where the distance over which the quantum-mechanical attraction acts is comparable to the molecular size, typically measured in angstroms. At intermediate volume fractions, e.g. $\phi \sim 0.25$, equilibrium phase separation, in the form of spinodal decomposition, is observed, as reported in [18] and chapter 11 in this thesis. These observations, made aboard the International Space Station in microgravity—both deeply quenched into the spinodal ($U/k_\mathrm{B}T \gg 1$) [18], and very close to the critical point ($U/k_\mathrm{B}T \sim 1$; see chapter 11)—show features and scaling quite similar to those known for molecular liquid-gas spinodal decomposition [99, 257]. This phase separation can be interrupted if the dense liquid phase crosses an attractive glass line; the particles within the bicontinuous network created by spinodal decomposition freeze to form a glass, which has been observed experimentally and described by mode-coupling theory [176]. The resulting spanning, arrested structure is called a *gel*, and is the predominant form of kinetic arrest in this area of the phase diagram. Note that while the *local* volume fraction of particles within the bulk may approach that of a hard-sphere glass at $\phi \approx 0.58$, the overall structure may be branched and tenuous, so that the volume fraction of the entire system is much lower; for example, the gel shown in figure 9.4 has $\phi = 0.15$.

Short-range attraction

The other regime relevant to a large class of attractive colloids is driven by "short-range" attractive potentials, where $\xi \ll 1$. Many important attraction mechanisms that drive gelation are short-range, including van der Waals forces [3, 266, 274, 302], surface chemistry [103, 300, 112, 283, 221, 71, 223, 305], hydrophobic effects [44], and some depletion interactions [25, 251, 252, 253, 38]. Here, there is far less consensus surrounding phase behavior and kinetic arrest.

Early equilibrium phase diagrams, computed using mean-field theory, found only the presence of a gas-crystal phase separation, analogous to the hard spheres, but with a wider coexistence as $U/k_{\mathrm{B}}T$ increases [154, 136]. Subsequent simulations found, within this gas-crystal coexistence (which in practice can be hidden by a small amount of polydispersity), a second liquid-gas binodal [76, 77, 78].

As for kinetic arrest, in addition to the glass transition at high-ϕ [193, 203], another limit is well understood: diffusion-limited cluster aggregation (DLCA), where particles at arbitrarily-low density stick irreversibly upon contact, and can form fractal clusters [292, 293, 294, 295, 79, 162, 165, 163, 164, 45, 47, 48]. In our notation, the DLCA limit is expressed as $\phi \to 0$, $U/k_{\mathrm{B}}T \to \infty$ and $\xi \to 0$. Gels can form under these conditions, as well, in the absence of gravity, where the lower limit for ϕ is set by thermal fluctuations [175].

But it is the intermediate volume fractions $0 \ll \phi \ll 0.5$ that are of greatest interest, particularly the widespread observation of gelation. Exactly what causes this gelation has been a matter of vigorous debate in the literature. Numerous explanations have been advanced for gelation in this small-ξ limit to predict the fluid-solid boundary in the $U - \phi$ plane. Non-equilibrium, kinetics-based models have extended the DLCA paradigm to lower $U/k_{\mathrm{B}}T$ [255, 256, 222, 167, 70, 195, 196, 197, 5, 34, 207]; have connected the gelation boundary to the percolation threshold [112, 283, 196, 146, 72]; have extended the glass transition to lower ϕ with mode-coupling theory (MCT) applied to local arrest of individual particles [25, 22, 23, 24, 90, 202, 203, 204, 205, 260], arrest of clusters [146], and in concert with microscopic modeling of the interparticle attractive potential using PRISM [251, 252, 253, 54, 53, 55, 57, 58, 97, 98, 214, 215, 216]); and have connected gelation in a general picture of arrest known as jamming [274, 275].

Thermodynamic models consider gelation initiated by fluid-crystal [208, 155], liquid-gas [70, 52, 38, 5, 284], or polymer-like “viscoelastic” [266, 267] phase separation, which may arrest due to percolation [196, 109, 135] or a glass transition [146]. These models make strikingly disparate predictions: there is no agreement on either the gelation mechanism, the location of the gelation boundary [304, 196, 199, 251, 252, 253, 214, 215], or even whether arrest is observed [15, 91, 92].

The most important contribution of this thesis is to address this debate with experiments, which demonstrate that the gelation of spherical particles with isotropic, short-range attractive interactions is initiated by spinodal decomposition, a thermodynamic instability that triggers the formation of clusters that span and dynamically arrest to create a gel. This simple, universal gelation picture does not depend on microscopic system-specific details—thus broadly describing any particle system with short-range attractions—and suggests that gelation, often considered a purely kinetic phenomenon, is in fact a direct consequence of equilibrium liquid-gas phase separation [12, 13, 241, 242, 46, 107, 217]. This is described in chapter 10.

1.3.3 Spheres with short-range attractions and longer-range repulsions

The situation is even more complicated when particles have both attractive and a repulsive potential components, specifically a short-range attraction and a longer-range repulsion [246, 114, 42, 94]. There are now two energy scales (the strengths of the attraction, and of the repulsion), and at least three length scales: a, ξ and the length characterizing the extent of the repulsion. In the absence of a clear, general understanding of attractive-only potentials, it is difficult to imagine that a deep understanding of these even more complex combination potentials is at hand.

Nevertheless, it has been claimed that this combination of short-ranged attraction and long-ranged repulsion is *required* to have large, stable clusters of colloids. In this scenario, particles are naturally drawn to aggregate with their short-ranged attraction, sticking to each other upon contact; after a sufficient number of particles have aggregated, though, the cluster acquires enough charge to Coulombically repel any additional particle, limiting the clusters to a characteristic size, and preventing further aggregation [114, 69, 244]. We have shown that, while this mechanism may in fact work in specific situations, it certainly is not *necessary* in general for the existence of large clusters stable on experimental timescales. We have observed large clusters in a purely attractive potential where the long-range Coulombic repulsion has been screened, as described in chapter 9. This combination potential has also been held responsible for the formation of colloidal gels [73, 80], sometimes with chains in the form of Bernal spirals, predicted in simulation and observed in experiment [42]. While this mechanism may drive these systems, the morphology of gels in general has not been characterized, and we have observed similar structures in gels of purely attractive particles.

1.4 Summary of Major Results

This thesis presents several contributions to the understanding of colloid physics. In chapter 9 (published as [171]), we show that experimentally long-lived clusters can exist in systems of purely-attractive colloids at various ϕ and ξ, without any long-range Coulombic repulsion. The experimental evidence we present responds to claims that this Coulombic repulsion is a necessary component for the existence of stable clusters; in fact, it is not.

In chapter 10 (published as [173]), we demonstrate how the formation of gels of colloidal particles with short-range attractions is driven by spinodal decomposition. This addresses a longstanding debate over the origins of short-range gelation, and in particular excludes a number of suggested mechanisms: neither purely kinetic theories extending the attractive glass transition to lower ϕ, nor phase separation that is not liquid-gas, is able to explain the formation of these gels.

In chapter 11, I report continuous, month-long observations of liquid-gas spinodal decomposition near the critical point of colloid samples with long-range attractions, in the microgravity environment of the International Space Station.

This thesis also describes in detail the technical developments that were necessary for the scientific investigations. I developed a high-speed confocal microscope system that exceeds commercial systems in robustness, speed, and ability to continuously collect data sets of arbitrary size, which is described in chapter 5. I significantly improved a number of image and data analysis routines, surpassing the performance of existing algorithms by two orders of magnitude, as described in chapter 6. By integrating the accelerated image-processing software that I wrote into the high-speed confocal microscope system that I developed has resulted in the creation of a completely new capability: active target-locking, the ability to translate a microscope stage to follow a freely-moving complex object—even as it changes size, shape, and orientation. This system is described in chapter 4 (published as [172]), and has been patented, as well.

1.5 Thesis Structure Overview

The thesis is organized into three major sections, each of which has several chapters. The first section (Background) provides general background on confocal microscopy and its application to nanotechnology (chapter 2, published as [170]), and a primer on spinodal and binodal decomposition (chapter 3). These are generally applicable well beyond the specific focus of the problems in attractive colloids.

The second section (Engineering) addresses the specific engineering and development challenges that we overcame in the successful pursuit of the scientific results. We recently received the following comment in anonymous referee report about the paper published as [173] and presented here in chapter 10:

> "I think this paper should be published in *Nature*. The techniques, analysis and reasoning are all interesting and useful on their own."

Alas, short papers do not leave much room to discuss the "techniques, analysis and reasoning." The chapters in this section were written to fill in these gaps, and detail the development of those tools and techniques used to produce the published scientific results. Chapter 4 gives a technical overview of the PLuTARC system, which enables real-time target-locking of objects moving along arbitrary three-dimensional paths in a confocal microscope sample, even as the objects change shape, size and orientation. Technical details of the system are elaborated in the three subsequent chapters. Chapter 5 describes the details the hardware and software design of the high-speed confocal microscope system, specifically discussing the choices made to ensure its robust stability when continuously acquiring tens of thousands of images at high speed, without interruption. Chapter 6 describes the software I wrote for locating colloidal particles quickly and accurately, building on previous algorithms but introducing several important features to increase performance by several orders of magnitude. All of the important steps in the image processing protocol are documented with the relevant C++ code fragments, and illustrated with example images. Chapter 7 describes and illustrates the code and algorithms I wrote for structural analysis to characterize gels and clusters, with C++ code and example results for density correlation functions, local nearest-neighbor distributions, cluster-mass distributions, and a novel algorithm for computing the internal volume fraction of a disordered gel. Chapter 8 describes the evolution of my work in visualizing disordered, chain-like gel structures, with examples of different types of shading and lighting, using Pixar's RenderMan. When deciding which specific topics to discuss, I took the minimalist approach of including only developments and analyses that led to actual published results.

These chapters are meant to describe exactly what I did—and why—to those already familiar with the basics of the relevant technical tools. They are not, however, a general primer on programming or hardware. The C++ code is included so that an experienced programmer can quickly devise the algorithm, and the optimizations made for performance reasons that crucially enable the analysis of far larger data sets than has previously been possible; only the final code that has actually been used in published work and other parts of this thesis are included. These chapters have been written specifically as a reference to others working in the field that want to take the code (which has all been made available online to accompany the publication of the PLuTARC system in [172]) and use and/or extend it for their own purposes.

The third section—Science—describes the scientific contributions. The results of chapters 9, 10 and 11 are discussed above, in section 1.4. Chapter 11 also includes detailed methods of sample preparation and characterization that apply to the rest of the colloid work presented here, again in a format so that practitioners of colloid physics can immediately read and understand specifically what was done.

Finally, note that for chapters 2, 4, 9 and 10, which have already been published, the thesis chapter text is exactly what appears in the publications, except for minor rearrangements and renumbering of figures and their captions, to improve clarity in the thesis format.

1.6 Colophon

This thesis was entirely prepared with LaTeX [151], using the MikTeX 2.6 distribution and WinEdt 5.5 as the text editor. The text is set with the default Times fonts; the equations and symbols, with MathTime Professional II Lite fonts. The default for formulae is to use the properly slanted Greek letters. Subscript characters that form part of the variable name are written in Roman letters, while those that are variables or indices are written in italics. The uppercase Greek omega, referring to the electrical resistance unit ohm, was created with the MathTime-specific code `\upOmega`; the entire document is otherwise entirely standard-coded LaTeX. I created all of the figures, choosing the Adobe Minion Pro font with the caption weight for all figure text. Vector images were saved as .EPS in Adobe

Illustrator CS3 format; Raster images (and composites with large raster components) were saved as Photoshop .EPS format from Adobe Photoshop CS3. MikTeX (through DVIPS) generated a PostScript file; Adobe Acrobat Distiller 8.1.2 was used to generate the final PDF file, with the "Press Quality" standard preset.

This thesis is permanently available through the archives of Harvard University, and online at:

```
http://www.physics.harvard.edu
http://www.peterlu.org
```

Any addenda or corrigenda will be posted at the above online addresses, as well.

Part I

Background

Chapter 2

Confocal Scanning Optical Microscopy and Nanotechnology

2.1 Abstract

Confocal scanning optical microscopy is the most developed and widely used of recent high-resolution optical microscopy techniques seeking to increase resolution over traditional far-field optical microscopes. Although the diffraction limit might hint that optical microscopy is unable to investigate nanoscale systems, the confocal has in fact contributed in unique and powerful ways. By rapidly localizing small objects in three dimensions with precision approaching single nanometers, and using spatially-resolved spectroscopy to characterize them, the confocal has yielded information about a number of nanotechnologically-relevant systems that is not accessible by any other technique. These systems include nanoemulsions, nanocapsules, ferro-electric thin films, nanopores, nanoholes, nanomembranes, nanotubes, nanowires, nanocrystals, viruses, and single-molecules.

2.2 Introduction

2.2.1 Optical Microscopy

Microscopy is the characterization of objects smaller than what can be seen with the naked human eye, and from its inception, optical microscopy has played a seminal role in the development of science. In the 1660s, Robert Hooke first resolved cork cells and thereby discovered the cellular nature of life [128]. Robert Brown's 1827 observation of the seemingly random movement of pollen grains [37] led to the understanding of the motion that still bears his name, and ultimately to the formulation of statistical mechanics. The contributions of optical microscopy continue into the present, even as the systems of interest approach nanometer size. What makes optical microscopy so useful is the relatively low energy of visible light: in general, it does not irreversibly alter the electronic or atomic structure of the matter with which it interacts, allowing observation of natural processes in situ. Moreover, light is cheap, abundant, and can be manipulated with common and relatively inexpensive laboratory hardware.

In an optical microscope, illuminating photons are sent into the sample. They interact with atoms in the sample, and are re-emitted and captured by a detection system. The detected light is then used to reconstruct a map of the sample. An ideal microscope would detect each photon from the sample, and measure with infinite precision the three-dimensional position from which it came, when it arrived, and all of its properties (energy, polarization, phase). An exact three-dimensional map of the sample could then be created with perfect fidelity. Unfortunately, these quantities can be known only to a certain finite precision, due to limitations in both engineering and fundamental physics.

One common high-school application of optical microscopy is to look at small objects, for example the underside of a geranium leaf. Micron-scale structure is easily revealed in the top layer of plant cells. But structure much smaller than a micron (such as individual macromolecules in the plant cell) cannot be seen, and looking deep into the sample (e.g. tens of cell layers) leads only to a nearly featureless blur. Clearly this is a far cry from the ideal microscope above.

2.2.2 Improving Resolution

Microscopes with improved resolution fall into two broad categories, near-field and far-field. Near-field techniques rely on scanning a nanoscale optical probe only nanometers above the surface of interest. Spatial resolution is then physically limited only by the lateral size of the tip of the probe, and information can only be gathered from the surface. This technique is the subject of another chapter in this text. In far-field, a macroscopic lens (typically with mm-scale lens elements) collects photons from a sample hundreds of microns away. Standard microscopes, like the one used in high-school, are of this type. The light detected often comes from deep within the sample, not just from the surface. Moreover, there are often enough photons to allow collection times sufficiently brief to watch a sample change in real time, here defined to be the video rate of about 25 full frames per second.

2.2.3 Far-field Microscopy and the Diffraction Limit

But all far-field techniques encounter the fundamental physical diffraction limit, a restriction on the maximum spatial resolution. In the present parlance, the precision with which the location of the volume generating a given detected photon (here termed the illumination volume) can be determined is roughly the same size as the wavelength of that photon [35]. Visible light has a wavelength of roughly a half micron, an order of magnitude greater than the feature size of interest to nanotechnology.

At first blush, then, the idea that far-field optical microscopy can contribute much to nanotechnology may appear absurd. However, a number of techniques have been developed to improve the precision with which the spatial position of an illumination volume can be determined. The most prevalent of these is confocal microscopy, the main subject of this chapter, where the use of a pinhole can dramatically improve the ability to see small objects. Other techniques have the potential for further improvements, but none so far has been applied widely to systems relevant to nanotechnology.

2.2.4 Resolution versus Localization

Several terms are commonly used to describe improvements in “seeing” small objects. Resolution, or resolving power, is the ability to characterize the distribution of sample inhomogeneities, for example distinguishing the internal structure of cells in Hooke’s cork or the geranium leaf. Resolution is ultimately restricted by the diffraction limit: no optical technique, including confocal, will ever permit resolution of single atoms in a crystal lattice with angstrom-scale structure. On the hand, localization is the determination of the spatial position of an object, and this is possible even when the object is far smaller than the wavelength. Localization can be of an object itself, if there is sufficient optical contrast with the surrounding area, or of a fluorescent tag attached to the object. The former is generally more common in the investigation of nanoscale materials, where in many instances (e.g. quantum dots) the nanomaterials are themselves fluorescent. The latter is common in biology, where the confocal is often used to localize single-molecule fluorescent probes attached to cellular substructures. But in many of these systems, the tags can be imaged without confocality, such as in thin cells where three-dimensional sectioning is unneeded, or when the tags are spaced out by microns or more.

Precise localization is of tremendous utility when the length scale relevant to the question at hand is greater than the wavelength being used to probe the sample, even if the sample itself has structure on a smaller length scale. For example, Brown observed micron-scale movements of pollen grains to develop his ideas on motion, while the nanoscale (i.e. molecular) structure of the pollen was entirely irrelevant to the question he was asking. The pollen served as ideal zero-dimensional markers that he could observe; their position as a function of time, not their structure, was ultimately important. In many instances, the confocal plays a similar role, where fluorescent objects serve as probes of other systems. By asking the right questions, the diffraction limit only represents a barrier to imaging resolution, not a barrier to gathering information and answering a properly formulated scientific question.

Ultimately, the confocal is not a fancy optical microscope that through special tricks allows resolution of nanoscale objects. Rather, the confocal makes the greatest contribution to nanotechnology with rapid, non-destructive three-dimensional nanoscale localization of the sample area generating a given detected photon, and the analysis (spectroscopy) of that photon. This localization property of the confocal allows real-time spectroscopy of *individual* nanoscale objects, instead of ensemble averages. As such, the confocal plays a singularly important role in the investigation of structure and dynamics of systems relevant to nanotechnology, complementing the other techniques described in this volume.

This review begins with a qualitative overview (no equations) of confocal microscopy, with a brief discussion of recent advances to improve resolution and localization. Following that is a survey of recent applications of confocal microscopy to systems of interest to nanotechnology.

2.3 The Confocal Microscope

2.3.1 Principles of Confocal Microscopy

Several texts comprehensively review the confocal microscope, how it works, and the practical issues surrounding microscope construction and resolution limitations [254, 63, 75, 115]. This section is a brief qualitative overview to confer a conceptual understanding of what a confocal is, namely how it differs from a regular optical microscope, and why those differences are important for gaining information from structures relevant to nanotechnology. All of the applications of confocal microscopy described here rely on fluorescence. That is, the incoming beam with photons of a given wavelength hits the sample, and interactions between illumination photons and sample atoms generates new photons of a *lower* wavelength, which are then detected. The difference in the two wavelengths must be large enough to allow separation of illumination and detection beams by mirrors, called dichroics, that reflect light of one color and pass that of another. In practice, the separation is usually tens of nanometers or more.

The noun “confocal” is shorthand for *confocal scanning optical microscope*. Parsing in reverse, *optical* microscope indicates that visible radiation is used, and confocals are often based on, or built directly as an attachment to, optical microscopes with existing technology. Unlike traditional widefield optical microscopes, where the whole sample is illuminated at the same time, in confocal a beam of laser light is *scanned* relative to the sample, and the only light detected is emitted by the interaction between the illuminating beam and a small sample illumination volume at the focus of the microscope objective; due to the diffraction limit, the linear extent of this volume is approximately the wavelength of light. In a confocal, light coming back from the illumination volume is focused down to a another diffraction-limited spot, which is surrounded by a narrow pinhole. The pinhole spatially filters out light originating from parts of the sample *not* in the illumination volume. Because it is positioned at a point conjugate to the focal point in the sample, the pinhole is said to be *confocal* to it, and the pinhole allows only the light from the focused spot (that is, the illumination volume) to reach the detector.

A schematic of a typical confocal is given in figure 2.1. Light from a laser beam is reflected by a dichroic and focused onto a spot on the sample in the $x - y$ plane by the microscope objective. The optic axis is along the z direction. Light from the sample, at a lower wavelength, comes back up from the illumination volume via the objective, passes through the dichroic, and is focused onto a point, surrounded by a pinhole, that is confocal with the objective's focal point on the sample. The detected light then passes to the detector. The laser beam illuminates parts of the sample covering a range of depths, which in an ordinary microscope contribute to the detected signal, and blur the image out; this is the reason that, tens of cell layers deep, the image of the geranium is blurry. In the confocal, however, the pinhole blocks all light originating from points not at the focus of the microscope objective, so that only the light from the illumination volume is detected;

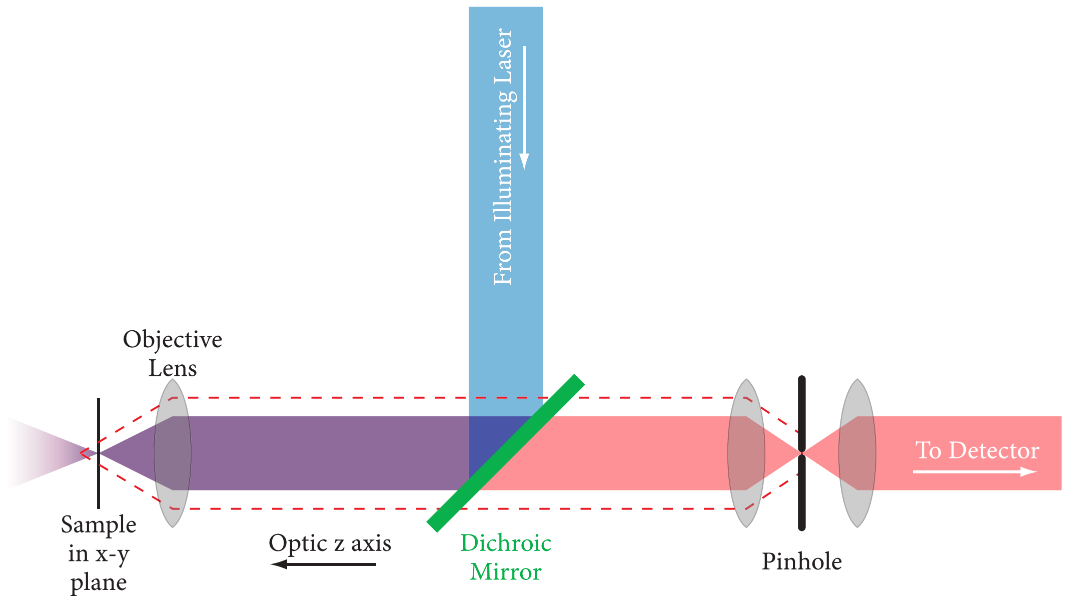

Figure 2.1: Schematic diagram of a confocal microscope. Laser light (blue) is reflected by the dichroic, and illuminates the sample at the focus of the objective. This excites fluorescence, and the sample then emits light at a lower wavelength (red), which goes through the objective, passes through the dichroic, and is focused down to a spot surrounded by a pinhole. Light from other locations in the sample goes through the objective and dichroic, but is rejected by the pinhole (red dotted line).

this effect is also known as optical sectioning. Translating the sample relative to a fixed laser beam, or moving the laser beam relative to a fixed sample, allows the point-by-point construction of the full three-dimensional map of the sample itself, with resolution limited by the size of the excitation volume, itself limited by the diffraction limit of the illuminating light.

2.3.2 Instrumentation

The different implementations of a confocal microscope differ primarily in how the illumination volume is moved throughout the sample. The simplest method from an op-

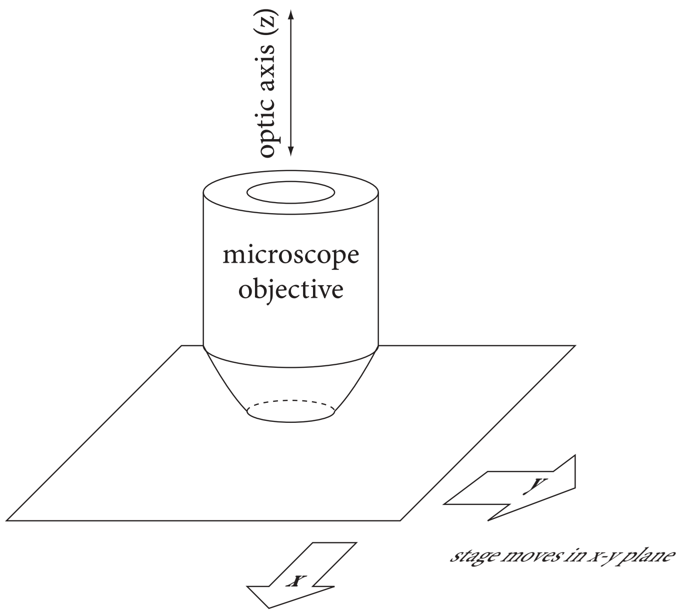

Figure 2.2: Confocal microscope instrumentation: stage-scanning, in which the optical train remains fixed and the stage is moved.

tical standpoint is to keep the optics fixed, and translate the sample (figure 2.2); modern piezo stages give precision and repeatability of several nanometers. Ideal from an image quality standpoint, as the optical path can be highly optimized and specific aberrations and distortions removed, sample translation is also the slowest; moving the piezo requires milliseconds, precluding the full-frame imaging at 25 frames/sec needed to achieve real-time speeds.

For higher speeds, the beam itself must be moved. Two galvanometer-driven mirrors can be used to scan the laser beam in x and y at up to a kilohertz, while maintaining beam quality (figure 2.3). While not quite fast enough to achieve real-time full-frame imaging, commercial confocal microscopes based on galvanometers can reach about ten full images a second, each with about a million pixels. Beam scanning is usually accomplished much like that of a television, by first quickly scanning a line horizontally, then shifting the beam

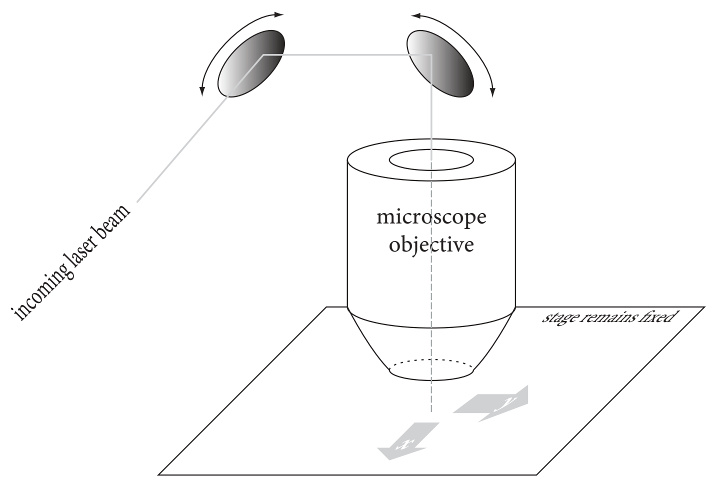

Figure 2.3: Confocal microscope instrumentation: beam scanning, with two moveable mirrors that move the beam itself.

(at the end of each horizontal scan) in the vertical direction, scanning another horizontal line, and so on. Replacing the galvanometer mirror that scans horizontally with an acousto-optical device (AOD) significantly increases speed (the galvanometer is fast enough to keep up with the vertical motion). However, the AOD severely degrades the quality of the beam, and image quality correspondingly suffers. AOD-based confocals are primarily useful where gathering data at high speed is more important than achieving high resolution, as is the case in dynamical situations with relatively large (i.e. greater than micron-sized) objects.

Another major approach to increasing beam-scanning speed is to split the main laser beam into thousands of smaller laser beams, parallelizing the illumination (figure 2.4). Each individual mini-beam then needs only to be moved a small amount in order for the total collection of beams to image an entire frame. This typically involves a Nipkow disk, where thousands of tiny microlenses are mounted in an otherwise opaque disk. These focus down to thousands of points, surrounded by thousands of tiny pinholes created in

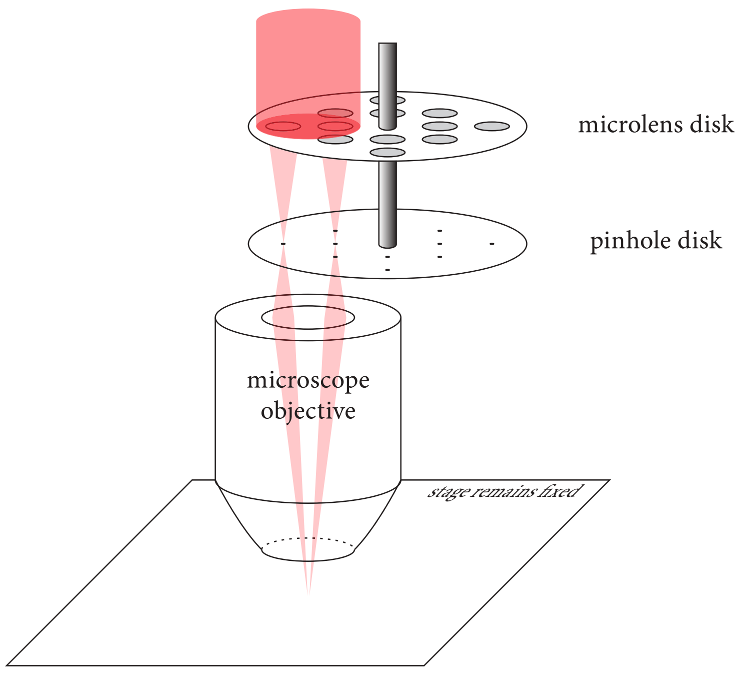

Figure 2.4: Confocal microscope instrumentation: Nipkow disk, where rotating disks of microlenses and pinholes parallelize the illumination beam.

another disk. The laser light is thus split and focused, and then the multiple tiny beams are focused onto the sample with a single objective lens. Light from the multiple illumination volumes comes back up first via the objective and then through the pinholes, then goes to the camera detector, where the thousands of mini-beams are simultaneously imaged. By spinning the disk and arranging the holes in a spiral pattern, full coverage of the frame can be achieved. The main advantage of this technique is that image quality can remain high (no AOD, for instance), and speed can be increased simply by spinning the disk faster. From an engineering standpoint, Nipkow disks are durable and easy to fabricate with existing technology; their major drawback is a total lack of flexibility: Nipkow disk systems are usually optimized for only one magnification, and after fabrication, the size of the pinholes cannot be changed to accommodate different conditions.

2.4 Techniques for improving imaging of nanoscale materials

2.4.1 4Pi Confocal

The biggest recent development in confocal microscopy has been the use of two objectives, focused on the same point, to collect light. The name 4Pi microscopy has been applied to this general technique, and is meant to evoke the idea that all of the light is collected from a sample simultaneously (i.e. the 4π steradians of a complete sphere); in reality, while most of the light is collected by the two objectives, they cannot image the whole sphere [124]. A full discussion of the principles and advances in 4Pi confocal microscopy is beyond the scope of this article (see [124, 115]); only a brief qualitative discussion to convey the underlying ideas behind the superior resolution of 4Pi confocal is included here.

A regular confocal rejects light coming from parts of the sample outside of the illumination volume by means of spatial filtering through a pinhole, but even if it is made arbitrarily small, the pinhole cannot localize the light coming from the sample to better than within the typical size of this region (i.e. the wavelength) because of diffraction. In addition, there is still a small contribution to the detected signal from light outside of the focal point, though that contribution decreases with greater distance from the focal point. Limitations to resolution therefore come from a combination of the finite size of the excitation volume in the sample, and the imperfect discrimination of the pinhole itself, both fundamental physical constraints inherent to the design of a confocal microscope; they cannot be overcome simply with better implementation of the same ideas. 4Pi confocal relies on coherent illumination or detection from both objectives simultaneously, effectively doubling amount

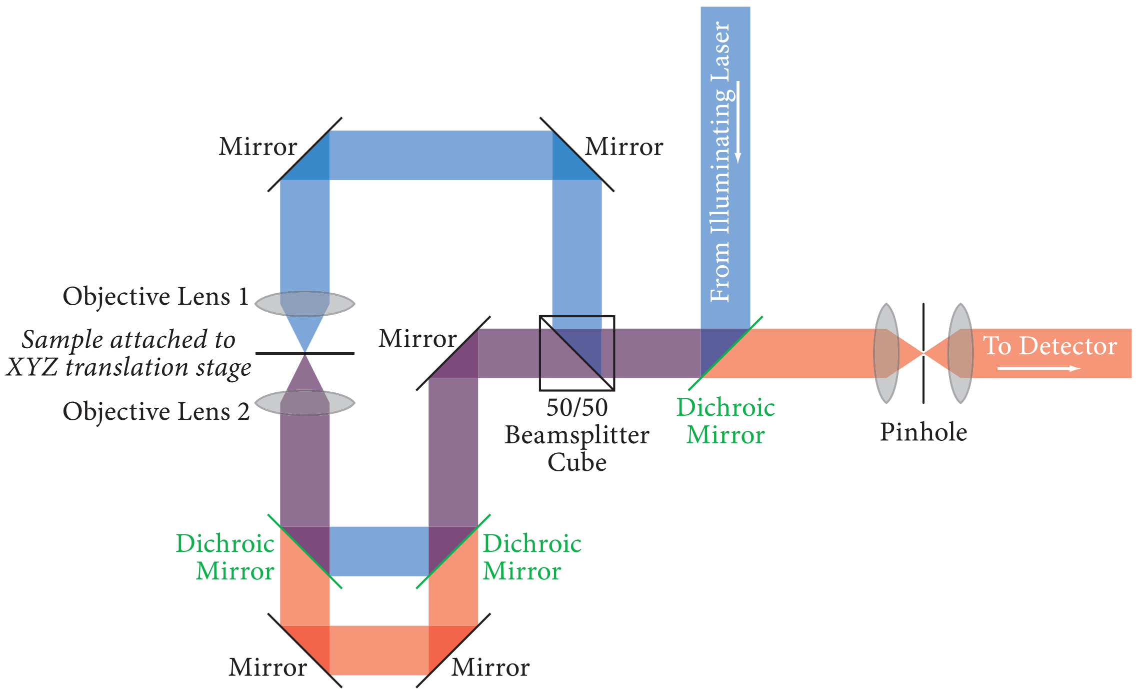

Figure 2.5: 4Pi-A configuration, with two illumination paths, but only one detection path.

of light involved and creating an interference pattern between the two beams. This allows a dramatic increase in axial resolution, often around five-fold, though lateral resolution is unchanged.

From an instrumentation standpoint, there are three different types of 4Pi confocal microscopes, A, B and C. In type-A 4Pi confocal, illumination beams are sent through both objectives and interfere in the sample, but the light coming out of only one objective is used for detection, as shown in figure 2.5. This is the earliest, and simplest, system, and has thus far been most widely used.

In type-B, illumination occurs through just one objective, but detection of interfering light from the sample comes through both objective lenses [126], as shown in figure 2.6; its theoretical optical properties are therefore identical to that of type-A 4Pi [238].

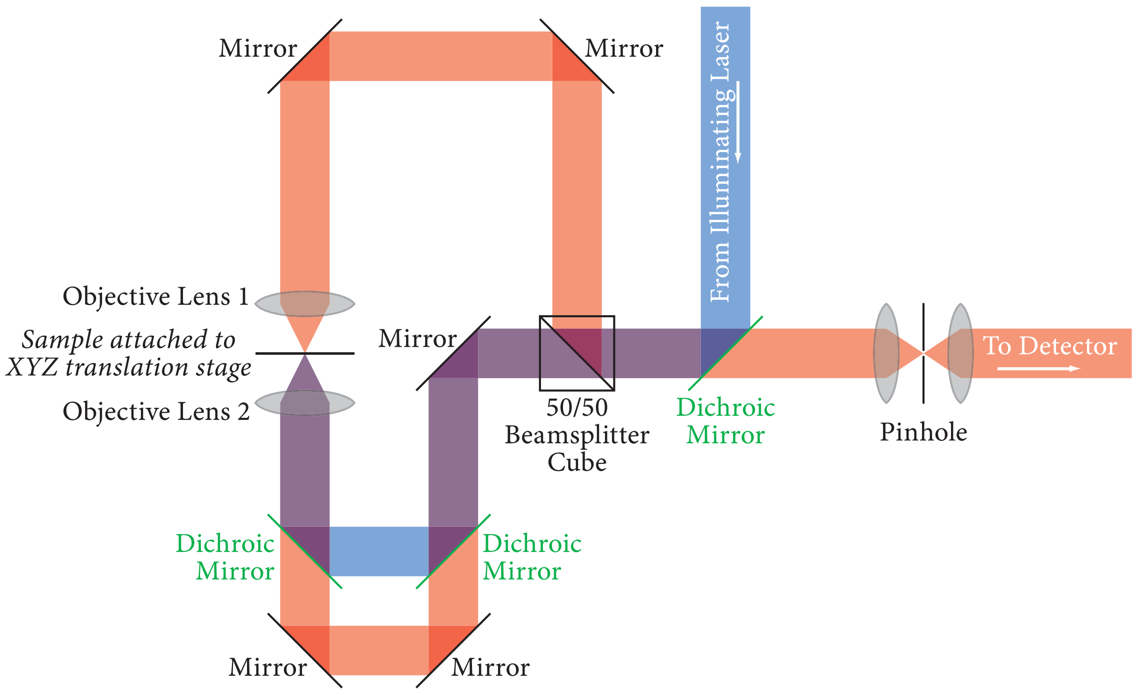

Figure 2.6: 4Pi-B configuration, with only one illumination path, but two detection paths.

In type-C, both illumination and detection are of interfering light in the sample volume, through both objectives [124], permitting even greater resolution, as shown in figure 2.7 [238].

Resolution is best understood in the context of the axial optical transfer function (OTF), also called the z-response function. Qualitatively, the OTF shows the contribution to the detected light from different depths in the sample (i.e. points along the optical axis). An ideal microscope would have only light from a single point in the focal plane contributing to the detected signal; in that case, the OTF would be delta function at the focus of the microscope objective. In a regular confocal, instead of a single delta function, the effects of finite-sized illumination volume and imperfect pinhole discrimination combine to smear out the delta function into a nearly gaussian OTF; with 633-nm HeNe laser illumi-

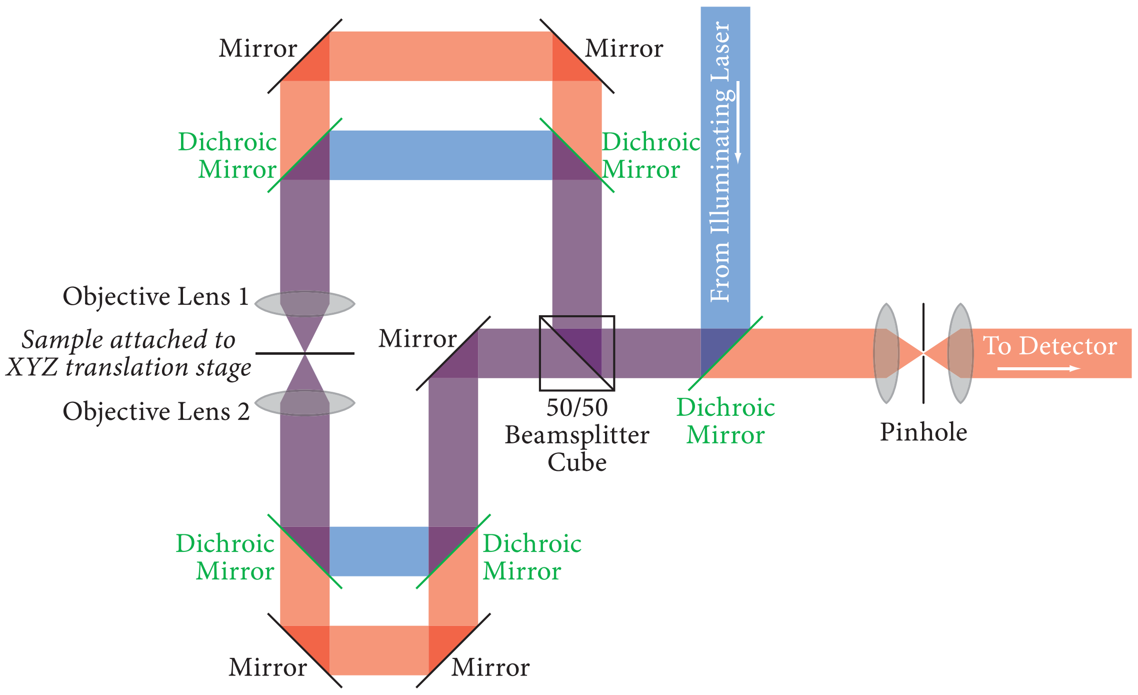

Figure 2.7: 4Pi-C configuration, with two illumination and detection paths.

nation, the OTF of a regular confocal has a full-width at half-maximum (FWHM) of 500 nm (theory and experiment) [238]. In 4Pi confocal microscopy, the counter propagating light waves of the same frequency and intensity that illuminate the sample create an interference pattern (a standing wave). Instead of a simply gaussian shape, the OTF now has one central peak and several so-called "side-lobes." The main advantage is that this central peak has a far narrower FWHM, theoretically calculated to be 130 nm for type-A (and thus for the optically equivalent type-B) and 95 nm for type-C, and measured at 140 nm and 95 nm, respectively [238]. The width of the central peak is independent of the relative phase between the two illuminating wavefronts (i.e. constructive or destructive interference are equivalent) [127], but nevertheless comes at the cost of having prominent side-lobes. That is, there is now a greater contribution to the light detected through the pinhole from some points farther away along the optic axis from the focal point than from some points closer, which creates artifacts. Almost all of the more recent technological developments in the 4Pi area have focused on optical “tricks” to eliminate the effects of those side bands: spatially filtering illuminating light beams with specifically-placed dark rings [28, 29] or illuminating with two photons [125, 259] to cut off the light that contributes mainly to side lobes, and computational modeling of an ideal microscope to reconstruct an “ideal” image from real data in a process known as deconvolution [259, 237, 239]. Such techniques have yielded a confocal with an effective point-spread function with a width as small as 127 nm for a type-A 4Pi confocal, with *no* significant contribution from the side lobes [28], allowing sub-10 nm distances between test objects to be measured with uncertainties less than a single nanometer [2].

Such high resolution may finally allow direct imaging of nanoscale structures, and Leica Microsystems has just introduced the first commercial 4Pi system, the TCS 4PI, in April 2004. Nonetheless, there still remain some limitations to current 4Pi technology. The number of optical elements to be aligned and controlled in a 4Pi setup is at least twice that of a regular confocal, and since the stage is usually scanned in a 4Pi setup, scanning speeds are much lower, requiring minutes to image a full frame. While fast enough to image stationary samples like fixed cells [240], or even slow-moving live ones [17], this is too slow to monitor most real-time dynamics at present, though scanning speed can be improved by using multiple beam scanning techniques in setups similar to the Nipkow disk, cutting imaging time down to seconds [86].

2.4.2 Other optical techniques to increase confocal resolution

Several other far-field optical techniques have achieved high resolution *without* spatial filtering by means of a pinhole. As they are neither confocal techniques, nor have been widely applied to systems relevant to nanotechnology, they will receive only brief mention.

Removing the pinholes and illuminating with an incoherent (non-laser) source in the 4Pi-A, 4Pi-B and 4-Pi-C geometries results in a setups known as I^{3}M, I^{2}M, and I^{5}M, respectively. [119, 117] Compared with 4Pi, these widefield techniques show an equivalent increase in axial resolution, though the lateral resolution is not as great. The main advantage is collection speed: light is collected from the entire imaging plane at once, as there is no beam to be scanned. The major drawback is the requirement for a large amount of computationally intense deconvolution to obtain images. Other techniques have used different geometries, objectives, mirrors, or multiple photons for illumination, but none thus far has achieved better resolution than 4Pi or I^{5}M, and have not been applied widely to systems of interest to nanotechnology; an excellent survey comparing the techniques is given in [118].

A couple of non-traditional optical techniques have also increased resolution in novel ways. Placing a solid hemispherical lens against the surface of the sample (figure 2.8) can improve resolution to a few times better than can be achieved with only a regular objective, with light collection efficiency improved five-fold. Interestingly, these improvements still persist even if the lens is slightly tilted, or there is a small air gap between the lens and the sample [183].

Also, common light detectors (PMT, APD, CCD) collect only intensity information, and can not measure phase directly. Interfering two beams, however, creates the a single output beam whose intensity is directly dependent on the phase difference of the two inter-

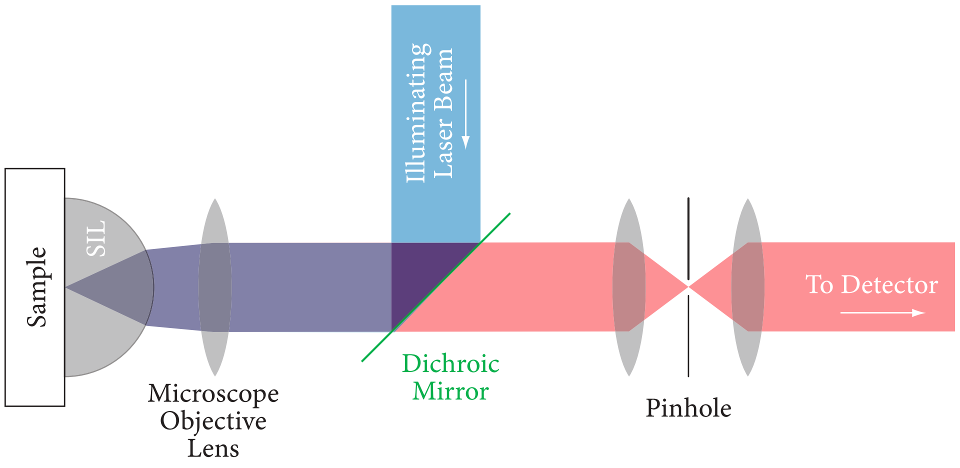

Figure 2.8: Other components to increase resolution: solid immersion lens, placed up against the sample.

fering beams. In practice, light can collected from two optical fibers (in place of the pinhole at the detector of the confocal), one along the optic axis, and one slightly displaced in the lateral direction. The signals from the two fibers are then interfered in a 2×1 optical fiber coupler (figure 2.9), which creates a single output beam whose intensity is measured. This interferometric technique is sensitive to single nanometer displacements on millisecond timescales [16].

Though not strictly an optical technique, another way to increase localization precision is to use objects that emit several colors. By detecting the different colors in separate channels, then combining the position data from different colors, the final position of the objects can be determined to an accuracy of better than 10 nm [150, 181]; the technique has also been used in 4Pi confocal setups [141] to achieve localization with single nanometer precision [236].

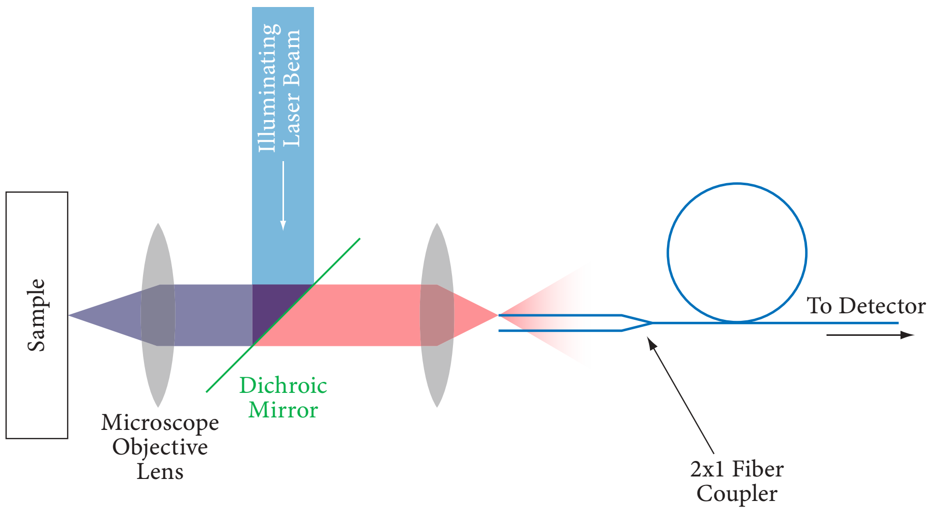

Figure 2.9: Other components to increase resolution: 2x1 optical coupler to interfere the light from the two fibers.

2.5 Applications to nanotechnology

As previously mentioned, the main contribution of confocal microscopy has not been to image nanoscale objects with light, but rather to use the capability to analyze the photons that have interacted with nanoscale structures, looking at their energies, temporal distribution, polarization, etc. As a result, the confocal can play a unique role in gathering information that can be obtained with no other technique. Many of the advances have come from the confocal's ability to characterize the properties of individual nanoscale objects, where previously only bulk ensemble averages could be measured. The discussion of how the confocal has been harnessed to gain information from nanoscale systems is organized by dimensionality of the system, in decreasing order.

2.5.1 Three-dimensional systems: Nanoemulsions

An emulsion is a mixture of two immiscible liquids: small droplets of the first liquid are dispersed in the second one, called the continuous phase. Such a mixture is intrinsically unstable, and droplets will coalesce unless a surfactant is added to the continuous phase. The surfactant helps to stabilize the interface between the two liquids by reducing the surface tension between them. Except in the case of microemulsions, emulsions are thermodynamically unstable [227]. Aging leads to a change in the size distribution of the droplets, and occurs via two mechanisms: coalescence, where smaller droplets combine to form larger ones, and Ostwald ripening, where larger droplets grow at the expense of smaller ones via diffusion of molecules through the continuous phase.

Confocal observation of the changing fluorescence intensity of single nanodroplets (as small as 50 nm) flowing through a capillary tube permitted investigation into the fundamental process of emulsion coarsening in a way not accessible to bulk measurement: the rate of drop growth could be measured for single nanoscale drops in the confocal, not a statistical average for the emulsion as a whole. The work demonstrated that Ostwald ripening and coalescence were occurring at different stages of emulsion coarsening [229], and dye diffusion was further studied by looking at fluorescence dynamics in mixtures of undyed and dyed emulsions [228]. Potential applications as isolated containers make nanoemulsions particularly interesting: encapsulation of a water-insoluble drug compound into the oil droplets of a nanoemulsion may allow controlled delivery of a substance to a designated target area in the body. Confocal has been used to monitor the uptake of dyed diblock copolymer nanoemulsions into cells [185], and the targeted delivery to cell organelles, each dyed a different color [232].

2.5.2 Three-dimensional systems: Nanocapsules

Nanocontainers can also be created by coating colloidal spheres, nanocrystals or other templates, then dissolving out the core to leave a rigid hollow shell, contrasting the "soft" surface of an emulsion. The templates have designable properties and are typically micron-sized; the term nanocapsule refers to the controllable organization of wall layers: starting with a charged core, layers of alternating charged polyelectrolytes can deposited with nanometer thicknesses in a technique known as layer-by-layer (LBL) self-assembly. The chemistry of the polyelectrolytes can be varied significantly, and they can also serve as hosts for a variety of other materials, particularly ones with interesting photonic properties [68].

Confocal characterization has been used for real-time, three-dimensional imaging of the growing nanocapsules and subsequent core dissolution in a number of systems: single-walled containers templated around a cadmium carbonate core [258], concentric sphere-in-sphere nanocapsules more mechanically robust than their single-wall counterparts [67], nanocapsules with silver ions in the walls for designable catalysis [7], biocompatible nanocapsules labeled with luminescent CdTe nanocrystals [101], and nanocapsules constructed by LBL assembly of dendrimers, large branching macromolecules with fractal structure [142].

Nanocapsules can also provide a controllable local environment for investigating nanoscale chemical reactions. LBL assembly and tuning the external environment allow very fine control over the permeability of the capsule to small molecules and ions, so that large enzyme molecules can be held inside nanocapsules whose walls are permeable to a fluorescent substrate [174]. By monitoring the fluorescence changes in real-time, the confocal gives a unique, quantitative, single-molecule view on enzyme activity [60], where previous techniques have only allowed bulk measurement of average activity; without the confocality, there is no way to isolate a single nanocapsule for study. Similarly, a pH gradient can be created between the inside and outside of a nanocapsule, and the confocal has monitored selective pH-induced precipitation of iron oxide nanocrystals inside single nanocapsules [211].

2.5.3 Other three-dimensional nanostructures

The confocal's ability to spatially resolve spectral information in three dimensions has been used to characterize the nanostructure of other heterogenous materials. These systems may be constructed of different phases, such as the low-temperature Shpol'skii systems, where confocal spectroscopy was used to quantify preferential alignment of individual aromatic hydrocarbons molecules in a host of alkanes [30]. The confocal can also image the negative space in a porous material (e.g. nano-scale holes or pores) filled with dye, such as the spaces between layers of hydrotalcite-like compounds, including anionic clays and layered double hydroxides [152].

2.5.4 Two-dimensional systems: Ferroelectric Thin Films

Even in the absence of an external electric field, ferroelectric materials exhibit an electric dipole below a certain transition temperature, and they are the electric-field analog of a ferromagnet [143]. Above this transition temperature, the net electric dipole is no longer present, but ferroelectric materials still have a nonlinear dielectric constant useful for creating elements (such as capacitors and phase shifters) in microwave integrated circuits for insertion into wireless and satellite communications devices [280]. These materials, often based on the barium/strontium titanate (BST) perovskite structure, have different properties whether in bulk or in a thin film.

Here, the confocal microscope has not contributed as a light-based imaging device in the traditional sense; rather, its capabilities to place a probing electric field precisely have been leveraged in a unique way to probe the physics of ferroelectrics [130, 131, 132]. First, pumping a BST thin film with a micron-scale capacitor aligns the film's molecular dipoles with an external electric field. After a controllable delay time, during which these dipoles begin to relax, the confocal microscope places a diffraction-limited spot of light at a particular location on the surface of a thin film. The electric field component of the illuminating laser beam serves as a sensitive probe, with emitted light having a different polarization or intensity as a result of its interaction with the ferroelectric material. Position and delay time are varied, with submicron and picosecond control. Changes in the emitted light from BST films grown under different pressures of oxygen suggest that reorientation of polarity on the nanoscale level is ultimately responsible for changes in ferroelectric behavior [133].

2.5.5 Two-dimensional systems: Nanopores, Nanoholes and Nanomembranes

The confocal has also been used to characterize the spatial distribution of negative space in two-dimensional systems, such as nanopores in titania thin film solar cell electrodes [272], luminescent conjugated polymers in nanoporous alumina [209], and pieces of fluorescently-labeled cell membrane stretched over nanoscale holes in silicon nitride [89].

This last technique is favored for AFM and other scanning-probe investigations of membranes, as isolated suspended membrane patches have improved stability and access relative to whole cells, and nanoholes are easily created with standard photolithography techniques. While SEM can characterize coverage of a cell membrane suspended over a nanohole, it is a two-dimensional technique that cannot discriminate between a suspended cell membrane and a pile of cell debris sitting on the silicon nitride surface. The three-dimensional sectioning ability of the confocal plays a critical role here: monitoring the height-dependence of fluorescence intensity yields a depth profile of fluorescent cell material, quickly distinguishing freely suspended cell membranes as small as 50 nm on a side [89].

2.5.6 One-dimensional systems: Carbon Nanotubes

The bulk processes (e.g. carbon vapor deposition) that create carbon nanotubes typically yield a mixture of diameters, lengths, and structures, each with different physical properties. A major goal of nanoengineering is narrowing the distribution of sizes and structures to create materials with better-defined properties. Although the typical nanotube diameter of a few nanometers is well below the threshold of optical visibility, differences in nanotube structure measurably change Raman spectral profiles. This confers upon the confocal a singularly important role in nanotube characterization, as its combination of spatial and spectral resolving capacity allow it to characterize individual nanotubes at speeds far greater than those accessible to other techniques. Characterizable attributes include nanotube diameter, (n,m) chirality [140, 14], and electrical conductivity [306], as well as distinguishing single-walled from multiple-walled [84], and free-standing from bundled nanotubes on insulating and conducting surfaces [230]. Confocal studies have been combined with AFM to better characterize the diameter-dependence of Raman peaks [306], and with HRTEM to precisely correlate atomic structure with spectral properties of the same nanotubes [137]. The confocal's high-speed, single nanotube characterization ability has also been harnessed as a screen in parallel microarray applications, including carbon vapor deposition on top of a microarray of different liquid catalysts to optimize synthesis [56], and a microarray constructed by linking DNA covalently to nanotubes to test sequence-specific binding interactions [123].

2.5.7 One-dimensional systems: Nanowires

Confocal Raman spectroscopy has also characterized other one-dimensional systems. Raman spectra of silicon nanowires (5 to 15 nm) collected at *low* laser power match those of bulk silicon. But as power of the confocal's illuminating laser was increased, the Raman peaks shifted from those of bulk silicon in a way not consistent with quantum confinement effects, but rather suggesting that the laser is heating the wire itself [116].

2.5.8 Zero-dimensional systems: Luminescent Nanocrystals and Quantum Dots

Nanocrystals are crystals ranging in size from nanometers to tens of nanometers, large enough so that single atoms do not drive their dynamics, while still small enough for their electronic and optical properties to be governed by quantum mechanical effects. Under confocal microscopy, where their size precludes resolution of their features, they effectively behave as zero-dimensional points. Of the common synonyms, including nanoparticle' and quantum dot,' only nanocrystal will be used here. The confocal has been used in two broad areas: spectrally characterizing individual nanocrystals, where confocality is required to isolate individual particles, and localizing nanocrystals as point tracers in other systems, utilizing the capability of three-dimensional, real-time localization.

Recent examples of confocal characterization of the energy spectra of individual nanocrystals include cryogenic (20 K) imaging of single 7-nm ZnS nanocrystals [273], and characterization of the optical extinction properties of nanocrystals created by conventional nanosphere lithography [122]. Quantifying the temporal dynamics of the intermittent fluorescence (blinking) typical of nanocrystals yields information on their electronic structure, and the fast on-and-off fluorescence can be captured with high-speed optical detectors, like the photomultiplier tube (PMT) or the avalanche photo-diode (APD), attached to the confocal. Studies of individual CdSe nanocrystals overcoated with ZnS have shown that several energy levels may drive the optical behavior [147, 148], with similar results found for InP nanocrystals [149]. Measuring energy spectra over time has quantified changes due to oxidation of nanocrystals in air (versus no change in pure nitrogen) [281], and the size and surface-property dependence of the behavior of silicon quantum dots, both porous [178] and crystalline stabilized with an organic monolayer [87]. By splitting the light coming back from the sample, sending half to an APD and the remainder through a prism onto a CCD, high resolution time and spectral data can be collected simultaneously. This analysis, in combination with TEM of the same individual nanocrystals to correlate optical properties with atomic structure, has shown that a single crystalline domain is not required to achieve fluorescence [144]. Finally, to characterize the heavy dependence of fluorescence behavior on nearby conductors, confocal microscopy imaged a fluorescent dye attached to both bulk gold and gold nanocrystals on the same substrate. While the dye attached to bulk gold was quenched, since the energy absorbed by the fluorophore gets transferred to the sea of electrons in the metal, dyes attached to nanocrystals remained bright, as there is no bulk into which to transfer electrons [157].

The confocal has also been used to localize nanocrystals embedded in other systems. This has aided synthesis of organic nanocrystals grown in inorganic sol-gel coatings, with confocal characterization of their size and distribution allowing systematic exploration of the phase space of the main synthesis parameters [231]. Embedding nanocrystals within polyelectrolyte layers in LBL assembly allows nanometer control of thin-film coatings that can be applied to three-dimensional objects of complex geometry, whose luminescent properties are then determined by the nanocrystals. Three-dimensional sectioning in the confocal has been crucial to characterizing these coatings, which have been applied to cylindrical optical fibers [65] and spherical latex colloidal particles [265]. In addition, the confocal has been used to characterize the three-dimensional structure of micron-sized domains of nanocrystals, including silver nanocrystals embedded between two layers in a thin film and coalesced by irradiation with a high-intensity laser beam to create planar diffractive and refractive micro-optics [177], and silicon nanocrystals patterned onto surfaces by directing a stream of silicon atoms through a mask for nanofabrication of light sources from all silicon with pre-existing tools from the electronics industry [153].

In addition to these static applications, the real-time imaging capability of the confocal is useful for monitoring nanocrystal dynamics at higher speeds. Confocal microscopy has been combined with flow cytometry to image and count fluorescent nanoparticles in a fluid flow, yielding real-time information on their concentration [88]. Fluorescent colloidal nanospheres have been coated on one side with gold, yielding an opaque hemispherical metal coating that appears dark, and floated on the surface of a liquid. Light intensity levels of individual nanospheres can be correlated with angular orientation, allowing real-time imaging of Brownian rotational diffusion to study molecular interactions, particularly the preferential attraction of fluid molecules toward one hemisphere over the other [61]. Metal oxide nanocrystals and their halogen adducts have been shown to kill bacteria, and a combination of confocal microscopy and electrostatic measurements has demonstrated that the particles and bacteria attracted each other on account of their electric charge, shedding light on nanoscale electrostatics in solution [262]. Finally, the active transport of single nanocrystals by a dynein motor protein along microtubules in live cells has been imaged in real time with a Nipkow disk confocal; interestingly, contrasting the blinking typical of nanocrystals in free solution, these nanocrystals in live cells have nearly constant intensity [186].

2.5.9 Zero-dimensional systems: Viruses

Far-field confocal microscopy has been combined with a near-field scanning ion microscopy in water, with a micropipette carrying an ion current tens of nanometers from an optically invisible cell membrane surface to map topography. This technique has been used to image virus-like particles, nanospheres composed of a few hundred viral proteins enclosing DNA, and their absorption into cells. The biological motivation is to understand how viruses infect living cells, but more generally, the investigation demonstrates confocal three-dimensional, real-time localization of a nanoscale object relative to an optically invisible surface [110].

By manipulating the genetic code of viruses, proteins on their surfaces can be modified to achieve desired properties in a technique called phage display, which early on was used to optimize highly-specific binding of viruses to a variety of semiconductor surfaces [297]. More recently, phage display has been used to control the morphology of calcium carbonate crystals precipitated from solution in the presence of viruses that template the crystals, in an effort to better understand biomineralization. While scanning electron microscopy can image and characterize the inorganic calcium carbonate after growth was stopped, confocal microscopy allowed three-dimensional localization of the fluorescently-labeled viruses relative to the growing crystals [159].

2.5.10 Zero-dimensional systems: Single Molecule Studies

Confocal studies of single molecules fall into two broad groups: spectral characterization of single molecules, and localization of fluorescent molecules as tags. Static single molecule systems characterized include individual dye molecules over a range of temperatures from liquid helium to room temperature [250], and optimized mutations of the fluorescent protein GFP in various three-dimensional substrates [59]; dynamically, time-correlation spectroscopy in the confocal has been used to measure solution concentrations down to 10^{-15} M [187]. While confocality may not be strictly necessary for these studies, increased resolution helps to isolate individual molecules. The confocal has also been used for three-dimensional localization of single fluorescent dye molecules attached as tags to another object of interest, such as single molecules inside a living cell [39], or correlating emission spectra of molecules on a mica substrate with AFM data to develop a way measure topography optically [145].

2.6 Summary and Future Perspectives

Confocal microscopy extends the characterization of ever smaller objects with optical microscopy to the nanometer scale. By using a pinhole to restrict detected light to only that coming from the focus of the microscope objective, the confocal allows three-dimensional sectioning and localization, often at rapid rates with proper scanning techniques. Spatially resolved spectroscopy has allowed investigation of a broad variety of systems of interest to nanotechnology, providing information accessible to no other technique.

Further developments may be anticipated in several areas. 4Pi-C confocal has already achieved resolution in the sub-100 nm range that defines "nanoscale;" it is possible that novel optical techniques will improve this even further. Other evolutionary engineering improvements will surely increase speed, perhaps allowing real-time 4Pi imaging comparable to regular confocal by using beam parallelization in the same spirit as the Nipkow disk.

The capabilities of the confocal to answer new kinds of questions are also being developed rapidly, in no small part due to the low cost and relative ease of construction of off-the-shelf laboratory optical components used to build instrumentation ancillary to the confocal. A large fraction of the applications described in this review utilized some form of home-built hardware, a trend which will surely continue. Spectroscopy will likely become faster, allowing the characterization of the changes in spectra on shorter timescales, with the ultimate goal of studying changes in single molecules, a new field with some early applications described. Better understanding of the behavior of isolated nanoscale objects will use the confocal microscope's three-dimensional, real-time localization capability to study not only the dynamics of single particles, but also the behavior of systems of thousands, or perhaps even millions, of particles with controllable interactions. Clearly, the unique capabilities of the confocal microscope will ensure its contribution to the development of nanotechnology for the foreseeable future.

2.7 Acknowledgements

I thank Itai Cohen, Laura Kaufman, Nicolas Tsapis and Eric Dufresne for comments on the manuscript.

This chapter has been published as the first chapter in *Handbook of Microscopy for Nanotechnology*, edited by Nan Yao and Zhong Lin Wang [170].

Chapter 3

Spinodal and Binodal Decomposition: A Primer

This chapter gives a self-contained, qualitative derivation from first-principles for the spinodal and binodal decomposition in liquid-gas phase separation. Although I summarize explanations and derivations presented elsewhere, I include the information here because I have not found a concise summary where I could follow all of the steps, and because spinodal decomposition underpins much of the work presented in this thesis.

3.1 Coexistence Minimizes Free Energy

To understand why a system in equilibrium may undergo phase separation, it is important to consider that these systems seek to minimize their Helmholtz free energy:

$$F = U - TS \tag{3.1}$$

where U represents the bonding energy between the constituents (e.g. colloids, particles, molecules) in a system, T is the temperature, and S is their entropy. At high temperature, entropy dominates, and a typically homogeneous disordered state is favored. At zero temperature, bonding energies will take over, often favoring much greater order (for example, in atomic or molecular crystals that form many everyday solids, like metals). At intermediate temperatures, however, the two terms can be of comparable magnitude, and in many cases, the system can best balance the contributions from U and S in minimizing F by splitting into two phases, each with a separate density, but in equilibrium with the other [139, 191].

We can show this explicitly in a simple system with two types of constituents, labeled A and B. The number of constituents of type A is represented by N_A; the number of type B, N_B. The total number of constituents is conserved, and fixed to a total number N, so that:

$$N = N_\mathrm{A} + N_\mathrm{B} \tag{3.2}$$

N can be thought of as the total number of discrete sites on a lattice, or possible positions; for a liquid or gas, these lattice sites do not fall on a periodic grid. The volume fractions of the two constituents are then defined:

$$\phi_\mathrm{A} = \frac{N_\mathrm{A}}{N} \tag{3.3}$$

$$\phi_\mathrm{B} = \frac{N_\mathrm{B}}{N} \tag{3.4}$$

$$\phi_\mathrm{A} + \phi_\mathrm{B} = 1 \tag{3.5}$$

3.1.1 Entropy of Mixing

The total number of configurations possible in this system is the number of ways in which N_A objects can occur in N positions (i. e. N choose N_A):

$$\Omega(N_\mathrm{A}, N_\mathrm{B}) = \frac{N!}{(N - N_\mathrm{A})!N_\mathrm{A}!} = \frac{N!}{N_\mathrm{B}!N_\mathrm{A}!} \tag{3.6}$$

The configurational entropy is the logarithm of the number of states, multiplied by Boltzmann's constant k_B:

$$\begin{aligned} S &= k_\mathrm{B} \log \Omega && (3.7) \\ &= k_\mathrm{B}(\log N! - \log N_\mathrm{B}! - \log N_\mathrm{A}!) && (3.8) \end{aligned}$$

We next apply the Stirling approximation:

$$\log N! \approx N \log N - N \tag{3.9}$$

The entropy of mixing S_m then becomes:

$$\begin{aligned} S_\mathrm{m} &\approx k_\mathrm{B}(N \log N - N_\mathrm{A} \log N_\mathrm{A} - N_\mathrm{B} \log N_\mathrm{B} - N + N_\mathrm{A} + N_\mathrm{B}) && (3.10) \\ &= k_\mathrm{B}(N \log N - N_\mathrm{A} \log N_\mathrm{A} - N_\mathrm{B} \log N_\mathrm{B}) && (3.11) \\ &= k_\mathrm{B}(N_\mathrm{A} \log N + N_\mathrm{B} \log N - N_\mathrm{A} \log N_\mathrm{A} - N_\mathrm{B} \log N_\mathrm{B}) && (3.12) \\ &= -k_\mathrm{B} \left(N_\mathrm{A} \log \frac{N_\mathrm{A}}{N} + N_\mathrm{B} \log \frac{N_\mathrm{B}}{N} \right) && (3.13) \\ &= -k_\mathrm{B} N \left(\frac{N_\mathrm{A}}{N} \log \frac{N_\mathrm{A}}{N} + \frac{N_\mathrm{B}}{N} \log \frac{N_\mathrm{B}}{N} \right) && (3.14) \\ &= -k_\mathrm{B} N \left(\phi_\mathrm{A} \log \phi_\mathrm{A} + \phi_\mathrm{B} \log \phi_\mathrm{B} \right) && (3.15) \end{aligned}$$

where the conservation relation in equation 3.2 has been applied in several places.

3.1.2 Energy of Mixing

To understand the energy involved in the mixing process, we define three constants to characterize the bonding between the different constituents: $U_{\mathrm{AA}}/k_{\mathrm{B}}T$ is the bonding energy between two constituents of type A; $U_{\mathrm{BB}}/k_{\mathrm{B}}T$ is the bonding energy between two constituents of type B, and $U_{\mathrm{AB}}/k_{\mathrm{B}}T$ is the bonding energy between one constituent of type A and one of type B. We treat the constituents in the mean-field approximation, so that every constituent is considered to have a coordination number of z (i.e. z nearest-neighbors on the lattice). In this case, each constituent has $z\phi_{\mathrm{A}}$ neighbors of type A, and $z\phi_{\mathrm{B}}$ neighbors of type B.

In an unmixed state, where all A constituents are individually segregated, as are those of type B, the total energy due to bonding is:

$$U_{\mathrm{unmixed}} = \frac{zN}{2}\left(\phi_{\mathrm{A}}U_{\mathrm{AA}} + \phi_{\mathrm{B}}U_{\mathrm{BB}} + O\left[\frac{1}{L}\right]\right) \tag{3.16}$$

where the factor of $\frac{1}{2}$ is included to cut out the double counting. The final term is a surface energy due to the interface—of order $\frac{1}{L}$, which vanishes for large systems—so we subsequently neglect this term.

When the constituents are mixed, the bonding energy has three terms, representing $A-A$, $B-B$ and $A-B$ terms. The contribution from $A-A$ bonding is the product of the fraction of A constituents ϕ_{A}, the energy per bond U_{AA}, the number A neighbors $z\phi_{\mathrm{A}}$ and $1/2$ to cut out the double counting,

$$N\phi_{\mathrm{A}} \times U_{\mathrm{AA}} \times z\phi_{\mathrm{A}} \times \frac{1}{2} \tag{3.17}$$

The contribution from $B-B$ bonding is analogous, and that from $A-B$ bonding follows a similar logic:

$$N\phi_{\rm A} \times U_{\rm AB} \times z\phi_{\rm B} \tag{3.18}$$

but now there is no double counting, so there is no need for the factor of $1/2$. The total energy from bonding in the fully mixed state is:

$$U_{\rm mixed} = \frac{zN}{2}\left(\phi_{\rm A}^2 U_{\rm AA} + \phi_{\rm B}^2 U_{\rm BB} + 2\phi_{\rm A}\phi_{\rm B} U_{\rm AB}\right) \tag{3.19}$$

The energy of mixing is the difference between the bonding energy in the fully mixed, and fully unmixed, states:

$$\begin{aligned}
U_{\rm m} &= U_{\rm mixed} - U_{\rm unmixed} && (3.20)\\
&= \frac{zN}{2}\left(\phi_{\rm A}^2 U_{\rm AA} + \phi_{\rm B}^2 U_{\rm BB} + 2\phi_{\rm A}\phi_{\rm B} U_{\rm AB}\right) - \frac{z}{2k_{\rm B}T}\left(\phi_{\rm A} U_{\rm AA} + \phi_{\rm B} U_{\rm BB}\right) && (3.21)\\
&= \frac{zN}{2}\left((\phi_{\rm A}^2 - \phi_{\rm A})U_{\rm AA} + (\phi_{\rm B}^2 - \phi_{\rm B})U_{\rm BB} + 2\phi_{\rm A}\phi_{\rm B} U_{\rm AB}\right) && (3.22)\\
&= \frac{zN}{2}\left(\phi_{\rm A}(\phi_{\rm A} - 1)U_{\rm AA} + \phi_{\rm B}(\phi_{\rm B} - 1)U_{\rm BB} + 2\phi_{\rm A}\phi_{\rm B} U_{\rm AB}\right) && (3.23)\\
&= \frac{zN}{2}\left(-\phi_{\rm A}\phi_{\rm B} U_{\rm AA} - \phi_{\rm B}\phi_{\rm A} U_{\rm BB} + 2\phi_{\rm A}\phi_{\rm B} U_{\rm AB}\right) && (3.24)\\
&= \frac{zN}{2}\left(2U_{\rm AB} - U_{\rm AA} - U_{\rm BB}\right)\phi_{\rm A}\phi_{\rm B} && (3.25)
\end{aligned}$$

where we have used the conservation relation in equation 3.5. Following the notation in [139], we define the energy constant χ,

$$\chi \equiv \frac{z}{2k_{\rm B}T}\left(2U_{\rm AB} - U_{\rm AA} - U_{\rm BB}\right) \tag{3.26}$$

so that the energy of mixing, in units of $k_{\rm B}T$, can be expressed:

$$\frac{U_{\rm m}}{k_{\rm B}T} \equiv \chi N \phi_{\rm A}\phi_{\rm B} \tag{3.27}$$

3.1.3 Free Energy of Mixing

The free energy of mixing, per particle, then, is:

$$\frac{F_{\mathrm{m}}}{Nk_{\mathrm{B}}T} = \frac{U_{\mathrm{m}} - TS_{\mathrm{m}}}{Nk_{\mathrm{B}}T} \tag{3.28}$$

$$= \chi\phi_{\mathrm{A}}\phi_{\mathrm{B}} + \phi_{\mathrm{A}}\log\phi_{\mathrm{A}} + \phi_{\mathrm{B}}\log\phi_{\mathrm{B}} \tag{3.29}$$

A plot of this free energy for a system where N_{A}=N_{B} is shown, for several different values of χ, in figure 3.1. For low values of $\chi \leq 2$, there is a single minimum of the u-shaped free energy curve, at $\phi_{\mathrm{A}} = \phi_{\mathrm{B}} = 0.5$. For $\chi > 2$, however, the free-energy curve becomes w-shaped, with two distinct minima; this represents a phase-separation, where the balance of energy and entropy is minimized by the system splitting into two distinct parts, each with significantly different values of ϕ_{A}. These values can be determined with a graphical construction, looking at where the free-energy curve intersects a common tangent line connecting the two minima. The equilibrium state is a coexistence between two phases, with these two values of ϕ_{A}. The behavior of the system, particularly whether it separates, can be seen as a function of the temperature, and the various bond energies contained in χ. In general, there will not be an enforced equality between the number of constituents, so that $N_{\mathrm{A}} \neq N_{\mathrm{B}}$. This tilts the free energy curve, so that the two minima no longer occur at the same value of the free energy, as shown in figure 3.2.

3.2 Helmholtz Free Energy and the Grand Potential

As described in section 3.1, the Helmholtz free energy F in general follows the form of a tilted w-shaped curve, where the two points connected by a common tangent determine

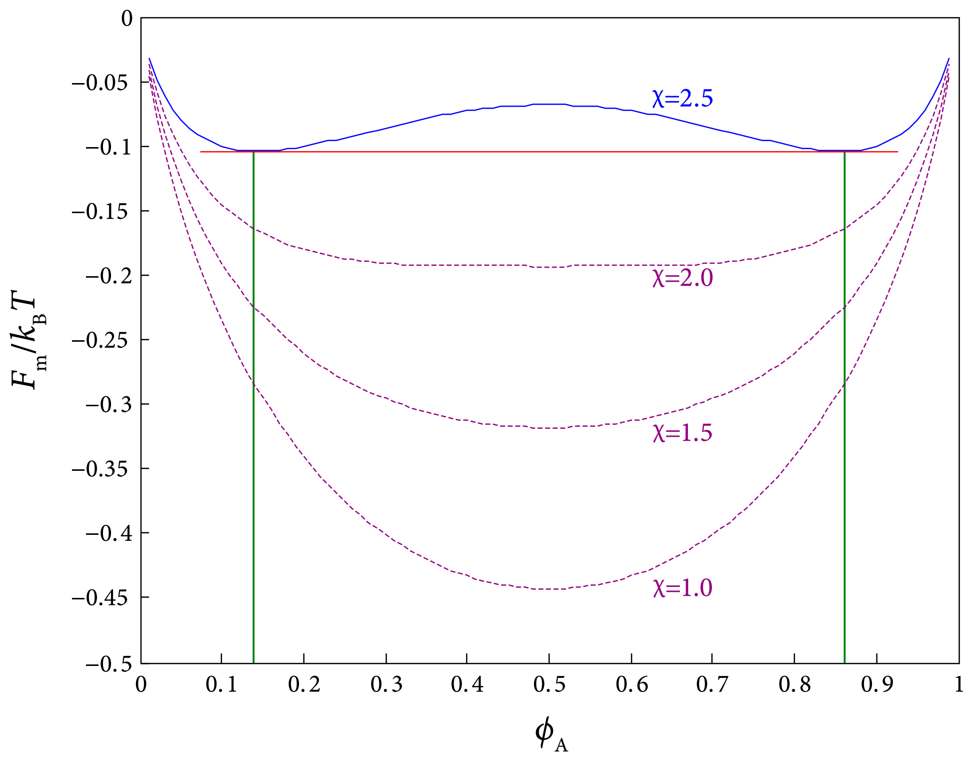

Figure 3.1: Free energy of mixing $F_{\mathrm{m}}/k_{\mathrm{B}}T$ for a system with for $N_{\mathrm{A}} = N_{\mathrm{B}}$, plotted for several values of χ. For low values of $\chi \leq 2$ (purple dotted lines), there is a single minimum of the u-shaped free energy curve, at $\phi_{\mathrm{A}} = \phi_{\mathrm{B}} = 0.5$. For $\chi > 2$, however, the free-energy curve becomes w-shaped (shown in solid blue line), with two distinct minima. This represents a phase-separation, where the balance of energy and entropy is minimized by the system splitting into two distinct parts, each with significantly different values of ϕ_{A}. These values can be determined with a graphical construction, looking at where the free-energy curve intersects a common tangent line (solid red line) connecting the two minima, which are marked with green vertical reference lines. The equilibrium state is a coexistence between two phases, with these two values of ϕ_{A}.

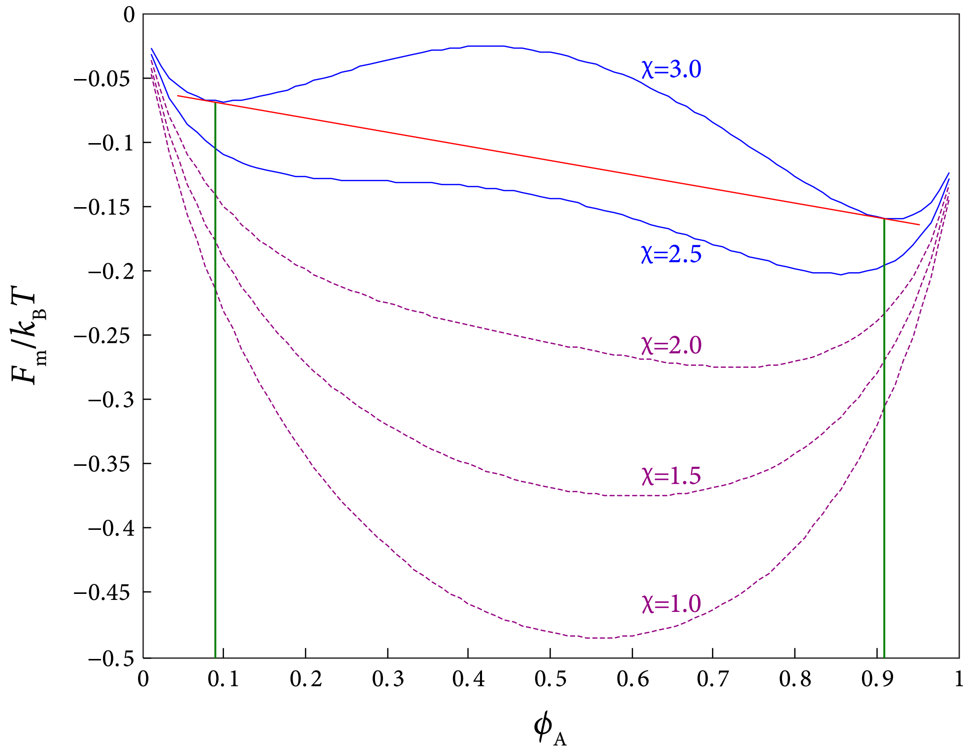

Figure 3.2: Free energy of mixing $F_{\mathrm{m}}/k_{\mathrm{B}}T$ for a system where $N_{\mathrm{B}} = 0.9N_{\mathrm{A}}$, plotted for several values of χ. For low values of $\chi \leq 2$ (purple dotted lines), there is a single minimum of the u-shaped free energy curve, but this no longer occurs at $\phi_{\mathrm{A}} = 0.5$. For $\chi > 2$, the free-energy curve becomes w-shaped (shown in solid blue line) as before, with two distinct minima. These minima, however, are no longer at the same free energy. Nevertheless, the volume fractions that the system splits into can be determined with the same graphical construction, looking at where the free-energy curve intersects a common tangent line (solid red line) connecting the two minima, which are marked with green vertical reference lines.

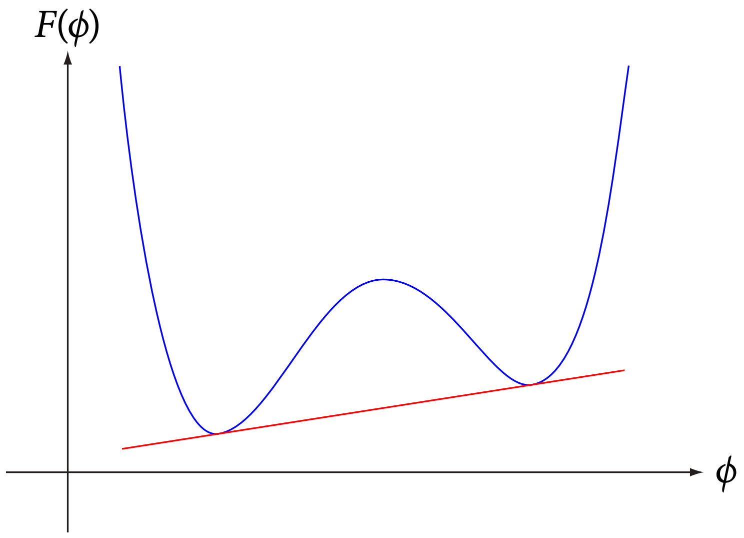

Figure 3.3: Schematic example of $F(\phi)$, the Helmholtz free energy as a function of volume fraction ϕ of one of the constituents in a phase-separating system, marked with a solid blue line. The two points connected by a common tangent (red line) determine the volume fractions of the coexisting phases

the volume fractions of the coexisting phases. An example is shown in figure 3.3, which is comparable to the higher-χ curves shown in figure 3.2.

In this section, we focus on the grand potential, which subtracts off the chemical potential from the Helmholtz free energy [194]:

$$G = F - \mu\phi \tag{3.30}$$

In this representation, the two minima occur at the same value of energy, as shown when the Helmholtz free energy from figure 3.3 is converted to a grand potential, plotted in figure 3.4 with a blue curve. Here, we have recentered the horizontal ϕ axis so that $\phi = 0$ corresponds to $\phi_{\mathrm{A}} = \phi_{\mathrm{B}} = 1/2$.

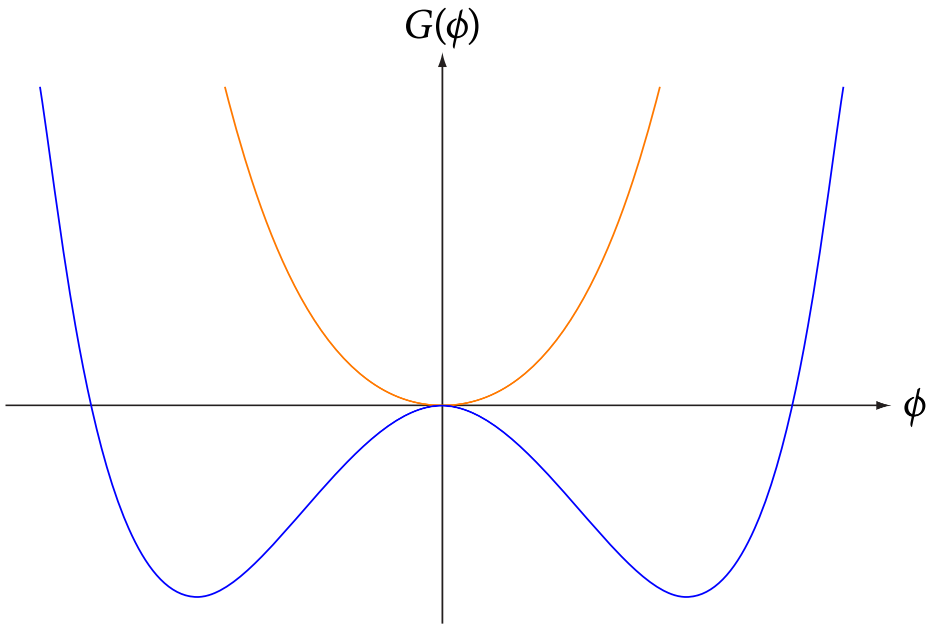

Figure 3.4: Grand potential as a function of ϕ. Expanding this curve as a polynomial, in a ϕ^4 mean-field treatment, yields $G(\phi) = a\phi^2 + b\phi^4$. The orange curve illustrates the case for $a > 0$, where a single minimum occurs at $\langle\phi\rangle = 0$. The blue curve illustrates the case for $a < 0$: the system splits into two phases, at $\phi_0 = \pm\sqrt{-a/2b}$, where $\delta G(\phi)/\delta\phi = 0$, which have equal energy. The Helmholtz free energy corresponding to this blue curve is plotted in figure 3.3.

3.3 Cahn-Hilliard Theory

We now explore this grand potential with a simple ϕ^4 mean-field theory treatment, approximating the grand potential curve as a polynomial:

$$G(\phi) = a\phi^2 + b\phi^4 \tag{3.31}$$

We choose the simplest, symmetric case, ignoring terms of $O[\phi^3]$ and powers of ϕ higher than 4. The minima occur at:

$$\phi_0 \equiv \pm\sqrt{\frac{-a}{2b}} \tag{3.32}$$

Here we consider $\phi(\vec{x}, t)$ is a continuous scalar field, varying with time and space, that represents the fluctuations around the average density (i.e. volume fraction) of one of the constituents in a phase-separating system, analogous to fluctuations of the discrete variable ϕ_A from section 3.1 about its average value. The total density of the system does not change over time:

$$\frac{\partial}{\partial t}\int \phi(\vec{x}, t) d^3\vec{x} = 0 \tag{3.33}$$

$\phi(\vec{x}, t)$ also obeys the continuity equation:

$$\frac{\partial \phi(\vec{x}, t)}{\partial t} = -\vec{\nabla} \cdot \vec{J}(\vec{x}, t) \tag{3.34}$$

where $\vec{J}(\vec{x}, t)$ is the current generated by the gradient in concentration:

$$\vec{J}(\vec{x}, t) = -\Gamma\vec{\nabla}\frac{\delta G(\phi)}{\delta \phi} + \vec{\zeta}(\vec{x}, t) \tag{3.35}$$

where $\vec{\zeta}(\vec{x}, t)$ is a stochastic, fluctuating part of the current that (because of the fluctuation-dissipation theorem) satisfies:

$$\left\langle \zeta_i(\vec{x}, t)\zeta_j(\vec{x}', t')\right\rangle = 2\Gamma k_\mathrm{B} T \delta(\vec{x} - \vec{x}')\delta(t - t')\delta_{ij} \tag{3.36}$$

Neglecting this thermal noise and combining equations 3.31, 3.34 and 3.35 yields an expression for the time-dependence of $\phi(\vec{x},t)$:

$$\begin{aligned} \frac{\partial\phi(\vec{x},t)}{\partial t} &= -\vec{\nabla}\cdot\vec{J}(\vec{x},t) && (3.37)\\ &= +\Gamma\vec{\nabla}\cdot\vec{\nabla}\frac{\delta G(\phi)}{\delta\phi} && (3.38)\\ &= \Gamma\nabla^2\frac{\delta G(\phi)}{\delta\phi} && (3.39)\\ &= \Gamma\nabla^2(2a\phi+4b\phi^3) && (3.40)\end{aligned}$$

3.3.1 Regular Diffusion

The first case we examine is for $a > 0$, looking near the minimum of the free energy at $\phi \approx 0$, for the symmetric case $\langle\phi\rangle = 0$, so that we can ignore the ϕ^3 term in equation 3.40. In this case,

$$\frac{\partial\phi(\vec{x},t)}{\partial t} \cong \Gamma\nabla^2(2a\phi) = 2\Gamma a\nabla^2\phi \equiv D\nabla^2\phi \tag{3.41}$$

This is simply the diffusion equation, with the diffusion constant D equal to:

$$D \equiv 2\Gamma a \tag{3.42}$$

This can be solved explicitly with a Fourier transform (given in one spatial dimension for simplicity):

$$\phi(x,t) = \frac{1}{2\pi L}\int \phi(q,t)e^{-iqx}dq \tag{3.43}$$

This spatial Fourier transform does not affect the time dependence:

$$\frac{\partial\phi(x,t)}{\partial t} = \frac{1}{2\pi L}\int \frac{\partial\phi(q,t)}{\partial t}e^{-iqx}dq \tag{3.44}$$

Taking the second spatial derivative,

$$\nabla^2\phi(x,t) = \frac{\partial^2\phi(x,t)}{\partial x^2} = \frac{-q^2}{2\pi L}\int \phi(q,t)e^{-iqx}dq \tag{3.45}$$

Substituting into equation 3.41:

$$\begin{aligned}\frac{\partial\phi(x,t)}{\partial t} &= 2a\Gamma\nabla^2\phi(x,t) &(3.46)\\ \frac{1}{2\pi L}\int\frac{\partial\phi(q,t)}{\partial t}e^{-iqx}dq &= 2a\Gamma\left(\frac{-q^2}{2\pi L}\right)\int\phi(q,t)e^{-iqx}dq &(3.47)\\ \frac{\partial\phi(q,t)}{\partial t} &= -2a\Gamma q^2\phi(q,t) &(3.48)\end{aligned}$$

This simple differential equation is trivially solved, yielding an explicit solution for $\phi(q,t)$

$$\phi(q,t) = \phi(q,0)e^{-2a\Gamma q^2 t} \equiv \phi(q,0)e^{-Dq^2 t} \quad (3.49)$$

where in the last step the definition of the diffusion constant from equation 3.42 is used. As a function of time, all modes at finite q decay away, leaving only the mode with $q = 0$. This reflects the conservation expressed in equation 3.33. Note that $D = 2\Gamma a$ formally goes *negative* for $a < 0$, which is a signature of spinodal decomposition. Once the system has phase separated, the revised analysis below—leading to a positive diffusion constant—is necessary.

Consider now the case $a < 0$ where $\langle\phi\rangle \neq 0 \equiv \phi_0$, so that there is a region enriched with one of the constituents. We now change variables to:

$$\Delta\phi = \phi - \phi_0 \quad (3.50)$$

For this purpose, we expand equation 3.40 in a Taylor series around ϕ_0:

$$\begin{aligned}\frac{\partial\phi}{\partial t} &\cong \left[\Gamma\nabla^2(2a\phi+4b\phi^3)\right]_{\phi_0} + \frac{\partial}{\partial\phi}\left[\Gamma\nabla^2(2a\phi+4b\phi^3)\right]_{\phi_0}(\phi-\phi_0) + \ldots &(3.51)\\ &= \Gamma\nabla^2\left[(2a\phi_0+4b\phi_0^3) + (2a+12b\phi_0^2)(\phi-\phi_0)\right] &(3.52)\\ &= \Gamma\nabla^2(2a+12b\phi_0^2)(\phi-\phi_0) &(3.53)\\ &= (2a+12b\phi_0^2)\Gamma\nabla^2(\phi-\phi_0) &(3.54)\\ &= (2a+12b\phi_0^2)\Gamma\nabla^2\Delta\phi &(3.55)\end{aligned}$$

Since ϕ_0 is a constant, all derivatives vanish:

$$\frac{\partial \phi_0}{\partial t} = \nabla^2 \phi_0 = 0 \tag{3.56}$$

So that,

$$\frac{\partial \phi}{\partial t} = \frac{\partial \phi}{\partial t} - 0 = \frac{\partial \phi}{\partial t} - \frac{\partial \phi_0}{\partial t} = \frac{\partial \Delta\phi}{\partial t} \tag{3.57}$$

This yields the final expression for $\Delta\phi$:

$$\frac{\partial \Delta\phi}{\partial t} = (2a + 12b\phi_0^2)\Gamma \nabla^2 \Delta\phi \tag{3.58}$$

which is of the same form as equation 3.41. We therefore recover ordinary diffusion and the same solution as in equation 3.49, except now with a slightly different effective diffusion constant D_f:

$$D_\text{f} = (2a + 12b\phi_0^2)\Gamma \tag{3.59}$$

This reduces to the original diffusion coefficient in equation 3.42 when $\phi_0 = 0$. Since, in accordance with equation 3.32, $\phi_0^2 = \frac{-a}{2b}$ in equilibrium, we see in this case that:

$$D_\text{f} = \left(2a + 12b\left(\frac{-a}{2b}\right)\right)\Gamma = -4a\Gamma = 4\Gamma|a| \tag{3.60}$$

which is twice the value of the diffusion constant for the case $a > 0$ in equation 3.42.

3.3.2 Spinodal and Binodal Decomposition

So long as a is greater than zero, the effective diffusion constant in equation 3.59 will be positive, and ordinary diffusion is observed. However, if a is sufficiently negative and ϕ_0^2 has not yet grown to its equilibrium value, so that $D_\text{f} < 0$, then all hell breaks loose. Fourier modes for $q \geq 0$ grow exponentially in time from the very beginning ($t = 0$), and the expansion quickly breaks down (the growth of Fourier components is stabilised for large q by higher-order gradient terms). The system is thermodynamically unstable from the very beginning, and makes frantic attempt to reach equilibrium; this is known as *spinodal decomposition*. Morphologically, the two phases form an interpenetrating bi-continuous network that coarsens until complete phase separation is achieved.

In between, where $a < 0$ but $D_\mathrm{f} > 0$, the system will eventually split into two separate phases, since the free energy still has two local minima at $\phi = \pm\phi_0$, away from $\Delta\phi = 0$. However, because the effective diffusion constant is still positive, no finite-q modes can grow in the way that occurs for spinodal decomposition. Instead, droplets of one phase nucleate in the other, and grow in a kinetically-limited process known as *binodal decomposition* (see, for instance, chapter XII in [161]).

The boundary between spinodal and binodal decomposition occurs exactly when $a < 0$ (so that the equilibrium state is phase-separated), but $D_\mathrm{f} = 0$, at particular points along the $G(\phi)$ curve. To understand how this happens, note that, from equations 3.31 and 3.59:

$$D_\mathrm{f} = \Gamma \frac{\delta^2 G(\phi)}{\delta\phi^2} \tag{3.61}$$

The places where D_f is zero are the same places where the second derivative of $G(\phi)$ is zero—that is, where $G(\phi)$ has an inflection point.

The language in the literature to describe how these relate to boundaries in a phase diagram is, however, somewhat confusing. The *coexistence curve*, or *binodal line*, is typically what is marked out in traditional phase diagrams, and refers to the volume fractions of the two phases in an equilibrium coexistence. The *spinodal line* refers to the boundary between spinodal and binodal decomposition. The equilibrium state of final coexisting volume fractions is determined by the binodal line, no matter whether the sample phase separates by binodal or spinodal decomposition.

For a given a and b, then, the spinodal and binodal lines can be determined in terms of ϕ. At the binodal line $\phi \equiv \phi_{\text{bin}}$, $G(\phi)$ is at a minimum:

$$\left.\frac{\delta G(\phi)}{\delta \phi}\right|_{\phi_{\text{bin}}} = 0 \tag{3.62}$$

$$\left.\frac{\delta}{\delta \phi}(a\phi^2 + b\phi^4)\right|_{\phi_{\text{bin}}} = 0 \tag{3.63}$$

$$\left.(2a\phi + 4b\phi^3)\right|_{\phi_{\text{bin}}} = 0 \tag{3.64}$$

$$\phi_{\text{bin}} = \pm\sqrt{\frac{|a|}{2b}} \tag{3.65}$$

This is represented by the points where the blue dashed curves meets the solid black curves, marked in the top plot of figure 3.5.

At the spinodal line $\phi \equiv \phi_{\text{spin}}$, $G(\phi)$ has an inflection point:

$$\left.\frac{\delta^2 G(\phi)}{\delta \phi^2}\right|_{\phi_{\text{spin}}} = 0 \tag{3.66}$$

$$2a + 12b\phi_{\text{spin}}^2 = 0 \tag{3.67}$$

$$\phi_{\text{spin}} = \pm\sqrt{\frac{|a|}{6b}} \tag{3.68}$$

This is represented by the points where the red dotted curve meets the blue dashed curves, marked in the top plot of figure 3.5.

Note that $|\phi_{\text{bin}}| > |\phi_{\text{spin}}|$, always, so that there are three different regions of the phase diagram:

1. $0 < |\phi| < |\phi_{\text{spin}}|$: The sample is thermodynamically *unstable*, and undergoes spinodal decomposition until the two phases reach $\phi = \pm\phi_{\text{bin}}$. This is represented by the red dotted curve in the top plot, and the filled red region in the bottom plot, of figure 3.5.

2. $|\phi_{\text{spin}}| < |\phi| < |\phi_{\text{bin}}|$: The sample is thermodynamically *metastable*. Without fluctuations, it will remain homogeneous because it must cross an activation barrier. In the real world, however, hopping over the barrier enables phase separation via binodal decomposition, until the two phases reach $\phi = \pm\phi_{\text{bin}}$. In reality, fluctuations tend to blur out the spinodal line. This is represented by the blue dashed curve in the top plot, and the filled blue region in the bottom plot, of figure 3.5.

3. $|\phi| > |\phi_{\text{bin}}|$: There is no phase separation, and the system remains homogeneous. This is represented by the red dotted line in the top plot, and the uncolored region in the bottom plot, of figure 3.5.

3.4 Growing Spinodal Length Scale

Returning to the definition of $G(\phi)$ in equation 3.31, a more precise definition would include higher-order gradient terms to account for concentration fluctuations:

$$G(\phi) = \int \left(a\phi^2 + b\phi^4 + \frac{1}{2}(\vec{\nabla}\phi)^2 + \frac{1}{2}\varepsilon\left(\nabla^2\phi\right)^2 \right) d^3\vec{x} \tag{3.69}$$

where ε is an arbitrary coefficient. Because everything stays linear, an additional term is added to the final expression for $\Delta\phi$ in equation 3.58:

$$\frac{\partial \Delta\phi}{\partial t} = (2a + 12b\phi_0^2)\Gamma\nabla^2\Delta\phi - \varepsilon\Gamma\nabla^2\nabla^2\Delta\phi \tag{3.70}$$

Using the Fourier transform method described above yields a solution of form similar to that of equation 3.49:

$$\Delta\phi(q,t) = \Delta\phi(q,0)e^{-\omega(q)t} \tag{3.71}$$

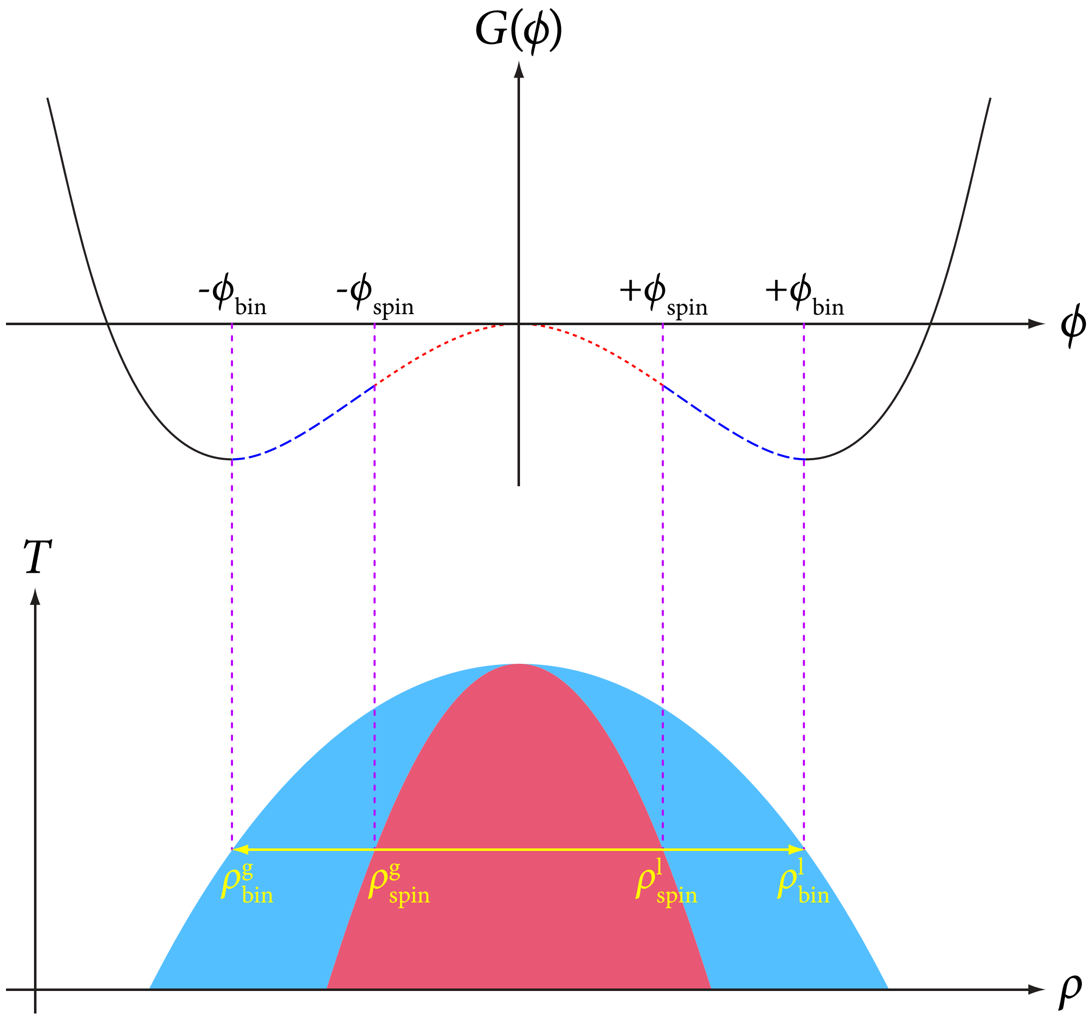
$G(\phi)$
$-\phi_{\text{bin}}$
$-\phi_{\text{spin}}$
$+\phi_{\text{spin}}$
$+\phi_{\text{bin}}$
ϕ
T
$\rho^{\text{g}}_{\text{bin}}$
$\rho^{\text{g}}_{\text{spin}}$
$\rho^{\text{l}}_{\text{spin}}$
$\rho^{\text{l}}_{\text{bin}}$
ρ

Figure 3.5: How to read the spinodal and binodal curves in a phase diagram. The top plot is a $G(\phi)$ and ϕ plot, whose shape is the same as that of the blue curve in figure 3.4. The spinodal region $0 < |\phi| < |\phi_{\text{spin}}|$ corresponds to the dotted red portion of the curve; the binodal curve $|\phi_{\text{spin}}| < |\phi| < |\phi_{\text{bin}}|$, to the dashed blue portion. Corresponding densities in a schematic temperature-density $T-\rho$ phase diagram in the bottom plot are connected with dotted purple lines. Note that $G(\phi)$ in the top plot corresponds to a *single* temperature—that is, a single horizontal line called a tie line—in the bottom phase diagram, marked in yellow. All points within the colored region of the bottom phase diagram are thermodynamically unstable and will phase-separate into a liquid and a gas. Those that occur in the red region will undergo spinodal decomposition; those in the blue region, binodal decomposition—but the final densities will be always along the outer boundary of the blue region. For instance, any point along the yellow line will, in equilibrium, separate into two phases with the densities marked by $\rho^{\text{g}}_{\text{bin}}$ and $\rho^{\text{l}}_{\text{bin}}$, regardless of whether the mechanism is spinodal or binodal. That is, samples are not stable at densities of $\rho^{\text{g}}_{\text{spin}}$ and $\rho^{\text{l}}_{\text{spin}}$.

Again, because everything stays linear and derivatives become powers of q in Fourier space, the function $\omega(q)$ in the exponent is then simply:

$$\omega(q) = (2a + 12\phi_0)\Gamma q^2 + \varepsilon\Gamma q^4 \equiv D_{\rm f} q^2 + \varepsilon\Gamma q^4 \tag{3.72}$$

Since $D_{\rm f} < 0$, $\omega(q)$ has a shape similar to that of the blue curve in figure 3.4, and minima at $q = \pm q^*$, defined by.

$$\left.\frac{\partial\omega(q)}{\partial q}\right|_{q\equiv q^*} = 0 \tag{3.73}$$

Because $-\omega(q)$ occurs in the exponent, the wavevectors at its minima, i.e. q^* will grow exponentially faster than any other length scales in the system; this is the mechanism responsible for creating the characteristic length scale that is one hallmark of spinodal decomposition.

3.5 Acknowledgements

The material in this chapter is largely based on conversations with David Nelson, who explained everything about spinodal decomposition to me, and to whom I am grateful for his time and patience in correcting my misconceptions and mistakes. Hopefully my additions of intermediate steps in the calculations do not confuse his clear pedagogical presentation. I also thank Eli Sloutskin for his careful proofreading and insightful comments.

Part II

Engineering

Chapter 4

Target-Locking in Real-Time Confocal (TARC) Microscopy

4.1 Abstract

We present a real-time target-locking confocal microscope that follows an object moving along an arbitrary path, even as it simultaneously changes its shape, size and orientation. This Target-locking Acquisition with Realtime Confocal (TARC) microscopy system integrates fast image processing and rapid image acquisition using a Nipkow spinning-disk confocal microscope. The system acquires a 3D stack of images, performs a full structural analysis to locate a feature of interest, moves the sample in response, and then collects the next 3D image stack. In this way, data collection is dynamically adjusted to keep a moving object centered in the field of view. We demonstrate the system's capabilities by target-locking freely-diffusing clusters of attractive colloidal particles, and actively-transported quantum dots (QDs) endocytosed into live cells free to move in three dimensions, for sev-

eral hours. During this time, both the colloidal clusters and live cells move distances several times the length of the imaging volume.

4.2 Introduction

The advent of high-speed confocal microscope systems has allowed the rapid, three-dimensional imaging of a number of dynamic processes in physics, materials science and biology [170, 171, 301]. Typically, a fixed three-dimensional volume within the sample is imaged periodically over time, which is normally adequate for samples and systems where objects remain within the field of view for the duration of the experiment. However, many investigations attempt to image dynamic phenomena, where the object of interest can move out of the three-dimensional imaging volume. In soft-condensed matter physics, for instance, free clusters of attractive colloidal particles will typically diffuse out of view on timescales comparable to their growth or internal rearrangement [171]. In biology, many *in vivo* investigations require the observation of processes involving freely-moving live cells, such as studies of motility or parasitic invasion [298, 105, 106], for even a basic qualitative understanding. For these experiments, immobilizing the cells can interfere with the ability to answer the question of interest [41, 9]. In whole-membrane investigations, for instance, surface-adhered areas of a cell membrane encounter a local chemical environment vastly different from the areas exposed to the medium.

An alternative approach is to target-lock by actively moving the sample to keep a moving object, such as a cell, in the center of the field of view; this enables observation for far longer periods of time. Well-established techniques can target-lock a single object in 3D at high speeds, treating it as an isolated point with no internal structure [21, 192, 210, 158, 43, 212]. However, these single-point techniques are inherently less adept at following objects with prominent internal structure, or multiple objects moving independently.

Here we describe the Target-locking Acquisition with Real-time Confocal (TARC) microscopy system, which can follow a collection of multiple objects as they move along arbitrary 3D paths, even with significant changes in shape, size and orientation. Instead of following a single bright spot, we image multiple fluorescent objects, determine their positions and structure in three dimensions, and target-lock by moving the sample in response to geometric analysis of these data. The system integrates rapid image analysis into the data acquisition process, so that the results of analyzing one 3D stack of images influence the collection of the next stack. A departure from many confocal experiments, the volume we image in the sample is not fixed in space, but instead is moved in response to dynamic changes within the sample. We demonstrate the TARC system by target-locking two systems: freely-diffusing clusters of attractive colloids, which change their shape, position, orientation and size throughout the experiment; and actively-transported quantum dots endocytosed into live cells free to move in three dimensions. The complete proprietary source code, project files, executables, electronic circuit diagrams and detailed schematics have been made freely available online for academic research purposes only (see website).

4.3 Materials and Methods

4.3.1 Apparatus overview

To target-lock a moving object, the TARC system first acquires a 3D stack of data, rapidly collecting a sequence of 2D confocal images from successive planes in the sample perpendicular to the optic axis. We use a Nipkow-disk confocal scanner (NCS) and CCD camera to collect these images, but any confocal, multi-photon or related technique could be equivalently used to acquire a 3D image stack. Next, the system processes the images and performs a full structural analysis to identify and characterize the object it is target-locking. In the examples here, the TARC system determines the exact position of the center of mass (COM) of the largest object in the sample, and moves the microscope stage to bring that point to the center of the 3D imaging volume. The next 3D image stack is then acquired. Image collection and image analysis alternate, so that the results of analyzing one stack determine the position where the next stack is acquired.

In order to target-lock moving objects as quickly as possible, we chose an NCS for high speed, and a piezo-based objective translator to allow rapid access to different sample planes perpendicular to the optic axis. Separately, to reposition the imaging volume between 3D stacks, where speed is not crucial, the microscope stage could be driven along three orthogonal axes with stepper motors.

A major challenge is coordinating the actions of all the hardware and software components quickly enough for effective target-locking. One particular issue with most NCS systems is that the disk spins freely at one rate, the camera acquires streaming images at a different rate, and there is no external synchronization between the two. This phase mismatch can significantly constrain the maximum frame rate: fringing moiré patterns, and eventually large overall intensity fluctuations, appear in the acquired images as frame rates increase. In addition, piezo-based microscope objective translation is usually controlled via software on the host PC in many commercial implementations, which does not allow the precise timing control needed to move the piezo only during the few milliseconds after each frame when the camera is not collecting data.

Sufficient sub-millisecond timing precision cannot be achieved in software (using a Windows PC) alone, and attempts with an internal control board were swamped by electrical noise generated inside the PC. Instead, we opted for hardware external to the host PC for timing control, designing and building a custom pulse generator which triggers and synchronizes camera exposure, spinning-disk rotation rate and piezo translation, with 10 μs temporal precision.

Before data collection begins, the host PC initializes and uploads control parameters to the camera, piezo controller and pulse generator. The PC then signals the pulse generator to begin data collection. From that point onward, the PC receives camera images and analyzes them, moving the automated stage once per 3D stack to implement target-locking, but otherwise performs no timing control. The rest of the hardware is synchronized by the pulse generator.

4.3.2 Hardware description

A block diagram of the TARC system's main hardware components is shown in figure 4.1. The main optical components are attached to an upright microscope (Leica DM-RXA). Laser excitation is provided by a 532-nm Nd:YVO_4 diode-pumped solid-state laser

(CrystaLaser CGL-050-L), with a shutter controlled by a TTL signal from the pulse generator. The laser beam is coupled into a single-mode (TEM_{00}) fiber, which delivers a few milliwatts of light into a commercial NCS (Yokogawa CSU-10B).

The internal components of the NCS are depicted within the dotted grey rectangle in figure 4.1, briefly summarized here (see [85, 291] and references therein for discussion of the optical characteristics of this NCS). Two parallel disks, one with microlenses and one with pinholes, are rigidly fixed to a single shaft driven by a variable-speed motor. The motor controller accepts a TTL pulse for synchronization (e.g. to phase-match an NTSC video signal), which is supplied by the pulse generator. The beam exiting the fiber into the NCS hits the upper disk, which contains thousands of micro-lenses, and is split into numerous small mini-beams. These pass through a dichroic mirror fixed between the two spinning disks and are focused through a set of pinholes in the second disk. The mini-beams are then focused by the objective onto the sample, where they excite fluorescence in the focal plane. The objective then focuses the corresponding emission mini-beams back through the pinholes, which block light originating from other planes in the sample and thereby create confocal depth-sectioning. The Stokes-shifted emission mini-beams are reflected by the dichroic and imaged onto a cooled-CCD camera (QImaging Retiga 1394 EXi Fast). Rotating the disks, which have a spiral pattern of microlenses and pinholes, moves the excitation mini-beams within the sample focal plane in such a way to ensure uniform sample coverage. The CCD camera is configured by, and transfers data to, the host computer via IEEE1394 firewire, but is triggered by separate electronically-independent TTL logic circuitry, which we access with signals from the pulse generator. To ensure smooth data collection at high rates, the host PC is equipped with a hardware-based RAID5

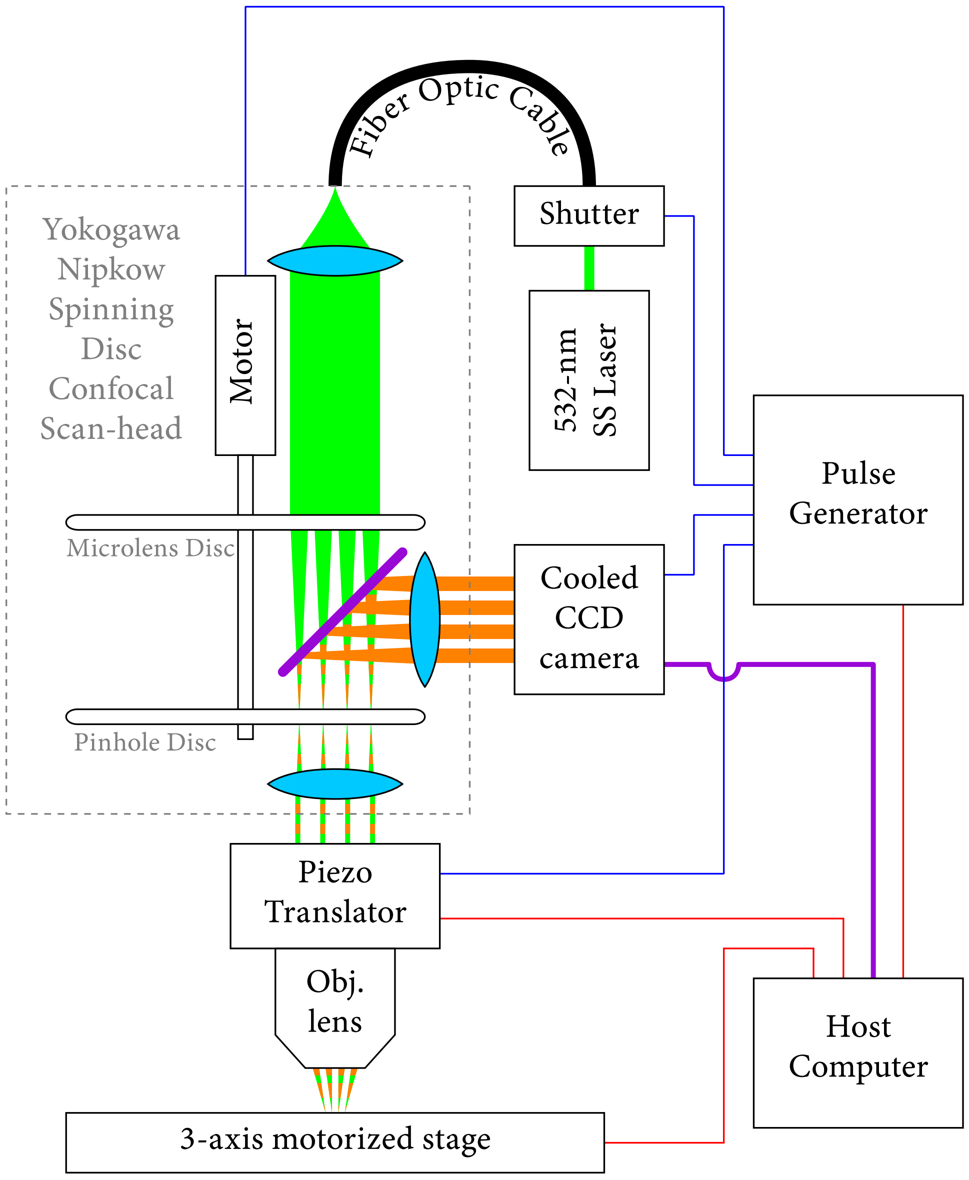

Figure 4.1: Block diagram of the TARC system (not to scale). Major components indicated by black boxes. The beam path is labeled in green (excitation) and orange (emission), with lenses in light blue. TTL signal connections are indicated with blue lines; RS-232, with red; and IEEE1394 firewire, with purple.

array of 10,000 rpm Ultra320 SCSI drives (Seagate).

Because of the confocal pinholes, only light from the focal plane of the objective reaches the detector, so the objective must be physically translated to access planes at different depths within the sample. This is accomplished using a piezo-based microscope objective translator (Physiks Instruments PiFOC) with a high-accuracy closed-loop controller (Physiks Instruments E662K001), configured via RS232 by the host PC, but triggered separately with TTL logic pulses from the pulse generator. The PC uploads a list of positions into a memory buffer on the controller, and each time a TTL pulse is received from the pulse generator (on a separate coaxial input, isolated from the RS232), the piezo moves to the next value on the list. In this way, the sequence of precise positions can be loaded and stored before the experiment begins, and accessed with great temporal precision via TTL triggering.

The pulse generator contains a microcontroller to manage RS232 communication with the host PC, and a number of counters and comparators implemented on several CPLDs, which generate repeated bursts of pulses of programmable number, period and delay, to several TTL outputs. An example pulse sequence, showing relative timings of the TTL signals sent by the pulse generator to the other parts of the TARC system, is shown in figure 4.2.

The microscope stage (Märzhäuser) is controlled, independent of the piezo, by stepper motors along three axes. The microscope stand's electronic focus control moves the stage up and down, along the z axis (the optic axis), while a separate controller (Leica DMSTC) controls the $x - y$ motion. Because the stage is moved only once per 3D image stack, precise timing control is not needed; the stage is controlled by software via RS232, with no

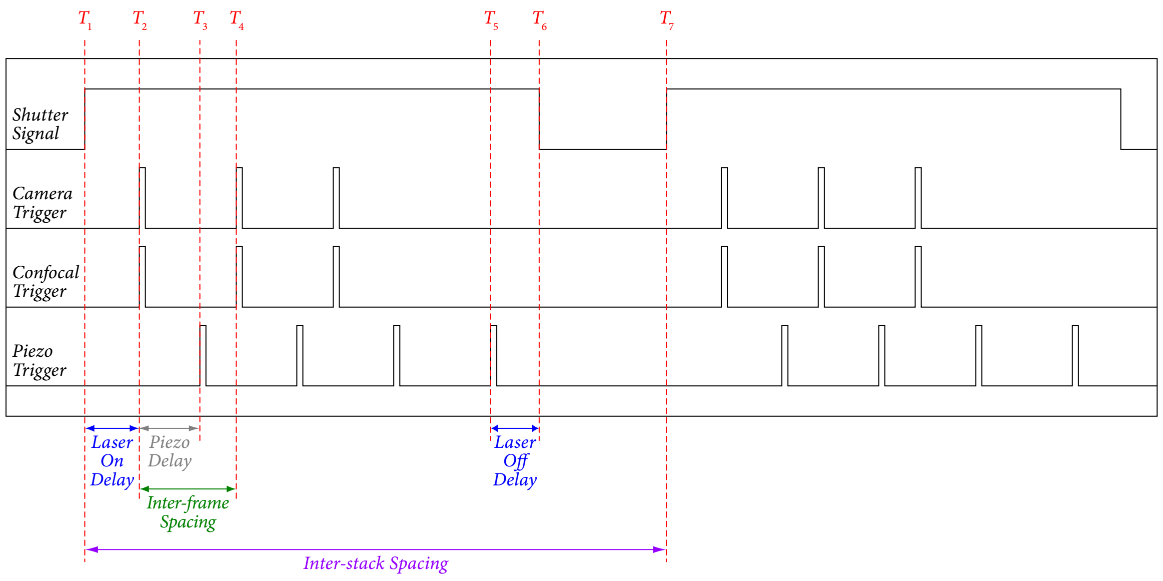

Figure 4.2: Pulse sequence for the acquisition of two 3D image stacks, each with three images. Data acquisition begins at T_1, when the pulse generator opens the laser shutter by raising *Shutter Signal* to a TTL-high value, which it maintains during the course of acquiring the first stack. At T_2, after delaying for *Laser On Delay* ($= T_2 - T_1$), the pulse generator sends a *Confocal Trigger / Camera Trigger* pulse to synchronize the confocal spinning disk and begin exposure of the CCD camera. At T_3, after delaying for *Piezo Delay* ($= T_3 - T_2$), the pulse generator then sends a *Piezo Trigger* pulse to move the piezo to the next position. At T_4, after delaying for *Inter-frame Spacing* ($= T_4 - T_2$) relative to T_2, the pulse generator sends another *Confocal Trigger / Camera Trigger* pulse to start acquisition for the next frame. And again, the piezo is then moved with a *Piezo Trigger* pulse following the end of acquisition of the second frame, after a delay of *Piezo Delay* relative to T_4. This process repeats for each frame in the 3D image stack. After the final frame in each stack is collected (i.e. the third frame here), the pulse generator sends several more *Piezo Trigger* pulses to move the objective back to the starting position in small steps: with immersion objectives, mechanical coupling via the viscous index-matching liquid will cause the sample to slip if the objective is moved too quickly. After the final piezo pulse, when the objective has returned to the starting position, the pulse generator waits for *Laser Off Delay* ($= T_6 - T_5$) before dropping *Shutter Signal* back to the TTL-low value, cutting off the laser and preventing sample bleaching during the waiting time between stacks ($= T_7 - T_6$). At T_7, after a delay of *Interstack Spacing* ($= T_7 - T_1$) relative to the acquisition start of the previous stack at T_1, the shutter is again opened and the acquisition of the second 3D image stack commences.

TTL triggering by the pulse generator.

4.3.3 Software overview

The main acquisition program performs several functions: it initializes and configures the pulse generator (with numbers and timings of the pulses), the piezo controller (list of positions to move through when triggered) and the camera (imaging parameters); and it subsequently manages the data acquisition by writing individual image files to disk as soon as each 2D image is delivered via firewire from the camera. Each image is stored as single compressed 8-bit grayscale TIF file, universally accessible from any image-editing program. This represents a significant departure from the operation of most commercial confocal implementations, which typically combine 2D images into 3D stacks in a temporary memory buffer before writing out huge, cumbersome aggregated data files to disk. The size of this temporary buffer, typically a few gigabytes, is comparable to the amount of system RAM or OS-dependent single-file maximum size, and represents the largest amount of data that can be collected without interruption. By contrast, writing each 2D frame to disk individually requires only small megabyte-size memory buffers, which are then cleared and recycled immediately. The main acquisition program therefore executes in just a few megabytes of RAM, with continuous real-time data-streaming to disk limited only by total disk capacity. Images have been acquired continuously for days without interruption, resulting in tens of gigabytes of uninterrupted image data.

After each 3D image stack has been collected, the main acquisition program launches a wrapper program that manages the target-locking system by calling several other programs to analyze the images and move the stage in response. All programs execute from the command-line to maximize speed and facilitate automated scripting, and were written in platform-independent C++. Using fully object-oriented classes and wrappers not only abstracts the hardware details from the programmer, but also facilitates a completely modular software architecture for the analysis. In particular, while the image analysis protocol in this example target-locks by moving the stage to keep the COM of the largest cluster of bright objects centered in the 3D imaging volume, any program that calculates a final stage displacement from analyzing 3D image data can be used in place of these routines, with only trivial changes to the wrapper program. Although this flexible modular approach implicitly prevents the main acquisition from knowing how quickly the independent image analysis and stage displacement will execute, the hardware delay between image stacks ($= T_7 - T_6$ in figure 4.2) is controlled by a single parameter in a configuration text file, and can be very easily measured in a few trial runs.

Our hybrid approach for 3D particle location identifies centroids in 2D images (largely based on a well-known algorithm [66]), then links up the text-data positions afterward into full 3D positions. Processing only a single 2D image at a time gives a number of performance advantages over the alternative approach of loading an entire 3D data set into memory at once. First, only a single image (of at most a few megabytes) needs to reside in memory at any given time, instead of the hundreds of megabytes of a typical 3D stack. Second, the optimized image-processing libraries used to increase performance are explicitly designed to work with 2D images, loading image data into the processor cache and parallel registers in a particular way to accelerate filtering operations that require access to adjacent rows of pixels; there is no corresponding method to do so for 3D data. Combining multi-threaded libraries with a vectorizing compiler (Intel) to take advantage of special features in recent processors yielded significant speed increases of several orders of magnitude relative to the standard implementations in MATLAB and IDL, even when compiled. This speed increase, resulting from the hybrid particle-location strategy and optimized code, ultimately enables the TARC system to target-lock fast enough to be useful experimentally, with very modest requirements for the host PC (here a 2 GHz Pentium 4 Xeon, 256 MB of RAM, Dell).

Finally, note that trivially changing a configuration text file can set the TARC system to acquire 3D image stacks with a fixed $x - y - z$ displacement between stacks, without running any image analysis. This capability can be used for sampling a much larger area, for example to gain better statistics in a measurement; for tiling adjacent 3D image stacks to make large composite images; or to sample a predetermined pattern or matrix of 3D volumes in the sample. Thus, in addition to running with full target-locking, the TARC system can also easily operate as a general-purpose, high-speed automated confocal acquisition system.

4.3.4 Sample preparation

To demonstrate the ability of the TARC system to properly target-lock highly anisotropic groups of objects over long times, we imaged aggregating clusters of attractive colloidal spheres [171]. Colloidal spheres (1.1 μm diameter) of polymethylmethacrylate (PMMA) with embedded DiIC18 fluorescent dye were suspended in a mixture of bromocyclohexane and decahydronaphthalene (Aldrich) in a proportion (nearly 5:1 by mass) that precisely matches the density of the particles, and sufficiently closely matches their index of refraction to enable confocal microscopy. Tetrabutyl ammonium chloride (Fluka), an organic salt, was added to screen Coulombic charge repulsion. Attraction between colloids was induced by the addition of nonadsorbing 11.6 MDa linear polystyrene (Polymer Labs), causing the colloidal spheres to aggregate into clusters several microns across, which diffuse as they continuously grow.

To demonstrate the capability to image living systems, we imaged live human lung cancer cells that actively transport endocytosed quantum dots (QDs) [186]. Human lung cancer cells (A549) were cultured in Dulbecco's Modified Eagle Medium (DMEM, ATCC) supplemented with 10% fetal bovine serum (FBS) at 37°C and 5% CO_2. For QD aggregate endocytosis, streptavidin-coated QDs (Invitrogen) with emission at 655 nm were combined with an equal volume of biotinylated poly-arginine (Invitrogen). The mixture was incubated at room temperature for 10 minutes, and the functionalized QDs were introduced to the cell culture at 200 pM. Following a one-hour incubation under normal culturing conditions, the medium was replaced and aggregate endocytosis was allowed to occur over 18 hours. In order to visualize the cell membrane, Alexa Fluor 532-labeled streptavidin (Invitrogen) was combined with the aforementioned biotinylated poly-arginine, and the resulting complex was introduced to the cell culture at about 1 nM one hour before imaging and incubated under the normal culturing conditions. Immediately prior to imaging, the cell culture was trypsinized, and the cells were introduced to the imaging chamber following trypsin inhibition.

To explore target-locking in faster-moving prokaryotic cells, we also imaged quantum dots inside *E. coli*. BL21(DE3)pLysS *E. coli* cells were grown to mid-log phase in standard LB medium in a 37°C shaker. The cells were then incubated for one hour at room temperature following the addition of 1 nM streptavidin-coated quantum dots conjugated to biotinylated poly-arginine. The cells were pelleted by centrifugation at $1500g$ for 10 minutes and re-suspended in fresh LB medium before imaging.

4.4 Results

4.4.1 Colloidal Clusters

We imaged the colloidal clusters with a 100× 1.4NA oil-immersion objective (Leica), collecting and analyzing a 3D stack of 61 images, each 500×500 pixels, every 40 seconds. Image collection took 6 s, and analysis took $\sim$ 2 s, for each stack. As shown in figure 4.3, the TARC system properly target-locked the freely-diffusing single central cluster under a variety of circumstances: when other, smaller clusters entered and left the imaging volume (figures 4.3a-c); when two smaller clusters merged to form a single cluster, dramatically changing shape and size (figures 4.3c-d); and when a highly non-spherical cluster changed orientation (figures 4.3d-e). Proper target-locking was observed for 36,000 seconds (10 hours), as the central cluster diffused a distance many times its own length, and several times that of the $24 \times 24 \times 16$ μm^3 imaging volume (figures 4.4 and 4.5).

4.4.2 Live Cells

We imaged the live human lung cancer cells with a 63× 1.2NA water-immersion objective (Leica) at 37°C, collecting and analyzing 3D stacks of 61 images, each 300×300 pixels, every 10 seconds. Image collection took 6 s, and analysis took $<$ 1 s, for each stack. As shown in figure 4.6, the TARC system properly target-locked the living lung-cancer cell for more than 5,000 seconds (1.4 hours). During this entire time, we observed

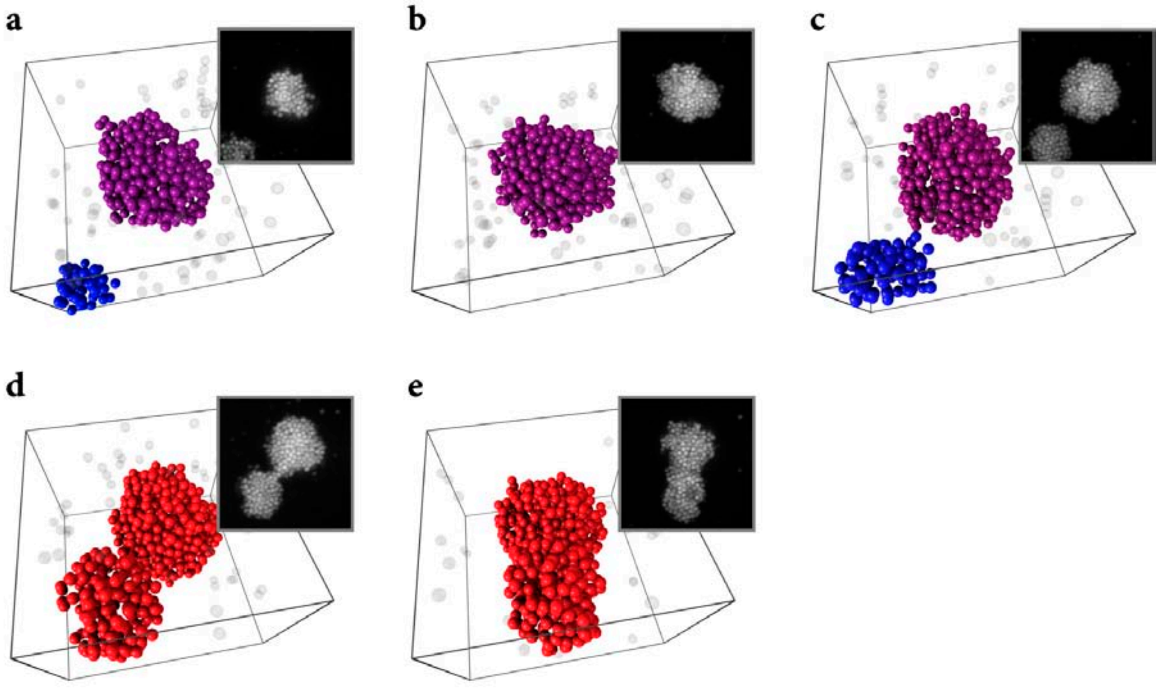

Figure 4.3: Target-locking freely-diffusing clusters of colloidal spheres: 3D reconstructions and (inset) 2D confocal images ($24 \times 24\ \mu m^2$) of a growing cluster. In 3D reconstructions, monomers and dimers are transparent grey, and the color of larger clusters indicates their number of spheres, following the color bar at the left of the graph in **(g)**. In **(a)**, a small cluster enters the volume in addition to the largest central cluster, and the TARC system properly follows the larger central cluster after **(b)** the smaller cluster has departed the imaging volume. **(c)** Later, another small cluster enters the volume and **(d)** merges to form a much larger cluster, which then **(e)** rotates and contracts. In all cases, the TARC system successfully follows the largest cluster in the imaging volume.

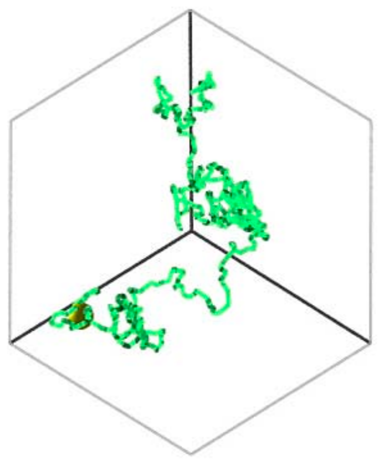

Figure 4.4: Target-locking freely-diffusing clusters of colloidal spheres: 3D plot of the trajectory of the largest cluster's center of mass.

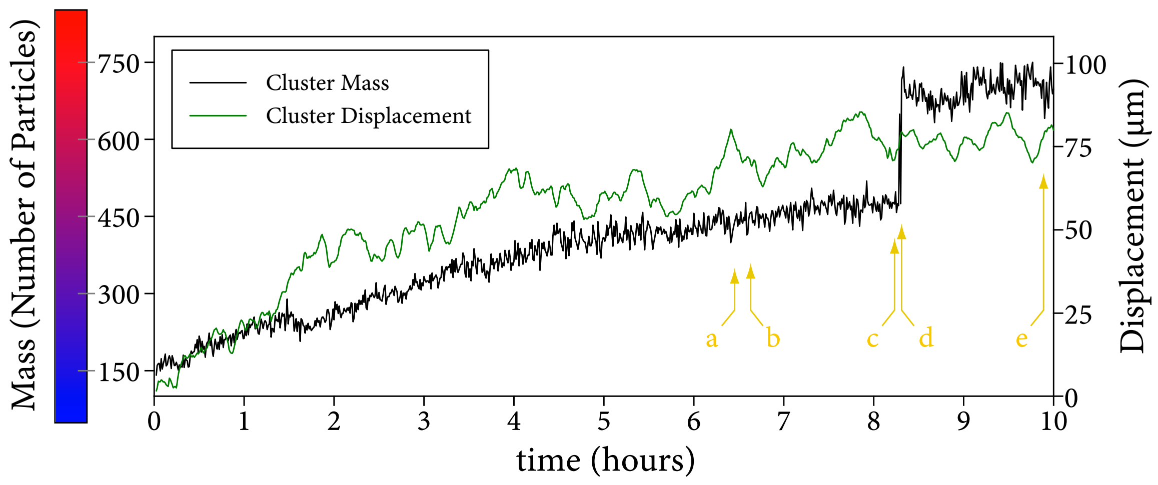

Figure 4.5: Target-locking freely-diffusing clusters of colloidal spheres: mass (number of particles; black line) of the largest cluster in the imaging volume, and the displacement of its center of mass relative to its initial position (green line) through time. Yellow arrows indicate times of structures depicted in figure 4.3.

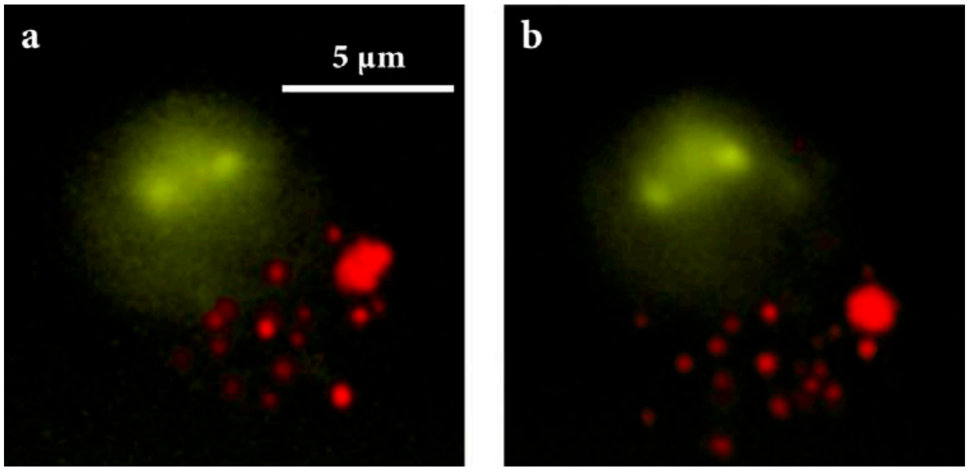

Figure 4.6: Target-locking actively-transported QDs in a freely-moving cell: confocal images of a human lung cancer cell, with cell membrane highlighted in green, and quantum dots undergoing active transport in red, at **(a)** 1020 seconds and **(b)** 2950 seconds elapsed time.

active transport of the vesicle-enclosed quantum dot aggregates, which moved significantly relative to the cell membrane (figure 4.6), while the cell itself moved 50 μm, many times its own length and that of the $14 \times 14 \times 13$ μm^3 imaging volume (figures 4.7 and 4.8). We also target-locked QDs absorbed by faster-moving *E. coli* by running at higher speeds, collecting and analyzing 3D stacks of 40 images every 5 seconds for several hours (data not shown).

In all of these examples, we collected ten 2D images per second with the CCD camera, limited by exposure time and readout speed. Our host PC processed each 3D image stack in at most a few seconds, and often much faster, so our efficient image processing scheme probably did not limit the speed in these cases. Ultimately, it appears that the mechanical stability of the piezo objective translator limits 3D acquisition to around thirty 2D images per second (video rate). Maximizing for speed, we successfully acquired complete 3D stacks of 40 images, each 512×400 pixels, every 2 seconds with an EMCCD camera

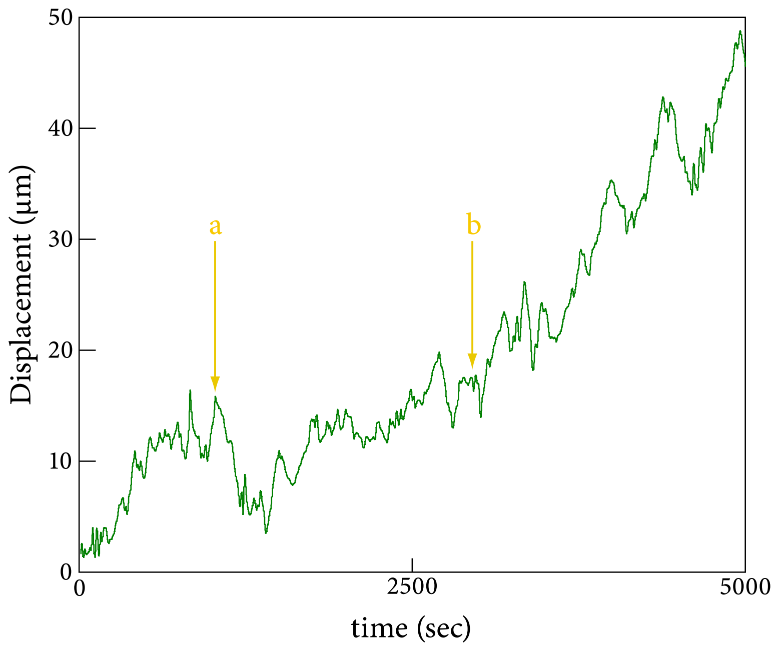

Figure 4.7: Target-locking actively-transported QDs in a freely-moving cell: displacement from original position, with yellow arrows indicating times depicted in 4.6.

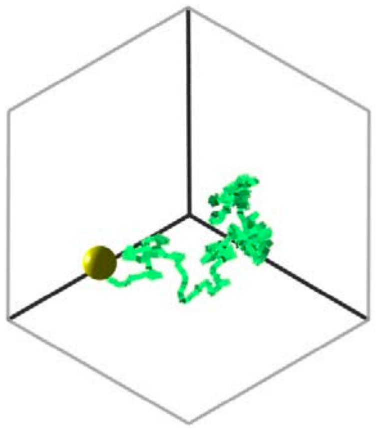

Figure 4.8: Target-locking actively-transported QDs in a freely-moving cell: 3D trajectory plot of the center of the cell.

(QImaging), whose greater sensitivity permitted far lower exposure times than the standard cooled CCD. In all cases, the TARC system ran indefinitely, and we have target-locked colloid clusters continuously for more than a day, generating thousands of 3D stacks.

4.5 Discussion

This long-time stability is only possible by performing a full 3D reconstruction and locking onto a specific geometric feature determined in a complete structural analysis. This is a significant advance over previous systems, where the image processing consists of finding the intensity maximum within the imaging volume and following it [210]. When multiple objects enter the imaging volume, systems with this approach can lock onto a point (the effective center of intensity) that lies outside of all the fluorescent objects, and may subsequently lose the proper target. By contrast, as shown in figure 4.3, the TARC system gracefully handles multiple objects coming in and out of the imaging volume, while keeping the largest cluster stably centered. Moreover, target-locking onto any well-defined point within a cluster, selected by any number of other structural characteristics (e.g. radius of gyration, fractal dimension, or density) instead of the mass, requires trivial changes to the code and will incur no performance penalty.

Even more generally, while the image analysis described here specifically identifies clusters of fluorescent objects, it is an independent program that executes separately from the main image acquisition program. This independence allows substitution of *any* analysis program, in any language, that takes a set of images as input and outputs a stage displacement. In this way, pre-existing image analysis routines, currently used to analyze data after image collection has ended, could be redeployed for active target-locking using the TARC system, thereby controlling the data acquisition process itself.

Finally, the examples here highlight direct imaging, but the TARC system could also be used as a target-locking system orthogonal to primary data collection, operating through one microscope camera-port and periodically moving the stage to track a freely-moving object, while data is collected simultaneously with an entirely separate technique. And as previously mentioned, while an NCS was chosen for several practical reasons, primarily high time resolution, our target-locking scheme should also work with other types of confocal or multi-photon systems.

By making the TARC system's designs and code freely available for academic research purposes only, we hope that its unique capabilities will enable new and unique contributions to understanding dynamic interactions in physics, materials science and biology.

4.6 Acknowledgements

This work was supported by NASA (NAG3-2284), the NSF (DMR-0243715), the Harvard MRSEC (DMR-0602684) and the Marie-Curie RTN for Dynamical Arrest (MRTN-CT-2003-504712). Peter J. Lu was supported by an NDSEG graduate fellowship and received additional funding from Calvin and Florence Lu. Peter A. Sims was supported by an NSF Graduate Research Fellowship. 3D reconstructions were created with Pixar's RenderMan. We thank Prof. X. Sunney Xie for providing quantum dots and cell culture facilities; Andrew Schofield for particle synthesis; David Hitrys for camera assistance; John Crocker and Eric Dufresne for help with algorithms; William Walker for circuit assembly; Daan Maas and Pixar for RenderMan assistance; David Cotter and Mark Christiansen for help with video production; and Conor Evans for critical reading of the manuscript.

This chapter has been published in *Optics Express*, co-authored with Peter Sims, Hidekazu Oki, Jim Macarthur and Dave Weitz [172].

Chapter 5

Confocal Microscope Hardware Control

5.1 Introduction

This chapter presents the hardware and software I developed with collaborators to create the high-speed confocal microscope system used to acquire the data that largely forms the scientific basis of this thesis. I have written the vast majority of code in C++, and for clarity present only relevant fragments while mostly omitting the housekeeping code, which should be self-evident to an experienced programmer going through the complete source code.

While documenting what I have done in the course of development, this chapter also serves as an additional reference for *how* and *why* certain technical decisions were made regarding code and hardware, as well as providing some additional information and commentary on algorithms, performance, implementation, etc. While this should be readable to anyone with a working knowledge of C++, no effort is being made to explain the basics of the language itself, which can be found in a number of excellent references[74, 166].

5.2 Hardware Control for Image Acquisition

Chapter 4 (published as [172]) describes the PLuTARC (**P**eter **Lu**’s **T**arget-Locking **A**cquisition with **R**eal-Time **C**onfocal) system, which unifies hardware control with rapid software image processing, but does not delve in the specifics of the code in great detail, on account of space constraints in the paper. Historically, these two parts—acquisition and analysis—were developed separately. I was originally tasked to design and construct a confocal microscope system for general imaging of colloids. I also recoded the image analysis routines, starting with the Pretrack IDL code developed by Crocker and Grier[66] as described in chapter 6. After using these separately, I thought that integrating the two sets of programs could yield a whole greater than the sum of the parts, which lead to the development of PLuTARC. But its automated confocal microscope image acquisition, *without* any target-locking, is the foundational data collection method underlying the other parts of the thesis, and is the third-generation system I developed in a sequence of improvements, which I briefly describe here.

5.3 Preliminary Control System Attempts

The first generation confocal imaging system I built relied on adding a simple NTSC video camera connected via co-axial cable to a framegrabber on the PC. A small circuitry box was designed using a sync-stripper to remove the timing signal from the video stream, which in turn triggered the piezo controller. Unfortunately, the camera itself could not be easily controlled (for instance, to program in the finite number of pulses to define a 3D stack). The only software used was the standard stream-to-memory tools included with the framegrabber; there was no efficient stream-to-disk provided.

I also controlled the Physiks Instruments (PI) piezo by means of an internal National Instruments (NI) analog output board, which would put out a programmable voltage ranging from 0 to 10 volts. When fed into the proper closed-loop input of the piezo controller, this signal would move the microscope objective between 0 and 100 microns. The major problem of internal control boards within the PC is noise, and in fact the NI board had a ripple greater than 10 mV, so that single-micron positioning was not generally possible: stepped sequences with micron-sized steps turned into triangle waves. This precluded the sub-micron position needed for proper $\hat{z}$-sectioning.

Some early attempts were made to trigger via the NI board, as well, but because of the general issues of noise inside the PC, we abandoned the internal-board-based strategy and moved the control hardware to an external box. Our first attempt was to construct a simple pulse generator, with knobs to control the timing of the pulses, which would trigger the camera via with a TTL signal over a coaxial input. Meanwhile, the piezo would be driven by its closed-loop controller's internal electronics (not an analog signal from the computer), which could be programmed via text input over RS232 to follow a predefined sequence, such as a triangle wave at a given frequency. Assuming the two frequencies of piezo motion and camera acquisition could be synchronized, 3D stacks could in principle be acquired, with the computer's only job to collect images from the framegrabber. Unfortunately, this solution essentially requires continuous acquisition, with no easy way to start or stop controllably, and there was no obvious way to exactly match the phase and frequency of the piezo controller's output with the camera acquisition. A new solution had to be devised to independently control the piezo and camera, and move the automated stage.

5.4 TTL-triggered Piezo Control

Although the piezo controller can be programmed to generate a predefined waveform or sequence of steps via RS232, there is no way to trigger the timing exactly on the standard controller (PI E662K). But PI also specially modifies these boxes for external triggering; the E662K001 has a TTL coax input that allows a programmed sequence of steps to be triggered, individually, every time a pulse is received. That is, a predefined sequence of displacements (i.e. 1 μm, 1.5 μm, 2 μm, etc. etc.) can be uploaded via RS232, and the controller can be set to step through the sequence every time a pulse is received. Up to 200 steps can be stored in memory, and when the final step is reached, the controller moves to the top of the list, and restarts. The PI E662K001 is controlled via simple ASCII text commands over a relatively slow serial connection (9600 baud 8-N-1), which cannot be changed.

We actually use relatively few commands, so I created some very simple wrappers to encapsulate the exact operations in `piezo_control.cpp`. Upon initialization, the program creates a new class to handle all communications with the piezo controller over RS232, with the following structure:

```
class PiezoController
{
public:
    PiezoController();
    PiezoController(const Params &Parameters);
    ~PiezoController();
    void StartListMotion(void);
    void StopListMotion(void);
    void RestartListMotion(void);
private:
    SerialPort PiezoSerial;
    int WriteString(const string text_to_send);
    void UploadSettingsToPiezo(const Params &Parameters);
    void ClearMemory();
    void MoveToPosition(const double position);
};
```

Upon creation of a `PiezoController` object, a new Windows object is created to send commands via the RS232 to the piezo controller. The `WriteString()` function handles the conversion and sending of ASCII data via RS232. Creating the class initializes the controller by switching control from the front panel to the RS232 (i.e. so that turning the knobs on the front of the piezo controller has no effect), stopping any motion that may be in progress, moving the piezo to a position of 15 microns, and flushing the buffer of any previous data:

```
PiezoController::PiezoController():PiezoSerial("COM4",9600)
{
    int bytes = WriteString(
        "DEV:CONT REM\nLIST:STOP\nPOS 15\nDATA:FLUS\n");
}
```

A similarly simple command, which is issued interactively, moves the piezo to any fixed position from 0 to 100 μm. Note that the piezo controller accepts micron positions as units, and internally converts to the proper voltage via its closed-loop circuitry, freeing us from this important calibration issue.

```
void PiezoController::MoveToPosition(const double position)
{
    ostringstream positionstringstream;
    positionstringstream << "POS " << position << endl;
    string outputstring = positionstringstream.str();
    int bytes = WriteString(outputstring);
}
```

5.4.1 Back steps

For automated acquisition, a string of properly-formatted numbers is uploaded via the `UploadSettingsToPiezo()` function, which takes four parameters as input: the start position, the step size, the number of steps, and the number of "back steps." This final parameter is of *crucial* importance to stability. Most microscopes get to the end of the 3D stack, and then go to the top of the list, often tens of microns away, in a single step. This is dangerous, because coupling between the objective and sample via immersion oil will lead to the sample often sticking and jumping off the stage slightly during this pullback operation. Because our pulse generator has a fixed frequency, the solution to this problem is to add a small number of steps, typically less than ten, and space them between the starting and ending position, so that the piezo takes smaller steps at the end of the list to return to its initial position. Because we have a limitation of 200 steps in the sequence from the piezo controller, typically a dozen or so back steps limits the number of slices in a 3D stack to around 180.

In the following description of the upload function, only relevant code fragments are presented. The function begins by clearing the memory buffer in the piezo controller.

```
int bytes = WriteString("DATA:FLUS\n");
```

5.4.2 Coding a manual time delay

Because of slow data transmission rates via RS232 at 9600 baud, a delay *must* be issued by the C++ the code, or it will execute too fast, and the next uploaded data via RS232 will not be received. There are several ways to invoke a delay, but perhaps the most efficient is to `ping` an internal address, sending the output of that command to `null`. This delay tactic is used throughout the hardware control. The number following the `-n` argument in the `ping` command is the delay in seconds, according to the documentation, but in reality is basically the delay plus one second. So in the following code fragment, the actual delay is about 2 seconds, not 3, before the next command is issued to bring the piezo to its start position of zero microns.

```
system("ping -n 3 127.0.0.1>nul");
bytes = WriteString("POS 0.0\n");
```

5.4.3 Formatting position data

Using the four parameters, a `stringstream` is created to hold all of the numbers in the list for both the forward step list, and the few backsteps. The positions in the buffer corresponding to the start and stop positions are set, and then the method of triggering is switched to the TTL coaxial input (as opposed to a programmable frequency which could be defined and uploaded). Finally, the `stringstream` is converted to a `string`, which is then uploaded via RS232:

```
ostringstream outputstringstream;
outputstringstream.setf(ios::fixed);
outputstringstream.setf(ios::showpoint);
outputstringstream << setprecision(2);
for (int i=0; i<=numsteps; i++) {
    outputstringstream << "DATA " << i+1 << ",";
    outputstringstream <<
        i * stepsize + startpos << "\n";
}
float total_distance = stepsize * (double) numsteps;
float step_fraction = 0, back_distance = 0;
for (int j=0; j<backsteps; j++) {
    outputstringstream << "DATA " <<
        j + 2 + numsteps << ",";
    step_fraction = (float) j / (backsteps-1);
```

```
        back_distance = exp(-5 * step_fraction)
            * total_distance;
        outputstringstream <<
            startpos + back_distance << "\n";
    }
    outputstringstream << "DATA:FIRS 1\n";
    outputstringstream << "DATA:LAST " << i+j << "\n";
    outputstringstream << "DATA:SAMP TRIG\n";
    outputstringstream << "POS " << startpos << "\n\n";
    string outputstring = outputstringstream.str();
    bytes = WriteString(outputstring);
```

Once this command is issued, the piezo controller is initialized and ready for automated acquisition. Several public `void` commands can then be called elsewhere to control motion through the list. When the start command is issued, the piezo moves through each step in the list when it receives an external pulse via the TTL input, but otherwise remains in this state indefinitely.

```
void PiezoController::StartListMotion(void)
{
    int bytes = WriteString("LIST:STAR\n");
}
```

Once acquisition is completed, sending a stop command decouples the motion of the piezo from the receipt of TTL pulses; that is, the piezo holds its current position and does not move through the list, irrespective of TTL input.

```
void PiezoController::StopListMotion(void)
{
    int bytes = WriteString("LIST:STOP\n");
}
```

5.5 Automated Stage Control

I followed a similar strategy for controlling the automated Leica DM-STC automated stage, creating a C++ class to handle the communication and control, presenting some simple wrapper functions externally to the other C++ programs (see source code for details, which is structurally very similar to the piezo control, and therefore will not be repeated here).

Although the controller and its application programming interface (API), come from Leica, the actual stepper-motor-driven $\hat{x} - \hat{y}$ stage is made by Märzhäuser. Moving the stage in $\hat{z}$, along the optic axis, involves driving the motorized focus of the microscope. So there are two separate serial connections, one to the DM-STC for $\hat{x} - \hat{y}$ control, and one to the DM-RXA microscope body itself, for $\hat{z}$ control. The C++ wrapper class makes this distinction invisible, though, so that externally the three dimensions are treated equally.

The control program uses just a few of the API commands; the extensive full Leica API allows control of almost all microscope functions remotely. Here, we focus simply on stage displacements. The wrapper function `MoveX` takes a boolean to control moving either left or right, and then a floating-point number for the absolute displacement, in μm. Once again, a `stringstream` is used to assemble all of the text data, which is then converted to a `string` and uploaded via RS232.

```
void LeicaStageController::
MoveX(const bool moveleft, const float delta_x)
{
    ostringstream movetextstream;
    movetextstream << "10006";
    if(moveleft == true) {
        movetextstream << "-";
    }
    else {
        movetextstream << "+";
    }
    movetextstream.setf(ios::right, ios::adjustfield);
    movetextstream.fill('0');
    movetextstream << setw(8) << int(40*delta_x);
    unsigned char newlinechar = 0x0D;
    movetextstream << newlinechar;
    string movetextstring = movetextstream.str();
    WriteStringTo_Stage(movetextstring);
}
```

A nearly identical command is used for motion along the $\hat{y}$ direction, except that a different API command is called, and the boolean argument controls whether the motion is forward or backward (in laboratory space):

```
void LeicaStageController::
MoveY(const bool moveup, const float delta_y)
{
    ostringstream movetextstream;
    movetextstream << "10007";
    if(moveup == true) {
        movetextstream << "-";
    }
    else {
        movetextstream << "+";
    }
    . . .
```

The $\hat{z}$-direction control also has similar commands, with slightly different parameters to convert to physical μm units, and determine the direction along the up-down axis. And, as previously mentioned, the destination of the data is to the microscope, not the external stage controller.

```
void LeicaStageController::
MoveZ(const bool movebottom, const float delta_z)
{
    ostringstream movetextstream;
    movetextstream << "60003";
    if(movebottom == true) {
        movetextstream << "+";
    }
    else {
        movetextstream << "-";
    }
    movetextstream.setf(ios::right, ios::adjustfield);
    movetextstream.fill('0');
    movetextstream << setw(7) << int(10*delta_z);
    unsigned char newlinechar = 0x0D;
    movetextstream << newlinechar;
    string movetextstring = movetextstream.str();
    WriteStringTo_Microscope(movetextstring);
}
```

5.6 CCD Camera Control

For actual image acquisition from the Yokogawa Nipkow disk confocal head attached to the Leica DM-RXA microscope, we selected a QImaging Retiga 1394 EXi Fast camera. Compared to the simple VGA camera, the Retiga has far higher resolution (1360×1064 pixels), and much greater sensitivity, with a quantum efficiency around 0.6 at the wavelengths around 550 to 600 nm where the fluorescence of red particles is detected. The camera is controlled via IEEE 1394 Firewire, over which collected image data is also sent, and can be programmed to be triggered by a TTL pulse sent over a cable to a different input with a keyboard connector.

The main job of the image acquisition software is to upload the imaging parameters to the camera, set it to external trigger operation, and then write the acquired images to disk as quickly as possible. Early versions of the software were based on a demonstration Windows-API graphical program in the QImaging SDK, which wrote images to a frame-buffer in memory. This was summarily modified to write to disk, and to work with the other hardware, but was not fast or reliable for very long 3D acquisitions because, frankly, too much software execution time was being used to handle all of the useless Windows under-the-hood API stuff. The application resulting, called `ALMA2.exe` (for **A**utomated **L**ight **M**icroscope **A**cquisition) is still used for quick previewing and testing of hardware (to make sure the piezo is properly initiated and functioning, for instance), and to collect 2D image sequences. The code for this program, however, is a gigantic nightmare. It began as a Windows demo program using the Microsoft Foundation Classes (MFC), to which a half-dozen other small programs for hardware control and image output were added, resulting in a amalgam that became progressively harder to maintain, and impossible to improve.

A complete rewrite was in order, and because the automated acquisition needed to be reliable and maximally fast, I decided to switch to a command-line driven program for simplicity, stability, performance, portability and maintainability. Hence I completely recoded the third generation confocal control program, `ALMA3.exe` and later renamed to `PLuTARC.exe`, from scratch.

The main software control program for the camera is `QCam_control.cpp`, and has just two main functions: initializing the camera, and then setting up a double-buffer scheme via call-back functions to (ultimately) write to a redundant array of inexpensive disks (RAID) as quickly as possible. Like the previous hardware control code, a C++ class encapsulates all of the details of interfacing directly with the hardware, in this case with the QImaging Retiga via the QCam API over firewire. The important parts of the class are listed below:

```
class QCam_DblBuffer
{
    friend void QCAMAPI
    CoreFrameCallback(void*, unsigned long,
        QCam_Err, unsigned long);

public:
    QCam_DblBuffer();
    QCam_DblBuffer(const Params &Parameters,
        PiezoController &Piezo);
    ˜QCam_DblBuffer();
    bool Initiate_Queuing(void);

private:
    struct {QCam_Frame frame1; QCam_Frame frame2;
        unsigned useframe;} twoframes;

    //callback functions
    QCam_Err Queue_Frame(unsigned long userData);
    void Frame_Callback( unsigned long userData,
        QCam_Err errcode, unsigned long flags );
    void Frame_Save(unsigned long whichframe);

    int Write_Frame_To_Disk(QCam_Frame &frame,
        const string outfilename);
    . . .
```

The constructor follows the standard QImaging protocol to load the driver and initialize the camera.

```
QCam_DblBuffer::QCam_DblBuffer
(const Params &Parameters, PiezoController &Piezo):
{
    //Load the driver.
    errcode = QCam_LoadDriver();
    if(errcode==qerrSuccess) {
        QCam_CamListItem listItem;
        unsigned long listLength = 1;
        errcode = QCam_ListCameras(&listItem, &listLength);
        if(errcode == qerrSuccess) {
            if(listLength == 0) {
            cout <<"No cameras found." << endl;
            }
            else {
            errcode = QCam_OpenCamera(listItem.cameraId,
                &handle);
            }
        }
    }
    . . .
```

The next step is to load the default settings for the camera, then set all of the individual imaging parameters, including resolution, binning, exposure time, gain, and offset, as well as the trigger type (e.g. external TTL triggering).

```
    //get default settings from the camera
    QCam_Settings settings;
    settings.size = sizeof(settings);
    errcode = QCam_ReadDefaultSettings(handle, &settings);

    //camera settings, in alphabetical order
    errcode = QCam_SetParam(&settings, qprmBinning,
            Parameters.get_binning());
    errcode = QCam_SetParam(&settings, qprmCoolerActive, 1);
    errcode = QCam_SetParam(&settings, qprmExposure,
            Parameters.get_exposure() * 1000);
    errcode = QCam_SetParam(&settings, qprmImageFormat,
            qfmtMono8);
    errcode = QCam_SetParam(&settings, qprmNormalizedGain,
            Parameters.get_gain() * 1000000);
    errcode = QCam_SetParam(&settings, qprmRoiHeight,
            Parameters.get_ROI_height());
    errcode = QCam_SetParam(&settings, qprmRoiWidth,
            Parameters.get_ROI_width());
    errcode = QCam_SetParam(&settings, qprmRoiX,
            Parameters.get_ROI_origin_x());
    errcode = QCam_SetParam(&settings, qprmRoiY,
            Parameters.get_ROI_origin_y());
    errcode = QCam_SetParamS32(&settings,
            qprmS32AbsoluteOffset, Parameters.get_offset());
errcode = QCam_SetParam(&settings, qprmTriggerType,
            qcTriggerEdgeHi);
errcode = QCam_SendSettingsToCam(handle, &settings);
errcode = QCam_SetStreaming(handle, true );
```

The last part of the constructor sets up the double-buffer data structure for data collection.

```
    unsigned long size; //Current image size, in bytes
    QCam_GetInfo(handle, qinfImageSize, &size);
    if(twoframes.frame1.pBuffer){
        delete [] twoframes.frame1.pBuffer;
    }
    if(twoframes.frame2.pBuffer) {
        delete [] twoframes.frame2.pBuffer;
    }
    twoframes.frame1.pBuffer = new unsigned char[size];
    twoframes.frame1.bufferSize = size;
    twoframes.frame2.pBuffer = new unsigned char[size];
    twoframes.frame2.bufferSize = size;
    memset(twoframes.frame1.pBuffer, size, 0);
    memset(twoframes.frame2.pBuffer, size, 0);
```

This sets aside two buffers in main memory, each the size of a single image frame, for buffering images coming from the camera. The first image goes to one of the buffers, and a second image goes to the second buffer while the first is being written to disk. This prevents data bottlenecks and allows continuous streaming without interruption at high speed.

The acquisition process begins by calling the `Initiate_Queuing()` function:

```
bool QCam_DblBuffer::Initiate_Queuing(void)
{
    if (((errcode=Queue_Frame(1))!=qerrSuccess)
    ||((errcode=Queue_Frame(2))!=qerrSuccess)) {
        return false;
    }
    return true;
}
```

If the error-checking condition is passed, the `Queue_Frame()` function is called, which first determines which of the two buffers is free to receive image data, then assigns that buffer tp the QImaging API function `QCam_QueueFrame()`:

```
QCam_Err QCam_DblBuffer::
Queue_Frame(unsigned long whichframe_to_use)
{
    QCam_Frame* pFrame;
    //figure out which frame to queue
    if(whichframe_to_use == 1) {
        pFrame = &twoframes.frame1;
    }
    else {
        pFrame = &twoframes.frame2;
    }
    //Use QCAMAPI QCam_QueueFrame to add new frame to queue
    errcode =
        QCam_QueueFrame(handle, pFrame, CoreFrameCallback,
        qcCallbackDone, this, whichframe_to_use);
    return errcode;
}
```

Once the camera acquires a frame, the QCam API calls the `CoreFrameCallBack()` function:

```
void QCAMAPI CoreFrameCallback(void* userPtr,
    unsigned long whichframe_to_use, QCam_Err errcode,
    unsigned long flags) {
    ((QCam_DblBuffer*) userPtr)->
        Frame_Callback(whichframe_to_use,errcode,flags);
    ((QCam_DblBuffer*) userPtr)->
        Frame_Save(whichframe_to_use);
}
```

The function `CoreFrameCallBack()` then calls two functions, using a pointer-to-function construct. The first, `Frame_Callback()`, checks to see if a frame was successfully captured by the camera. It then checks the variable `useframe` to determine if the camera is finished with the last frame. If the value is zero, then the camera is finished, and value of `useframe` is then set to `whichframe_to_use` to specify which of the two frame buffers to read out and save to disk. If the value of `useframe` is not zero, then the camera has not finished acquiring the last frame. In that case, the frame is re-sent to the buffer queue by calling `Queue_Frame()`.

```
void QCam_DblBuffer::Frame_Callback (unsigned long
    whichframe_to_use, QCam_Err errcode,
    unsigned long flags) {
    if(flags & qcCallbackDone) {
        if(!((whichframe_to_use == 1) ||
        (whichframe_to_use == 2))) {
            cout << "Callback has failed" << endl;
        }
        if ((errcode == qerrSuccess) ||
            (errcode == qerrBlackFill)) {
            if (twoframes.useframe == 0) {
                twoframes.useframe = whichframe_to_use;
            }
            else {
                Queue_Frame(whichframe_to_use);
            }
        }
    }
}
```

The actual image data is written to disk with the `Frame_Save()` function, which copies data from the appropriate image buffer (i.e. 1 or 2), as designated by the function argument `whichframe_to_use`.

```
void QCam_DblBuffer::
Frame_Save(unsigned long whichframe_to_use)
{
    unsigned long imageSize;
    QCam_Frame frame;
    if(whichframe_to_use == 1) {
        frame = twoframes.frame1;
    }
    else {
        frame = twoframes.frame2;
    }
```

The function also keeps a running counter of both the frame and the stack numbers, which are therefore incremented *only* when images are written to disk. If the frame number is the last in a given stack, it is reset to zero and the stack number incremented. The piezo is also reset at this point, in an attempt to prevent a missed step from accumulating an offset that will throw off the numbering in successive stacks.

```
    if(frame_counter < total_num_frames-1) {
        frame_counter++;
    }
    else {
        frame_counter=0;
        stack_counter++;
        piezo_control.RestartListMotion();
    }
```

Note that there is no direct checking—or even a way to physically check—between the pulse generator, piezo, and frame number written to disk, so that if something physically interferes with the stepping through the piezo program, the program will blithely continue to increase the file number, and things will get out of phase. In particular, if something ties up the computer so that it can't buffer fast enough, the file numbers and pulse sequence will get out of sync, and the data set will not be easily recovered without major renumbering.

Finally, the frame and stack numbers are assembled into a `stringstream`, which pads those numbers with zeros to three and six digits, respectively. The `stringstream` is then converted into a text `string`, which is passed as an argument, along with the frame, to a function that writes the frame to disk.

```
    ostringstream filenamestream;
    filenamestream << imagefilestem
```

```
        << Make_File_Number_Text
            (frame_counter, stack_counter)
        << ".tif";
    string filenamestring = filenamestream.str();
    Write_Frame_To_Disk(frame,filenamestring);
}
```

The function that writes the frame to disk converts between the QImaging buffer format into one used by FreeImage, an open-source library for reading, writing and converting images. Unfortunately, this function is rather ugly due to a number of idiosyncrasies. First, the image cannot be treated as a greyscale, but instead has to be considered an RGB image with equal values for the three color components.

```
    FIBITMAP *fi_bitmap
        = FreeImage_Allocate(frame.width, frame.height,
        8, 0, 0, 0);
    RGBQUAD *pal = FreeImage_GetPalette(fi_bitmap);
    for(int i=0;i<256;i++) {
        pal[i].rgbRed=i;
        pal[i].rgbGreen=i;
        pal[i].rgbBlue=i;
    }
```

Then, the pointer to the frame's image data is cast to an `unsigned char*`, which can then be manipulated, allowing the image data to be copied from the QImaging format to the FreeImage format one line at a time, to account for byte-alignment issues in how image data is stored in memory. Unfortunately, the simple copying of bytes does not work properly, and leads to funny offsets between lines.

```
    unsigned char* srcPtr
        = static_cast<unsigned char*>(frame.pBuffer);
    int row=0, rowoffset = 0;
    for (row=0; row<frame.height;row++) {
    memcpy(FreeImage_GetScanLine(fi_bitmap,row),
        srcPtr+rowoffset,frame.width);
```

```
rowoffset += frame.width - (rowoffset & 1);
}
```

Next, several functions in the library convert it to a final 8-bit greyscale bitmap, in a rather strange and unintuitive sequence that nevertheless works correctly and incurs no significant performance penalty.

```
FIBITMAP *fi_bitmap_cropped
    = FreeImage_Copy(fi_bitmap, crop_left, crop_top,
    crop_right, crop_bottom);
FIBITMAP *fi_bitmap16
    = FreeImage_ConvertToType(fi_bitmap_cropped,
    FIT_UINT16, true);
FIBITMAP *fi_bitmap_final
    = FreeImage_ConvertToStandardType(fi_bitmap16,true);
```

Finally, the frame is written to disk as an LZW-compressed TIFF. Note that saving as a PNG is slightly more space-efficient, but a bit slower in processing time.

```
if(fi_bitmap_final != NULL) {
    FreeImage_Save(FIF_TIFF, fi_bitmap_final,
        outfilename.c_str());
}
```

In summary, there are a number of non-intuitive steps required in the software to actually get the images out of the camera at maximum speed, with no delays. However, the ALMA3/PLuTARC acquisition has collected a terabyte of images, one megabyte-sized frame at a time, with no major stability problems. It almost never crashes, and can be used for multiple-day data collections without interruption.

5.7 Pulse Generator Control

The final piece of hardware is a pulse generator to control the timing via triggering of the camera and piezo. External hardware synchronization is absolutely crucial to reliable high-speed 3D operation, and distinguishes this particular system from most of its commercial counterparts (at least when it was first built circa 2004).

The pulse generator is based on Jim Macarthur's Digimite board design, built around a CPLD that handles the creation of various trains of pulses with programmable timing, delays and durations. A hardware diagram for internal logic is shown in figure 5.1.

The operation of this circuitry, with the different timings and triggers as relating to the other devices in the system is described in chapter 4. What follows here is a description of the control code on the host PC. As with the other hardware devices, a wrapper class (here called `Digimite`) was created to manage all communications with the pulse generator, encapsulated in a number of functions. The code presents simple wrappers to other programs to control the pulse generator with regular units (i.e. milliseconds), while handling all of the command translation and formatting behind the scenes.

```
class Digimite
{
public:
    Digimite();
    Digimite(const Params &Parameters);
    ˜Digimite();

    //One Byte Commands
    void ResetController();
    void UnResetController();
    void SlowDelayMode();
    void FastDelayMode();
    void SinglePiezoPulse();

    //Three Byte Commands
    void SetPulsePeriod(const int pulse_period);
    void SetPulseCount(const int number_of_counts);
    void SetStackDelay(const long pulse_delay);
    void SetShutterOnTime(const long shutter_on_time);
    void SetShutterOffTime(const long shutter_on_time);
```

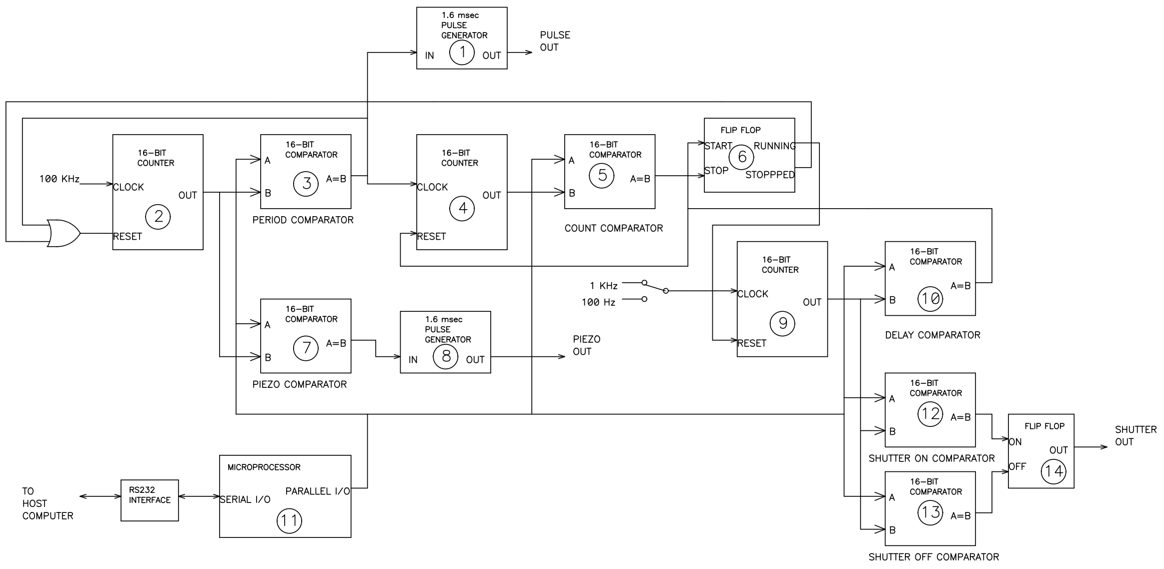

Figure 5.1: Pulse generator block diagram. At its heart is a set of three 16-bit counters and five 16-bit comparators. Counter (2) and comparator (3) form a 16-bit modulo-n counter where n is set by the microcontroller (11), which in turn gets its commands from the host PC via RS-232. Counter (2) starts at zero and counts up until it reaches n, at which point the $A = B$ output of comparator (3) goes high, which resets the counter. This brings the comparators output low again, and the counter resumes its count. The result is a thin pulse every n counts, or $10n$ μs, with our 100 KHz input clock. This pulse is stretched by a digital one-shot (1) to create a 1.6 ms output pulse. The output of counter (2) feeds a second comparator (7), which compares the count with a piezo delay value, also set by the microcontroller. When the count equals this value, the comparators output fires a second one-shot (8) which generates a pulse that is delayed from the main pulse by the programmed amount. The pulse from (3) feeds a second counter (4), which counts the number of pulses. Once this number reaches the programmed value N_s, comparator (5) clears flip-flop (6) which turns off the pulse generator and starts a delay counter (9). When comparator (10) detects that the counter has reached the programmed delay, it sets flip-flop (6), which restarts the pulse counter. The result is repeated bursts of a programmable number of pulses of a programmable period, one pulse for every frame acquired in a 3D image stack. Between each burst is a programmable delay, during which no pulses are sent to the camera or piezo, which remain idle. The last pair of comparators (12) and (13) are used to set and reset flip-flop (14) at programmable times during the delay, creating the shutter signal to block the laser during this delay period.

```
    void SetPiezoDelay(const int piezo_delay);

    //No byte commands
    void FlushBuffer();

private:
    void write3bytes(const long data, const int numbits);
    unsigned char checksum(const int numdatabytes);

    SerialPort DigimiteSerial;
    void write_data(const bool nonzerodata);
};
```

The class constructor initializes an RS232 connection, and creates a temporary array `tempbytes` of type `unsigned char` to hold command information, initializes the pulse generator, and uploads all parameters passed to it from a configuration file.

```
Digimite::Digimite(const Params &Parameters):
DigimiteSerial("COM3",57600), command(0), tempbytes(NULL),
deviceidbyte(0xC1),paritybyte(0), zerobyte(0)
{
    tempbytes = new unsigned char[3];
    tempbytes[0]=0;
    tempbytes[1]=0;
    tempbytes[2]=0;
    . . .
```

The `tempbytes` array holds the commands sent to the pulse generator via RS232 in the format Jim Macarthur uses for several of his devices, called the BiasDAC protocol, summarized from his documentation here. In this protocol, all commands consist of at least two bytes, followed by a checksum byte and a zero pad byte.

BYTE 1 (ID) in the form 11dddddd, where dddddd is the target Device ID. Note that the sync bit (bit 7) is set to “1,” indicating the start of a command or status message. The

C/S bit (bit 6) is also set to "1," indicating that the following message is a command to device dddddd.

BYTE 2 (Command) The command 00000000 is reserved; commands in the range of 00000001 to 00111111 are universal commands supported by all devices; commands in the range of 01000000 to 01111111 are device-specific commands, which will be described below for Digimite.

BYTES 3-N (Data) There are 0 to 31 bytes of argument data, depending on the command. If the host is sending information to the device, it sends it here. If the host is requesting information, it sends a string of 0s, which the device replaces with the requested data. A command may both send and request data in the same string. In the present usage for Digimite, data is only sent, and not received.

BYTE N+1 (Parity) for error-checking purposes. The host (i.e. the C++ program) creates this byte by XORing every preceding byte in the command (and setting the MSB to 0). The target (i.e. pulse generator) checks parity by XORing all of the command bytes, including the parity byte (and setting the MSB to 0). The result should be 0. The target replaces this byte with a parity byte generated from its outgoing data stream (which may be different from the incoming stream).

BYTE N+2 (Zero-pad) Indicates status. The host sends a byte set to 0. The selected target replaces it with a status byte (defined below).

The wrapper function `write_data` implements this command protocol in preparing the properly-formatted stream of bytes and sending it to the serial port:

```
void Digimite::write_data(const bool nonzerodata)
{
    DigimiteSerial.Write(&deviceidbyte, 1);
    DigimiteSerial.Write(&command, 1);
    if (nonzerodata == true) {
        DigimiteSerial.Write(tempbytes, 3);
        paritybyte = checksum(3);
    }
    else {
        paritybyte = checksum(0);
    }
    DigimiteSerial.Write(&paritybyte,1);
    DigimiteSerial.Write(&zerobyte,1);
}
```

The checksum is then explicitly calculated using the previously described bitwise operations. Because it is a `private` member function, it can access directly the information in the `tempbytes` array without calling another function, and thus calculates the checksum as follows:

```
unsigned char Digimite::checksum(const int numdatabytes)
{
    unsigned char checksumbyte = 0;
    checksumbyte = deviceidbyte;
    checksumbyte ^= command;
    if(numdatabytes == 3)
    {
        for(int i=0; i<3; i++) {
            checksumbyte ^= tempbytes[i];
        }
    }
    checksumbyte &= 0x7F;
    return checksumbyte;
}
```

The pulse generator command set includes two types of commands: one-byte commands (that pass no data, just control various settings on the CPLD), and three-byte commands (which set parameters that require data). The complete command set, and accompanying source code implementation, is described here.

5.7.1 One-byte Commands

Implementing the one-byte commands is relatively straightforward. The command byte is translated into hexadecimal, and uploaded with the `write_data` command, which takes care of the formatting details.

Reset Controller (01000000) Disables the pulses and sets the internal counters to 0.

```
void Digimite::ResetController()
{
    command = 0x40;
    write_data(false);
}
```

Unreset Controller (01000001) Starts the pulse bursts. Any external reset switches will be logically ORed with the Reset Controller command, so that the controller may be reset either in software or with a front panel switch.

```
void Digimite::UnResetController()
{
command = 0x41;
write_data(false);
}
```

Send Single Piezo Pulse (01001100) Sends a single pulse out of the piezo pulse output, which (assuming piezo is set to TTL trigger) advances one step through the list of displacements. Used for synchronization between pulse counts from the pulse generator, and images received by the program, to make sure everything is in phase (for instance, to account for back steps).

```
    void Digimite::SinglePiezoPulse()
    {
    command = 0x4C;
    write_data(false);
    }
```

Set Slow Delay Mode (01001101) Sets the resolution for delay and shutter timing to 10 milliseconds. This is the default for general PLuTARC operation.

```
    void Digimite::SlowDelayMode()
    {
    command = 0x4D;
    write_data(false);
    }
```

Set Fast Delay Mode (01001110) Sets the resolution for delay and shutter timing to 1 milliseconds. On power up, the unit is in fast delay mode.

```
    void Digimite::FastDelayMode()
    {
    command = 0x4E;
    write_data(false);
    }
```

5.7.2 Three-byte Commands

Implementing the three-byte commands is slightly more complicated. Several shifting and masking bitwise operations are required to split the command bytes into the proper format for uploading via RS232.

```
void Digimite::write3bytes(const long data,
    const int numbits)
{
    if (numbits == 16) {
        tempbytes[0] = (unsigned char) (data >> 14)
            & 0x0003;
    }
    else if (numbits == 20) {
        tempbytes[0] = (unsigned char) (data >> 14)
            & 0x003F;
    }
    tempbytes[1] = (unsigned char) (data >> 7)
        & 0x007F;
    tempbytes[2] = (unsigned char) data & 0x007F;
}
```

Once the `tempbytes` elements are properly formatted, `write_data` takes care of the rest. The three-byte commands are used when numerical parameters must accompany the commands.

Set Pulse Period Time (01000010) Followed by 3 bytes of data which update the period of the pulse chain, in 10-microsecond increments. The first byte of data is 000000aa. The second byte is 0bbbbbbb. The third byte is 0ccccccc. They get combined into 16-bit data of the form aabbbbbbbccccccc. The shortest practical period is set by the pulse width, which is fixed at 1.6 ms. The longest period is $65535 \times 10\ \mu\text{s} = 0.6553$ seconds.

```
void Digimite::SetPulsePeriod(const int pulse_period)
{
    command = 0x42;
    write3bytes(pulse_period, 16);
    write_data(true);
}
```

Set Pulse Count (01000011) Followed by 3 bytes of data which set the number of pulses in each burst. The first byte of data is 000000aa. The second byte is 0bbbbbbb. The third byte is 0ccccccc. They get combined into 16-bit data of the form aabbbbbbbccccccc. The smallest number of pulses allowed is 1. The largest is 65535.

```
void Digimite::SetPulseCount(const int number_of_counts)
{
    command = 0x43;
    write3bytes(number_of_counts, 16);
    write_data(true);
}
```

Set Delay Time (01000100) Followed by 3 bytes of data which set the delay between pulse bursts, in 1 ms increments (fast mode) or 10 ms increments (slow mode). The first byte of data is 00aaaaaa. The second byte is 0bbbbbbb. The third is 0ccccccc. They get combined into 20-bit data of the form aaaaaabbbbbbbccccccc. The smallest allowed delay is 0. The longest delay is 1,013,695 ms, which is 1,013 seconds, or 16.8 minutes.

```
void Digimite::SetStackDelay(const long pulse_delay)
{
    command = 0x44;
    write3bytes(pulse_delay, 20);
    write_data(true);
}
```

Set Shutter On Time (01000101) Followed by 3 bytes of data which set the time between the onset of the inter-burst delay and the activation of the shutter signal, in 1 ms increments (fast mode) or 10 ms increments (slow mode). The first byte of data is 00aaaaaa. The second byte is 0bbbbbbb. The third is 0ccccccc. They get combined into 20-bit data of the form aaaaaabbbbbbbccccccc. The shortest allowed time is 0.

```
void Digimite::SetShutterOnTime(const long
    shutter_on_time)
{
    command = 0x45;
    write3bytes(shutter_on_time, 20);
    write_data(true);
}
```

Set Shutter Off Time (01000110) Followed by 3 bytes of data which set the time between the onset of the inter-burst delay and the deactivation of the shutter signal, in 1 ms increments (fast mode) or 10 ms increments (slow mode). The first byte of data is 00aaaaaa. The second byte is 0bbbbbbb. The third is 0ccccccc. They get combined into 20-bit data of the form aaaaaabbbbbbbccccccc. In typical use, the Shutter Off Time is the Delay Time minus some fixed time interval.

```
void Digimite::SetShutterOffTime(const long
    shutter_off_time)
{
    command = 0x46;
    write3bytes(shutter_off_time, 20);
    write_data(true);
}
```

Set Piezo Holdoff Time (01000111) Followed by 3 bytes of data which set the delay between the start of each main pulse and the start of its corresponding piezo pulse, in 10-microsecond increments. The first byte of data is 000000aa. The second byte is 0bbbbbbb. The third byte is 0ccccccc. They get combined into 16-bit data of the form aabbbbbbbccccccc. The shortest offset is 0. The longest is the value set by the Set Pulse Period command. Piezo pulses are a constant width of approximately 1.6 ms.

```
void Digimite::SetPiezoDelay(const int piezo_delay)
{
    command = 0x47;
    write3bytes(piezo_delay, 16);
    write_data(true);
}
```

These commands are all called in the class constructor, which initializes the pulse generator based on parameters read from an external file. The full constructor listing:

```
Digimite::Digimite(const Params &Parameters):
DigimiteSerial("COM3",57600), command(0), tempbytes(NULL),
deviceidbyte(0xC1), paritybyte(0), zerobyte(0)
{
    tempbytes = new unsigned char[3];
    tempbytes[0]=0;
    tempbytes[1]=0;
    tempbytes[2]=0;

    ResetController();
    SlowDelayMode();
    SetPulsePeriod(Parameters.get_frame_interval_time()
        *100);
    SetPulseCount(Parameters.get_total_number_of_frames());
    SetPiezoDelay((Parameters.get_frame_interval_time()-6)
        *100);
    SetStackDelay(Parameters.get_stack_delay()
        *Parameters.get_slowdelaymode_factor());
    SetShutterOnTime(Parameters.get_laser_on_delay_time()
        *Parameters.get_slowdelaymode_factor());
    SetShutterOffTime((Parameters.get_stack_delay()
        -Parameters.get_laser_off_delay_time())
        *Parameters.get_slowdelaymode_factor());
    ResetController();
}
```

This leads to the enormous simplification that, once the class is properly initialized, the device is already configured and ready for use.

5.8 Integrated 3D Image Stack Acquisition

Integrating all of these pieces and wrappers, the main `PLuTARC.exe` program is relatively short. The `main()` function takes only a configuration file as a command-line parameter, then reads that into a `Parameters` class (which is a gigantic encapsulation of a great deal of housekeeping, and does not need to be discussed further than what is self-evident from the C++ source code). If no configuration file is specified, the program simply resets the piezo and pulse generator (providing a quick way to reinitialize all devices from the command-line directly). Also, at the outset, a timer is started to monitor performance.

```
int main(int argc, char* argv[])
{
starttime = clock();
    Params Params1(argc);
    if (argc < 3) {
        cout << "[No Command-line Options:
            Reset Piezo and Pulse Generator]" << endl;
        Digimite PulseGen;
        PiezoController Piezo;
        return -1;
    }
    static Params Parameters(argc, argv);
```

A logging file is then opened, into which the parameters are read.

```
    ofstream logfile;
    string logfilename = Parameters.get_filestem() +
        "_2Dimages_log.txt";
    logfile.open(logfilename.c_str(),ios::out);
    if(!logfile) {
    cerr << "Cannot open output data log file: make sure
        specified output path actually exists!" << endl;
        return -1;
    }
    Parameters.PrintOutParameters(logfile);
```

Then, all of the hardware is initialized, and date and time are logged:

```
    Digimite PulseGen(Parameters);
    PiezoController Piezo(Parameters);
    QCam_DblBuffer QCam(Parameters, Piezo);
    Piezo.RestartListMotion();
    Delay_via_Ping(2);
    PulseGen.SinglePiezoPulse();
    Delay_via_Ping(2);
    Write_Date_and_Time(cout,
        "Setup completed and data collection initiated");
    Write_Date_and_Time(logfile,
        "Setup completed and data collection initiated");
```

To initiate data acquisition, only two commands are issued. The first tells the pulse generator to start sending pulse trains to the rest of the devices, and the second tells the camera to start collecting images.

```
    PulseGen.UnResetController();
    QCam.Initiate_Queuing();
```

That's it. No other commands need be issued, and acquisition will then proceed until collecting the specified number of stacks, with TIFF files written to disk.

If external programs are run between stacks, either to implement target-locking, or for automated acquisition of periodically-spaced stacks (for instance to create tiled images), a delay comparable to the time it takes to acquire a single stack is created using the `ping` command strategy previously described.

```
    int first_delay_time =
        Parameters.get_total_frame_data_time() / 1000
        - Parameters.get_external_program_exec_time();
    Delay_via_Ping(first_delay_time);
```

Then the external program is run, which periodically polls for the presence of the last frame in the next expected stack. External programs therefore do not rely on precise timing control from PLuTARC, but rather just a rough delay estimate, usually set several seconds short, so that the external program is polling for a little while before PLuTARC writes the final frame to disk.

```
    if(Parameters.get_run_external_program() == true) {
        system(Parameters.Make_cmd_line(
            QCam.get_stack_counter()).c_str());
    }
    while(QCam.get_stack_counter()
            < Parameters.get_total_num_stacks()){
        int centerfind_delay_time =
            Parameters.get_stack_interval_time()
            - Parameters.get_external_program_exec_time();

        Delay_via_Ping(centerfind_delay_time);
        if(Parameters.get_run_external_program() == true) {
            system(Parameters.Make_cmd_line(
                QCam.get_stack_counter()).c_str());
        }
    }
```

Operation of PLuTARC's target-locking, with its several additional programs, is described in chapter 4. An example of a 5×5 tiling of images from a colloidal gel is shown in figure 5.2.

If there is no program called to run between stacks, then the imaging volume will remain fixed and the above code is simply bypassed. After the last stack is collected, the program logs the date and time at completion, closes the log file, and exits.

```
    Write_Date_and_Time(cout,
        "Program successfully terminated");
    Write_Date_and_Time(logfile,
        "Program successfully terminated");
    logfile.close();
return 0;
}
```

The results are 3D stacks of numbered single TIFF images, which are ready for analysis.

5.9 Acknowledgements

The pulse generator hardware was designed by Jim Macarthur, with whom I collaborated to develop its control code. Hidekazu Oki assisted with software development.

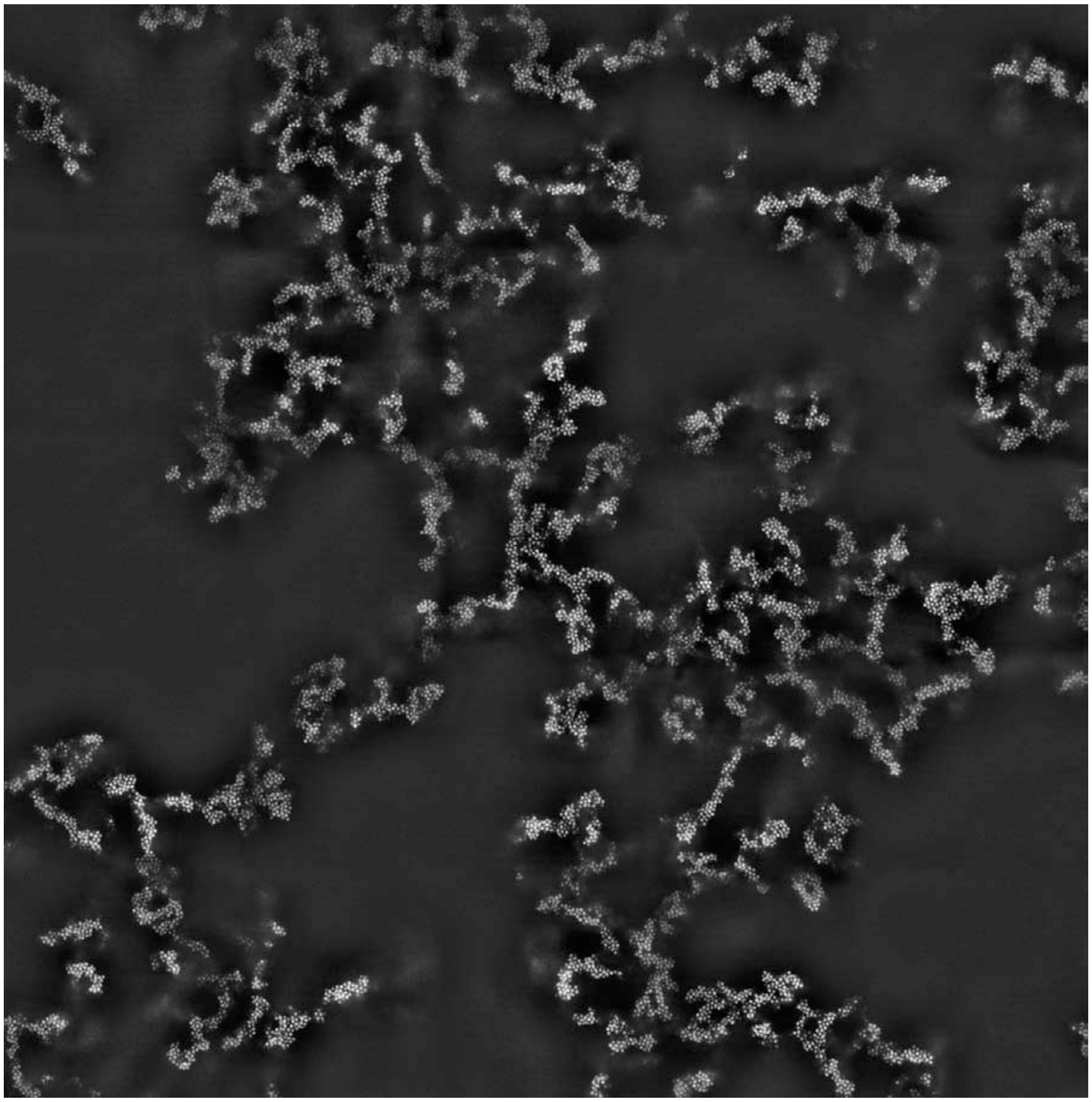

Figure 5.2: 5 × 5 tiling of images of a gel composed of fluorescent spherical colloidal particles.

Chapter 6

Rapid Reconstruction of 3D Particle Configurations

6.1 Overview of Performance Enhancements

I focus in this chapter on the software used to analyze images of fluorescent colloidal spheres, after they are imaged as 3D stacks in the confocal microscope described in chapter 5. This analysis, building on many ideas from the pioneering work of John C. Crocker and David G. Grier [66], forms the basis for the rest of the work in this thesis.

How this processing occurs ultimately controls which quantities can be measured from the data, and how robust and accurate those quantities ultimately can be. For small data sets, accuracy is typically paramount, as additional refinements can be performed—even past the point of diminishing returns—with little actual cost. An extra few minutes of analysis is rarely significantly costly. On the other hand, for large quantities (i.e. terabytes) of image data, performance becomes at least equally important. A fraction of a second per image can sum up to days of additional analysis time for the hundreds of thousands of images I analyzed in the preparation of chapters 9 (published as [171]) and 10 (published as [173]) of this thesis. And fast performance can enable new capabilities, for example the PLuTARC system described in chapter 4 (published as [172]). Inevitably, some tradeoffs have to be made in order to balance the competing priorities of accuracy and performance. This chapter discusses specific technical reasoning behind choices made in selecting and optimizing the software to analyze images of fluorescent colloids from the confocal microscope, for the purpose of locating the 3D position of all of the particles. The reader is urged to first read chapter 4 to get a wider overview of the system.

6.1.1 Language choices

Typically, most image-processing programs involving analysis of colloid-related data are written in a high-level, proprietary language, such as IDL or MATLAB [108]. These languages are ostensibly easier to program in, and often a large bank of well-used (if not necessarily well-tested) programs exists for common scientific calculations, especially within the colloid community; in particular, the original Crocker and Grier particle feature-finding and tracking programs were distributed in IDL.

These proprietary languages, however, have significant limitations:

- There is generally a much smaller body of knowledge, and publicly-available references, when compared with standard programming languages, like C++ or Fortran.
- Distribution of final programs is limited to those individuals able (and willing) to purchase the required commercial software; IDL generally costs thousands of dollars per license.

- Certain users may prefer specific platforms and hardware combinations that may not be compatible with the commercial application.
- Improvments and additions to the language can only be performed by a limited set of developers working for the company producing the proprietary tools, which may be supporting a range of products with fluctuating priorities.

With open standards, especially C++, these problems are largely eliminated. The source code can be compiled, modified and extended on any number of platforms, which almost invariably have free compilers. And there is an extraordinary amount of information available [74, 166], as well as expertise from programmers without specific knowledge of the particular program's scientific function.

6.1.2 Optimizing compilers

The C++ code can also be far more heavily optimized than programs in proprietary languages, such as IDL, because so many tools are commercially available for this specific purpose. Although hand-tuning with inline assembler might be seen as the ultimate in performance maximization, this requires detailed architecture-specific knowledge well beyond the expertise of the author, and is not portable between architectures [104]. Instead, relying on proper vectorizing compilers [27] shifts most of the optimization to the experts, i. e. Intel. This architectural independence has been extremely important to the present code described here. The original program (c. 2002) was written in C++ and originally compiled for the Pentium 4 processor (Netburst architecture) using the Intel Compiler, and used to analyze all of the data in chapters 4 and 9. More recently, the same C++ source code was recompiled for the Intel Core2 Duo processor, directing the Intel compiler to take advantage of the features of the new processor by setting a few simple options. The resulting speed—of the same program, on the same data—*doubled*, when comparing processors at the same clock speed (Xeon Northwood vs. Xeon Woodcrest, both at 2.66 GHz).

This increase in performance, without any additional coding or work, is only possible with properly optimized code and the right compiler. The impressive optimization capabilities, specifically for cache usage and automatic vectorization [27], of the Intel Compiler has led to similar across-the-board performance improvements in all analysis programs, relative to the Microsoft Visual C++ and free (i.e. Gnu gcc) compilers. In many cases, the Intel-optimized programs run several times faster than the "optimized" code created by other compilers.

6.1.3 Image-processing libraries

Additional increases in performance come with the use of the Intel Integrated Performance Primitives (IPP), a library of C++ functions created by Intel to perform common functions and calculations [271]. I used a number of image processing functions in this highly-optimized library, so that the length of the code is actually comparable to that of a higher-level language, like IDL or MATLAB. The IPP library is admittedly somewhat arcane, requiring careful attention to function input parameters to give the correct output. But the benefits are enormous: the highly optimized image processing functions automatically take advantage of multiple processors, hide all of the details of carrying common image-processing operations, and are continually optimized by Intel to leverage the latest processor developments. As with the Intel Compiler, simply updating the software and recompiling—with no additional programmer effort—can lead to significant performance increases on newer processing hardware.

Combined together, these choices and optimizations have allowed the present software, which also underlies the PLuTARC system described in chapter 4, to analyze colloid images more than an order of magnitude faster than the original Crocker and Grier code. For a typical stack of 181 images, each 1000×1000 pixels, the 4-core ($2\times$ dual-core) Woodcrest-processor workstation can locate around ten-thousand particles in approximately 20 seconds; the dual-processor Northwood workstation takes around 45 seconds. By contrast, the IDL code typically exceeds ten minutes—after loading the data, which itself often takes several minutes—and the MATLAB code is even slower.

This can make a qualitative difference to the type of scientific problem able to be pursued. With only tens of seconds required to analyze each stack, it is quite practical to take hundreds, or thousands, of stacks, in the course of an investigation. It also brings the analysis nearly (but not quite) to the threshold of interactivity: the parameter space of the analysis programs can be explored rapidly, then the results quickly applied to the whole data set, maximizing the accuracy of the resulting particle positions. This is simply not feasible if the analysis takes tens of minutes per stack.

Finally, engineering robust and bug-free software is a crucial part of the science undertaken in this thesis. If the analysis programs cannot be counted upon to run, without crashing, errors or instability, when analyzing huge numbers of images, then the total amount of data that can be acquired (and therefore the scientific problems pursued) will be drastically reduced. The code described in this chapter has been well tested over many versions on millions of images—comprising terabytes of data.

6.2 Two-Dimensional Center Location

The confocal microscope system that I constructed and described in chapter 5 creates large numbers of 3D image stacks, storing each 2D slice as a single 8-bit greyscale TIFF. For practically all of the situations described in this thesis, the images were 1000×1000 pixels. In every 3D stack, I typically collected 181 images, each separated by 0.333 μm along the microscope's optic axis, which I assign to be the $\hat{z}$-axis. Therefore, the convention is that each image represents a single $\hat{x} - \hat{y}$ slice.

A typical image of a colloidal gel, acquired by the confocal microscope system described in chapter 5 is shown in figure 6.1. The original image, as collected, is shown in figure 6.1a. I have extracted a small region, surrounded by the red square, and zoomed in by a factor of ten to create the image in figure 6.1b, where contrast has been maximized for clarity. I have also taken that zoomed image, and plotted it in three dimensions (where the height and false color corresponds to image intensity) in figure 6.1c, to further improve visibility. All example images illustrating the various stages of image processing throughout this chapter follow this display convention.

The processing of these types of images in a reliable, accurate and rapid manner is the foundation for the ability to analyze large amounts of data in the confocal. I therefore followed many of the ideas in the original paper by Crocker and Grier [66], basing the code on what they originally created in the IDL language, but with several modifications to achieve the significant performance increases described above.

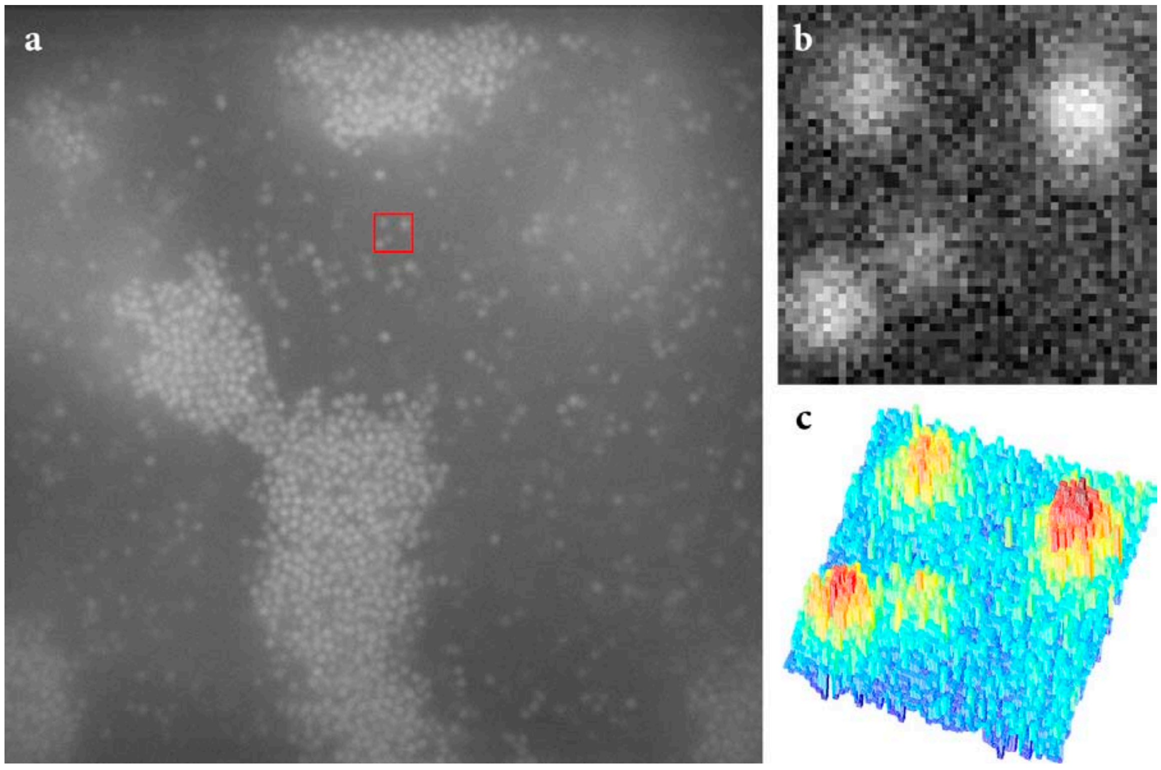

Figure 6.1: Typical original image of a gel composed of fluorescent spherical colloidal particles. **(a)** Original image, as collected in the confocal microscope, at left. A small square region, outlined by a red square at left, is enlarged by a factor of ten and shown in **(b)**, where contrast is maximized for clarity. **(c)** The image data in the zoomed image, rendered in three dimensions; height and false color scale correspond to pixel intensity in the original image. The image's noise, at the length-scale of a few pixels, is particularly apparent in this representation, where it makes the "bumps" representing the particles are quite rough. Subsequent figures in this chapter, showing intermediate stages of processing, are all based on this original image, and follow the same display convention.

6.2.1 Departures from the Crocker and Grier algorithm

One significant departure from the Crocker and Grier algorithm is to locate the 2D centers of particles in each image, generating a text file with (x, y) particle positions for each image. These positions are then linked up, processing the *text* data, into 3D (x, y, z) positions. As described in chapter 4, the main reason for this choice is performance. Only one image must be loaded into memory at a time, so that the memory footprint is only a few megabytes; moreover, most image processing routines are optimized for these 2D image arrays (not 3D data), in particular the single-instruction multiple-data (SIMD) image processing operations within the IPP [27, 104].

Another major change is to decouple some of the correlation of internal control parameters, relative to the `pretrack.pro` IDL programs that represent the commonly-used implementations of the Crocker and Grier algorithms. Certain mask and filter sizes were automatically set to be equal in the wrapper programs; by "unbundling" these parameters, flexibility (in terms of imaging conditions under which particles can be located accurately) is greatly increased at the expense of only a couple more parameters to input to the program.

In addition, I added thresholding at several stages, to discard low-intensity noise. The Crocker and Grier algorithm identifies particles by locating local intensity maxima (bright spots). By thresholding to discard noisy low-intensity data, spurious identifications of local maxima are dramatically reduced.

Finally, for convenience, I combined the functions of two IDL programs, `bpass.pro` and `feature.pro` into a single C++ executable, `plu_Centerfind.exe`. This program loads a single TIFF image, performs a number of processing operations, then locates the particle positions in 2D. Its output is a text file, with the image's unique slice and stack number, then the $\hat{x}$- and $\hat{y}$-coordinates (in units of pixels), as well as brightness and radius-of-gyration quantities, for each identified particle center.

6.2.2 Encapsulation in the Image2D container class

The `plu_Centerfind.exe` program consists of several subfiles. All images are handled as a class called `Image2D`, described in `image.cpp`. This class handles the loading of the original image, saving of the processed images, and has member functions to return the image attributes, such as size parameters, declared as inline functions.

```
class Image2D
{
public:
    Image2D();
    Image2D(const int image_length, const int image_width);
    ~Image2D();

    Ipp32f *get_image2D() const {return imagedata;}
    int get_stepsize() const {return stepsize;}
    int get_width() const {return width;}
    int get_length() const {return length;}
    int get_height() const {return length;}
    int get_numberofpixels() const {return numberofpixels;}
    IppiSize get_ROIfull() const {return ROIfull;}
    FREE_IMAGE_FORMAT get_freeimageinfo() const
        {return freeimageinfo;}
```

Other member functions convert the coordinates of pixels in a 2D array to a 1D list coordinate, to facilitate some imaging operations that only require 1D data (like average brightness, for instance).

```
int Image2D::getx(const int index1D)
{
    return index1D % width;
}
```

```
int Image2D::gety(const int index1D)
{
    return (int) floor((float) index1D / width);
}

int Image2D::get1Dindex(const int x, const int y)
{
    return y * width + x;
}
```

Finally, the class transparently converts from the FreeImage format (which loads the image from disk) to and from the IPP format (which is needed for all of the accelerated processing functions). Because this operation is somewhat idiosyncratic (with some experimentation necessary to get the images to load correctly), I include the entire source code here. There are a number of intermediate format conversions that are necessary, but do not have any noticeable impact on performance. While a cleaner code is probably possible, this was the most efficient way I could find to do the conversions properly:

```
Image2D TIFF_to_IPP(const string infilename)
{
    IppStatus status;
    //open TIFF files
    FIBITMAP * dib = NULL;
    FREE_IMAGE_FORMAT fif =
        FreeImage_GetFIFFromFilename(infilename.c_str());
    dib = FreeImage_Load(fif, infilename.c_str());

    //Initialize Images
    int width = FreeImage_GetWidth(dib);
    int height = FreeImage_GetHeight(dib);

    //even for grayscale, need to use 24-bit images (R=G=B)
    int pitch8 = 4 * ceil ((float) width / 4);
    int pitch24 = pitch8 *3;
    int size8 =  pitch8 * height;
    int size24 = 3 * size8;
    //Create 24-bit IPP image
    Ipp8u *image8 = ippsMalloc_8u(size8);
    Ipp8u *image24 = ippsMalloc_8u(size24);
    IppiSize roi = {width, height};

    FreeImage_ConvertToRawBits
        (image24, dib, pitch24, 24, 255, 255, 255, true);

    //create IPP image to receive TIFF data
    Image2D image_in(height, pitch8);
    status = ippiCopy_8u_C3C1R
        (image24, pitch24, image8, pitch8, roi);

    //scale 8-bit IPP data to 32-bit floating-point IPP data
    status = ippiScale_8u32f_C1R(image8,pitch8,
        image_in.get_image2D(), image_in.get_stepsize(),
        image_in.get_ROIfull(),0,255);

    image_in.set_freeimageinfo(fif);

    //Cleanup memory
    FreeImage_Unload(dib);
    dib = NULL;

    ippsFree(image8);
    ippsFree(image24);
    image8 = NULL;
    image24 = NULL;
    return image_in;
}

IppStatus IPP_to_TIFF(string out_filename,
    Image2D &image_out)
{
    IppStatus status;
    Ipp8u *ippi_out8 =
        ippsMalloc_8u(image_out.get_numberofpixels());
    Ipp8u *ippi_out24 =
        ippsMalloc_8u(3 * image_out.get_numberofpixels());
    int step8 = image_out.get_width();
    int step24 = 3 * step8;
    int width = image_out.get_width();
    int height = image_out.get_height();
    IppiSize roi = {width, height};
    Ipp32f maxbrightness = 0;

    //scale 32-bit floating point image to 8-bit integer
    status = ippiMax_32f_C1R(image_out.get_image2D(),
        image_out.get_stepsize(), image_out.get_ROIfull(),
        &maxbrightness);
    status = ippiScale_32f8u_C1R(image_out.get_image2D(),
        image_out.get_stepsize(), ippi_out8,step8,
        image_out.get_ROIfull(), 0, maxbrightness);

    //copy each 8-bit image to a separate channel in
    //a 24-bit image, such that R = G = B
    status = ippiCopy_8u_C1C3R(ippi_out8,
        step8, ippi_out24, step24, roi);
    status = ippiCopy_8u_C1C3R(ippi_out8,
        step8, ippi_out24+1, step24, roi);
    status = ippiCopy_8u_C1C3R(ippi_out8,
        step8, ippi_out24+2, step24, roi);

    //Convert to FreeImage format and reduce back to 8-bits
    FIBITMAP *dib24 = NULL, *dib8 = NULL;

    dib24 = FreeImage_ConvertFromRawBits(ippi_out24,
        width, height, step24, 24, 255, 255, 255, true);
    dib8 = FreeImage_ConvertTo8Bits(dib24);

    if(dib8 != NULL) {
        FreeImage_Save(image_out.get_freeimageinfo(),
            dib8, out_filename.c_str());
    }

    //Cleanup memory
    FreeImage_Unload(dib8);
    FreeImage_Unload(dib24);
    dib8 = NULL;
    dib24 = NULL;

    ippsFree(ippi_out8);
    ippsFree(ippi_out24);
    ippi_out8 = NULL;
    ippi_out24 = NULL;

    return status;
}
```

After each image is loaded into an `Image2D` class, a series of operations is performed by the function `find_centers_2D()`.

6.2.3 2D Image Filtering

Because the feature-location algorithm detects isolated local maxima in an image, the first step is to process each image so that the fluorescent colloids (or other bright objects) appear as a bright, well-separated, circular objects on a black background. These operations are handled by the `BandPass_2D` function in the `data_proc1.cpp` file.

```
IppStatus BandPass_2D(Image2D &image_in,
    Image2D &image_bandpassed, const int feature_radius,
    const int hwhm_length)
{
    RecenterImage(image_in);
```

Any image can be expressed as $I(x, y)$, the intensity (in this case, an 8-bit integer greyscale level from 0 to 255) with discrete pixel coordinates (x, y), where $0 < x < x_{\max}$ and $0 < y < y_{\max}$. After loading, the input data from the original image file, the function first converts the intensities I to floating-point numbers, and rescales their values to the range from 0 to 100.

```
void RecenterImage(Image2D &image)
{
    IppStatus status;
    Ipp32f minval = 0, maxval = 0;
    status = ippiMinMax_32f_C1R(image.get_image2D(),
        image.get_stepsize(), image.get_ROIfull(),
        &minval, &maxval);
    status = ippiAddC_32f_C1IR(-minval, image.get_image2D(),
        image.get_stepsize(), image.get_ROIfull());
    status = ippiMulC_32f_C1IR(100 / (maxval-minval),
        image.get_image2D(), image.get_stepsize(),
        image.get_ROIfull());
}
```

Several filter operations follow, which are implemented as *convolutions*. These involve successive multiplications of image pixels by a *kernel*, a separate, fixed intensity distribution $K(i, j)$, with its local discrete pixel coordinates (i, j), for which $-r_i < i < r_i$ and $-r_j < j < r_j$. In general, the kernel is much smaller than the image, i.e. $r_i \ll x_{\max}$ and $r_j \ll y_{\max}$. In a convolution operation, the output image $I_o(x, y)$ is generated by multiplying the neighborhood around each pixel in the original input image by the kernel:

$$I_o(x, y) = \sum_{x'=-r_i}^{r_i} \sum_{y'=-r_i}^{r_i} I(x - x', y - y')K(x', y') \tag{6.1}$$

The intensity of each *individual* pixel $I_o(x, y)$ in the output image is the product of a multiplication of the kernel with *many* pixels surrounding the pixel in the input image at the same (x,y) location [108].

In the present case, two one-dimensional convolution kernels, tophat and gaussian, are created, with sizes determined by the command-line parameters.

```
IppStatus BandPass_2D(Image2D &image_in,
    Image2D &image_bandpassed, const int feature_radius,
    const int hwhm_length)
{
    . . .
    Gaussian_Kernel GaussKernel(feature_radius, hwhm_length,
        image_in.get_width(), image_in.get_length());
    Tophat_Kernel TopHatKernel(feature_radius,
        image_in.get_width(), image_in.get_length());
    int number_of_pixels = image_in.get_numberofpixels();
    int step_size = image_in.get_stepsize();
```

These kernels are created in two classes contained in `kernel1.cpp`, correspondingly named `Gaussian_Kernel` and `Tophat_Kernel`. The initialization of these kernels, generating the arrays representing their values, is handled in the constructors to the two classes.

Gaussian Kernel

The purpose of filtering by means of a convolution with a gaussian kernel is to remove high-frequency spatial noise, typically a few pixels. The two parameters characterizing the gaussian are its radius (the overall kernel radius) r, limiting the distance out to which neighboring pixels are included in the convolution's multiplication, and the "fatness" of the gaussian, expressed as a half-width, half-maximum distance r_h. Both are expressed in units of pixels, set with command-line parameters, and passed to the class constructor. The size of the gaussian kernel should be chosen to be about the typical size of the fluorescent particles as in the original image, i.e. $r \approx a$. By sizing the kernel using the diameter, calculated as

$$d = 2r + 1 \tag{6.2}$$

integer values are guaranteed for all size parameters. Also, in order to get the proper gaussian for specified r_h, the value specified in pixels must be divided by a factor α_g, which can easily be derived:

$$e^{-\left(r_\mathrm{h}/\alpha_\mathrm{g}\right)^2} = \frac{1}{2} \tag{6.3}$$

so that,

$$\begin{aligned} \alpha_\mathrm{g} &= \frac{r_\mathrm{h}}{\sqrt{\log 2}} && (6.4) \\ &= \frac{r_\mathrm{h}}{0.8325546} && (6.5) \end{aligned}$$

The properly-sized 1D gaussian function is then created as an array, which is then properly sized and thresholded (low values away from the peak are set to zero, since they don't influence any numerical calculation and simply add unnecessary performance-decreasing calculations).

```
Gaussian_Kernel::Gaussian_Kernel(const int radius,
    const float HWHM, const int tif_width,
    const int tif_length): anchor_point(0), offset(0),
    kernel(NULL), kernel_length(0)
{
    //Gaussian kernel is a one-dimensional array
    //kernellength is an odd number
int kernel_length_big = 2 * radius + 1;
    int i=0, j=0;
    Ipp32f *gausskernel_big =
        ippsMalloc_32f((int) kernel_length_big);
    Ipp32f total = 0, delta_x = 0;

    const int gaussian_divisor = HWHM / 0.8325546;
    for(i=0; i<kernel_length_big; i++) {
        delta_x = (Ipp32f) (i-radius)/gaussian_divisor;
        gausskernel_big[i] = exp(-delta_x * delta_x);
        total += gausskernel_big[i];
    }
    for(i=0; i<kernel_length_big; i++) {
        gausskernel_big[i] /= total;
    }

    //keep only that part of the kernel
    //that is greater than threshold
    int k = 0;
    i=0;
    while(gausskernel_big[i] <= epsilon) {
        k=i;
        i++;
    }
    kernel_length = kernel_length_big - 2 * k - 2;
    kernel = ippsMalloc_32f((int) kernel_length);
    for(j=0; j < kernel_length; j++) {
        kernel[j] = gausskernel_big[j+k+1];
    }
    . . .
}
```

The desired 2D kernel effect is a circularly-symmetric gaussian. This is an example of a separable kernel, meaning that the multiplications can equivalently be carried out by multiplying two 1D gaussians in $\hat{x}$ and $\hat{y}$ and combining the results, since:

$$e^{-r^2} = e^{-x^2-y^2} = e^{-x^2} \cdot e^{-y^2} \tag{6.6}$$

The two successive 1D multiplications involve far fewer operations—and are therefore far faster—than the direct 2D gaussian multiplication.

```
IppStatus BandPass_2D(Image2D &image_in,
    Image2D &image_bandpassed,
    const int feature_radius, const int hwhm_length)
{
    . . .
    status = ippiFilterColumn_32f_C1R(
        image_in.get_image2D() + GaussKernel.get_offset(),
        step_size, image_gauss_col.get_image2D() +
        GaussKernel.get_offset(), step_size,
        GaussKernel.get_ROI_size(),
        GaussKernel.get_gaussian_kernel(),
        GaussKernel.get_kernel_length(),
        GaussKernel.get_anchor_point());
    status = ippiFilterRow_32f_C1R(
        image_gauss_col.get_image2D() +
        GaussKernel.get_offset(),
        step_size, image_gauss_rowcol.get_image2D() +
        GaussKernel.get_offset(), step_size,
        GaussKernel.get_ROI_size(),
        GaussKernel.get_gaussian_kernel(),
        GaussKernel.get_kernel_length(),
        GaussKernel.get_anchor_point());
```

The result of processing the example in figure 6.1 with a 1D Gaussian filter with a radius of 14 pixels and a HWHM radius of 3 pixels in the vertical direction is shown in

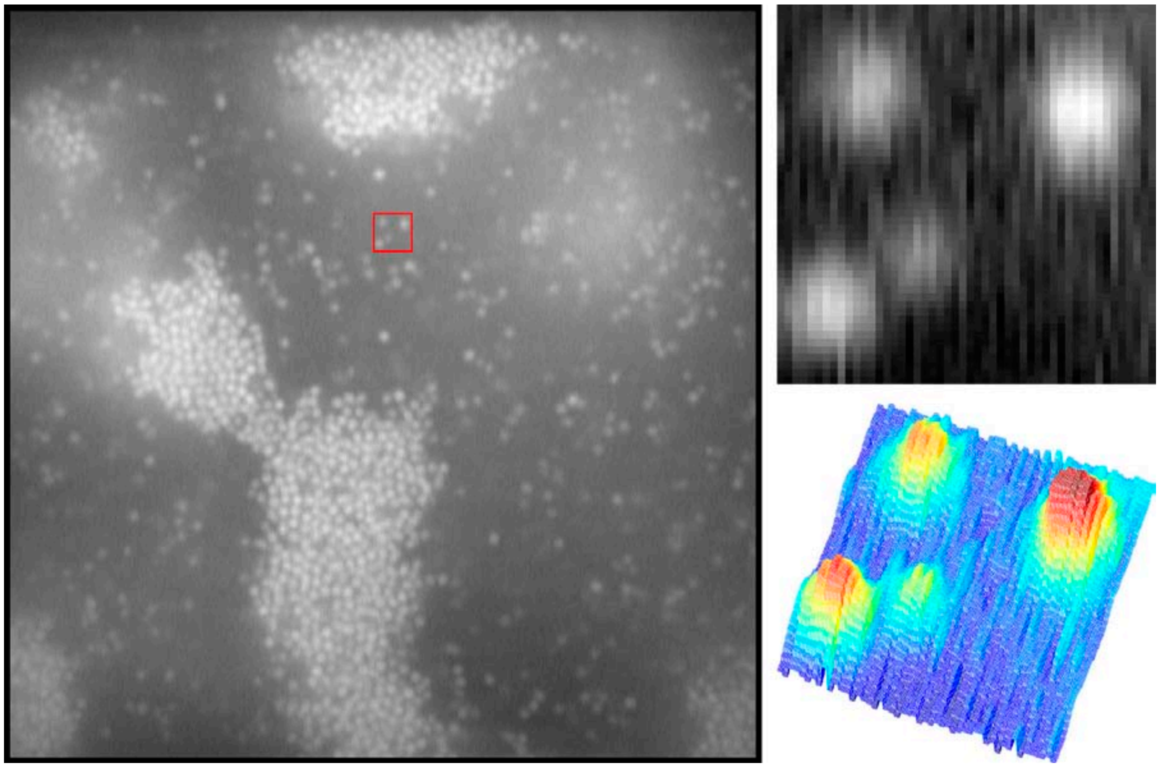

Figure 6.2: Result of processing the gel image in figure 6.1 with a 1D Gaussian filter in the vertical direction. Note that the image is smoothed along the vertical direction, while still very rough in the horizontal direction; this is particularly apparent in the 3D intensity plot at lower right.

figure 6.2. Processing that image with the same 1D Gaussian but in the horizontal direction (yielding the result of applying 2D Gaussian filtration to the original image in figure 6.1) is shown in figure 6.3.

Tophat Kernel

The purpose of the tophat kernel is to remove large-scale (low-frequency) variation in the background of the image, at length-scales greater than that of a particle. The filter operation, alternatively known as a box filter, is a very strong blurring that wipes out particles, leaving just the intensity background. In this case, the radius is set to the same total radius

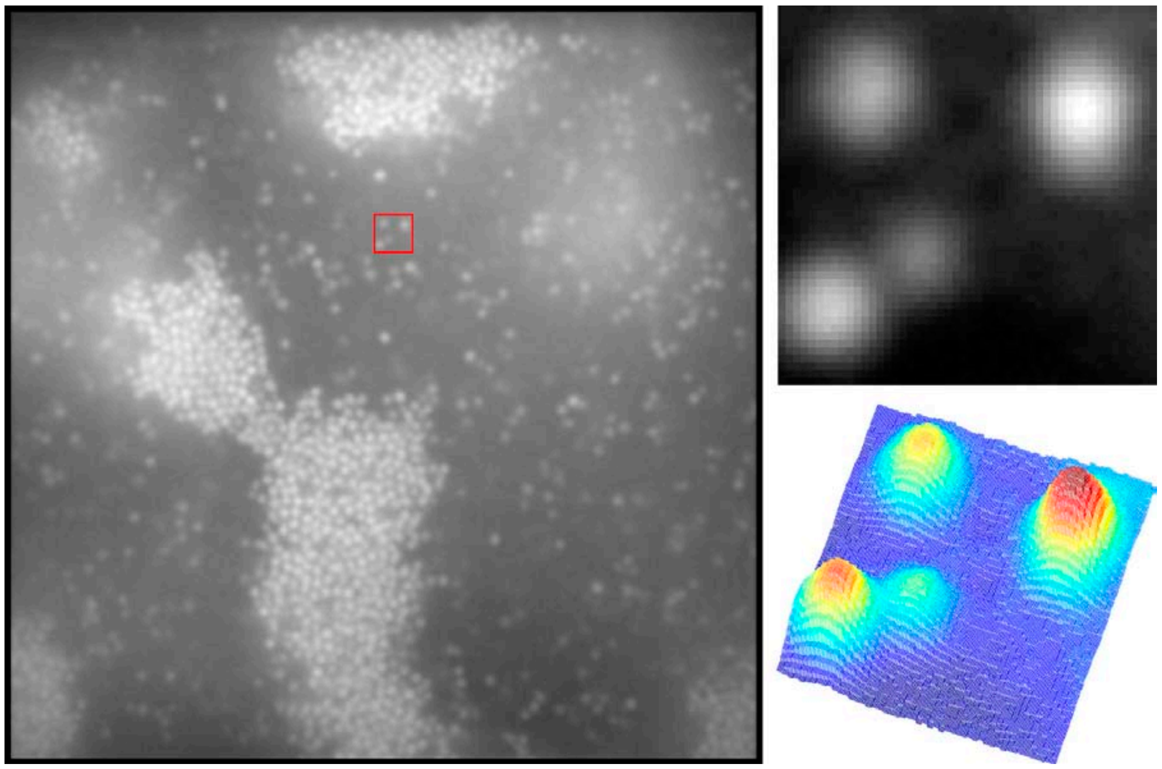

Figure 6.3: Result of processing the gel image in figure 6.2 with a 1D Gaussian filter in the horizontal direction, completing the 2D Gaussian filtration of the image in figure 6.1. Noise has been drastically reduced, so that the particles appear round and even.

as the gaussian kernel. But since this is the only parameter affecting the tophat kernel, the constructor code for its creation is consequently far simpler.

```
Tophat_Kernel::Tophat_Kernel(const int radius,
    const int tif_width, const int tif_length): offset(0)
{
    int maskwidth = 2 * radius + 1;
    int delta = maskwidth - 1;

    //construct regions for mask and ROI
    ROI_size.width = tif_width - delta;
    ROI_size.height = tif_length - delta;
    mask_size.width = maskwidth;
    mask_size.height = maskwidth;

    anchor_point.x = radius;
    anchor_point.y = radius;
    offset = (anchor_point.y*tif_width+anchor_point.x);
}
```

This constructor calculates a few numerical parameters to pass to the IPP `FilterBox()` function, which does the convolution without requiring an explicit data structure to hold the tophat kernel:

```
IppStatus BandPass_2D(Image2D &image_in,
    Image2D &image_bandpassed, const int feature_radius,
    const int hwhm_length)
{
    . . .
    status = ippiFilterBox_32f_C1R(image_in.get_image2D() +
        TopHatKernel.get_offset(), step_size,
        image_tophat.get_image2D() +
        TopHatKernel.get_offset(), step_size,
        TopHatKernel.get_ROI_size(),
        TopHatKernel.get_mask_size(),
        TopHatKernel.get_anchor_point());
```

Processing the original image in figure 6.1 with a tophat kernel of radius 14 pixels (the same radius as for the gaussian filter operations used to create the images in figures 6.2 and

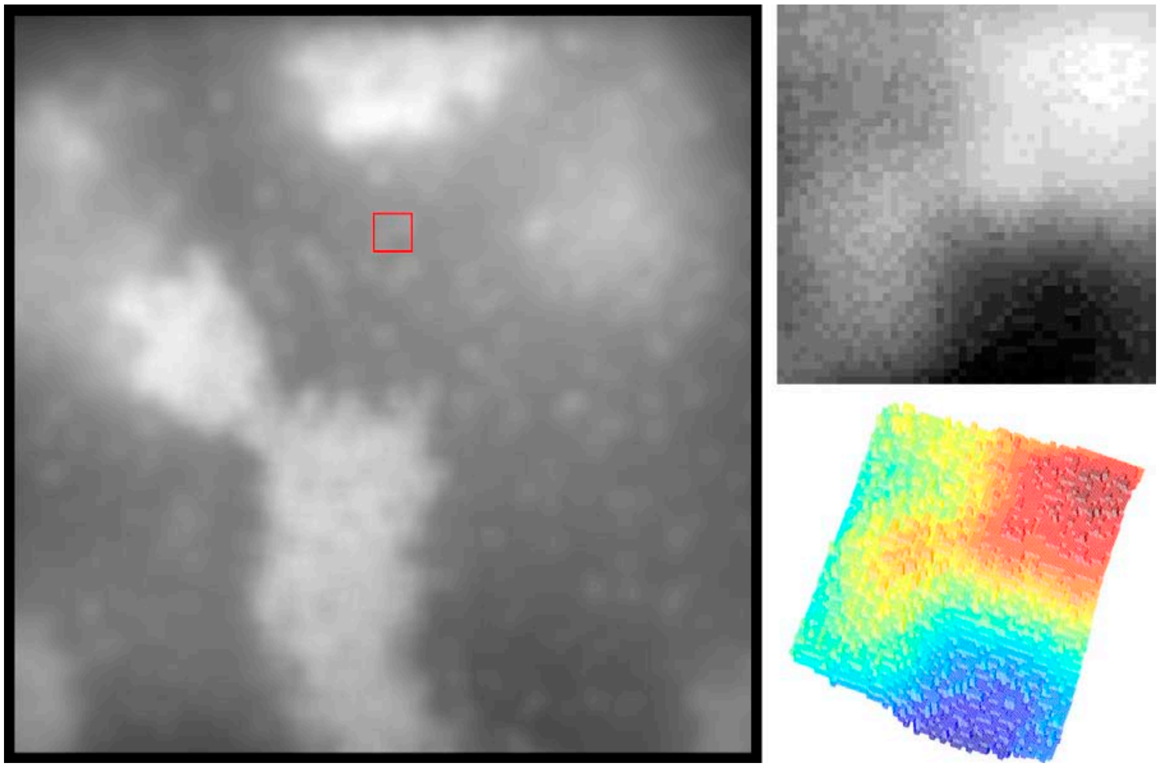

Figure 6.4: Result of processing the gel image in figure 6.1 with a tophat, or box, filter, severely blurring out particle features and leaving the background intensity.

6.3) yields a far blurrier image, shown in figure 6.4.

The tophat-filtered image in figure 6.4 is subtracted from the gaussian-filtered image in figure 6.3 to remove the background, then thresholded to remove negative values.

```
IppStatus BandPass_2D(Image2D &image_in,
    Image2D &image_bandpassed, const int feature_radius,
    const int hwhm_length)
{
    . . .
    status = ippiSub_32f_C1R(image_tophat.get_image2D()
        + TopHatKernel.get_offset(), step_size,
        image_gauss_rowcol.get_image2D() +
        TopHatKernel.get_offset(), step_size,
        image_bandpassed.get_image2D() +
        TopHatKernel.get_offset(), step_size,
        TopHatKernel.get_ROI_size());
```

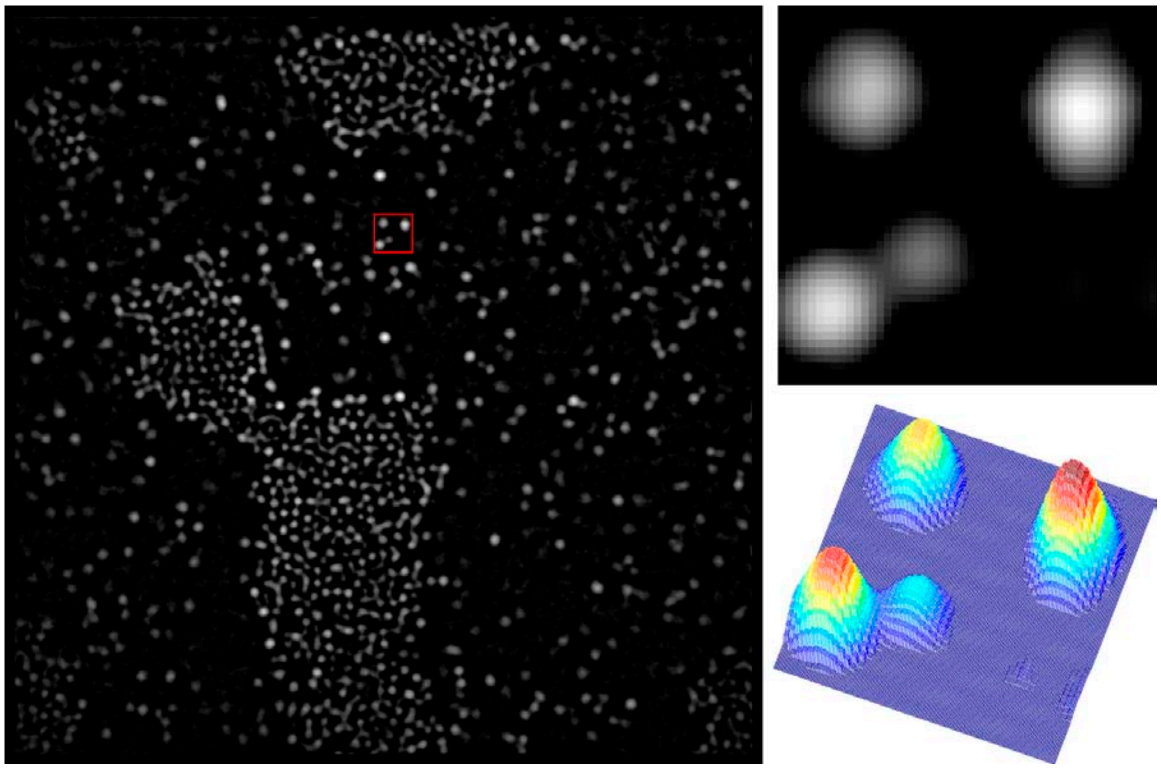

Figure 6.5: Result of subtracting the tophat-filtered image in figure 6.4 from the gaussian-filtered image in figure 6.3, then thresholding the image to remove negative values. This yields a filtered image where the particles are well-separated circles, brightest in the center, on an even black background.

```
    //cutoff values below zero
    status = ippiThreshold_LTVal_32f_C1IR(
        image_bandpassed.get_image2D() +
        TopHatKernel.get_offset(), step_size,
        TopHatKernel.get_ROI_size(), 0, 0);
```

This yields a filtered image where the particles are well-separated circles, brightest in the center, on an even black background, as shown in figure 6.5. Although some call this image "bandpassed," because both long- and short-wavelength noise have been suppressed, true bandpass filtering typically involves a gaussian kernel applied in Fourier space [49], and so the designation is not strictly applicable here.

6.2.4 Locating Local Maxima

Locating the local maxima occurs in the `FindLocalMax_2D()` function in the source file `data_proc1.cpp`, and begins with thresholding the processed image (shown in figure 6.5), to remove the low-intensity noise containing local maxima that could be mistaken for particles. The threshold value is set by a command-line parameter, and sets all pixels lower than a certain percentage to zero.

```
IppStatus FindLocalMax_2D(Image2D &image_bpass,
    Image2D &image_bpass_thresh, Image2D &image_subtracted,
    const int intensity_threshold,
    const int dilation_radius)
{
    RecenterImage(image_bpass);
    status = ippiThreshold_LTVal_32f_C1R(
        image_bpass.get_image2D(),
        image_bpass.get_stepsize(),
        image_bpass_thresh.get_image2D(),
        image_bpass_thresh.get_stepsize(),
        image_bpass.get_ROIfull(),
        intensity_threshold, intensity_threshold);
```

Applying this threshold to the image in figure 6.5 leads to the image shown in figure 6.6. The resulting image is then *dilated*, which for each pixel checks all surrounding pixels, and sets that pixel to the brightest value in the neighborhood. The neighborhood is defined as a circle of radius R_D, set at the command line in units of pixels. The dilated image, then, is:

$$I_\text{D}(x,y) = \max\left\{I(x-x', y-y')\ s.t.\ \left(x-x'\right)^2 + \left(y-y'\right)^2 < R_\text{D}^2\right\} \tag{6.7}$$

That is, each pixel in the dilated image at $I_\text{D}(x,y)$ is the brightest value from the set of pixels $I(x-x', y-y')$ surrounding the pixel at the same location in the original image at (x,y), such that $(x-x')^2 + (y-y')^2 < R_\text{D}^2$. This dilation operation expands, or dilates, bright regions in the image, and is described in greater detail in a number of references [108,

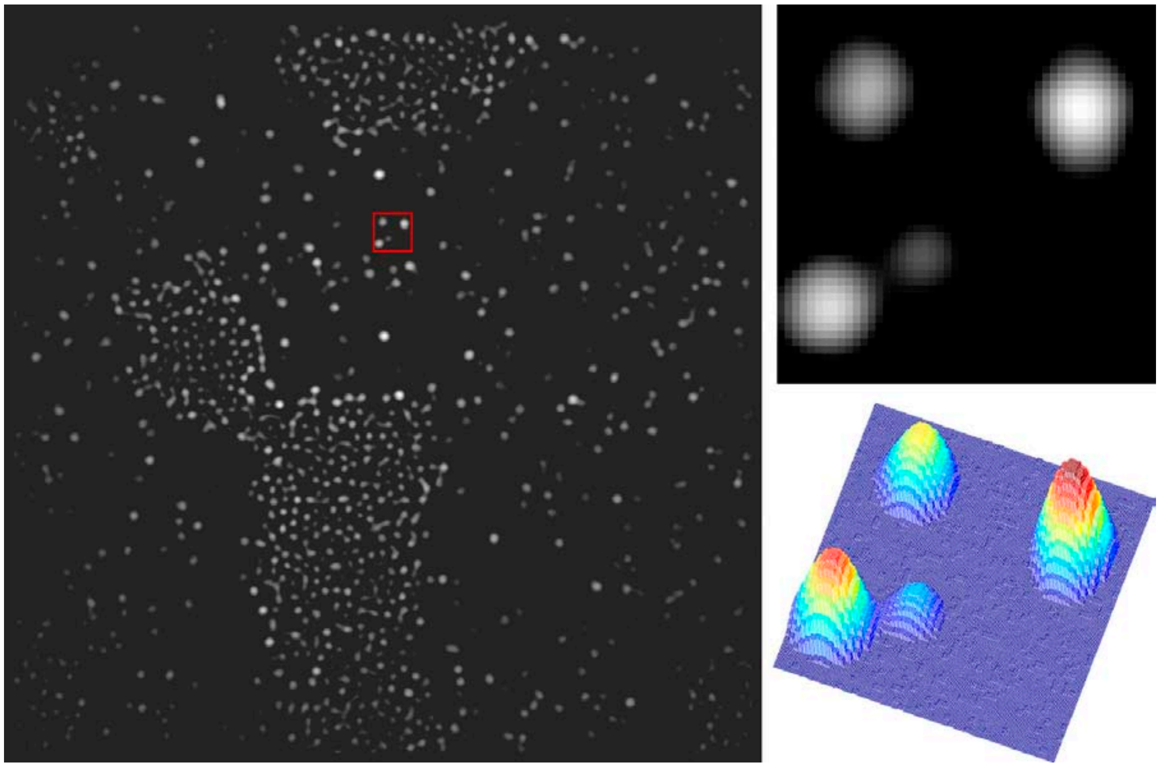

Figure 6.6: Result of thresholding the image in figure 6.5 to remove low-intensity background noise that could interfere with the identification of local maxima.

49, 19, 201]. Note that this operation requires more computation time than the rest of the steps in the image-processing protocol *combined*. Therefore, selecting a small dilation radius R_D is crucial for enabling good performance.

For the present purpose, the local maxima (i.e. particle centers) occur at pixel locations where the original image and the dilated image have the same value—so long as the dilation radius is set to a reasonable value smaller than the particle radius. The dilation kernel is like the other kernels implemented as a class in `kernel1.cpp` and created in class constructor. The desired kernel has a constant value of 1 within a circle with radius set at the command line, and zero elsewhere. The simplest and fastest way to create this is to generate a mask where the intensity of each 2D element is proportional to the square of the radius away from the center, then use an intensity threshold on the mask to select which pixels are to be set to non-zero values (as opposed to calculating, for each pixel, a distance threshold from the center):

```
Dilation_Kernel::Dilation_Kernel(const int radius,
    const int tif_width, const int tif_length):
    dilation_kernel(NULL), offset(0)
{
    IppStatus status;

    int diameter = 2 * radius + 1;
    int area = diameter * diameter;
    int ramp_step = 2 * diameter;
    int dilation_kernel_step = diameter;
    IppiSize ramp_roi_size = {diameter, diameter};

    //allocate memory for kernel and ramps
    Ipp16s *ramp_x = ippsMalloc_16s(area);
    Ipp16s *ramp_y = ippsMalloc_16s(area);
    Ipp16s *circle = ippsMalloc_16s(area);
    dilation_kernel = ippsMalloc_8u(area);

    //create ramps from -radius to +radius
    //for rˆ2 and circular masks
    status = ippiImageRamp_16s_C1R(ramp_x, ramp_step,
        ramp_roi_size, -radius, 1, ippAxsHorizontal);
    status = ippiImageRamp_16s_C1R(ramp_y, ramp_step,
        ramp_roi_size, -radius, 1, ippAxsVertical);

    //square the ramps and then finally add
    status = ippiSqr_16s_C1IRSfs(ramp_x,
        ramp_step, ramp_roi_size, 0);
    status = ippiSqr_16s_C1IRSfs(ramp_y,
        ramp_step, ramp_roi_size, 0);
    status = ippiAdd_16s_C1RSfs(ramp_x, ramp_step,
        ramp_y,ramp_step,circle,ramp_step,ramp_roi_size,0);

    //trim away values greater than radiusˆ2
    //for circle kernel
    status = ippiThreshold_GTVal_16s_C1IR(circle,
        ramp_step, ramp_roi_size,radius * radius, -1);
    status = ippiThreshold_GTVal_16s_C1IR(circle,
        ramp_step, ramp_roi_size,-1, 1);
    status = ippiThreshold_LTVal_16s_C1IR(circle,
        ramp_step, ramp_roi_size,1, 0);

    //Convert kernel to 8-bit unsigned integer
    status = ippiConvert_16s8u_C1R(circle, ramp_step,
        dilation_kernel,dilation_kernel_step,ramp_roi_size);
    . . .
}
```

This kernel is then used to dilate the processed image. Selecting a dilation radius of 3 pixels results in the image shown in figure 6.7

```
IppStatus FindLocalMax_2D(Image2D &image_bpass,
    Image2D &image_bpass_thresh, Image2D &image_subtracted,
    const int intensity_threshold,
    const int dilation_radius)
{
    . . .
    status = ippiSet_32f_C1R(intensity_threshold,
        image_dilated.get_image2D(),
        image_dilated.get_stepsize(),
```

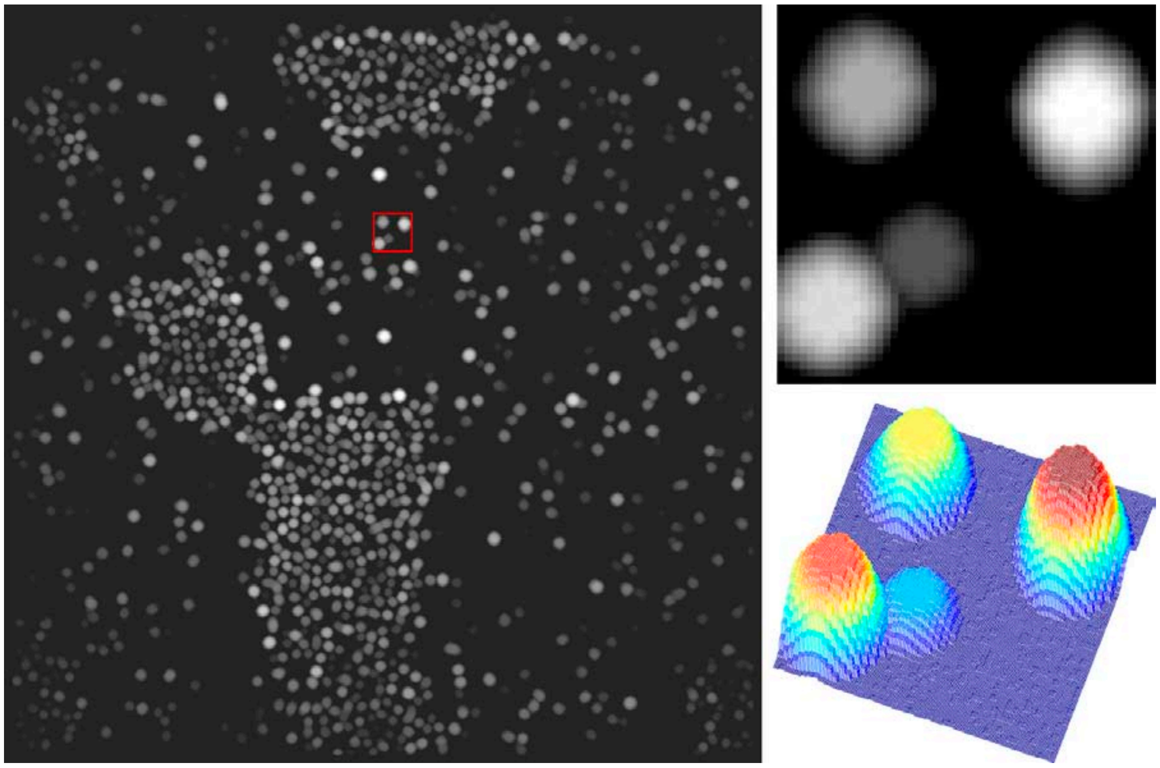

Figure 6.7: Result of dilating the image in figure 6.6. Particles appear as flattened, round features.

```
        image_dilated.get_ROIfull());
    status = ippiDilate_32f_C1R(
        image_bpass_thresh.get_image2D() +
        DilationKernel.get_offset(),
        image_bpass_thresh.get_stepsize(),
        image_dilated.get_image2D() +
        DilationKernel.get_offset(),
        image_dilated.get_stepsize(),
        DilationKernel.get_ROI_size(),
        DilationKernel.get_dilation_kernel(),
        DilationKernel.get_mask_size(),
        DilationKernel.get_anchor_point());
```

To identify those pixels that have the same intensity in both the filtered image of figure 6.6 and the dilated image in figure 6.7, the two images are subtracted.

```
    status = ippiSub_32f_C1R(
        image_dilated.get_image2D(),
        image_dilated.get_stepsize(),
        image_bpass.get_image2D(),
        image_bpass.get_stepsize(),
        image_subtracted.get_image2D(),
        image_subtracted.get_stepsize(),
        image_bpass.get_ROIfull());
```

Subtracting the dilated image from the filtered image yields a difference image with occasional zero values, corresponding to local maxima, each surrounded by pixels with negative values. To accentuate these differences by increasing the brightness of any areas where the two are not exactly identical, this difference image is exponentiated, resulting in the image shown in figure 6.8.

```
    status = ippiExp_32f_C1IR(
        image_subtracted.get_image2D(),
        image_subtracted.get_stepsize(),
        image_subtracted.get_ROIfull());
```

The brightest pixels here represent values of $e^0 = 1$, that is, places where the dilated and filtered image are identical, as shown in figure 6.8. Surrounding pixels will have negative values, which when exponentiated, will lead to brightness values strictly below 1. Preserving only the white pixels is easily accomplished by thresholding, setting all pixels with a brightness value less than one to zero.

```
    status =
        ippiThreshold_LTValGTVal_32f_C1IR(
        image_subtracted.get_image2D(),
        image_subtracted.get_stepsize(),
        image_subtracted.get_ROIfull(),
        1-epsilon, 0, 1-epsilon, 1);
```

This yields a final image that is a totally black, with isolated single white pixels representing the local maximum for each particle, as shown in figure 6.9. These white dots are overlaid

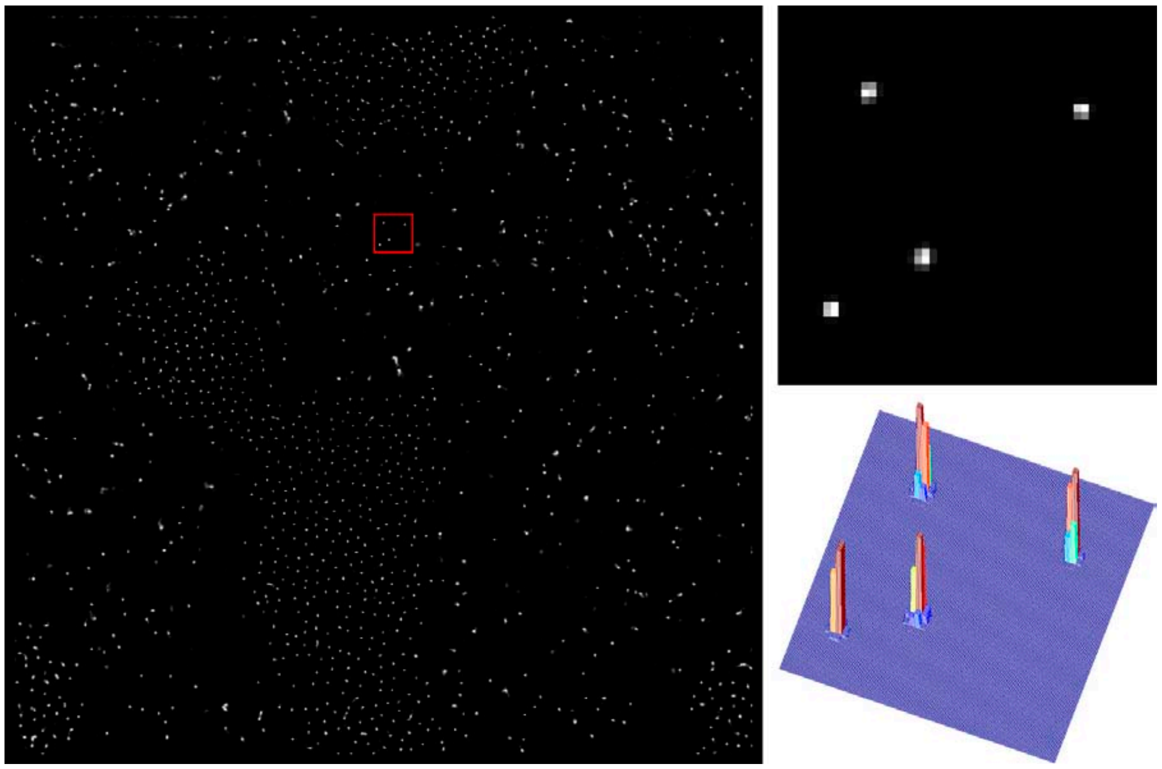

Figure 6.8: Result of exponentiating the difference image created by subracting the dilated image shown in figure 6.7 from the filtered image shown in figure 6.6. Pixels with zero intensity in the difference image transform into the brightest pixels in the exponentiated image, and occur at the positions of the local maxima in the filtered image.

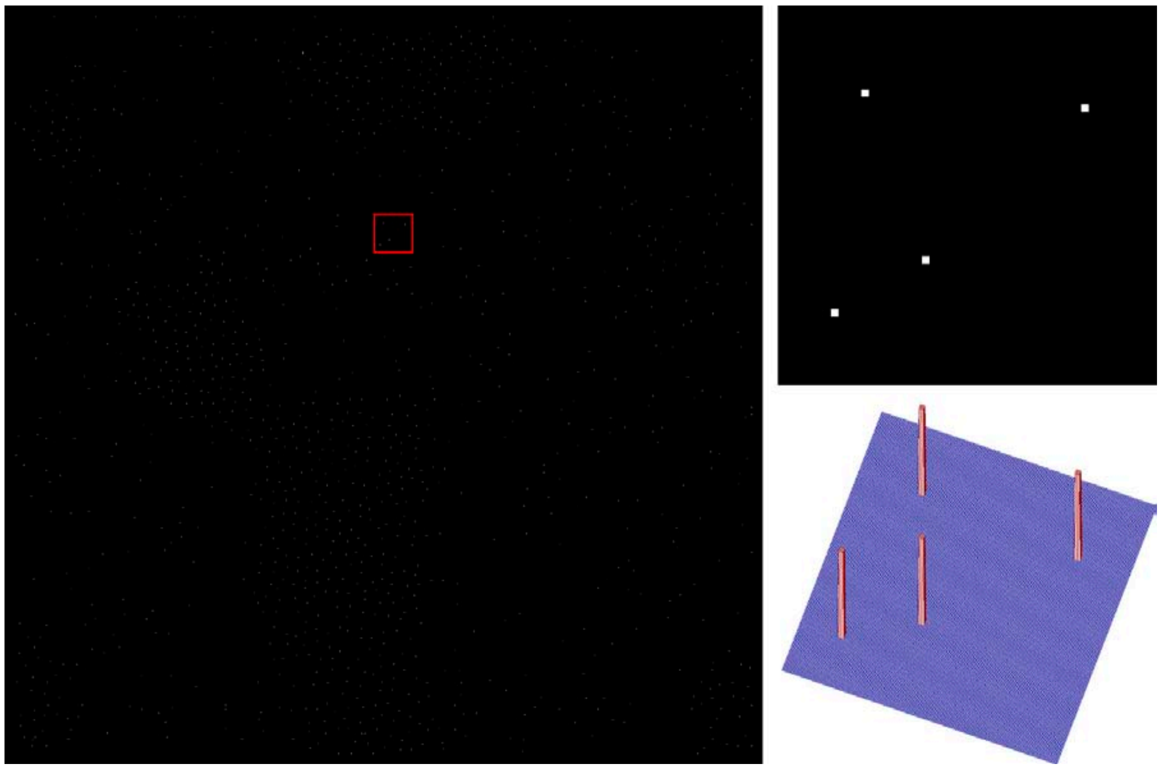
Figure 6.9: Result of thresholding the exponentiated image of figure 6.8. Non-white (grey) pixels are clipped, leaving only single white dotes indicated the center of the particles on an otherwise entirely black background.

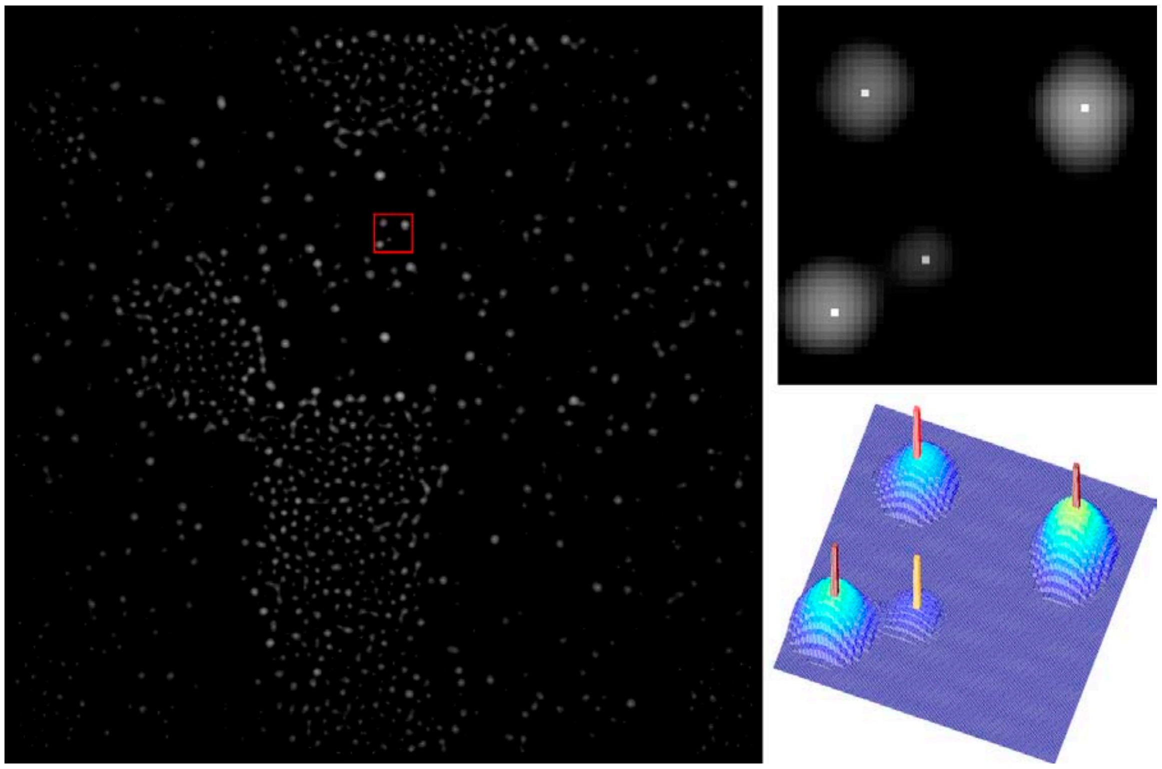

Figure 6.10: Center positions of local maxima after thresholding, as shown in figure 6.9 are overlaid on top of the filtered image shown in figure 6.5.

on top of the processed image of figure 6.5 in figure 6.10. For comparison, and to provide a quick way to assess how accurate the algorithm that finds local-maxima, the white dots corresponding to located particle centers (as shown in figure 6.9) can be overlaid on top of the original, unprocessed image (as shown in figure 6.1). The result is shown in figure 6.11.

6.2.5 Sub-pixel Position Refinement

The local maximum for each particle represents the single pixel (with an integer address within a 2D image) closest to the true brightness center. But greater precision is achievable by averaging the intensity values, determining the center of intensity (analogous to the center of mass) and radius of gyration.

A computationally efficient way to do this is again to use various techniques within IPP, implemented in the `ParticleStatistics()` function in `data_proc1.cpp`. This function begins by determining the discrete (x, y) coordinates of each local maximum (excluding particles near the boundary), and determining an overall count of the maxima:

```
Ipp32f (*ParticleStatistics(Image2D &image_localmax,
    Image2D &image_in, const int mask_radius,
    const int feature_radius, int &counter))[8]
{
    . . .
    counter = 0;
    int imagewidth = image_in.get_width();
    for(int j = 0; j < numberofpixels; j++) {
        if(localmaxdata[j] == 1) {
            xval = image_in.getx(j);
            yval = image_in.gety(j);

            if (xval > minx && xval < maxx
                && yval > miny && yval < maxy) {
                particledata[counter][0] = j;
```

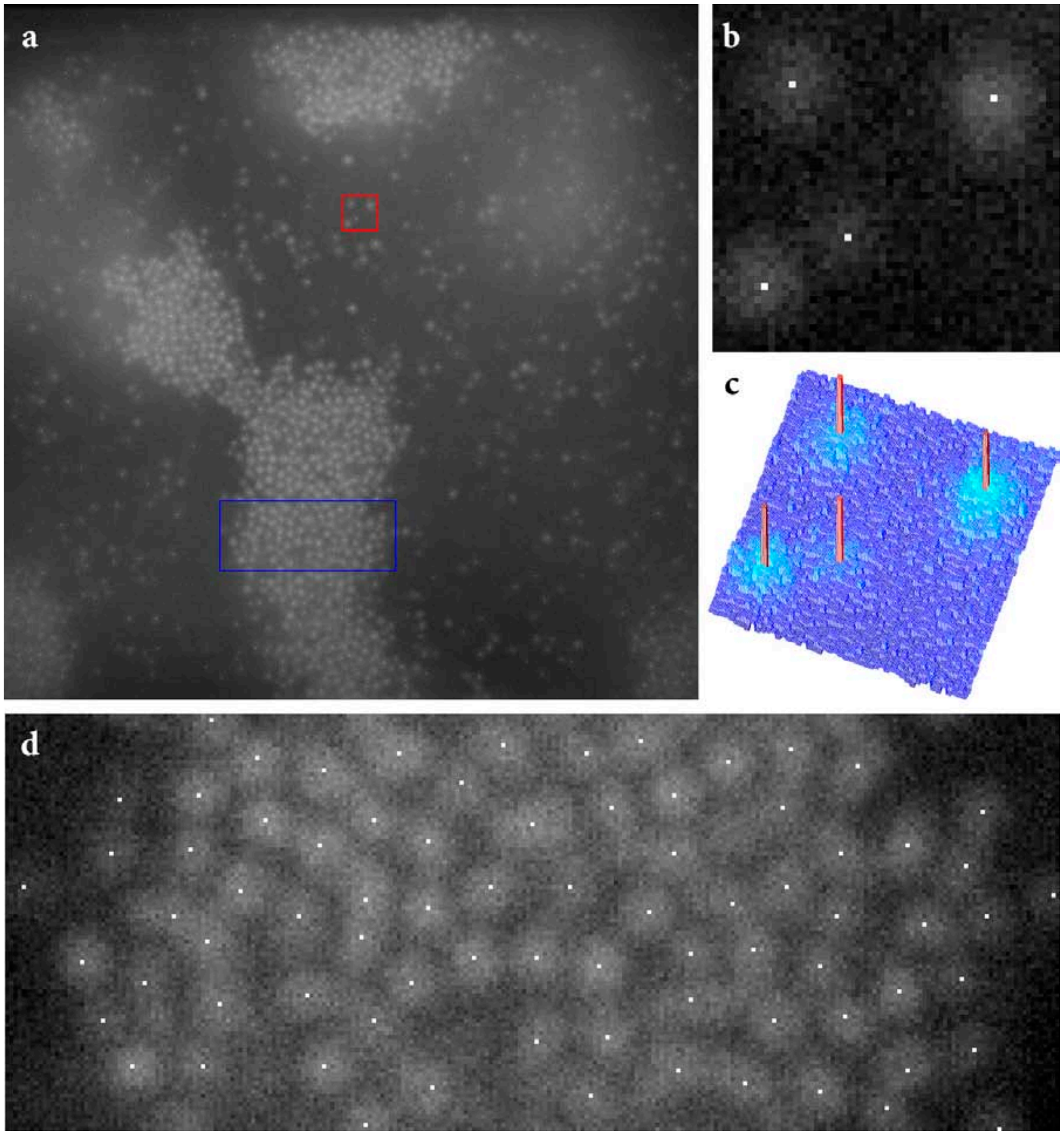

Figure 6.11: Center positions of local maxima after thresholding, as shown in figure 6.9 are overlaid on top of the original image shown in figure 6.1. **(a)** full image. **(b)** 10× zoom of the region marked with a red square in **(a)**, displayed in 3D in **(c)**, as with all other examples in this chapter. **(d)** 6× zoom of the region marked with a blue rectangle in **(a)**, demonstrating the accuracy of the particle location algorithm for large numbers of dense, disordered particles.

```
                particledata[counter][1] = xval;
                particledata[counter][2] = yval;
                counter++;
            }
        }
    }
```

Then, for each local maximum at $\vec{r}_{xy}$, the local pixels in the neighborhood are extracted and copied to a new, much smaller, image. It is on this smaller image that all of the calculations will be performed.

```
    for(int i = 0; i < counter; i++) {
        int extract_index = particledata[i][0];
        int copy_offset = extract_index - extract_offset;
        status = ippiCopy_32f_C1R(inputimage
            + copy_offset, image_in.get_stepsize(),
            extracted_square, extract_step, extract_ROI);
    . . .
```

Because the chosen extraction neighborhood is typically four orders of magnitude smaller in area (i.e. 10 pixels across for the extracted region, vs. 1000 pixels across for the original image), the memory requirements and number of operations required for the calculation drop dramatically, leading to a huge increase in performance over multiplying with the full image. All sums are therefore performed for pixels within a specified mask radius R_m, set by command-line parameter, around each ith local maximum at (x, y), $\vec{r}^{\,i}_{xy}$. For convenience, we define the following notation for these sums, and leave out the index i with the understanding that the quantities are determined for each particle.

$$\sum_{x,y\in R_\mathrm{m}} \equiv \sum_{|\vec{r}-\vec{r}_{xy}|\leq R_\mathrm{m}} \tag{6.8}$$

The total intensity (referred to as “mass” within the code) is defined as the sum of the pixel intensity values in the neighborhood of the ith particle center:

$$I_{\mathrm{T}} = \sum_{x,y \in R_{\mathrm{m}}} I(x,y) \tag{6.9}$$

where $I(x,y)$ is the intensity of the pixel with coordinates (x,y). The $\hat{x}$-coordinate of the center-of-intensity $\bar{x}$ is defined as:

$$\bar{x} = \frac{1}{I_{\mathrm{T}}} \sum_{x,y \in R_{\mathrm{m}}} xI(x,y) \tag{6.10}$$

Similarly, the $\hat{y}$-coordinate of the center-of-intensity $\bar{y}$ is defined as:

$$\bar{y} = \frac{1}{I_{\mathrm{T}}} \sum_{x,y \in R_{\mathrm{m}}} yI(x,y) \tag{6.11}$$

Finally, the radius of gyration R_{g} is defined as:

$$R_{\mathrm{g}}^2 = \frac{1}{I_{\mathrm{T}}} \sum_{x,y \in R_{\mathrm{m}}} \left[(x-\bar{x})^2 + (y-\bar{y})^2\right] I(x,y) \tag{6.12}$$

These quantities are calculated using kernels which are then multiplied over the entire extracted image (and so are not a convolution, per se), and then the results added to yield the desired numerical sums, as above. Four kernels are created in the constructor for the (not ideally named) `Convolution_Kernel` class in `kernel1.cpp`. As in the case for the dilation kernel, ramps and thresholds are used to create several kernels, which are then trimmed outside a certain radius.

```
Convolution_Kernel::Convolution_Kernel(const int radius,
    const int tif_width, const int tif_length):
    circle_kernel(NULL), r2_kernel(NULL),
    ramp_x_kernel(NULL), ramp_y_kernel(NULL)
{
    . . .
    //create ramps from -radius to +radius
    //for r^2 and circular masks
    status = ippiImageRamp_16s_C1R(ramp_x, ramp_step,
        ramp_roi_size, -radius, 1, ippAxsHorizontal);
    status = ippiImageRamp_16s_C1R(ramp_y, ramp_step,
        ramp_roi_size, -radius, 1, ippAxsVertical);

    //square the ramps and then finally add
    status = ippiSqr_16s_C1IRSfs(ramp_x, ramp_step,
        ramp_roi_size, 0);
    status = ippiSqr_16s_C1IRSfs(ramp_y, ramp_step,
        ramp_roi_size, 0);
    status = ippiAdd_16s_C1RSfs(ramp_x, ramp_step,
        ramp_y,ramp_step,r2,ramp_step,ramp_roi_size,0);

    status = ippiCopy_16s_C1R(r2, ramp_step, circle,
        ramp_step, ramp_roi_size);

    //trim away values greater than radiusˆ2 for r2 kernel
    status = ippiThreshold_GTVal_16s_C1IR(r2,
        ramp_step, ramp_roi_size, radius * radius, 0);

    //trim away values greater than radiusˆ2
    //for circle kernel
    status = ippiThreshold_GTVal_16s_C1IR(circle,
        ramp_step, ramp_roi_size,radius * radius, -1);
    status = ippiThreshold_GTVal_16s_C1IR(circle,
        ramp_step, ramp_roi_size,-1, 1);
    status = ippiThreshold_LTVal_16s_C1IR(circle,
        ramp_step, ramp_roi_size,1, 0);

    //create ramps from constant-radius to constant+radius
    //for rˆ2 and circular masks
    constantoffset = 1;
    status = ippiImageRamp_16s_C1R(ramp_x, ramp_step,
        ramp_roi_size, constantoffset, 1, ippAxsHorizontal);
    status = ippiImageRamp_16s_C1R(ramp_y, ramp_step,
        ramp_roi_size, constantoffset, 1, ippAxsVertical);

    //multiply ramps by circle kernel
    //to eliminate all values beyond the radius
    status = ippiMul_16s_C1IRSfs(circle, ramp_step,
        ramp_x, ramp_step, ramp_roi_size, 0);
    status = ippiMul_16s_C1IRSfs(circle, ramp_step,
        ramp_y, ramp_step, ramp_roi_size, 0);

    //Convert kernels to final forms
    status = ippiConvert_16s32f_C1R(circle, ramp_step,
        circle_kernel, convolution_kernel_step,
        ramp_roi_size);
    status = ippiConvert_16s32f_C1R(r2, ramp_step,
        r2_kernel, convolution_kernel_step, ramp_roi_size);
    status = ippiConvert_16s32f_C1R(ramp_x, ramp_step,
        ramp_x_kernel, convolution_kernel_step,
        ramp_roi_size);
    status = ippiConvert_16s32f_C1R(ramp_y, ramp_step,
        ramp_y_kernel, convolution_kernel_step,
        ramp_roi_size);
```

The final four kernels are then multiplied by the extracted image as part of a loop over all local maxima in `ParticleStatistics()` to calculate the quantities given in equations 6.9-6.12.

For I_T, a simple circle kernel is used:

```
Ipp64f total_mass = 0;
status = ippiMul_32f_C1R(extracted_square, extract_step,
    ConvolutionKernels.get_circle_kernel(),
    ConvolutionKernels.get_kernel_step(),
    multiply_result, extract_step, extract_ROI);
status = ippiSum_32f_C1R(multiply_result, extract_step,
    extract_ROI, &total_mass, ippAlgHintNone);
```

For $\bar{x}$ and $\bar{y}$, the $\hat{x}$-ramp and $\hat{y}$-ramp kernels are used:

```
Ipp64f x_sum = 0;
status = ippiMul_32f_C1R(extracted_square, extract_step,
    ConvolutionKernels.get_ramp_x_kernel(),
    ConvolutionKernels.get_kernel_step(),
    multiply_result, extract_step, extract_ROI);
status = ippiSum_32f_C1R(multiply_result, extract_step,
    extract_ROI, &x_sum, ippAlgHintNone);
Ipp32f x_offset = (x_sum/total_mass)-extract_radius-1;
```

```
    Ipp64f y_sum = 0;
    status = ippiMul_32f_C1R(extracted_square, extract_step,
        ConvolutionKernels.get_ramp_y_kernel(),
        ConvolutionKernels.get_kernel_step(),
        multiply_result, extract_step, extract_ROI);
    status = ippiSum_32f_C1R(multiply_result, extract_step,
        extract_ROI, &y_sum, ippAlgHintNone);
    Ipp32f y_offset = (y_sum/total_mass)-extract_radius-1;
```

For R_g, the r^2 kernel is used:

```
    Ipp64f r2_sum = 0;
    status = ippiMul_32f_C1R(extracted_square, extract_step,
        ConvolutionKernels.get_r2_kernel(),
        ConvolutionKernels.get_kernel_step(),
        multiply_result, extract_step, extract_ROI);
    status = ippiSum_32f_C1R(multiply_result, extract_step,
        extract_ROI, &r2_sum, ippAlgHintNone);
    Ipp32f r2_val = r2_sum / total_mass;
```

Finally, a last check sums up the number of local maxima in the extracted image, to determine if more than one local maximum occurs in the neighborhood, indicative of a possible error (in that too many particles are being found) due to poor choices of control parameters entered at the command line.

```
    Ipp64f multiplicity = 0;
    status = ippiSum_32f_C1R(image_localmax.get_image2D()
        + copy_offset, image_localmax.get_stepsize(),
        extract_ROI, &multiplicity, ippAlgHintNone);
Ipp32f multiplicity_val = multiplicity;
```

6.2.6 Final Text Output

The final step of the `plu_Centerfind.exe` program is to output a large text file, one row per particle, with the following data: stack number, slice number, $\bar{x}$, $\bar{y}$, I_T and R_g.

The first four and last four lines of the file analyzing the stack (number 101) of the sample (E13C) illustrated in the previous examples in this chapter are:

```
101 1   760.0   13.0    16.2    2.7
101 1   963.8   13.4    41.4    3.4
101 1   608.1   15.0    698.3   4.5
101 1   518.0   16.0    373.4   4.4
. . .
101 181 727.4   987.0   17.0    3.0
101 181 775.0   986.3   102.3   3.9
101 181 879.5   987.0   1.1     0.5
101 181 911.0   987.0   7.3     1.6
```

6.3 Three-Dimensional Center Reconstruction

After the (x, y) position of each local maximum is determined for each slice, the next step is to link these into a 3D (x, y, z) position for each particle by connecting the proper local maxima in adjacent images that correspond to different slices of the same particle. The program begins with housekeeping code that loads the data into an array, and then identifies which particle rows correspond to which slice of which stack.

6.3.1 Linking positions in adjacent slices

The first step is to pass through all particles in the first slice, and assign each one a unique particle number index.

```
    for(i=0; i<slice_rows[1][0]; i++) {
        data_list[i][9] = i;
    }
```

The heart of the program is a series of nested loops. For each kth particle in slice j (starting with the second slice), the program checks the lth particle in slice $j-1$, and determines if their (x, y) coordinates are sufficiently close. That is, it checks if,

$$(x_k - x_l)^2 + (y_k - y_l)^2 \leq R_\mathrm{m}^2 \tag{6.13}$$

where R_m is the maximum allowed radial distance in pixels, set by command-line parameter. If this condition is satisfied, the kth particle in slice j is assigned the particle index of the lth particle in slice $j-1$, and the corresponding slice count, the number of slices in which a given particle appears, is also increased by one.

```
for(int j=1; j<num_slices; j++) {
    for(k=slice_rows[j][0]; k<=slice_rows[j][1]; k++){
        cur_xpos = data_list[k][3];
        cur_ypos = data_list[k][4];
        for(l=slice_rows[j-1][0];
            l<=slice_rows[j-1][1]; l++) {
            xpos = data_list[l][3];
            ypos = data_list[l][4];
            r2 = (cur_xpos-xpos) * (cur_xpos-xpos)
                + (cur_ypos-ypos) * (cur_ypos-ypos);
            if(r2 < max_r2_dev) {
                //particle number index
                data_list[k][9] = data_list[l][9];

                //slice count
                data_list[k][5] = data_list[l][5]+1;

                //row number of lth particle
                data_list[k][6] = l;
            }
        }
    }
```

If two particles are sitting on top of one another, the algorithm will identify all of the slices as belonging to one particle (even though it is unphysically extended in the $\hat{z}$-direction). Although iterative strategies and refinements have been implemented elsewhere [100], I used a simpler algorithm that looks at the intensity of the located (x, y) center as a function of height. When passing through a particle along the optic $\hat{z}$-axis, the intensity should increase, reach a local maximum at the center, then decrease. If it increases again, then this is likely due to the presence of another particle adjacent in $\hat{z}$. Therefore, a change in the sign of the derivative of intensity with respect to distance along the $\hat{z}$-axis indicates a new particle. In that case, a new particle index is assigned to the new particle, and the slice count is reset. In addition, all particles must be located based on local maxima in more than a certain minimum number of slices, set at the command line, to eliminate orphan local maxima that most likely correspond to spurious peaks or noise.

```
for(int j=1; j<num_slices; j++) {
. . .
    for(k=slice_rows[j][0]; k<=slice_rows[j][1]; k++) {
        if((int) data_list[k][5] >= min_slices) {
            current_intensity = data_list[k][7];
            previous_index = data_list[k][6];
            previous_intensity =
                data_list[previous_index][7];
            if(current_intensity > previous_intensity) {
                data_list[k][9] = i;
                data_list[k][5] = 0;
                i++;
                }
            }
```

If there are still too many slices contributing to a given particle, after the slice count exceeds a hard-limit threshold set at the command line, the program automatically assigns a new particle index.

```
        if((int) data_list[k][5] >= max_slices) {
            data_list[k][9] = i;
            data_list[k][5] = 0;
            i++;
        }
```

Finally, if there is no lth particle in slice $j-1$ within R_m of the kth particle in slice j, then the latter is assigned a new particle index.

```
else if((int) data_list[k][9] == 0) {
    data_list[k][9] = i;
    i++;
    }
}
```

6.3.2 Refining the 3D position

The $\hat{x}$-, $\hat{y}$- and $\hat{z}$-coordinates are then determined by a weighted average of the positions in each of the slices contributing to the particle with particle index p. That is, for the pth particle,

$$\bar{x}^p = \frac{1}{I_\mathrm{T}^p}\sum_j x_j^p I_{\mathrm{T},j}^p \tag{6.14}$$

$$\bar{y}^p = \frac{1}{I_\mathrm{T}^p}\sum_j y_j^p I_{\mathrm{T},j}^p \tag{6.15}$$

$$\bar{z}^p = \frac{1}{I_\mathrm{T}^p}\sum_j z_j^p I_{\mathrm{T},j}^p \tag{6.16}$$

$$\tag{6.17}$$

where the sum over j is restricted to those slices assigned to the pth particle, $I_{\mathrm{T},j}^p$ is the summed intensity for the jth slice contributing to the pth particle, and I_T^p is the total intensity for all slices contributing to the pth particle:

$$I_\mathrm{T}^p = \sum_j I_{\mathrm{T},j}^p \tag{6.18}$$

These quantities are simply calculated with basic operations on the particle data loaded into an array, and in the last step normalized to convert from units of pixels to meaningful distances (i.e. μm) with conversion factors specified at the command line.

```
    for(current_row = 0; current_row < rows_in_stack;
        current_row++) {
        current_particle = data_list[current_row][9];
        weightfactor = data_list[current_row][7];
        data_zavg[current_particle][0] +=
            weightfactor * data_list[current_row][3];
        data_zavg[current_particle][1] +=
            weightfactor * data_list[current_row][4];
        data_zavg[current_particle][2] +=
            weightfactor * data_list[current_row][1];
        data_zavg[current_particle][3] +=
            weightfactor;
        data_zavg[current_particle][4] +=
            data_list[current_row][8];
        data_zavg[current_particle][5] =
            data_list[current_row][5]+1;
}

    for (current_particle = 0; current_particle <=
        max_part_ind; current_particle++) {
        data_zavg[current_particle][0] = xyfactor *
            data_zavg[current_particle][0] /
            data_zavg[current_particle][3];
        data_zavg[current_particle][1] = xyfactor*
            data_zavg[current_particle][1] /
            data_zavg[current_particle][3];
        data_zavg[current_particle][2] = zfactor *
            data_zavg[current_particle][2] /
            data_zavg[current_particle][3];
}
```

Note that, while in the present case, the positions are all weighted by I_{T}^{p}, they could also have been weighted by the R_{g} (defined in equation 6.12) values calculated for each particle in each slice, but the result is the same, and matches a fit to a gaussian, as shown in figure 6.12.

The final output of the program is a list of 3D coordinates for particles in each stack, with each particle occupying one line in the final text file. The final columns output are

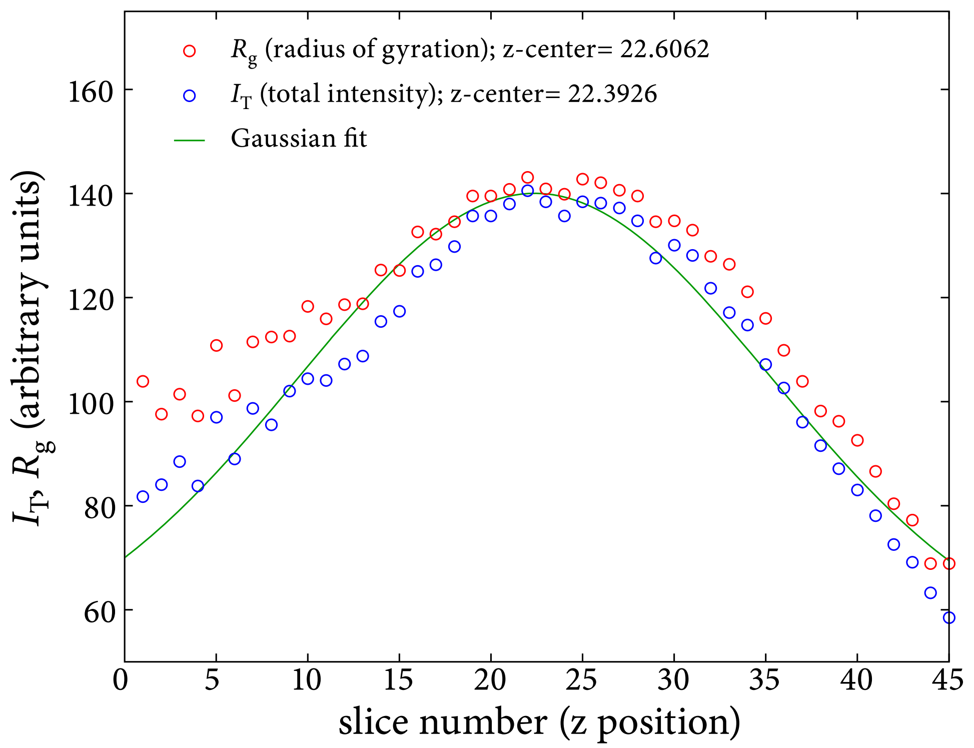

Figure 6.12: Comparison of determining the z-center $\bar{z}^p$ by weighting the image of one particle (with a=1.5μm) with the total intensity I_T^p and the radius of gyration R_g^p. In both cases, the estimate of $\bar{z}^p$ is the same, well within the uncertainty of the location process, and also coincides with the maximum of a best-fit gaussian.

stack number, $\bar{x}$, $\bar{y}$ and $\bar{z}$, with all positions given in μm. As an example, the first four lines of the `E13C_xyzt.txt` text file listing the 3D coordinates of particles (for the 101st stack of sample E13C, used in all examples illustrated previously) are:

```
101 43.7533 2.19656 1.56836
101 50.6803 33.6606 1.61641
101 48.4546 4.71838 1.9188
101 6.45937 5.41685 1.68192
. . .
```

This data can then be directly rendered in 3D. The particular example of E13C shows a gel coexisting with a dilute gas of small clusters, as can be seen in figure 6.13.

6.4 Acknowledgments

I thank John Crocker for explaining in great detail his original algorithms, and am particularly grateful to Hidekazu Oki, who on many occasions has found ways to remove bugs and (significantly) enhance performance and robustness of the C++ programs described in this chapter. I also thank Eli Sloutskin, Mike Massa, Tom Kodger and Kate Jensen for helpful comments when writing this chapter.

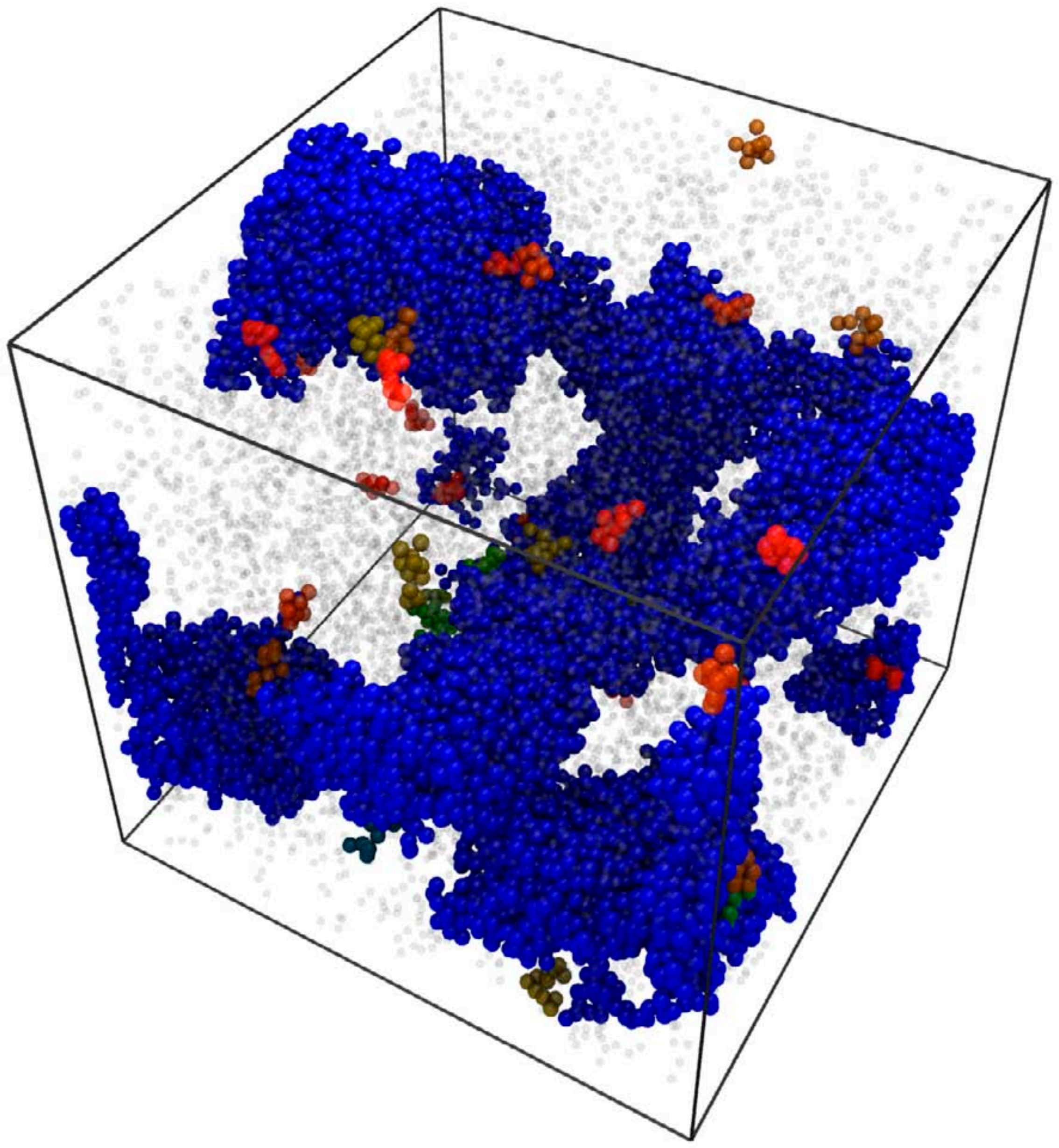

Figure 6.13: Three-dimensional reconstruction, rendered with Pixar's RenderMan, of the gel whose 2D confocal images were used to illustrate the stages of image processing earlier in this chapter. The central spanning gel is in coexistence with a dilute gas of small clusters.

Chapter 7

Gel and Cluster Structural Analysis

7.1 Introduction

In chapter 6, I described the techniques I used to locate particles in three dimensions starting with confocal microscope images, which are broadly applicable to a large class of scientific investigations, from colloids to embedding beads as markers in biological systems. In this chapter, I will describe the analysis of these 3D particle positions for attractive colloid systems, specifically describing those metrics I calculate to understand the gels and clusters, whose results are described in chapter 9 (published as [171]), and chapter 10 (published as [173]). As an example, I will also present the results of each of these metrics applied to the gel sample ($\phi = 0.08$ and $\xi = 0.059$, taken around 100,000 seconds after mixing) shown in figure 6.13.

All of the functions are contained in the program `plu_Struct3Dt.exe`, which loads the list of positions from the text file `filestem_xyzt.txt`. The $\hat{x}$- $\hat{y}$- and $\hat{z}$-coordinates of the position $\vec{r}_j \equiv (x_j, y_j, z_j)$ of the jth particle are passed to the various functions as a 3-column array called `positionlist[j][3]`. These functions calculate various quantities and generate text file outputs with the results.

7.2 Volume Fraction

Perhaps the simplest quantity to calculate based on particle positions is the overall volume fraction ϕ, defined as:

$$\phi \equiv \frac{4}{3}\pi a^3 \frac{N}{V} \tag{7.1}$$

where a is the colloid radius, and N is the number of colloidal spheres in the volume V. The particles in a stack are counted in the initial loading of the text data, while the system volume is determined by looking at the maximum range of particle positions, so that the volume fraction is unaffected by changes in parameters that, for instance, might reject more particles near the boundaries of the imaging volume. That is, the extent in the x-direction of the volume is $\Delta_x = x_{\max} - x_{\min}$, where $x_{\max}$ is the largest x position of any particle in the system, and $x_{\min}$ is the smallest. With analogous definitions for Δ_y and Δ_x, the volume of the system is $V \equiv \Delta_x \cdot \Delta_y \cdot \Delta_z$.

```
float systemvolume(float (*positionlist)[3],
    const int total_particles) {
    float minx = 1000, maxx = 0, miny = 1000,
        maxy = 0, minz = 1000, maxz = 0;
    for (int j=0; j < total_particles; j++) {
        if(positionlist[j][0] < minx) {
            minx = positionlist[j][0];
        }
        if(positionlist[j][1] < miny) {
            miny = positionlist[j][1];
        }
        if(positionlist[j][2] < minz) {
            minz = positionlist[j][2];
        }
        if(positionlist[j][0] > maxx) {
            maxx = positionlist[j][0];
        }
        if(positionlist[j][1] > maxy) {
            maxy = positionlist[j][1];
        }
        if(positionlist[j][2] > maxz) {
            maxz = positionlist[j][2];
        }
    }
    float deltax = maxx - minx;
    float deltay = maxy - miny;
    float deltaz = maxz - minz;
    return deltax * deltay * deltaz;
}
```

For the sample shown in figure 6.13, the dimensions of the imaging volume are $\Delta_x = 58.38$ μm, $\Delta_y = 59.09$ μm and $\Delta_z = 57.66$ μm. The overall volume fraction is $\phi = 8.4\%$.

7.3 Radial Distribution Function

The radial distribution function $g(r)$ is a widely-used metric to quantify how the average local density evolves in the neighborhood around each particle, as a function of distance r from the particle center [268, 269, 270]. It is normalized so that the average density at $g(r \rightarrow \infty) \equiv 1$, and calculated as an average for all particles in the distribution (i.e. all particles in a 3D stack in the present case). Explicitly, $g(r)$ is defined such that the quantity

$$\frac{N}{V} g(r) 4\pi r^2 dr \tag{7.2}$$

is the average number of particles in a spherical shell of width $r \rightarrow r + dr$ at a distance r from any particle in a fluid, assuming the particles interact with a spherically-symmetric potential $v(r)$ [219]. In this case, N is the total number of particles, V is the total system volume, and r is the Euclidean distance between the two points:

$$r_{ij} \equiv \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} \tag{7.3}$$

The function $g(r)$ can also be related to the total system energy, when all particle interactions are pair-wise [219]:

$$U(V,T,N) = \frac{3}{2}Nk_{\mathrm{B}}T + \frac{1}{2}\frac{N^2}{V}\int v(r)g(r)4\pi r^2 dr \tag{7.4}$$

I have implemented an algorithm originally created by David Grier, in which $g(r)$ is calculated by summing up distances between two particles, and normalizing by inverse volumes $\varDelta V_{ij}^{-1}$ to account for the finite size of the system:

$$\varDelta V_{ij}^{-1} \equiv \frac{1}{\left(\varDelta_x - |x_i - x_j|\right)\left(\varDelta_y - |y_i - y_j|\right)\left(\varDelta_z - |z_i - z_j|\right)} \tag{7.5}$$

So that $g(r)$ is calculated by summing all inverse volumes for all pairs of particles.

The function calculates $\varDelta V_{ij}^{-1}$ only for pairs of particles separated by $r_{ij} < r_{\mathrm{m}}$, where r_{m} is a maximum distance set at the command line, and discretizes the value of r_{ij} to add to the proper discretized bin for $g(r)$, whose width is also set by a command-line parameter. The discrete bin number becomes the index of the array used to hold the $g(r)$ data, which significantly increases performance in this inner loop by minimizing the number of necessary memory accesses:

```
for(i=0; i<totalnumparticles; i++) {
. . .
    for(j=i+1; j<totalnumparticles; j++) {
        dx = particle_list[i][0]-particle_list[j][0];
        dy = particle_list[i][1]-particle_list[j][1];
        dz = particle_list[i][2]-particle_list[j][2];
        r2 = dx*dx + dy*dy + dz*dz;
```

```
            if(r2 < max_radius_gr3_2) {
                r2_norm_gr3 = numbins2*r2/max_radius_gr3_2;
                inv_area = 1 / ( (deltax-fabs(dx))*
                    (deltay-fabs(dy))*(deltaz-fabs(dz)) );
                g_r2[r2_norm_gr3][0]+=inv_area;
                g_r2[r2_norm_gr3][1]+=inv_area * inv_area;
            }
        . . .
        }
    }
```

The $g(r)$ calculated for the gel sample of figure 6.13 is shown in figure 7.1.

Density distribution functions do not depend on the connectivity of the particles involved, only their positions, as they are averaged over all particles equally. As such, particularly at short length scales where $r \sim a$, uncertainties in particle location can smear out the peak in $g(r)$, and so its height as measured from confocal data may not be a very robust quantity. Scattering, in some cases, may be a better route to quantifying $S(q)$ and $g(r)$.

7.4 Particle Bonding and Nearest-Neighbor Distributions

Perhaps one of the greatest strengths of using confocal microscopy is the ability to uniquely identify individual particles, and to see which specific individual particles are located nearby. In the ideal case, where particles could be located exactly and did not move, then two particles could be considered bonded if they are touching; that is, if their centers were separated by a bond distance r_b equal to the particle diameter $d = 2a$. I define the separation between the ith particle at $\vec{r}_i$ and the jth particle at $\vec{r}_j$ as:

$$\Delta r_{ij} \equiv \left|\vec{r}_i - \vec{r}_j\right| \tag{7.6}$$

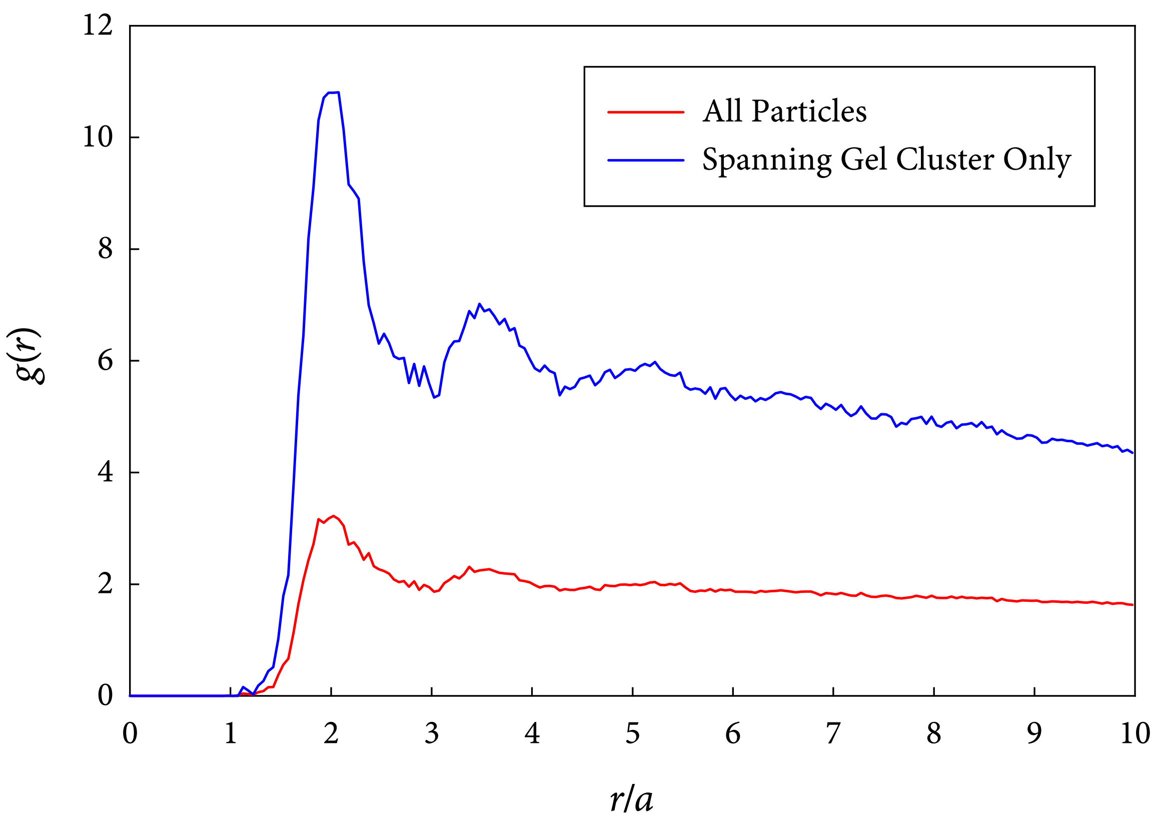

Figure 7.1: Radial distribution function $g(r)$ for all particles (red) and for the spanning cluster (blue) in the gel sample illustrated in figure 6.13. The value of $g(r)$ for the spanning cluster approaches 1 for much larger length scales than is plotted here, reflecting its inhomogeneity on length scales of tens of particles, as can be seen in figure 6.13.

In the ideal case, the ith and jth particles are bonded only if:

$$\Delta r_{ij} \leq r_\mathrm{b} \tag{7.7}$$

In real-world experimental systems, such as those used in the preparation of this thesis, the ideal conditions are absent: there is always uncertainty in the determination of particle center positions, due to imaging noise and the fact that all particles diffuse on account of thermal motion. A free micron-sized particle will diffuse its own size in a few seconds, greater than—but still on the order of—the time it takes for the confocal microscope to acquire several image slices of it. Of course, more densely-packed and bonded particles will diffuse far less, but thermal motion can nevertheless blur the image. Therefore, two particles that may be separated by slightly more than d should in many cases still be considered bonded; that is, $r_\mathrm{b} > d$.

There are several ways, then, to select r_b. One is to include all r up to the first minimum after the first peak in $g(r)$, a generous definition that is nonetheless appropriate for dense gels (perhaps coexisting with a dilute monomeric gas where the typical interparticle separation is many d), where only large, obvious differences in gel structures need to be rudimentarily characterized, as in chapter 9. But this generous a definition for r_b then includes many particles which are located nearby, but not touching. A more conservative definition, is to select r_b as $d(1+\epsilon)$, where $d\epsilon$ is approximately equal to the particle position location uncertainty. This definition was chosen to determine bonding in chapter 10.

Regardless, once r_b has been set with a command-line parameter, determining particle bonds is a straightforward subtraction of particle positions, held in the first three elements of `particle_list`. Similarly, determining the number of nearest neighbors for each particle is a simple matter of counting how many particles are separated by less than r_b.

When two particles are considered bonded, the nearest neighbor counts are incremented by one for both.

```
for(i=0; i<totalnumparticles-1; i++) {
    for(j=i+1; j<totalnumparticles; j++) {
        r2 = (particle_list[i][0]-particle_list[j][0]) *
            (particle_list[i][0]-particle_list[j][0]) +
            (particle_list[i][1]-particle_list[j][1]) *
            (particle_list[i][1]-particle_list[j][1]) +
            (particle_list[i][2]-particle_list[j][2]) *
            (particle_list[i][2]-particle_list[j][2]);

        if(r2 < bond_r2) {
            //increase nearest neighbor count
            particle_list[i][4]++;
            particle_list[j][4]++;
        . . .
        }
    }
}
```

Nearest-neighbor distributions are shown for several different clusters in figure 9.7. For the example image stack for the gel sample of figure 6.13, the nearest-neighbor distribution is shown in figure 7.2.

7.5 Cluster Analysis

Slightly less trivial, however, is keeping track of which particles are part of the same clusters, which forms the basis of the analysis in chapter 10. In general, two bonded particles are part of the same cluster, but particles cannot be bonded to particles of other clusters. Alternatively stated, all particles within a cluster share at least one bond with at least one other particle within the same cluster, but share no bonds with particles in other clusters. Keeping track of which clusters each particle belongs to requires several steps.

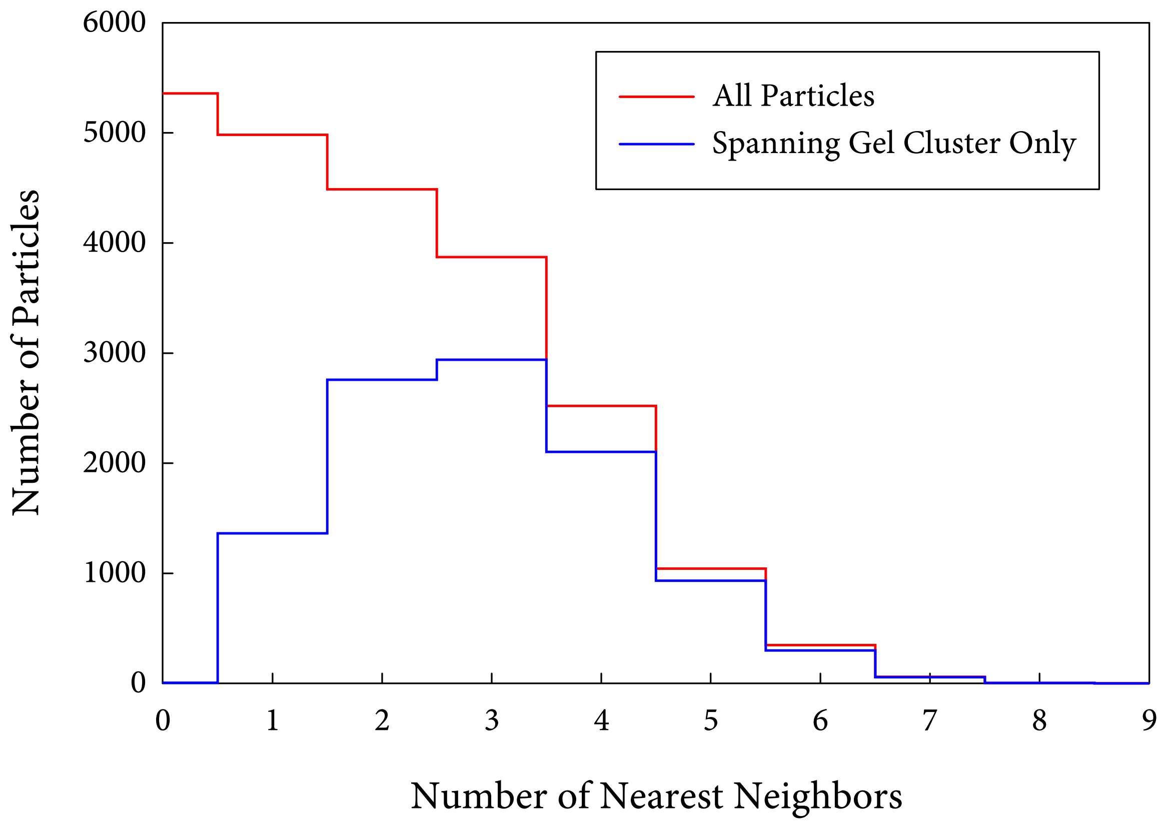

Figure 7.2: Nearest-neighbor distribution for all particles (red) and for the spanning cluster (blue) in the gel sample illustrated in figure 6.13.

7.5.1 Assigning Particles to Clusters

In the first step, the jth particle is assigned a cluster index of j, so that all particles before checking bonds have a unique cluster index. Then, once it is established that the ith and jth particles are bonded (and the loop forces $j > i$ always), if this is the first bond to the jth particle, then it is simply assigned the cluster index of the ith particle. However, if the jth particle is already bonded, it will already have a cluster index, and more importantly, other particles with that index need to be updated to have their cluster indices updated, reflecting the fact that they are now part of the cluster containing particle i. This is done by selecting the lower cluster index from particle i or particle j, and assigning this lower cluster index to all of the particles that are bonded to either particle i or particle j. In this way, the cluster indices are constantly updated, and proper connectivity is maintained.

```
for(i=0; i<totalnumparticles-1; i++) {
for(j=i+1; j<totalnumparticles; j++) {
    . . .
    if(r2 < bond_r2) {
        . . .
        if(cluster_index_list[j] == j) {
            cluster_index_list[j] =
                cluster_index_list[i];
        }
        else {
            oldsmallerindex =
                cluster_index_list[j];
            oldsmallerindex2 =
                cluster_index_list[i];
            newsmallerindex = smallest(
                oldsmallerindex, oldsmallerindex2,
                cluster_index_list[oldsmallerindex]);
            for(k=0; k<totalnumparticles; k++) {
                if(cluster_index_list[k]==
                    oldsmallerindex ||
                    cluster_index_list[k]==
                       oldsmallerindex2) {
                    cluster_index_list[k]=
                        newsmallerindex;
                }
            }
        }
    }
}
}
```

Finally, because the assignment of cluster number leaves many numbers unused, the list of clusters is renumbered sequentially. The new cluster labels are then added back to the lists of clusters and of particles. The final count of clusters is then simply the highest cluster index reached, stored in the variable `count`.

```
    int count = 1, raw_cluster_number = 0,
        consolidated_cluster_number = 0;
    for(i = 0; i<=totalnumparticles; i++) {
        if( (int) cluster_list[i][0] !=0) {
            cluster_list[i][7]=count++;
        }
    }
    for(i = 0; i<totalnumparticles; i++) {
        raw_cluster_number = particle_list[i][3];
        consolidated_cluster_number =
            cluster_list[raw_cluster_number][7];
        particle_list[i][5] = consolidated_cluster_number;
    }
    logfile << "Total number of clusters: "
        << --count << endl;
```

7.5.2 Self-Consistency Checks

The code then runs several self-consistency checks to detect obvious errors in the cluster determination and label assignment and propagation. The first is to identify any particles identified as monomers (i.e. are in a cluster of size one) that are nevertheless bonded to other particles (i.e. number of nearest neighbors not zero); their presence clearly must be erroneous:

```
for(i=0;i<totalnumparticles;i++) {
    cluster_label = (int) particle_list[i][3];
    if( (int) particle_list[i][4] != 0 &&
        (int) cluster_list[cluster_label][0]==1 ) {
        if (errorcount1 == 0) {
            logfile << "The following particles have >0
                nearest neighbors, but
                are clusters of size 1" << endl;
        }
        logfile << cluster_label << "\t";
        errorcount1++;
    }
}
```

The second test is to check for particles with no bonds (no nearest-neighbors) that register as having a cluster size of > 1, also obviously erroneous:

```
for(i=0;i<totalnumparticles;i++) {
    cluster_label = (int) particle_list[i][3];
        if( (int) particle_list[i][4] == 0 &&
            (int) cluster_list[cluster_label][0]!=1) {
        if (errorcount2 == 0) {
            logfile << "\nThe following particles have 0
                nearest neighbors,
                but are clusters of size >1" << endl;
        }
        logfile << cluster_label << "\t";
        errorcount2++;
    }
}
logfile << endl << "Total monomer mismatches (should
    be zero): " << errorcount1 + errorcount2 << endl;
```

7.5.3 Cluster Structural Analysis

Once the cluster assignments have been made and validated, a number of quantities are determined by analyzing the structure of all particles included in a given cluster.

The cluster mass N (i.e. number of particles) in each cluster is trivially calculated by a simple loop through the particle positions:

```
for(i=0;i<totalnumparticles;i++) {
    cluster_label = (int) particle_list[i][3];
    cluster_list[cluster_label][0]++;
    . . .
}
```

These counts can then be re-binned logarithmically, by first creating an array whose index corresponds to the logarithmically-spaced cluster sizes, then incrementing that element by direct addressing to count clusters of different size ranges.

```
for(i=10; i<NUMLOGBINS; i++) {
    tenpower = (float) i/10;
    logbins[i][0] = (int) 100 * pow(10,tenpower);
    logbins[i][1] = 0;
}
const int biggest_cluster_size = 10000000;
int *num_clusters_x100 = new int[biggest_cluster_size];
for(i=0;i<largest_cluster;i++) {
    num_clusters_x100[100*i]=cluster_size_list[i];
}
for(i=0; i<NUMLOGBINS-1; i++) {
    for(j=logbins[i][0]; j<logbins[i+1][0]; j++) {
    logbins[i][1]+=num_clusters_x100[j];
    }
}
```

Example logarithmically-binned cluster-mass distributions are shown in figures 10.5, 10.7 and 10.8.

Clusters are defined to be *spanning* (or percolated) if particles touch *both* of a pair of opposing faces of the imaging volume. Particles are considered to touch a boundary if their centers are within `perc_distance` of that boundary, and these boundaries are set to be the x_{max}, x_{min}, etc., previously determined.

```
for(i=0;i<totalnumparticles;i++) {
. . .
    if( (particle_list[i][0] - minx) < perc_distance) {
        cluster_list[cluster_label][1] = 1;
    }
    if( (maxx - particle_list[i][0]) < perc_distance) {
        cluster_list[cluster_label][2] = 1;
    }
    if( (particle_list[i][1] - miny) < perc_distance) {
        cluster_list[cluster_label][3] = 1;
    }
    if( (maxy - particle_list[i][1]) < perc_distance) {
        cluster_list[cluster_label][4] = 1;
    }
    if( (particle_list[i][2] - minz) < perc_distance) {
        cluster_list[cluster_label][5] = 1;
    }
    if( (maxz - particle_list[i][2]) < perc_distance) {
        cluster_list[cluster_label][6] = 1;
    }
}
```

The gels shown in figures 6.13, 9.4 and 10.4 are examples of large spanning clusters.

The position of a cluster's center of mass $\vec{R}_{\mathrm{cm}}$ is obtained by averaging the positions its constituent N particles:

$$\vec{R}_{\mathrm{cm}} \equiv (x_{\mathrm{cm}}, y_{\mathrm{cm}}, z_{\mathrm{cm}}) = \frac{1}{N}\left(\sum_{j=1}^{N} x_j, \sum_{j=1}^{N} y_j, \sum_{j=1}^{N} z_j\right) \tag{7.8}$$

This is implemented by a pair of simple loops, which for performance reasons calculate all clusters at the same time, so that the loop through the long list of particles is only done twice—not once for each cluster.

```
for(i=0;i<totalnumparticles;i++) {
    cluster_label = (int) particle_list[i][3];
    //x particle position
    cluster_list[cluster_label][8]
        += particle_list[i][0];
    //y particle position
    cluster_list[cluster_label][9]
        += particle_list[i][1];
    //z particle position
    cluster_list[cluster_label][10]
        += particle_list[i][2];
}
for(i = 0; i<=totalnumparticles; i++) {
    if( (int) cluster_list[i][0] !=0) {
        //x-center
        cluster_list[i][8] /= cluster_list[i][0];
        //y-center
        cluster_list[i][9] /= cluster_list[i][0];
        //z-center
        cluster_list[i][10] /= cluster_list[i][0];
    }
}
```

The radius of gyration R_g characterizes the size of the cluster, and is defined as:

$$R_\mathrm{g}^2 = \frac{1}{N}\sum_{j=1}^{N}\left[\left(x_j - x_\mathrm{cm}\right)^2 + \left(y_j - y_\mathrm{cm}\right)^2 + \left(z_j - z_\mathrm{cm}\right)^2\right] \tag{7.9}$$

Again, this is calculated for all clusters in a single pass through the list of particle positions for performance reasons.

```
for(i=0;i<totalnumparticles;i++) {
    cluster_label = (int) particle_list[i][3];
    delta_x = particle_list[i][0] -
        cluster_list[cluster_label][8];
    delta_y = particle_list[i][1] -
        cluster_list[cluster_label][9];
    delta_z = particle_list[i][2] -
        cluster_list[cluster_label][10];
    R2 = delta_x * delta_x + delta_y *
        delta_y + delta_z * delta_z;
```

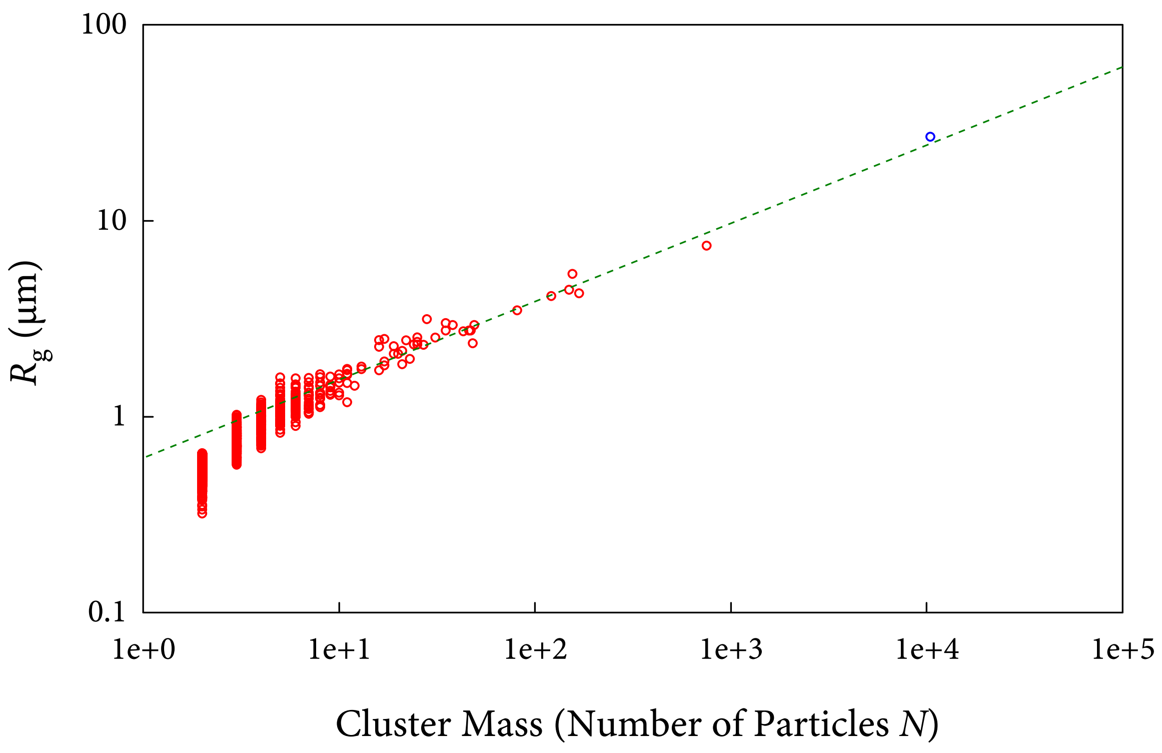

Figure 7.3: Scatter plot of cluster radius of gyration R_g and number of particles in the cluster N, for all clusters in the gel sample illustrated in figure 6.13. The large spanning cluster is indicated with a blue circle. A best-fit green, dashed line has a slope of 2.5, providing some estimate for the fractal dimension.

```
        particle_list[i][6] = delta_x;
        particle_list[i][7] = delta_y;
        particle_list[i][8] = delta_z;
        particle_list[i][9] = R2;
        cluster_list[cluster_label][11] += R2;
    }
    for(i = 0; i<=totalnumparticles; i++) {
        if( (int) cluster_list[i][0] !=0) {
            cluster_list[i][11] /= cluster_list[i][0];
        }
    }
```

Looking at N as a function of R_g for clusters is one way to estimate a fractal dimension, as shown in figure 9.6, and in figure 7.3 for the gel sample of figure 6.13.

7.6 Gel Internal Volume Fraction

While calculating the average volume fraction ϕ is simply a matter of counting particles, determining the internal volume fraction ϕ_g for spanning gel clusters is not straightforward. Several initial strategies involved looking at the Voronoi volumes of internal densely-packed particles in gels and clusters in chapter 9 (see also [282]), but a major problem arose with choosing particles for the more tenuous structures in chapter 10. By selecting particles with only a large number of nearest neighbors, e.g. more than nine or ten, clearly selects dense particles, but the overall average changes significantly with the choice of minimum nearest-neighbor cutoff.

Instead, we estimate ϕ_g by measuring the free volume accessible to a spherical test particle of radius a. Splitting the imaging volume into a fine grid of cubes with edge length $l_\mathrm{c} \ll a$, we place a test particle in each cube, and if no part of it intersects with cluster particles, the volume occupied by the test particle is considered within the free volume. The fraction of sample volume not part of the free volume is considered to be the total cluster volume. The total volume of the particles within the cluster is their number times the volume per particle; dividing this by the total cluster volume yields ϕ_g. We use this method to determine ϕ_g for only spanning clusters in the present work, but can also apply it without modification to characterize any other clusters or groups of particles.

Although simple to describe in principle, actually performing this calculation is not trivial. The simplest brute-force approach would be just to create one giant grid and do the calculation with a series of simple loops. Unfortunately, this is computationally impractical; to understand this, consider that the typical length of the imaging volume is 60 μm, about 100 times the particle radius $a = 0.56$ μm used in much of the work described in this thesis. If the cube grid has $l_{\mathrm{c}} = 0.1a$, then the total number of volume elements needed for the calculation is roughly a billion. So several gigabytes of memory would be needed just to load the grid into memory.

A more efficient and elegant strategy takes into account the fact that most of the empty free space around the clusters, as well as the volume within the clusters themselves, is contiguous, and therefore does not need to be sampled with great precision. So a two-scale approach was pursued. First, the imaging volume is broken up into large cubes, setting l_{c} to around a twentieth of the edge length of the imaging volume. That yields a 3D grid with roughly ten thousand elements. Then, cubes occupied by particles or portions thereof are identified. Then, *only* those occupied cubes are broken into a another cubic grid of even smaller cubes with edge length $l_{\mathrm{c}}/20$. Finally, subcubes of this subgrid that are not part of the free volume are identified. Summing up the total volume of these subgrid cubes gives the total cluster volume.

This accelerates the calculation by orders of magnitude, but is still somewhat subtle, and thus the latest version of the code, contained in `plu_gel_block_071122.cpp`, is documented in detail.

7.6.1 Discretization onto the cubic grid

As with all previous cases, the particle data enters as the Cartesian coordinates of the centers of the spherical colloids, here in microns (and with several decimal places of accuracy). The first step for the function `make_RIB_boxes()` (so named because it also handles creating the RenderMan geometry used for the renderings) is to discretize the particle centers to the nearest grid cube, and in doing so determines which of these cubes contain the particle centers. The edge length of the cubes l_c is passed as a parameter to the function, from which several discrete quantities are derived: `num_elements_on_grid_edge`, the number of cubes along each edge; `total_volume_elements`, the total number of cubes in the entire volume; and `discrete_radius`, the particle radius, set to the bond distance r_b, to allow for some particle location uncertainty, in units of l_c. At this stage, the more conservative approach is to use a slightly larger effective radius for the particles (e. g. r_b), since this will simply flag more of the gridcubes, with the detailed analysis being performed on the smaller subgrid, with greater precision, where the actual particle radius a is used.

```
const int num_elements_on_grid_edge = (int)
    ceil( 2 * CENTER_POS / cubegrid_edge_unitlength);
const int total_volume_elements =
    (num_elements_on_grid_edge+1) *
    (num_elements_on_grid_edge+1) *
    (num_elements_on_grid_edge+1);
const int discrete_radius = (int) 1 +
    ceil(bond_distance / cubegrid_edge_unitlength);
```

Then, a loop over all particles establishes which grid cube each particle center falls into:

```
for(int l=0; l<total_particles; l++) {
    if((int) particle_list[l][7] > mass_threshold) {
        float particle_x_pos = particle_list[l][1];
        float particle_y_pos = particle_list[l][2];
        float particle_z_pos = particle_list[l][3];
        . . .

        int x_gridpos = (int) floor(particle_x_pos /
            cubegrid_edge_unitlength);
        int y_gridpos = (int) floor(particle_y_pos /
            cubegrid_edge_unitlength);
        int z_gridpos = (int) floor(particle_z_pos /
            cubegrid_edge_unitlength);
        . . .
```

Next, for each particle, all grid cubes that share faces with the one that contains that particle's center are checked to see if they also contain portions of other particles (as determined by the calculation during consideration of a different particle). An inner loop checks nearby cubes, whose coordinates are located within `discrete_radius`. If any of those cubes already contain the center of another particle, no other processing is necessary. Otherwise, if that box is empty, then further checking must be done, performed within the final `if` statement below:

```
for(i = x_gridpos - discrete_radius;
    i<= x_gridpos + discrete_radius; i++) {
    if(i >=0 && i < num_elements_on_grid_edge) {
        for(j = y_gridpos - discrete_radius;
            j<= y_gridpos + discrete_radius; j++) {
            if(j>=0 && j<num_elements_on_grid_edge) {
                for(k = z_gridpos - discrete_radius;
                    k <= z_gridpos + discrete_radius;
                    k++) {
                    if(k >=0 &&
                        k<num_elements_on_grid_edge) {
                    int index_1D = Index_3Dcube_to_1D(
                        i, j, k,
                        num_elements_on_grid_edge);
                    if(discrete_positions[index_1D] ==
                        false) {
                        status_boxfilled = false;
```

Within the conditional, the box is then assumed to be empty, and is subsequently tested to see whether any part of the particle falls within the box. This multi-step test begins with the determination of the physical Cartesian coordinates (i.e. in μm) of the box under consideration:

```
box_x_min = i * cubegrid_edge_unitlength;
box_y_min = j * cubegrid_edge_unitlength;
box_z_min = k * cubegrid_edge_unitlength;
box_x_max = (i + 1) * cubegrid_edge_unitlength;
```

```
    box_y_max = (j + 1) * cubegrid_edge_unitlength;
    box_z_max = (k + 1) * cubegrid_edge_unitlength;
```

Then, the six faces of the cube are checked to see if they intersect any part of the particle, by simply subtracting the individual coordinates from those of the faces and seeing if they are closer than the bond radius. For example, the following code fragment checks the $\hat{x}$-coordinate of the particle center, and that of the cube's two opposing faces perpendicular to the $\hat{x}$-axis (with analogous code for the $\hat{y}$ and $\hat{z}$ axes not shown).

```
    if( (abs(particle_x_pos-box_x_min) < bond_distance ||
        abs(particle_x_pos-box_x_max) < bond_distance) &&
        (particle_y_pos > box_y_min) &&
        (particle_y_pos < box_y_max) &&
        (particle_z_pos > box_z_min) &&
        (particle_z_pos < box_z_max) ) {
        status_boxfilled = true;
        }
        . . .
```

Next, each edge is checked to see if it penetrates inside a particle. For example, in the $\hat{x} - \hat{y}$ plane, edges parallel to the $\hat{z}$-axis are checked, by determining whether their coordinates (x', y') are within a bond radius of the particle's projected center position at (x, y). That is, whether

$$(x - x')^2 + (y - y')^2 < r_\mathrm{b}^2 \tag{7.10}$$

The $\hat{z}$-axis code is included below (analogous code to check edges parallel to the $\hat{y}$- and $\hat{x}$-axes not shown):

```
if( (particle_z_pos>box_z_min)
    && (particle_z_pos<box_z_max)
    && ( (delta_r_2D(particle_x_pos, box_x_min,
    particle_y_pos, box_y_min) < bond_distance)
    || (delta_r_2D(particle_x_pos, box_x_max,
    particle_y_pos, box_y_min) < bond_distance)
    || (delta_r_2D(particle_x_pos, box_x_min,
    particle_y_pos, box_y_max) < bond_distance)
    || (delta_r_2D(particle_x_pos, box_x_max,
    particle_y_pos, box_y_max) < bond_distance) )
    ) {
    status_boxfilled = true;
}
```

where the function `delta_r_2D()` finds the magnitude of the distance between two 2D points whose coordinates form the function's arguments:

```
float delta_r_2D(const float x1, const float x2,
    const float y1, const float y2) {
    float delta_r = 0;
    delta_r = sqrt( ((x1-x2) * (x1-x2)) +
        ((y1-y2) * (y1-y2)) );
    return delta_r;
}
```

The final check is to see whether any of the cube's corners, at (x'', y'', z''), fall within the particle with center (x, y, z). That is, whether

$$(x - x'')^2 + (y - y'')^2 + (z - z'')^2 < r_b^2 \tag{7.11}$$

```
if( (delta_r_3D(particle_x_pos, box_x_min, particle_y_pos,
    box_y_min, particle_z_pos, box_z_min) < bond_distance)
    ||(delta_r_3D(particle_x_pos,box_x_max,particle_y_pos,
    box_y_min, particle_z_pos, box_z_min) < bond_distance)
    ||(delta_r_3D(particle_x_pos,box_x_min,particle_y_pos,
    box_y_max, particle_z_pos, box_z_min) < bond_distance)
    ||(delta_r_3D(particle_x_pos,box_x_max,particle_y_pos,
    box_y_max, particle_z_pos, box_z_min) < bond_distance)
    ||(delta_r_3D(particle_x_pos,box_x_min,particle_y_pos,
    box_y_min, particle_z_pos, box_z_max) < bond_distance)
    ||(delta_r_3D(particle_x_pos,box_x_max,particle_y_pos,
    box_y_min, particle_z_pos, box_z_max) < bond_distance)
    ||(delta_r_3D(particle_x_pos,box_x_min,particle_y_pos,
    box_y_max, particle_z_pos, box_z_max) < bond_distance)
    ||(delta_r_3D(particle_x_pos,box_x_max,particle_y_pos,
    box_y_max, particle_z_pos, box_z_max) < bond_distance)
    ){
    status_boxfilled = true;
}
```

where the function `delta_r_3D()` finds the magnitude of the distance between two 3D points whose coordinates form the function's arguments:

```
float delta_r_3D(const float x1, const float x2,
    const float y1, const float y2,
    const float z1, const float z2) {
    float delta_r = 0;
    delta_r = sqrt( ((x1-x2) * (x1-x2)) +
        ((y1-y2) * (y1-y2)) + ((z1-z2) * (z1-z2)) );
    return delta_r;
}
```

If one or more of the above conditions is satisfied, then that box contains part of the particle, and its status is updated.

```
    if(status_boxfilled == true) {
        discrete_positions[index_1D]=status_boxfilled;
    }
```

The rough volume fraction after passing through this larger cubic grid is then determined, with a simple checksum to make sure all of the counting is at least self-consistent:

```
    for(i=0; i<total_volume_elements; i++) {
        if(discrete_positions[i] == true) {
            boxes_occupied_by_perc_cluster++;
        }
        else {
            boxes_empty++;
        }
    }
    if(boxes_occupied_by_perc_cluster + boxes_empty !=
        total_volume_elements) {
        logfile << "Box counting error!" << endl;
    }
```

7.6.2 Subgrid Refinement

In the second major step, all cubes that contain at least a portion of a particle are then broken down into a grid of subcubes with edge length $l'_c \equiv l_c/20$, and all of these subcubes are checked to determine whether they fall within the free volume of a test particle. That is, opposite to the method used for the larger grid cubes, it is the *empty* subgrid cubes which are counted. This calculation is performed by the `cubegrid_occupied_volume()` function, called for every larger grid cube that contains at least a portion of a particle.

```
for(i = 0; i< num_elements_on_grid_edge; i++) {
for(j = 0; j< num_elements_on_grid_edge; j++) {
for(k = 0; k< num_elements_on_grid_edge; k++) {
    int index_1D = Index_3Dcube_to_1D(i, j, k,
        num_elements_on_grid_edge);
    if(discrete_positions[index_1D] == true) {
        float x_pos = i * cubegrid_edge_unitlength;
        float y_pos = j * cubegrid_edge_unitlength;
        float z_pos = k * cubegrid_edge_unitlength;
        internal_cluster_volume +=
            cubegrid_occupied_volume(perc_cluster_particles,
            num_perc_particles, x_pos, y_pos, z_pos,
            cubegrid_edge_unitlength, particle_radius,
            bond_distance, out_file, do_render);
    }
}
}
}
```

The larger gridcube is first divided into smaller cubes, with all of the various parameters recalculated in units of this smaller grid. An expanded subgrid is built around the gridcube to see if there are any particles whose centers are not within the gridcube, but nonetheless are partially located in the gridcube.

```
float cubegrid_occupied_volume(. . .) {
    const float subgrid_edge_length =
        (float) cube_edge_length / subgrid_edge_numcubes;
    const int particle_radius_discrete = (int)
        ceil(particle_radius / subgrid_edge_length);
    const int half_bond_distance_discrete = (int)
        ceil(0.5 * bond_distance / subgrid_edge_length);
    const int expanded_subgrid_edge_numcubes =
        subgrid_edge_numcubes + 2 *
        half_bond_distance_discrete;
    const int final_subgrid_total_cubes =
        subgrid_edge_numcubes * subgrid_edge_numcubes *
        subgrid_edge_numcubes;
    const int expanded_subgrid_total_cubes =
        expanded_subgrid_edge_numcubes *
        expanded_subgrid_edge_numcubes *
        expanded_subgrid_edge_numcubes;
    const int max_particles_in_gridcube = (int) 20 *
        (expanded_subgrid_edge_numcubes *
        expanded_subgrid_edge_numcubes *
        expanded_subgrid_edge_numcubes) /
        (particle_radius_discrete *
        particle_radius_discrete *
        particle_radius_discrete);
```

The list of particles is passed through, to see if any particle is in the gridcube or surrounding gridcubes; if so, the coordinates are transferred.

```
    for(int i=0; i<totalnumparticles; i++) {
        if((positionlist[i][0]>(x_origin - bond_distance))
            && (positionlist[i][0] <
            (x_origin +cube_edge_length +bond_distance)) &&
            (positionlist[i][1]>(y_origin - bond_distance))
            && (positionlist[i][1] <
            (y_origin +cube_edge_length +bond_distance)) &&
            (positionlist[i][2]>(z_origin - bond_distance))
            && (positionlist[i][2] <
            (z_origin + cube_edge_length + bond_distance))
            ) {
            if(particles_in_gridcube <
                max_particles_in_gridcube) {
                intersecting_particle_centers
                    [particles_in_gridcube][0]
                    = positionlist[i][0];
                intersecting_particle_centers
                    [particles_in_gridcube][1]
                    = positionlist[i][1];
                intersecting_particle_centers
                    [particles_in_gridcube][2]
                    = positionlist[i][2];
                particles_in_gridcube++;
            }
            else {
                cout << "Error: Too many particles
                    in gridcube!" << endl;
            }
        }
    }
```

Then, a first pass is made through the subgrid, and subgrid cubes are assigned to the free volume if a test particle of radius a, centered in that particular subgrid cube, does not intersect any particles in the percolated cluster:

```
for(i=0; i<expanded_subgrid_edge_numcubes; i++) {
for(j=0; j<expanded_subgrid_edge_numcubes; j++) {
for(k=0; k<expanded_subgrid_edge_numcubes; k++) {
    subcube_is_near_particle = false;
    x_pos = x_origin + (i - half_bond_distance_discrete)
        * subgrid_edge_length;
    y_pos = y_origin + (j - half_bond_distance_discrete)
        * subgrid_edge_length;
    z_pos = z_origin + (k - half_bond_distance_discrete)
        * subgrid_edge_length;
    for(p=0; p<particles_in_gridcube; p++) {
    r2 = (x_pos - intersecting_particle_centers[p][0])
        * (x_pos - intersecting_particle_centers[p][0])
        + (y_pos - intersecting_particle_centers[p][1])
        * (y_pos - intersecting_particle_centers[p][1])
        + (z_pos - intersecting_particle_centers[p][2])
        * (z_pos - intersecting_particle_centers[p][2]);
        if (r2 < threshold_r2) {
            subcube_is_near_particle = true;
        }
    }
    if (subcube_is_near_particle == false) {
        subcube_is_in_freevolume[Index_3Dcube_to_1D(i,j,k,
            expanded_subgrid_edge_numcubes)]=true;
    }
}
}
}
```

In the second pass through the subgrid, around every subgrid cube found in the first pass to contain the center of a particle in the free volume, the other subgrid cubes that fill in the volume of that particle are also added to the free volume. This strategy automatically takes care not to open up holes within a dense gel, and thus is able to approximate the correct internal volume.

```
for(i=0; i<expanded_subgrid_edge_numcubes; i++) {
for(j=0; j<expanded_subgrid_edge_numcubes; j++) {
for(k=0; k<expanded_subgrid_edge_numcubes; k++) {
    if(subcube_is_in_freevolume[Index_3Dcube_to_1D(i,j,k,
        expanded_subgrid_edge_numcubes)] == true &&
        neighborcount_3D(subcube_is_in_freevolume,i,j,k,
        expanded_subgrid_edge_numcubes) < 26
        ) {
        . . .
        for(x=xmin; x<xmax; x++) {
        for(y=ymin; y<ymax; y++) {
        for(z=zmin; z<zmax; z++) {
            if( ((x-i)*(x-i) + (y-j)*(y-j) + (z-k)*(z-k))
            * (subgrid_edge_length * subgrid_edge_length)
            < (particle_radius * particle_radius)
            ) {
            subgrid_freevolume_full[Index_3Dcube_to_1D(x,
            y,z,expanded_subgrid_edge_numcubes)]=true;
            }
        }
        }
        }
    }
}
}
}
```

The portion of the subgrid array that corresponds to the original gridcube is then copied, and the number of subgrid cubes that are part of the cluster (i. e. not in the free volume) are then enumerated:

```
int num_subcubes_in_freevolume=0;
    for(i=0; i<final_subgrid_total_cubes; i++) {
        if(subgrid_final[i] == true) {
            num_subcubes_in_freevolume++;
        }
    }
```

These are then converted to a physical volume, which the function returns.

```
    float volume_in_cluster = (final_subgrid_total_cubes -
        num_subcubes_in_freevolume) * (cube_edge_length *
        cube_edge_length * cube_edge_length /
        final_subgrid_total_cubes);
    return volume_in_cluster;
```

It is this physical volume calculated on the subgrid that is summed up to equal the total volume for each larger gridcube that contains at least part of a particle. Because the variable `internal_cluster_volume` is increased with the `+=` operator every time the function `cubegrid_occupied_volume()` called, `internal_cluster_volume` automatically represents the total cluster volume once the entire cubic grid has been passed through. The cluster volume fraction ϕ_g is then a simple quotient:

```
    perc_vol_frac=
        (single_particle_volume*num_perc_particles) /
        internal_cluster_volume;
```

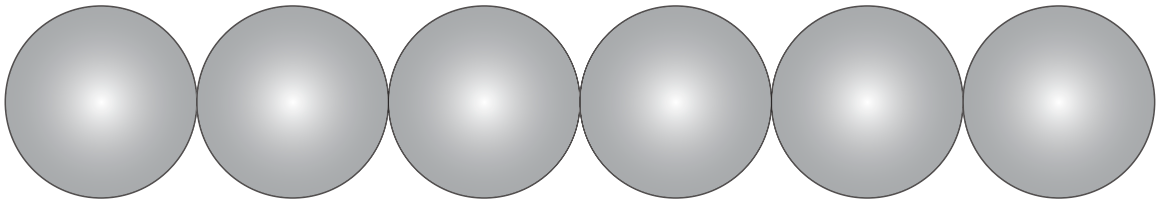

Figure 7.4: Straight line of touching grey spheres.

7.6.3 Measurements of known structures

We demonstrate this algorithm on two structures for which the internal volume can be calculated analytically, a straight line of touching spheres, and 3D clusters of spheres packed into a face-centered cubic (FCC) lattice.

The schematic of the line of particles is shown in figure 7.4. The “cluster” is an infinite chain of spheres in a straight line, colored in grey.

A green test particle is free to move anywhere away from the cluster, but cannot penetrate any of the particles, and cannot get any closer to the line through the center of the spheres than the position touching two of them, as shown in figure 7.5. The volume around the grey spheres that the test sphere cannot enter into is designated the *excluded* volume, marked in red in figure 7.5. This red volume is considered part of the cluster, and not part of the *free* volume, the total space accessible to the test sphere (i.e. that is neither within the grey spheres, or part of the excluded volume).

The cluster volume fraction is defined as the total volume of the spheres within a cluster divided by the total volume of the cluster, illustrated in figure 7.6 (where only the colors are kept for clarity, but actually correspond to spheres, as shown in figure 7.5). The region colored blue is the volume belonging to the particles, which by symmetry can be extended to the whole chain. The red region is, as before, the excluded volume. The total cluster

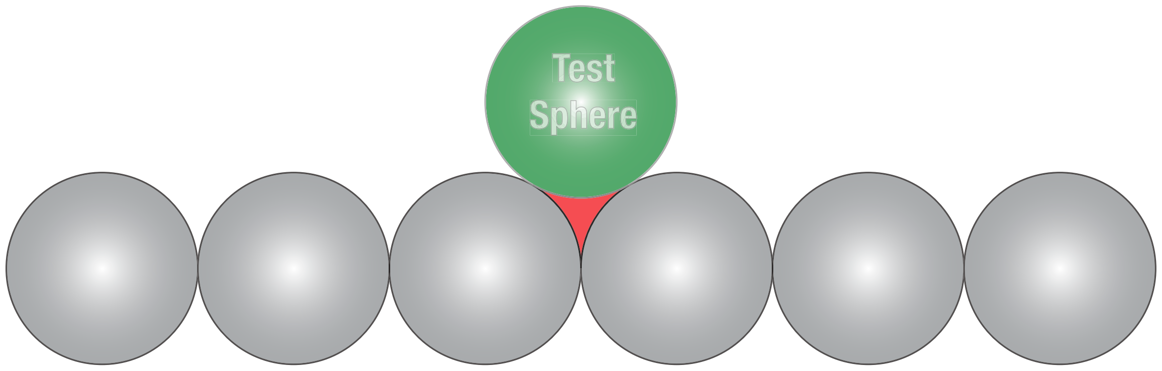

Figure 7.5: Straight line of touching grey spheres, with green test sphere unable to penetrate them. The green test sphere is shown touching two grey spheres, at the position closest to the line going through the center of the touching grey spheres. The region that the green test sphere cannot enter is called the excluded volume, and is marked in red.

volume corresponds to the blue + red regions, while the volume of the spheres within the cluster is just the blue regions. The volume fraction of the cluster, then, is the volume of the blue region divided by the combined volumes of the blue and red regions.

Because the test particle is identical to the spheres that comprise the cluster, the volume fraction can be calculated analytically with a few straightforward volume integrals. First, the region in blue, corresponding to the region of one of the spheres in figure 7.6 is, simply:

$$\begin{aligned} V_1 &= \int \pi [y(x)]^2 dx = \pi \int_0^R (R^2 - x^2) dx && (7.12) \\ &= \pi \left(R^2 x - \frac{x^3}{3} \right) \Bigg|_0^R = \pi R^3 \left(1 - \frac{1}{3} \right) && (7.13) \\ &= \frac{2\pi R^3}{3} && (7.14) \end{aligned}$$

For the cluster volume (blue + red), it is convenient to divide the volume into two portions, as shown in figure 7.7. The cluster volume in this representation corresponds to the blue + purple. The blue portion is again part of a sphere, and therefore straightforward to calculate.

$$V_2 = \pi x \left(R^2 - \frac{x^2}{3} \right) \Bigg|_0^{R/2} \qquad (7.15)$$

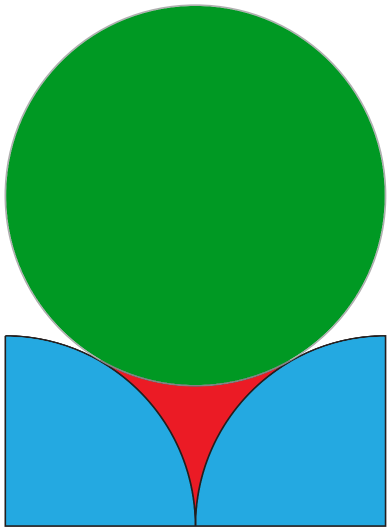

Figure 7.6: Region of figure 7.5 zoomed in, without shading. With our metric to estimate ϕ_g, the blue regions are part of the spheres and of the cluster, whereas the red region is the part of the volume of the cluster that is outside the spheres. The volume fraction of the cluster ϕ_g is the volume of the blue region divided by the combined volumes of the blue and red regions.

$$= \pi \frac{R}{2}\left(R^2 - \frac{R^2}{12}\right) \tag{7.16}$$

$$= \frac{11}{24}\pi R^3 \tag{7.17}$$

The final piece is the region marked in purple, the area between the green circle of radius R, centered at $(0, \sqrt{3}R)$, and the $\hat{x}$-axis, over the range $0 < x < R/2$. The circle is described by $x^2 + (y - \sqrt{3}R)^2 = R^2$, so that the lower half of the circle is:

$$y(x) = \sqrt{3}R \pm \sqrt{R^2 - x^2} \tag{7.18}$$

$$y^2(x) = 3R^2 + (R^2 - x^2) - 2\sqrt{3}R\sqrt{R^2 - x^2} \tag{7.19}$$

$$= 4R^2 - x^2 - 2\sqrt{3}R\sqrt{R^2 - x^2} \tag{7.20}$$

The volume of the purple region, is then:

$$V_3 = \pi \int_0^{R/2} (4R^2 - x^2)dx - 2\sqrt{3}\pi R \int_0^{R/2} \sqrt{R^2 - x^2}dx \equiv V_3^{\mathrm{A}} - V_3^{\mathrm{B}} \tag{7.21}$$

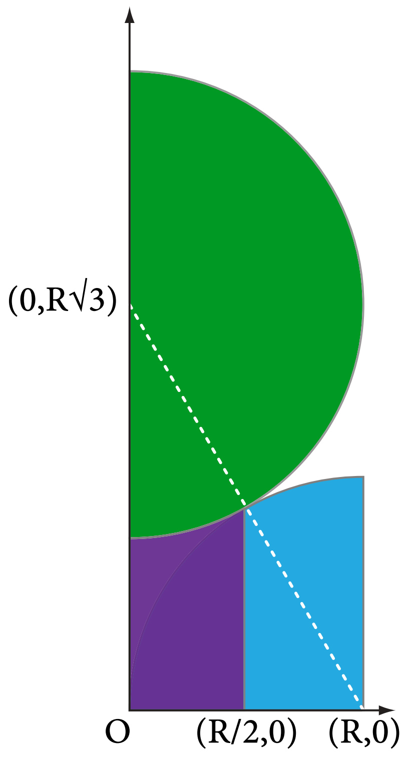

Figure 7.7: Half of figure 7.6, with new area marked in purple that represents part of the cluster volume.

The first term is a simple integral:

$$
\begin{aligned}
V_3^{\mathrm{A}} &= \pi \left(4R^2 x - \frac{x^3}{3} \right) \bigg|_0^{R/2} && (7.22)\\
&= \pi \left(\frac{4R^3}{2} - \frac{R^3}{24} \right) && (7.23)\\
&= \frac{47}{24} \pi R^3 && (7.24)
\end{aligned}
$$

The second term requires a trigonometric substitution:

$$
\begin{aligned}
x &= R\sin(u) && (7.25)\\
dx &= R\cos(u)du && (7.26)\\
u &= \arcsin\left(\frac{x}{R}\right) && (7.27)
\end{aligned}
$$

The limits of integration transform to $\arcsin(0) = 0$ and $\arcsin(1/2) = \pi/6$, so that:

$$
\begin{aligned}
V_3^{\mathrm{B}} &= 2\sqrt{3}\pi R \int_0^{R/2} \sqrt{R^2 - x^2} dx && (7.28)\\
&= 2\sqrt{3}\pi R \int_0^{\pi/6} \sqrt{R^2 - R^2 \sin^2(u)} \cdot R\cos(u)du && (7.29)\\
&= 2\sqrt{3}\pi R \int_0^{\pi/6} \left[R\cos(u)\right]^2 du && (7.30)\\
&= 2\sqrt{3}\pi R^3 \int_0^{\pi/6} \cos^2(u)du && (7.31)\\
&= 2\sqrt{3}\pi R^3 \left(\frac{u}{2} + \frac{\sin(2u)}{4} \right) \bigg|_0^{\pi/6} && (7.32)\\
&= \pi R^3 \left(\frac{\sqrt{3}\pi}{6} + \frac{3}{4} \right) && (7.33)
\end{aligned}
$$

where a standard trigonmetric integral identity has been used (equation 440.20 in [82]). Combining the two terms,

$$
V_3 = \pi R^3 \left(\frac{47}{24} - \frac{\sqrt{3}\pi}{6} - \frac{3}{4} \right) = \pi R^3 \left(\frac{29 - 4\sqrt{3}\pi}{24} \right) \qquad (7.34)
$$

The final volume fraction is,

$$\begin{aligned}
\phi_{\mathrm{g}} &= \frac{V_1}{V_2 + V_3} && (7.35)\\
&= \frac{\frac{2}{3}\pi R^3}{\pi R^3 \left(\frac{11}{24} + \frac{29-4\sqrt{3}\pi}{24}\right)} && (7.36)\\
&= \frac{16/24}{(11 + 29 - 4\sqrt{3}\pi)/24} && (7.37)\\
&= \frac{16}{40 - 4\sqrt{3}\pi} && (7.38)\\
&= \frac{4}{10 - \sqrt{3}\pi} \cong 0.877 && (7.39)
\end{aligned}$$

The program asymptotes to this theoretical value for a line of spheres as the subgrid size l'_{c} shrinks, as illustrated in figure 7.8.

We also examine the internal volume fraction of finite clusters of spheres packed into an FCC lattice, with a known internal volume fraction of $\sqrt{2}\pi/6 \cong 0.74$ [11]. An example showing the overlay of gridcubes on a small FCC cluster $3 \times 3 \times 3$ unit cells is shown in figure 7.9. We use the algorithm to measure the internal volume fraction of two clusters of FCC-packed spheres of radius $a = 0.56\ \mu$m, with $10 \times 10 \times 10$ and $25 \times 25 \times 25$ unit cells, as function of grid size. In both cases, the values estimated by the program code approach the theoretically well-known value of $\phi \to 0.74$ as l'_{c} shrinks, as shown in Fig. 7.10.

Both examples provide confirmation that the algorithm behaves as expected, and reproduces the internal volume fraction of structures for which this value can be analytically calculated.

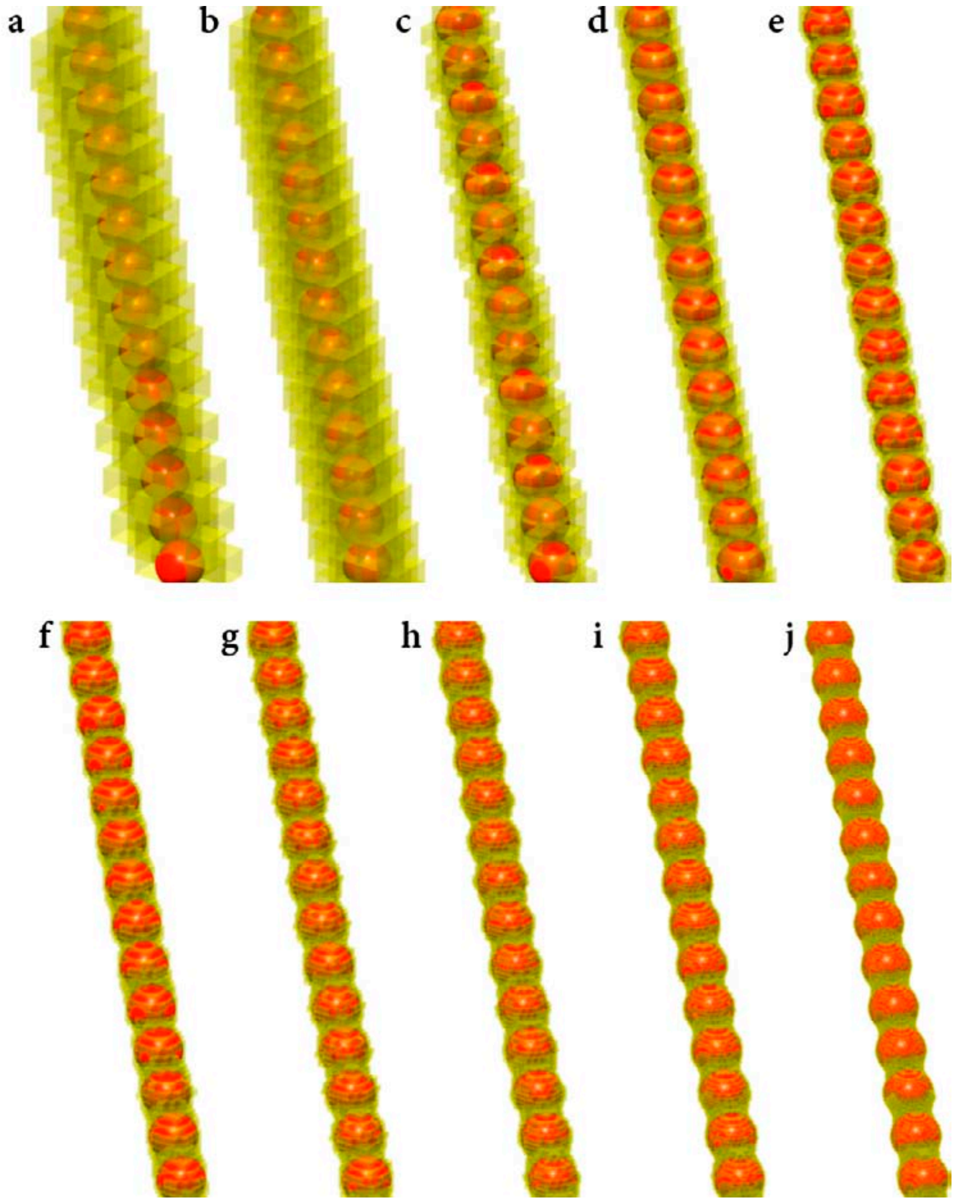

Figure 7.8: One-dimensional chain of red spheres of radius a, with subgrid cubes of edge length l'_c rendered in transparent yellow, showing how l'_c affects the calculation of cluster volume fraction ϕ_g. **(a)** $l'_\mathrm{c} = 1.2a$, $\phi_\mathrm{g} = 0.16$. **(b)** $l'_\mathrm{c} = a$, $\phi_\mathrm{g} = 0.19$. **(c)** $l'_\mathrm{c} = 0.8a$, $\phi_\mathrm{g} = 0.37$. **(d)** $l'_\mathrm{c} = 0.6a$, $\phi_\mathrm{g} = 0.54$. **(e)** $l'_\mathrm{c} = 0.4a$, $\phi_\mathrm{g} = 0.73$. **(f)** $l'_\mathrm{c} = 0.3a$, $\phi_\mathrm{g} = 0.76$. **(g)** $l'_\mathrm{c} = 0.25a$, $\phi_\mathrm{g} = 0.73$. **(h)** $l'_\mathrm{c} = 0.2a$, $\phi_\mathrm{g} = 0.75$. **(i)** $l'_\mathrm{c} = 0.15a$, $\phi_\mathrm{g} = 0.80$. **(j)** $l'_\mathrm{c} = 0.1a$, $\phi_\mathrm{g} = 0.83$. For $l'_\mathrm{c} = 0.05a$, $\phi_\mathrm{g} = 0.85$ (not shown). These data demonstrate how, as the grid size shrinks, the estimated cluster volume fraction increases, converging to the theoretical value of 0.877 calculated in equation 7.39.

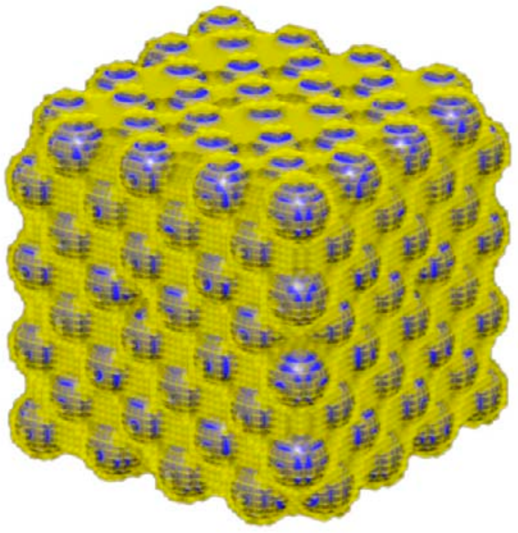

Figure 7.9: Cluster of $3 \times 3 \times 3$ FCC unit cells. In this case, the gridcube edge length is 0.28 μm.

7.6.4 Application to Gel Clusters

The main application of the algorithm is to estimate the internal volume fraction ϕ_g of disordered gel and cluster structures where ϕ_g cannot be calculated analytically. We illustrate some grids of different l'_c for the gel used in the above examples in this chapter, in figure 7.11. Graphically, the boxes track the particle structure closely, with increasing fidelity as l'_c decreases. For very small l'_c, the 3D structure of the gel is in some ways more easily visualized in the boxes than by looking at the particles directly (the algorithm in effect rasterizes the gel structure onto a discrete cubic grid), as shown in figure 7.12, where $\phi_\mathrm{g} = 0.58$.

The results of this ϕ_g calculation for all of the other gels with short-ranged attraction are illustrated in figure 10.15, where we selected $l'_\mathrm{c} = 0.25a$, but the measured ϕ_g values do not depend on l'_c for values below $\approx a/2$.

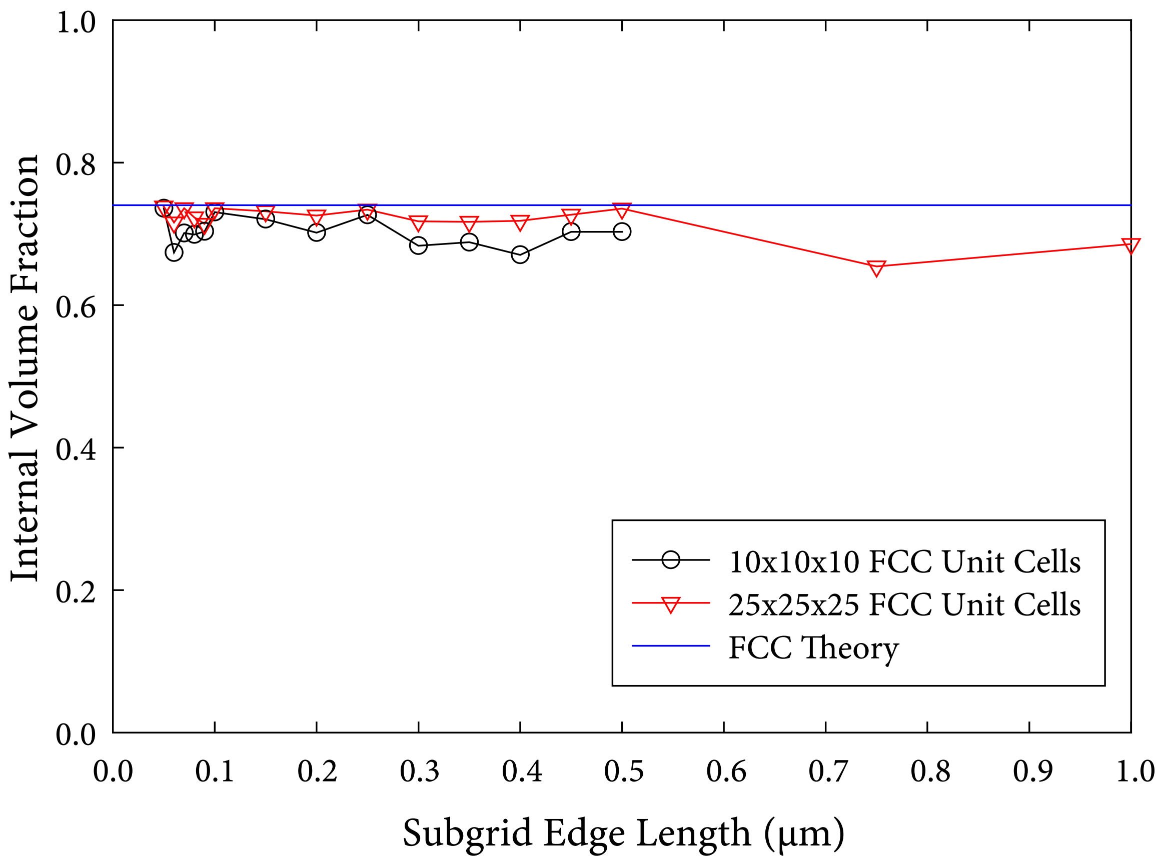

Figure 7.10: Internal cluster volume fraction as a function of subgrid size for finite clusters of FCC close-packed hard spheres. Results for clusters 10 and 25 spheres on a side are shown in black circles and red triangles, respectively. The theoretical value is shown in solid blue line.

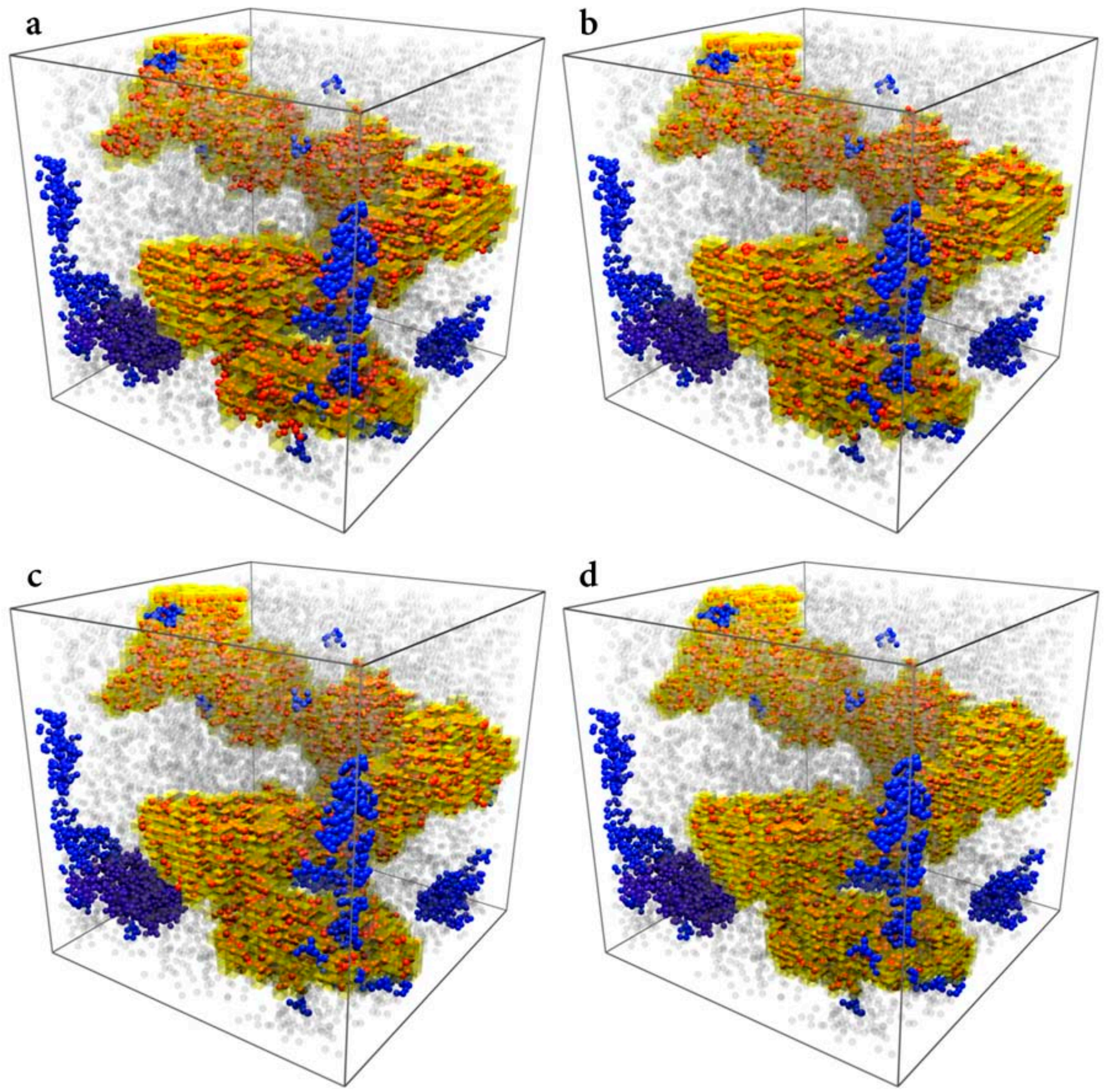

Figure 7.11: Example gel from figure 6.13, with subgrid cubes with edge length l'_c rendered in transparent yellow. **(a)** $l'_\mathrm{c} = 4.3a$. **(b)** $l'_\mathrm{c} = 3.6a$. **(c)** $l'_\mathrm{c} = 2.9a$. **(d)** $l'_\mathrm{c} = 2.1a$. The boxes track the particle structure closely, with increasing fidelity as l'_c decreases. Note that the l'_c values shown in the renderings in this figure are far larger than those used to calculate the actual volume to generate the phase diagram in figure 10.15, where the value used there of $l'_\mathrm{c} = 0.25a$ generates too many boxes to render on my workstation.

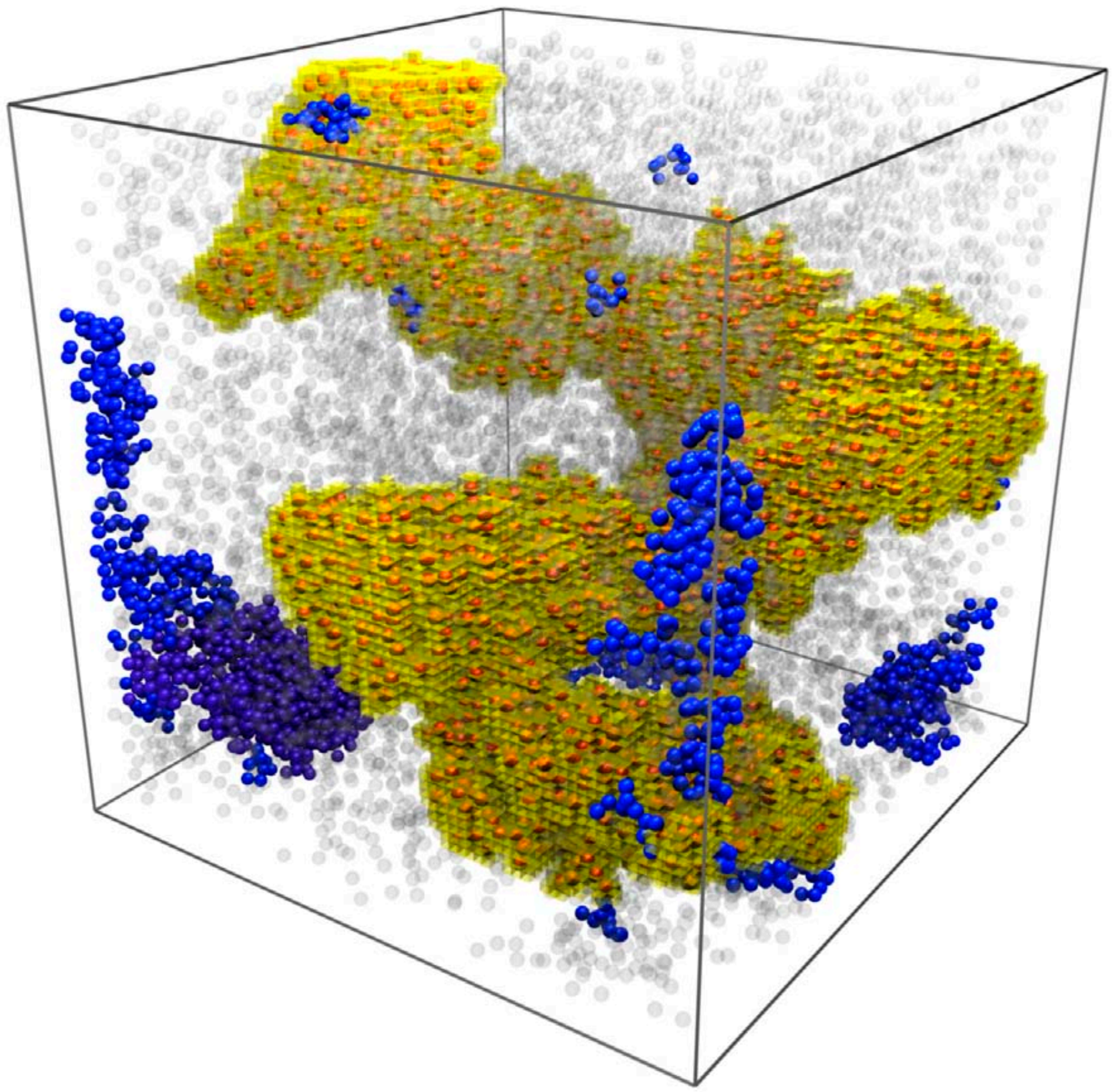

Figure 7.12: Example gel from figure 6.13, with subgrid cubes of edge length $l'_c = 1.4a$ rendered in transparent yellow. The boxes track the particle structure closely; in some ways, the 3D structure of the gel is more easily visualized in the boxes than by looking at the particles directly, as the algorithm in effect rasterizes the gel structure onto a discrete cubic grid.

7.7 Acknowledgements

I thank Eli Sloutskin and Tom Kodger for comments on this chapter. Dave Weitz conceived the idea for using the free volume of a test particle for the measurement of ϕ_{g}.

Chapter 8

Three-dimensional Visualization and Rendering

8.1 Introduction

Groups of spherical colloidal particles, whose positions are measured in three dimensions with confocal microscopy, have long been presented as rendered, or ray-traced, structures. For a general primer on rendering concepts used in this chapter—and, indeed, in all modern films with computer-generated animation or special effects—I highly recommend [213]. In the colloid and soft-matter physics communities, the most commonly used software applications are MATLAB (with its embedded OpenGL based rendering capabilities) and POV-Ray, a free open-source ray-tracer.

With colloids in particular, one of the major advantages of rendering is to simultaneously display the 3D structure while conveying other types of structural information through attributes of displayed particles, such as color, size, opacity and surface material.

For crystals, features highlighted in this manner include bond-orientation order parameters [102], stacking faults [233], and nearest-neighbor distortions due to defects [234]; for glasses, rates of relaxation [296] and local shear strains [235].

For crystals and glasses, where the colloids are dense, rendering is a relatively straightforward task. Typically, only a few hundred particles are displayed, since there is little additional information to be gained by showing many more particles. For crystals, perfect periodic translational order means that showing a few unit cells is often enough to demonstrate the full long-range structural order [156]. Moreover, for larger crystalline structures, which are densely packed, internal particles are not visible, and therefore only the surface of what is essentially a prismatic box is shown [233, 234, 279, 278]. Alternatively, sections of dense crystal [102] and glass [296] structures can be visualized by shrinking the vast majority of particles until they appear as structureless small dots. But these previous examples involve the rendering of only a few hundred spheres, with only rudimentary lighting and shading. The main emphasis is on coloring or sizing of spheres within structures that are long known (glass and FCC crystal); the structure itself is not particularly important in the visual presentation.

With disordered gels and phase-separating systems at lower volume fractions, however, a different set of constraints apply. The structure itself is of great interest, and most of the particles, which can number into the tens of thousands, cannot simply be shrunken to nondescript dots. Moreover, visualizing and lighting these structures is not an altogether obvious, trivial or easily-automated process. Rather, for each gel structure, individual lighting and shading has to be applied, depending on the scientific point to be communicated. This requires far more powerful specialized visualization software—that far more efficiently renders a large number of objects—than what is available with the above-mentioned, commonly-used software packages, which have limited previous work to rendering only very small sections of disordered colloid structures [94, 80, 73]. Instead, moving beyond these constraints requires tools from the computer-graphics, animation and movie industries.

8.2 OpenGL

My first attempt to show these structures in 3D was to use the OpenGL language to control directly the graphics hardware present on a video card [189, 190], displaying each particle as a simple sphere. Changing color, opacity and size are very straightforward in OpenGL, and displaying images this way requires only a small amount of coding. A very simple C++ program controls the loading of a text file with particle information, and then uses a few powerful but concise OpenGL commands to create the window for display and then to render the particles in 3D. The `main()` function is very brief (where housekeeping details have been omitted):

```
int main(int argc, char** argv)
{
    . . .
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE|GLUT_DEPTH|GLUT_RGB);
    window_width = 800;
    window_height = 800;
    glutInitWindowSize(window_width, window_height);
    glutInitWindowPosition(100, 100);
    glutCreateWindow("Peter Lu’s 3D Gel Display");
    init(translationlist, numparticles);
    . . .
    // Give control over to glut.
glutMainLoop();
return 0;
}
```

There are only half a dozen OpenGL commands, with self-explanatory functions, plus one final function to prepare the particle data: `init()`. This function is passed an array containing particle position data and first initializes specific OpenGL parameters; then it generates a list of spheres to display, setting position, size and color based on the array data.

```
void init(const double (*spheredata)[NUMCOLUMNS],
    const int numparticles)
{
    GLUquadricObj *qobj;
    GLfloat mat_ambient[] = {0.1, 0.3, 0.6, 1.0 };
    GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
    GLfloat mat_shininess[] = { 80.0 };
    GLfloat light_position[] = { 1.0, 1.0, 1.0, 0.0 };
    GLfloat model_ambient[] = { 1.0, 1.0, 1.0, 1.0 };
    glClearColor(0.0, 0.0, 0.0, 0.0);
    glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
    glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);
    glLightfv(GL_LIGHT0, GL_POSITION, light_position);
    glLightModelfv(GL_LIGHT_MODEL_AMBIENT, model_ambient);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    glEnable(GL_DEPTH_TEST);
    GLfloat fogcolor[4]={0.0, 0.0, 0.0, 0.0};

    glFogfv(GL_FOG_COLOR, fogcolor);
    glFogf(GL_FOG_START, 9.7);
    glFogf(GL_FOG_END, 10.2);
    glFogi(GL_FOG_MODE, GL_LINEAR);
    glEnable(GL_FOG);

    startList = glGenLists(1);
    qobj = gluNewQuadric();
    gluQuadricDrawStyle(qobj, GLU_FILL);
    gluQuadricNormals(qobj, GLU_SMOOTH);
    glNewList(startList, GL_COMPILE);
    for(int i=0; i<numparticles; i++) {
        glTranslatef(spheredata[i][0], spheredata[i][1],
            spheredata[i][2]);
        mat_ambient[0] = spheredata[i][4];
        mat_ambient[1] = spheredata[i][5];
        mat_ambient[2] = spheredata[i][6];
        glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
        gluSphere(qobj, spheredata[i][3], 15, 10);
    }
    glEndList();
}
```

The `glutMainLoop()` function then takes over the windowing interface, and allows mouse-driven control of viewing angle, zooming and panning. The image saved to disk is essentially a screen capture, read out from the image buffer of the video card. A simple way to simulate the effects of depth was to add fog, so that particles more distant from the viewer fade into the background of a specifiable color. An example image (created six years ago, in 2002) is shown in figure 8.1.

The main advantage of this approach is speed. There are fewer ways to more quickly render thousands of spheres than by using optimized graphics hardware within the graphics processing unit (GPU) of the video card. Structures with thousands of particles can be rendered in real time, so that dragging the mouse in the window allows rapid repositioning of particle structures without the need for a time-consuming final rendering at the end.

However, there are a number of serious drawbacks to this direct use of OpenGL. First, any changes to the imaging parameters (for instance, even something simple like the position of the lights) requires a recompilation, making setting up the image time-consuming and tedious. The quality of the final image, for this simple implementation, is also rather unimpressive (certainly no improvement upon MATLAB, which also uses OpenGL). Sec-

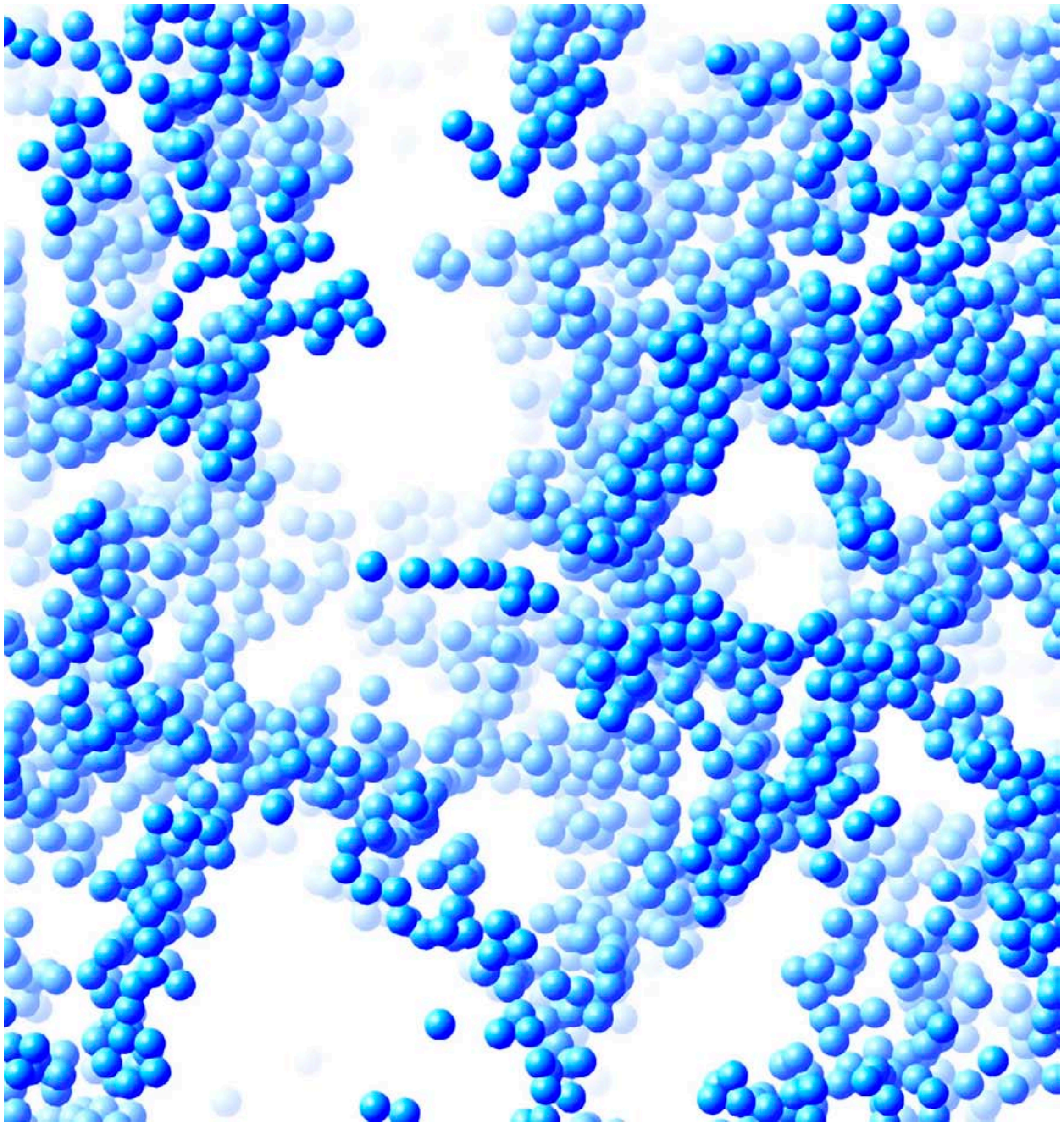

Figure 8.1: Gel structure rendered with OpenGL. All particles in the gel are colored the same, and fog has been added in an early attempt to simulate depth; deeper particles fade into the white background.

ond, the interactivity may be convenient, but is not particularly useful when generating the *final* high-quality image. Even if a fine adjustment of position requires a few seconds (several orders of magnitude slower than the video rates that the graphic cards support when displaying these structures) there is no significant loss in productivity, and it certainly is not worth the quality trade-off. But perhaps most importantly, improving the image involves adding progressively more advanced commands within OpenGL—but this is essentially reinventing the wheel by trying to write a high-quality renderer from scratch. This is a problem that has been solved many ways, and so I abandoned OpenGL, after these initial attempts, in favor of a commercial solution.

8.3 Alias|WaveFront Maya

Probably the most popular commercially-available modeling and rendering software is Maya, made by Alias|Wavefront when I used it (version 6.5), but since acquired by Autodesk. An extremely powerful and complicated package, it allows interactive modeling of 3D data, which can then be rendered with several renderers (Maya software, graphics-card hardware, and Mental Ray are included with the default package). Once a collection of particles is imported, the viewpoint, camera angle, zoom, lighting and shadows can all be adjusted. After the geometry is properly set and lighting and textures applied, a final scene is rendered. All parameters can be adjusted interactively, without any programming. This is particularly powerful when lighting a disordered gel phase with mesoscale structure (i.e. on the length scale of several particles) that has large inhomogeneities. All of these objects and parameters, including lighting and the camera, can be animated, also without coding. This enables a host of new visualization options not really possible without a great deal of pain in MATLAB or POV-Ray, such as flying through a gel structure, or animating successive 3D stacks while the lighting and camera moves. The only programming involves formatting the analyzed particle data, after the processing described in chapter 7. I took two different approaches within Maya, for different purposes.

8.3.1 Particle Data Cache

First, I hacked the Particle Data Cache (PDC) available to particle systems in Maya, to quickly load in large collections of particles. This extremely inelegant and ugly solution is nonetheless perhaps the highest-performance approach, as all particles share the same geometry, so it's far simpler for the program to display during modeling and final render. This is a double-edged sword, however, since it does not allow changing the attributes (such as radius, or color) of individual particles—there is only one object, which has many instances. This was the chosen technique for time-series movies, where successive 3D stacks comprise movie frames. Here, performance is crucial, since a different collection of thousands of particles is rendered in each frame. Using Maya particles was the only way for the Maya modeler to be fast enough to work with thousands of particles with some modicum of interactivity, needed to place and animate the camera and lights to maximize clarity through a changing time-series. The source code, and complete details of the Maya PDC format, are available on the San Diego Supercomputer Center (SDSC) site at:

`http://visservices.sdsc.edu/software/maya/utils/pdc.php`.

8.3.2 Maya Embedded Language (MEL)

For single images for publication, where quality is paramount, I used a second approach, utilizing the Maya Embedded Language (MEL) [111, 299, 264] to import a large collection of spheres into the Maya workspace. Based on the cluster size analysis, I set a threshold value: particles belonging to a cluster larger than this value are assigned to one group and rendered at full size; particles below that (e.g. monomers and dimers) are assigned to a second group and rendered at a smaller size, mimicking a visual scheme used previously [296, 102].

```
outMELdatafile.open(outMELfilename.c_str(),ios::out);
outMELdatafile << "group -name bigcluster -em;
    group -name smallcluster -em;" << endl;

for(int row = 0; row < totalnumparticles; row++) {
    outMELdatafile << "sphere -radius ";
    if(positionlist[row][column_for_thresh]
        > value_threshold) {
        outMELdatafile << real_radius;
    }
    else {
        outMELdatafile << shrunken_radius;
    }
    outMELdatafile << " -name colloid" << row<< "; move ";
    outMELdatafile << positionlist[row][1]-xcenter << " ";
    outMELdatafile << positionlist[row][2]-ycenter << " ";
    outMELdatafile << positionlist[row][3]-zcenter << " ";
    outMELdatafile << "colloid" << row
        << "; parent colloid" << row;
    if(positionlist[row][column_for_thresh]
        > value_threshold) {
        outMELdatafile << " bigcluster";
    }
    else {
        outMELdatafile << " smallcluster";
    }
    outMELdatafile << ";" << endl;
```

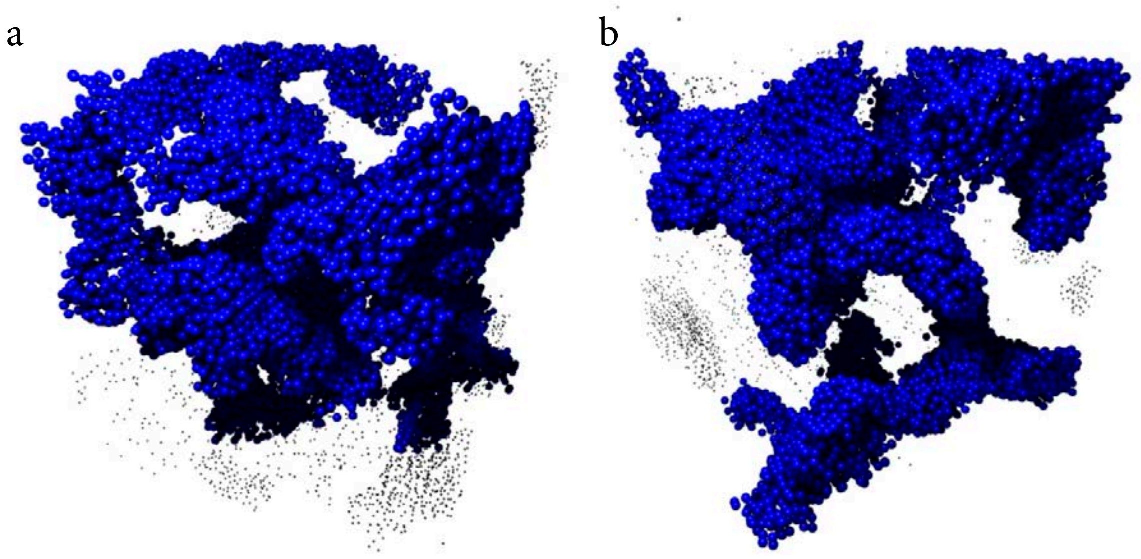

Figure 8.2: Rendering of a gel structure with surrounding clusters created with Maya. The two images are different rotations of the structure around the vertical axis. The lights remained fixed, so that shadows are cast from the same point, and in the same direction.

```
}
outMELdatafile.close();
```

Once these spheres have been imported in two separate groups (with their radii already set), their other attributes, particularly color, opacity and surface material, can be adjusted interactively in Maya without further coding. I used this technique to create the renderings in chapter 9 (published as [171]). Renderings from images in the same sequence as figure 9.4, but at two different rotations of the entire structure around the vertical axis, are shown in figure 8.2. Note that the lights are fixed even though the structure rotates, so that the shadows are cast from the same angle in both images. The gel particles were selected to be blue, with a reflective phong shader. The smaller monomers were selected to be dark gray, and non-reflective.

The main challenge with gel structures is communicating the depth of the structure. So lights were set to cast shadows, and for their intensity to fall off as $1/r^2$, as in real-life.

After the best viewing angle was selected that best highlights the 3D structure, colors of the two groups of particles, as well as the direction and intensity of the lighting, were all optimized interactively. The final, high-resolution version of the image in [171] is shown in figures 8.3 and 9.4.

8.4 Pixar's RenderMan

While interactive modeling is useful for making fine adjustment to individual images, the fact that it is required in Maya is less than ideal for rendering large sets of similar data in an automated and fast way. Moreover, the particular task of visualizing colloidal particles is one where modeling is actually not required: the particles are represented as spheres, and their positions and sizes are prescribed ahead of time. All of the capabilities that interactive modeling offers—for instance, the ability to move an individual vertex to distort the surface of any individual sphere in an exact way—are in fact never used, and instead serve only to decrease performance. This particular modeling task is made even easier because spheres (and other geometric primitives like cubes, cones, rods, etc.) are built into almost all modern rendering engines (as is the case with OpenGL), so reconstructing colloidal structures is simply a matter of translating the 3D coordinates of the particles in an efficient way to a format that the rendering engine can use directly. So a more efficient strategy is simply to create externally the geometry in the form of sphere center coordinates, and then pass the list directly to a rendering engine.

There are, of course, many choices for modern rendering engines. Maya itself has several, including the increasingly well-regarded Mental Ray. A pure ray-tracing engine, like POV-Ray, has been used in the past, but this popular free, open-source program is extremely

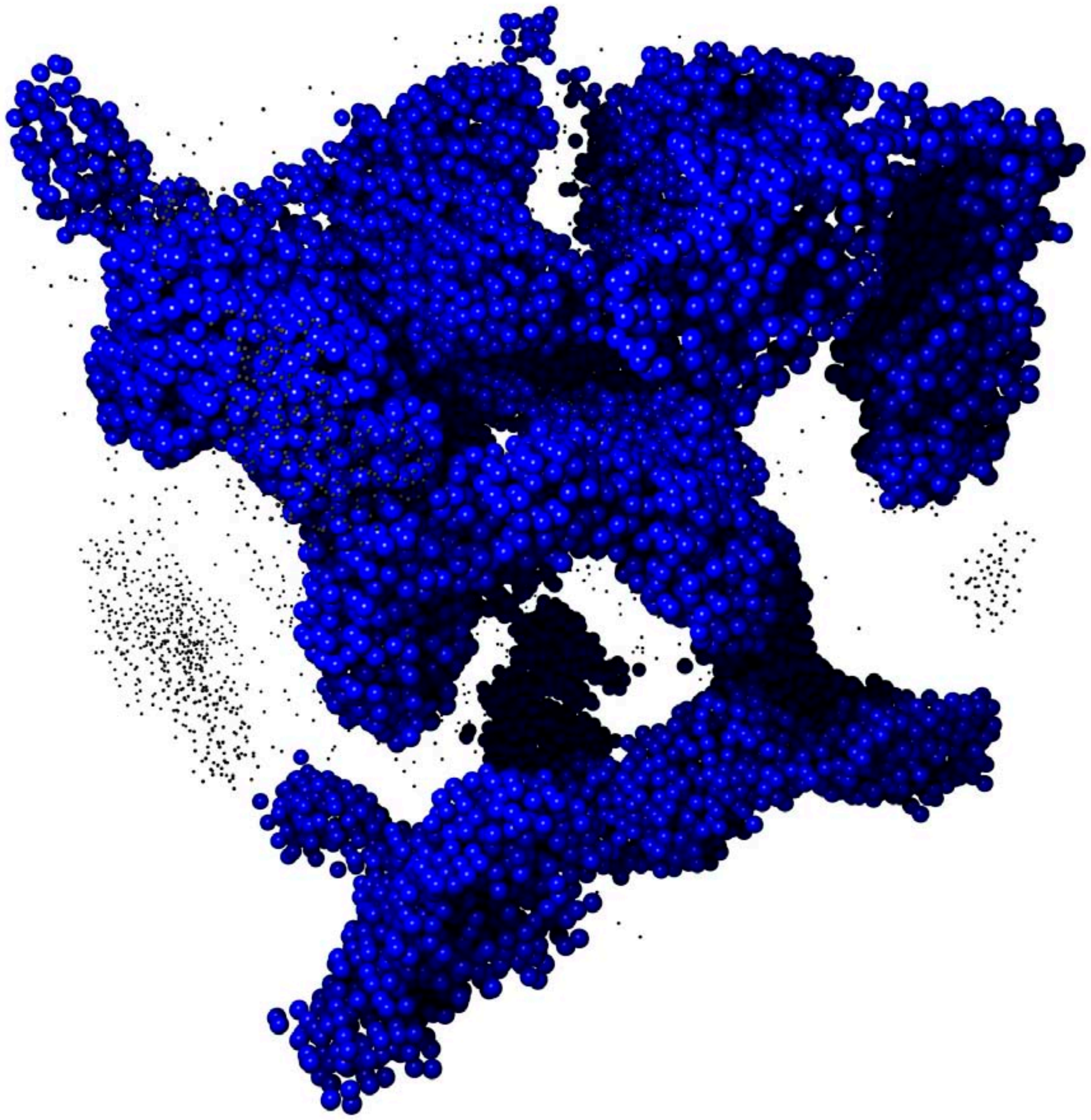

Figure 8.3: Final rendering of a gel structure created with Maya, a larger version of the structure shown in figure 9.4.

slow and falling further out of date; at the time of the writing of this thesis, the current version is nearly four years old, is explicitly opposed to acceleration with graphics hardware or processor extensions—and the new beta version is only now incorporating multiprocessor support. Though a favorite of quixotic scientists, this and similar free programs simply can't compete with the commercial rendering engines designed to process huge data sets on large, massive-parallel computer clusters, whose performance and quality are driven by feature films with billion-dollar revenue streams. Very recently, extremely good performance has been achieved with high-quality renders that leverage hardware acceleration, such as the GPU-based Gelato renderer by nVidia, but this is a very new technology, tied explicitly to hardware that changes frequently. So it remains to be seen whether this will be a stable long-term path worth investing a lot of time into.

Instead, I settled on Pixar's RenderMan [277, 8, 213, 261, 64]. One of the earliest renderers, it was used to create many landmarks of computer graphics and animation, particularly all of the major films by Pixar, including *Toy Story*, *Toy Story 2*, *Finding Nemo*, *The Incredibles*, *Cars* and, most recently, *Ratatouille*. Moreover, the language itself, the RenderMan Interface Specification, is an open standard that controls not only Pixar's PhotoRealistic Renderman (PRMan) software, but also a number of others (Aqsis, 3Delight). Finally, RenderMan is designed to handle enormous data sets in a highly memory-efficient way, and to render robustly with the fastest performance in a commercial environment—where computation time literally is money—by using an algorithm (called REYES) that breaks up all objects in the scene into polygons that are comparable to the pixel size in the final image. As stated earlier, a major goal from a technical standpoint is using open standards and highly-optimized, commercially available tools, to the greatest degree possible;

the choice of Pixar's RenderMan is certainly in line with that philosophy.

There are two ways to control PRMan, either using the C API [277], or by sending text files, called RIB files, to `PRMan.exe` from the command line [213, 261]. Though the first approach may offer slightly better performance, it requires recompilation, of course, to change any parameters, and is therefore not as commonly used. Instead, with RIB text files, all aspects of the rendering process can be controlled using a simple text editor. The importance of this should not be underestimated: with text input and a well-defined standard for RIB files used by several different programs, the ability to use the same input to create 3D images in the future should remain relatively straightforward. In fact, even nVidia's Gelato supports RIB, within certain limits, so that all of the examples presented here could in principle be rendered in hardware.

The complete text of a RIB file to render a colloidal gel structure is presented here:

```
Display "A15A46_Ambient.tif" "file" "rgba"
Format 700 700 1
Projection "perspective" "fov" 32
PixelSamples 16 16
PixelFilter "gaussian" 3 3
Clipping 0.1 1000
ShadingInterpolation "smooth"

#Set viewing angle and zoom
Translate 0 -27 0
Rotate -24 1 0 0
Translate 2 -30 150

WorldBegin
    Rotate 35 0 1 0

    #Set up lighting
    LightSource "ambientlight" 1
        "intensity" .5

    #Read geometry from external file
    ReadArchive "A15A_0046_data.rib"
    ReadArchive "BoundingBox.rib"
WorldEnd
```

First, basic imaging parameters are set, namely the graphics file to be created, and its dimensions in pixels. Then, the field-of-view for the virtual camera that captures the final scene is set by the `projection` setting. The next four parameters reduce image noise and make smooth surfaces appear smooth, but are essentially standard settings that never change (and are well-described in [213]). The following section of the RIB file sets the viewing angle and zoom of the virtual camera, ending the preliminary setup.

The final section, between `WorldBegin` and `WorldEnd`, defines all of the elements in the 3D scene to be rendered, specifically the lighting and objects. The `rotate` command rotates the objects in the scene. The geometry, discussed in section 8.4.1, is a list of the spheres representing the colloids, read in from an external file to facilitate the automated rendering of, for instance, a 3D stack taken from the same location at different times. But the commands are all the same, and this is simply a matter of convenience. In the above example, the lighting is very simple, a single `"ambientlight"`. Different forms of lighting are discussed and illustrated in sections 8.4.2 to 8.4.5.

But that is it. RenderMan has encapsulated a huge amount of technology and calculations beneath a surprisingly simple surface. In just a couple dozen lines of text (aside from a big list of spheres defining particle positions), the entire image is defined. And simply invoking `PRMan.exe filename.RIB` will create the output file.

8.4.1 Creating the Geometry

Only two kinds of geometry appear in the rendered scenes of colloidal gels and clusters: the wireframe box to delineate the extent of the 3D imaging volume, and the actual particles. The box is a simple set of grey lines, whose coordinates are contained in `BoundingBox.rib`, shown in its entirety:

```
AttributeBegin
    color [0.5 0.5 0.5]
    Curves "linear" [2] "nonperiodic" "P"
        [-28 -28 -28 -28 -28 28] "constantwidth" [0.25]
    Curves "linear" [2] "nonperiodic" "P"
        [-28 -28 -28 28 -28 -28] "constantwidth" [0.25]
    Curves "linear" [2] "nonperiodic" "P"
        [28 28 28 -28 28 28] "constantwidth" [0.25]
    Curves "linear" [2] "nonperiodic" "P"
        [28 28 28 28 -28 28] "constantwidth" [0.25]
    Curves "linear" [2] "nonperiodic" "P"
        [28 28 28 28 28 -28] "constantwidth" [0.25]
    Curves "linear" [2] "nonperiodic" "P"
        [28 -28 28 -28 -28 28] "constantwidth" [0.25]
    Curves "linear" [2] "nonperiodic" "P"
        [28 -28 28 28 -28 -28] "constantwidth" [0.25]
    Curves "linear" [2] "nonperiodic" "P"
        [-28 28 28 -28 -28 28] "constantwidth" [0.25]
    Curves "linear" [2] "nonperiodic" "P"
        [28 28 -28 28 -28 -28] "constantwidth" [0.25]
    Curves "linear" [2] "nonperiodic" "P"
        [28 28 -28 -28 28 -28] "constantwidth" [0.25]
    Curves "linear" [2] "nonperiodic" "P"
        [-28 -28 -28 -28 28 -28] "constantwidth" [0.25]
    Curves "linear" [2] "nonperiodic" "P"
        [-28 28 28 -28 28 -28] "constantwidth" [0.25]
AttributeEnd
```

All geometry within a single `AttributeBegin`-`AttributeEnd` block share the same attributes, so that the color only needs to be set once, at the top of the block, and will apply to all elements within it, without affecting elements outside. That is, in the next

`AttributeBegin`-`AttributeEnd` block, objects take on the usual default properties, until otherwise set.

This makes coding for clusters extremely efficient, as attributes that are set for all particles in a cluster—particularly color, opacity and surface material properties—are simply set once at the beginning of the `AttributeBegin`-`AttributeEnd` block, and then do not have to be repeated, as in the following RIB file fragment from `A15A_0046_data.rib`:

```
AttributeBegin
    Color [0.5 0.5 0.5]
    Opacity [0.05 0.05 0.05]
    Surface "plastic" "color specularcolor" [ 1 1 1 ]
        "float Ka" [ 0.7 ] "float Kd" [ 0.7 ]
        "float Ks" [ 0.900 ] "float roughness" [ 0.1 ]

    TransformBegin
        Translate -20.4739 27.1414 5.6725
        Sphere 0.56 -0.56 0.56 360
    TransformEnd

    TransformBegin
        Translate 21.2052 27.0513 -1.7677
        Sphere 0.56 -0.56 0.56 360
    TransformEnd
    . . .
AttributeEnd
```

Surface material information is conveyed in the form of a *shader*, a piece of programming code that takes light as input (with a certain direction, intensity, etc.) and, as a function of object geometry, returns a color value to the camera [213, 64]. Glossy materials reflect more light, while matte materials reflect less—and the specific values are calculated by code within the shader. In this example, the surface material is the built-in plastic shader, included standard with RenderMan. Color and opacity are set independently of the surface shader, and in this example, all of the spheres are grey colored and very transparent, used for colloidal particles that are either monomers or dimers (they are so abundant that, if rendered opaque, would obscure any other gel structure in the image).

Once these parameters are set, the colloidal particles themselves are drawn as spheres, using the geometric primitive built into RenderMan. Changing the location before drawing each sphere is performed with the `Translate` command. Note that the same scoping structure applies to the `TransformBegin`-`TransformEnd` blocks, in which a single particle is created within each block. First, the coordinates are translated relative to the origin, then a sphere is drawn (which takes on the attributes previously set). Because the translation operation only applies within that block, the default coordinate origin reverts to zero for the next one. In this way, spheres are drawn at the absolute position of the particles relative to a fixed origin, not, for instance, the position relative to the last drawn sphere (as is the case in the OpenGL example at the beginning of the chapter).

This makes the generation of the geometry in RIB format extremely easy with C++: just print out the particle coordinates from the cluster analysis, with the various static text blocks around. And it is straightforward to automate this process to allow rapid generation and production of movies of successive 3D stacks, with very little marginal effort.

8.4.2 Basic Lighting

Once the geometry is set, the final task is to establish the lighting, and this is where the visualization process becomes a bit more non-trivial. The overall goal is to illustrate a disordered 3D gel structure, perhaps surrounded by numerous smaller clusters, but because the structures are highly varied, there is no one particularly straightforward or easy general solution. In this section, then, I will step through the different types of lighting I used, and illustrate with examples on a single colloidal gel structure (a gel with ξ=0.018 and ϕ=0.045, taken approximately 45 minutes after mixing), where clusters have been colored according to their mass, following the color bar in figure 10.3.

Lights are implemented in RenderMan as shaders [213, 64], which control the geometry, intensity, propagation and reflection of light off of surfaces (in some cases, the same effects can be achieved with changing either the lighting or the surface shader, giving additional flexibility when using RenderMan). The simplest type of light shader is ambient light, which just shows the color of the object being rendered, without any reflections or interactions with the object's geometry. Spheres, in this case, appear as flat circles with their color as specified in the geometry file. This is the case of the single line in the lighting section of the complete RIB file shown above:

```
    LightSource "ambientlight" 1
        "intensity" .5
```

(In this and subsequent examples, I show only the fragment of the RIB file corresponding to the light shaders, as everything else remains the same). Ambient lighting is useful for controlling the overall brightness of the objects being displayed, without changing the reflections and specular highlights on their surface [213]. An example of ambient lighting is shown in figure 8.4a, and has been used with previous simple gel representations [42].

Another simple light is *directional*, mimicking a parallel beam of light, of constant, uniform intensity. It illuminates all objects equally (no shadows, or distance-based falloff), with only the angle relative to the objects being an important parameter [213]. In the example shown in figure 8.4b, the default settings are used for the light position, while the intensity is set directly.

```
    LightSource "distantlight" 2
        "intensity" 1.5
```

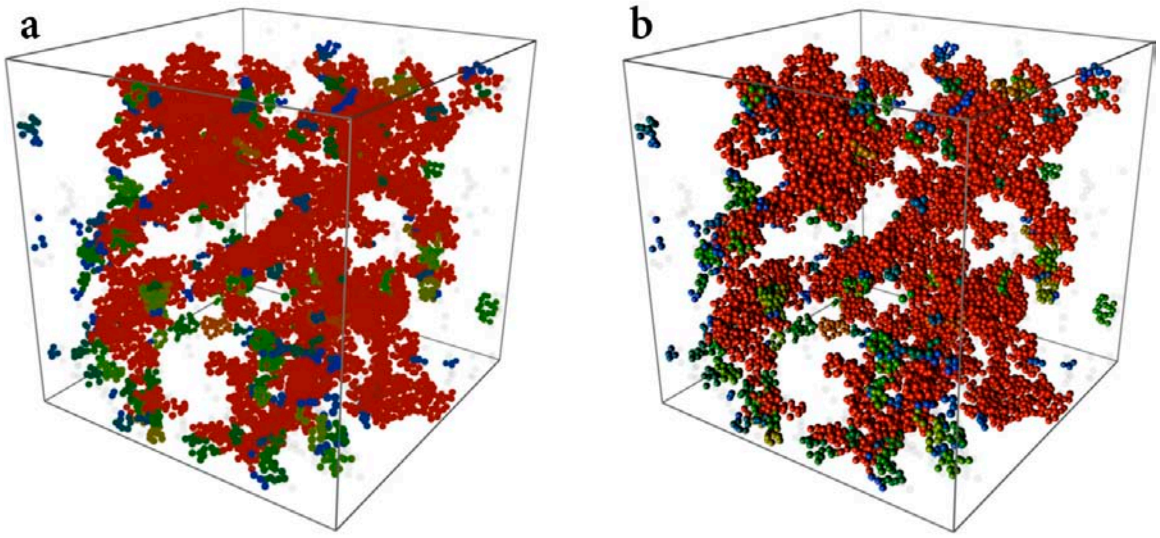

Figure 8.4: **(a)** Colloidal gel structure lit only with *ambient lighting*, which controls the overall brightness of the objects, without any regard to surfaces or reflections. **(b)** The same gel lit only with *distant lighting*, parallel rays of light, which come from a distance of infinity and do not diverge, much like light from the sun.

Early renderings of particle configurations in the last decade have relied primarily on directional lighting [279, 278]. Combining the two yields the image in figure 8.5, described by the following lighting code:

```
LightSource "ambientlight" 1
    "intensity" .5
LightSource "distantlight" 2
    "intensity" 1
```

The structure is rather flat and does not really show much depth. It is difficult to get a sense of which particles are closer to or farther away from the viewer, with only apparent radius for a visual clue. This is basically the best that can be attained with most scientific programs not specifically dedicated to visualization, and many of the example renderings in the colloid literature consequently look similar [233, 234, 235, 102, 296, 94]. Figure 8.5 is roughly comparable to the maximum achievable quality that can be achieved with MATLAB.

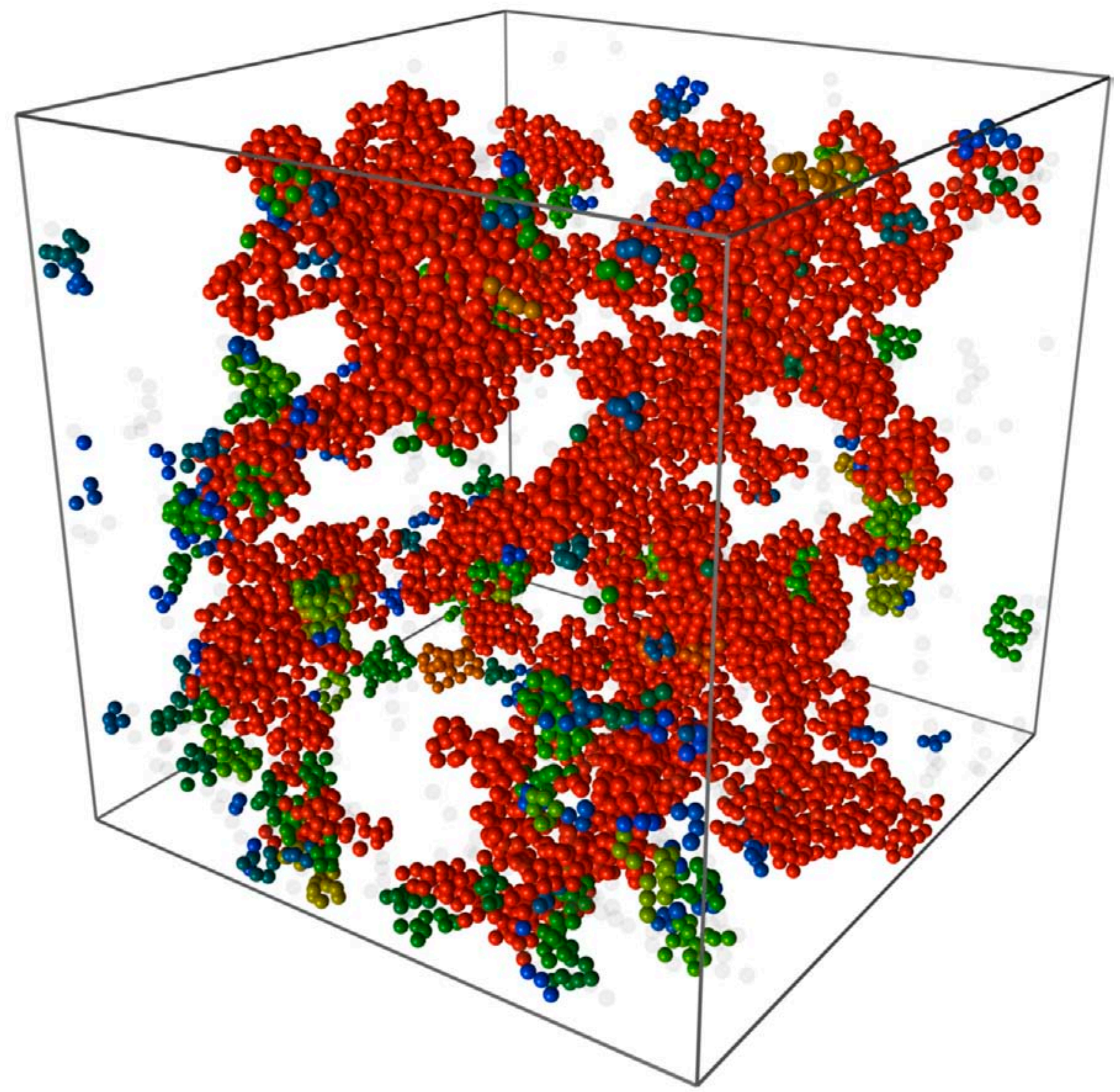

Figure 8.5: Colloidal gel lit with both ambient and distant lighting, without shadows. This comparable to output from MATLAB, and similar to many published 3D colloid structures. Note that the absence of shadows makes it difficult to visualize the 3D shape and structure of the gel at lengthscales greater than the particle size. It is not immediately evident, for instance, which parts of the chains are in front or in back.

8.4.3 Cast Shadows with ShadowMaps

Light sources in the real world cast shadows, and interplay between objects and their shadows provides a great deal of information to the human brain in establishing spatial relationships and visualizing different forms. This is particularly evident for natural structures such as mountains—where dramatic photographs owe their impact as much to shadows as to the lit part (e.g. Ansel Adams)—and disordered structures, where the brain is not aided by a simple repeating motif, for instance, in reconstructing a pattern.

There are several ways to add shadows to 3D renderings. One of the simplest is to use a shadowmap: place a virtual camera in the position of a spotlight, and then illuminate only what this particular camera “sees.” That is, light rays coming from that light source only illuminate the first object surface that they hit; all others (behind that object relative to the light) are not illuminated by that light, though they may receive light from other sources (e.g. ambient). In RenderMan, this involves positioning a camera at the light that casts shadows, and making a map of the distances to every surface that is visible to that camera; the resulting image is termed a shadowmap, and is loaded into the light shader when the scene is rendered with the regular camera. The code below uses the standard spotlight with shadowmaps included with RenderMan:

```
LightSource "ambientlight" 1
    "intensity" .2
LightSource "distantlight" 2
    "intensity" .4
LightSource "shadowspot" 3
    "intensity" 4000
    "coneangle" 5
    "from" [20 50 -30]
    "to" [0 0 0]
    "shadowname" ["A15A46_shadowmap1.tx"]
    "samples" 16.0
```

```
        "width" 3.0
```

The structure illuminated by an overhead spotlight set above and slightly to the left of the structure, using a shadowmap to determine cast shadows, is shown in figure 8.6. The intensity of the light coming from the spotlight falls off with the square of the distance from the light itself—as in real life—so the particles at the top (closest to the light) are brighter than those at the bottom. The overall effect is a dramatic improvement over the example shown in figure 8.5: the large-scale 3D structure is far more apparent, and which chains are in front and back can be seen far more clearly. This type of illumination is very similar to what I created using Maya, for instance the structure illustrated in figures 8.2 and 8.3. But while the large-scale structure has been improved, the lighting on the individual particle scale is still a bit phony-looking. In particular, the particles that are in the shadows are all relatively uniform and flat, and the surface of the gel is not clearly shown; the shadowmap technique creates harsh shadows which are not illuminated at all, which doesn't really capture how light interacts in the real world.

8.4.4 Ambient Occlusion

To mimic more realistic behavior of light, more complex lighting and shadow calculations are needed to simulate better how light rays behave. Conceptually, this is straightforward: begin with each light, and send out a collection of rays. Calculate which objects they intersect, and then determine which new light rays are generated as reflections. These are then propagated until they hit surfaces, and the process repeats. Called *ray-tracing*, the technique is ultimately the most realistic, since it in effect runs a simulation of all light rays in the scene. Unfortunately, it is horrendously expensive computationally (and is used

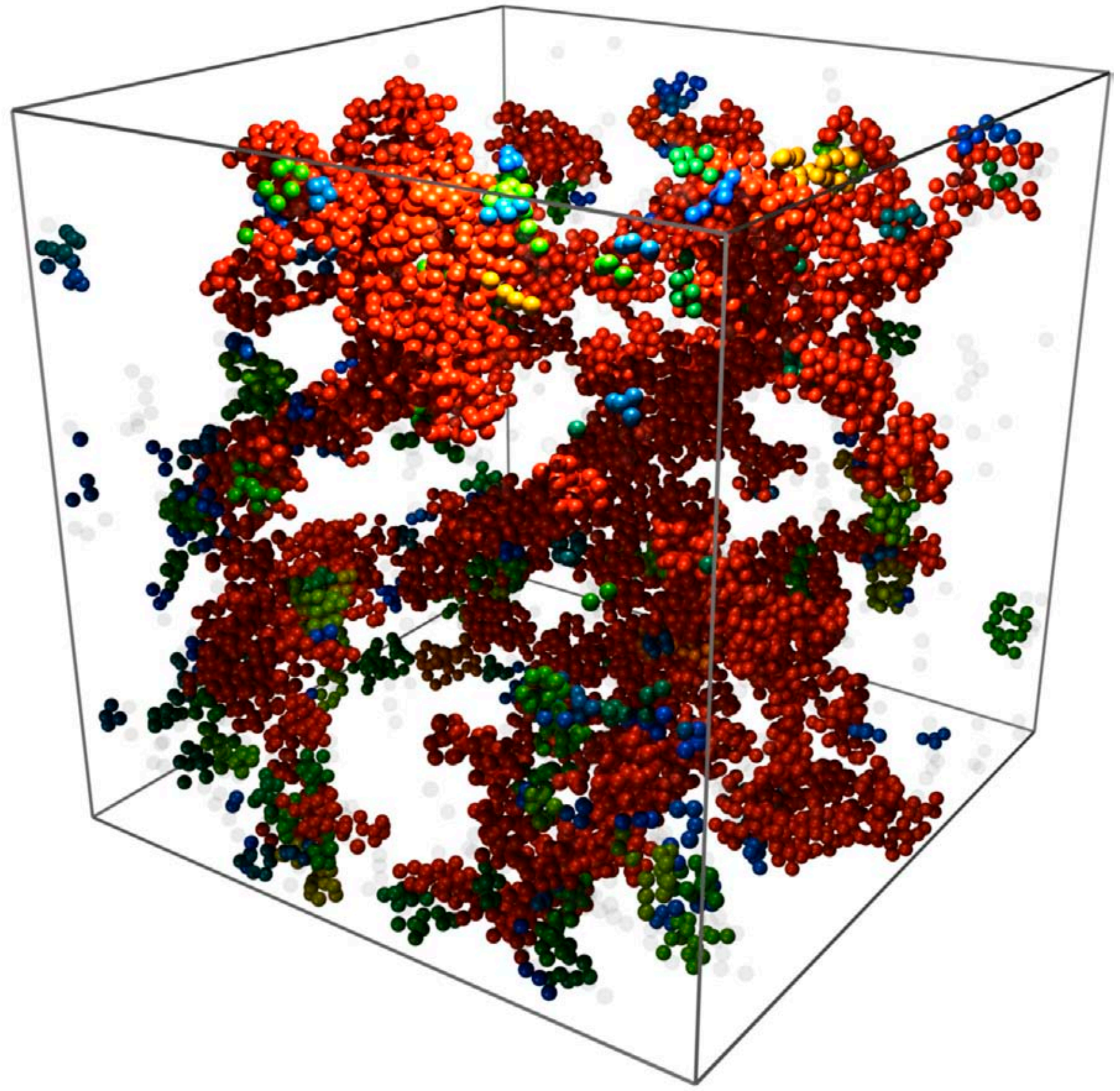

Figure 8.6: Gel illuminated with ambient and distant lighting, as before, with an additional spotlight that casts shadows using a shadowmap. The overall 3D structure of the gel is far more apparent, and which chains are in front and which are behind is far more clear.

exclusively by POV-Ray, explaining its sluggish performance). However, full ray-tracing is not always necessary or even desirable; calculating all of light rays to hit small, dim objects in the background does not make any appreciable difference to the final image, but adds a lot of computation time. So various computational tricks have been employed to mimic these *global illumination* effects, without the full-blown ray-tracing calculation.

The one most relevant to lighting around each particle is called *ambient occlusion* [64]. This is a geometric calculation that determines which light rays, reflected off of an object's surface, run into other surfaces and therefore don't make it back to the camera. If two spheres are sitting next to each other, for instance, the region around the point where they touch will naturally look darker. This calculation depends on the geometry and arrangement of objects, independent of lighting (this region between the touching spheres will generally be darker no matter where the light source is placed), and so does not require full ray-tracing, while still achieving many of its effects.

Ambient occlusion has been added to recent versions of RenderMan in the form of a light shader (which I only slightly modified from the default):

```
light plu_occlusionlight1(
    float samples = 64,
    maxvariation = 0.02,
    maxdist = 1e30,
    coneangle = PI/2;
    color filter = color(1);
    output float __nonspecular = 1;
    )
{
    vector  i = normalize(I);
    normal  n = normalize(N),
    Ns = faceforward(n, i);
    illuminate (Ps + Ns)
    {
        float occ = occlusion(Ps, Ns, samples,
            "maxvariation", maxvariation, "maxdist",
            maxdist, "coneangle", coneangle);
    Cl = filter * (1 - occ);
    }
}
```

Adding this to the scene requires just a couple of new lines in the RIB file.

```
    LightSource "ambientlight" 1
        "intensity" .2
    LightSource "plu_occlusionlight1" 4
        "samples" 36
        "maxvariation" 0.1
        Attribute "visibility" "int diffuse" 1
        Attribute "visibility" "int specular" 1
        Attribute "trace" "bias" 0.005
```

The gel, rendered only with ambient occlusion (and an ambient shader which only alters the overall brightness), is shown in figure 8.7. The shadows around each particle are far more realistic, and even the overall 3D structure is quite clear—in the absence of any other directional lighting or a spotlight. Moreover, adding this involved a very simple addition to the RIB file, and practically no other work. (Note that Maya's Mental Ray renderer also supports global illumination, ambient occlusion, and ray-tracing, but is less easy to automate). Ambient occlusion and directional lighting, without shadows, were used to render the PLuTARC clusters in figure 4.3.

Ambient occlusion can be combined with shadowmaps for greater realism in shadows. Adding the ambient occlusion light shader to the example gel lit with the spotlight using shadowmaps in figure 8.6 yields the image shown in figure 8.8, and the following RIB file fragment:

```
    LightSource "ambientlight" 1
        "intensity" .2
    LightSource "distantlight" 2
```

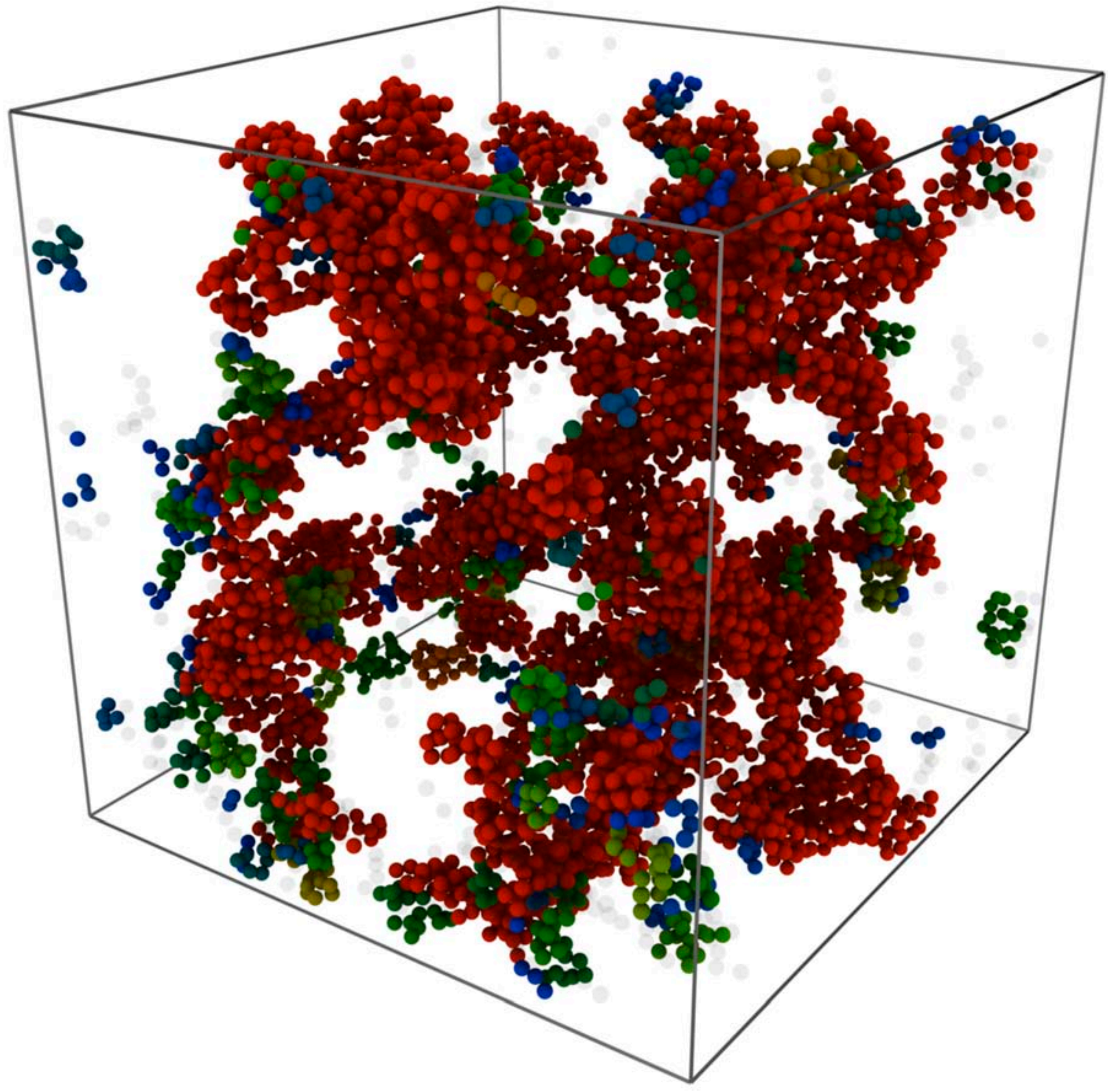

Figure 8.7: Colloidal gel lit with ambient occlusion, with some ambient lighting to adjust overall brightness. Note that the 3D structure can clearly be visualized, without any additional directional lighting or spotlights, or cast shadows.

```
        "intensity" .4
    LightSource "shadowspot" 3
        "intensity" 4000
        "coneangle" 5
        "from" [20 50 -30]
        "to" [0 0 0]
        "shadowname" ["A15A46_shadowmap1.tx"]
        "samples" 16.0
        "width" 3.0
    LightSource "plu_occlusionlight1" 4
        "samples" 36
        "maxvariation" 0.1
        Attribute "visibility" "int diffuse" 1
        Attribute "visibility" "int specular" 1
        Attribute "trace" "bias" 0.005
```

This particular combination of ambient occlusion and shadowmaps was used to create the renderings in figures 10.3 and 10.4.

8.4.5 Ray-Traced Shadows

Even greater realism can be achieved by performing the full-blown ray-tracing calculation to determine the reflections and shadows of the structure as a whole. One of RenderMan's great strengths is the ability to add ray-tracing selectively, while leaving the far more efficient REYES algorithm to perform the remainder of the work. This hybrid approach adopts the best of both worlds, maximizing quality and performance in a way not possible with rendering programs that exclusively follow one single approach. As with the other lighting techniques, RenderMan implements ray-tracing in a light shader, included with recent versions of the program:

```
light spotlight_rts(
    float intensity = 1;
    float falloff = 2.0;
    color lightcolor = 1;
```

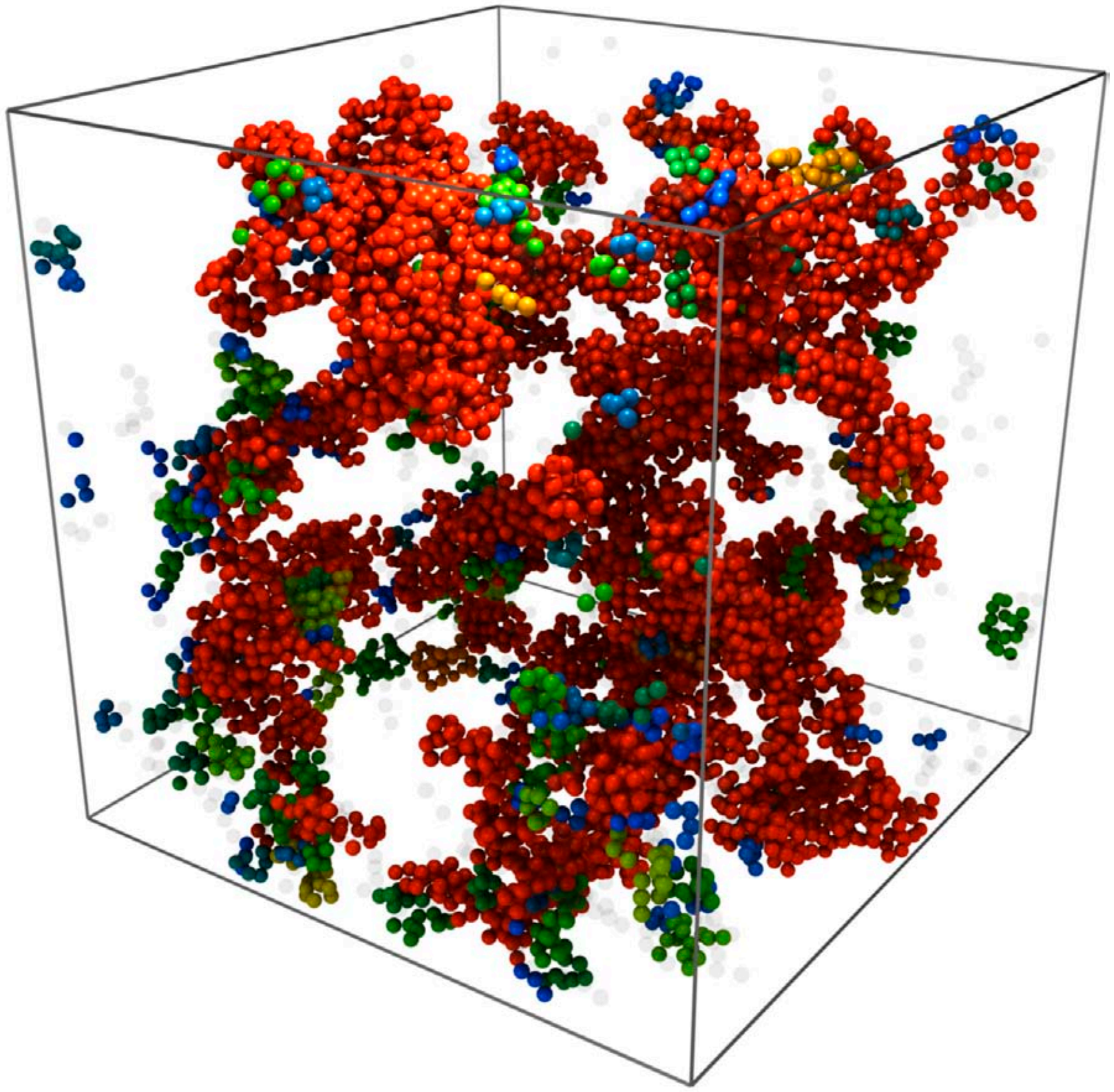

Figure 8.8: Colloidal gel lit with ambient occlusion, and the same combination of ambient, distant and spot lighting with shadowmapped cast shadows, as in figure 8.6.

```
    point from = point "shader" (0,0,0);
    point to = point "shader" (0,0,1);
    float coneangle=radians(30);
    float conedeltaangle=radians(5);
    float samples=1;
    float blur=0;
    float bias=0;
    )
{
    uniform vector A=(to-from)/length(to-from);
    uniform float cosoutside=cos(coneangle);
    uniform float cosinside=cos(coneangle-conedeltaangle);
    float atten, cosangle;
    float strength, ldist;
    color Ct;
    illuminate(from,A,coneangle)
    {
        cosangle = L.A / length(L);
        atten = smoothstep(cosoutside,cosinside,cosangle);
        ldist=length(L);
        strength = pow((1+ldist),-falloff);
        Cl = strength*atten*intensity*lightcolor;
        Ct = transmission(Ps, from, "samples", samples,
            "samplecone", blur);
        Cl *= Ct;
    }
}
```

Adding ray-tracing to the scene again involves only a simple addition to the RIB file, as has been the case in all other examples in this chapter. In the following RIB fragment, a ray-tracing spotlight replaces the previous spotlight using shadowmaps, in the same position.

```
    LightSource "plu_occlusionlight1" 4
        "samples" 36
        "maxvariation" 0.1
        Attribute "visibility" "int diffuse" 1
        Attribute "visibility" "int specular" 1
        Attribute "trace" "bias" 0.005
    LightSource "spotlight_rts" 5
        "color lightcolor" [ 1 1 1 ]
        "float bias" [ -1 ]
        "float blur" [ 0 ]
        "float coneangle" [ 2 ]
        "float conedeltaangle" [ 1 ]
        "float falloff" [ 2.000 ]
        "float intensity" [2000.]
        "float samples" [ 8.000 ]
        "point from" [20 50 -30]
        "point to" [ 0 0 0 ]
```

The resulting image, created with just two light sources, ambient occlusion and a single spotlight with ray-traced shadows, is shown in figure 8.9. Note the more realistic look of the image, with software shadows subtly reflecting the gel's rough surface, particularly in the darker areas toward the back.

A different view of the gel shown in figure 6.13, illuminated by two spotlights with ray-traced shadows, and using ambient occlusion, is shown in figure 8.10. Its light shaders are shown in the following RIB fragment:

```
    LightSource "plu_occlusionlight1" 3
        "samples" 36
        "maxvariation" 0.1
        Attribute "visibility" "int diffuse" 1
        Attribute "visibility" "int specular" 1
        Attribute "trace" "bias" 0.005
    LightSource "spotlight_rts" 4
        "color lightcolor" [ 1 1 1 ]
        "float bias" [ -1 ]
        "float blur" [ 0 ]
        "float coneangle" [ 1 ]
        "float conedeltaangle" [ 1 ]
        "float falloff" [ 2.000 ]
        "float intensity" [1800.]
        "float samples" [ 8.000 ]
        "point from" [ -10 50 30 ]
        "point to" [ 0 0 0 ]
    LightSource "spotlight_rts" 5
        "color lightcolor" [ 1 1 1 ]
```

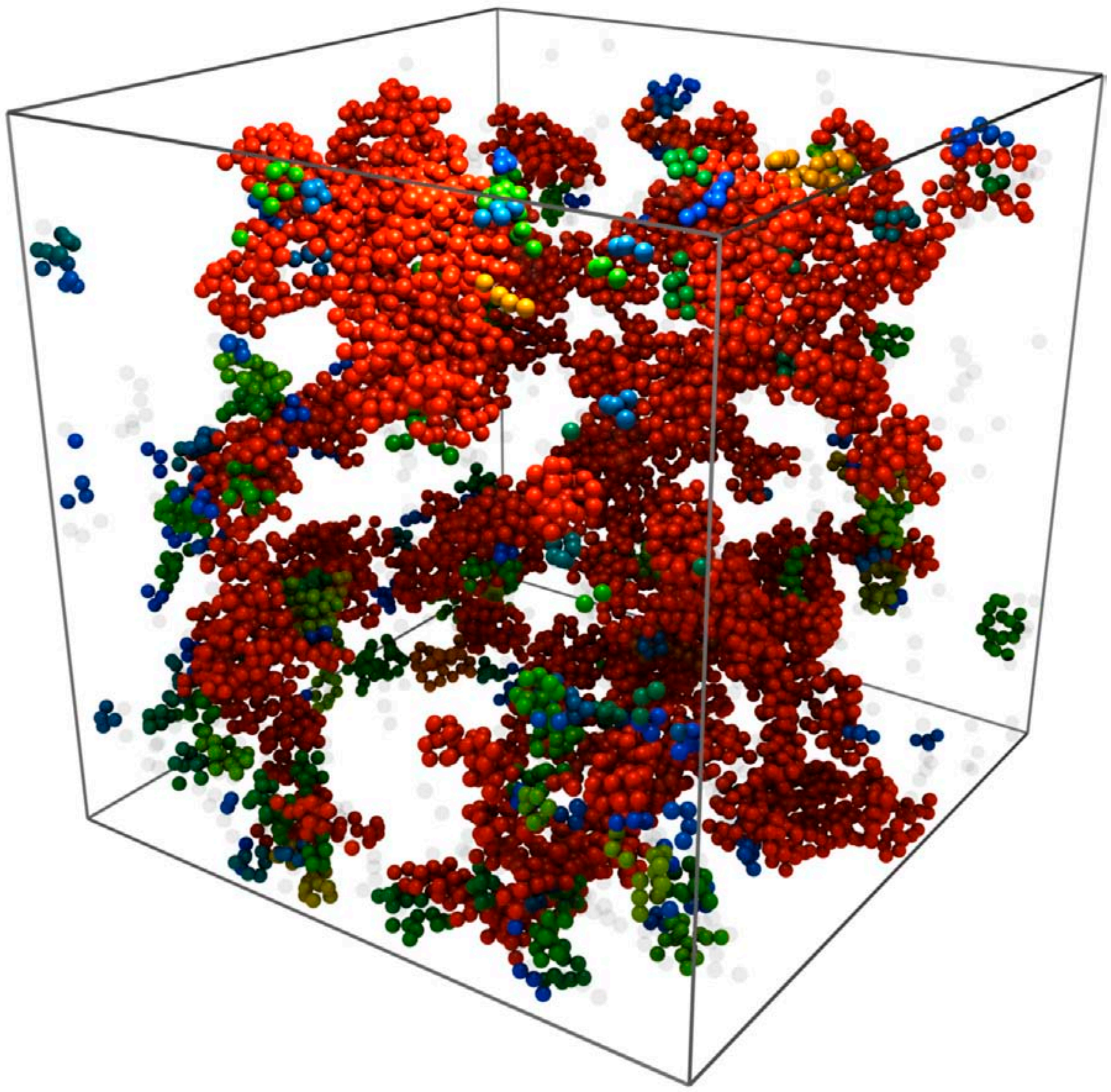

Figure 8.9: Final image of a colloidal gel illuminated by a spotlight with ray-traced shadows, and ambient occlusion. Notice the more realistic lighting and subtleties of the particles toward the back in the darker regions. The 3D structure is clearly conveyed, as is the local geometry of all particles.

```
        "float bias" [ -1 ]
        "float blur" [ 0 ]
        "float coneangle" [ 1 ]
        "float conedeltaangle" [ 1 ]
        "float falloff" [ 2.000 ]
        "float intensity" [3600.]
        "float samples" [ 8.000 ]
        "point from" [ -20 60 -10 ]
        "point to" [ 0 0 0 ]
```

Obviously, this is just scratching the surface of the possibilities with RenderMan; I have not, for instance, discussed writing shaders. Rather, the examples presented are meant to demonstrate how complex scientific visualizations can be created with a surprisingly small amount of actual coding.

8.5 Acknowledgments

Hacking the Maya particle data cache was done in collaboration with Amit Chourasia at SDSC, who also introduced me to MEL. I got started with RenderMan at with the SIGGRAPH 2006 day-long workshop, “RenderMan for Everyone,” held in Boston. There I was privileged to learn from some of the contemporary masters, especially Dan Maas and Saty Raghavachary, who have guided me in learning everything that I know. I also grateful to Ram Sampath for additional helpful discussions, and thank Eli Sloutskin, Tom Kodger, and Dylan Sisson for comments on the chapter.

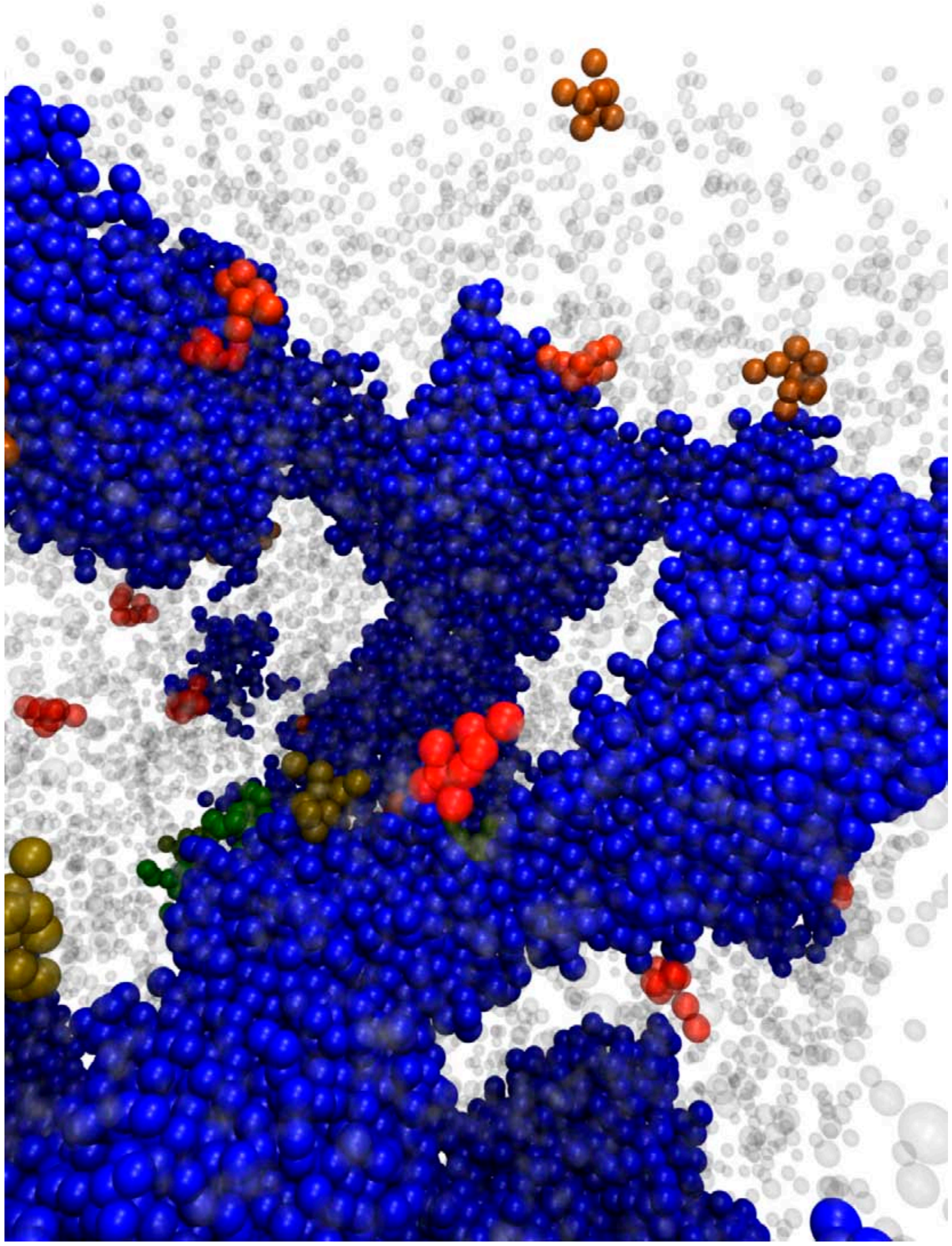

Figure 8.10: Portion of a colloidal gel at higher volume fraction, and using a different color scheme than the previous examples, coexisting with a gas of monomers rendered in transparent grey. The gel is lit using ambient occlusion and two spotlights with ray-traced shadows.

Part III

Science

Chapter 9

Fluids of Clusters in Attractive Colloids

9.1 Abstract

We show that colloidal particles with attractive interactions induced by nonadsorbing polymer exhibit a stable phase consisting of a fluid of clusters of particles. This phase persists even in the absence of any long-range repulsion due to charge, contrary to expectations based on simulation and theory. Cluster morphology depends strongly on the range of the interparticle attraction: With a shorter range, clusters are tenuous and branched; with a longer range, they are more compact.

9.2 Introduction

Mixing non-adsorbing polymer with a colloidal suspension can induce an effective attraction interaction between the particles, leading to very rich phase behavior [154, 136]. This system also can serve as an excellent model of attractive suspensions of complex fluids encountered in technological applications such as food and personal care products, where the attractive interaction may significantly impact product stability and shelf life. Colloid-polymer mixtures are convenient for these studies: Both the range and magnitude of the attractive interaction can be precisely controlled, and the constituent colloids are large enough to be imaged with light, while still small enough to have their dynamics driven by $k_\mathrm{B}T$, the thermal energy. With sufficient polymer, these mixtures form non-equilibrium gels, networks of particles that percolate across the sample to form a solid [249]. Though the exact mechanism is not well understood, aggregation of particles into clusters is a precursor to gelation. Cluster formation may be driven by phase separation, where the system lowers its free energy by splitting into two phases, a colloid-poor gas and colloid-rich liquid [244]; alternatively, clusters may grow when particles stick irreversibly upon approach in the kinetic process of diffusion-limited cluster aggregation (DLCA) [179]. Clusters are expected to be transient in both cases: In phase separation, cluster growth is thermodynamically favored and either proceeds to completion, where all clusters have merged into one liquid droplet, or kinetically arrests to form a gel [244]; in DLCA, clusters grow and merge until forming a connected cluster that spans the system [179]. In neither case should multiple freely diffusing clusters of particles persist in steady state. Nonetheless, experiments have found stable diffusing clusters in regions of phase space near the gel transition line [249]. This contradicts the expectation that colloid-polymer mixtures should either phase separate or gel. One resolution to this dilemma is that the colloidal particles are charged [113]. After clusters grow to a certain size, they accumulate enough charge to repel additional particles; this long-range repulsion introduces a second length scale, the characteristic maximum cluster size [244]. Indeed, theories for stable clusters generally require long-range repulsion [114], which has been observed experimentally to have significant effects in colloid-polymer mixtures [303, 263, 42]. These results apparently resolve how an equilibrium fluid of clusters can form: the competition between short-range attraction and long-range Coulombic repulsion.

In this chapter we report the observation of fluids of colloidal clusters in which long-range Coulombic repulsion does not play a role. These buoyancy-matched clusters are stable, diffuse freely, and neither phase separate nor form a percolated gel on experimental timescales. Their morphology depends on the range of interparticle attraction: With a longer range, the clusters are compact; with a shorter range, they are more branched and fractal-like.

9.3 Experimental Methods

9.3.1 Sample Preparation

We suspend sterically-stabilized colloidal spheres of polymethylmethacrylate (PMMA) in a solvent mixture of decahydronaphthalene (DHN) and either bromocyclohexane (CXB) or bromocycloheptane (CHB). The solvent matches PMMA's index of refraction, enabling the use of light scattering and confocal microscopy; it also matches PMMA's density, preventing gravitational sedimentation that shears particle aggregates. The solvent has dielectric constant $\epsilon = 7$, and Bjerrum length $\lambda_{\mathrm{B}} = 8$ nm; λ_{B} is the distance in a dielectric medium at which the electrostatic interaction energy between two monovalent charges is equal to $k_{\mathrm{B}}T$. To screen any long-range electrostatic repulsion that can arise between colloids in these solvents [303], we add 4 mM of the organic salt tetrabutyl ammonium chloride (TBAC). We measure the solution's conductivity to be 20 μS/m, and, assuming TBAC's hydrodynamic radius is 1 nm, we estimate a Debye screening length of 12 nm [129]. Thus, any long-range Coulombic repulsion is completely screened.

We add non-adsorbing linear polystyrene (PS) to induce a depletion attraction between PMMA spheres, and vary the molecular weight, M_w (amu), and free-volume concentration, c_p (mg PS per mL of solvent, corrected for colloid volume fraction ϕ [154]). Our mixture is a good solvent for PS, so increasing the number of PS coils in a fixed volume decreases their size; we therefore use static light scattering to measure the concentration-dependent polymer radius r_p, and report the size ratio $\xi = r_\mathrm{p}/a$, where a is the colloid radius. Our interparticle potential has two parts, a hard-core repulsion for $r < 2a$, and an attraction for $2a < r < 2(1+\xi)a$, which we calculate using the Asakura-Oosawa model: When two colloids touch, the strength of their attraction is $U = PV$, where V is the volume excluded to polymers between the spheres, a geometric factor depending only on a and ξ [136], and U is measured in units of $k_\mathrm{B}T$. The polymer osmotic pressure P is determined by integrating the polymer osmotic compressibility measured with light scattering [200]. Because charge has been screened, the interparticle potential is dominated by the hard-sphere repulsion and the depletion attraction; any electrostatic repulsion is restricted to very short range.

9.3.2 Light Scattering

We explore the bulk phase behavior using dynamic light scattering, where the macroscopic sample volume provides good statistics for a large number of particles. We use particles with $a = 136$ nm at $\phi = 0.15$ and PS with $M_\mathrm{w} = 2.0 \times 10^6$, yielding $\xi \sim 0.15$.

We collect the light scattered from a laser operating at a wavelength of $\lambda = 514.5$ nm *in vacuo*, and we measure the intensity autocorrelation function to determine the dynamic structure factor $f(q,t)$. We normalize the delay time as $t\eta_0/\eta$, where η_0 and η are the viscosities of background solvent and polymer solution, respectively. The normalized scattering vector $qa = 3.52$ is chosen at the peak in the static structure factor corresponding to the nearest-neighbor separation, though behavior is similar at nearby scattering vectors. For the sample with no polymer, where $U = 0$, $f(q,t)$ follows the exponential decay of a suspension of colloids (solid line, figure 9.1), and our calculated diffusion coefficient agrees with the value expected at $\phi = 0.15$ when hydrodynamic interactions are included [20]. At increased polymer concentration, the decay time increases, and $f(q,t)$ becomes somewhat nonexponential (dotted line, figure 9.1), consistent with a broader distribution of timescales and, hence, a broader size distribution of diffusing species. Upon addition of yet more polymer, $f(q,t)$ decorrelates even more slowly, though still fully (dashed line, figure 9.1). That particle motion is slowed suggests they are aggregated; that particle motion is not fully arrested suggests they are not gelled.

9.3.3 Confocal Microscopy

To elucidate the structure of these colloid-polymer mixtures, we use confocal microscopy to image similar samples at the single-particle level. The particles used for light scattering are too small to image; we instead use larger spheres with $a = 574$ nm and labeled with fluorescent Nile Red dye in CXB/DHN. Our spinning Nipkow disk confocal system operates with a laser at $\lambda = 532$ nm and collects images fast enough to capture the three-dimensional structure before the colloids diffuse significantly. A major challenge is

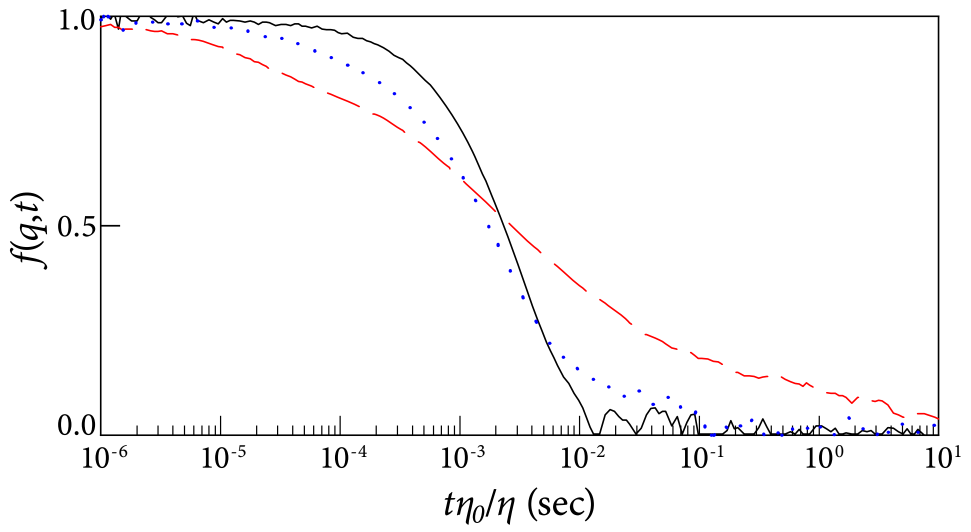

Figure 9.1: Dynamic structure factor $f(q,t)$ at $qa = 3.52$ versus viscosity-corrected delay time $t\eta_0/\eta$ for samples at $\phi = 0.15$. Solid black line: $c_\mathrm{p} = 0$, $U = 0$; dotted blue line: $c_\mathrm{p} = 6.23$, $\xi = 0.17$, $U = 2.3$; dashed red line: $c_\mathrm{p} = 7.62$, $\xi = 0.15$, $U = 2.6$.

keeping larger particles neutrally buoyant: A sphere's Stokes velocity scales as a^2, so the colloids used in imaging may settle an order of magnitude faster than those used in light scattering. We add CXB or DHN dropwise to ensure that particles remain in suspension after centrifugation at $100g$ for 12 hours. Adding a $\sim$10 mg drop of CXB to a balanced $\sim$1 g sample causes the colloid to sediment after centrifugation, implying that the PMMA and solvent densities are matched to $\Delta\rho/\rho \approx 1.1 \times 10^{-3}$. Because the volume thermal expansion of organic liquids ($\sim 10^{-3}$/°C [160]) is much greater than that of solid PMMA (2.5×10^{-4}/°C [36]), a given solvent ratio will match the colloid density only over a narrow temperature range. Thus, we further improve our buoyancy match by keeping both centrifuge and microscope at 27 ± 0.2°C, limiting density fluctuations from thermal expansion to $\Delta\rho/\rho \approx 1.5 \times 10^{-4}$. We can thus keep single particles and small clusters of less than $\sim$100 particles from sedimenting for weeks, while larger clusters with thousands of particles do not sediment for several days.

9.4 Results

9.4.1 Stable fluids of clusters

We explore structure and behavior varying ϕ and ξ. In samples at $\phi = 0.15$ with a "long" polymer ($\xi \sim 0.15$, $M_\mathrm{w} = 11.4 \times 10^6$), corresponding to ϕ and ξ of the samples used in light scattering, we find three distinct phases. At the lowest polymer concentration, we observe a homogeneous fluid of single particles (figure 9.2) with no large clusters. At the highest polymer concentration, we observe a connected gel (figure 9.4) where all particles are part of a single space-spanning cluster. At an intermediate polymer concentration, we observe something qualitatively different: Interspersed throughout a fluid of predominantly single particles are compact clusters with thousands of particles each (figure 9.3). These clusters neither form a network structure with internal holes nor percolate, remaining geometrically distinct from a gel at all times. Confocal microscopy thus confirms that a stable phase of a fluid of clusters exists, even in the absence of a long-range repulsion due to electrostatic interactions.

9.4.2 Cluster morphology

Cluster morphology depends strongly on ξ but not on ϕ. In a long-polymer sample at a lower $\phi = 0.04$, we observe compact clusters with thousands of colloidal particles (figure 9.5a), similar morphology to that at $\phi = 0.15$ (figure 9.3). By contrast, while we also find the same fluid, gel and cluster phases at shorter ξ, clusters in samples with

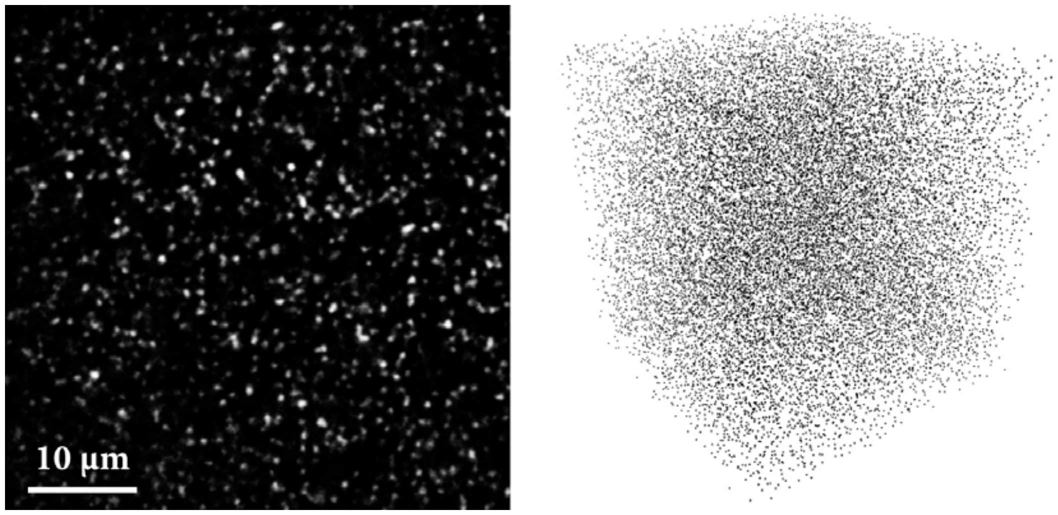

Figure 9.2: (left) 2D confocal microscope image (60 μm)^2 and (right) 3D reconstruction (60 μm)^3 of a colloid-polymer sample with $\phi = 0.15$ and long-polymer depletant, which is an homogeneous fluid of primarily single particles ($c_\mathrm{p} = 0.34$, $\xi = 0.17$, $U = 0.9$).

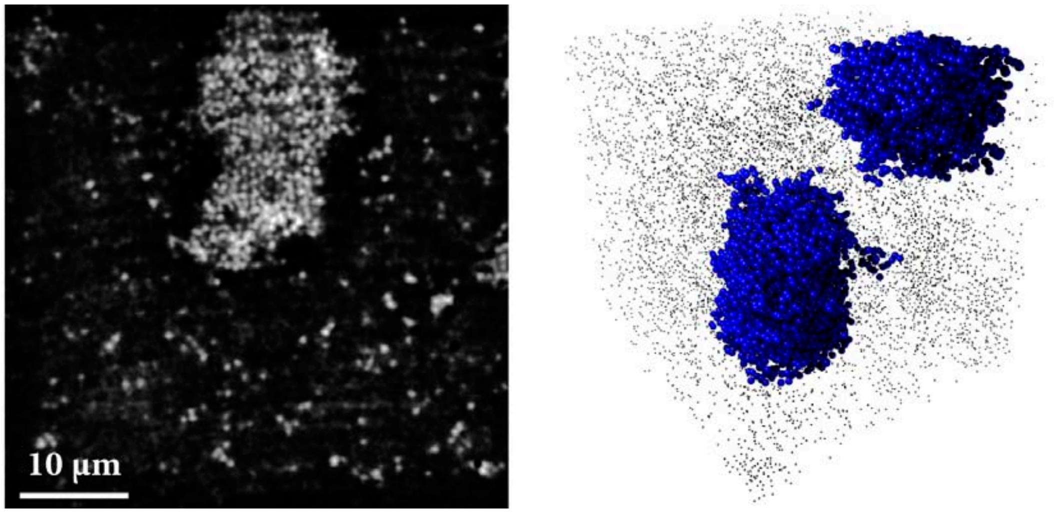

Figure 9.3: (left) 2D confocal microscope image (60 μm)^2 and (right) 3D reconstruction (60 μm)^3 of a colloid-polymer sample with $\phi = 0.15$ and long-polymer depletant, which is a fluid of clusters ($c_\mathrm{p} = 0.67$, $\xi = 0.15$, $U = 1.6$).

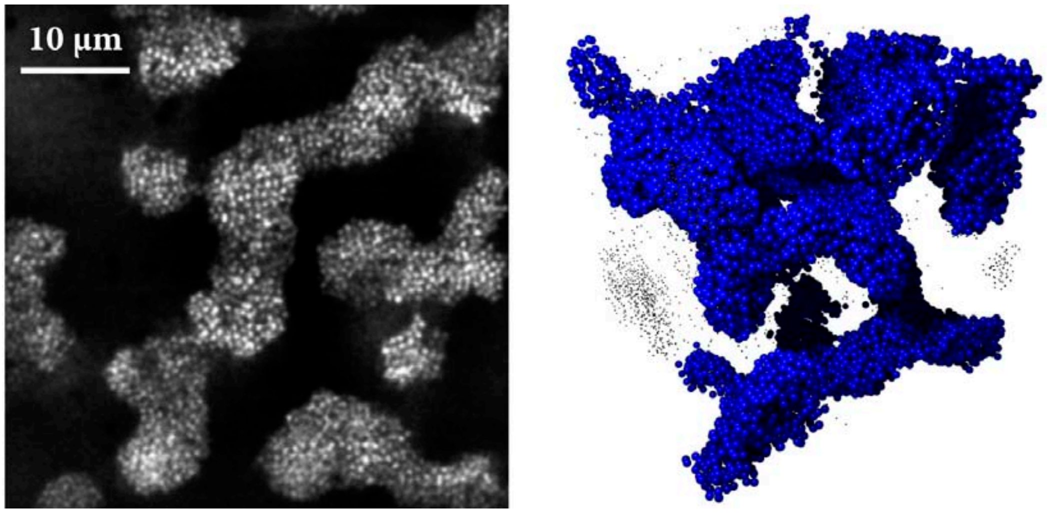

Figure 9.4: (left) 2D confocal microscope image (60 μm)^2 and (right) 3D reconstruction (60 μm)^3 of a colloid-polymer sample with $\phi = 0.15$ and long-polymer depletant, which is a spanning (percolated) gel ($c_\text{p} = 1.31$, $\xi = 0.11$, $U = 2.7$).

smaller polymers have dramatically different morphologies. In samples at $\phi = 0.04$ with a "short" ($\xi \sim 0.02$, $M_\text{w} = 6.67 \times 10^5$) polymer, the far smaller clusters are more tenuous and branched (figure 9.5c). By comparison, in samples at $\phi = 0.04$ with a "medium" ($\xi \sim 0.04$, $M_\text{w} = 3.07 \times 10^6$) polymer, the clusters have a size and degree of branching intermediate between the short- and long-polymer samples (figure 9.5b).

9.4.3 Cluster structural analysis

To quantify the differences in how these clusters fill space, so apparent in figure 9.5, we determine their fractal dimension d_f. We examine how the number of particles in a cluster N scales with radius of gyration R_g for all clusters in a sample; we find a power-law dependence, $N = R_\text{g}^{d_\text{f}}$. We also roughly estimate d_f within individual large clusters in each sample, by counting the number of particles enclosed within spheres of increasing radius centered at the cluster center-of-mass. Both methods yield the same estimate for

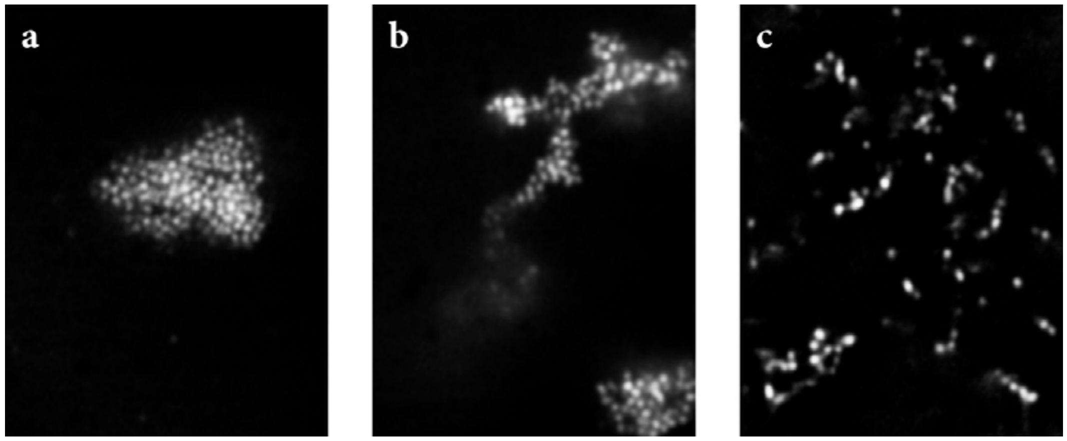

Figure 9.5: Two-dimensional confocal images of clusters at $\phi = 0.04$, with different polymer sizes. **(a)** Long polymer ($c_\mathrm{p} = 1.24$, $\xi = 0.11$, $U = 2.6$). **(b)** Medium polymer ($c_\mathrm{p} = 4.92$, $\xi = 0.04$, $U = 10$). **(c)** Short polymer ($c_\mathrm{p} = 9.83$, $\xi = 0.02$, $U = 12$).

d_f. For the compact clusters in long-polymer samples, $d_\mathrm{f} = 2.4 - 2.6$ for both $\phi = 0.04$ and $\phi = 0.15$, and the largest clusters have at most thousands of particles (figure 9.6a). By contrast, all short-polymer clusters have fewer than a hundred particles; their $d_\mathrm{f} = 1.7 - 1.8$ (figure 9.6b), similar to clusters formed by DLCA. Medium-polymer clusters have an intermediate $d_\mathrm{f} = 2.0 - 2.1$. Thus, clusters with longer attraction range are more compact and have a higher d_f.

To quantify the differences in density at short length scales apparent in the images, we count the number of nearest neighbors for cluster particles. We define two particles as nearest neighbors if the distance between their centers is smaller than the minimum after the first peak in the radial distribution function $g(r)$ [42]. The nearest-neighbor distribution for compact long-polymer clusters (figure 9.7) is peaked around $10 - 12$ for both $\phi = 0.04$ and $\phi = 0.15$, approaching the value expected for close-packed spheres. By contrast, the nearest-neighbor distribution for short-polymer clusters is peaked at $3 - 5$, consistent with their chain-like appearance, while that for medium-polymer clusters is centered at $6 - 7$.

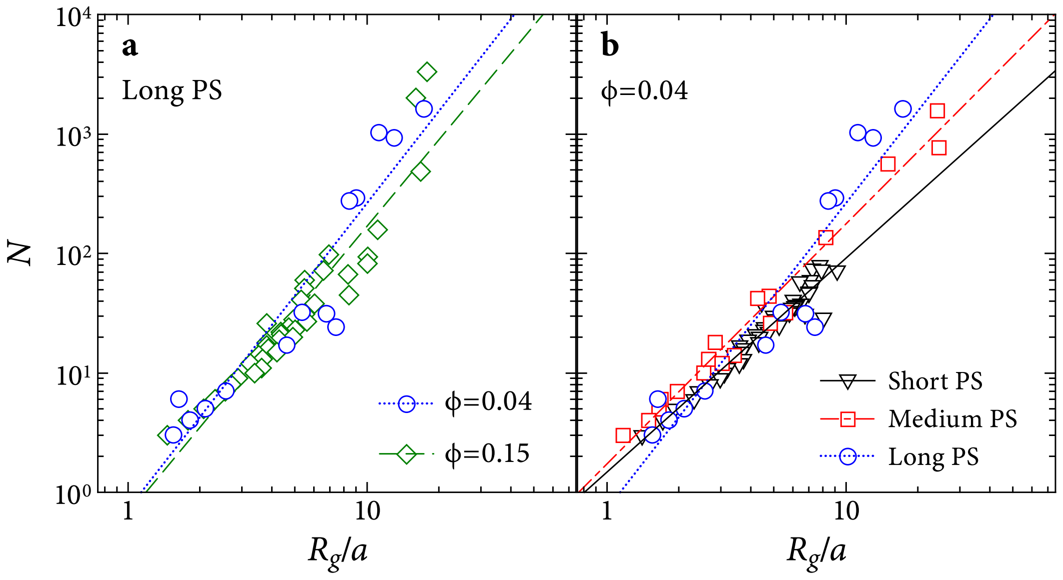

Figure 9.6: Number of particles in a cluster N as a function of cluster radius of gyration R_g for all clusters. Long-polymer clusters at $\phi = 0.04$ (blue circles, best-fit dotted blue line; cf. figure 9.5a) scale **(a)** similarly to long-polymer clusters at $\phi = 0.15$ (green diamonds, best-fit dashed green line; cf. figure 9.3), but **(b)** differently from medium-polymer clusters (red squares, best-fit dotted-dashed red line; cf. figure 9.5b) and short-polymer clusters (black triangles, best-fit solid black line; cf. figure 9.5c) at $\phi = 0.04$.

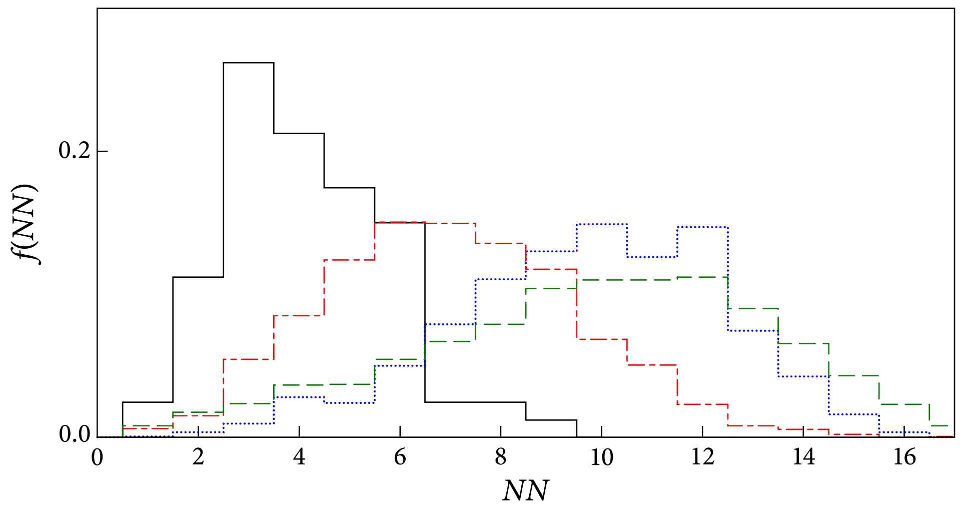

Figure 9.7: Fraction of particles $f(NN)$ with NN nearest neighbors in characteristic clusters. Lines are colored as in figure 9.6: $\phi = 0.04$, short polymer (solid black line), $\phi = 0.04$, medium polymer (dotted-dashed red line), $\phi = 0.04$, long polymer (dotted blue line), and $\phi = 0.15$, long polymer (dashed green line).

Longer attraction range results in more nearest neighbors, consistent with the higher d_f in describing greater cluster compactness. These quantitative metrics confirm that ξ, not ϕ, determines cluster morphology.

9.5 Discussion

Morphological differences may also provide insight into how these structures form. Because long-polymer clusters form at a relatively low interaction strength of $U = 1-2k_\mathrm{B}T$, their constituent particles can explore many configurations. Internal rearrangements allow cluster compactification, resulting in higher d_f and more nearest neighbors. We determine the cluster internal volume fraction ϕ_i in two ways: counting particles enclosed by a sphere inscribed in a cluster, and total Voronoi volume for all non-surface cluster particles [40];

both methods yield $\phi_\mathrm{i} = 0.46 \pm 0.02$. These compact clusters resemble those created by the nucleation and growth of binodal decomposition. By contrast, short-polymer clusters do not form until $U \sim 10k_\mathrm{B}T$. The more tightly-bound particles rearrange less before new ones add, resulting in more tenuous, branched clusters, with few nearest-neighbors and a low d_f. These clusters resemble the fractal aggregates formed by DLCA, or in the early stages of spinodal decomposition [46]; their ϕ_i is not well defined.

Our results illustrate how gelation depends on ξ; this is not inherent in a $U - \phi$ state diagram [146]. To capture this behavior, ξ can be represented on a third axis, shown in figure 9.8. Gels occur in the region above the upper checkered surface, while fluids are below the lower surface. Stable fluids of clusters occur in the region between the two surfaces, at lower U as ξ increases.

While the experimental evidence for the existence of a fluid of clusters is unambiguous, these results are at odds with the expectation that attractive clusters should either phase separate completely or gel. One possibility is that the structure is limited by kinetics. However, in the case of the larger particles, typical clusters have $\sim$10 μm radius, for which the Smoluchowski doubling time τ_d is a few hours; we observe stable clusters persisting for days. More strikingly, for typical clusters of the smaller particles, τ_d is $\sim$1 s; we observe clusters persisting for $> 10^5$ s. Therefore, though diffusion kinetics may slow the growth of larger clusters, it seems unlikely that this alone can explain why these clusters are stable. Instead, cluster stability may depend on the details of the attraction mechanism: when two large clusters approach and make contact, their particles do not rearrange quickly enough for the aggregates to merge before they diffuse away from each other; this prevents further cluster growth [146, 50]. Moreover, in no cases do we observe these clusters to break into

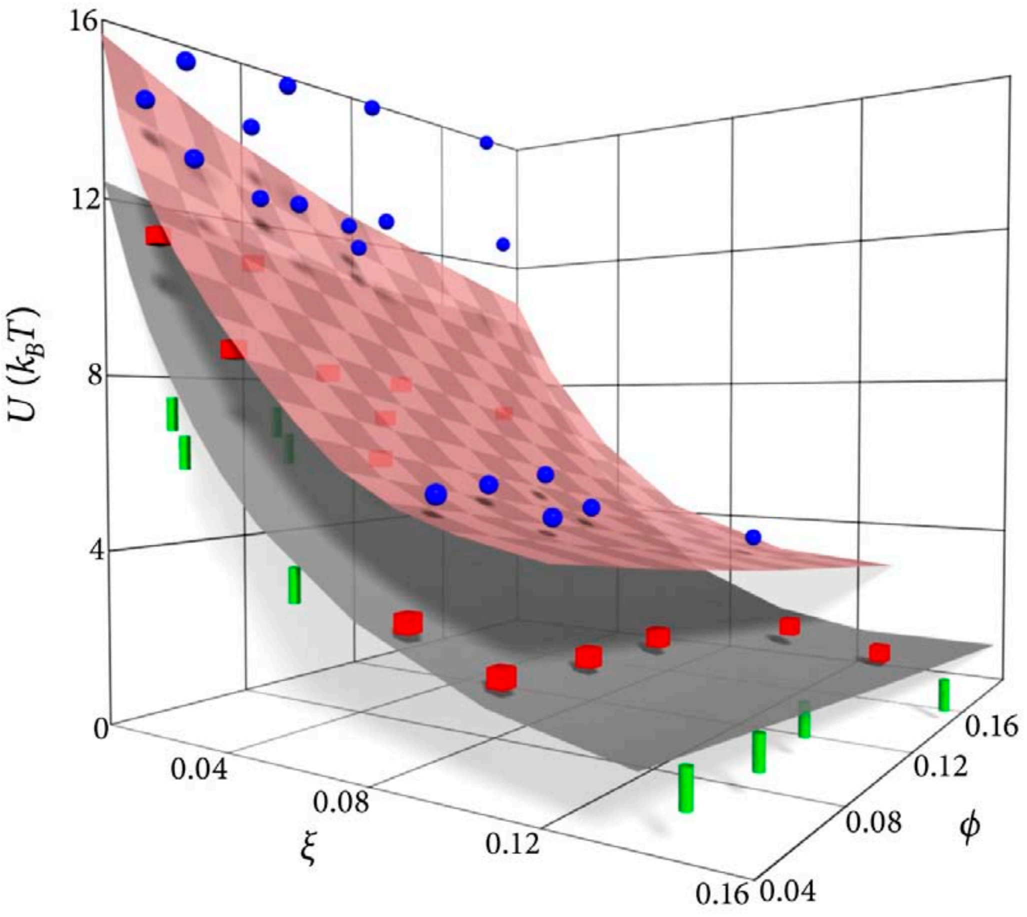

Figure 9.8: U-ϕ-ξ state diagram for colloid-polymer mixtures. Symbols represent the three phases observed in microscopy: gel (sphere), fluid of clusters (cube), and monomeric fluid (cylinder). Surfaces are guides for the eye: Gels are above the checkered surface; monomeric fluids are below the plain surface; fluids of clusters are in between.

smaller ones.

Our results highlight the existence of a stable fluid of colloidal clusters in several attraction ranges, in the absence of long-range repulsion. However, this state's origin remains a fundamental puzzle requiring further inquiry.

9.6 Acknowledgements

We thank James B. Macarthur and Hidekazu Oki for engineering assistance. This work was supported by NASA (NAG3-2284), NSF (DMR-0243715), Harvard MRSEC (DMR-0213805) and Marie Curie RTN for Dynamical Arrest (MRTN-CT-2003-504712).

This chapter has been published in *Physical Review Letters*, co-authored with Jaci Conrad, Hans Wyss, Andy Schofield and Dave Weitz [171].

Chapter 10

Gelation of particles with short-range attraction

10.1 Abstract

Nanoscale or colloidal particles are important in many realms of science and technology. They can dramatically change the properties of materials, imparting solid-like behavior to a wide variety of complex fluids [180, 103]. This behavior arises when particles aggregate to form mesoscopic clusters and networks. The essential component leading to aggregation is an interparticle attraction, which can be generated by many physical and chemical mechanisms. In the limit of irreversible aggregation, infinitely strong interparticle bonds lead to diffusion-limited cluster aggregation (DLCA) [162]. This is long-understood as a purely kinetic phenomenon that can form solid-like gels at arbitrarily low particle volume fraction [146, 304]. Far more important technologically are systems with weaker attractions, where gel formation requires higher volume fractions. Numer-

ous scenarios for gelation have been proposed, including DLCA [70], kinetic or dynamic arrest [274, 25, 146, 206, 44], phase separation [208, 196, 52, 70, 91, 266, 38, 304], percolation [112, 283, 196, 146], and jamming [274]. No consensus has emerged and, despite its ubiquity and significance, gelation is far from understood—even the location of the gelation phase boundary is not agreed on [304]. Here we report experiments showing that gelation of spherical particles with isotropic, short-range attractions is initiated by spinodal decomposition; this thermodynamic instability triggers the formation of density fluctuations, leading to spanning clusters that dynamically arrest to create a gel. This simple picture of gelation does not depend on microscopic system-specific details, and should thus apply broadly to any particle system with short-range attractions. Our results suggest that gelation—often considered a purely kinetic phenomenon [25, 146, 274, 206]—is in fact a direct consequence of equilibrium liquid-gas phase separation [91, 52, 38, 304]. Without exception, we observe gelation in all of our samples predicted by theory and simulation to phase-separate; this suggests that it is phase separation, not percolation [196], that corresponds to gelation in models for attractive spheres.

10.2 Introduction

Gelation occurs in a wide range of systems where particles attract each other [70, 274, 112, 283, 208, 196, 266, 304, 38, 103, 44]. When this attraction is infinitely strong, particles form permanent bonds and grow as fractal clusters that, in turn, bond irreversibly, and can ultimately span the system as a solid-like gel, even as particle volume fraction ϕ tends to zero [196, 146, 304, 46]. This DLCA limit occurs in many colloidal systems where the interparticle attraction strength, U, is much larger than the thermal energy $k_{\mathrm{B}}T$ [196, 146, 304]; examples include gold [167, 162], silica [162], polymeric lattices [162, 46, 70], calcium carbonate [3], alumina [103] and silicon carbide [103]. Because bonds once formed never break, DLCA is governed entirely by diffusion; it has thus been considered a purely kinetic phenomenon [162]. Other mechanisms can cause kinetic arrest at far higher ϕ [304]. Above $\phi \approx 0.58$, particles can arrest because of crowding to form repulsive glasses, even when $U = 0$; weakly attractive particles can form attractive glasses at lower ϕ [304]. Because glasses and DLCA are observed in the same experimental systems, they have been linked within unified pictures of kinetic arrest [25, 146, 206, 44] or jamming [274].

More generally, the onset of gelation can be parameterized by three quantities: namely ϕ, $U/k_{\mathrm{B}}T$, and ξ. The last is the range of the attractive potential in units of a, the particle radius [146, 171]. These three parameters define a three-dimensional state diagram in which a gelation surface demarcates the well-defined boundary between liquid-like and solid-like behavior. Many important attraction mechanisms that drive gelation are short-range ($\xi < 0.1$), including van der Waals forces [3, 266, 274], surface chemistry [103, 112, 283], hydrophobic effects [44], and some depletion interactions [25, 251, 38]. Numerous explanations have been advanced for gelation in this small-ξ limit to predict the fluid-solid boundary in the U-ϕ plane. Non-equilibrium, kinetics-based models have extended the DLCA model to lower $U/k_{\mathrm{B}}T$ by treating bond breakage probabilistically [167, 70, 196]; have connected the gelation boundary to the percolation threshold [112, 283, 196, 146]; and have extended the glass transition to lower ϕ with mode-coupling theory (MCT) applied to local arrest of individual particles [25], to arrest of clusters [146], and in concert with microscopic modelling of the interparticle attractive potential [251]. Thermodynamic models consider gelation initiated by fluid-crystal [208], liquid-gas [70, 52, 38], or polymer-like 'viscoelastic' [266] phase separation, which may arrest owing to percolation [196] or a glass transition [146]. These models make strikingly disparate predictions: there is no agreement on either the gelation mechanism, or the location of the gelation boundary [304, 196, 251].

10.3 Results and Discussion

Here, we explore gelation experimentally with a widely-used model colloid-polymer system [208, 171, 70], where $U/k_{\mathrm{B}}T$ and ξ are controlled by the polymer size and free-volume concentration c_{p}, but in a fashion that is not precisely known. Fixing $\phi = 0.045 \pm 0.005$ and $\xi = 0.059$, we mix samples at various c_{p}; we summarize the samples studied by plotting their values of c_{p}, normalized by the polymer overlap concentration c_{p}^{*}, in the phase diagrams shown in figures 10.1 and 10.2. We eliminate gravitational sedimentation on multiple-day timescales by meticulously matching the colloid and solvent densities to within $< 10^{-4}$. After breaking up particle aggregates by shearing, we observe sample evolution with a high-speed confocal microscope [172].

We observe two phases. In samples with low c_{p}, below the experimental gelation boundary $c_{\mathrm{p}}^{\mathrm{g}}$, we observe a fluid of many clusters that is stable for days; we show a full three-dimensional image of these clusters in the fluid phase for a sample with $c_{\mathrm{p}} = 3.20 \pm 0.03$ mg/mL, the closest fluid-phase value below $c_{\mathrm{p}}^{\mathrm{g}}$, in figure 10.3. By contrast, in samples with $c_{\mathrm{p}} > c_{\mathrm{p}}^{\mathrm{g}}$, particles aggregate immediately into clusters, which in turn form a network that spans the macroscopic sample. This network subsequently arrests to create a gel, which we illustrate for a sample with $c_{\mathrm{p}} = 3.31 \pm 0.03$ mg/mL, the closest gel-phase

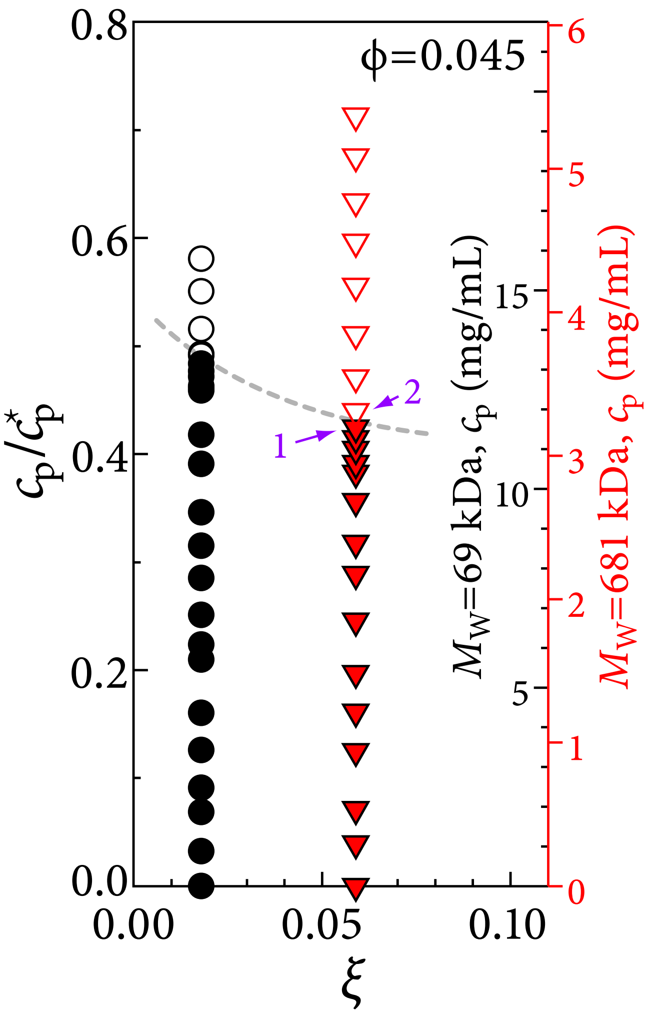

Figure 10.1: Composition of experimental gel and fluid samples: experimental samples in a c_p/c_p^* and ξ phase diagram for constant $\phi = 0.045$. Black circles and red triangles indicate samples with 69 kDa and 681 kDa polystyrene polymers, respectively. Solid symbols mark fluid samples; open symbols, gels. Actual measured c_p values are on secondary vertical axes of the same colour at right. Dashed grey gelation boundary is drawn to guide the eye.

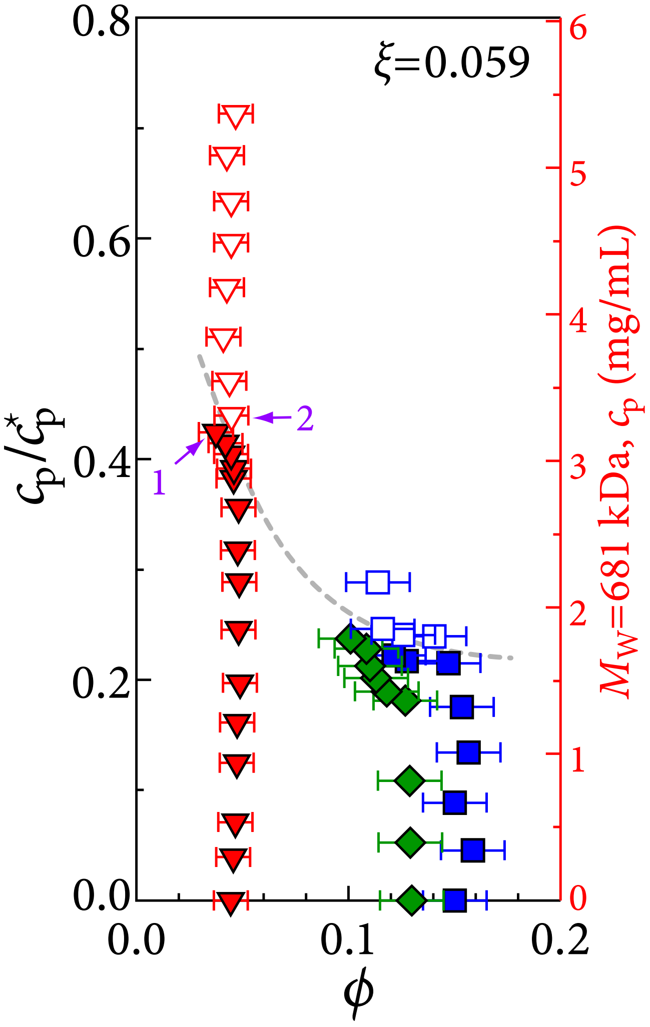

Figure 10.2: Composition of experimental gel and fluid samples: experimental samples in a c_p/c_p^* and ϕ phase diagram for constant $\xi = 0.059$, with c_p of the 681 kDa polymer used in all samples indicated on the secondary red axis at right. Error bars represent the variation in ϕ for different particle configurations from the same sample. Dashed grey gelation boundary is drawn to guide the eye.

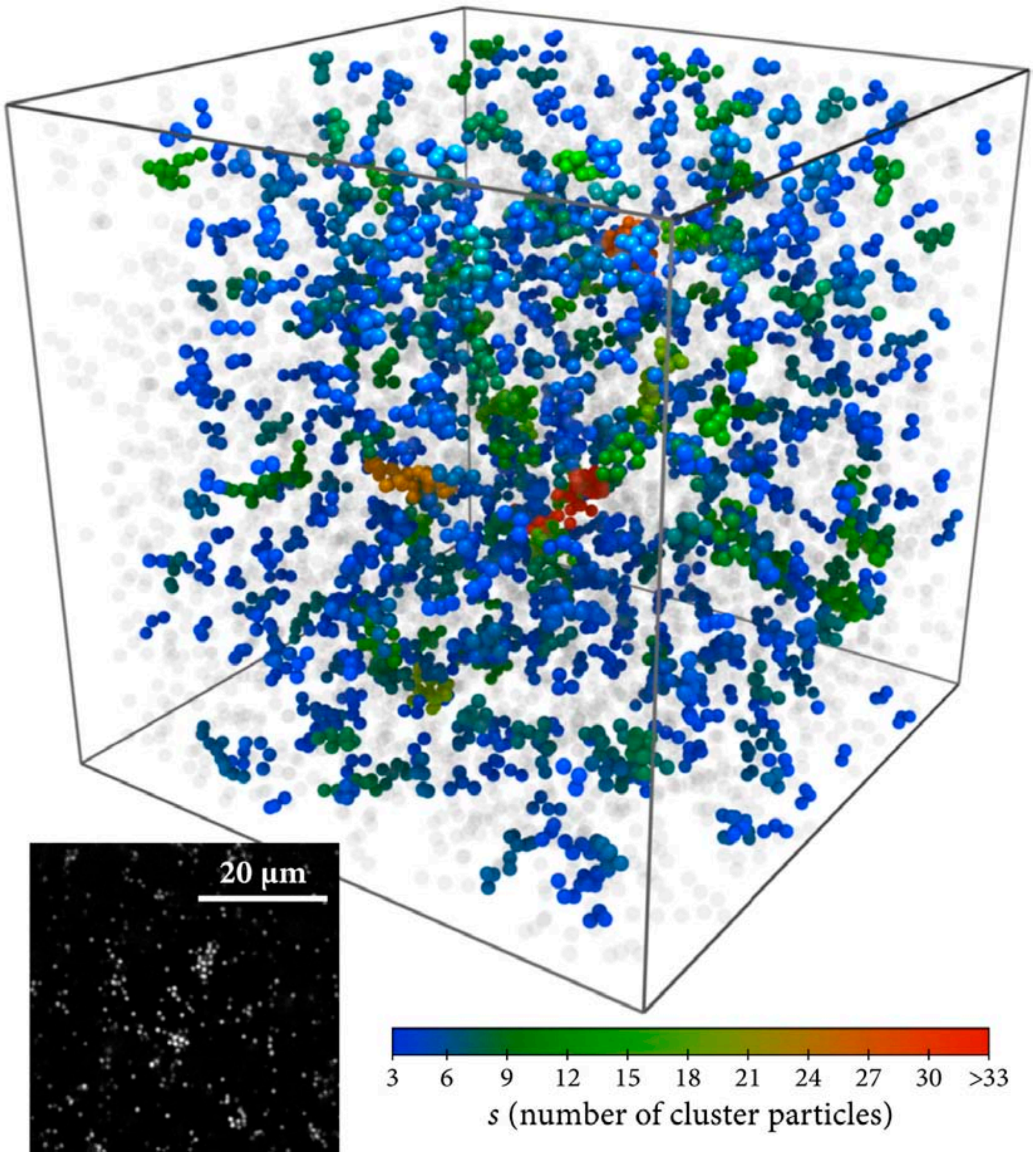

Figure 10.3: Structure of an experimental fluid sample: 3D reconstruction ($56 \times 56 \times 56$ μm^3), and (inset) single 2D confocal microscope image, for the fluid with the highest $c_\mathrm{p} = 3.20$ mg/mL. The fluid's clusters are coloured by their mass s (number of particles) according to the colour bar, with monomers and dimers rendered in transparent grey to improve visibility. The sample is in the long-time steady state four hours after mixing, and its composition is marked in figures 10.1 and 10.2 with a purple numeral 1.

value above c_p^g, in figure 10.4. The gel undergoes no major structural rearrangement for days, even though it exchanges particles with a dilute gas. These phases are separated by a very sharp boundary: the gel and fluid illustrated differ in c_p by only a few percent. Our observation of only these two dramatically different phases contrasts findings of more complex phase behavior in non-buoyancy-matched systems, where sedimentation can shift or obscure the observed phase boundaries [3, 70, 196, 25, 38].

Locating the gelation boundary in general requires a means to compare among experiments and with theory or simulation, using universal thermodynamic quantities, like $U/k_\mathrm{B}T$, instead of system-specific parameters, like c_p [25, 251]. Unfortunately, it is impossible to precisely determine $U/k_\mathrm{B}T$ from a known c_p, even using microscopic models for the potential. Instead, we use the finding that the behavior of an attractive particle system for $\xi < 0.1$ depends not on the shape of the potential, but only on its reduced second virial coefficient B_2^* [188]:

$$B_2^* \equiv (3/8a^3)\int_0^\infty (1 - e^{-U(r)/k_\mathrm{B}T})r^2 dr \tag{10.1}$$

After each fluid sample has reached its long-term steady state, we determine its cluster mass distribution $n(s)$, the fraction of total clusters that contain s particles. We then simulate hard spheres with isotropic short-range attractions at the same ϕ, determining $n(s)$ for different values of B_2^*. For each experimental $n(s)$, we select the closest-matching simulated $n(s)$ using a least-squares minimization. This allows us to associate each c_p with a unique B_2^*, with no adjustable parameters. These fits all work remarkably well, irrespective of the interparticle attractive potential shape, so long as the potentials have the same B_2^*, as shown in figures 10.5, 10.6, 10.7 and 10.8. Identical $n(s)$ are observed for the square-well, generalized Lennard-Jones, and Asakura-Oosawa forms, commonly used for

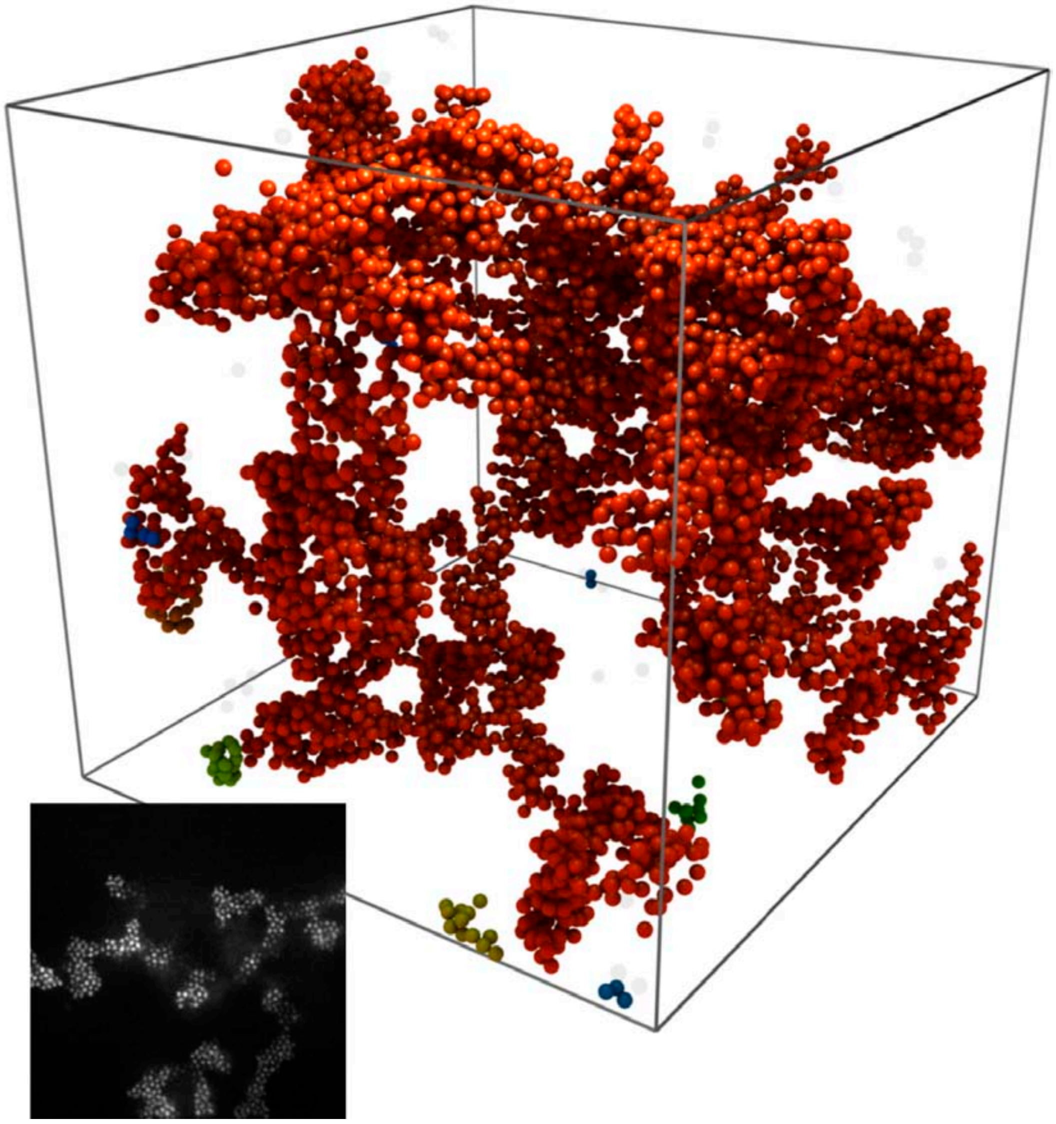

Figure 10.4: Structure of an experimental gel sample: 3D reconstruction ($56\times56\times56\ \mu\text{m}^3$), and (inset) single 2D confocal microscope image of the gel with the lowest $c_\text{p} = 3.31$ mg/mL shown at same scale, containing a single spanning cluster. The sample is in the long-time steady state four hours after mixing; its composition is marked in figures 10.1 and 10.2 with a purple numeral 2.

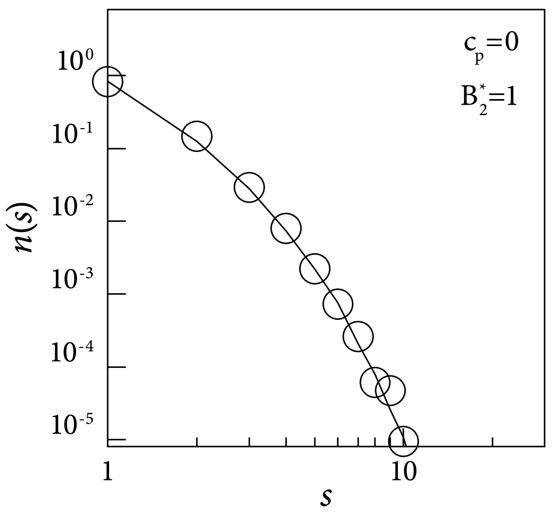

Figure 10.5: Comparisons among cluster mass distributions $n(s)$ at $\phi = 0.045$ between experimental data for $c_\mathrm{p} = 0$ (circles), and simulation results for a hard-sphere potential ($U/k_\mathrm{B}T = 0$ and $B_2^* = 1$; solid line), demonstrating an exact match.

colloid-polymer mixtures [33, 251, 25], substantiating our $c_\mathrm{p} - B_2^*$ mapping even though the exact experimental potential shape remains unknown. Measuring $n(s)$ requires only straightforward counting of particle bonds; by contrast, determining B_2^* with similar precision from scattering [112] or radial distribution functions [224] requires far more accurate identification of particle positions.

From B_2^*, other thermodynamic quantities can be derived directly, including $k_\mathrm{B}T/U$ for different potential forms [188]. Considerable insight is obtained by using $n(s)$ fits to determine the values of $k_\mathrm{B}T/U$, calculated for an Asakura-Oosawa potential with $\xi = 0.059$ to match the experiment, and plotting these as a function of c_p for all fluid samples. The data exhibit an unexpected linear dependence near the experimentally-determined gelation boundary at $c_\mathrm{p}^\mathrm{g} = 3.25 \pm 0.05$ mg/mL, as shown in figure 10.9. We calculate the onset of phase separation both in the Baxter model and with simulation, which, in all cases, yield the same results. Remarkably, these correspond precisely to the experimentally determined

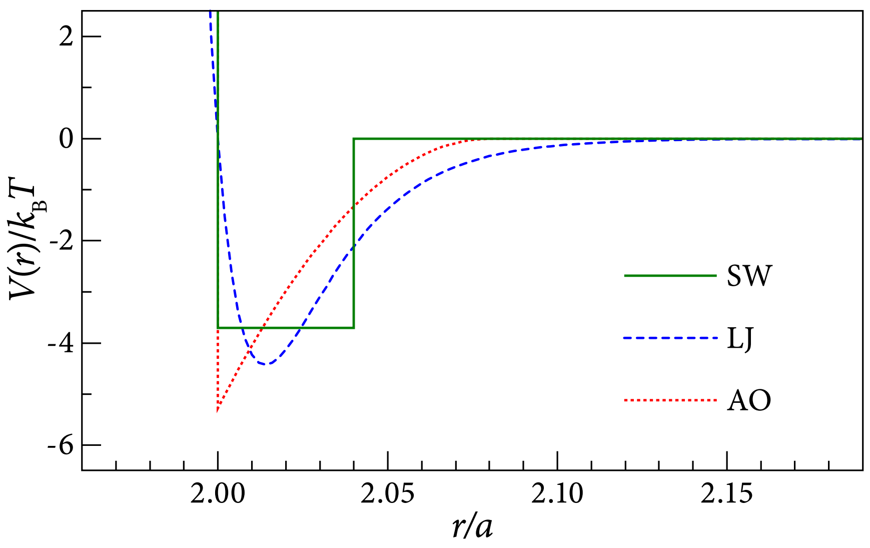

Figure 10.6: Three potentials at the same B_2^* used to generate $n(s)$ in simulations with finite attractions; solid green, dashed blue and dotted red lines denote square-well (SW), generalized Lennard-Jones (LJ) and Asakura-Oosawa (AO) potentials, respectively. Example potentials shown for $B_2^* = -1.47$.

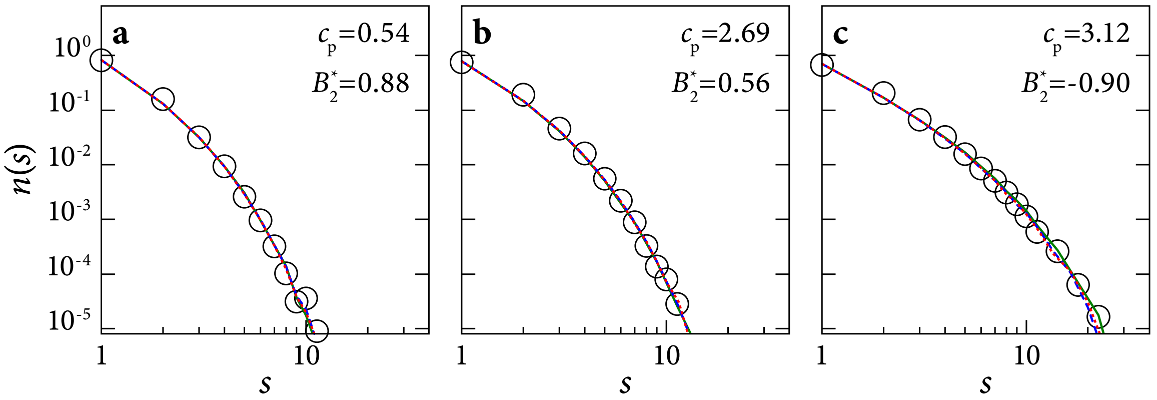

Figure 10.7: Example comparisons at $\phi = 0.045$ and $\xi = 0.059$ between experimental $n(s)$, marked by circles, and simulation $n(s)$, by lines colored as the corresponding potentials in figure 10.6. **(a)** $c_p = 0.54$ mg/mL and $B_2^* = 0.88$. **(b)** $c_p = 2.69$ mg/mL and $B_2^* = 0.56$. **(c)** $c_p = 3.12$ mg/mL and $B_2^* = -0.90$. All data sets match exactly, confirming that $n(s)$ both usefully maps experimental to simulation results and does not depend on potential shape.

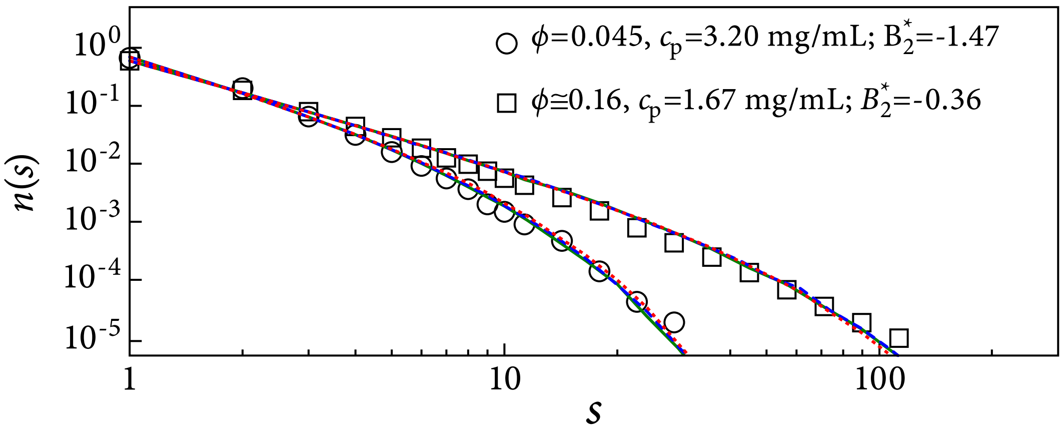

Figure 10.8: Comparisons among cluster mass distributions $n(s)$ for $\xi = 0.059$ for the fluids with the highest c_p closest to the gel boundary at c_p^g. Circles denote the fluid with $\phi = 0.045$ ($c_p = 3.20$ mg/mL and $B_2 = -1.47$; sample illustrated in figure 10.3). Squares denote the fluid with $\phi = 0.16$ ($c_p = 1.67$ and $B_2 = -0.36$), whose significantly larger clusters are expected as the $\phi_c = 0.27$ critical point is approached. All data sets match exactly, confirming that $n(s)$ both usefully maps experimental to simulation results and does not depend on potential shape.

value of $k_B T/U$ at the gel boundary, as shown in figure 10.9. This suggests that the gel boundary occurs exactly at the boundary of spinodal decomposition. Because the spinodal and binodal lines are very close for all short-range potentials, such as those here, we do not observe nucleation and growth—instead, the observed phase separation is always driven by spinodal decomposition.

To confirm the generality of these results, we repeat the experiment for different ϕ and ξ. Again fixing $\xi = 0.059$, we create additional samples at $\phi \approx 0.13$ and $\phi \approx 0.16$, as shown in the phase diagram in figure 10.2. Increasing ϕ results in larger clusters, whose mass distribution broadens to more closely resemble a power law, as shown in figure 10.8; this is reminiscent of an approach to the critical point predicted at $\phi_c \approx 0.27$ [182]. In addition, for $\phi = 0.045$, we also reduce ξ to 0.018; this yields more tenuous, branched, thinner clusters [171]. These samples are shown in the phase diagram in figure 10.1. In

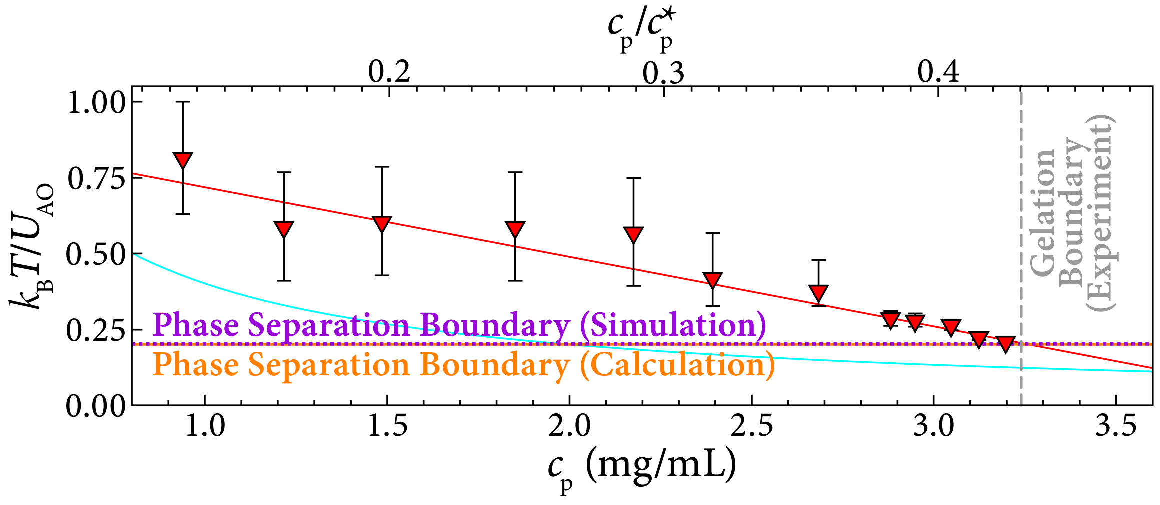

Figure 10.9: Comparison of $n(s)$ mapping of experimental c_p to k_BT/U. Data are shown for $\phi = 0.045$ and $\xi = 0.059$. Grey dashed vertical lines demarcate the experimental gelation boundary at c_p^g; horizontal lines demarcate the theoretical phase separation boundary calculated in the Baxter model (orange solid line) and with simulation (purple dotted line), which always coincide. Coloured symbols (as used in figures 10.1 and 10.2 and shown in the key in figure 10.12) with best-fit lines represent the results of the $n(s)$ mapping illustrated in figures 10.7 and 10.8; error bars correspond to the uncertainty from the least-squares fitting. The experimental gelation boundary exactly matches the theoretical phase separation boundary for all ϕ and ξ; by contrast, analytic approximation to the Asakura-Oosawa potential, shown in light blue, does not match at all.

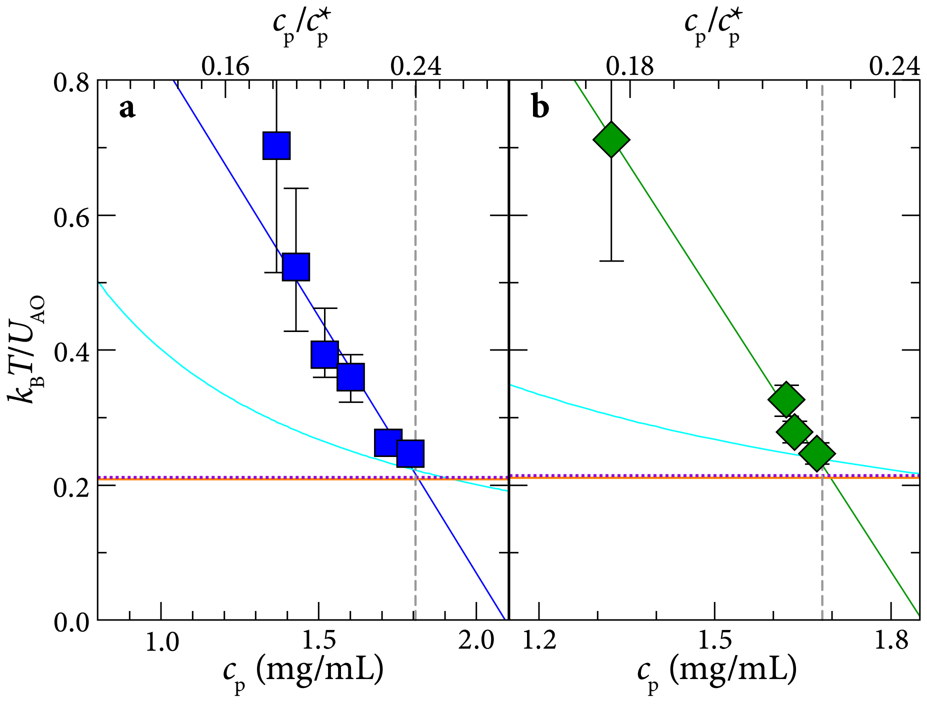

Figure 10.10: Comparison of $n(s)$ mapping of experimental c_p to $k_B T/U$ at $\xi = 0.059$ for **(a)** $\phi \approx 0.13$ and **(b)** $\phi \approx 0.16$. Data are shown for $\phi = 0.045$ and $\xi = 0.059$. Grey dashed vertical lines demarcate the experimental gelation boundary at c_p^g; horizontal lines demarcate the theoretical phase separation boundary calculated in the Baxter model (orange solid line) and with simulation (purple dotted line), which always coincide. Coloured symbols (as used in figures 10.1 and 10.2 and shown in the key in figure 10.12) with best-fit lines represent the results of the $n(s)$ mapping illustrated in figures 10.7 and 10.8; error bars correspond to the uncertainty from the least-squares fitting. The experimental gelation boundary exactly matches the theoretical phase separation boundary for all ϕ and ξ; by contrast, analytic approximation to the Asakura-Oosawa potential, shown in light blue, does not match at all.

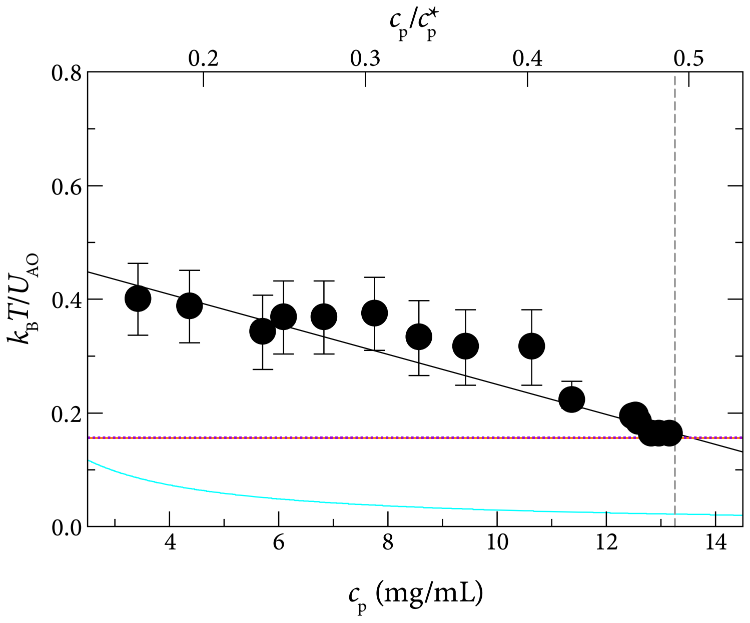

Figure 10.11: Comparison of $n(s)$ mapping of experimental c_p to k_BT/U for $\phi = 0.045$ and $\xi = 0.018$. Data are shown for $\phi = 0.045$ and $\xi = 0.059$. Grey dashed vertical lines demarcate the experimental gelation boundary at c_p^g; horizontal lines demarcate the theoretical phase separation boundary calculated in the Baxter model (orange solid line) and with simulation (purple dotted line), which always coincide. Coloured symbols (as used in figures 10.1 and 10.2 and shown in the key in figure 10.12) with best-fit lines represent the results of the $n(s)$ mapping illustrated in figures 10.7 and 10.8; error bars correspond to the uncertainty from the least-squares fitting. The experimental gelation boundary exactly matches the theoretical phase separation boundary for all ϕ and ξ; by contrast, analytic approximation to the Asakura-Oosawa potential, shown in light blue, does not match at all.

all cases, the experimentally-determined value of k_BT/U at the gelation boundary coincides exactly with the theoretical phase separation boundary, as shown in figures 10.10 and 10.11. Finally, we consider the dependence of $B_2^* - 1$, normalized by the value at the phase separation boundary, as a function of c_p/c_p^g. Unexpectedly, despite significant variation in cluster morphology, all sample data scale onto a single master curve, shown in figure 10.12. This highlights the similarities in behavior of all samples on approach to the spinodal line and points to a universal mechanism for gelation.

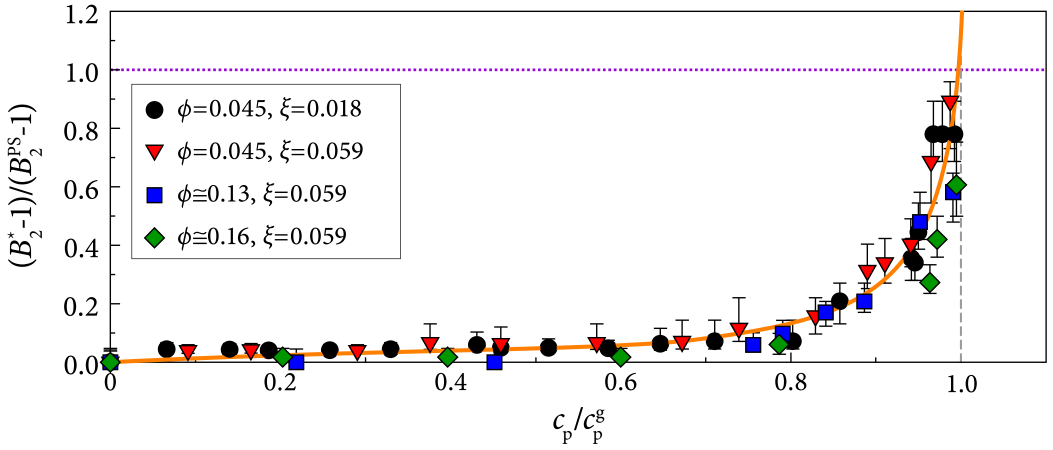

Figure 10.12: Mapping between c_{p} and $B_2^* - 1$ for all fluid samples, where c_{p} is normalized by $c_{\mathrm{p}}^{\mathrm{g}}$ (dashed grey vertical line), and $B_2^* - 1$ by $B_2^{\mathrm{PS}} - 1$, its value at the phase separation boundary (purple dotted horizontal line). All data collapse onto a single master curve, highlighted with an orange line to guide the eye. Gelation exhibits universal scaling independent of ϕ, ξ or shape of the short-range potential.

These data suggest that, for isotropic short-range interactions, all gelation is triggered by spinodal decomposition, a phase separation process driven by a thermodynamic instability. If this is so, then we should independently observe other characteristics of equilibrium phase separation in samples that form gels. One such feature is the coexistence of gel and colloidal gas: we observe occasional exchange of particles between gas and gel; this is not readily explained by kinetic gelation models based on local arrest [25, 206]. An even more distinctive hallmark of spinodal decomposition is the development of a peak in the static structure factor $S(q)$ at finite scattering vector q [46, 241]. We again observe this: in fluid samples with $\phi = 0.045$, $\xi = 0.059$, and $c_{\mathrm{p}} < c_{\mathrm{p}}^{\mathrm{g}}$, $S(q)$ shows only a slight rise at low q; however, increasing c_{p} by just a few per cent across $c_{\mathrm{p}}^{\mathrm{g}}$ increases the height of the peak in $S(q)$ by two orders of magnitude, as shown in figure 10.13a. Further distinguishing characteristics of spinodal decomposition occur in the temporal evolution of $S(q)$, where the peak narrows and moves toward lower q, and in its first moment $q_1(t)$, which exhibits

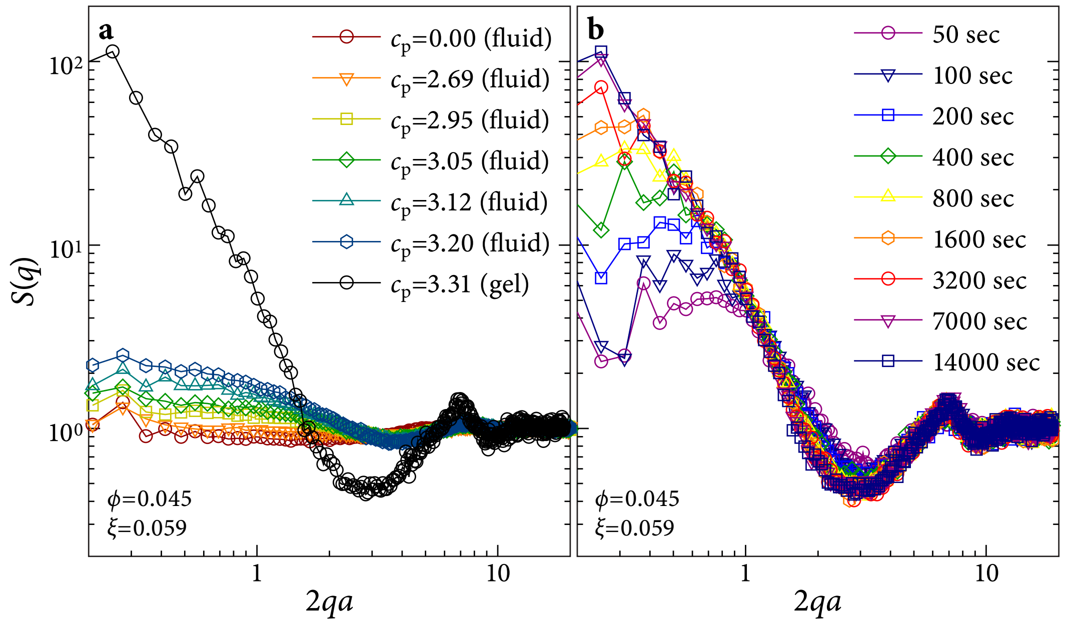

Figure 10.13: Spinodal decomposition in samples that form gels. **(a)** $S(q)$ in the long-time steady-state limit for fluid samples at $\phi = 0.045$ and $\xi = 0.059$ with $c_p \leq 3.20$ mg/mL (coloured symbols) and the gel sample with $c_p = 3.31$ mg/mL (black circles). Blue hexagons and black circles denote the fluid and gel samples illustrated in figures 10.4 and 10.3. All fluid samples show $S(q)$ rising slightly at low q as $c_p \to c_p^g$. As c_p crosses c_p^g into the gel region, $S(q)$ develops a significant peak two orders of magnitude higher. **(b)** Time evolution of $S(q)$ for this gel. Immediately after sample homogenization, a finite-q peak grows, narrows, and shifts to lower q, as expected for spinodal decomposition.

a power law dependence. Once again, the gel samples unambiguously demonstrate these features: at the earliest times, the peak in $S(q)$ narrows and moves to lower q, as shown in figure 10.13b; moreover, q_1 scales as $t^{-1/6}$, as shown in figure 10.14, exactly as in molecular spinodal decomposition [99]. Two hours after mixing, the spinodal decomposition toward the equilibrium phase-separated state is interrupted, as the sample dynamically arrests to form a gel; $S(q)$ and q_1 no longer change with time, as shown in figures 10.13b and 10.14. Similar dynamics for $S(q)$ are observed in all gel samples, further demonstrating that liquid-gas spinodal decomposition ubiquitously induces gelation for short-range potentials.

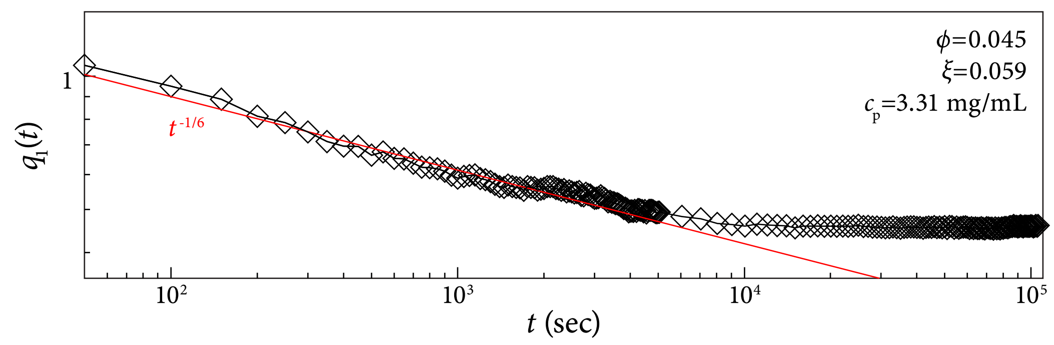

Figure 10.14: $q_1(t)$ (black diamonds) follows the $t^{-1/6}$ power law (red line), another hallmark of spinodal decomposition. After two hours, the sample arrests to form a gel, and $S(q)$ and q_1 do not change.

Together, these results provide strong, quantitative physical evidence that the gelation boundary for short-range attractive particles is precisely equivalent to the boundary for equilibrium liquid-gas phase separation. Gelation requires spinodal decomposition to generate the clusters that span the system and dynamically arrest. Our findings experimentally confirm previous theoretical predictions [91, 52, 304], and support the suggestion that the ostensibly purely-kinetic DLCA regime is in fact a deeply quenched limit of spinodal decomposition [46, 241]. Thus, thermodynamic instability appears to drive all gelation of particles with isotropic short-range attractions.

We cannot harmonize our results with predictions from phase separation that is not liquid-gas [208, 266], nor from purely-kinetic paradigms [274, 25, 146, 206]. However, the expression of these predictions as system-specific c_p/c_p^* values calculated for the Asakura-Oosawa potential may affect comparison of results. To test this, we plot k_BT/U vs. c_p/c_p^* for an analytic approximation to the Asakura-Oosawa potential [25] in figures 10.9, 10.10 and 10.11, which in all cases dramatically misses the actual potential strength determined from the $n(s)$ mapping; this corroborates previous findings that the Asakura-Oosawa model does not quantitatively describe colloid-polymer mixtures [251, 33, 224].

Instead, universal system-independent parameters, such as B_2^* [112, 196, 283, 91, 38, 304] and ϕ, allow meaningfully quantitative comparison between different experiments and with theory. We present such a comparison, as a universal phase diagram for short-range gelation, in figure 10.15. Without exception, all samples predicted within the Baxter model to phase-separate form gels. This suggests that the gelation line coincides with the phase separation boundary in the Baxter model; other isotropic short-range potentials have similar behaviour. For gel samples, we estimate the volume fractions in both colloidal gas and gel phases by numerically determining the free volume accessible to a test particle of radius a; we consider this the total volume of the gas phase, and assign the remaining volume to the gel. Surprisingly, we find the that all spanning gel clusters have $\phi_\mathrm{g} \approx 0.55$, independent of both c_p and the average ϕ before phase separation. We never observe arrested spanning clusters with significantly lower ϕ_g; the attractive glass line must therefore intersect the phase separation boundary at $\phi \approx 0.55$ [91, 304], consistent with the origin of kinetic arrest arising from the dense phase undergoing an attractive glass transition [91, 304]. Furthermore, ϕ_g does not decrease with increasing attraction strength [146, 25, 44], suggesting that the attractive glass line does not extend into the phase separation region, but instead follows its boundary.

Our results could shed light on non-equilibrium behavior in technological systems. Even approximate measures of structural parameters, such as $n(s)$, may, when compared with simulations, allow mapping between thermodynamic quantities and experimental parameters when even the rough form of the potential cannot be measured. Moreover, because the onset of non-equilibrium behavior is in fact governed by equilibrium phase separation,

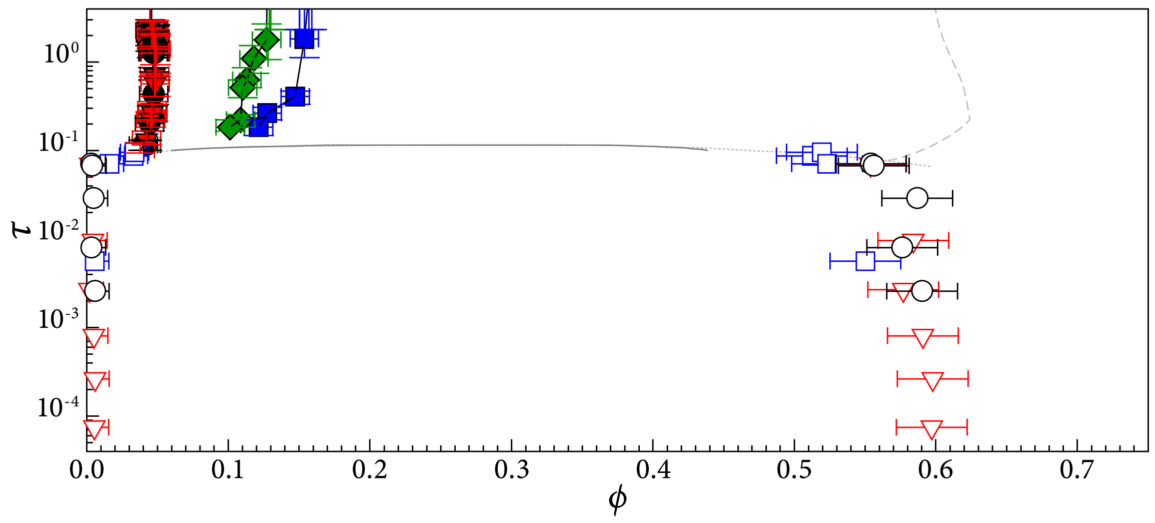

Figure 10.15: Universal phase diagram of the Baxter temperature $\tau \equiv 1/4(B_2^* - 1)$ and ϕ for all samples, with symbols as in figures 10.1 and 10.2 and estimates of ϕ shown for both gas and gel phases after phase separation. Error bars represent the variation in ϕ for different particle configurations from the same sample. All samples predicted to phase-separate within the Baxter model, falling below the theoretical phase separation boundary from [182] (solid grey line), form gels with the same ϕ_g. Speculative extensions of this boundary (dotted grey line) and of the glass transition (dashed grey line) are plotted to guide the eye.

thermodynamic calculations may facilitate quantitative prediction of product stability, a critically important problem in the formulation and manufacture of commercial complex fluids.

10.4 Methods

10.4.1 Methods Summary

We suspend polymethylmethacrylate (PMMA) colloidal spheres of radius $a = 560$nm in a solvent mixture with matching buoyancy and refractive index, adding an organic salt to screen Coulombic repulsion and linear polystyrene to induce a depletion attraction [171, 172]. We determine the radii of colloid and polymer coils with light scattering. We image all samples in a high-speed, automated confocal microscope [172], collecting 181 images at 10 frames per second in each three-dimensional (3D) stack, which occupies a $60 \times 60 \times 60 \mu\text{m}^3$ cube within the sample. We use previously described image-processing software [172] to determine the 3D positions of all colloidal particles in each sample. In total, we collected half a terabyte of image data and located $\sim 10^8$ particles. We use Pixar's RenderMan (https://renderman.pixar.com) to create 3D reconstructions. We perform simulations of fluid samples of 10,000 particles in a cubic box with periodic boundary conditions for several values of B_2^*, using several simulated potentials: a hard-sphere potential, a square well potential of width $0.04a$, an Asakura-Oosawa potential of maximum width $0.08a$, and a generalized $2\alpha - \alpha$ LennardJones potential with exponent $\alpha = 100$. Following a constant-temperature equilibration run, we generate 100 independent realizations in the micro-canonical ensemble for subsequent analysis. We estimate the spinodal line following the temperature-dependence of the energy and of the small angle structure factor within simulations [91], and using the energy route in the Percus-Yevick approximation to the Baxter model for hard spheres with an infinitesimally-short attraction range [182]. We use the same procedure in experiment and simulation to assign particles to clusters by considering which particles share common bonds; two particles are considered bonded if they are separated by less than the bond distance r_b, fixed by matching the $c_\mathrm{p} = 0$ cluster-mass distributions. We use a least-squares minimization to best match numerical distributions to the experimental results with no free parameters.

10.4.2 Colloid Sample Preparation

Following our previously-reported procedure [171, 172], we equilibrate sterically stabilized colloidal spheres of polymethylmethacrylate (PMMA) with DiIC$_{18}$ fluorescent dye in a 5:1 (by mass) solvent mixture of bromocyclohexane (CXB, Aldrich) and decahydronaphthalene (DHN, Aldrich) for several months. We add tetrabutylammonium chloride (TBAC, Fluke) until saturated ($\sim$4 mM) to screen long-range Coulombic repulsion. We then split the colloid suspension to create two stock solutions, adding linear polystyrene (Polymer Labs) depletant polymer to one. We buoyancy-match each stock solution individually by adding either CXB or DHN dropwise until particles remain neutrally buoyant after centrifuging at 1000g for 30 min at 25 ± 0.1°C. Mixing various ratios of the two stock solutions generates samples at varying c_p, while maintaining constant ϕ, TBAC concentration, and buoyancy match.

We determine the radius $a = 560 \pm 10$ nm of our particles with dynamic light scattering [96]. The solvent has viscosity $\eta = 1.96$ mPa-sec at 25 ± 0.1°C, measured with a Cannon-Fenske viscometer. For the depletant polystyrene, we selected two molecular weights, $M_W = 69.2$ kDa and $M_W = 681$ kDa. From Zimm plots of static light scattering data, we determine the radii of gyration r_g of the two polymers to be 10.0 and 33.0 nm, respectively. This yields

$$\xi \equiv r_g/a \tag{10.2}$$

of 0.018 and 0.059, respectively, and overlap concentrations

$$c_p^* \equiv 3M_W/4\pi r_g^3 N_A \tag{10.3}$$

of 27.2 and 7.5 mg/mL, respectively, where N_A is Avogadro's number. In all cases, we directly measure the raw polymer concentrations as a mass ratio of mg polystyrene per g of total sample mass, which we express as a ϕ-dependent free-volume c_p (mg/mL) according to [154].

10.4.3 Confocal Microscopy

Following our previously-reported imaging protocol [171, 172], we load each sample into a glass capillary of internal dimension $50 \times 2 \times 0.1$ mm^3 (VitroCom), along with a small piece of magnetic wire with 25 μm diameter; we then seal the capillary with 5-minute epoxy (DevCon). After sealing, we can rehomogenize the sample at any time by agitating the magnetic wire with a magnetic stirrer. We maintain the temperature of the microscope stage and surrounding air at 25 ± 0.2°C, yielding a buoyancy match between colloid and solvent that is better than 10^{-4}. With the confocal microscope, we collect 3D stacks of 181 8-bit images, each 1000×1000 pixels, at 10 frames per second. Each image stack covers a volume of $60 \times 60 \times 60$ μm^3, taken from the center of the sample at least 20 μm away from any capillary surface to minimize edge effects.

Although larger clusters persist in these samples, the confocal microscope can collect 3D stacks only a few times a minute, far too slowly to track monomers, dimers and other small clusters. Therefore, to ensure a broad sampling, after homogenization and equilibration for four hours, we collect 26 independent 3D image stacks within each fluid sample, separated by 100μm laterally, using our automated confocal microscope [172]. To observe the evolution of gel samples, we homogenize and immediately start observations, collecting 3D stacks of the same sample volume every 50 s for the first 5000 s, then every 1000 s for 100,000 s. In each 3D stack, we determine the 3D position of each particle more than 1 μm from the boundary of the imaging volume using previously-described image processing software [172], and measure ϕ for each sample from these particle counts. In total, we collected half terabyte of image data and determined the positions of $\sim 10^8$ particles. Our 3D reconstructions were rendered with Pixar's RenderMan.

10.4.4 Simulations

We perform simulations of N=10,000 particles in a cubic box with periodic boundary conditions. For comparison to experimental samples with $c_\mathrm{p} = 0$, we use the hard-sphere potential. For comparison to fluid samples with $c_\mathrm{p} \geq 0$, we use three different attractive potential shapes, as shown in figure 10.6: a square-well of width 0.04a, an Asakura-Oosawa potential [10] of maximum width 0.08a, and a generalized 2α-α Lennard-Jones potential with exponent $\alpha = 100$ [289]. For the Asakura-Oosawa potential, we use standard Monte Carlo simulations [4]; for the hard-sphere and square-well potentials, a standard event-driven algorithm [218]; and for Lennard-Jones potential, molecular dynamics [4]. In all cases, the system is at first equilibrated in the *NVT* ensemble, followed by a production run in the *NVE* ensemble, where 100 independent realizations are collected and analyzed.

10.4.5 Cluster Mass Distribution Comparisons

In particle configurations from both experiment and simulation, we define two particles as bonded if their centres are separated less than the bond distance r_b. All particles in a cluster share at least one bond with at least one other particle in the same cluster. Particles in one cluster share no bonds with particles in other clusters. Experimental uncertainties in particle locations arise from particle diffusion during confocal imaging, forcing the choice of r_b to be slightly larger than its ideal value of the particle diameter $d = 2a$ plus the interaction range, e.g. $1.08d$ for the previously-described Asakura-Oosawa potential. We therefore set r_b by matching the hard-sphere simulations to the sample with $c_\mathrm{p} = 0$, fixing this value for all samples at $r_\mathrm{b} = 1.16d$; $n(s)$ comparisons are independent of the particular choice of r_b, so long as a consistent definition is applied to both experiments and simulations. For each experimental sample, we ran the simulations at the same ϕ. The least-squares procedure to match $n(s)$ from experiment and simulation equally weights all clusters.

10.4.6 Static Structure Factor

For fluid samples, we average the static structure factor

$$S(q) \equiv \frac{\left\langle \left| \sum_{j=1}^{N} e^{i\mathbf{q}\cdot\mathbf{r}_j} \right|^2 \right\rangle}{N} \tag{10.4}$$

where $\mathbf{r}_j$ are the coordinates of the particle j, over the 26 independent configurations. For the gel samples, we follow a single configuration over time. We calculate $S(q)$ for all particles more than 4 μm away from all boundaries of the imaging volume to minimize edge effects, which, if present, would affect only the range $2qa \leq 0.2$. For the first moment

$$q_1(t) \equiv \frac{\int_0^{q_\mathrm{c}} S(q,t) q dq}{\int_0^{q_\mathrm{c}} S(q,t) dq} \tag{10.5}$$

we select the cutoff value $2q_\mathrm{c}a = 3$ to ensure the inclusion of all large wave-length contributions.

10.4.7 Estimation of volume fraction and B_2^* for gel samples.

We extend the linear fit of the $U/k_\mathrm{B}T$ vs. c_p for the fluid samples into the gel region at each ϕ to estimate

$$\tau \equiv \frac{1}{4(B_2^* - 1)} \tag{10.6}$$

for the gel samples shown in figure 10.15. We estimate ϕ_g, the internal volume fraction for spanning gel clusters, defined as those touching opposite faces of the cubic imaging volume, by measuring the free volume accessible to a spherical test particle of radius a. Splitting the imaging volume into a fine grid of cubes with edge length $l_c \ll a$, we place a test particle in each cube, and if no part of it intersects with spanning cluster particles, the volume occupied by the test particle is considered to be in the free volume. The fraction of sample volume not part of the free volume is considered to be the total cluster volume. The total volume of the particles within the cluster is their number times the volume per particle; dividing this by the total cluster volume yields ϕ_g. We selected $l_c = 0.25a$, but the measured ϕ_g values do not depend on l_c for values below $\sim a/2$, and converge as expected for tests on standard structures, such as a cluster of the f.c.c. lattice, where $\phi \rightarrow 0.74$. This approach is strictly applicable only to structures, such as the present gels, where the solid phase is more dense at the scale of a single particle; our centrosymmetric interparticle attraction allows bond rotation without energy cost, thereby requiring multiple bonds for stable structures, leading to locally higher densities at the single-particle scale. By contrast, in the $\phi \to 0$ limit of DLCA, the permanent particle bonds are fixed and do not allow rotation, resulting in a more string-like local structure. For a straight line of spheres, our measure yields the analytic result

$$\phi = \frac{4}{(10 - \sqrt{3}\pi)} \approx 0.88 \tag{10.7}$$

but this is less meaningful in this regime.

10.5 Acknowledgments

I thank Dan Maas (Maas Digital), Mark Christiansen (Christiansen Creative) and Saty Raghavachary (DreamWorks Animation) for assistance in producing the renderings and movies. This work was supported by NASA, the NSF, the Harvard MRSEC, MIUR-Prin and the Marie Curie Research and Training Network on Dynamical Arrested States of Soft Matter and Colloids.

This chapter has been published as a Letter to *Nature*, co-authored with Emanuela Zaccarelli, Fabio Ciulla, Francesco Sciortino, Andy Schofield and Dave Weitz [173].

Chapter 11

Spinodal Decomposition near the Critical Point in Microgravity

11.1 Introduction

The final chapter of this thesis describes an experiment we are conducting on-board the International Space Station (ISS), entitled BCAT (Binary Colloidal Alloy Test), which involves photographing phase separation in colloid-polymer mixtures in the microgravity environment of low-earth orbit.

There are several key differences between the BCAT samples and the colloid-polymer mixtures described in chapters 4, 9 and 10. Here, the PMMA particles are undyed and far smaller than the diffraction limit, so even if they were fluorescent, they could not be imaged on the single-particle level. Structural characterization, performed mainly with macro-lens photography, is consequently far less detailed; even such parameters as colloid volume fraction ϕ, which are trivial to determine from confocal images by simply counting particles, become surprisingly difficult to assess accurately here. On the other hand, because the particles have no fluorescent dye, they are chemically far cleaner; they do not suffer from dye leakage, for example, and can therefore be heat-shocked to a much greater degree without any degradation. Another major difference is that, by going to microgravity, the solvent and colloid need not be buoyancy-matched. Therefore, we used a solvent mixture of cis-decahydronaphthalene (cDHN) and tetrahydronaphthalene (THN), which is generally considered the best system to create as close to a hard-sphere system as can relatively easily be achieved. The main drawback is that, because the samples are not neutrally buoyant on earth, there is no simple way to connect observations in the lab here with those made on orbit.

This chapter, then, details what we have done in the experiment, the specific methods used, and presents the data we have acquired and analyzed so far. We have not yet reached firm conclusions, so the brief discussion section will primarily focus on the issues regarding the data and its interpretation, and some strategies that will be pursued in the near future.

11.2 BCAT Sample Preparation and Characterization

In stark contrast to the buoyancy-matched colloid-polymer mixtures whose preparation and imaging were detailed in chapters 4, 9 and 10, the colloids in the present work are not density-matched. Instead, by launching samples into orbit and imaging them on the ISS, we rely on the natural microgravity environment to remove sedimentation—on timescales of months or longer, far better than could be achieved reasonably on earth, and with much larger samples, without any fine temperature control.

On the other hand, relative to those used in confocal imaging, the samples are much larger, and therefore must have a much closer match between colloid and solvent refractive index, in order to image these samples without the multiple scattering that would lead to photograph-obscuring turbidity. The solvent system I use here is a mixture of cDHN and THN in a ratio of 45.00 to 55.00 by mass. This very closely matches the index of refraction, so that, when particles are close packed, a suspension several centimeters thick is still transparent (i.e. single-scattering). So the methods for sample preparation and characterization, while related to those previously described in previous chapters, are nonetheless different in important ways.

In this section, I describe the characterization of solvent density ρ_s, solvent viscosity η_s, solvent refractive index n_s, colloid radius a, polymer gyration radius r_g, and colloid volume fraction ϕ, and then detail the colloid-polymer sample preparation protocol.

11.2.1 Solvent Density

I measure the density in two ways. The first is to use a macroscopic pycnometer, which is essentially a small flask with a very narrow neck, so that it can be filled to the same volume with different fluids, and an integrated thermometer to measure the temperature of the solvent directly. The procedure to measure density, then, relies on measuring several different masses, which I did using a Mettler AT261DR balance:

1. Weigh the empty pycnometer: 32.3435 ± 0.0001 g.

2. Fill with deionized water (18 MΩ resistance, from the Millipore Milli-Q filtration system), and weigh again: 41.8725 ± 0.0001 g, which yields a net mass of water of 9.5290 ± 0.0001 g.

3. Drain the water, clean and dry pycnometer, weighing again to confirm: 32.3421 ± 0.0001g.

4. Fill the pycnometer with the solvent (45:55 cDHN:THN), and weigh again: 41.2621 ± 0.0001g, which yields a net mass of solvent of 8.9200 ± 0.0001g.

5. Multiplying the ratio of the two solvent masses by the known density of water ($\rho_\text{w} =$ 0.9975 mg/mL at 22.4°C) yields the final solvent density:

$$\rho_\text{s} = 0.9361\rho_\text{w} = 0.933 \text{ g/mL} \tag{11.1}$$

I measured the density in a second way, using similar principles, with a microgram balance and a glass capillary. Note that the Mettler AX26DR I used is sensitive to ± 0.002 mg, and quite reproducible—given that the mass measurements are done carefully, and the balance is given sufficient time (often minutes) to stabilize. The following sequence takes roughly a quarter of an hour to do properly:

1. Weigh the empty capillary: 51.422 ± 0.002 mg.

2. Fill with deionized water (18 MΩ resistance, from the Millipore filtration system), and weigh again: 61.994 ± 0.002 mg. The net mass of the water is: 10.572 ± 0.002 mg.

3. Drain water, clean and dry capillary with rinsing out three times with acetone (low viscosity helps to drain all fluid), weighing again to confirm: 51.422 ± 0.002 mg.

4. Fill with the solvent (45:55 cDHN:THN), and weigh again: 61.304 ± 0.002 g. The net mass of solvent: 9.882 ± 0.002 mg.

5. From the known density of water, multiplying the ratio of the two solvent masses by the known density of water (0.998 mg/mL at 23°C) yields the final solvent density: 0.933 g/mL.

I performed this measurement in the capillaries on seven samples, resulting in an average density of:

$$\rho_\mathrm{s} = 0.933 \pm 0.002\ \mathrm{mg/mL} \tag{11.2}$$

where the uncertainty is the standard deviation from the set of measurements, and the result agrees with the value determined with the pycnometer, as listed in equation 11.1.

11.2.2 Solvent Viscosity

Solvent viscosity η_s can be measured in several ways. One is to use a rheometer, and look at the loss modulus as a function of frequency. But for solvents with very low viscosity, the measurements approach the noise limit for the rheometer, and the resulting data can be unreliable. Instead, I opted for a more traditional approach: the glass viscometer. Essentially, this is a simple glass tube with a well-characterized inner profile. Fluid is flowed through the tube, and the time it takes to pass (the efflux time) is directly related to its kinematic viscosity (in units, for instance, of centiStokes). A viscometer is calibrated in units of centiStokes per second (cSt/s), so that the efflux time multiplied by this calibration constant yields the *kinematic* viscosity. The kinematic viscosity is multiplied by the solvent density (ρ_s, as determined by the methods described in section 11.2.1) to give the *dynamic* viscosity, measured in the more familiar units of centipoise or milliPascal-seconds. The measurement involves timing how long a fluid takes to flow through a glass tube, which sounds kind of trivial—indeed, getting the number correct to $\pm 10\%$ is straightforward.

However, actually doing the measurement and getting reproducible results within a percent turns out to be quite tedious.

I used a Cannon-Manning semi-micro viscometer of size 50, with a viscometer constant that depends on temperature equal to 0.003921 cSt/s at 100°C and 0.003931 cSt/s at 40°C. I used a constant of 0.003934 cSt/s at 25°C for the measurements here. I first flowed the 45:55 cDHN:THN solvent without being particularly careful, just to get a very rough approximation of the viscosity. The efflux time was about ten minutes, yielding a kinematic viscosity of about 2.4 cSt, and a dynamic viscosity of about 2.3 mPa-s.

Because both the viscosity of the solvent and calibration constant of the viscometer are temperature-dependent, the viscometer must be carefully temperature-controlled to give an accurate measurement. This is not completely trivial, however, since the tube itself is nearly 8 inches tall, longer than any of the water baths in the lab is deep. So I set up a large 2-liter beaker on a probe-controlled programmable hot plate (Ika), and set the plate to keep the water temperature at 25.0 ± 0.2°C. This large volume of water took more than 24 hours to stabilize, and measurements while the bath wasn't stable led to erratic results (large scatter in the efflux time). After the bath settled down, I made several measurements of the efflux time: 590.19, 590.38, 590.28, 590.13 588.22, and 588.94 s, yielding an average of 589.83 s. The kinematic viscosity is 2.320 ± 0.006 cSt, where the uncertainty is the standard deviation of the six measurements. Multiplying by the density ρ_s determined in section 11.2.1 yields a dynamic viscosity of

$$\eta_s = 2.165 \pm 0.005 \text{ mPa-s} \tag{11.3}$$

As a check, I performed the same set of measurements on deionized water from the Millipore, measuring efflux times of 234.91, 233.91, 232.15, 233.41, 232.31 and 234.34 s, with an average of 233.51 s. This yields the kinematic viscosity of 0.919 ± 0.005 cSt and a dynamic visocity of $\eta_w = 0.916 \pm 0.005$ mPa-s. This is approximately 2% lower than the value in the literature at the temperature of 25°C, but almost exactly the value at 24°C. The cause of the discrepancy could be the accuracy of the hotplate's external thermal probe, since thermocouples are known to give the correct absolute temperature to only about ± 1°C (even if they can measure relative temperatures far more accurately). This limits the accuracy, in general, of all measurements on this system to about ± 1%.

11.2.3 Solvent index of refraction

A drop of solvent was placed in a standard refractometer, and its index of refraction was measured at 22.2°C to be:

$$n_\mathrm{s} = 1.511 \tag{11.4}$$

As a calibration check on the refractometer, I also measured the value of water (18 MΩ deionized H_2O from the Millipore filtration system) at 22.2°C and found it to be $n_\mathrm{w} = 1.333$, in agreement with the literature value.

11.2.4 Colloid Radius

Accurate values of ρ_s and η_s are crucial for proper determination of colloid and polymer sizes. The size ratio,

$$\xi = \frac{r_\mathrm{g}}{a} \tag{11.5}$$

where r_g is the polymer radius of gyration, and a is the colloid radius, is important for understanding the behavior of colloid-polymer mixtures. Together with ϕ and B_2^*, ξ forms a set of three universal parameters that determine the location of the phase boundaries and critical point, as discussed in chapters 9 and 10 and in [146].

I determine a by using dynamic light scattering to measure the diffusion constant D_c from a dilute dispersion of colloidal spheres, from which a radius is determined following the Stokes-Einstein relation:

$$D_\mathrm{c} = \frac{k_\mathrm{B}T}{6\pi\eta_\mathrm{s}a} \tag{11.6}$$

where $k_\mathrm{B}T$ is the thermal energy, and η_s is the solvent viscosity measured in section 11.2.2. I use the method for determining D_c from dynamic light scattering (DLS) [26] described in [96]. The intensity-intensity autocorrelation function $g^{(2)}(\tau)$ is defined as:

$$g^{(2)}(\tau) \equiv \frac{\langle I(t)I(t+\tau)\rangle}{\langle I(t)\rangle^2} \tag{11.7}$$

I collect $g^{(2)}(\tau)$ at different angles from a dilute suspension of colloid, then fit to a function derived in [96]:

$$g^{(2)}(\tau) = B + \beta\exp(-2\Gamma\tau)\left(1 + \frac{\mu_2}{2!}\tau^2 + \frac{\mu_3}{3!}\tau^3 \ldots\right)^2 \tag{11.8}$$

where Γ is the rate of exponential decay of the field correlation function $g^{(1)}(\tau)$:

$$g^{(1)}(\tau) \equiv \frac{\langle E(t)E^*(t+\tau)\rangle}{\langle E(t)E^*(t)\rangle} = \exp(-\Gamma\tau) \tag{11.9}$$

Examples fits (solid lines) of equation 11.8 to the intensity autocorrelation raw data (open symbols) at several scattering angles is shown in figure 11.1 for PMMA particle stock ASM87. From the scattering angle θ (measured by the goniometer in the light scattering apparatus [138]), the wavelength of light λ, and the refractive index n_s, the scattering wavevector q is defined as:

$$q = \frac{4\pi n_\mathrm{s}}{\lambda}\sin\left(\frac{\theta}{2}\right) \tag{11.10}$$

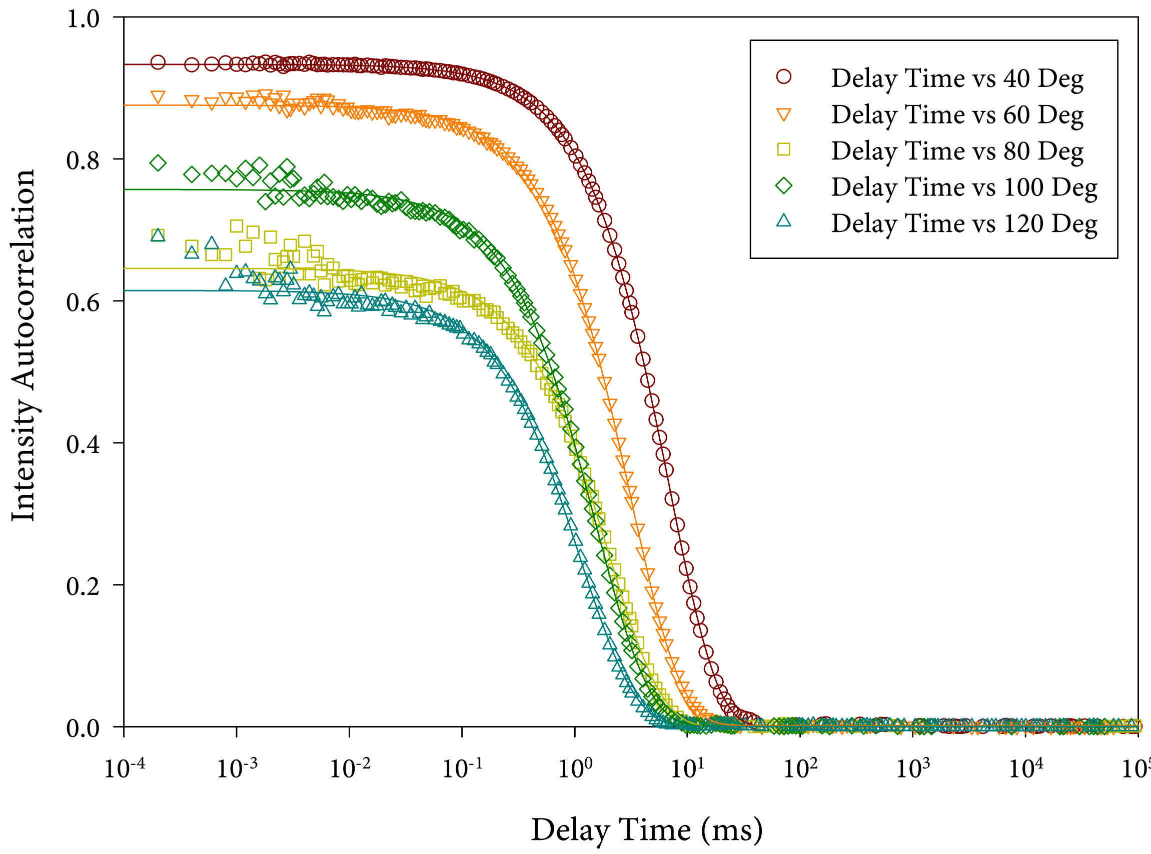

Figure 11.1: Example intensity autocorrelation $g^{(2)}(\tau)$ from dynamic light scattering, for data gathered at several angles from a dilute colloid suspension of PMMA spheres in a 45:55 cDHN:THN solvent mixture. Open symbols are raw data; solid lines are fits to the function defined in equation 11.8.

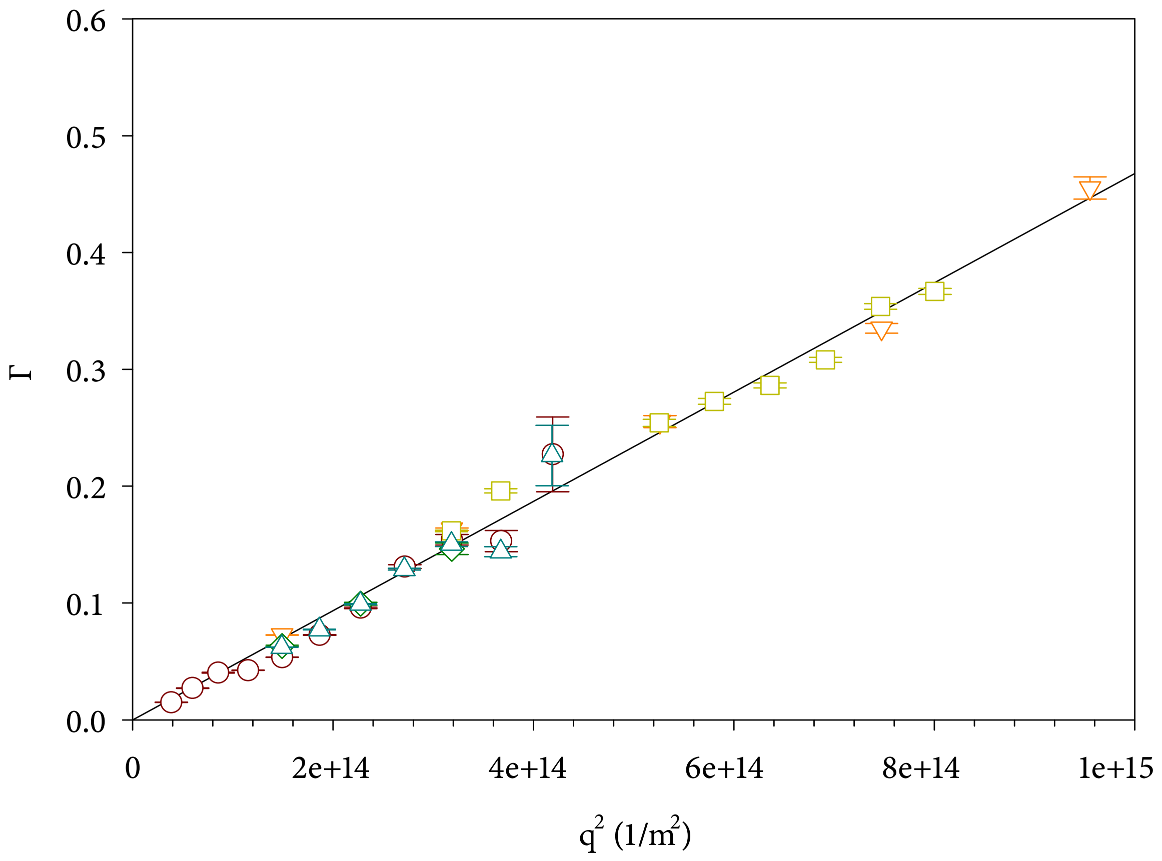

Figure 11.2: Plot of Γ vs. q^2 for all measurements (several repeated for the same angle). Raw data shown by colored symbols, with the error bars corresponding to the uncertainties in the fits of $g^{(2)}(\tau)$ to equation 11.8. The slope of the black solid best-fit line is the diffusion constant, $D_c = 4.68 \pm 0.03 \times 10^{-13}$ m^2/s.

The decay rate Γ is related to the square of the scattering wave vector q by the diffusion constant [96]:

$$\Gamma = D_c q^2 \tag{11.11}$$

From the fits, examples of which are shown in figure 11.1, a different value of Γ is associated with each angle, which can be expressed in terms of q^2. Plotting Γ vs. q^2 yields a linear relation, as shown in figure 11.2, whose slope is the diffusion constant D_c, as expected from equation 11.11. In this particular case,

$$D_{\mathrm{c}} = 4.68 \pm 0.03 \times 10^{-13}\ \mathrm{m}^2/\mathrm{s} \tag{11.12}$$

Combining this with the viscosity measurement (equation 11.3) and Stokes-Einstein relation (equation 11.6) yields the particle size:

$$a = 216 \pm 2\ \mathrm{nm} \tag{11.13}$$

11.2.5 Polymer Size

Exactly what defines the polymer size, and how to characterize it, is still somewhat a matter of debate within the field. Traditionally, the zero-concentration radius of gyration r_{g} has been used. This is usually determined with static light scattering (SLS) [138], where measuring the angle- and concentration-dependent light scattering intensities are extrapolated using a Zimm plot [225], for example as shown in figure 11.3 (for this figure, note that the data is presented for the polymer used in the gel samples in chapter 10, not the BCAT samples here). From this plot, the radius of gyration r_{g} is calculated to be 33.0 nm, though the range is not so important for small ξ on account of the Noro-Frenkel correspondence [188], as discussed in chapter 10.

By contrast, the positions of the phase boundaries are highly dependent on the exact size ratios in the approximate region we expect the BCAT samples to occupy. At the time of the writing of this thesis, however, we are still awaiting a measurement of the value of dn/dc for the 11.4 MDa polystyrene in cDHN:THN solvent that we used for the BCAT samples, which is needed to get an exact value for r_{g} from the SLS measurements.

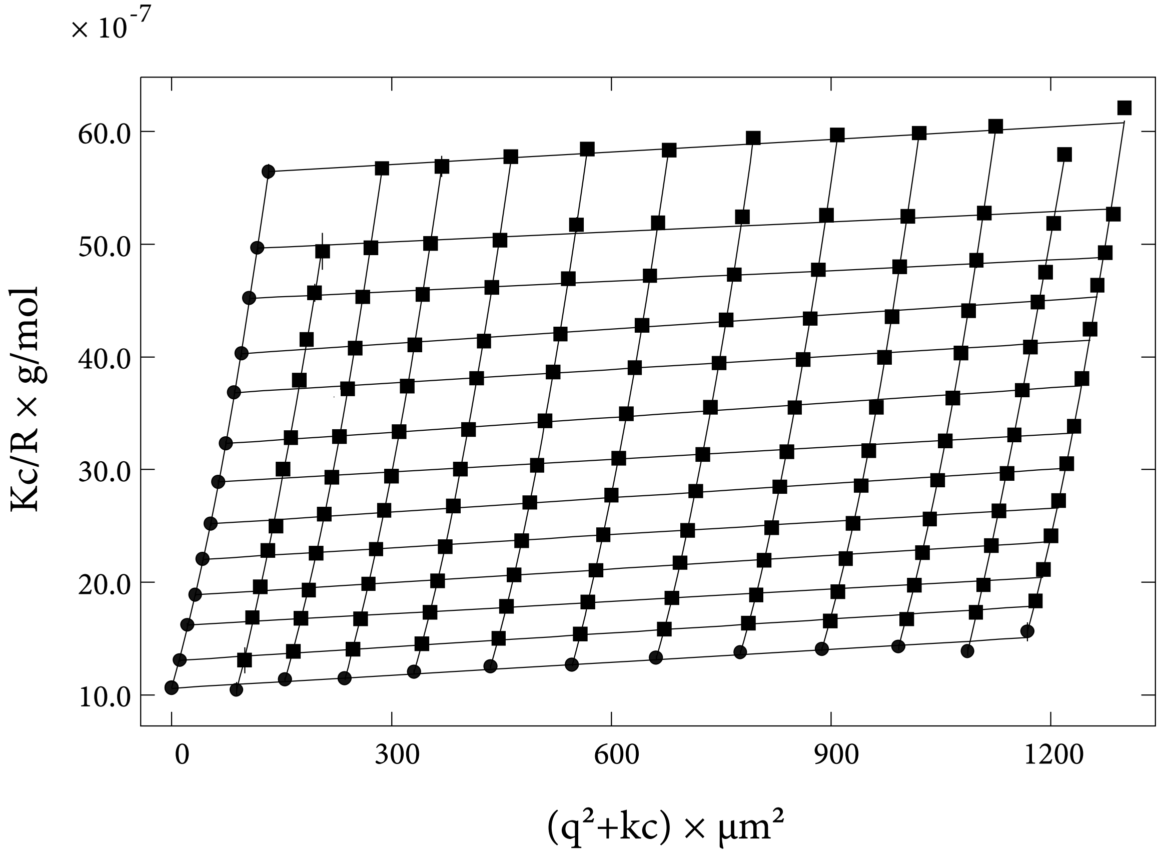

Figure 11.3: Zimm plot of a linear polystyrene with molecular weight 681 kDa, in a solvent mixture of 5:1 bromocyclohexane:decahydronaphthalene. Note that this is the polymer used for the gel samples in chapter 10, not the BCAT samples. From this plot, the radius of gyration r_g is calculated to be 33.0 nm.

11.2.6 Colloid Volume Fraction

A parameter of great interest is the difference in colloid volume fraction $\Delta\phi$ in coexisting phases *after* phase separation in colloid-polymer mixtures, for example colloidal liquid and gas. How the interfacial surface tension varies as a function of $\Delta\phi$ gives a measure of the critical exponent, and hence the universality class, of these mixtures [286, 287, 288, 168].

The average volume fraction (before phase separation) can be determined rather precisely by diluting down a close-packed colloid dispersion with a known quantity of solvent, with the assumption that colloidal particles pack during centrifugation at the random close-packed (RCP) volume fraction of $\phi_{\mathrm{RCP}} = 0.64$. The problem is how to determine ϕ after phase separation has occurred, without the aid of microscopy (determining ϕ from confocal microscope images of particles whose centers have been located is straightforward—just count the particles—as described in section 7.2). Although it might initially seem equally straightforward to measure the volume fraction of undyed particles whose size is far smaller than the wavelength of visible light (see equation 11.13), this turns out not to be the case.

At first, we tried a time-honored technique commonly used to measure the concentration of molecules in solution: absorption spectroscopy in a spectrophotometer. This instrument measures I_{a}, the intensity of an attenuated beam of light that has passed through a sample, normalized by I_0, the intensity of the same beam with no sample present. According to Beer's law [35]:

$$I_{\mathrm{a}} = I_0 e^{-\alpha_{\mathrm{a}} l_{\mathrm{p}}} \tag{11.14}$$

where l_{p} is the path length through the sample, and α_{a} is the absorption coefficient of the medium, which is directly proportional to the concentration of absorbing molecules in solution. The absorbance A_a is defined as:

$$A_\mathrm{a} \equiv \alpha_\mathrm{a} l_\mathrm{p} = -\log\left(\frac{I_\mathrm{a}}{I_0}\right) \tag{11.15}$$

The spectrophotometer measures the beam intensity both with and without the sample, and reports A_a, which shares the units with optical density. For equally-sized samples, A_a should be directly proportional to molecular concentration, and possibly for colloid samples, as well. Alas, this does not work at all. All of the samples with any colloid have a strong absorption for $\lambda < 500$ nm, comparable to the diameter of the particles, and there is no monotonic relationship between A_a and ϕ, as shown in figure 11.4. What the spectrophotometer may be measuring as this“absorption” could in fact be forward Mie scattering, since the length scales of particle and light are similar [35].

We therefore investigated other angles, in addition to the straight-through transmission in the spectrophotometer. Static light scattering in the ALV did not yield any clear ϕ-dependence over a range of angles. We also looked at 90° scattering with a fluorometer, exciting at 340 nm and looking at the integrated intensity over nearby wavelengths, 338 to 342 nm. The data is not monotonic, so that a given level of emission intensity can correspond to more than one volume fraction, as shown in figure 11.5. Because the values are non-monotonic in the range of $0.4 < \phi < 0.6$, of particular interest when looking at liquid-phase samples, we couldn’t use this to reliably determine volume fraction, either.

Instead, we looked at THN fluorescence of colloid suspensions at various ϕ in the fluorometer, exciting at 340 nm and looking at the integrated intensity in the range of 370 to 390 nm. This yielded a monotonic relationship between volume fraction and total emitted intensity—but the reasons why are not entirely clear. We suspect that the self-quenching or self-absorption of the THN (this fluorescence peak is not present in cDHN or

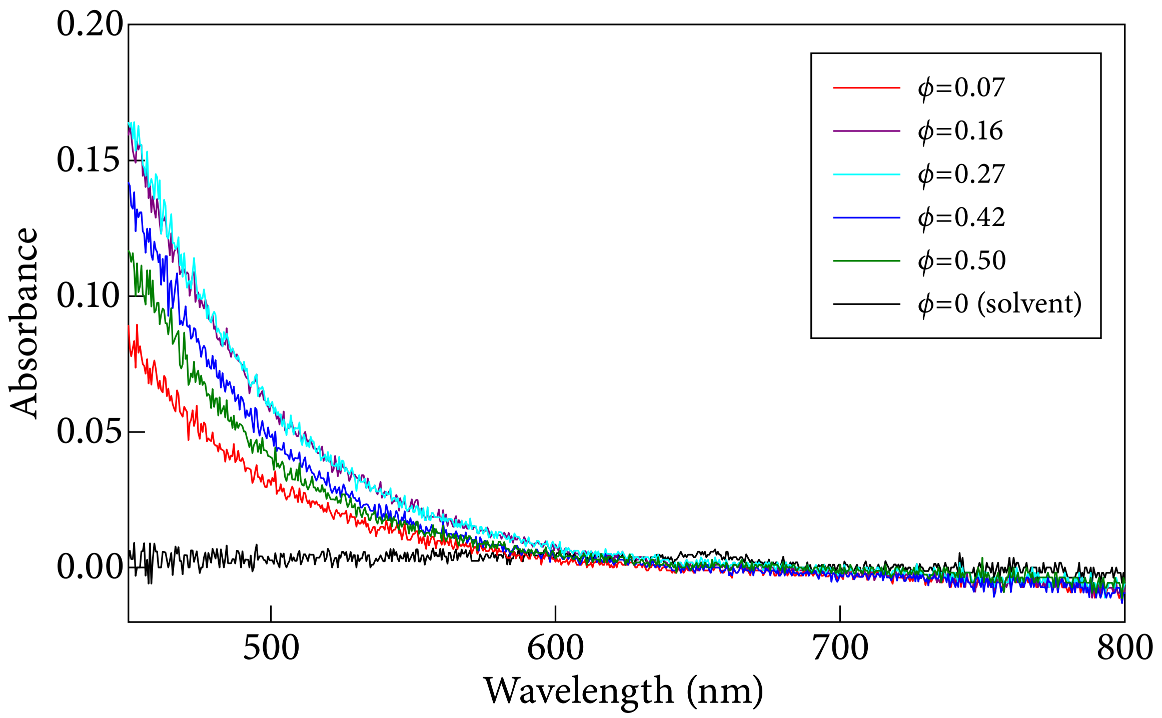

Figure 11.4: Absorption spectra for several colloidal volume fractions ϕ. There is strong absorption for $\lambda < 500$ nm, on the order of the size of the particles (equation 11.13), and no monotonic relationship between A_a and ϕ. The spectrophotometer may not measuring only absorption, but also may be affected by forward Mie scattering.

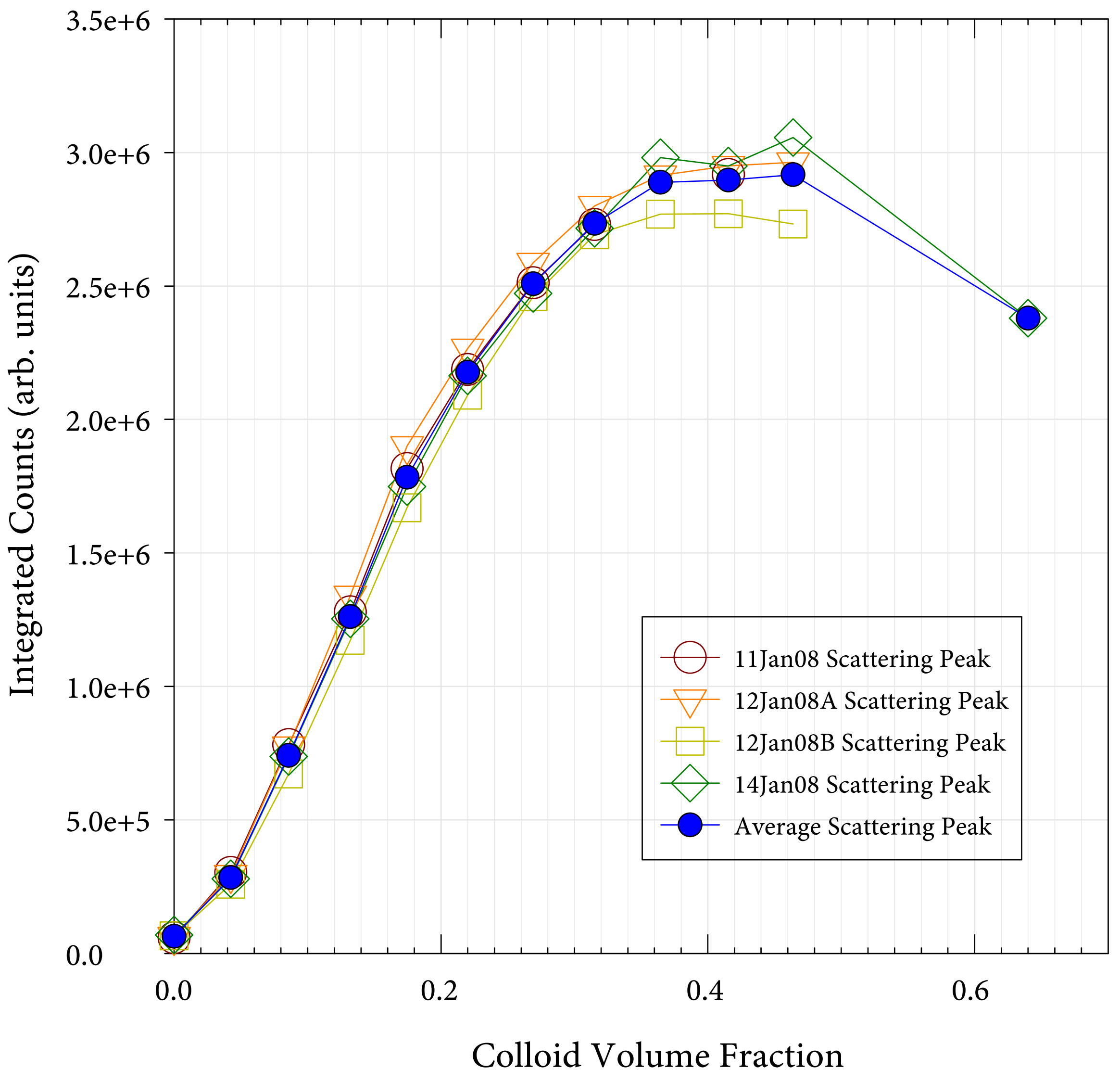

Figure 11.5: Fluorometer scattering measurement of colloid suspensions at different ϕ. Samples are illuminated in the fluorometer at 340 nm, and data is integrated over the band from 338 to 342 nm. The relationship between ϕ and emission intensity is not monotonic, so that a given level of emission intensity can correspond to more than one volume fraction. Because the values are non-monotonic in the range of $0.4 < \phi < 0.6$, of particular interest when looking at liquid-phase samples, we can't use this to reliably determine volume fraction. Open symbols are individual data runs; solid blue symbol and line represent the average for all samples.

in PMMA) is suppressed by the presence of more colloid, so that higher ϕ leads to a higher fluorescence intensity, as shown in figure 11.6. This allows a volume fraction determination with precision of a few percent.

11.2.7 Colloid-Polymer Mixtures

The BCAT colloid-polymer mixtures were prepared according to a sequence of steps, which except for the identity of the solvents and the lack of density-matching, is otherwise identical to the procedure used to make the gel, cluster and fluid samples for confocal imaging presented in chapters 4, 9 and 10.

1. The particle stock numbered ASM87 was chosen for its size, and known monodispersity (it has been observed to crystallize). The particles are delivered in dodecane, which was then washed several times in cis-decahydronaphthalene (cDHN). The typical washing procedure:

   (a) Centrifuge the particles until they are packed at the bottom of the vial, and the vial can be inverted without any mixing of the particles. Typically, a vial must be centrifuged at $3000g$ for several hours.

   (b) Decant the solvent, either by pouring or with a glass Pasteur pipette.

   (c) Invert the vial and let drain on a Kimwipe for several minutes, so that any excess solvent is wicked away from the mouth of the vial by capillary action.

   (d) Add fresh solvent, in a quantity approximately $5\times$ to $10\times$ the mass of the colloid, to the vial.

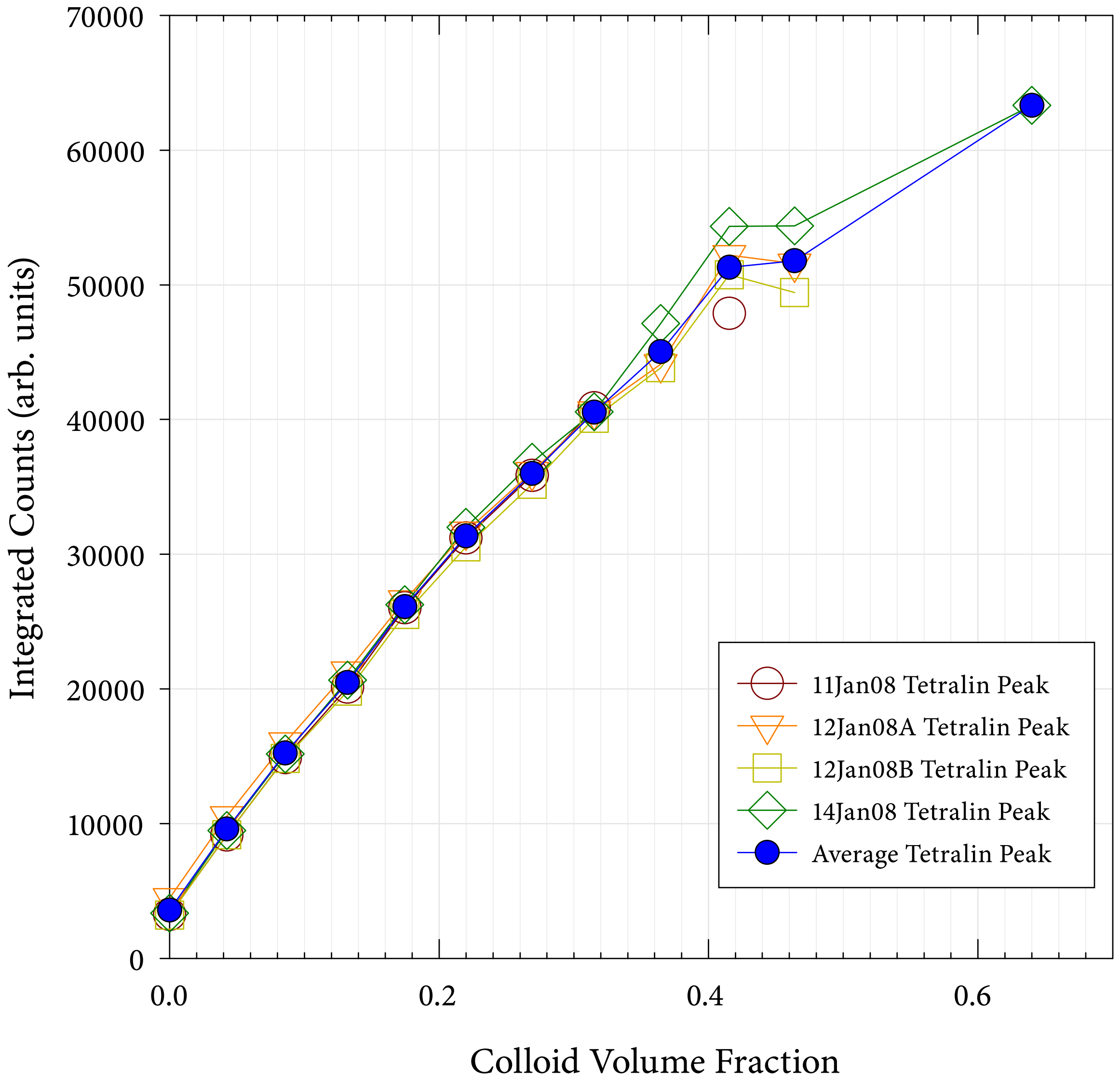

Figure 11.6: Fluorometer measurement of THN fluorescence in colloid suspensions at different ϕ. Samples are illuminated in the fluorometer at 340 nm, and data is integrated over the band from 370 to 390 nm. The relationships between ϕ and emission intensity is monotonic, and therefore potentially useful in measuring volume fractions with fluorescence. Open symbols are individual data runs; solid blue symbol and line represent the average for all samples.

(e) Vortex, vigorously as necessary, until all particles are suspended back in the solvent, and there are no macroscopic clumps. This typically takes half an hour at maximum speed for these small particles.

(f) Centrifuge the particles until they are packed at the bottom of the vial, and repeat the procedure.

Note that each washing step, if done properly, replaces about 90% of the solvent, so within a few washing cycles, any original solvent should be completely replaced.

2. After the dodecane has been washed out and replaced by decalin, an abundance of tetrahydronaphthalene (THN) is added (i.e. much more than the 55% dictated by the final mass ratio). The THN-rich solvent and particles are then heat-shocked by placing the vial in an oven at 80°C. This swells the particles a bit because THN is rich in conjugated bonds, and therefore a better solvent for PMMA, with its high density of aromatics, than the entirely-alkyl cDHN. The vial should turn clear in the oven, demonstrating an improvement to the index match. This will become more cloudy after the sample cools.

3. The next step is to centrifuge sample and decant the solvent, which due to selective absorption of THN now has an unknown ratio of THN to cDHN.

4. Correcting this problem, thereby suspending the heat-shocked particles into a solvent of known ratio, involves repeated washings with a stock solution, made separately, where the 55:45 THN:cDHN is typically correct to one part in 10^4.

5. After the multiple washings, the colloid is stable and suspended in an index-matching fluid. Specifically, after centrifugation, the close-packed particles, in the final solvent,

should be transparent and single-scattering through centimeters of suspension. If this is not the case, then something is wrong with the preparation.

6. After the final redispersal with vortexing, dry polymer is added to the vial with the particle suspension. The vial is then sealed and tumbled for at least 24 hours, with one 90° rotation about the long axis of the vial (perpendicular to the tumbler rotation axis) in the middle of the tumbling run. This ensures that all parts of the vial are evenly mixed.

7. Different samples are then made by diluting this colloid-polymer stock solution mixture with the same 55:45 solvent batch used earlier, guaranteeing a constant ratio of colloid to polymer mass.

Specific colloid-polymer ratios were chosen to approach closely to where previous experiments had suggested the critical point would be [31, 32]. Our experimental phase diagram of the samples launched as part of BCAT3 (Feb 2004) and BCAT4 (Mar 2008) is shown in figure 11.7. Here samples are shown in both experiments based on whether they phase separate on the ground—that is, if a sample left overnight shows an interface between colloidal liquid and gas. Because samples have a fixed ratio of colloid to polymer mass, all samples in the same dilution series fall on a straight line that passes through the origin.

The vertical axis in figure 11.7 is given in the experimental units of c_{p}, expressed in mg/mL. How to convert this to any more universal unit, e.g. $U/k_{\mathrm{B}}T$ is an open question. Moreover, the ϕ values reported are whole-sample averages before phase separation.

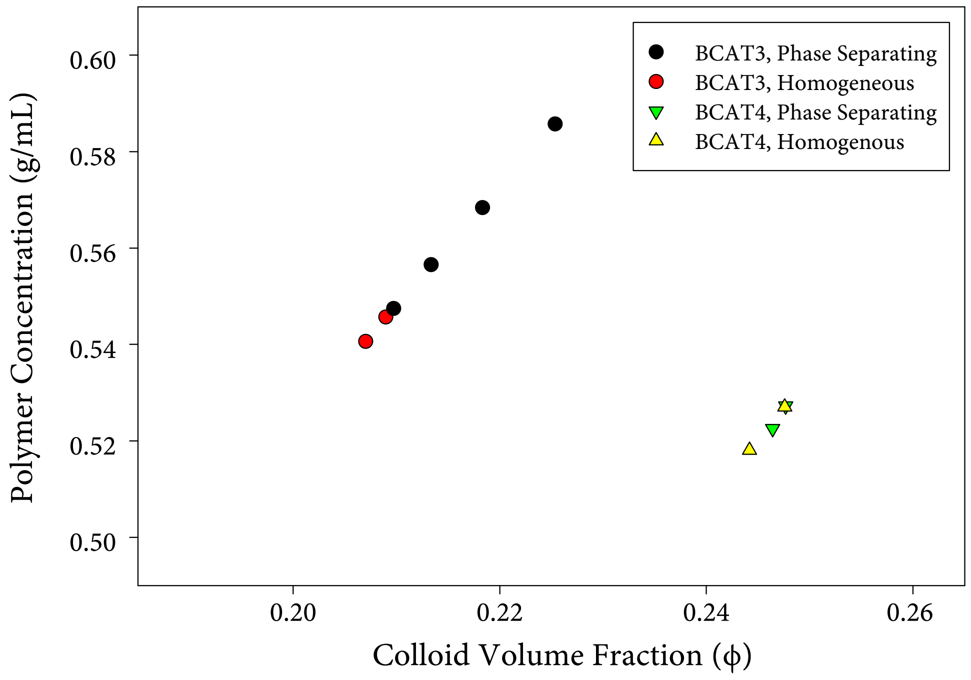

Figure 11.7: Experimental sample phase diagram of the samples launched as part of BCAT3 (Feb 2004) and BCAT4 (Mar 2008). Sample phase separation is as determined on the ground by whether an interface forms between two phases when samples are left to sit overnight.

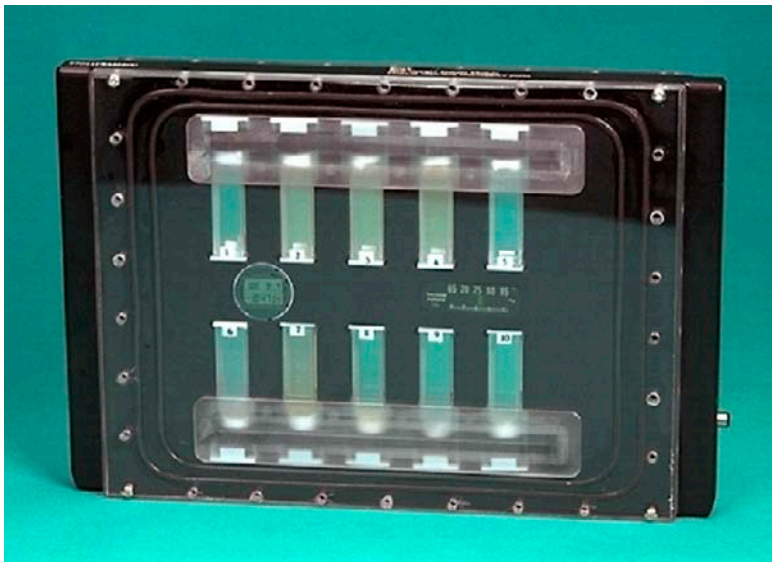

Figure 11.8: Sample holder for BCAT3 and BCAT4. Ten glass cuvettes hold all samples. The BCAT3 colloid-polymer samples described here occupy samples 1 through 6. Samples 1 through 5 fill the top row; sample 6 is the leftmost sample on the bottom row).

11.3 Photography aboard the International Space Station

Colloid-polymer mixtures were loaded and sealed into glass cuvettes (Hellma). Ten of these cuvettes were then loaded into a sample module, as shown in figure 11.8.

Two sample holders have been flown to the International Space Station on separate occasions. BCAT3 went up on a Progress module (in essence a Soyuz module without any people, so the whole capsule is available to send cargo) in 2004, while BCAT4 was launched in the payload of the Space Shuttle in 2008. The samples were then photographed in several ways.

The differences between what is observed in microgravity and on earth is dramatic. Two samples are illustrated in figure 11.9, one photographed several days after mixing on earth, and the other (a sample with similar composition) at a comparable time after mixing in microgravity aboard the ISS. The earth-bound sample has undergone complete

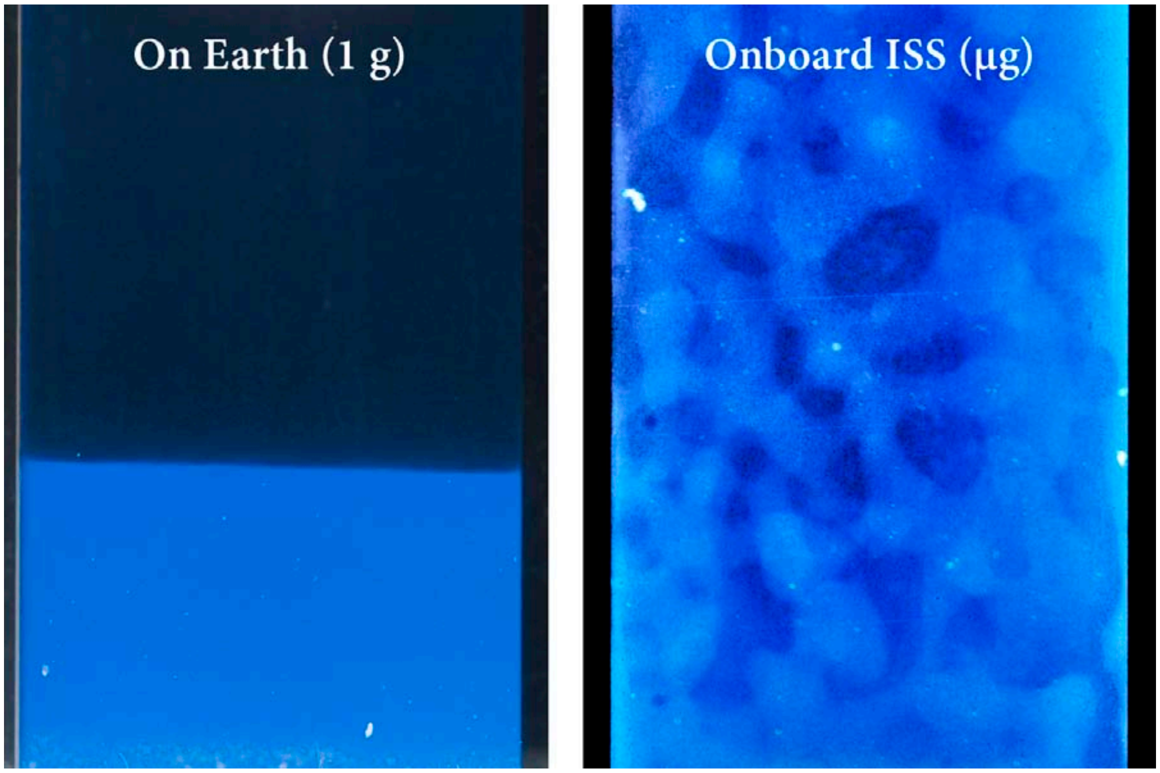

Figure 11.9: Comparison between a sample photographed several days after mixing on earth ($1g$), and a similar sample at a comparable time after mixing aboard the International Space Station in microgravity (μg). The earth-bound sample has undergone complete gravitational collapse, and shows no interesting structure whatsoever. By contrast, the sample in microgravity strikingly displays the structure of a bicontinuous network characterizing spinodal decomposition.

gravitational collapse, and shows no interesting structure whatsoever. By contrast, the sample in microgravity displays clear features of a bicontinuous network characterizing spinodal decomposition (discussed in chapter 3). So, clearly, there is little in this area that can be discerned by considering samples that are not carefully density-matched. Note that we have imaged only samples from BCAT3, so the following discussion does not include any images (which have not yet been taken) from BCAT4.

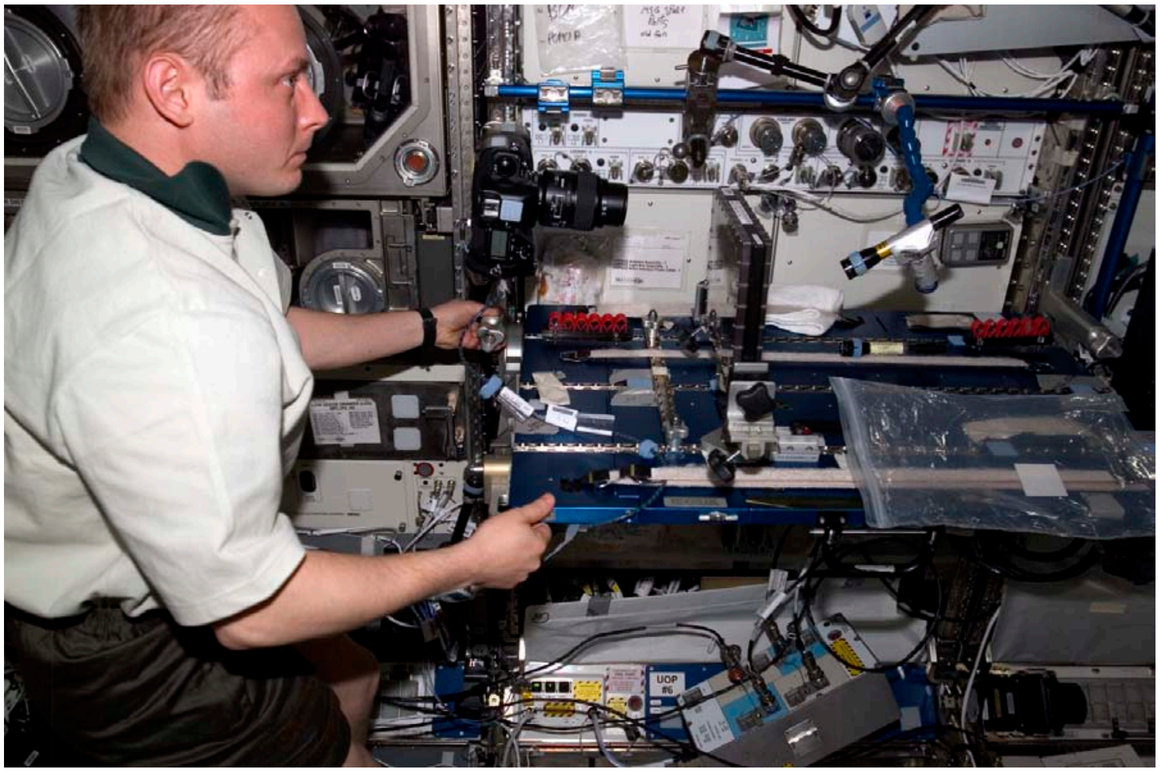

Figure 11.10: Astronaut Mike Fincke aboard the International Space Station, operating the BCAT3 experiment by taking photographs of the samples in their holder, illuminated with a flashlight from the rear.

11.3.1 Flashlight Illumination

In 2004, we began with a protocol whereby the astronauts unstowed the entire module, set it up on a temporary table called the Maintenance Work Area (MWA), and individually focused and photographed each sample, one at a time. Astronaut Mike Fincke is shown with the setup, comprising a camera (Kodak DCS760, based on a Nikon F5 body) with a macro lens (Nikon 105mm F2.8D), the sample holder, and flashlight behind, in figure 11.10.

This setup has the advantage of simplicity, since at the outset we didn't know what might be observed. Two astronauts, Mike Foale and Mike Fincke, were able to complete

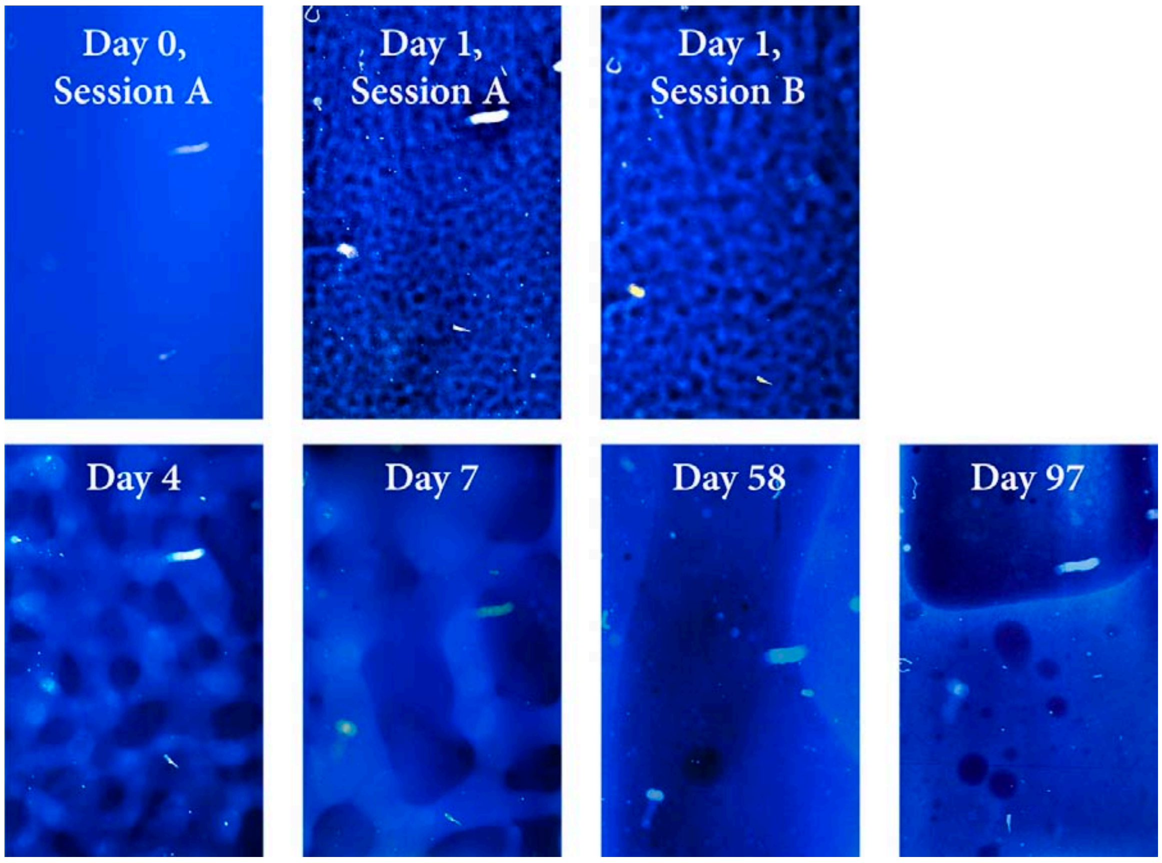

Figure 11.11: Sample 2, as photographed manually with flashlight illumination by astronauts Mike Foale and Mike Fincke, for several months in the spring and summer of 2004. The sample starts out homogeneous right after being mixed, then demixes (phase separates) over time. A bicontinuous network forms, which then subsequently coarsens until complete phase separation is achieved by Day 97: a bubble of the colloidal gas phase almost entirely surrounded by the colloidal liquid (which preferentially wets the cuvette walls).

a sequence of images over the course of several months, for most of the samples. The sequence for Sample 2 is shown in figure 11.11.

Note that the sample starts out homogeneous right after being mixed, then demixes (phase separates) over time. A bicontinuous network forms, which then subsequently coarsens until complete phase separation is achieved by Day 97: a bubble of the colloidal gas phase almost entirely surrounded by the colloidal liquid (which preferentially wets the

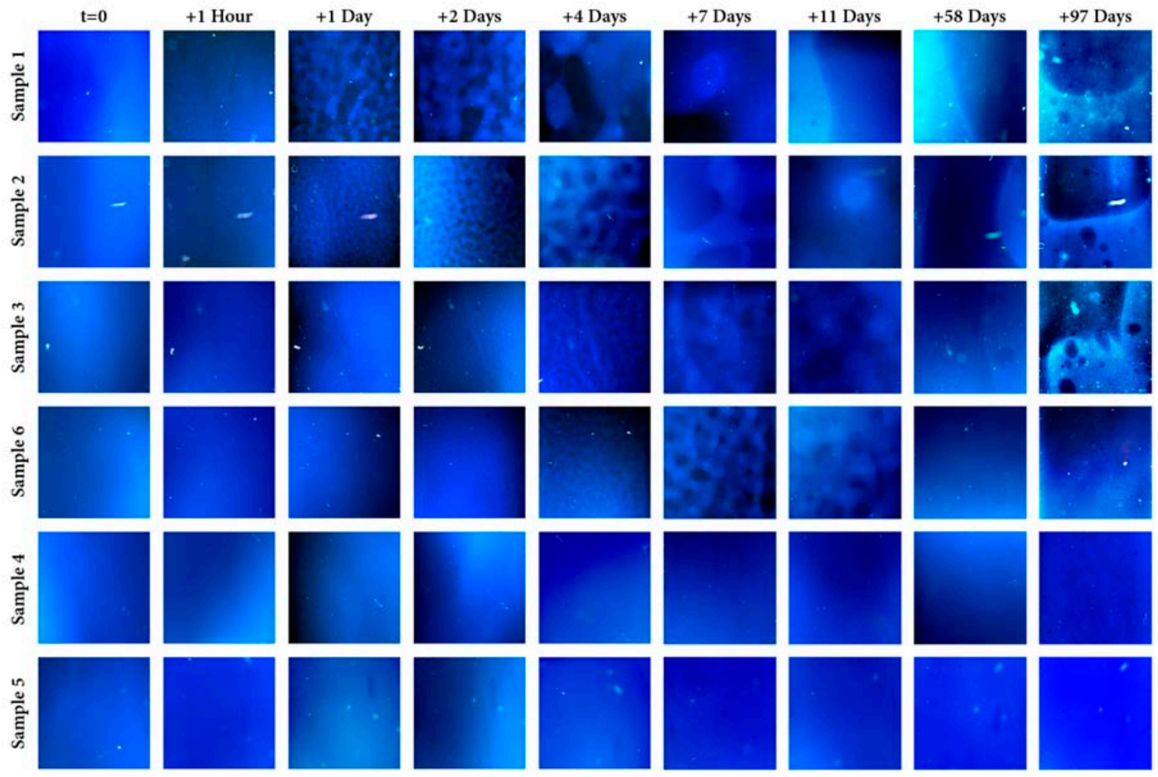

Figure 11.12: Matrix of individual sample images, all taken manually by Mike Foale and Mike Fincke in 2004, with flashlight illumination. As polymer and colloid concentration decrease (moving toward the bottom of the figure), the rate of phase separation slows. That is, at any given time, the length scale is smaller going down the page.

walls).

Similar photographs, presented for all six samples in BCAT3, taken by Mike Foale and Mike Fincke, are presented in figure 11.12. As polymer and colloid concentration decrease (moving toward the bottom of the figure), the rate of phase separation slows. That is, at any given time, the length scale is smaller going down the page.

The final state of the samples, captured about three months after the first homogenization, is presented in figure 11.13. Complete phase separation is clear and obvious in samples 1, 2 and 3. Sample 6 has phase separated, but because it is on the bottom row of the sample holder, the cuvette is inverted, and the phase separation boundary is not visible in the area shown. In samples 4 and 5, no clear boundary is visible.

The main drawback to this manual approach to photography is its extensive use of crew time. Because astronauts need to spend time setting up and later putting away the camera and samples each time individual sample photos are taken, it becomes prohibitive to get more than a few images in this fashion.

11.3.2 Camera Flash Illumination and EarthKAM

In response, we altered the image acquisition protocol to control the camera in an automated fashion with EarthKAM, a software application meant to remotely trigger the Nikon DCS 760 camera over its firewire connection. EarthKAM (**Earth K**nowledge **A**cquired by **M**iddle school students) is a Windows-based program running on the laptops aboard ISS, that inputs a list of times to trigger the camera, and automatically downloads the resulting images from the camera onto the laptop's hard drive. Because the laptop is connected to one of the networks aboard ISS, the EarthKAM software can be controlled from the ground and images downloaded without astronaut intervention.

This has revolutionized our data acquisition. The astronaut sets up the camera, and the time sequence of images is uploaded from the ground (e.g. to take a picture every hour for several weeks); the camera setup can then be left to run undisturbed. Images are periodically (typically daily) sent down to Johnson Space Center (JSC) in Houston, where they can be viewed after approximately a 1-day delay, and modifications to the time sequence uploaded, if necessary. Astronauts periodically check to make sure the set up is operating properly, but otherwise do not need to intervene. From a crew utilization standpoint, this is a significant improvement—the total time to set up and stow away are

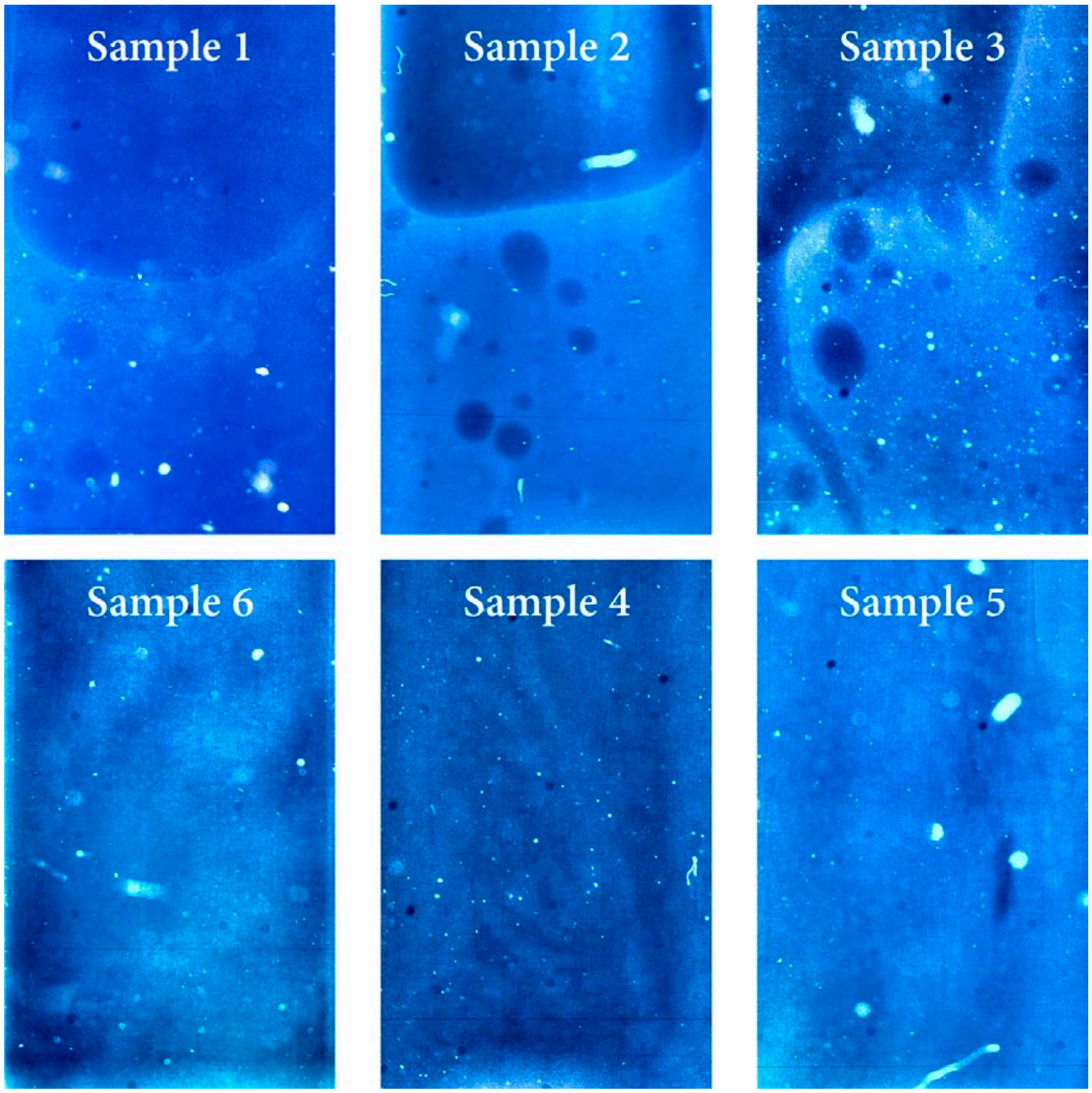

Figure 11.13: BCAT3 samples illuminated by flashlight and photographed manually about three months after homogenization. Complete phase separation is clear and obvious in samples 1, 2 and 3. Sample 6 has phase separated, but because it is on the bottom row of the sample holder, the cuvette is inverted, and the phase separation boundary is not visible in the area shown. In samples 4 and 5, no clear boundary is visible.

each fixed at approximately an hour—but instead of getting just a few pictures for that usage of crew time, hundreds can be acquired in an automated fashion, and examined on the ground shortly after they have been taken. The new setup requires one physical difference: illuminating with a flash, so that light is only emitted when a picture is being taken. This contrasts with the flashlight setup, where the continuously operating bulb would drain the batteries in a matter of hours, precluding unattended acquisition over multiple weeks.

Astronauts have made several other significant improvements to the setup. As suggested by Mike Fincke and Bill McArthur, the sample holder, camera and flash were clamped directly to the walls, without the MWA, which can then be free for other use (and therefore its use for other purposes does not interfere with the scheduling of the setup). This was perfected by Dan Tani, whose set up is shown in figure 11.14; he has been able to acquire continuous sequences of several hundred images, with minimal movement of the camera between frames.

With EarthKAM, we were able to acquire several hundred frames for each of the samples, which gives a much better picture of sample evolution, improving upon the techniques optimized for flashlight-illuminated photography.

11.4 Image Processing

Acquired images are large, high-resolution files (approximately 6 megapixels), but lighting, position and rotation of the sample within the frame vary significantly between photos—and not in a predictable, continuous or monotonic way. The camera tends to move over the course of weeks, in an essentially random, almost diffusion-like manner. So a relatively complex set of steps was required to:

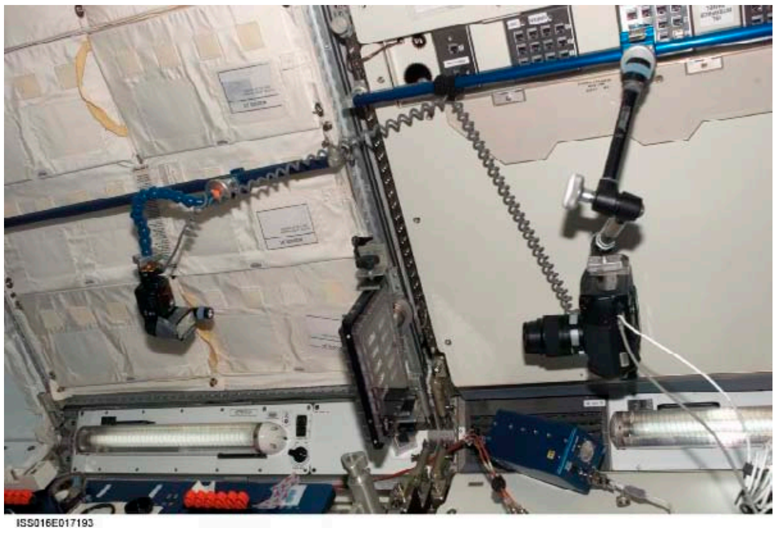

Figure 11.14: Image of the most recent EarthKAM-driven BCAT setup by Dan Tani, demonstrating a number of improvements over the MWA-based manual photography setup. The set up is clamped to the ceiling, out of the way and less prone to vibration. The illumination is now provided by a camera flash, and the camera is now tethered to, and controlled by, a laptop running the EarthKAM software.

- Maximize information recovered from the camera CCD.
- Stabilize the image sequence, so that each frame shows the same part of the sample, to enable direct comparison over time.
- Remove lighting inhomogeneities and uneven backgrounds.
- Extract specks of dust in an automated fashion.
- Maximize contrast and output a final image for subsequent analysis.

I used a combination of Adobe After Effects CS3, and Adobe Photoshop CS3, both part of the Adobe Creative Suite CS3 Master Collection.

11.4.1 Camera RAW conversion

The images captured by the Kodak DCS760 camera are downloaded in their raw, uncompressed format, with a file extension of `.DCR`. These files belong to a class of file formats called Camera Raw, which contain the raw CCD sensor readouts for each pixel, and various metadata (camera settings, date, time, etc.). Because the data stored is raw sensor output, the intensity levels are linear (i.e. not gamma encoded) [95]. A crucial component in maximally preserving image contrast is to do as much processing on the Camera Raw image (e.g. with the Adobe Camera Raw input filter) before importing it into an image editor like Photoshop, where all future operations are destructive. A large number of parameters can be tweaked, and the values are different for each sample, arising from differences in contrast between the two separating phases that result in different values of $\Delta\phi$. An example of an original (color) image converted directly from Camera Raw without changing any settings, is shown in figure 11.15a; the result of adjusting the settings in Camera Raw is shown in figure 11.15b. Camera Raw is designed to be automated, so once the settings are established on one image, it can be applied immediately to the several hundred representing the same sample at different times.

11.4.2 Stabilizing the image sequence

The resulting image sequence output from Camera Raw reveals a lot of random camera movement. This movement prevents a uniform way to analyze the same parts of subsequent images: cropping based on pixel coordinates, for instance, does not work. The image sequence must be stabilized, which requires software that can look at multiple images in the same sequence. For this, I turned to Adobe After Effects, a bit less familiar to most

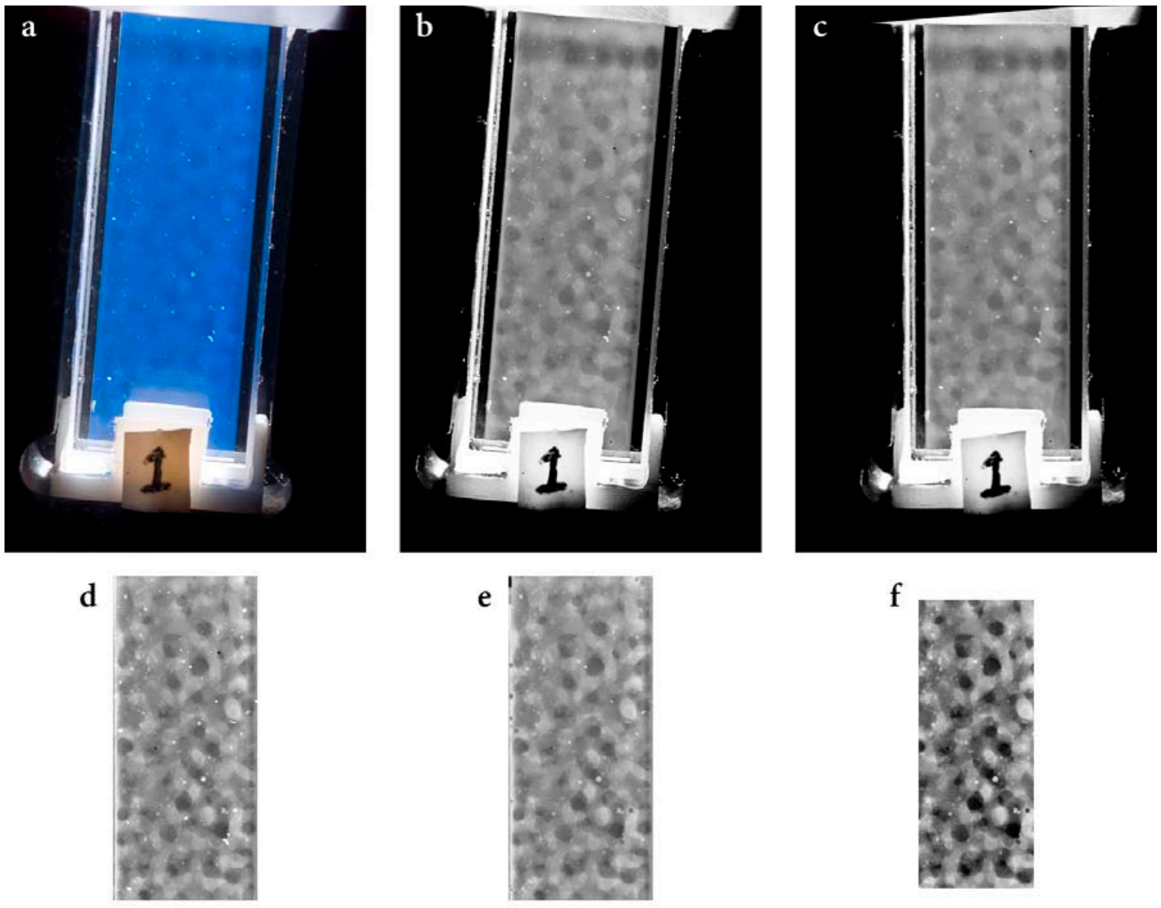

Figure 11.15: Demonstration of stages in the automated image processing of photographs from the EarthKAM-operated BCAT3 experiment. **(a)** Original image, as shot, with no processing. Note that the blue pattern, while visible, does not have a lot of contrast. **(b)** Image after conversion from Camera Raw, with higher contrast in the pattern. **(c)** Rotated image where the pattern is roughly centered in the frame. **(d)** Cropped image, where the uneven lighting background has been removed. **(e)** Image after automated dust removal. **(f)** Final image, after further cropping and final contrast enhancement.

than Photoshop, but crucial for this and subsequent steps [62].

One of the techniques used in special effects and compositing is tracking a moving object in subsequent frames of a movie or video. After Effects has a built-in tracking tool, which I use in stabilization mode; this tool automatically tracks the same feature from frame to frame, and then transforms each frame to use this feature as a new pixel coordinate origin. By tracking a pair of points—I looked at opposite corners of the bright-looking sample holder, which has high contrast in all frames—the stabilized images can all be rotated and scaled, as well [62]. This was among the most difficult steps to automate (before I used After Effects), and without automation, each of hundreds of frames would have had to have been rotated, scaled and moved, by hand. By applying several rounds of stabilization, subpixel registration of images can be achieved, allowing a much clearer view of the motion due to phase separation inside the sample. The rotated example image in figure 11.15c is one of the full sequence of images, all of which are properly scaled and rotated.

11.4.3 Background Removal

Each image is then cropped to preserve only the center section, where the sample within the holder is visible, and the uneven background, due to the lighting of the sample with a flash (with uneven lighting profile) at a high angle, is removed. This is accomplished in After Effects using a set of steps that mimics the Photoshop high-pass filter, but again showing the utility of the After Effects approach. Each image is gaussian blurred, using a radius that increases for later frames. This permits later images, with larger feature sizes (because the phase separation has progressed) to be blurred at a greater length scale, so that the background subtraction can be appropriately sized for every frame. This is simply accomplished in After Effects by keying the value of the gaussian blur radius at the beginning and the end of the sequence. This blurred image is then inverted. The original image is placed on a second layer, with the blending mode set to overlay, and opacity set to 50%. The overlay blending mode combines the functions of the screen and multiply modes, using the screen mode to enhance the contrast of pixels lighter than 50% brightness, and the multiply mode to enhance the contrast of pixels darker than 50% brightness [62]. The cropped and background-removed image is shown in figure 11.15d.

11.4.4 Automated Dust Removal

Both black and white specks are visible in the colored photo in figure 11.15a, and clearly not part of the sample. The black specks probably come from dust on the CCD imager inside the camera, since they remain fixed for the entire set of images, while the sample moves. The bright white specks move with the sample, and therefore are probably dust trapped outside the glass but within the cuvette holder. Both white and black specks must be removed, as they interfere with subsequent analysis if left in place.

Because the dust is at the extreme ranges of the pixel intensities in the images, I used After Effects to select these pixels with the Color Key filter. This knocks out those pixels (i.e. setting them to zero opacity), so that the resulting image is a bit like Swiss cheese. Then, I made several copies of this image on separate layers within an After Effect composition, but offset them by a few tens of pixels, so that regions with a hole in one layer would show through to the one beneath, showing parts of the sample. Because these holes are small (just a few pixels across), I chose this technique because it simply filled in places where there was no image information (due to the dust blocking the sample in the photo) with pixels of the same average intensity distribution. The end result is that all of the black and white pixels are gone, and there are grey dots of average intensity in their place, as shown in figure 11.15e. Because this step is implemented with layers and filters, it can be applied automatically to all frames in a sequence without user intervention; that is, manual dust removal is not needed.

11.4.5 Contrast Enhancement and Final Output

The final step is a contrast enhancement to maximize the dynamic range of the image, as shown in figure 11.15f. Note that the After Effects project was set to 16-bit greyscale, and all of these steps are performed as nested (or precomposed) projects, so that the only images generated are the final sequence, as in figure 11.15f. This allows a lot of changes and fiddling in a completely nondestructive manner, without accumulating data loss over the many intermediate steps of the processing protocol. When the final project is rendered, the entire sequence of hundreds of images (represented by the single image in figure 11.15f) is processed in a matter of a few minutes. I then used Photoshop to batch-convert the images to the losslessly-compressed 8-bit PNG format, for convenience and to minimize disk storage required [220].

11.5 Autocorrelation Calculation

Once the images of the sample undergoing phase separation are processed, we want to extract a characteristic length scale (discussed in chapter 3), to quantify the sample evolution as a function of time. This follows previous characterizations of spinodal decomposition with scattering and imaging [18].

Considering each processed image as an intensity distribution of pixels $I(x, y)$, where x and y are integer pixel coordinates, we calculate the 2D intensity-intensity autocorrelation $C(x_0, y_0)$, explicitly defined as:

$$C(x_0, y_0) = \frac{\sum_{x,y} I(x, y) I(x - x_0, y - y_0)}{\sum_{x,y} I^2(x, y)} \tag{11.16}$$

For each image, a copy is made and offset by (x_0, y_0). The overlapping portions of the two images are then multiplied, and that product is then divided by the square of the integrated pixel intensity in the overlapping area. When the offset is zero, i.e. $(x_0 = 0, y_0 = 0)$, the overlap is complete and exactly equal to (obviously) the square of the intensities in the overlap region, so that $C(0, 0) = 1$. The function decays as the offset is increased, until eventually, at infinite distances, $C(x_0 \to \infty, y_0 \to \infty) = 0$. At intermediate distances, a peak (or a ridge in 2D) will form at distances characterizing how far, on average, features in the images are separated. This represents the characteristic lengthscale in the image, and is the quantity we seek to calculate.

The code to do this incorporates a number of performance enhancements over the simple, direct multiplication. First, each 8-bit greyscale image is loaded into a 2D array of 8-bit integers using the FreeImage library, as in section 5.6; the 2D integer array is then read into a 1D floating point array using a short function that converts the 2D coordinates to a 1D index. This allows greater compiler optimization for faster memory access by placing pixels near each other in the image into nearby locations in memory.

The multiplication between an image and its copy for every offset from the minimum at $(x_0 = 0, y_0 = 0)$ to the command-line-set maximum at $(x_0 = x_{\max}, y_0 = y_{\max})$ is performed for all overlapping pixels, in a set of several nested loops. The outermost loop is over these offsets, in which several constants are defined:

```
for(int y_offset=-max_offset;y_offset<=0;y_offset++) {
    int num_nonzero_pixels = 0, x_min = 0, y_min = 0,
        x_max = 0, y_max = 0, i = 0, j = 0;
    float temp_sum;
    float *mult_result_sum = new float[max_offset*2+1];
    for (i=0; i<max_offset*2+1; i++) {
        mult_result_sum [i] = 0;
    }
    y_min = max(0,y_offset);
    y_max = min(image_length,image_length+y_offset);
    . . .
}
```

Next within this outer loop over all offsets, the inner loops over i and j are for image pixels within the range of the offsets. The loop structure is set up to maximize cache hits, as nearby addresses in memory are all loaded into the cache at the same time. The multiplication itself is then vectorized by the compiler [27], so that the code actually performs several parallel multiplications during each clock cycle.

```
int row_index = 0, offset_row_index = 0;
for(j=y_min;j<y_max;j++) {
    row_index = j*image_width;
    offset_row_index = (j-y_offset) * image_width;
    for(int x_offset = -max_offset;
        x_offset <= max_offset; x_offset++) {
        x_min = max(0,x_offset);
        x_max = min(image_width,image_width+x_offset);
        temp_sum =
            mult_result_sum[x_offset + max_offset];
        for (i=x_min;i<x_max;i++){
            temp_sum += (image_data[i+row_index]) *
                (image_data[i - x_offset
                    + offset_row_index]);
        }
        mult_result_sum[x_offset+max_offset]=temp_sum;
    }
}
for (int x_offset = -max_offset;
    x_offset <= max_offset; x_offset++) {
    x_min = max(0,x_offset);
    x_max = min(image_width,image_width+x_offset);
    //assign data to correlation matrix,
    //but check for nonzero number of pixels
    num_nonzero_pixels = (y_max-y_min) * (x_max-x_min);
    //Calculate Correlation for offset = x_offset
    if(num_nonzero_pixels == 0) {
        correl_matrix[(x_offset + max_offset) +
            (y_offset + max_offset) * offset_size] = 0;
    } else {
        correl_matrix[(x_offset + max_offset) +
            (y_offset + max_offset) * offset_size]
            = mult_result_sum[x_offset+max_offset]
            / num_nonzero_pixels;
        if(x_offset == 0 && y_offset== 0) {
            norm_constant = mult_result_sum[x_offset +
                max_offset] / num_nonzero_pixels;
        }
    }
}
```

Combining these optimizations allows the code to run approximately an order of magnitude faster than the standard, naive direct multiplication implemented in C (which itself is an order of magnitude faster than using MATLAB). Also, remembering that the correlation $C(x_0, y_0)$ is equivalent to $C(-x_0, -y_0)$ gives an additional factor of two improvement in calculation speed. The resulting 2D autocorrelation for the image in figure 11.15f is shown in figure 11.16.

This 2D autocorrelation is then azimuthally-averaged, to give a 1D correlation $C(r)$ as a function of $r \equiv \sqrt{x^2 + y^2}$, similar to that defined in [18].

```
    //integer bin number in 1-D array
    int bin_number = 0;
```

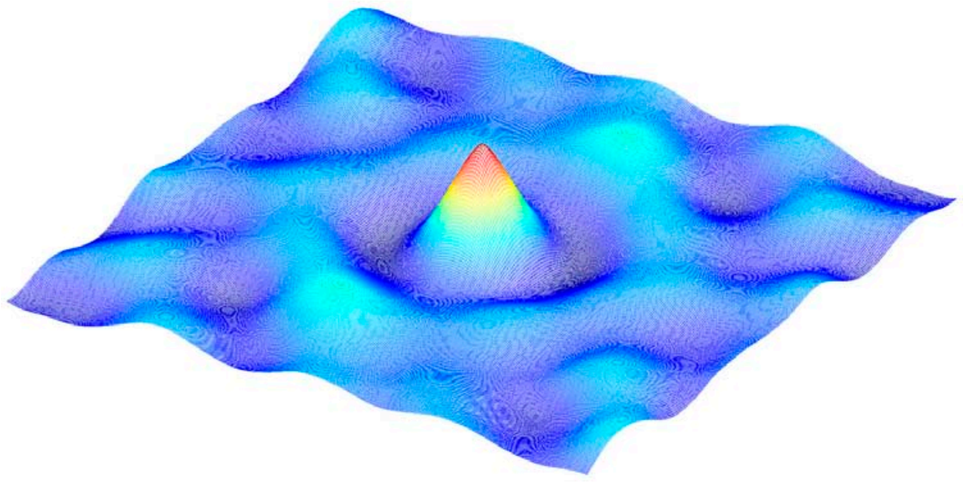

Figure 11.16: Two-dimensional intensity autocorrelation function $C(x, y)$, for the image of Sample 1 shown in figure 11.15f.

```
float *integrated_intensity = new float[offset_size];
int *pixel_counts = new int[offset_size];
int z=0;
for (z=0; z<offset_size; z++) {
    integrated_intensity[z] = 0;
    pixel_counts[z] = 0;
}
int x = 0, y = 0;
for(y = 0; y <= max_offset; y++) {
    for(x = 0; x < offset_size; x++) {
        int bin_number = sqrt( (x-center)*(x-center)
            + (y-center)*(y-center) ) + 0.5;
        integrated_intensity[bin_number]
            += correl_matrix[x + y * offset_size]
            /norm_constant;
        pixel_counts[bin_number]++;
    }
}
```

Finally, the 1D correlation function $C(r)$ is convolved with a gaussian kernel to reduce noise [49] on the length scale of a few pixels (far smaller than the typical observed characteristic length scales of tens to hundreds of pixels):

```
float *raw_1DCF = new float[max_offset];
float *smooth_1DCF = new float[max_offset];
//approximate gaussian kernel: e^-0.25 = 0.7788
const float kernel_offset1 = 0.7788;
//approximate gaussian kernel: e^-1 = 0.3679
const float kernel_offset2 = 0.3679;
for(z=0;z<max_offset;z++) {
    raw_1DCF[z]=integrated_intensity[z]/pixel_counts[z];
}
smooth_1DCF[0]= raw_1DCF[0];
smooth_1DCF[1]= raw_1DCF[1];
smooth_1DCF[max_offset-2]= raw_1DCF[max_offset-2];
smooth_1DCF[max_offset-1]= raw_1DCF[max_offset-1];
for(z=2;z<max_offset-2;z++) {
    smooth_1DCF[z]=
    kernel_offset2 * raw_1DCF[z-2] +
    kernel_offset2 * raw_1DCF[z+2] +
    kernel_offset1 * raw_1DCF[z-1] +
    kernel_offset1 * raw_1DCF[z+1] +
    raw_1DCF[z];
    smooth_1DCF[z] /= 1 +
        2 *(kernel_offset1 + kernel_offset2);
}
```

The final, smoothed $C(r)$ is shown in figure 11.17.

Next, the first minimum is located at r_{min}, and then the next maximum r_{max} after that, as a function of r. Note that the algorithm searches for r_{max} only for $r \geq 1.2 r_{\text{min}}$:

```
int zmin=2;
while(smooth_1DCF[zmin+1] <= smooth_1DCF[zmin]) {
    zmin++;
}
int zmax = zmin * 1.2;
while(smooth_1DCF[zmax+1] >= smooth_1DCF[zmax]) {
    zmax++;
}
```

Nearby pixels around r_{max}, within a radius of $\frac{1}{2.2}(r_{\text{max}} - r_{\text{min}})$ pixels, are averaged to estimate, with sub-pixel resolution, the actual position of the peak maximum at r'_{max}, which

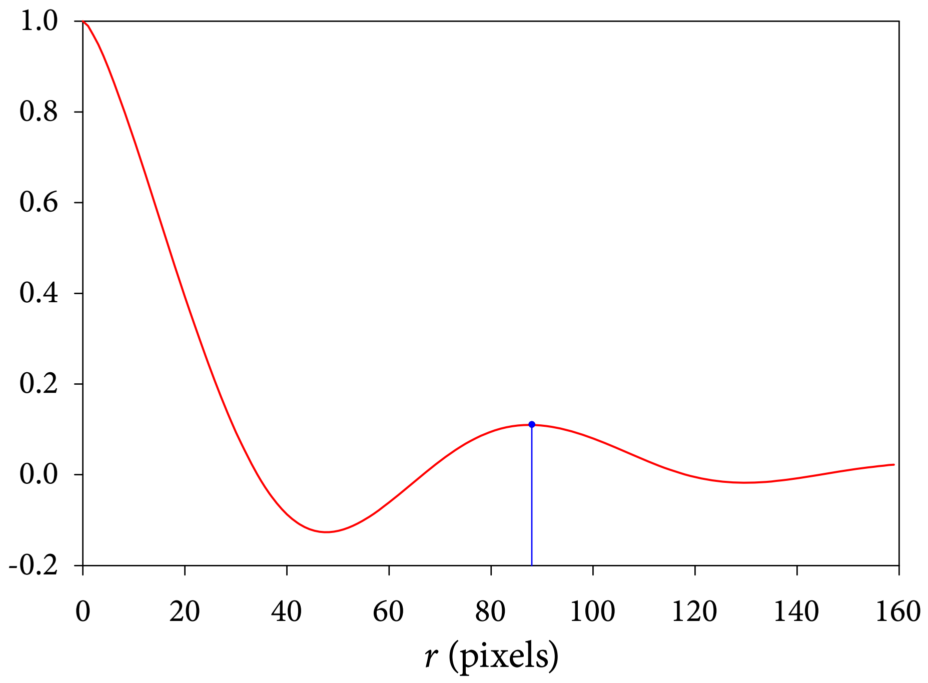

Figure 11.17: One-dimensional intensity autocorrelation function $C(r)$, calculated by azimuthally averaging the 2D correlation function shown in figure 11.16. As expected, the function decays from $C(r = 0) = 1$ to a trough, and then rises to a secondary local maximum peak at around $r_{\text{max}} = 88$ pixels, indicated by the blue dot. This is the characteristic length scale for the image.

defines the characteristic length scale of the image:

```
float zmax_fit = 0, zmin_fit=0;
int averaging_radius = (zmax - zmin) / 2.2;
float weighted_avg = 0, weight_sum = 0;
for(z=zmax-averaging_radius;
    z<=zmax+averaging_radius;z++) {
    if(z<max_offset){
        weighted_avg += z * smooth_1DCF[z];
        weight_sum += smooth_1DCF[z];
        }
    }
zmax_fit = weighted_avg / weight_sum;
```

Determining $r'_{\max}$ from this final 1D function is quite precise and relatively straightforward. However, one might imagine that, instead of autocorrelation, a 2D fast-fourier transform (FFT) would be a better choice. In that case, though, because the result is given in q-space, the peak occurs at $q'_{\max} \equiv 2\pi L/r'_{\max}$ pixels. That means that calculating a characteristic length for large $r'_{\max}$ involves fitting to a peak very close to the origin, and this is not as accurate as doing the multiplication in real space.

11.6 Results and Discussion

Once the characteristic length scale $r'_{\max}$ is calculated for each image, this can be plotted as a function of time for each sample, using the timestamp embedded as metadata in each image. The original data for $r'_{\max}(t)$, with half a dozen photos for each sample (using the flashlight illumination), is shown with solid symbols in 11.18. In general, data points for each sample fall approximately upon a straight line, as expected [257], which is marked by the corresponding colored dashed lines in the figure.

Repeating the data acquisition with EarthKAM increases the number of data points

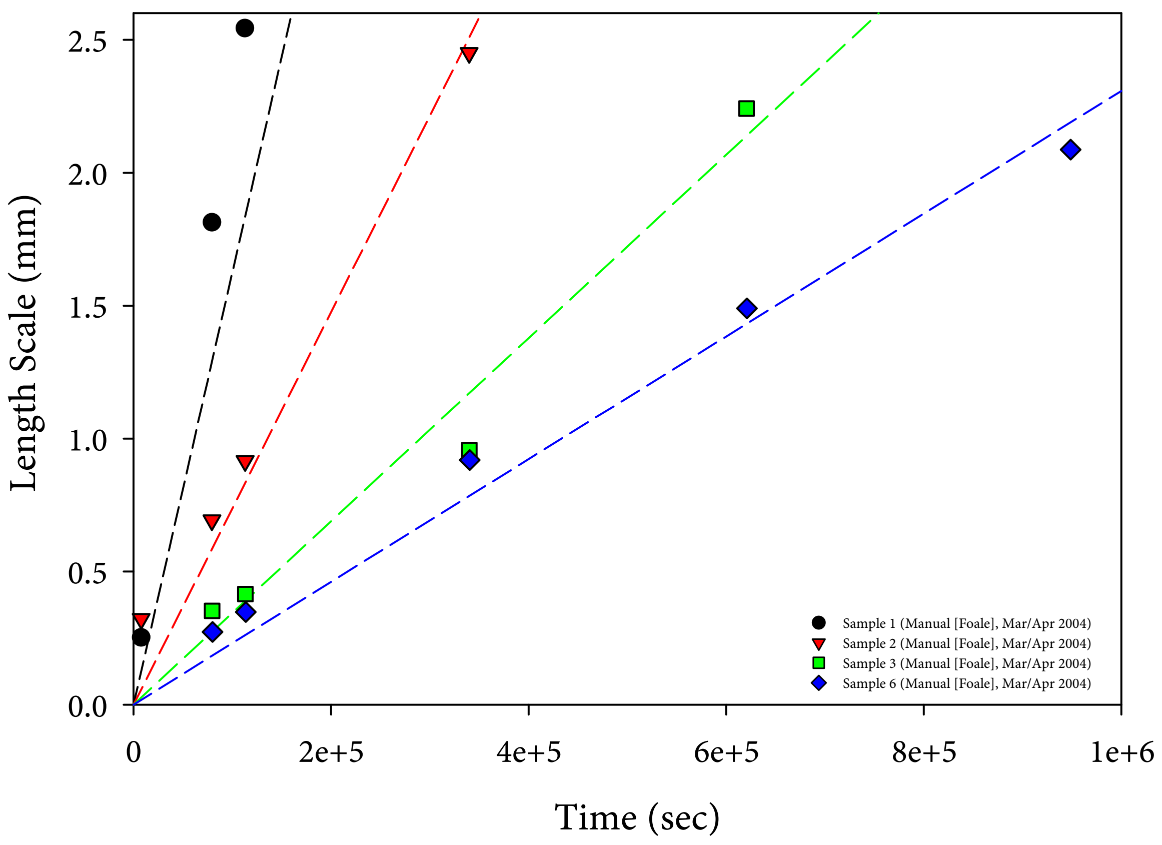

Figure 11.18: Results of the BCAT3 autocorrelation analysis, plotted with solid symbols for samples photographed one-at-a-time with the flashlight. Best-fit dashed lines giving the rate of phase separation (and are linearly related to the surface tension between the two phases) are colored the same as the corresponding symbols.

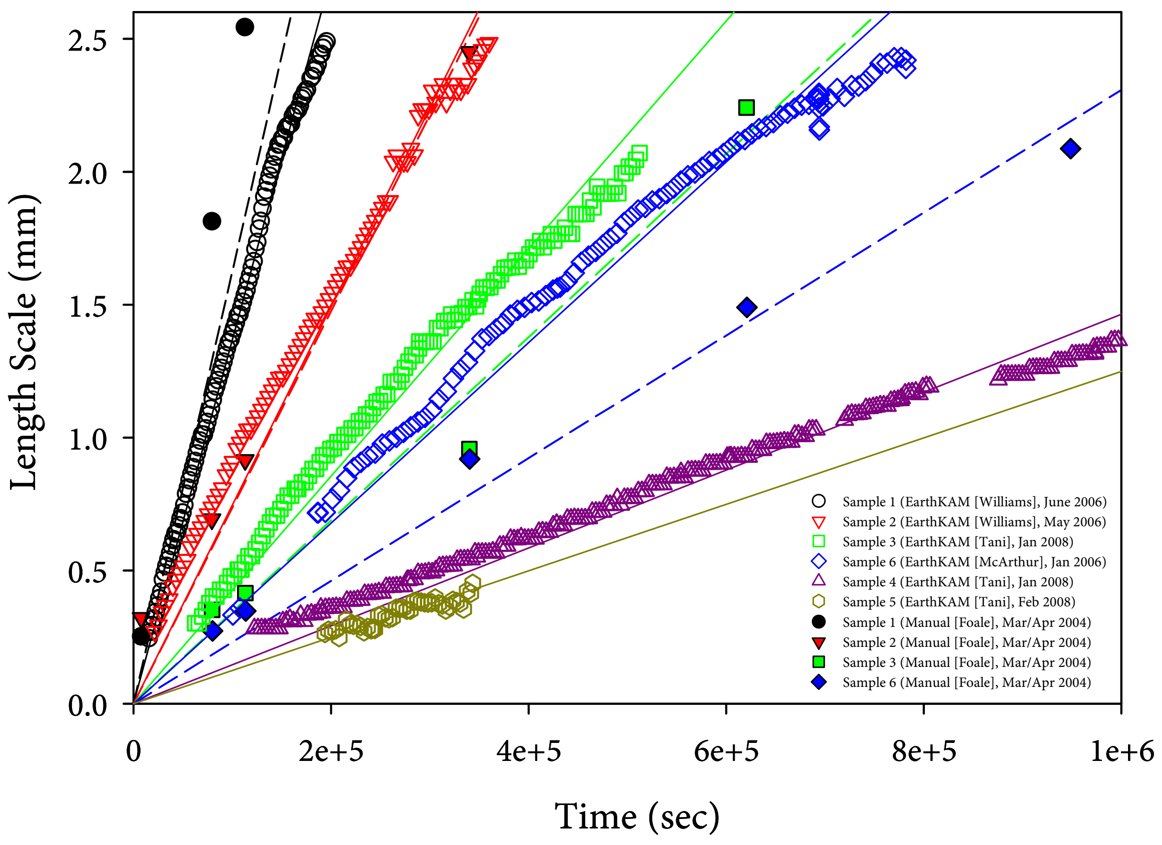

Figure 11.19: Results of the complete BCAT3 autocorrelation analysis, including data from all samples. Data for manually-photographed samples with flashlight illumination and their best fit lines are indicated with solid symbols and dashed lines, as in figure 11.18. Data for samples imaged automatically with EarthKAM and flash illumination are plotted with open symbols, and solid best-fit lines with corresponding colors, which give the rate of phase separation and are linearly related to the surface tension between the two phases.

for $r'_{\max}(t)$ by roughly two orders of magnitude, which are shown with open symbols in figure 11.19. Best-fit lines to the EarthKAM data are shown in solid lines of the same color. With the exception of Sample 6, where an air bubble passed through during image acquisition and stirred the sample, the rates of phase separation (i.e. the slope of the best-fit lines) are well defined and reproducible.

The interpretation of this data is ongoing.

11.7 Acknowledgments

This chapter represents the combined efforts of a large group of people, of which I am fortunate to be a part. I have worked most closely with Cathy Frey and Bill Meyer at the NASA GRC in actually conducting the experiment, and navigating the twists and turns of the NASA organization. Our experiment has received tremendous support from other parts of the NASA hierarchy, as well, from Bill Gerstenmeier at NASA HQ, to many flights, payloads and communications operators at MSFC and JSC, such as Janice Voss and Matt Abbott. On the technical side, Brion Au has been instrumental in adopting EarthKAM to BCAT, which has revolutionized our experiment. But perhaps the greatest contribution has come from the astronauts on board the ISS over many years, who have worked very hard to get us our data—spending far more of their time than was asked, let alone required—and improved the experiment well beyond what we thought possible. My personal gratitude is extended to Mike Foale, E. “Mike” Fincke, Leroy Chiao, Bill McArthur, Jeff Williams, Peggy Whitson and Dan Tani.

Back on earth, I quantified colloid volume fraction with the spectrophotometer and fluorometer together with Peter Sims in the laboratory of X. Sunney Xie, who generously made the facilities available. Mark Christiansen was absolutely instrumental in getting me started with After Effects, and helping me to figure out various of the techniques I used for image analysis; without his help, the data would still be unprocessed. Hidekazu Oki spent a significant amount of time and effort to speed up the autocorrelation code, originally provided by Art Bailey, impressively improving the performance by an order of magnitude. This work was supported by NASA.

Bibliography

[1] D. G. A. L. Aarts, R. P. A. Dullens and H. N. W. Lekkerkerker, “Interfacial dynamics in demixing systems with ultralow interfacial tension,” *New J. Phys.* **7**, 40 (2005).

[2] B. Albrecht, A. V. Failla, A. Schweitzer and C. Cremer, “Spatially modulated illumination microscopy allows axial distance resolution in the nanometer range,” *Appl. Opt.* **41**, 80-87 (2002).

[3] C. Allain, M. Cloitre and M. Wafra, “Aggregation and Sedimentation in Colloidal Suspensions,” *Phys. Rev. Lett.* **74**, 1478-1481 (1995).

[4] M. P. Allen and D. J. Tildesley, *Computer Simulation of Liquids* (Oxford: Oxford UP, 1989).

[5] V. J. Anderson, E. H. A. de Hoog and H. N. W. Lekkerkerker, “Mechanisms of phase separation and aggregation in colloid-polymer mixtures,” *Phys. Rev. E* **65**, 011403 (2001).

[6] V. J. Anderson and H. N. W. Lekkerkerker, “Insights into phase transition kinetics from colloid science,” *Nature* **416**, 811-815 (2002).

[7] A. A. Antipov, G. B. Sukhorukov, Y. A. Fedutik, J. Hartmann, M. Giersig and H. Möhwald, “Fabrication of a Novel Type of Metallized Colloids and Hollow Capsules,” *Langmuir* **18**, 6687-6693 (2002).

[8] A. A. Apodaca and L. Gritz, *Advanced RenderMan: Creating CGI for Motion Pictures* (San Francisco: Morgan Kaufmann, 2000).

[9] N. Arhel, A. Genovesio, K.-A. Kim, S. Miko, E. Perret, J.-C. Olivo-Marin, S. Shorte and P. Charneau, “Quantitative four-dimensional tracking of cytoplasmic and nuclear HIV-1 complexes,” *Nat. Meth.* **3**, 817-823 (2006).

[10] S. Asakura and F. Oosawa, “On Interaction between Two Bodies Immersed in a Solution of Macromolecules,” *J. Chem. Phys.* **22**, 1255-1256 (1954).

[11] N. W. Ashcroft and N. D. Mermin, *Solid State Physics* (Fort Worth, TX: Saunders, 1976).

[12] D. Asnaghi, M. Carpineti, M. Giglio and M. Sozzi, “Coagulation kinetics and aggregate morphology in the intermediate regimes between diffusion-limited and reaction-limited cluster aggregation,” *Phys. Rev. A* **45**, 1018-1023 (1992).

[13] D. Asnaghi, M. Carpineti, M. Giglio and A. Vailati, “Light Scattering Studies of Aggregation Phenomena,” *Physica A* **213**, 148-158 (1995).

[14] J. Azoulay, A. Débarre, A. Richard, P. Tchénio, S. Bandow and S. Iijima, “Polarised spectroscopy of individual single-wall nanotubes: Radial-breathing mode study,” *Europhys. Lett.* **53**, 407-413 (2001).

[15] S. Babu, J. C. Gimel and T. Nicolai, “Phase separation and percolation of reversibly aggregating spheres with a square-well attraction potential,” *J. Chem. Phys.* **125**, 184512 (2006).

[16] J. H. Bae, K. H. Kim, M. H. Hong, C. H. Gim and W. Jhe, “High resolution confocal detection of nanometric displacement by use of a 2 x 1 optical fiber coupler,” *Opt. Lett.* **25**, 1696-1698 (2000).

[17] K. Bahlmann, S. Jakobs and S. W. Hell, “4Pi-confocal microscopy of live cells,” *Ultramicroscopy* **87**, 155-164 (2001).

[18] A. E. Bailey, W. C. K. Poon, R. J. Christianson, A. B. Schofield, U. Gasser, V. Prasad, S. Manley, P. N. Segre, L. Cipelletti, W. V. Meyer, M. P. Doherty, S. Sankaran, A. L. Jankovsky, W. L. Shiley, J. P. Bowen, J. C. Eggers, C. Kurta, T. Lorik, Jr., P. N. Pusey and D. A. Weitz, “Spinodal Decomposition in a Model Colloid-Polymer Mixture in Microgravity,” *Phys. Rev. Lett.* **99**, 205701 (2007).

[19] G. A. Baxes, *Digital Image Processing: Principles and Applications* (New York: Wiley, 1994).

[20] C. W. J. Beenakker and P. Mazur, “Self-diffusion of spheres in a concentrated suspension,” *Physica A* **120**, 388-410 (1983).

[21] H. Berg, “How to track bacteria,” *Rev. Sci. Instruments* **42**, 868-871 (1971).

[22] J. Bergenholtz and M. Fuchs, “Nonergodicity transitions in colloidal suspensions with attractive interactions,” *Phys. Rev. E* **59**, 5706-5715 (1999).

[23] J. Bergenholtz and M. Fuchs, “Gel transitions in colloidal suspensions,” *J. Phys.: Condens. Matter* **11**, 10171-10182 (1999).

[24] J. Bergenholtz, M. Fuchs and Th. Voightmann, “Colloidal gelation and non-ergodicity transitions,” *J. Phys.: Condens. Matter* **12**, 6575-6583 (2000).

[25] J. Bergenholtz, W. C. K. Poon and M. Fuchs, “Gelation in Model Colloid-Polymer Mixtures,” *Langmuir* **19**, 4493-4503 (2003).

[26] B. J. Berne and R. Pecora, *Dynamic Light Scattering* (New York: Wiley, 1976).

[27] A. J. C. Bik, *The Software Vectorization Handbook.* (Hillsboro, OR: Intel, 2004).

[28] C. M. Blanca and S. W. Hell, “Sharp Spherical Focal Spot by Dark Ring 4Pi-Confocal Microscopy,” *Single Mol.* **2**, 207-210 (2001).

[29] C. M. Blanca, J. Bewersdorf and S. W. Hell, “Determination of unknown phase difference in 4Pi-confocal microscopy through the image intensity,” *Opt. Comm.* **206**, 281-285 (2002).

[30] A. Bloess, Y. Durand, M. Matsushita, R. Verberk, E. J. J. Groenen and J. Schmidt, “Microscopic Structure in a Shpol’skii System: A Single-Molecule Study of Dibenzanthanthrene in n-Tetradecane,” *J. Phys. Chem. A* **105**, 3016-3021 (2001).

[31] I. Bodnár, J. K. G. Dhont and H. N. W. Lekkerkerker, “Pretransitional Phenomena of a Colloid Polymer Mixture Studied with Static and Dynamic Light Scattering,” *J. Phys. Chem.* **100**, 19614-19619 (1996).

[32] I. Bodnár and W. D. Oosterbaan, “Indirect determination of the composition of the coexisting phases in a demixed colloid polymer mixture,” *J. Chem. Phys* **106**, 7777-7780 (1997).

[33] P. G. Bolhuis, A. A. Louis and J.-P. Hansen, “Influence of Polymer-Excluded Volume on the Phase-Behavior of Colloid-Polymer Mixtures,” *Phys. Rev. Lett.* **89**, 128302 (2002).

[34] E. Bonanomi, P. Sandkühler, J. Sefcik, M. Morari, and M. Morbidelli, “Precursor Aggregate Distributions for Particulate Gels,” *Ind. Eng. Chem. Res.* **43**, 4740-4752 (2004).

[35] M. Born and E. Wolf, *Principles of Optics*, 6th ed. (Cambridge: Cambridge UP, 1997).

[36] J. Brandrup and E. H. Immergut, eds., *Polymer Handbook*, 3rd ed. (New York: Wiley, 1989).

[37] R. Brown, *A Brief Account of Microscopical Observations Made in the Months of June, July and August 1827 on the Particles Contained in the Pollen of Plants; and on the General Existence of Active Molecules in Organic and Inorganic Bodies* (London: Taylor, 1828).

[38] S. Buzzaccaro, R. Rusconi and R. Piazza, “ ‘Sticky’ Hard Spheres: Equation of State, Phase Diagram, and Metastable Gels,” *Phys. Rev. Lett.* **99**, 098301 (2007).

[39] T. A. Byassee, W. C. W. Chan and S. Nie, “Probing Single Molecules in Single Living Cells,” *Anal. Chem.* **72**, 5606-5611 (2000).

[40] A. Cacciuto, S. Auer and D. Frenkel, “Breakdown of Classical Nucleation Theory near Isostructural Phase Transitions,” *Phys. Rev. Lett.* **93**, 166105 (2004).

[41] T. A. Camesano, M. J. Natan and B. E. Logan, “Observation of Changes in Bacterial Cell Morphology Using Tapping Mode Atomic Force Microscopy,” *Langmuir* **16**, 4563-4572 (2000).

[42] A. I. Campbell, V. J. Anderson, J. S. van Duijneveldt and P. Bartlett, “Dynamical Arrest in Attractive Colloids: The Effect of Long-Range Repulsion,” *Phys. Rev. Lett.* **94**, 208301 (2005).

[43] H. Cang, C. M. Wong, C. S. Xu, A. H. Rizvi and H. Yang, “Confocal three dimensional tracking of a single nanoparticle with concurrent spectroscopic readouts,” *Appl. Phys. Lett.* **88**, 223901 (2006).

[44] F. Cardinaux, T. Gibaud, A. Stradner and P. Schurtenberger, “Interplay between Spinodal Decomposition and Glass Formation in Proteins Exhibiting Short-Range Attractions,” *Phys. Rev. Lett.* **99**, 118301 (2007).

[45] M. Carpineti, F. Ferri, M. Giglio, E. Paganini and U. Perini, “Salt-induced fast aggregation of polystyrene latex,” *Phys. Rev. A* **42**, 7347-7354 (1990).

[46] M. Carpineti and M. Giglio, “Spinodal-Type Dynamics in Fractal Aggregation of Colloidal Clusters,” *Phys. Rev. Lett.* **68**, 3327-3330 (1992).

[47] M. Carpineti and M. Giglio, “Transition from semiorder to disorder in the aggregation of dense colloidal solutions,” *Phys. Rev. Lett.* **70**, 3828-3831 (1993).

[48] M. Carpineti, M. Giglio and V. Degiorgio, “Mass conservation and anticorrelation effects in the colloidal aggregation of dense solutions,” *Phys. Rev. E* **51**, 590-596 (1995).

[49] K. R. Castleman, *Digital Image Processing* (Englewood Cliffs, NJ: Prentice Hall, 1996).

[50] M. E. Cates, M. Fuchs, K. Kroy, W. C. K. Poon and A. M. Puertas, “Theory and simulation of gelation, arrest and yielding in attracting colloids,” *J. Phys: Condens. Matter* **16**, S4861-S4875 (2004).

[51] P. M. Chaikin and T. C. Lubensky, *Principles of condensed matter physics* (Cambridge: Cambridge UP, 1995).

[52] P. Charbonneau and D. R. Reichman, “Systematic characterization of thermodynamic and dynamical phase behavior in systems with short-ranged attraction,” *Phys. Rev. E* **75**, 011507 (2007).

[53] A. P. Chatterjee and K. S. Schweizer, “Microscopic theory of polymer-mediated interactions between spherical particles,” *J. Chem. Phys.* **109**, 10464-10476 (1998).

[54] A. P. Chatterjee and K. S. Schweizer, “Correlation effects in dilute particle-polymer mixtures,” *J. Chem. Phys.* **109**, 10477-10488 (1998).

[55] A. P. Chatterjee and K. S. Schweizer, “Influence of solvent quality and thermal fluctuations on polymer-mediated depletion interactions,” *Macromol.* **32**, 923-924 (1999).

[56] B. Chen, G. Parker II, J. Han, M. Meyyappan and A. M. Cassell, “Heterogeneous Single-Walled Carbon Nanotube Catalyst Discovery and Optimization,” *Chem. Mater.* **14**, 1891-1896 (2002).

[57] Y. L. Chen, K. S. Schweizer and M. Fuchs, “Phase separation in suspensions of colloids, polymers and nanoparticles: Role of solvent quality, physical mesh, and nonlocal entropic repulsion,” *J. Chem. Phys.* **118**, 3880-3890 (2003).

[58] Y. L. Chen and K. S. Schweizer, “Microscopic theory of gelation and elasticity in polymer-particle suspensions,” *J. Chem. Phys.* **120**, 7212-7222 (2004).

[59] G. Chirico, F. Cannone, S. Beretta, A. Diaspro, B. Campanini, S. Bettati, R. Ruotolo and A. Mozzarelli, “Dynamics of green fluorescent protein mutant2 in solution, on spin-coated glasses, and encapsulated in wet silica gels,” *Protein Sci.* **11**, 1152-1161 (2002).

[60] D. T. Chiu, C. F. Wilson, A. Karlsson, A. Danielsson, A. Lundqvist, A. Strömberg, F. Ryttsén, M. Davidson, S. Nordholm, O. Orwar and R. N. Zare, “Manipulating the biochemical nanoenvironment around single molecules contained within vesicles,” *Chem. Phys.* **247**, 133-139 (1999).

[61] J. Choi, Y. Zhao, D. Zhang, S. Chien and Y. H. Lo, “Patterned Fluorescent Particles as Nanoprobes for the Investigation of Molecular Interactions,” *Nano Lett.* **3**, 995-1000 (2003).

[62] M. Christiansen, *Adobe After Effects 7.0 Studio Techniques* (Berkeley: Peachpit, 2006).

[63] T. R. Corle and G. S. Kino, *Confocal Scanning Optical Microscope and Related Imaging Systems* (San Diego: Academic Press, 1996).

[64] R. Cortes and S. Raghavachary, *The RenderMan Shading Language Guide* (Boston: Thomson Course Technology, 2008).

[65] M. T. Crisp and N. A. Kotov, “Preparation of Nanoparticle Coatings on Surfaces of Complex Geometry. *Nano Lett.* **3**, 173-177 (2003).

[66] J. C. Crocker and D. G. Grier, “Methods of Digital Video Microscopy for Colloidal Studies,” *J. Colloid Interface Sci.* **179**, 298-310 (1996).

[67] Z. F. Dai, H. Möhvald, B. Tiersche and L. Dähne, “Nanoengineering of Polymeric Capsules with a Shell-in-Shell Structure,” *Langmuir* **18**, 9533-9538 (2002).

[68] G. Decher, “Fuzzy Nanoassemblies: Toward Layered Polymeric Multicomposites,” *Science* **277**, 1232-1237 (1997).

[69] A. de Candia, E. Del Gado, A. Fierro, N. Sator and A. Coniglio, “Colloidal gelation, percolation and structural arrest,” *Physica A* **358**, 239-248 (2005).

[70] E. H. A. de Hoog, W. K. Kegel, A. van Blaaderen and H. N. W. Lekkerkerker, “Direct observation of crystallization and aggregation in a phase-separating colloid-polymer suspension,” *Phys. Rev. E* **64**, 021407 (2001).

[71] B. A. de L. Costello, P. F. Luckham and Th. F. Tadros, “Investigation of the Interaction Forces of Polymer-Coated Surfaces Using Force Balance, Rheology, and Osmotic Pressure Results,” *Langmuir* **8**, 464-468 (1992).

[72] E. Del Gado and W. Kob, “Network formation and relaxation dynamics in a new model for colloidal gelation,” *J. Non-Newtonian Fluid Mech.* **149**, 28-33 (2008).

[73] C. J. Dibble, M. Kogan and M. J. Solomon, “Structure and dynamics of colloidal depletion gels: Coincidence of transitions and heterogeneity,” *Phys. Rev. E* **74**, 041403 (2006).

[74] H. M. Dietel and P. J. Dietel, *C++ How to Program*, 2nd ed. (Upper Saddle River, NJ: Prentice Hall, 1998).

[75] A. Diaspro, ed. *Confocal and Two-Photon Microscopy: Foundations, Applications, and Advances* (New York: Wiley-Liss, 2002).

[76] M. Dijkstra, R. van Roij and R. Evans, “Phase Behavior and Structure of Binary Hard-Sphere Mixtures,” *Phys. Rev. Lett.* **81**, 2268-2271 (1998).

[77] M. Dijkstra, J. M. Brader and R. Evans, “Phase behavior and structure of model colloid-polymer mixtures,” *J. Phys: Condens. Matter* **11**, 10079-10106 (1999).

[78] M. Dijkstra, R. van Roij and R. Evans, “Direct Simulation of the Phase Behavior of Binary Hard-Sphere Mixtures: Test of the Depletion Potential Description,” *Phys. Rev. Lett.* **82**, 117-120 (1999).

[79] P. Dimon, S. K. Sinha, D. A. Weitz, C. R. Safinya, G. S. Smith, W. A. Varady, and H. M. Lindsay, “Structure of Aggregated Gold Colloids,” *Phys. Rev. Lett.* **57**, 595-598 (1986).

[80] A. D. Dinsmore, V. Prasad, I. Y. Wong and D. A. Weitz, “Microscopic Structure and Elasticity of Weakly Aggregated Colloidal Gels,” *Phys. Rev. Lett.* **96**, 185502 (2006).

[81] R. P. A. Dullens, D. G. A. L. Aarts and W. K. Kegel, “Direct measurement of the free energy by optical microscopy,” *Proc. Nat. Acad. Sci USA* **103**, 529-531 (2006).

[82] H. B. Dwight, *Tables of Integrals and Other Mathematical Data* (New York: Macmillan, 1961).

[83] T. Eckert and E. Bartsch, “Re-entrant Glass Transition in a Colloid-Polymer Mixture with Depletion Attractions,” *Phys. Rev. Lett.* **89**, 125701 (2002).

[84] P. C. Ecklund, B. K. Pradhan, U. J. Kim, Q. Xiong, J. E. Fischer, A. D. Friedman, B. C. Holloway, K. R. Jordan and M. W. Smith, “Large-scale production of single-walled carbon nanotubes using ultrafast pulses from a free electron laser,” *Nano Lett.* **2**, 561-566 (2002).

[85] A. Egner, V. Andresen and S. W. Hell, “Comparison of the axial resolution of practical Nipkow-disk confocal fluorescence microscopy with that of multifocal multiphoton microscopy: theory and experiment,” *J. Microsc.* **206**, 24-32 (2002).

[86] A. Egner, S. Jakobs and S. W. Hell, “Fast 100-nm resolution three-dimensional microscope reveals structural plasticity of mitochondria in live yeast,” *Proc. Nat. Acad. Sci. USA* **99**, 3370-3375 (2002).

[87] D. S. English, L. E. Pell, Z. Yu, P. F. Barbara and B. A. Korgel, “Size Tunable Visible Luminescence from Individual Organic Monolayer Stabilized Silicon Nanocrystal Quantum Dots,” *Nano Lett.* **2**, 681-685 (2002).

[88] M. M. Ferris and K. L. Rowlen, “Detection and enumeration of single nanometric particles: A confocal optical design for fluorescence flow cytometry,” *Rev. Sci. Instruments* **73**, 2404-2410 (2002).

[89] N. Fertig, A. Tilke, R. H. Blick, J. P. Kotthaus, J. C. Behrends and G. ten Bruggencate, “Stable integration of isolated cell membrane patches in a nanomachined aperture: A step towards a novel device for membrane physiology,” *Appl. Phys. Lett.* **77**, 1218-1220 (2000).

[90] G. Foffi, G. D. McCullagh, A. Lawlor, E. Zaccarelli, K. A. Dawson, F. Sciortino, P. Tartaglia, D. Pini and G. Stell, “Phase equilibria and glass transition in colloidal systems with short-ranged attractive interactions: Application to protein crystallization,” *Phys. Rev. E* **65**, 031407 (2002).

[91] G. Foffi, C. De Michele, F. Sciortino and P. Tartaglia, “Scaling of Dynamics with the Range of Interaction in Short-Range Attractive Colloids,” *Phys. Rev. Lett.* **94**, 078301 (2005).

[92] G. Foffi, C. De Michele, F. Sciortino and P. Tartaglia, “Arrested phase separation in a short-ranged attractive colloidal system: A numerical study,” *J. Chem. Phys.* **122**, 224903 (2005).

[93] A. Fortini, M. Dijkstra and R. Tuinier, “Phase behaviour of charged colloidal sphere dispersions with added polymer chains,” *J. Phys.: Condens. Matter* **17**, 7783-7803 (2005).

[94] A. Fortini, M. Schmidt and M. Dijkstra, “Phase behavior and structure of model colloid-polymer mixtures confined between two parallel planar hard walls,” *Phys. Rev. E* **73**, 051502 (2006).

[95] B. Fraser and J. Schewe, *Real World Camera Raw with Adobe Photoshop CS3* (Berkeley: Peachpit, 2008).

[96] B. J. Frisken, “Revisiting the method of cumulants for the analysis of dynamic light-scattering data,” *Appl. Opt.* **40**, 4087-4091 (2001).

[97] M. Fuchs and K. S. Schweizer, “Structure and thermodynamics of colloid-polymer mixtures: A macromolecular approach,” *Europhys. Lett.* **51**, 621-627 (2000).

[98] M. Fuchs and K. S. Schweizer, “Structure of colloid-polymer suspensions,” *J. Phys.: Condens. Matter* **14**, R239-R269 (2002).

[99] H. Furukawa, “A dynamic scaling assumption for phase separation,” *Adv. Phys.* **34**, 703-750 (1985).

[100] Y. Gao and M. L. Kilfoil, “Direct Imaging of Dynamical Heterogeneities near the Colloid-Gel Transition,” *Phys. Rev. Lett.* **99**, 078301 (2007).

[101] N. Gaponik, I. L. Radtchenko, M. R. Gerstenberger, Y. A. Fedutik, G. B. Sukhorukov and A. L. Rogach, “Labeling of Biocompatible Polymer Microcapsules with Near-Infrared Emitting Nanocrystals,” *Nano Lett.* **3**, 369-372 (2003).

[102] U. Gasser, E. R. Weeks, A. Schofield, P. N. Pusey and D. A. Weitz, “Real-Space Imaging of Nucleation and Growth in Colloidal Crystallization,” *Science* **292**, 258-262 (2001).

[103] L. J. Gauckler, Th. Graule and F. Baader, “Ceramic forming using enzyme catalyzed reactions,” *Mat. Chem. Phys.* **61**, 78-102 (1999).

[104] R. Gerber, *The Software Optimization Cookbook.* (Hillsboro, OR: Intel, 2002).

[105] B. Gligorijevic, R. McAllister, J. S. Urbach and P. D. Roepe, “Spinning Disk Confocal Microscopy of Live, Intraerythrocytic Malarial Parasites. 1. Quantification of Hemozoin Development for Drug Sensitive versus Resistant Malaria,” *Biochem.* **45**, 12400-12410 (2006).

[106] B. Gligorijevic, R. McAllister, J. S. Urbach and P. D. Roepe, “Spinning Disk Confocal Microscopy of Live, Intraerythrocytic Malarial Parasites. 2. Altered Vacuolar Volume Regulation in Drug Resistant Malaria,” *Biochem.* **45**, 12411-12423 (2006).

[107] A. E. González and G. Ramírez-Santiago, “Spatial Ordering and Structure Factor Scaling in the Simulations of Colloid Aggregation,” *Phys. Rev. Lett.* **74**, 1238-1241 (1995).

[108] R. C. Gonzalez, R. E. Woods and S. L. Eddins, *Digital Image Processing using MATLAB* (Upper Saddle River, NJ: Pearson, 2004).

[109] V. Gopalakrishnan and C. F. Zukoski, “Microstructure of equilibrium fluid clusters in colloid-polymer suspensions,” *Phys. Rev. E* **75**, 021406 (2007).

[110] J. Gorelik, A. Shevchuk, M. Ramalho, M. Elliott, C. Lei, C. F. Higgins, M. J. Lab, D. Klenerman, N. Krauzewicz and Y. Korchev, “Scanning surface confocal microscopy for simultaneous topographical and fluorescence imaging: Application to single virus-like particle entry into a cell,” *Proc. Nat. Acad. Sci. USA* **99**, 16018-16023 (2002).

[111] D. A. D. Gould, *Complete Maya Programming: An Extensive Guide to MEL and the C++ API* (San Francisco: Morgan Kaufmann, 2003).

[112] M. C. Grant and W. B. Russel, “Volume-fraction dependence of elastic moduli and transition temperatures for colloidal silica gels,” *Phys. Rev. E* **47**, 2606-2614 (1993).

[113] J. Groenewold and W. K. Kegel, “Anomalously Large Equilibrium Clusters of Colloids,” *J. Phys. Chem. B* **105**, 11702-11709 (2001).

[114] J. Groenewold and W. K. Kegel, “Colloidal cluster phases, gelation and nuclear matter,” *J. Phys.: Condens. Matter* **16**, S4877-S4886 (2004).

[115] M. Gu, *Principles of Three-Dimensional Imaging in Confocal Microscopes* (Singapore: World Scientific, 1996).

[116] R. Gupta, Q. Xiong, C. K. Adu, U. J. Kim and P. C. Eklund, “Laser-Induced Fano Resonance Scattering in Silicon Nanowires,” *Nano Lett.* **3**, 627-631 (2003).

[117] M. G. L. Gustafsson, D. A. Agard and J. W. Sedat, “I5M: 3D widefield light microscopy with better than 100 nm axial resolution,” *J. Microsc.* **195**, 10-16 (1999).

[118] M. G. L. Gustafsson, "Extended resolution fluorescence microscopy," *Curr. Opin. Struct. Biol.* **9**, 627-634 (1999).

[119] M. G. L. Gustafsson, "Surpassing the lateral resolution limit by a factor of two using structured illumination microscopy," *J. Microsc.* **198**, 82-87 (2000).

[120] A. Hanke, E. Eisenriegler and S. Dietrich, "Polymer depletion effects near mesoscopic particles," *Phys. Rev. E* **59**, 6853-6878 (1999).

[121] J. P. Hansen and I. R. McDonald, *Theory of Simple Liquids* (London: Academic Press, 2006).

[122] C. L. Haynes and R. P. van Duyne, "Dichroic Optical Properties of Extended Nanostructures Fabricated Using Angle-Resolved Nanosphere Lithography," *Nano Lett.* **3**, 939-943 (2003).

[123] M. Hazani, R. Naaman, F. Hennrich and M. M. Kappes, "Confocal Fluorescence Imaging of DNA-functionalized Carbon Nanotubes," *Nano Lett.* **3**, 153-155 (2003).

[124] S. W. Hell and E. H. H. Stelzer, "Properties of a 4Pi confocal fluorescence microscope," *J. Opt. Soc. Am. A* **9**, 2159-2166 (1992).

[125] S. W. Hell and E. H. H. Stelzer, "Fundamental improvement of resolution with a 4Pi-confocal fluorescence microscope using two-photon emission," *Opt. Comm.* **93**, 277-282 (1992).

[126] S. W. Hell, E. H. K. Stelzer, S. Lindek and C. Cremer "Confocal microscopy with an increased detection aperture: type-B 4Pi confocal microscopy" *Opt. Lett.* **19**, 222-224 (1994).

[127] S. W. Hell and M. Nagorni, "4Pi confocal microscopy with alternate interference," *Opt. Lett.* **23**, 1567-1569 (1998).

[128] R. Hooke, *Micrographia* (London: Royal Society, 1667).

[129] M. F. Hsu, E. R. Dufresne and D. A. Weitz, "Charge Stabilization in Nonpolar Solvents," *Langmuir* **21**, 4881-4887 (2005).

[130] C. Hubert, J. Levy, A. C. Carter, W. Chang, S. W. Kiechoefer, J. S. Horwitz and D. B. Chrisey, "Confocal scanning optical microscopy of $\mathrm{Ba_xSr_{1-x}TiO_3}$ thin films," *Appl. Phys. Lett.* **71**, 3353-3355 (1997).

[131] C. Hubert and J. Levy, "New optical probe of GHz polarization dynamics in ferroelectric thin films," *Rev. Sci. Instruments* **70**, 3684-3687 (1999).

[132] C. Hubert, J. Levy, E. J. Cukauskas and S. W. Kiechoefer, "Mesoscopic Microwave Dispersion in Ferroelectric Thin Films," *Phys. Rev. Lett.* **85**, 1998-2001 (2000).

[133] C. Hubert, J. Levy, T. V. Rivkin, C. Carlson, P. A. Parilla, J. D. Perkins and D. S. Ginley, “Nanopolar reorientation in ferroelectric thin films,” *Appl. Phys. Lett.* **79**, 2058-2060 (2001).

[134] R. J. Hunter, *Foundations of Colloid Science,* 2nd ed. (Oxford: Oxford UP, 2001).

[135] K. Hwang, H. J. Wu, M. A. Bevan, “Specific Ion-Dependent Attraction and Phase Behavior of Polymer-Coated Colloids,” *Langmuir* **20**, 11393-11401 (2004).

[136] S. M. Ilett, A. Orrock, W. C. K. Poon and P. N. Pusey, “Phase behavior of a model colloid-polymer mixture,” *Phys. Rev. E* **51**, 1344-1352 (1995).

[137] C. Jiang, J. Li, J. Zhao, U. Kolb and A. Mews, “Combination of Confocal Raman Spectroscopy and Electron Microscopy on the Same Individual Bundles of Single-Walled Carbon Nanotubes,” *Nano Lett.* **2**, 1209-1213 (2002).

[138] C. S. Johnson, Jr. and D. A. Gabriel, *Laser Light Scattering* (New York: Dover, 1995).

[139] R. A. L. Jones, *Soft Condensed Matter* (Oxford: Oxford UP, 2002).

[140] A. Jorio, R. Saito, J. H. Hafner, C. M. Lieber, M. Hunter, T. McClure, G. Dresselhaus and M. S. Dresselhaus, “Structural (n,m) Determination of Isolated Single-Wall Carbon Nanotubes by Resonant Raman Scattering,” *Phys. Rev. Lett.* **86**, 1118-1121 (2001).

[141] H. Kano, S. Jakobs, M. Nagorni and S. W. Hell, “Dual-color 4-Pi confocal microscopy with 3D-resolution in the 100 nm range,” *Ultramicroscopy* **90**, 207-213 (2002).

[142] A. J. Khopade and F. Caruso, “Electrostatically Assembled Polyelectrolyte / Dendrimer Multilayer Films as Ultrathin Nanoreservoirs,” *Nano Lett.* **2**, 415-418 (2002).

[143] C. Kittel, *Introduction to Solid State Physics.* 7th ed. (New York: Wiley, 1996).

[144] F. Koberling, A. Mews, G. Philipp, U. Kolb and I. Potapova, “Fluorescence Spectroscopy and transmission electron microscopy of the same isolated semiconductor nanocrystals,” *Appl. Phys. Lett.* **81**, 1116-1118 (2002).

[145] L. A. Kolodny, D. M. Willard, L. L. Carillo, M. W. Nelson and A. Van Orden, “Spatially Correlated Fluorescence/AFM of Individual Nanosized Particles and Biomolecules,” *Anal. Chem.* **73**, 1959-1966 (2001).

[146] K. Kroy, M. E. Cates and W. C. K. Poon, “Cluster Mode-Coupling Approach to Weak Gelation in Attractive Colloids,” *Phys. Rev. Lett.* **92**, 148302 (2004).

[147] M. Kuno, D. P. Fromm, H. F. Hamann, A. Gallagher and D. J. Nesbitt, "Nonexponential 'blinking' kinetics of single CdSe quantum dots: A universal power law behavior," *J. Chem. Phys.* **112**, 3117-3120 (2000).

[148] M. Kuno, D. P. Fromm, H. F. Hamann, A. Gallagher and D. J. Nesbitt, " 'On'/'off' fluorescence intermittency of single semiconductor quantum dots," *J. Chem. Phys.* **115**, 1028-1040 (2001).

[149] M. Kuno, D. P. Fromm, A. Gallagher, D. J. Nesbitt, O. I. Micic and A. J. Nozik, "Fluorescence Intermittency in Single InP Quantum Dots," *Nano Lett.* **1**, 557-564 (2001).

[150] T. D. Lacoste, X. Michalet, F. Pinaud, D. S. Chemla, A. P. Alivisatos and S. Weissdagger, "Ultrahigh-resolution multicolor colocalization of single fluorescent probes," *Proc. Nat. Acad. Sci. USA* **97**, 9461-9466 (2000).

[151] L. Lamport, LaTeX: *a Document Preparation System* (Reading, MA: Addison-Wesley, 1994).

[152] L. Latterini, F. Elisei, G. G. Aloisi, U. Costantino and M. Nocchetti, "Space-resolved fluorescence properties of pheolphthalein-hydrotalcie nanocomposites," *Phys. Chem. Chem. Phys.* **4**, 2792-2798 (2002).

[153] G. Ledoux, D. Amans, J. Gong, F. Huisken, F. Cichos and J. Martin, "Nanostructured films composed of silicon nanocrystals," *Mat. Sci. Eng. C* **19**, 215-218 (2002).

[154] H. N. W. Lekkerkerker, W. C. K. Poon, P. N. Pusey, A. Stroobants and P. B. Warren, "Phase Behavior of Colloid+Polymer Mixtures," *Europhys. Lett.* **20**, 559-564 (1992).

[155] H. N. W. Lekkerkerker, J. K. G. Dhont, H. Verduin, C. Smits and J. S. van Duijneveldt "Interactions, phase transitions and metastable states in concentrated colloidal dispersions," *Physica A* **213**, 18-29 (1995).

[156] M. E. Leunissen, C. G. Christova, A.-P. Hynninen, C. P. Royall, A. I. Campbell, A. Imhof, M. Dijkstra, R. van Roij and A. van Blaaderen, "Ionic colloidal crystals of oppositely charged particles," *Nature* **437**, 235-240 (2005).

[157] S. A. Levi, A. Mourran, J. P. Spatz, F. C. J. M. van Veggel, D. N. Reinhoudt and M. Möller, "Fluorescence of Dyes Adsorbed on Highly Organized, Nanostructured Gold Surfaces," *Chem. Eur. J.* **8**, 3808-3814 (2002).

[158] V. Levi, Q. Q. Ruan and E. Gratton, "3-D Particle Tracking in a Two-Photon Microscope: Application to the Study of Molecular Dynamics in Cells," *Biophys. J.* **88**, 2919-2928 (2005).

[159] C. Li, G. D. Botsaris and D. L. Kaplan, “Selective in Vitro Effect of Peptides on Calcium Carbonate Crystallization” *Cryst. Growth Des.* **2**, 387-393 (2002).

[160] D. R. Lide, ed., *CRC Handbook of Chemistry and Physics*, 79th ed. (Boca Raton, FL: CRC Press, 1999).

[161] E. M. Lifshitz and L. P. Pitaevskii, *Physical Kinetics* (Oxford: Pergamon, 1981).

[162] M. Y. Lin, H. M. Lindsay, D. A. Weitz, R. C. Ball, R. Klein and P. Meakin, “Universality in colloid aggregation,” *Nature* **339**, 360-362 (1989).

[163] M. Y. Lin, H. M. Lindsay, D. A. Weitz, R. C. Ball, R. Klein and P. Meakin, “Universality of Fractal Aggregates as Probed by Light Scattering,” *Proc. R. Soc. Lond. A* **423**, 71-87 (1989).

[164] M. Y. Lin, H. M. Lindsay, D. A. Weitz, R. Klein, R. C. Ball and P. Meakin, “Universal diffusion-limited colloid aggregation,” *J. Phys.: Condens. Matter* **2**, 3093-3113 (1990).

[165] H. M. Lindsay, R. Klein, D. A. Weitz, M. Y. Lin and P. Meakin, “Structure and anisotropy of colloid aggregates,” *Phys. Rev. A* **39**, 3112-3119 (1989).

[166] S. B. Lippman and J. Lajoie, *C++ Primer*, 3rd ed. (Reading, Mass: Addison-Wesley, 1998).

[167] J. Liu, W. Y. Shih, M. Sarikaya and I. A. Aksay, “Fractal colloidal aggregates with finite interparticle interactions: Energy dependence of the fractal dimension,” *Phys. Rev. A* **41**, 3206-3213 (1990).

[168] F. Lo Verso, R. L. C. Vink, D. Pini and L. Reatto, “Critical behavior in colloid-polymer mixtures: Theory and simulation,” *Phys. Rev. E* **73**, 061407 (2006).

[169] A. A. Louis, P. G. Bolhuis, E. J. Meijer, J.-P. Hansen, “Polymer induced depletion potentials in polymer-colloid mixtures,” *J. Chem. Phys.* **117**, 1893-1907 (2002).

[170] P. J. Lu, “Confocal Scanning Optical Microscopy and Nanotechnology,” in *Handbook of Microscopy for Nanotechnology*, N. Yao, and Z. L. Wang, eds. (Boston: Kluwer, 2005), pp. 3-24.

[171] P. J. Lu, J. C. Conrad, H. M. Wyss, A. B. Schofield and D. A. Weitz, “Fluids of Clusters in Attractive Colloids,” *Phys. Rev. Lett.* **96**, 028306 (2006).

[172] P. J. Lu, P. A. Sims, H. Oki, J. B. Macarthur and D. A. Weitz, “Target-locking acqusition with real-time confocal (TARC) microscopy,” *Opt. Express* **15**, 8702-8712 (2007).

[173] P. J. Lu, E. Zaccarelli, F. Ciulla, A. B. Schofield, F. Sciortino and D. A. Weitz, “Gelation of particles with short-range attraction,” *Nature* **453**, (2008).

[174] Y. Lvov, A. A. Antipov, A. Mamedov, H. Möhwald and G. B. Sukhorukov, “Urease Encapsulation in Nanoorganized Microshells,” *Nano Lett.* **1**, 125-128 (2001).

[175] S. Manley, L. Cipelletti, V. Trappe, A. E. Bailey, R. J. Christianson, U. Gasser, V. Prasad, P. N. Segre, M. P. Doherty, S. Sankaran, A. L. Jankovsky, B. Shiley, J. Bowen, J. Eggers, C. Kurta, T. Lorik, and D. A. Weitz, “Limits to Gelation in Colloidal Aggregation,” *Phys. Rev. Lett.* **93**, 108302 (2004).

[176] S. Manley, H. M. Wyss, K. Miyazaki, J. C. Conrad, V. Trappe, L. J. Kaufman, D. R. Reichman and D. A. Weitz, “Glasslike Arrest in Spinodal Decomposition as a Route to Colloidal Gelation,” *Phys. Rev. Lett.* **95**, 238302 (2005).

[177] J. Martin, A. Kiesow, A. Heilmann, and R. Wannemacher, “Laser Microstructuring and Scanning Microscopy of Plasmapolymer-Silver Composite Layers,” *Appl. Opt.* **40**, 5726-5730 (2001).

[178] M. D. Mason, G. M. Credo, K. D. Weston and S. K. Buratto, “Luminescence of Individual Porous Si Chromophores,” *Phys. Rev. Lett.* **80**, 5405-5408 (1998).

[179] P. Meakin, “Formation of Fractal Clusters and Networks by Irreversible Diffusion-Limited Aggregation,” *Phys. Rev. Lett.* **51**, 1119-1122 (1983).

[180] R. Mezzenga, P. Schurtenberger, A. Burbidge and M. Michel, “Understanding foods as soft materials,” *Nature Mater.* **4**, 729-740 (2005).

[181] X. Michalet, T. D. Lacoste and S. Weiss, “Ultrahigh-Resolution Colocalization of Spectrally Separable Point-like Fluorescent Probes,” *Methods* **25**, 87-102 (2001).

[182] M. A. Miller and D. Frenkel, “Competition of Percolation and Phase Separation in a Fluid of Adhesive Hard Spheres,” *Phys. Rev. Lett.* **90**, 135702 (2003).

[183] S. Moehl, Hui Zhao, B. Dal Don, S. Wachter and H. Kalt, “Solid immersion lens-enhanced nano-photoluminescence: Principle and applications,” *J. Appl. Phys.* **93**, 6265-6272 (2003).

[184] A. Moussad, W. C. K. Poon, P. N. Pusey and M. F. Soliva, “Structure of Marginal and Fully Developed Colloidal Liquids,” *Phys. Rev. Lett.* **82**, 225-228 (1999).

[185] Y. S. Nam, H. S. Kang, J. Y. Park, T. G. Park, S.-H. Han and I.-S. Chang, “New micelle-like polymer aggregates made from PEI-PLGA diblock copolymers: micellar characteristics and cellular uptake,” *Biomaterials* **24**, 2053-2059 (2003).

[186] X. L. Nan, P. A. Sims, P. Chen and X. S. Xie, "Observation of Individual Microtubule Motor Steps in Living Cells with Endocytosed Quantum Dots," *J. Phys. Chem. B.* **109**, 24220-24224 (2005).

[187] S. Nie and R. N. Zare, "Optical Detection of Single Molecules," *Ann. Rev. Biophys. Biomol. Struct.* **26**, 567-596 (1997).

[188] M. G. Noro and D. Frenkel, "Extended corresponding-states behavior for particles with variable range attractions," *J. Chem. Phys.* **113**, 2941-2944 (2000).

[189] OpenGL Architecture Review Board, M. Woo, J. Neider, T. Davis and D. Shreiner, *OpenGL Programming Guide*, 3rd ed. (Boston: Addison-Wesley, 2001).

[190] OpenGL Architecture Review Board, *OpenGL Reference Manual*, 3rd ed. D. Shreiner, ed. (Boston: Addison-Wesley, 2001).

[191] R. K. Pathria, *Statistical Mechanics* (Oxford: Butterworth-Heineman, 1996).

[192] I. M. Peters, B. G. de Grooth, J. M. Schins, C. G. Figdor and J. Greve, "Three dimensional single-particle tracking with nanometer resolution," *Rev. Sci. Instruments* **69**, 2762-2766 (1998).

[193] K. N. Pham, A. M. Puertas, J. Bergenholtz, S. U. Egelhaaf, A. Moussad, P. N. Pusey, A. B. Schofield, M. E. Cates, M. Fuchs and W. C. K. Poon, "Multiple Glassy States in a Simple Model System," *Science* **296**, 104-106 (2002).

[194] M. Plischke and B. Bergersen, *Equilibrium Statistical Physics* (Singapore: World Scientific, 1994).

[195] W. C. K. Poon, A. D. Pirie and P. N. Pusey, "Gelation in colloid-polymer mixtures," *Faraday Discuss.* **101**, 65-76 (1995).

[196] W. C. K. Poon and M. D. Haw, "Mesoscopic structure formation in colloidal aggregation and gelation," *Adv. Colloid Interface Sci.* **73**, 71-126 (1997).

[197] W. C. K. Poon, A. D. Pirie, M. D. Haw and P. N. Pusey, "Non-equilibrium behaviour of colloid-polymer mixtures," *Physica A* **235**, 110-119 (1997).

[198] W. C. K. Poon, F. Renth, R. M. L. Evans, D. J. Fairhurst, M. E. Cates and P. N. Pusey, "Colloid-Polymer Mixtures at Triple Coexistence: Kinetic Maps from Free-Energy Landscapes," *Phys. Rev. Lett.* **83**, 1239-1242 (1999).

[199] W. C. K. Poon, "The physics of a model colloid-polymer mixture," *J. Phys.: Condens. Matter* **14**, R859-R880 (2002).

[200] V. Prasad, Ph.D. thesis, Harvard University (2002).

[201] W. K. Pratt, *Digital Image Processing* (New York: Wiley, 1991).

[202] A. M. Puertas, M. Fuchs and M. E. Cates, "Comparative Simulation Study of Colloidal Gels And Glasses," *Phys. Rev. Lett.* **88**, 098301 (2002).

[203] A. M. Puertas, M. Fuchs and M. E. Cates, "Simulation study of nonergodicity transitions: Gelation in colloidal systems with short-range attractions," *Phys. Rev. E* **67**, 031406 (2003).

[204] A. M. Puertas, M. Fuchs and M. E. Cates, "Dynamical heterogeneities close to a colloidal gel," *J. Chem. Phys.* **121**, 2813-2822 (2004).

[205] A. M. Puertas, M. Fuchs and M. E. Cates, "Mode Coupling and Dynamical Heterogeneity in Colloidal Gelation: A Simulation Study," *J. Phys. Chem. B* **109**, 6666-6675 (2005).

[206] A. M. Puertas, M. Fuchs and M. E. Cates, "Competition between glass transition and liquid-gas phase separation in attracting colloids," *J. Phys.: Condens. Matter* **19**, 205140 (2007).

[207] A. M. Puertas and G. Odriozola, "Linking Phase Behavior and Reversible Colloidal Aggregation at Low Concentrations: Simulations and Stochastic Mean Field Theory," *J. Phys. Chem. B* **111**, 5564-5572 (2007).

[208] P. N. Pusey, A. D. Pirie and W. C. K. Poon, "Dynamics of colloid-polymer mixtures," *Physica A* **201**, 322-331 (1993).

[209] D. Qi, K. Kwong, K. Rademacher, M. O. Wolf and J. F. Young, "Optical Emission of Conjugated Polymers Adsorbed to Nanoporous Alumina," *Nano Lett.* **3**, 1265-1268 (2003).

[210] G. Rabut and J. Ellenberg, "Automatic real-time three-dimensional cell tracking by fluorescence microscopy," *J. Microsc.* **216**, 131-137 (2005).

[211] I. L. Radtchenko, M. Giersig and G. B. Sukhorukov, "Inorganic Particle Synthesis in Confined Micron-Sized Polyelectrolyte Capsules," *Langmuir* **18**, 8204-8208 (2002).

[212] T. Ragan, H. Huang, P. So and E. Gratton, "3D Particle Tracking on a Two-Photon Microscope," *J. Fluorescence* **16**, 325-336 (2006).

[213] S. Raghavachary, *Rendering for Beginners: Image sythesis using RenderMan* (Oxford: Focal, 2005).

[214] S. Ramakrishnan, M. Fuchs, K. S. Schweizer and C. F. Zukoski, "Entropy driven phase transitions in colloidpolymer suspensions: Tests of depletion theories," *J. Chem. Phys.* **116**, 2201-2212 (2002).

[215] S. Ramakrishnan, M. Fuchs, K. S. Schweizer and C. F. Zukoski, “Concentration Fluctuations in a Model Colloid-Polymer Suspension: Experimental Tests of Depletion Theories,” *Langmuir* **18**, 1082-1090 (2002).

[216] S. Ramakrishnan, Y.-L. Chen, K. S. Schweizer and C. F. Zukoski, “Elasticity and clustering in concentrated depletion gels,” *Phys. Rev. E* **70**, 040401 (2004).

[217] G. Ramírez-Santiago and A. E. González, “Growth laws and spinodal decomposition type of scaling in fractal aggregation of colloids,” *Physica A* **236**, 75-84 (1997).

[218] D. C. Rapaport, *The art of Molecular Dynamic Simulation* (Cambridge: Cambridge UP, 1995).

[219] L. E. Reichl, *A Modern Course in Statistical Physics* 2nd ed. (New York: Wiley, 1998).

[220] G. Roelofs, *PNG: The Definitive Guide* (Sebastopol, CA: O’Reilly, 1999).

[221] S. Roke, O. Berg, J. Buitenhuis, A. van Blaaderen and M. Bonn, “Surface molecular view of colloidal gelation,” *Proc. Nat. Acad. Sci USA* **103**, 13310-13314 (2006).

[222] P. W. Rouw and C. G. de Kruif, “Adhesive hard-sphere colloidal dispersions: Fractal structures and fractal growth in silica dispersions,” *Phys. Rev. A* **39**, 5399-5408 (1989).

[223] P. W. Rouw, A. T. J. M. Woutersen, B. J. Ackerson and C. G. De Kruif, “Adhesive hard sphere dispersions V. Observation of spinodal decomposition in a colloidal dispersion,” *Physica A* **156**, 876-898 (1989).

[224] C. P. Royall, A. A. Louis and H. Tanaka, “Measuring colloidal interactions with confocal microscopy,” *J. Chem. Phys.* **127**, 044507 (2007).

[225] M. Rubinstein and R. H. Colby, *Polymer Physics* (New York: Oxford UP, 2003).

[226] W. B. Russel, D. A. Saville and W. R. Schowalter, *Colloidal Dispersions* (Cambridge: Cambridge UP, 1989).

[227] S. Safran, *Statistical Thermodynamics of Surfaces, Interfaces and Membranes* (Boulder: Westview, 1994).

[228] T. Sakai, Y. Takeda, F. Mafune, M. Abe and T. Kondow, “Dye Transfer between Surfactant-Free Nanodroplets Dispersed in Water,” *J. Phys. Chem. B* **106**, 5017-5021 (2002).

[229] T. Sakai, Y. Takeda, F. Mafune, M. Abe and T. Kondow, “Monitoring Growth of Surfactant-Free Nanodroplets Dispersed in Water by Single-Droplet Detection,” *J. Phys. Chem. B* **107**, 2921-2926 (2003).

[230] L. Sangaletti, S. Pagliara, F. Parmigiani, P. Galinetto, R. Larciprete, S. Lizzit and A. Goldoni, “Carbon nanotube bundles and thin layers probed by micro-Raman spectroscopy,” *Euro. Phys. J. B* **31**, 203-208 (2003).

[231] N. Sanz, A. C. Gaillot, Y. Usson, P. L. Baldeck and A. Ibanez, “Organic nanocrystals grown in sol-gel coatings,” *J. Mater. Chem.* **10**, 2723-2726 (2000).

[232] R. Savic, L. Luo, A. Eisenberg and D. Maysinger, “Micellar Nanocontainers Distribute to Defined Cytoplasmic Organelles,” *Science* **300**, 615-618 (2003).

[233] P. Schall, I. Cohen, D. A. Weitz and F. Spaepen, “Visualization of Dislocation Dynamics in Colloidal Crystals,” *Science* **305**, 1944-1948 (2004).

[234] P. Schall, I. Cohen, D. A. Weitz and F. Spaepen, “Visualizing dislocation nucleation by indenting colloidal crystals,” *Nature* **440**, 319-323 (2006).

[235] P. Schall, D. A. Weitz and F. Spaepen, “Structural Rearrangements That Govern Flow in Colloidal Glasses,” *Science* **318**, 1895-1899 (2007).

[236] M. Schmidt, M. Nagorni and S. W. Hell, “Subresolution axial distance measurements in far-field fluorescence microscopy with precision of 1 nanometer,” *Rev. Sci. Instruments* **71**, 2742-2745 (2000).

[237] M. Schrader, S. W. Hell and H. T. M. van der Voort, “Potential of confocal microscopes to resolve in the 50-100 nm range,” *Appl. Phys. Lett.* **69**, 3644-3646 (1996).

[238] M. Schrader, M. Kozubek, S. W. Hell and T. Wilson, “Optical transfer functions of 4Pi confocal microscopes: theory and experiment,” *Opt. Lett.* **22**, 436-438 (1997).

[239] M. Schrader, S. W. Hell and H. T. M. van der Voort, “Three-dimensional superresolution with a 4Pi-confocal microscope using image restoration,” *J. Appl. Phys.* **84**, 4033-4042 (1998).

[240] M. Schrader, K. Bahlmann, G. Giese and S. W. Hell, “4Pi-Confocal Imaging in Fixed Biological Specimens,” *Biophys. J.* **75**, 1659-1668 (1998).

[241] F. Sciortino and P. Tartaglia, “Structure Factor Scaling during Irreversible Cluster-Cluster Aggregation,” *Phys. Rev. Lett.* **74**, 282-285 (1995).

[242] F. Sciortino, A. Belloni and P. Tartaglia, “Irreversible diffusion-limited cluster aggregation: The behavior of the scattered intensity,” *Phys. Rev. E* **52**, 4068-4079 (1995).

[243] F. Sciortino, “Disordered materials - One liquid, two glasses,” *Nat. Nater.* **1**, 145-146 (2002).

[244] F. Sciortino, S. Mossa, E. Zaccarelli and P. Tartaglia, “Equilibrium Cluster Phases and Low-Density Arrested Disordered States: The Role of Short-Range Attraction and Long-Range Repulsion,” *Phys. Rev. Lett.* **93**, 055701 (2004).

[245] F. Sciortino and P. Tartaglia, “Glassy colloidal systems,” *Adv. Phys.* **54**, 471-524 (2005).

[246] R. P. Sear and W. M. Gelbart, “Microphase separation versus the vapor-liquid transition in systems of spherical particles,” *J. Chem. Phys.* **110**, 4582-4588 (1999).

[247] R. P. Sear, “Absence of the liquid phase when the attraction is not pairwise additive,” *Phys. Rev. E* **61**, 651-655 (2000).

[248] R. P. Sear, “Fluid-fluid transitions of hard spheres with a very-short-range attraction,” *Phys. Rev. E* **61**, 6019-6022 (2000).

[249] P. N. Segrè, V. Prasad, A. B. Schofield and D. A. Weitz, “Glasslike Kinetic Arrest at the Colloidal-Gelation Transition,” *Phys. Rev. Lett.* **86**, 6042-6045 (2001).

[250] J.-M. Segura, A. Renn and B. Hecht, “A sample-scanning confocal optical microscope for cryogenic, operation,” *Rev. Sci. Instruments* **71** 1706-1711 (2000).

[251] S. A. Shah, Y. L. Chen, K. S. Schweizer and C. F. Zukoski, “Phase behavior and concentration fluctuations in suspensions of hard spheres and nearly ideal polymers,” *J. Chem. Phys.* **118**, 3350-3361 (2003).

[252] S. A. Shah, Y.-L. Chen, K. S. Schweizer and C. F. Zukoski, “Viscoelasticity and rheology of depletion flocculated gels and fluids,” *J. Chem. Phys.* **119**, 8747-8760 (2003).

[253] S. A. Shah, S. Ramakrishnan, Y. L. Chen, K. S. Schweizer and C. F. Zukoski, “Scattering Studies of the Structure of Colloid-Polymer Suspensions and Gels,” *Langmuir* **19**, 5128-5136 (2003).

[254] C. J. R. Sheppard and D. M. Shotton, *Confocal Laser Scanning Microscopy* (Oxford: BIOS Scientific Publishers, 1997).

[255] W. Y. Shih, I. A. Aksay and R. Kikuchi, “Reversible-growth model: Cluster-cluster aggregation with finite binding energies,” *Phys. Rev. A* **36**, 5015-5019 (1987).

[256] W.-H. Shih, W. Y. Shih, S.-I. Kim, J. Liu and I. A. Aksay, “Scaling behavior of the elastic properties of colloidal gels,” *Phys. Rev. A* **42**, 4772-4779 (1990).

[257] E. D. Siggia, “Late stages of spinodal decomposition in binary mixtures,” *Phys. Rev. A* **20**, 595-605 (1979).

[258] D. Silvano, S. Krol, A. Diaspro, O. Cavalleri and A. Gliozzi, “Confocal Laser Scanning Microscopy to Study Formation and Properties of Polyelectrolye Nanocapsules Derived From $CdCO_3$ Templates,” *Microsc. Res. Tech.* **59**, 536-541 (2002).

[259] J. T. Soini, M. Schrader, P. E. Hänninen and S. W. Hell, “Image formation and data acquisition in a stage scanning 4Pi confocal fluorescence microscope,” *Appl. Opt.* **36**, 8929-8932 (1997).

[260] M. Sperl, “Dynamics in colloidal liquids near a crossing of glass- and gel-transition lines,” *Phys. Rev. E* **69**, 011401 (2004).

[261] I. Stephenson, *Essential RenderMan*, 2nd ed. (London: Springer, 2007).

[262] P. K. Stoimenov, R. L. Klinger, G. L. Marchin and K. J. Klabunde, “Metal Oxide Nanoparticles as Bactericidal Agents,” *Langmuir* **18**, 6679-6686 (2002).

[263] A. Stradner, H. Sedgwick, F. Cardinaux, W. C. K. Poon, S. U. Egalhaaf and P. Schurtenberger, “Equilibrium cluster formation in concentrated protein solutions and colloids,” *Nature* **432**, 492-495 (2004).

[264] D. Stripinis, *The MEL Companion: Maya Scripting for 3D Artists* (Hingham, MA: Charles River Media, 2003).

[265] A. S. Susha, F. Caruso, A. L. Rogach, G. B. Sukhorukov, A. Kornowski, H. Mohwald, M. Giersig, A. Eychmuller and H. Weller, “Formation of luminescent spherical core-shell particles by the consecutive adsorption of polyelectrolye and CdTe(S) nanocrystals on latex colloids,” *Coll. Surf. A* **163**, 39-44 (2000).

[266] H. Tanaka, Y. Nishikawa and T. Koyama, “Network-forming phase separation of colloidal suspensions,” *J. Phys.: Condens. Matter* **17**, L143-L153 (2005).

[267] H. Tanaka, T. Araki, T. Koyama and Y. Nishikawa, “Universality of viscoelastic phase separation in soft matter,” *J. Phys.: Condens. Matter* **17**, S3195-S3204 (2005).

[268] Y. Tang and B. C. Y. Lu, “Improved expressions for the radial distribution function of hard spheres,” *J. Chem. Phys.* **103**, 7463-7470 (1995).

[269] Y. Tang and B. C. Y. Lu, “Direct calculation of radial distribution function for hard-sphere chains,” *J. Chem. Phys.* **105**, 8262-8265 (1996).

[270] Y. Tang and B. C. Y. Lu, “Analytical representation of radial distribution function for classical fluids,” *Mol. Phys.* **90**, 215-224 (1997).

[271] S. Taylor, *Intel Integrated Performance Primitives.* (Hillsboro, OR: Intel, 2004).

[272] T. Tesfamichael, G. Will, T. Bostrom and J. Bell, “Investigations of dye-sensitised titania solar cell electrode using confocal laser scanning microscopy,” *J. Mat. Sci.* **38**, 1721-1726 (2003).

[273] J. Tittel, W. Göhde, F. Koberling, Th. Basché, A. Kornowski, H. Weller and A. Eychmüller, “Fluorescence Spectroscopy on Single CdS Nanocrystals,” *J. Phys. Chem. B* **101**, 3013-3016 (1997).

[274] V. Trappe, V. Prasad, V., L. Cipelletti, P. N. Segre and D. A. Weitz, “Jamming phase diagram for attractive particles,” *Nature* **411**, 772-775 (2001).

[275] V. Trappe, P. Sandkühler, “Colloidal gels—low-density disordered solid-like states,” *Curr. Opin. Colloid Interface Sci.* **8**, 494-500 (2004).

[276] R. Tuinier, J. Rieger and C. G. de Kruif, “Depletion-induced phase separation in colloidpolymer mixtures,” *Adv. Colloid Interface Sci.* **103**, 1-31 (2003).

[277] S. Upsill, *The RenderMan Companion: A Programmer’s Guide to Realistic Computer Graphics* (Reading, MA: Addison-Wesley, 1990).

[278] A. van Blaaderen, R. Ruel and P. Wiltzius, “Template-directed colloidal crystallization,” *Nature* **385**, 321-324 (1997).

[279] A. van Blaaderen and P. Wiltzius, “Real-space structure of colloidal hard-sphere glasses,” *Science* **270**, 1177-1179 (1995).

[280] F. W. Van Keuls, R. R. Romanofsky, N. D. Varaljay, F. A. Miranda, C. L. Canedy, S. Aggarwal, T. Venkatesan and R. Ramesh, “A Ku-Band Gold/$Ba_xSr_{1-x}TiO_3$/$LaAlO_3$ Conductor/Thin Film Ferroelectric Microstrip Line Phase Shifter for Room-Temperature Communications Applications,” *Microw. Opt. Technol. Lett.* **20**, 53-56 (1999).

[281] W. G. J. H. M. van Stark, P. L. T. M. Frederix, D. J. Van den Heuvel and H. C. Gerritsen, “Photooxidation and Photobleaching of Single CdSe/ZnS Quantum dots Probed by Room-Temperature Time-Resolved Spectroscopy,” *J. Phys. Chem. B* **105**, 8281-8284 (2001).

[282] P. Varadan and M. J. Solomon, “Direct Visualization of Long-Range Heterogeneous Structure in Dense Colloidal Gels,” *Langmuir* **19**, 509-512 (2003).

[283] H. Verduin and J. K. G. Dhont, “Phase Diagram of a Model Adhesive Hard-Sphere Dispersion,” *J. Colloid Interface Sci.* **172**, 425-437 (1995).

[284] N. A. M. Verhaegh D. Asnaghi, H. N. W. Lekkerkerker, M. Giglio and L. Cipelletti, “Transient gelation by spinodal decomposition in colloid-polymer mixtures,” *Physica A* **242**, 104-118 (1997).

[285] E. J. W. Verwey and J. Th. G. Overbeek, *Theory of the Stability of Lyophobic Colloids* (New York: Elsevier, 1948).

[286] R. L. C. Vink and J. Horbach, “Grand canonical Monte Carlo simulation of a model colloid-polymer mixture: Coexistence line, critical vehavior and interfacial tension,” *J. Chem. Phys.* **121**, 3253-3258 (2004).

[287] R. L. C. Vink and J. Horbach, “Critical behavior and interfacial fluctuations in a phase-separating model colloid-polymer mixture: grand canonical Monte Carlo simulations,” *J. Phys.: Condens. Matter* **16**, S3807-S3820 (2004).

[288] R. L. C. Vink, J. Horbach and K. Binder, “Critical phenomena in colloid-polymer mixtures: Intefacial tension, order parameter, susceptibility, and coexistence diameter,” *Phys. Rev. E* **71**, 011401 (2005).

[289] G. A. Vliegenthart, J. F. M. Lodge and H. N. W. Lekkerkerker, “Strong Weak and Metastable Liquids Structural and Dynamical Aspects of the Liquid State,” *Physica A* **263**, 378-388 (1999).

[290] A. Vrij, M. H. G. M. Penders, P. W. Rouw, C. G. Kruif, J. K. G. Dhont, C. Smits and H. N. W. Lekkerkerker, “Phase-transition phenomena in colloidal systems with attractive and repulsive particle interactions,” *Faraday Discuss. Chem. Soc.* **90**, 31-40 (1990).

[291] E. Wang, C. M. Babbey and K. W. Dunn, “Performance comparison between the high-speed Yokogawa spinning disc confocal system and single-point scanning confocal systems,” *J. Microsc.* **218**, 148-159 (2005).

[292] D. A. Weitz and M. Oliveria, “Fractal Structures Formed by Kinetic Aggregation of Aqueous Gold Colloids,” *Phys. Rev. Lett.* **52**, 1433-1436 (1984).

[293] D. A. Weitz, J. S. Huang, M. Y. Lin and J. Sung, “Dynamics of Diffusion-Limited Kinetic Aggregation,” *Phys. Rev. Lett.* **53**, 1657-1660 (1984).

[294] D. A. Weitz, J. S. Huang, M. Y. Lin and J. Sung, “Limits of the Fractal Dimension for Irreversible Kinetic Aggregation of Gold Colloids,” *Phys. Rev. Lett.* **54**, 1416-1419 (1985).

[295] D. A. Weitz and M. Y. Lin,“Dynamic Scaling of Cluster-Mass Distributions in Kinetic Colloid Aggregation,” *Phys. Rev. Lett.* **57**, 2037-2040 (1986).

[296] E. R. Weeks, J. C. Crocker, A. C. Levitt, A. Schofield and D. A. Weitz, “Three-Dimensional Direct Imaging of Structural Relaxation Near the Colloidal Glass Transition,” *Science* **287**, 627-631 (2000).

[297] S. R. Whaley, D. S. English, E. L. Hu, P. F. Barbara and A. M. Belcher, “Selection of peptides with semiconductor binding specificity for directed nanoparticle assembly,” *Nature* **405**, 665-668 (2000).

[298] M. E. Wickham, M. Rug, S. A. Ralph, N. Klonis, G. I. McFadden, L. Tilley and A. F. Cowman, “Trafficking and assembly of the cytoadherence complex in Plasmodium falciparum-infected human erythrocytes,” *EMBO J.* **20**, 5636-5649 (2001).

[299] M. R. Wilkins and C. Kazmier, *MEL Scripting for MAYA Animators* (San Francisco: Morgan Kaufman, 2005).

[300] H. M. Wyss, E. V. Tervoort and L. J. Gauckler, “Mechanics and Microstructures of Concentrated Particle Gels,” *J. Am. Ceram. Soc.* **88**, 2337-2348 (2005).

[301] X. S. Xie, J. Yu and W. Y. Yang, “Living Cells as Test Tubes,” *Science* **312**, 228-230 (2006).

[302] J. A. Yanez, E. Laarz and L. Bergström, “Viscoelastic Properties of Particle Gels,” *J. Colloid Interface Sci.* **209**, 162-172 (1999).

[303] A. Yethiraj and A. V. Blaaderen, “A colloidal model system with an interaction tunable from hard sphere to soft and dipolar,” *Nature* **421**, 513-517 (2003).

[304] E. Zaccarelli, “Colloidal Gels: Equilibrium and Non-Equilibrium Routes,” *J. Phys.: Condens. Matter* **19**, 323101 (2007).

[305] A. S. Zackrisson, A. Martinelli, A. Matic and J. Bergenholtz, “Concentration effects on irreversible colloid cluster aggregation and gelation of silica dispersions,” *J. Colloid Interface Sci.* **301**, 137-144 (2006).

[306] J. Zhao, C. Jiang, Y. Fan, M. Burghard, T. Basché and A. Mews, “Diameter-Dependent Combination Modes in Individual Single-Walled Carbon Nanotubes,” *Nano Lett.* **2**, 823-826 (2002).