Coarsening dynamics of domains in lipid membranes

Cynthia Ann Stanich

A dissertation

submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

University of Washington

2012

Reading Committee:

Sarah L. Keller, Chair

Phillip J. Reid

Kenneth Krohn

Program Authorized to Offer Degree:

Department of Chemistry

University of Washington

Abstract

Coarsening dynamics of domains in lipid membranes

Cynthia Ann Stanich

Chair of the Supervisory Committee:
Professor Sarah L. Keller
Department of Chemistry

We investigate isothermal diffusion and growth of micron-scale liquid domains within membranes of free-floating giant unilamellar vesicles (GUVs) with diameters between 80μm and 250μm. Domains appear after a rapid temperature quench, when the membrane is cooled through a miscibility phase transition such that coexisting liquid phases form. In membranes quenched far from a miscibility critical point, circular domains nucleate and then progress within seconds to "late stage" coarsening in which domains grow via two mechanisms: (1) collision and coalescence of liquid domains, and (2) Ostwald ripening. Both mechanisms are expected to yield the same growth exponent, $\alpha = 1/3$, where domain radius grows as time$^{\alpha}$. We measure $\alpha = 0.28 \pm 0.05$, in excellent agreement. In membranes close to a miscibility critical point, the two liquid phases in the membrane are bi-continuous. A quench near the critical composition results in rapid changes in morphology of elongated domains. In this case, we measure $\alpha = 0.50 \pm 0.16$, consistent with theory and simulation.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Acknowledgements

This work would not have been possible without the encouragement from so many wonderful professors over the years, not all of who I can mention here. They have fostered in me a desire to mentor my own students and share with them the thrill of science. I would especially like to thank Sarah Keller. I am grateful for her enthusiasm, mentorship, and her insistence that I follow my passion. Science is never work when Sarah is around, her joy is infectious in the laboratory and in the classroom. The first question I ask myself when presenting is, “How would Sarah say it?” I would also like to thank Aurelia R. Honkerkamp-Smith for making huge, beautiful vesicles for my study. My success at Matlab coding is thanks to the help of Pietro Cicuta, Sarah Veatch, Aurelia R. Honerkamp-Smith, and Matthew Blosser. I want to thank my friends and labmates for helping me over the years, particularly the undergraduates without whom I would not have been so productive, including Chris Warth, Andrea Lamprecht, Thien-An Hua, Pokuan (Paul) Ho, and Peter Holmes.

This work is dedicated to my undergraduate research advisor, Robert W. Wiseman, for opening the door to the world of research for me. To him I am eternally grateful.

COARSENING DYNAMICS OF DOMAINS IN LIPID MEMBRANES

I. Introduction

Lipid membranes as 2D liquids

I study the diffusion dynamics of domains in lipid bilayers. Lipids, which have hydrophobic tails and hydrophilic heads, self-assemble into a 2-dimensional sheet of a membrane. Giant unilamellar vesicles (GUVs) are spherical lipid bilayer membranes with water inside and outside. GUVs are used as simplified model systems to study lipids in membranes without interference from other components typically found in cell membranes, such as proteins and the cytoskeletal network. We study GUVs composed of mixtures of three lipid types: 1) phospholipids with high melting temperatures, 2) phospholipids with low melting temperatures, and 3) cholesterol. For the work here, our vesicles are 80-250 μm in diameter. The lipids in vesicle membranes exhibit interesting miscibility phase behavior that has implications in biological function of cell membranes [Veatch et al. 2008]. I study the dynamics of this phase separation process.

Most people are familiar with miscibility phase separation in 3-dimensions, such as the separation of oil and water. Consider shaking a bottle of salad dressing composed of oil and vinegar: if the oil is the minority phase, many small droplets of oil diffuse throughout the vinegar, collide with each other, and then coalesce into larger oil droplets. This process continues until the bottle contains one single contiguous volume of vinegar separated by a single interface from one volume of oil. Intermolecular interactions drive this separation. Individual water molecules have lower energy when surrounded by other water molecules, and oil molecules have lower energy when surrounded by oil molecules. The energy of the whole system is lowered by separating the molecules into two different phases, and by decreasing the boundary between the two phases. GUV membranes are unique because the lipids behave as a

This section details research conducted in collaboration with Aurelia R. Honerkamp-Smith, Gregory Garbès Putzel, Christopher S. Warth, Pritam Mandal, Elizabeth Mann, Thien-An D. Hua, Andrea K. Lamprecht, and Sarah L. Keller. Parts of this chapter are excerpted from a manuscript in preparation and from Chapter VI of the doctoral dissertation of Aurelia R. Honerkamp-Smith.

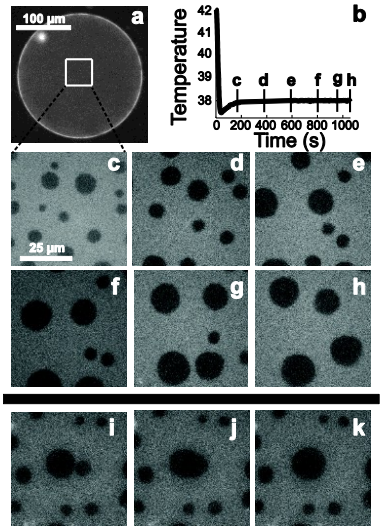

Figure 1: Temperature quench of a giant unilamellar vesicle with an area fraction of liquid ordered (dark) phase less than 0.3. Panel a is a fluorescence micrograph focused at the equator of the vesicle. Upon a temperature quench as in panel b, the lipids separate into two distinct liquid phases. Panel c - h show frames of a movie during constant temperature and corresponds to time points c-h in panel b. Frames i - k illustrate an example of domain collision and coalescence. Frames i – k are from a different vesicle than a – h.

quasi-2-dimensional liquid in the plane of the bilayer. It is not a purely 2-dimensional liquid because there is water inside and outside of the vesicle with which the lipids can couple, but the membrane still behaves differently from a 3-dimensional system.

At high temperatures, the lipids in the bilayer are well mixed in one phase. After the temperature is quenched, a miscibility transition occurs, which means that the lipids separate into two liquid phases. Multiple micron-scale liquid domains nucleate, undergo Brownian diffusion, and coarsen [Veatch & Keller 2003] (Figure 1). The historical names given to the two coexisting liquid phases in the membranes are “liquid-disordered” (L_d) and “liquid-ordered” (L_o). Unless the system is close to a miscibility critical point, the boundary between phases has a significant line tension (>1 pN) such that the free energy of the entire system is lowest when the length of all boundaries is minimized [Honerkamp-Smith et al. 2008, Esposito et al. 2007]. Therefore, when domains collide, they coalesce. This decreases the number of domains while increasing the average size of the domains, and therefore decreases the overall boundary length between the two phases. I measure spatial properties of the domains using image analysis of videos collected by fluorescence microscopy by Dr. Aurelia Honerkamp-Smith. It is possible to image the two different phases because a fluorescent dye added to the membrane partitions preferentially into the more disordered of the two phases. The evolution of domain number, size, and boundary is expected to be described by a growth exponent if the system

falls within a scaling regime.

Theory

Diffusion

Prediction of the exponent of radius growth, α, is based on a diffusion coefficient, $D(r)$, of a domain of radius r that varies as $D \propto {}^{1}/_{r}$. I measure diffusion coefficients of domains with a range of sizes. In detail, I track domains for ten frames (5 seconds) to measure their mean square displacements and radii. Diffusion of a disk embedded in a membrane bounded by bulk solution on either side has been solved analytically in two limiting cases. The two cases are distinguished by a length scale, L_h, characterized by the ratio of the 2-D membrane viscosity, η_m, to the viscosity of the bulk liquid, η_{bulk}. The first case considers domains of small radius and/or membranes of high viscosity [Saffman & Delbruck 1975]. In this limit, the diffusion coefficient, $D(r)$, is:

$$D(r) = \frac{k_B T}{4\pi\eta_m}\left[\ln\frac{\eta_m}{r\eta_{bulk}} - \gamma + \frac{1}{2}\right] \qquad \text{(eq 1)}$$

where γ is the Euler constant, $\gamma = 0.5572$. In the opposite limit of large domain radius and/or low membrane viscosity [Hughes et al. 1981]:

$$D(r) = \frac{k_B T}{16\pi\eta_{bulk} r} \quad \text{Or} \quad \ln D(r) = -(constants)(\ln r) \qquad \text{(eq 2)}$$

A small correction was made to equation 2 by assuming that the single inclusion is fluid, with the same viscosity as the infinite planar membrane through which it travels [De Koker 1996, Seki 2011]. That correction yields:

$$D(r) = \frac{2k_B T}{3\pi^2\eta_{bulk} r}. \qquad \text{(eq 3)}$$

For intermediate cases, the approximation below is valid [Petrov & Schwille 2008]:

$$D(r) = \frac{k_B T}{4\pi\eta_m}\left[\frac{\ln\frac{2\eta_m}{r\eta_{bulk}} - \gamma + \frac{4r\eta_{bulk}}{\pi\eta_m} - \frac{r^2\eta_{bulk}^2}{2\eta_m^2}\ln\frac{2\eta_m}{r\eta_{bulk}}}{1 - \frac{(r\eta_{bulk})^3}{\pi\eta_m^3}\ln\frac{2\eta_m}{r\eta_{bulk}} + \frac{c_1\left(r\eta_{bulk}/\eta_m\right)^{b_1}}{\left(1+c_2\left(r\eta_{bulk}/\eta_m\right)^{b_2}\right)}}\right], \qquad \text{(eq 4)}$$

where $c_1 = 0.73761;\ b_1 = 2.74819;\ c_2 = 0.52119;\ b_2 = 0.6145.$ The fitting parameters $c_{1,2}$and $b_{1,2}$ are determined through minimization of the square of the residuals over 161 different values of the reduced radius, $r(\eta_{bulk}/\eta_m)$ [Petrov & Schwille 2008]. The equations above illustrate how a change in the viscosity of the bulk liquid should result in a change in

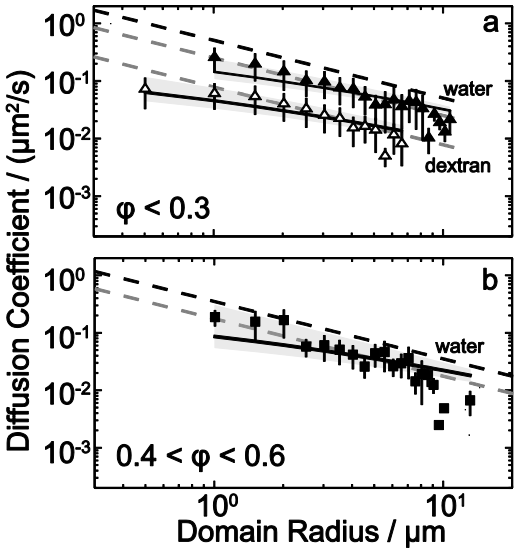

Figure 2: Diffusion coefficients versus binned domain radii for vesicles for which the L_o phase covers an **(a)** area fraction, φ, less than 0.3 and **(b)** area fraction between 0.4 and 0.6. In this plot, only domains that are circular and/or grow predominantly by collision and coalescence are represented. For φ < 0.3, domains are circular for all times analyzed. For 0.4 < φ < 0.6, domains are circular only at long times. All relations that satisfy $D \propto 1/r$ appear as a line with a slope of –1 on this log-log plot. In both panels, dashed black lines are fits to $D(r) = 2k_BT / 3\pi^2\,\eta_{3D}\,r$ with no free parameters, where T is the average final temperature of all quenches. Black curves are best fits to Eq 4 [Petrov & Schwille 2008], and shaded regions are 95% confidence intervals of those fits. Vertical uncertainties are standard deviations. Measurement uncertainty in domain size is at most two pixels, or 0.4 microns. Data is binned every 1 μm in domain diameter. **(a)** Filled vs. open triangles represent vesicles diluted into water vs. dextran solution. T = 40.3˚C and $\eta_{3D} = \eta_{\text{water at 40.3°C}} = 0.652 \times 10^{-3}$ Pa s [Kestin 1978]. The upper dashed grey line is offset by a factor 2. The lower dashed grey line is offset by an additional factor of 3.2. **(b)** Squares represent diffusion coefficients of nearly circular liquid domains in vesicles with area fraction $0.4 < \phi < 0.6$ and compositions between 25:20:55 and 30:20:50. T = 26.2˚C and $\eta_{3D} = \eta_{\text{water at 29°C}} = 0.864 \times 10^{-3}$ Pa s. The dashed grey line is offset by a factor 2.

domain diffusion coefficient, which will in turn affect domain growth kinetics.

Growth exponent

Off-critical collision & coalescence

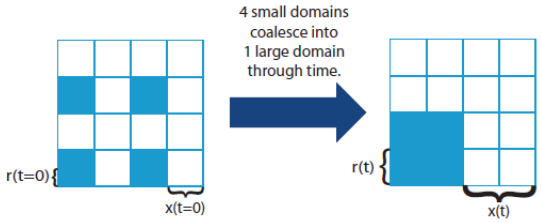

Figure 3: A schematic of a membrane with an area fraction of dark domains of ¼. Each domain must diffuse a distance that scales linearly with its radius to reach and collide with another domain. If area is constant, the average radius increases and the perimeter of the shaded area decreases proportionately.

Here I consider the growth of domains that diffuse, collide, and coalesce, as illustrated by a simple example. The sketch in Figure 3 shows coalescence of 4 domains. The area fraction of the total membrane covered by domains is constant. Let x equal the minimum distance that one domain must travel before it collides with another domain. Let r equal the domain size (radius). From Figure 3, $x \propto r$. Domains diffuse by Brownian motion in *2*-dimensions as $\langle x^2 \rangle = 4Dt$, where t is time and D is the diffusion coefficient. From my measurements of diffusion coefficients, I know that it is roughly true that $D \propto 1/r$ for my system (Figure 2). Therefore, the equation $\langle x^2 \rangle = 4Dt$ becomes $r^2 \propto (1/r)t$, or $r \propto t^{1/3}$. In other words, I expect the average domain radius to grow with an exponent of α = 1/3 when domains are circular and diffusing randomly. Notice that within figure 3 the area fraction of shaded domains stays constant as the domains merge. As the average radius increases, the average perimeter decreases at an equal but opposite rate.

Off-critical Ostwald ripening

By eye, we find that collision and coalescence of domains is the dominant mechanism for domain growth through time in vesicles. Another possible mechanism is Ostwald ripening, in which individual lipids detach from the perimeter of a small domain, diffuse across the membrane, and join with a larger domain. Ostwald ripening is also known as evaporation condensation. Ostwald ripening of liquid domains has been observed in membranes on a solid

support [Garcia-Saez et al. 2008], a system in which collective motion of lipids is hindered [Przybylo et al. 2006, Stottrup et al. 2004]. Ostwald ripening is expected to give a growth exponent of α = 1/3 in a 2-dimensional system [Gomez et al. 2008] (the same value as for collision and coalescence). The hallmark of Ostwald ripening would be that the differences in sizes between neighboring large and small domains would increase through time in the absence of domain collision and merging. Most of the data I have analyzed here show constant domain sizes between merging events, although I can anecdotally report that some small domains under the resolution of my thresholding program appear to shrink and disappear during the course of an experiment. Although these Ostwald ripening events may be contributing to the increase in average size of domains, the main contributor to the 1/3 exponent I measure is collision and coalescence.

Literature values for growth exponents of circular domains

During domain coarsening, the average radius of domains increases with time as t^{α}, where α is the power law exponent of radius growth. For the moment, consider only the case in which circular, liquid domains comprise a small area fraction, ϕ, of the entire vesicle surface. For coarsening of micron-sized domains based on only collision and coalescence, the exponent α is usually predicted to be 1/3 [Saeki et al. 2006, Laradji & Sunil Kumar 2004, Taniguchi 1996, Gomez et al. 2008, Camley & Brown 2011], and in one case has been predicted to be 1/4 [Yanagisawa et al. 2007]. To date, published experiments have not reproduced expected values of α = 1/3 for micron-sized (and larger) domains, as shown in the table 1. In this dissertation, I

Table 1: Growth exponents reported in the literature.

Theory $r \propto t^{\alpha}$	Assumptions	Experiment α	System	Reference
$radius \propto t^{1/3}$	$D \propto 1/r$	0.15	Unilamellar vesicles, diameter ~50 µm	Saeki et al. 2005
$\propto t^{1/4}$	$D \propto 1/r^2$	$2/3$	Unilamellar vesicles, diameter = 10-150 µm	Yanagisawa et al. 2007
$\propto t^{1/3}$	$D \propto 1/r$	$1/3$ * 1 **	Unilamellar vesicles, diameter < 20 µm	Liang et al. 2010 ***

* For domain sizes ≤ 1 µm. ** For domain sizes ≥ µm. *** Temperature is not constant.

present an independent measurement of the power law exponent with the goal of understanding the discrepancy between predicted and measured values. I intend to clear up the discrepancy in the literature by carefully measuring domain growth on vesicles by avoiding the pitfalls I see in the literature.

A difficult experimental aspect of measuring growth exponents is that only a small percentage of the surface of the vesicle is in focus at one time. This can lead to low statistics and undercounting of large domains even in the very large vesicles that we employ. Undercounting large domains would lead to a lower average radius in time, making the measured growth exponent lower than it would be if it were possible to measure the large domains. Saeki et al. find a growth exponent of 0.15 on small vesicles with radii on the order of only 10 μm (Table 1), for which any large domain would continue outside of the area being studied and therefore not measured. We avoid problems of undercounting that are inherent with measuring domain radius by instead measuring a normalized domain size, *R*, which is defined as the area of the minority phase divided by the total perimeter between the two phases measured in each frame. Another factor that would lead to a low growth exponent would be if the domains have hindered kinetics [Ursell et al. 2009]. This would happen if the domains were bulging out of the membrane (Figure 4), which was reported by Saeki et al. to be the case with some of their vesicles. We avoid hindered kinetics by creating a slight osmotic pressure inside of the vesicle.

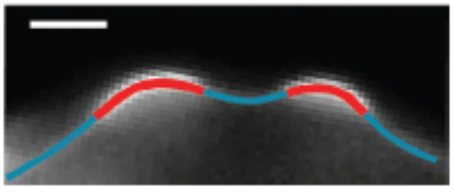

Figure 4: Part of a figure by Ursell et al. 2009 showing the bending of the membrane (blue) between two neighboring domains (red).

Yanagisawa et al. find a growth exponent of 2/3 for the size of domains in time when collision is not kinetically hindered. Their measured value is larger than theory expects. A possible explanation of this high growth exponent could be that their compositions are near critical and that ϕ ~ 1/2, since they suggest spinodal decomposition happens at early times and then the

domains become circular. Simulations discussed below show growth exponents for membranes with ϕ = 1/2 are expected to be around α = 1/2. As I will show in this dissertation, circular domains in membranes with ϕ = 1/2 grow with an exponent of 1/3. Another reason Yanagisawa et al. could have measured a high growth exponent could be due to photo-oxidation of the double bonds in dioleoylphosphatidylcholine (DOPC). This will increase the miscibility temperature of the system. To avoid this problem we use diphytanoylphosphatidylcholine (DiphyPC), which has methyl groups on the lipids.

Finally, it is important to look at coarsening at a constant temperature. For example, Liang et al. report a growth exponent of 1/3 for domains $\leq$ 1 μm. Although this value is in good agreement with the theoretically expected exponent for the diffusion regime of domains on vesicles, Liang et al. are looking at a system in which the temperature is changing during the coarsening process and their measured growth exponents of domains $\geq$ 1 μm changes to α = 1. Comparison with the values reported by Saeki et al. and Yanagisawa et al. is not straightforward.

Simulations

Off-critical

In order to place my experimental results in the context of existing literature, here I will review simulations performed by others. Coarsening proceeds via two mechanisms, which have both been termed "coalescence" in the literature. In the first mechanism, "grains" (domains)

Table 2: Simulation results for diffusion and coalescence of circular domains.

α	Model	Reference
0.3	Dissipative particle dynamics. ϕ = 0.3	Taniguchi 1996
$^1/_3$	Purely dissipative dynamics. ϕ = 0.3	Laradji et al. 2004
0.31	200 spherical caps on a vesicle. ϕ = 0.09	Putzel (Northwestern U.)
$^1/_3$	Continuum approach with hydrodynamics	Fan et al. 2010
$^1/_3$	Stochastic phase field model + hydrodynamics	Camley et al. 2010

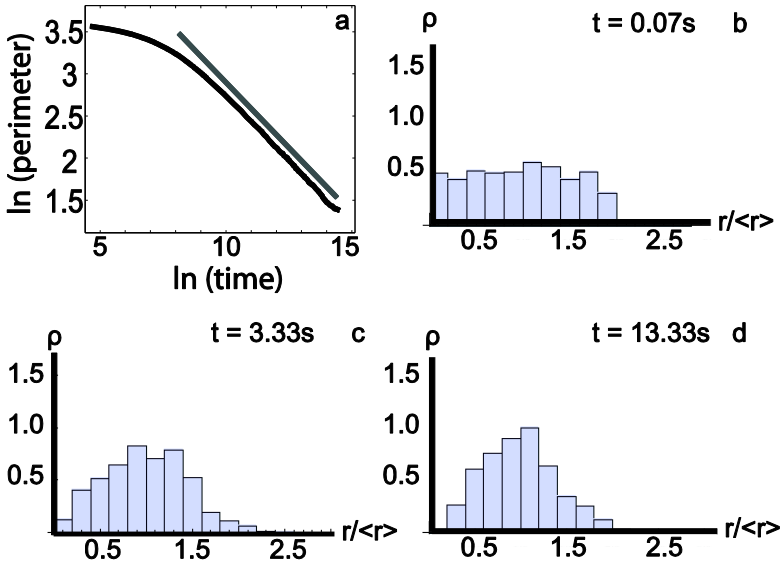

Figure 5: **(a)** Log-log plot of average arc length from the center of a domain to the edge versus time. The black line represents the results of the simulation described in the text. The grey line shows perimeter $\propto t^{-0.31}$ for comparison. The radius of the vesicle is set to 1. **(b-d)** Histograms of domain sizes normalized to the average for different time. Simulations and graphs by Greg Putzel (Northwestern Univ. Physics).

smaller than a critical size dissolve into the background fluid, and large grains accrete material [Pitaevskii & Lifshitz 1981]. This mechanism is called "condensation-evaporation" or Ostwald-Ripening. In this mechanism, domains would grow even if they did not diffuse. During the "early stage", the distribution of domain areas evolves, normalized with respect to an average domain area. We do not intend to study the early stage, which occurs during the interval we allow for temperature equilibration. Roughly sixty seconds elapse between the time Aurelia instigates a quench of 2^{o}C and when she begins acquiring images. Moreover, we analyze too few domains in each vesicle to determine whether the shapes of distributions of domain sizes are changing. During the "late stage", the normalized distribution remains static, although the average domain size increases [Pitaevskii & Lifshitz 1981].

In the second mechanism, domains grow by colliding and merging. Diffusion of domains must occur in order for them to grow. As I will discuss in this dissertation, I measure different growth exponents depending on the area fraction of the vesicle, which is either ϕ < 0.3 or 0.4 <

ϕ < 0.6. Several simulations have been performed for low minority phase area fractions, as listed below and seen in table 3.

(1) In unpublished work, Dr. Greg G. Putzel (Northwestern University, Physics) seeded a spherical surface with ~ 200 spherical caps and allowed the caps to diffuse across the surface of the sphere with a diffusion coefficient proportional to 1/r, where r is the arc length from the center of the cap to the edge. His area fraction was set to 9%. Greg's first simulation results are shown in figure 5a. Greg also ran a simulation of domain sizes on a vesicle with an area fraction set to 1.5%. In an early stage, the shape of the distribution of normalized domain sizes changes dramatically (Figure 5b-d). At a later stage, the growth exponent α = 0.31. This is followed by a terminal stage in which the total number of caps on the spherical surface is very small.

(2) Laradji and Sunil Kumar have conducted DPD (dissipative particle dynamics) simulations in which each lipid is taken to be a linear chain of one hydrophilic particle and three hydrophobic particles [Lardaji & Sunil Kumar 2004, 2005]. Each particle is given a label "A" or "B" and assigned an interaction strength between it and the other particles in the system, including water. Parameters are set such that lipids are strongly segregated into A-rich and B-rich domains. For vesicles with little excess area, the authors state that "coarsening proceeds mainly through coalescence of flat circular patches." When the ratio of lipid types is set to be 0.3 by

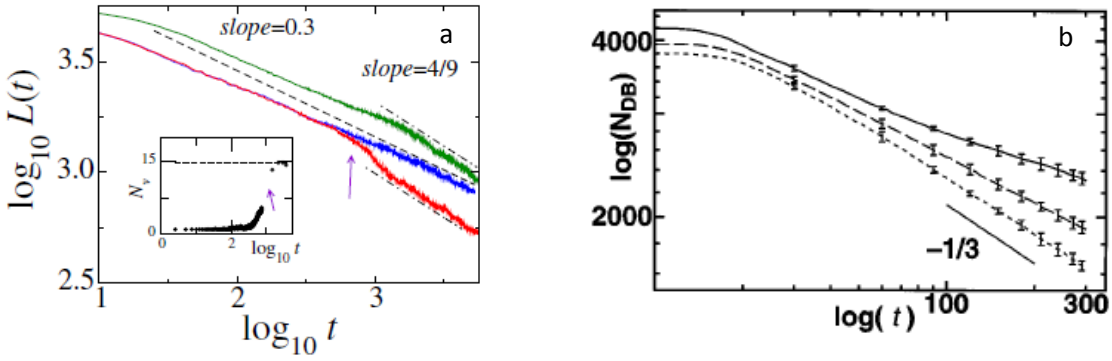

Figure 6: **(a)** Figure taken from Laradji & Kumar, 2004. A log-log plot of perimeter, *L*, between the two phases in time, *t*. The blue curve is a simulation of a vesicle with high line tension which fits to a line with a slope of -1/3. **(b)** Figure taken from Taniguchi, 1996. A log-log plot of the number of domain lattice points, N_{DB}, over time. The bottom dotted line is a simulation done on a rigid sphere. The guide line is a slope of -1/3.

volume, the total perimeter of all domains decreases with a growth exponent of α that is slightly less than α = 0.3 (Figure 6a). When the same simulation was done at an area fraction of 50%, Laradji et al. found that α = 0.5.

(3) Taniguchi (1996) conducted a simulation using a purely dissipative dynamical model of a two-component vesicle in which the area fraction of one component was 30%. The total perimeter of domains was defined as the number of boundary points between phases on a lattice. The total perimeter decreased in time with an exponent of α = 1/3 (Figure 6b).

(4) Fan et al. (2010) used a two component simulation in a planar membrane. They focused on three hydrodynamic lengths, L_h, where L_h is the ratio of the 2D membrane viscosity to the 3D bulk viscosity. If $L_h \rightarrow \infty$, hydrodynamic effects are confined within the membrane and are independent of the solvent. If $L_h \rightarrow 0$, the hydrodynamic effects are dominated by the solvent. If L_h is finite, hydrodynamic effects couple the membrane and the solvent and govern the growth of the apparent domain size, R. In the case of $R << L_h$, domains diffuse and coarsen with α = 0.3. They also find α = 0.3 in the case of Ostwald ripening when the membrane viscosity is high.

(5) Camley & Brown (2010) used a stochastic phase-field model of a quasi-two-dimensional thin fluid membrane in a 3-dimensional aqueous solution. This simulation also took into account thermal fluctuations in the boundaries of the phases. For off-critical mixtures, they find that if domain radius is much greater than Lh, then α = 1/3. The opposite case, in which domain radius is much smaller than Lh, does not apply to our system, and yields α = 1/2.

Near critical

A vesicle with a 50% area fraction of both phases is close to a critical composition. In a quench, near-critical vesicles phase separate into elongated domains instead of small circular domains. Elongated domains in a membrane that is quenched to a low temperature change

shape from long, thin worm-like structures to more circular structures. Scaling is not always observed in this case, which means that a single value of α cannot always be assigned. When scaling is seen in simulations, normalized domain size, *R*, increases as $t^{1/2}$ when R >> L_h [Ramachandran et al. 2009, Laradji & Sunil Kumar 2005, Camley & Brown 2010, Camley & Brown 2011]. In this dissertation, I show that *R* indeed increases with α ~ 1/2 when domains are elongated, which occurs at short time after a quench (Table 3). I also show that once the domains become roughly circular in shape, they begin to grow by collision and coalescence. In this regime, the growth exponent becomes 1/3.

Table 3: Simulation results for domains on vesicles with an area fraction of 1/2.

α	Model	Reference
$1/2$	Purely dissipative dynamics (DPD). ϕ = ½	Laradji et al. 2005
$1/2$	DPD, membrane is not surrounded by water.	Ramachandran et al. 2009
$1/2$	Binary fluid with hydrodynamics, $R \gg {}^{\eta_m}/_{\eta_{bulk}}$.	Fan et al. 2010
$1/2$	Binary fluid with hydrodynamics and thermal fluctuations. $R \gg {}^{\eta_m}/_{\eta_{bulk}}$.	Camley et al. 2011

II. Methods

Collection of raw data

I study vesicles whose lipid membranes contain coexisting liquid phases. My experimental work is a collaboration with Dr. Aurelia Honerkamp-Smith, who produces vesicles by electroformation from a mixture of dipalmitoylphosphatidlycholine (DiphyPC), diphytanoylphosphatidylcholine (DPPC), and cholesterol. A dye, Texas Red DPPE, at 0.8 mole % labels the L_d phase. Vesicles are formed in a solution of either 100 μM sucrose in water or 4-5% (by weight) dextran plus 1 mM sucrose. Depending on the composition of lipids used (see table 4), vesicles contain some area fraction of dark, liquid-ordered domains diffusing on a bright, liquid-disordered, low-viscosity background (Figure 1). In the case of vesicles with an area fraction $\geq$ 0.5, domains can also be bright on a dark background.

Immediately before observation, vesicles are diluted ~ 40-fold in water or in 4-5%

dextran, respectively. In both cases, a slight osmotic pressure difference eliminates excess area in the membrane. This is important because excess membrane hinders the coarsening of domains, which bulge out of the membrane surface [Vind-Kezunovic et al. 2008, Taniguchi 1996, Yanagisawa et al. 2007, Laradji & Sunil Kumar 2004, Ursell et al. 2009]. In order for bulged domains to approach each other, the membrane between them must curve. Figure 4 shows part of a figure by Ursell et al. that illustrates the bending energy of a membrane section (blue) that is between two neighboring domains (red). The bending energy associated with this membrane deformation results in a kinetic barrier to domain coalescence [Ursell et al. 2009].

Aurelia collects movies via an upright fluorescence microscope with an air objective. The Texas Red dye is excited with a mercury lamp and emitted light is filtered and collected with a CCD camera at either 2 frames/second or 10 frames/second. She controls vesicle temperature via a home-built stage and controlled as described previously [Honerkamp-Smith et al. 2008]. Upon commencing acquisition of images of vesicles in a movie, Aurelia quenches the temperature ~2 ^{o}C (Figure 1).

Pre-processing

I delete all frames of the movie before the time when temperature stabilizes to a

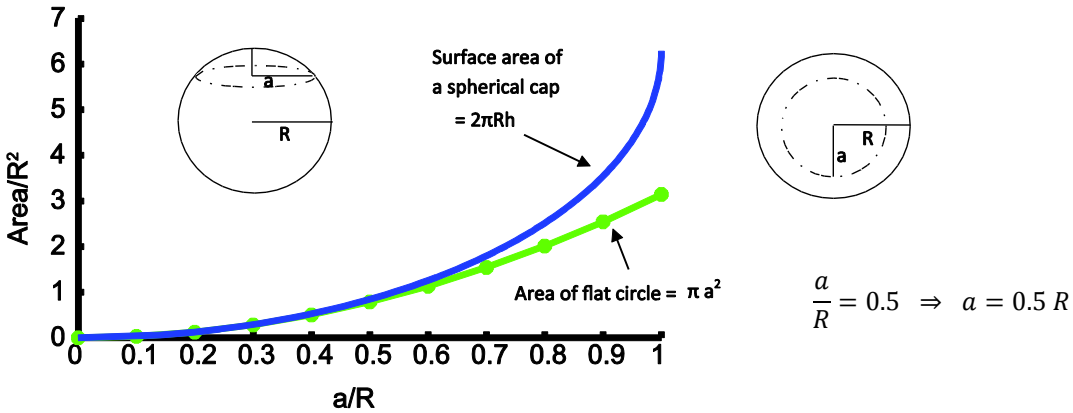

Figure 7: Dimensionless area, which is area/R^2, is shown for a flat circle (green curve with symbols) and a spherical cap (blue curve). For the flat circle, R is a maximum radius and a is the actual radius of the circle. For the spherical cap, R is the radius of the sphere and a is the radius at the base of the cap. Here I show that the area of a circle and a spherical cap are roughly equal until the ratio of a/R exceeds roughly 0.5. This ratio is roughly 5 times larger than for any vesicle domain analyzed in the research here.

standard deviation in the temperature of ~ 0.02 ^{o}C. For example, in figure 1b all of the points before point c are not analyzed and are therefore deleted from the movie I use for measuring the dynamics of phase separation. I remove vesicle drift from movies by aligning each image frame with respect to the vesicle center in the 2D images collected by the CCD camera. I crop images to select areas that are in focus and relatively flat. Flatness can be evaluated from the geometry of a sphere with a diameter of 200 µm. The area, *A*, that is in focus is a spherical cap ~ 25 µm in radius. If $A = 2\pi Rh$, then this spherical cap has a height of 3.2 µm in the center with respect to the edges. For an arbitrary vesicle of radius *R*, the true surface area of a domain, a spherical cap with a circular base with area *a*, does not differ significantly from the area of the base until the domain radius is approximately half the radius of the vesicle (Figure 7). Typical values of *a/R* in our experiment are < 6%. However, domains near the edge of images can appear ovoid and can appear to diffuse shorter distances. To correct for this, I map the 2-D image back onto a 3-D sphere by incorporating a Matlab program written by Sarah Veatch into my code. When this is applied, a slight stretching of domains near the corners of the frame can be observed; ovoid domains become more circular (Figure 8).

Overview of coding

There are four main areas of coding required for my project; (1) centering and cropping movies for removing drift, (2) tracking domains for the measurement of diffusion coefficients, (3) measuring the boundary between liquid phases for the calculation of growth exponents, and (4) individual domain size tracking for Ostwald Ripening studies. With the exception of the first, all of the project areas require a parameter file specific for each data set and a "run_all" file

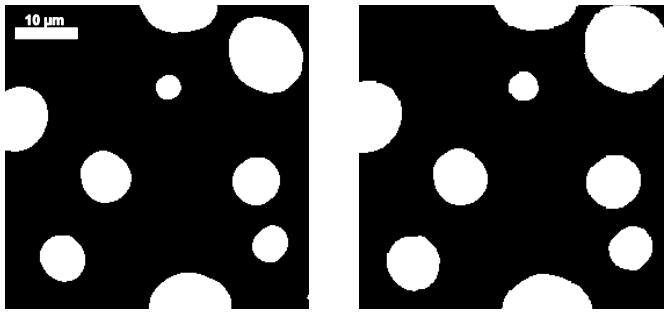

Figure 8: The left panel is a thresholded image of one frame of a movie of domains on the surface of a giant unilamellar vesicle. The right panel shows the same frame after I apply a geometric fix.

that analyzes the data and also reruns parameter files if necessary.

Centering and cropping movies

The vesicles used for all projects are free-floating in water. The vesicles exhibit Brownian motion, are subject to flows in the water, and often move during a temperature quench. Vesicles can undergo translation and/or rotation, which I will discuss separately. Both must be removed to ensure that the domain displacement and growth that I measure is due only to Brownian motion of domains in the membrane. To accomplish this, each frame of a movie must be aligned with respect to the center of the vesicle, thereby deleting vesicle translation from the movement of domains. This step allows for easy, long-term tracking of domains. In order to identify collective movement due to rotating vesicles, I trained and directed an undergraduate in a project to map domain trajectories (Figure 9) using a modified version of the diffusion coefficient program described below, which was later used to investigate whether Ostwald Ripening of domains occurs on vesicles. The trajectories clearly show which vesicles exhibit collective movement of domains, and are hence rolling or spinning (Figure 9b), and which have no net movement of domains (Figure 9a). Once a movie is collected via the commercially available program NIS elements, it can be centered automatically via a Matlab program called **track_vesicle** (Appendix A). The original program **track_vesicle** was written by Dr. Aurelia Honerkamp-Smith and I further modified it for use here.

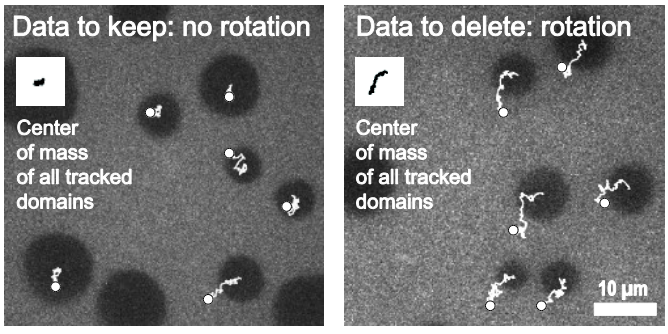

Figure 9: **a)** Trajectory of domains over 61 frames. Average trajectory shows no common direction of domain movement. **b)** Trajectory for 61 frames of a vesicle exhibiting common movement in one direction suggesting a rotating vesicle.

Track_vesicle places a ring on top of the brightest part of the image, which corresponds to the perimeter of the portion of the vesicle that is in focus. Given the radius of this ring, the program finds a list of x and y-coordinates of the

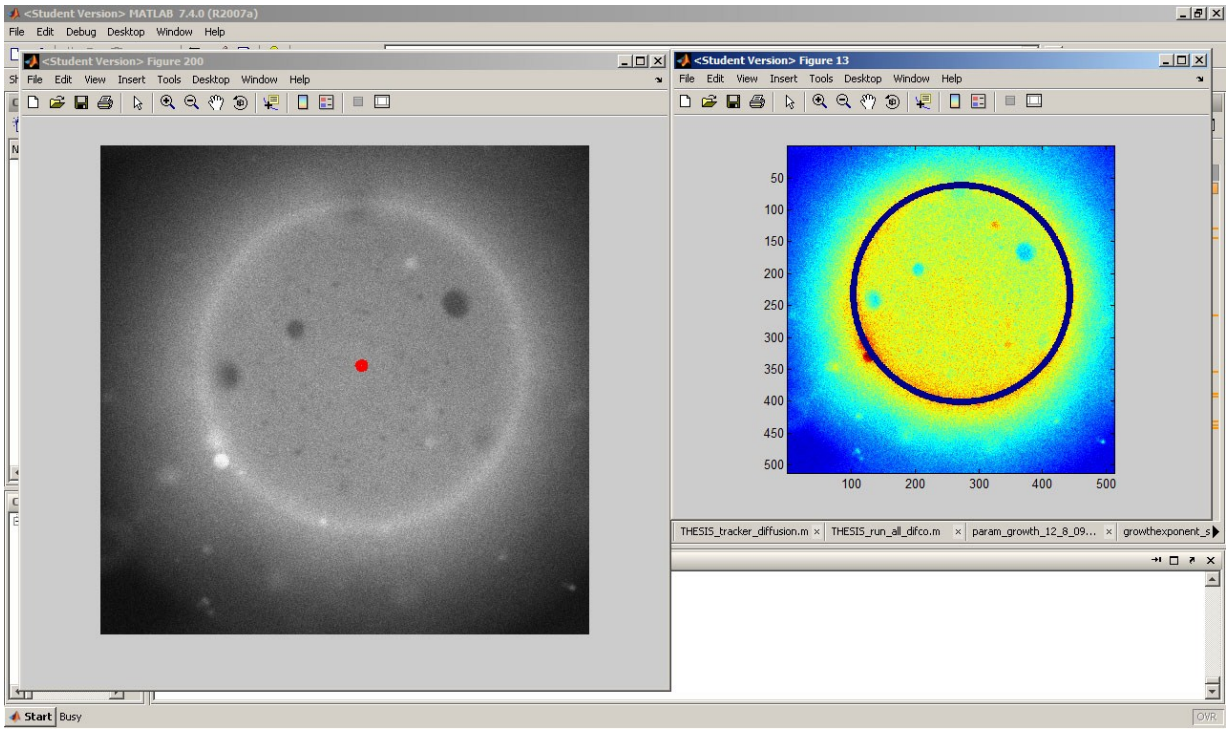

Figure 10: Screen shot of the Matlab function, track_vesicle, which outputs the center of the vesicle as a red dot (left) as a result of fitting the dark ring overlaid on the high-intensity perimeter (shown in red in the false-color representation at right) that bounds the area in focus.

center of each frame. All of these coordinates are plotted on an image of the first frame and saved as a data file in the folder containing the movie (Figure 10). The user has two options for finding the radius: the program can choose the best fitting ring, or the user can explicitly enter the radius. We have configured the fitting program so that users click the center of the vesicle and the outer edge, and then the code finds the best fitting ring (Figure 11). The radius of the area in focus is collected in a data file placed in the folder containing the movie. A limitation of this code arises if the edge of the area in focus goes out of the frame. In this case, the ring's radius will appear to be terminated at the edge of the image and the center of the area in focus will be incorrect. If a movie contains frames in which the edge of the area in focus goes off screen, then it is necessary for the user to input a value for a guess of the radius of the area in focus. This allows for the ring to continue outside of the frame, leaving the center of the ring in the correct position.

The **track_vesicle** program can encounter problems fitting a ring around the area in focus for various scenarios: when bright artifacts such as vesicle aggregates or tubes are near the ring, when additional vesicles appear at the edge of the frame, when the ring around the area in focus is dim, and when the vesicle is so large that the ring around the area in focus is off of the frame. In order to track these movies, undergraduate Chris Warth augmented the **track_vesicle** program to make a new version called **manual_track**. He added the functionality to allow the user to click the center of the area in focus at chosen intervals in the movie (Figure 10 left). The program then interpolates the movement of the vesicle from the earlier click to the later click. This program is very useful for tracking the centers of vesicles that drift smoothly.

The program **center_crop** (written by Dr. Aurelia Honerkamp-Smith) is called at the end of **track_vesicle**. It uses the x and y-coordinates collected to center each frame with respect to the center of the area of the vesicle that is in focus. The coordinates are smoothed by a box car averaging program to eliminate any jumps in the center coordinates of the fitted ring due to slight variations in the brightness levels from frame to frame. It then uses the shortest distance

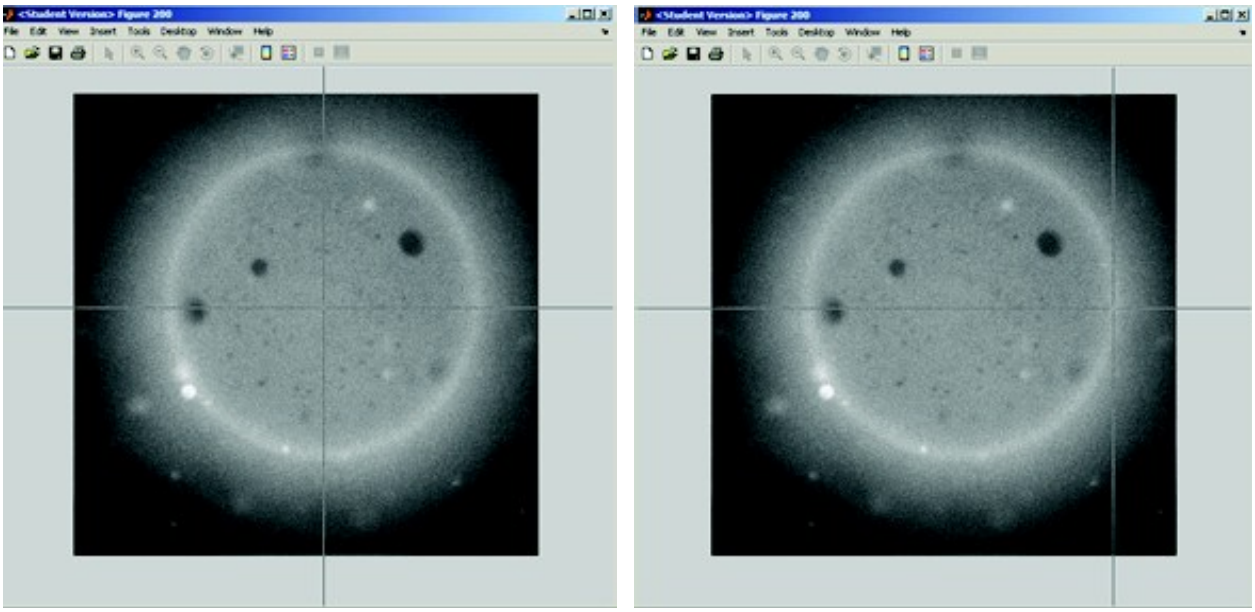

Figure 11: **Left** – The crosshairs show the location of the first click to tell the Matlab function track_vesicle where to start looking for the center of the vesicle. **Right** - Second click to tell the function a starting point to find the edge of the region in focus.

from the center to the edge of the frame to crop a square centered on the vesicle (Figure 12). The output is a cropped tif stack that can be used for the next three parts.

Diffusion Coefficient Measurement

I find the diffusion coefficient by measuring the mean squared displacement of domains of different radii versus time. To do this, I have adapted Matlab code from an original program by Pietro Cicuta [Cicuta et al. 2007]. We have chosen a system in which domains diffuse within a membrane for which the 2-dimensional viscosity, η_m, has been previously shown to be low (42:25:30 DiphyPC:DPPC:chol) [Cicuta et al. 2007]. In figure 2a, I verify that the diffusion coefficient varies as $D \propto {}^{1}/_{r}$ within experimental uncertainty for vesicles with low area fraction in water.

Next, I tested how increasing the 3-D bulk viscosity of the solution affects the diffusion coefficient of domains. Aurelia produced and diluted vesicles in a higher viscosity solution, an aqueous solution of 4-5% dextran (average molar mass 400-500 kDa). An undergraduate whom I advised, Andrea Lamprecht, used a viscometer to determine that the viscosity of a 4-5% dextran solution at 23°C is ~3.2 times the viscosity of water. By fitting our data in Figure 2a (bottom dashed line), I independently found that the viscosity of our dextran solution is ~3.2 times the viscosity of water.

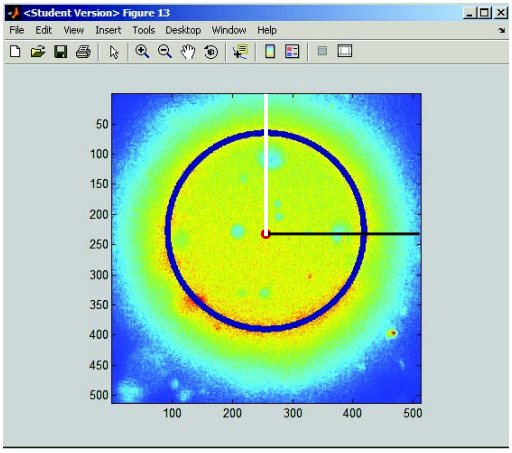

Figure 12: The dark ring is overlaid on the high-intensity perimeter that bounds the area in focus. The black bar is longer than the white bar, so the frame will be cropped from the center to the end of the white bar.

Tracking domains requires establishing parameter files. Domains are tracked in this dissertation for four different applications: determination of vesicle rotation, measurement of diffusion coefficients, analysis of growth exponents, and evaluation of Ostwald ripening. Parameter files for measuring diffusion coefficients are called

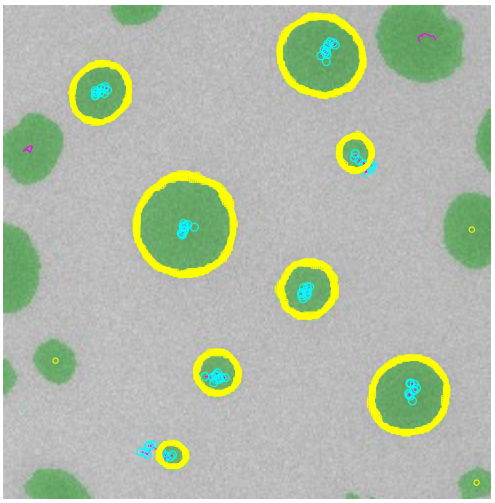

Figure 13: Illustration of the challenges of tracking domains. Trackable domains are circled in yellow. These are domains that never touch the edge of the image over the course of a specific number of frames, which is 10 frames in this case. Domains not circled in yellow but with faint pink lines inside are false positives, which means that the program has incorrectly identified these domains as trackable even though they touch the edge of the image. The inability to resolve false positives (for example, by asking the program to delete all domains with black pixels at the image edge) mean that domains are required to be identified by hand. The process could not be automated. Domains not circled in yellow but with a yellow dot inside are correctly identified by the program as untrackable.

param_difco (Appendix A) and are the simplest of the parameter file types because the **tracker_diffusion** (Appendix A) program requires the user to identify which domains can be tracked and does so only in ten frame intervals. Since each interval needs a new parameter file, there is no need for lists of parameters that can analyze entire movies or even lists of movies. Intervals of five, ten, and twenty frames were tested for a single movie to determine the best interval of measurement. Movies broken into five-frame intervals took the longest to analyze. Movies broken into twenty-frame intervals had the lowest R^2 for a linear fit of mean square displacement (msd) versus time. Ten frame intervals were chosen for all subsequent calculations of the diffusion coefficient as a compromise between maximizing efficiency and minimizing R^2 of the fit.

The **tracker_diffusion** program is called by the **param_difco** file. It includes a Gaussian filter and a thresholding algorithm that uses the Otsu method [Otsu 1979] to find a minimum in the histogram of pixel values in order to distinguish between grey scale values in the domains. After each set of ten frames is thresholded and after the image is stretched geometrically, the program assigns a different number to each of the features in the frame that are trackable (Figure 13). Trackable domains are those that do not reach an eccentricity that differs by more than 25% from the eccentricity of a circle, do not merge with other domains, and do not touch the edge of the image. Allowing the program to determine all trackable domains produces false

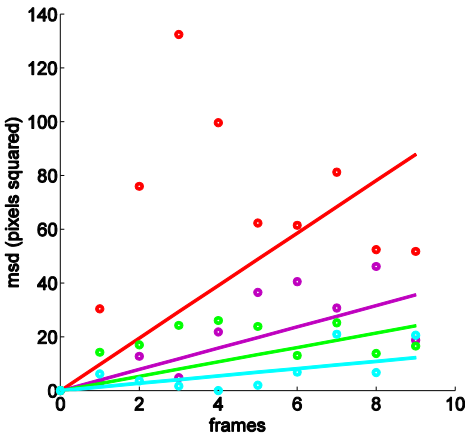

Figure 14: Mean square displacement versus frame for 4 different domains. Aurelia wrote a linear fit function to force the fit through the point (0,0).

positives. Therefore, the user must select the trackable domains from among the features that the Matlab program labels as acceptable features for diffusion coefficient calculations. Two examples in which false positives arise are when (1) two domains complete a merge between frames such that a figure-8 shaped intermediate is not captured in the movie and (2) if most of the domain is in the frame, but a small portion is out of the frame.

The displacement of each domain is found by tracking its center of mass over ten frames. The displacement of each center of mass from its initial position is calculated for each frame in both the x and y direction. The sum of the msd in the x and y directions is then plotted versus frame number (Figure 14). Calculation of the diffusion coefficient requires a simple conversion from pixel^2/frame to μm^2/time. Since we are interested in the dependence of the diffusion coefficient on domain radius, the diameter (in pixels) of each clicked domain is recorded and saved with the diffusion coefficient. The diameter of chosen domains is found using Matlab's standard regionprops function.

To make automation easier, I wrote a series of **run_all** files. Their first function is to search for all the output files generated by **tracker_diffusion** programs within a specified folder. Their second function is to convert units of pixels and frames into units of micrometers and seconds. The conversion to μm from pixels depends on the magnification used while imaging. In most cases Aurelia collected the images with a 40x objective, which means that each pixel is a square with an edge of length 0.18 μm. For images collected with a 10x objective, each pixel is a square with an edge of length 0.72 μm. Converting the slope from pixel^2/frames to μm^2/seconds requires knowing how many frames were collected per second using the commercial software that drives our camera (NIS Elements). Since many domains have similar

radii, it is easiest to view the data by binning diffusion coefficients by radius and plotting the average diffusion coefficient with standard deviations. My **run_all** file plots this without using Matlab's internal errorbar plotting function so that conversion of the plots format to log-log will not change the appearance of the error bars. My **run_all** file also plots the expected values of diffusion coefficients over the range of radii using equation 3, called from my function called **Ediffco** (Figure 2, dashed lines), and using equation 4, called from my function called **PetrovSchwille** (Figure 2, solid lines) (Appendix A).

Growth Exponent Measurement

I have found that domain radius is not the best parameter to evaluate when measuring growth laws (Figure 15). The measurement uncertainty in average domain radius is large because as time progresses there are fewer domains, and statistics become poor. Moreover, radii cannot be accurately found for domains whose edges touch the edge of the cropped viewing window (Figure 13). Disregarding data from domains in contact with the edge leads to undercounting of large domains and a growth exponent that is lower than theory predicts.

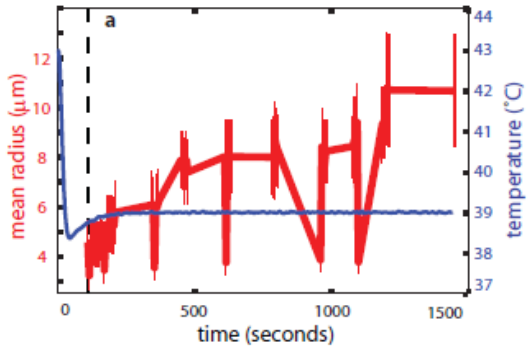

Figure 15: Illustration of the problems inherent in measuring mean radius of domains. The plot to the left shows mean radius over time (red) and temperature (blue). Notice that data collection does not start until the time at which temperature has attained a value of at least 0.5°C from the final set temperature (black dashed line). The slope of a line fit to log(time) versus log(mean radius) should yield the growth exponent. This plot illustrates how mean radius can vary significantly as a result of poor statistics. Drastic dips in mean radius are due to undercounting of large domains that touch the edge of the cropped image.

Instead, I measure domain growth exponents by evaluating two different parameters: (1) perimeter of all domains, including those touching the edge of the image, and (2) normalized domain size. Since average domain radius is expected to grow as $t^{1/3}$, perimeter should decrease as $t^{-1/3}$, provided that the area fraction of domains is constant (Figure 3).

In our images, the local area fraction is not conserved throughout the movie, although the area fraction for the entire vesicle is constant. If a domain travels into the frame, the measured area fraction, AF_m, and measured perimeter, P_m, will increase. In order to distinguish changes in perimeter that occur due to coarsening from changes due to domains diffusing into or out of view, we define an effective total perimeter, P_E, given an ideal area fraction, AF_I:

$$P_E = \frac{P_m AF_I}{AF_m}. \qquad \text{(eq 5)}$$

Figure 16 shows the effective perimeter versus time for domains that coarsen after a temperature quench. The ideal area fraction is arbitrary and in all cases I set it to be the area fraction in the first frame of the movie.

There are four advantages of measuring normalized domain size, R, which is the measured area of the minority phase divided by the measured perimeter of all domains in the image. First, as in the measurement of effective perimeter, corrections are applied for changes in area fraction in each frame. Second, comparison with other literature values [Fan et al. 2010, Camley & Brown 2011] is more straightforward. Third, readers gain an intuition that domain size is increasing. If area (μm^2) is divided by perimeter (μm), the result is a normalized length that should still increase as $t^{1/3}$ just as domain radius does (Figure 19a). Fourth, normalized domain size can be used to evaluate length scales of elongated

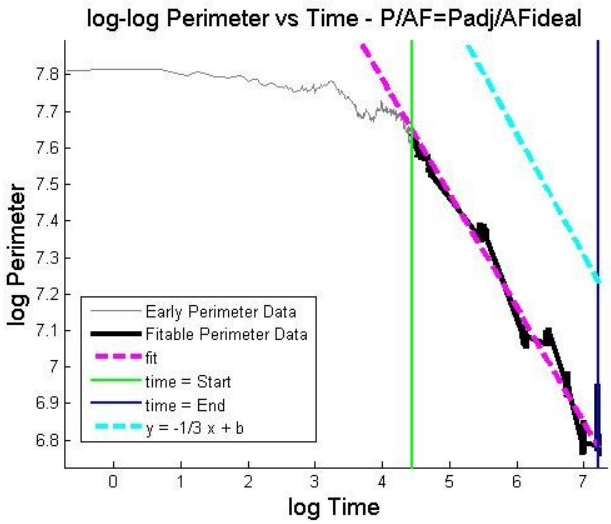

Figure 16: Log-log plot of effective perimeter versus time for one quench of one vesicle. Data to the left of the green line corresponds to early times, before temperature stabilized. The purple line fits all data at time points after the green line, and yields a slope near -1/3. The offset blue line has a slope of -1/3 and is shown simply for comparison of the slopes. A slope of -1/3 means that perimeter $\propto$ (time)$^{-1/3}$.

domains in membranes near miscibility critical points, for which measurement of domain radii is not well defined.

The parameter files that I have established to measure growth exponents (called **param_growth**) involve the most pre-processing effort. Since the program for measuring the growth exponent can run multiple movies, the parameter file must contain all of the parameters for all of the movies. Different parameters are needed because the size and number of domains change throughout each movie. Most of these changes occur in the first 400 frames of the movie. I have found that it is best to create one **param_growth** file for each quench, including all of the filter parameters for all of the movies in one location. The example **param_growth** file in Appendix A illustrates the options available for processing, which are described within the comments.

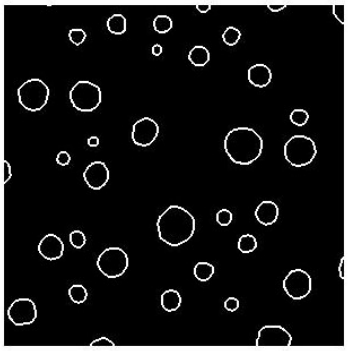

Figure 17: An image of the perimeters of domains in one frame of a movie.

I wrote three versions of the growth exponent measurement program. All programs threshold greyscale images using the Otsu method [Otsu 1979]. My complete program is called **growthexponent** and allows the user to run multiple movies without any input during analysis. This program calls Matlab's standard regionprops function to count the pixels in the boundary between the black and white regions (Figure 17) after each frame is thresholded and stretched geometrically. About 1000 frames of a 200 pixel^2 movie requires ~ 2 hours to run, but can be run unattended. Data analysis is checked in the output files upon completion. If a subset of movies within a group process correctly, the user can specify which data require further analysis in the **param_growth** file and re-run only the data necessary.

The second variation of the growth exponent program is called **growthexponent_startanywhere** and allows for overlapping of starting and ending points for

each set of parameters so that the filtering parameters can be compared as they are changed. This is for testing the parameters on each movie. If a change in parameters causes a large change in the output data, those data can be re-examined to discover why the parameter change has such a large effect. The first and second programs could be combined if needed, but keeping track of parameters becomes complicated. I have chosen to keep the individual programs and call the one I need from the parameter files.

The third version of the growth exponent program is called **growthexponent_choose** and produces the same types of output as the **growthexponent** program does, except that the program contains code from **tracker_diffusion.** This feature is designed to eradicate the same type of false positives that arise in identifying trackable domains in measurements of diffusion coefficients and to measure the area of specific domains. The **growthexponent_choose** program requires manual input to select the measureable domains within an image. Using the standard Matlab program regionprops, several different parameters can be measured for each individual domain. The problem is that some domains are untrackable in some frames, so the measurement must be done over small enough sets of frames that every domain is measured over as much time as possible. This requires the creation of several parameter files as in the diffusion coefficient project, but is not limited to intervals of only ten frames since individual domains are not required to be individually identified from frame to frame.

Ostwald Ripening Measurement

The hallmark of Ostwald ripening should be that the differences in sizes between neighboring large and small domains would increase through time in the absence of domain collision and merging. When I track individual domains, I find that domain areas are roughly constant between merging events (Figure 18). By eye, I believe that I occasionally observe very small domains slowly shrinking in size until they become indistinguishable against the background. However, these small domains are below the resolution for thresholding and

processing by my Matlab programs, so their sizes are difficult to quantify over time. I have attempted several procedures for simultaneously measuring the sizes of all domains spanning large length scales in a single image. Unfortunately, the filtering process used in my code requires a minimum and maximum length scale for domain determination rendering simultaneous measurement impossible with my current code. When domains are very different in size, large domains are broken into smaller domains, or small domains are consumed in the background.

I established parameter files called **param_ost** for the Ostwald ripening project. These files are similar to the **param_difco** files, except that each domain is assigned a letter and every parameter file is named for that individual domain. Individual domains are then tracked for as long as possible until an untrackable frame is reached for that specific domain. Then a new

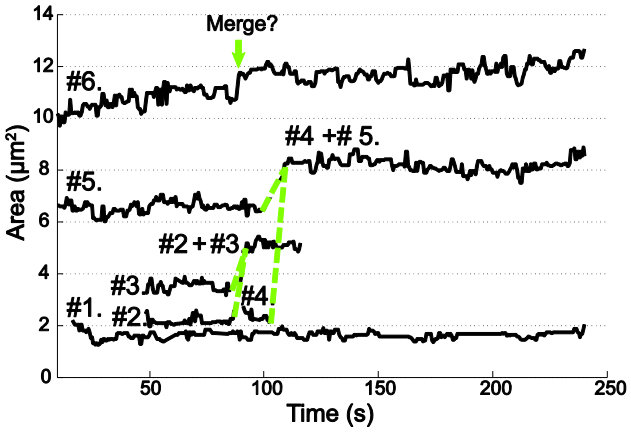

Figure 18: Area in time for 6 domains on the same vesicle. The hallmark of Ostwald ripening should be that large domains become larger and small domains become smaller without collisions. This figure also shows merges between domains 2 and 3 and between domains 4 and 5. Domain 4 is measured for a short time because it diffuses into the viewing area around t = 85 seconds and then merges with domain 5 at t=105 seconds. Domain 6 likely merges with a small domain around t= 90 seconds, where the small domain is below the resolution of the filtering process used to identify domains. Before and after this putative merge, domain 6 is a possible candidate for Ostwald ripening since it gets bigger with time. No other domains are viable candidates, although small domains that may be getting even smaller through time may be below the resolution of the filter process. The conclusion from the figure above is that domain coarsening is dominated by merging processes rather than Ostwald ripening.

parameter file is generated for that domain, which picks up at the next trackable frame. In this way, several data files containing the radii for individual domains named a1, a2, …, b1, b2, … are generated. The program **growthexponent_choose** was optimized for the Ostwald ripening measurement and has been equipped to save the initial frame and the number of frames to a list to be called by a **run_all** file. With this information, my **run_all** file pieces together all of the size information in time for individual domains to visualize what is happening on a vesicle throughout a whole data set. The result is a trace of several domains over the timeframe of the entire data set with the few untrackable points excluded (Figure 18).

III. Results

D(r) and η_{2D} for $\phi < 0.3$

When area fraction $\phi < 0.3$, vesicle membranes of DiphyPC/DPPC/chol contain circular L_o-phase domains that coarsen over hundreds of seconds at constant temperature. Figure 2a shows diffusion of micron-scale L_o domains within an L_d background. The largest angle subtended by a domain tracked in figure 4a is 20˚. By eye, $D \propto 1/r$ in figure 2a, which means that Eq. 1 and its condition that $r << L_h$ do not hold in this case. Since diffusion in figure 4a is roughly a factor of two slower than predicted by either Eq. 2 or Eq. 3, the condition that $r >> L_h$ does not hold. By applying Eq. 4, which is an approximation valid between the two limiting cases of Eq. 1 and Eq. 2 [Petrov & Schwille 2008] and by using $\eta_{3D} = 0.652 \times 10^{-3}$ Pa s at the average experimental temperature of 40.3˚C [Kestin et al. 1978], we find a best-fit value of $\eta_{2D} = (3.3 \pm 1.1) \times 10^{-9}$ Pa s m, with a 95% confidence interval from $\eta_{2D} = 1.0 \times 10^{-9}$ Pa s m to 5.6×10^{-9} Pa s m. This result is self-consistent in that we find hydrodynamic length $L_h = \eta_{2D}/\eta_{3D}$ to be roughly 5 µm, which is of the same order of magnitude as domain radius, r. As such, the approximation by Petrov and Schwille [Petrov & Schwille 2008] should indeed be justified. Applying equations relevant to fluid rather than solid domains (e.g. applying an approximation based on Eq. 3 rather than Eq. 2) would likely increase the value of η_{2D} that we find by ~10%.

Our value of η_{2D} = (3.3 ± 1.1) x 10^{-9} Pa s m is in good agreement with results previously found in vesicles composed of the same three lipids used here, albeit at different ratios. In that work, temperatures and compositions were tuned to place vesicles near membrane miscibility critical points. Analysis of structure factors of membrane critical composition fluctuations yielded η_{2D} = (5.5 ± 1.5) x 10^{-9} Pa s m [Honerkamp-Smith et al. 2012], and analysis of shape fluctuations of domain boundaries yielded (4 ± 1) x 10^{-9} Pa s m [Camley et al. 2010]. The good agreement suggests that equations formulated for diffusion of a single inclusion within a uniform, flat membrane adequately describe diffusion of a domain within a curved GUV membrane containing multiple domains, at least within experimental uncertainty. In figure 4a, we make three minor improvements on the previous measurement of Cicuta et al. [Cicuta et al. 2007], namely, we confine our results to vesicles with diameters >80μm, we correct for curvature, and we exclude vesicle rotation without subtracting the center of mass of all domains.

We test our results by varying the viscosity of the bulk solution, specifically by placing vesicles in a dextran solution with a viscosity that is 3.2 times that of water. Our most basic expectation is that diffusion coefficients should decrease by roughly a factor of 3.2, which they do (Figure 4a). Another test is to compare η_{2D} in the dextran vs. water solutions; they should be the same within experimental uncertainty. Using the same approximation above, we fit diffusion coefficients for vesicles in dextran using a 3D viscosity that is 3.2 times the viscosity of water at the average experimental temperature of 38.1˚C, or η_{3D} = 3.2 * (0.675 x 10^{-3} Pa s). We find a best-fit value of η_{2D} = (9.6 ± 3.0) x 10^{-9} Pa s m, within a 95% confidence interval from η_{2D} = 3.0 x 10^{-9} Pa s m to 1.6 x 10^{-8} Pa s m. The confidence intervals of η_{2D} from the two systems overlap as we expect they should.

Growth exponent α for ϕ < 0.3

Coarsening results in an increase in normalized domain size, where $R \propto time^{\alpha}$ and α is

the growth exponent. Circular domains observed at area fractions of $\phi < 0.3$ appear to coarsen primarily by a mechanism of collision and coalescence rather than by evaporation-condensation. To illustrate this, figure 18 tracks a group of coarsening domains over 250 seconds. A hallmark of evaporation-condensation is that small domains become smaller and large domains become larger. No domains appear to be shrinking, although domains $\leq 1\ \mu m^2$ elude our tracking program. The growth of only one domain, the largest one, is not explained by merges with other tracked domains. It is unclear how much of this growth is due to evaporation-condensation vs. merges with untracked domains.

We expect to measure a growth exponent of $\alpha = 1/3$ whether or not domains grow by evaporation-condensation or by collision and coalescence (since the data in the log-log plot of figure 4a have a slope of –1, so roughly follow $D \propto 1/r$). Figure 19a shows a collapse plot of 17 measurements of R vs. time for vesicles with area fractions $\phi < 0.3$. The resulting average growth exponent of $\alpha = 0.29 \pm 0.05$ (Table 4) is within experimental uncertainty of the

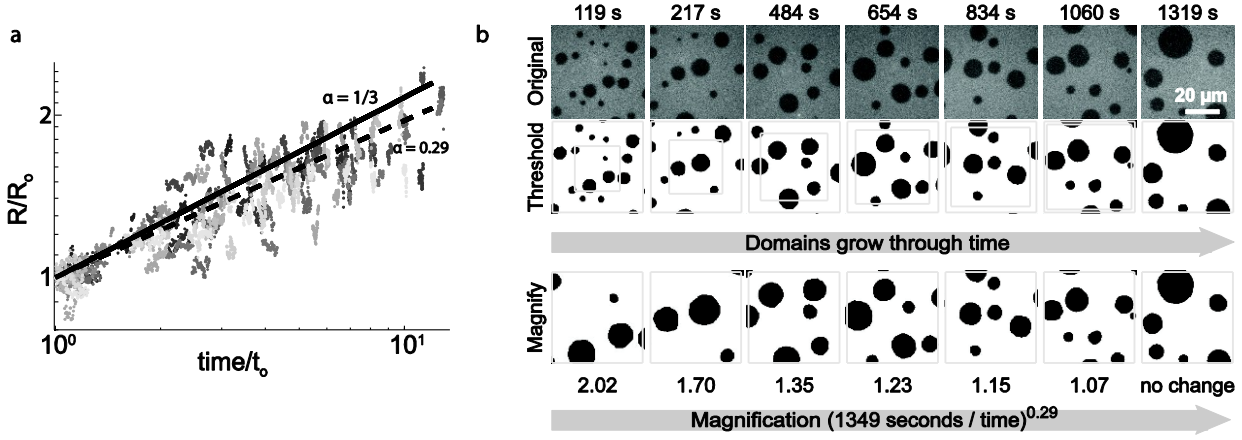

Figure 19: **a)** Collapse plot of normalized domain size, R, vs. time for circular liquid domains in vesicles with area fraction of liquid ordered phase of $\phi < 0.3$. Vesicles are diluted in water and 17 distinct measurements are denoted by different grey scales. Measurements are rescaled by dividing each point by the values of the first data point at (R_0, t_0), where $time - t_0$ ranged from 593 to 1349 seconds. The average growth exponent for this data is $\alpha = 0.29 \pm 0.05$ (dashed line). A solid line denoting $\alpha = 1/3$ is shown for comparison. **b)** Top row: Cropped images of domains on the surface of a 246μm-diameter vesicle of 25:45:30 DiPhyPC:DPPC:chol grow through time. No correction for vesicle curvature and no image contrast enhancement has been applied. Middle row: Thresholded and curvature corrected versions of the same images. Grey boxes show the areas magnified in the bottom row. Bottom row: Domains appear to not grow through time when they are rescaled by a factor of $[(1349 \text{ seconds})/(\text{time} - \text{time}_0)]^{0.29}$.

predicted value of $\alpha = 1/3$. In this experiment, vesicles were diluted in water. Coarsening was followed for a minimum of 593 seconds (and a maximum of 1349 seconds). Roughly the same growth exponent, $\alpha = 0.27 \pm 0.06$, was found when the bulk fluid in contact with the vesicles was dextran solution instead of water.

Visual confirmation that $\alpha \approx 1/3$ for vesicles with area fraction $\phi < 0.3$ is shown for a single vesicle in figure 19b. With time, domain sizes increase. When the micrographs in figure 19b are rescaled for the growth exponent found for this data set, which is $\alpha = 0.29$, domain sizes appear roughly constant through time. For all experiments, data was collected from only the largest vesicles produced (> 80 μm diameter) in order to minimize deviations from $\alpha = 1/3$

Table 4: Growth exponent results

Area fraction (ϕ)	Lipid mol% composition (DiPhyPC: DPPC:chol)	Post-quench temperature **	Total number of quenches	Number of vesicles	Bulk solution	Early vs. late quench	Growth exponent for this subset (α)*	Overall growth exponent (α)*	Predicted growth exponent (α)
$0.015 < \phi < 0.3$	25:45:30	38.7±2.2˚C	12	5	water	N.A.	0.29 ± 0.05	0.28 ± 0.05	1/3
	40:30:30	43.4±0.9˚C	5	3					
$0.015 < \phi < 0.3$	25:45:30	38.1±2.9˚C	9	7	dextran	N.A.	0.27 ± 0.06		
	40:30:30	42.1˚C	1	1					
$0.40 < \phi < 0.60$	25:20:55 to 30:20:50	29.2±0.9˚C	6	5	water	late	N.A.	0.31 ± 0.05	1/3
		17.3±2.5˚C	4	2	dextran				
$0.40 < \phi < 0.60$	30:20:50	28.2±0.8˚C	2	2	water	early	N.A.	0.50 ± 0.16	1/2
		19.4±2.1˚C	6	3	dextran				

* The quoted measurement uncertainty is the standard deviation of all measured growth exponents. It ignores the uncertainty in measuring each individual exponent, which is an order of magnitude smaller than the standard deviation.

**The quoted measurement uncertainty is the standard deviation of post-quench temperatures for all experimental runs. It is roughly an order of magnitude greater than the variation in post-quench temperature throughout the course of each movie.

N.A. = Not applicable.

arising from purely geometric considerations when domain radii approach the size of vesicle radii (Figure 7) and from hydrodynamic coupling of two or more domains via the bulk fluid inside the vesicle.

Line tensions, η_{2D}, and growth exponents for $0.4 < \phi < 0.6$.

By choosing a membrane composition that results in nearly equal area fractions of L_o and L_d phases, and by making a shallow quench below the transition temperature, we place our membrane near a miscibility isothermal critical point (also called a plait point). Henceforth in this document I will use the shorter term "critical point" to denote the plait point. Experimental signatures of proximity to a critical point include fluctuating domain edges with a correlation length ξ, and low line tension between L_o and L_d phases [Honerkamp-Smith 2008 et al., Tian et al. 2007]. By analyzing fluctuations in the shape of domain edges as described in [Honerkamp-Smith et al. 2008], we find that a shallow quench reliably sets membrane tension to a low value of 0.35 ± 0.08 pN. Another way of finding line tension is to fit shapes of merging domains [Wintersmith et al. 2007], using a first-order approximation that Eq. 2 holds.

Line tension found is this way is indeed low (1.25 ± 0.15 pN) for the domain boundary shown in figure 20, which is within a couple of degrees of its miscibility temperature and has area fraction $0.4 < \phi < 0.6$. Analyzing the same movie to extract fluctuations in the shape of non-merging domain edges yields 0.43 ± 0.05 pN. The conclusion remains the same, that

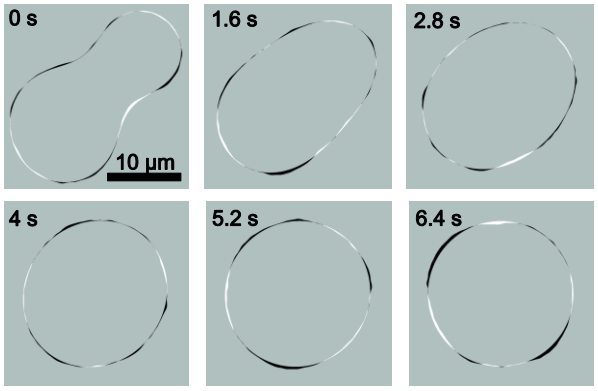

Figure 20: Time series after a merge of two L_o domains within a background membrane of L_d phase in a vesicle composed of 30/20/50 mole% DPPC/DiPhyPC/Chol. Black denotes areas in which L_o domains were observed, but not captured by simulations. White denotes the opposite. The small area of black and white highlights close agreement between data and simulations used to find domain line tension. Line tension is calculated to be 1.25±0.15 pN using the method of [Wintersmith et al. 2007]. This figure was generated by Pritam Mandal and Elizabeth Mann at Kent State University.

membranes prepared with $0.4 < \phi < 0.6$ and a shallow quench exhibit critical behavior.

We measured growth rates of domains in membranes with near-critical compositions in two different time regimes: (1) early after a quench, when phases appear bicontinuous and domains are elongated, and (2) late after a quench, when domains are nearly circular, even though their edges fluctuate. With time, all domains undergo a transition from elongated to circular shapes and then begin to grow primarily via coalescence. Small domains transition before large domains do. Increases in *R* occur both when domains change shape and when domains coalesce. Early quench times are defined here as occurring when increases in *R* are due to shape changes, before small domains grow via collision and coalescence with each other (Figure 21). Late quench times commence when no further increases in *R* are due to shape changes (Figure 22). It is worth keeping in mind that "early" here mens well after spinodal decomposition has commenced since recording does not commence until temperature has

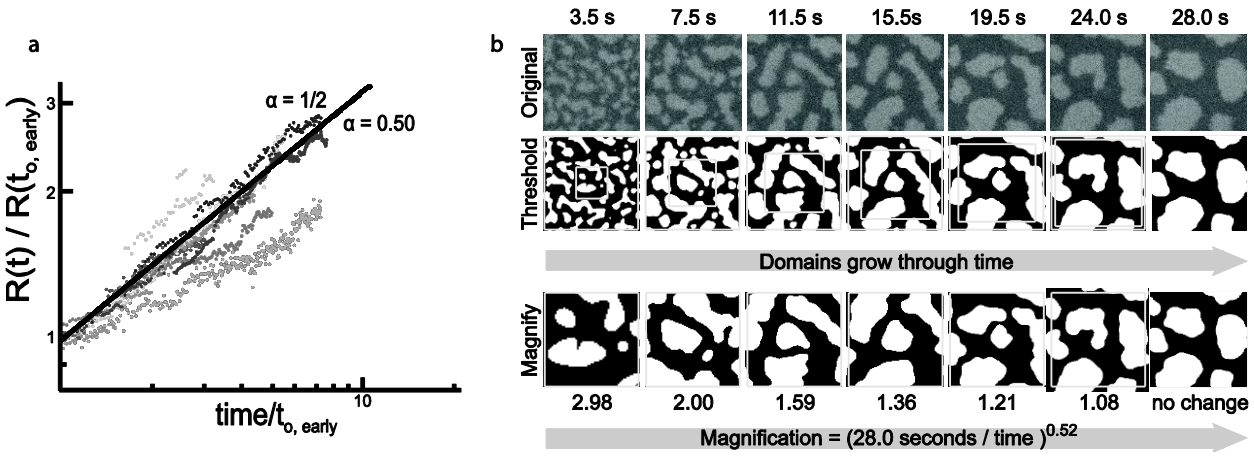

Figure 21: **a)** Collapse plot of normalized domain size, *R*, vs. time for elongated liquid domains in vesicles with area fraction of liquid ordered phase of $0.4 < \phi < 0.6$. Vesicles are diluted in water and 8 distinct measurements are denoted by different grey scales. Measurements are rescaled by dividing each point by the values of the first data point at (R_0, t_0), where $time - t_0$ ranged from 8 to 33 seconds. The average growth exponent for this data is $\alpha = 0.50 \pm 0.16$ (dashed line). A solid line denoting $\alpha = 1/2$ is also plotted, although it lays on top of the dashed line. **b)** Top row: Cropped images of domains through time on the surface of a 101µm-diameter vesicle composed of 30:20:50 DiPhyPC:DPPC:chol. No correction for vesicle curvature has been applied. Middle row: Thresholded and curvature corrected versions of the same images. Grey boxes mark the boundaries of the areas magnified in the bottom row. Bottom row: Domains appear to not grow through time when they are rescaled by a factor of $[(28.0 \text{ seconds})/(time - time_0)]^{0.52}$.

largely stabilized.

For late quench times, we expect the growth exponent to be α = 1/3, just as it was for circular domains with ϕ < 0.3. Experimentally, we find α = 0.31 ± 0.05 for late quench times within a membrane with area fraction 0.4 < ϕ < 0.6, in good agreement (Table 4, Figure 22a). The expectation that α = 1/3 rests on an assumption that, to first order, domains diffuse with $D \propto 1/r$. The data in figure 4b uphold this assumption for late quench times and 0.4 < ϕ < 0.6. The data are fit even better by Eq. 4, which is an approximation valid between the limiting cases of Eq. 1 and Eq. 2 [Petrov & Schwille 2008], and which yields a best-fit value of η_{2D} = (7.0 ± 3.3) x 10^{-9} Pa s m, with a 95% confidence interval from η_{2D} = 1.5 x 10^{-11} Pa s m to 1.4 x 10^{-8} Pa s m.

The scenario is more complex for early quench times in a membrane near a critical composition. The membrane contains elongated domains, which become more circular with time (Figure 21b). In other words, growth in normalized domain size, *R*, is heavily influenced by domain morphological changes and is not simply due to changes in domain area as a result of

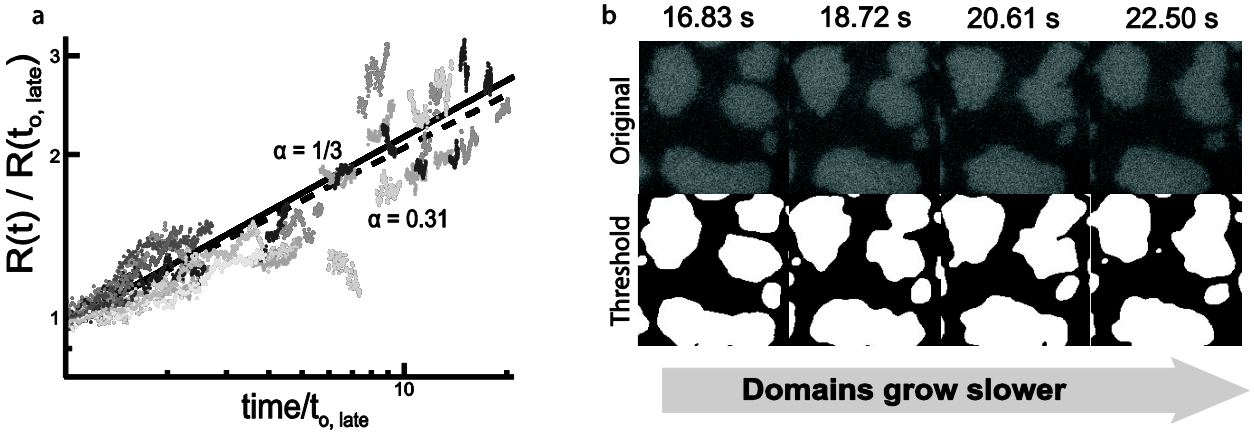

Figure 22: **a)** Collapse plot of normalized domain size, *R*, vs. time for circular liquid domains in vesicles with area fraction of liquid ordered phase of 0.4 < ϕ < 0.6 after the early period of growth. Vesicles are diluted in water and 10 distinct measurements are denoted by different grey scales. Measurements are rescaled by dividing each point by the values of the first data point at (R_0, t_0), where *time* – t_0 ranged from 1.5 to 20.4 seconds. The average growth exponent for this data is α = 0.31 ± 0.05 (dashed line). A solid line denoting α = 1/3 is shown for comparison. **b)** Continuation of the micrographs in figure 21 showing diffusion and collision of domains.

either collision-coalescence or evaporation-condensation. Experimentally, we find $\alpha = 0.50 \pm 0.16$ at early quench times for elongated domains within membranes with $0.4 < \phi < 0.6$. As in figure 19b, domains within a near-critical membrane early after a quench appear self-similar; when micrographs in figure 21b are thresholded and rescaled by the growth exponent, domain sizes appear constant through time.

Figure 23 shows a vesicle with $\phi \sim ½$ for which no scaling operation can be performed as in figure 19b or 21b that makes all panels appear similar. This situation is described by Fan et al. (2010) and Camley & Brown (2011) in reference to simulations they conducted in which R is not well governed by a growth exponent. Two distinct events occur in figure 23 that contribute to the inability to assign a growth exponent to this sequence. First, at the top left in panel a, there is a white domain inside a black domain, which is inside a white domain. A merge between

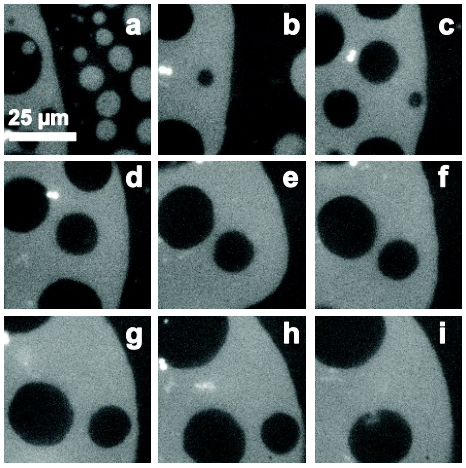

Figure 23: Sequence of images of coexisting liquid domains on the surface of a vesicle with a critical composition. Over the entire vesicle (although not in this cropped image), the area of dark and bright areas is roughly equal. No scaling operation can be performed to make the images appear the same. In other words, the images are not self-similar, and no growth exponent can be extracted. Corresponding morphologies are described by Fan et al. 2010 and in Camley and Brown 2011. Panel time points are as follows: a = 0s, b = 122s, c = 250s , d = 369s, e = 549s, f= 609s, g = 760s, h = , i = 971s.

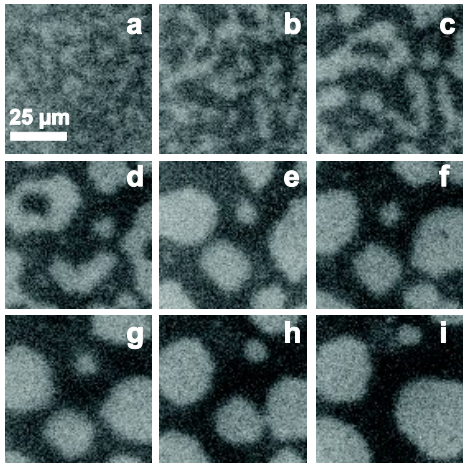

Figure 24: Coarsening of liquid domains in a membrane in which a miscibility transition was initiated by photo-oxidation of lipids. Panel time points are as follows: a = 0s, b = 3s, c = 6s, d = 9s, e = 12s, f = 15s, g = 18s, h = 21s, and i = 24s.

these two white areas cannot be described as a merge between two circular domains. Second, the entire vesicle has phase separated so that one side contains predominantly white domains on a black background (the right side of panel a in figure 23) and the other contains the opposite (left side). If the vesicle rotates, as appears to be happening in figure 23, the measured total perimeter will change as the boundary between the predominantly black and predominantly white regions drift out of the viewing window.

Triggering changes in T_{mix} via photo-oxidation

Jumps in the miscibility temperature (T_{mix}) of fluorescently-labeled membranes can be initiated by high light levels, which causes changes in membrane composition due to lipid photo-oxidation [Honerkamp-Smith et al. 2008]. In this method, the process of crossing a miscibility transition is entirely isothermal. Figure 24 shows domains coarsening after a transition initiated by photo-oxidation. In our setup, jumps initiated by high light levels are not superior to jumps initiated by a quench in sample temperature because the time scales of jumps are similar via both mechanisms. More importantly, once photo-oxidation is initiated, it is not easily curtailed. Maintaining a constant offset between T_{mix} and the sample temperature, which is required for quantitative analysis of growth exponents and diffusion coefficients, is difficult.

IV. Discussion

Within lipid vesicles, four natural length scales arise, and their interplay determines the rate of domain diffusion and coarsening. The first is the hydrodynamic length $L_h = \eta_{2D}/\eta_{3D}$. The second is the normalized domain size R. The third is the correlation length, ξ, for domains in a membrane near a miscibility critical point. Correlation length is inversely proportional to line tension. When correlation lengths are large and line tension is small, boundaries of domains fluctuate, resulting in noncircular domains. The fourth is the vesicle diameter, which ranges between 80 μm and 250 μm in the experiments here.

Here we find that fitting diffusion coefficients of circular, L_o domains in a free-floating unilamellar vesicle of DiPhyPC/DPPC/chol yields membrane viscosities of (3.3 ± 1.1) x 10^{-9} Pa s m (for $\phi < 0.3$ and vesicles in water), of (9.6 ± 3.0) x 10^{-9} Pa s m (for $\phi < 0.3$ and vesicles in dextran), and of (7.0 ± 3.3) x 10^{-9} Pa s m (for $0.4 < \phi < 0.6$ and vesicles in water). These values are in good agreement with previous values (η_{2D} = (5.5 ± 1.5) x 10^{-9} Pa s m and (4 ± 1) x 10^{-9} Pa s m) found by analyzing structure factors of composition fluctuations [Honerkamp-Smith et al. 2012] and by analyzing shape fluctuations of domain boundaries within membranes of the same three lipids at a similar lipid ratio and temperature [Camley & Brown 2010]. This agreement implies that reasonably accurate measurements of diffusion coefficients can be made over the range of domain sizes probed here, even though domains can appear crowded [Aliaskarisohi et al. 2010]. To give a sense of the effect of lipid ratios on membrane viscosity, lateral diffusion coefficients of lipids within L_d vs. L_o domains in vesicles and planar bilayers differ by a factor of ~2 – 10 [Kahya et al. 2005, Lindblom et al. 2009, Honigmann et al. 2010].

Measured values of membrane viscosities in other systems are on the same order of magnitude. In fluid membranes of DOPC/DPPC/chol, $\eta_{2D} \approx 5 \times 10^{-10}$ to 3×10^{-9} Pa s m (from [Petrov & Schwille 2008] using data from Cicuta et al. 2007). In fluid membranes of SOPC, η_{2D} = (3 ± 1) x 10^{-9} Pa s m [Dimova et al. 1999]. Taken together, these results imply that in many vesicle membranes in water, length L_h is on the order of 1 μm and that deviations from theories that treat domain diffusion as being entirely dominated by the effects of momentum dissipation into water are expected for even planar membranes. Previous work has suggested that such deviations are primarily due to confinement of domains on a curved surface and hydrodynamic interactions between domains [Aliaskarisohi et al. 2010].

We find that circular (or nearly circular) domains in our vesicles isothermally coarsen with a growth exponent of $\alpha \approx 1/3$, independent of whether the bulk fluid in which the vesicles are embedded is water or a more viscous solution containing dextran. These circular domains appear in membranes with $0.4 < \phi < 0.6$ at late times after a quench or in membranes with $\phi <$

0.3. Elongated domains that appear in our vesicle membranes with $0.4 < \phi < 0.6$ early after a quench isothermally coarsen with $\alpha \approx 1/2$.

A range of experimentally-measured growth exponents for circular domains has been published (Table 1). Previous measurements did not uniformly exclude cases with inconstant temperature, bulged domains, or small vesicles. In 2005, Saeki et al. reported $\alpha \approx 0.15$ using unilamellar vesicles of 35:35:30 DOPC/DPPC/chol with diameter ~50μm. The researchers noted that they observed domains that curved out of the membrane. Domains that bulge out of the spherical shell of the vesicle (known by a variety of names in the literature, including "dimples") interact through an elastic deformation of the surrounding membrane (Figure 4). Bulged domains are kinetically hindered from coarsening [Ursell et al. 2009]. In 2007, Yanagisawa et al. [Yanagisawa et al. 2007} reported $\alpha \approx 2/3$ using unilamellar vesicles of 40:40:20 DOPC/DPPC/chol with diameters ~10-150μm. Their value of $\alpha \approx 2/3$ applied to vesicles that did not exhibit "trapped coarsening" of bulged domains. They speculated that domains attract each other. Work by other groups [Honerkamp-Smith et al. 2008, Esposito et al. 2007] imply that no net attractive or repulsive interactions exist between unbulged domains since fluctuations in their boundaries fit normal capillary theory (Tobias Baumgart, personal communication). Our measurement of $\alpha \sim 1/3$ implies that there are no net attractive or repulsive interactions between domains. In 2007, Liang et al. reported $\alpha \approx 1/3$ for domains smaller than 1 μm and $\alpha \approx 1$ for domains larger than 1 μm within unilamellar vesicles of 1:1:1 bovine brain sphingomyelin/DOPC/chol with diameter > 20 μm. They speculated that a growth exponent of $\alpha \approx 1$ at long observation times could be explained if a merge of two domains triggered subsequent merges in the vicinity. Their longest observation time was roughly an order of magnitude shorter than the shortest run in figure 20a.

On the other hand, the growth exponents measured here for the coarsening of circular domains are in excellent agreement with predictions from theory and simulation, which give $\alpha = 1/3$ [Pitaevskii & Lifshitz 1961,Taniguchi 1996, Laradji & Sunil Kumar 2004, Laradji & Sunil

Kumar 2005]. Our results are limited to domains that have not yet grown to be the size of the vesicle; circular domains coarsen until membranes eventually contain only one L_d domain and one L_o domain. At this point, domains on opposite sides of the vesicle become coupled via hydrodynamics [Aliaskarisohi et al. 2010]. Even when hydrodynamics are neglected, deviations from a growth exponent of $\alpha = 1/3$ arise from geometry and/or poor statistics when domain and vesicle sizes are comparable. Figure 5 shows the results of a simulation in which domains diffuse on a spherical surface with $D \propto 1/r$ and coarsen purely by coalescence events after collisions. The surface was seeded with ~200 spherical caps at $\phi = 0.09$ and r was defined as the arc length from the center to the edge of each cap. Strong deviations from the overall growth exponent of $\alpha = 0.31$ occurred at short times during a size equilibration period and weak deviations occurred at long times, when the average domain radius divided by the vesicle radius reached values greater than $10^{-0.8}$, or ~16%. Deviations at long times are due to geometric reasons or to poor statistics, since few domains populate the vesicle at long times.

We find that $\alpha \sim 1/2$ at long time for vesicles with $0.4 < \phi < 0.6$ for which scaling is possible. In this growth regime, we find that changes in R are heavily influenced by changes in domain shape. The DPD simulations of Laradji & Sunil Kumar (2005) and Ramachandran et al. (2009) find $\alpha = 1/2$. The more complex simulations of Fan et al. (2010) and Camley & Brown (2010, 2011) find cases in which an apparent growth exponent of $\alpha = 1/2$ is observed, most notably when $R >> L_h$.

A range of experimental conditions can produce anomalously low or high values of α. The growth exponent will be too low (1) if domains bulge out of the membrane, (2) if the range between the experimental temperature and the membrane miscibility transition temperature (T_{mix}) is decreasing, either because photo-oxidation lowers T_{mix} or because the sample temperature is increasing, (3) if large domains are undercounted in a measurement of domain radius instead of domain perimeter, (4) if large domains are undercounted because vesicles are too small, or (5) if the viewing area is stuck on a glass substrate such that domains move too

slowly [Stottrup et al. 2004]. In contrast, the growth exponent will be too high (1) if the range between the experimental temperature and T_{mix} is decreasing, either because photo-oxidation increases T_{mix} (as can occur in the DOPC/DPPC/chol system), or because the sample temperature is decreasing, or (2) if flow of the exterior bulk solution brings domains frequently in contact with a substrate that has a preferential interaction with one phase vs. the other (data not shown). All of these experimental difficulties are surmounted in the system of free floating giant unilamellar vesicles of DiPhyPC/DPPC/chol used here.

Section 2: Chemical Education Research

Math assessment of UW chemistry students shows mathematics skills atrophy with disuse

I. Introduction

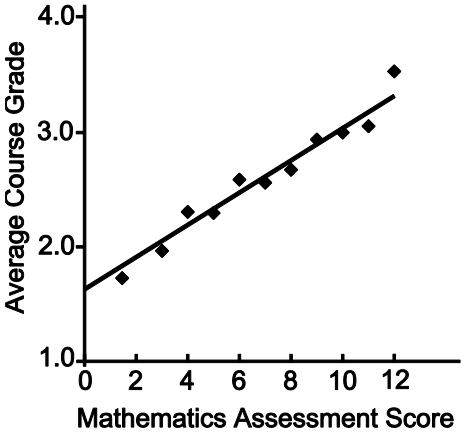

Figure 25: Leopold & Edgar (2008) show that the average course grade is correlated to 1st semester general chemistry student aptitude on a mathematics assessment.

A strong math background is important for students' success in chemistry. My results (Figure 26 & 27) support the results of other researchers who have shown that a student's grade in General Chemistry correlates with the student's performance in math, including remedial math [Rixes & Pickering 1985, Ozsogomonyan et al. 1979, Leopold & Edgar 2008 (figure 25)]. In my own teaching experiences and in those of others [Koopman et al.2008], time devoted to teaching chemical concepts was not maximized because students needed to be taught basic mathematics (e.g. logarithms). Leopold & Edgar point out that students commonly

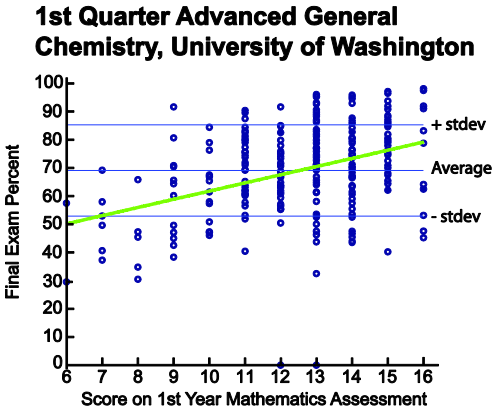

Figure 26: The final exam grade for first quarter advanced General Chemistry (Chem 144) students correlated with scores on the mathematics assessment. The green line is a best fit line to the data and the blue lines show the average final exam grade and the standard deviations from the average.

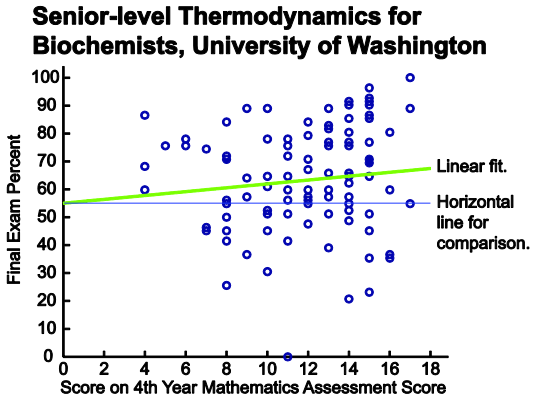

Figure 27: The final exam grade for first quarter senior level Thermodynamics for Biochemists correlated with scores on the mathematics assessment. The green line is a linear best fit and the blue line is a horizontal line for comparison.

misunderstand logarithms as being equivalent to square roots. This single misunderstanding can affect a student's understanding of pH. Students should have learned all math needed for the first term of college chemistry while in high school, yet only half of students in Leopold & Edgar's study could simplify the term $\ln(e^x)$.

In order to evaluate UW students' math preparation, Sarah Keller developed a math quiz (Figure 28) covering only the remedial math required in General Chemistry. Students performed poorly (Figure 29 & 30), on par with similar students at the University of Minnesota [Leopold & Edgar 2008 (Table 5)]. The results of this study motivated us to study the mathematics ability of students in undergraduate General Chemistry and senior level Thermodynamics for Biochemists. In all cases, the students did not show high mathematical aptitude in areas where prerequisites would suggest that the students should excel. We have modified the mathematics quiz into a mathematics intervention that tests the students' mathematics aptitude, then guides students to grade their own work and to retry problems after seeing the key. The intent is that students will realize they have previously learned the math required to be successful in chemistry, but that they must work to remind themselves how to do it. In this way, we can hopefully help students excel by learning to review prerequisite material before a course.

Table 5: Selected questions from Leopold & Edgar (2008) showing results from the mathematics quiz given to second quarter general chemistry students.

Question text	Correct Answer	Most Common Wrong Answer	% Students Correct
What is the log of 100?	2	10	50%
What is log(ab)?	log a + log b	(log a)(log b)	64%
What is $(2 \times 10^4)(3 \times 10^2)$?	6×10^6	6×10^8	86%
Given $d = \frac{a-b}{c-b}$, simplify	Cannot be simplified	d = a/c	72%

Nationally, there is a current emphasis on increasing diversity within STEM fields [Malcom 2010]. We found that women performed slightly worse in our Chem 144 math quiz

(data not shown), as in the Minnesota study [Leopald & Edgar 2008]. Similarly, we found that students who qualified for a UW program serving first generation college students, students who are economically disadvantaged, and/or are members of an underrepresented ethnic minority also performed slightly worse (data not shown). These students are more likely to

UNGRADED MATHEMATICS BRUSH-UP Name: _____

This pre-test is a quick way for us to understand your background so that we can teach you better. Evaluate and/or simplify the expressions. Do not use a calculator. Do not convert fractions to decimals. Work alone. At the end of 20 minutes, write either "don't know" or "out of time" in any blanks. You will not be graded on how many of your answers are correct; you will earn full credit by filling in all blanks.

1) $\frac{(5/9)}{(3/2)} = ?$ Ans: 10/27 (90% of students got it right)

[Did not accept (10×10^{-23}) or ($1/10^{22}$). Want sci. notation]

2) $\frac{(1 \times 10^{-15})}{(5 \times 10^{12})(2 \times 10^{-6})} = ?$ Ans: 10^{-22} OR 1×10^{-22} (43% right)

3) $(a^3bc^{-1})(ab^{-3}) = ?$ Ans: $a^3b^{-2}c^{-1}$ OR $a^3/(b^2c)$ (80% right) [$a^x a^y = a^{x+y}$]

4) $(a^{2/3})^{4/3} = ?$ Ans: $a^{8/9}$ (76% right) [$(a^b)^y = a^{xy}$]

5) $\log_{10}100 = ?$ Ans: 2 (51% right) [$\log_a(a^b) = b$]

6) express as one term: ln 7 – ln 4 = ? Ans: ln(7/4) (63% right) [ln a – ln b = ln(a/b)]

7) $\sum_{x=1}^{3} x^2 = ?$ Ans: 14 (85% right) [1+4+9 = 14]

8) solve for °F in the equation °C = (°F – 32)(5/9) to yield the equation °F = ? Ans: (9/5) C + 32 (89% right)

[Also accepted 9/5 (C + 160/9) and (C / (5/9) + 32)]

9) A courtyard contains 10 identical paving stones and 3 blind mice that scurry around at random. What is the probability that at some instant all three mice are found on stone #3? Ans: 1/1000 or 0.001 or 0.1% (63% right)

[(probability of mouse 1) x (mouse 2) x (mouse 3) = $1/10 \cdot 1/10 \cdot 1/10$]

10) Formula for the area of a circle with radius *r* is… Ans: *Area* = πr^2 (94% right)

11) Number of significant digits in the value 2700 is… Ans: 2 (93% right)

For each relationship below, write the letter corresponding to the single best description of the relationship.

	Letter of your answer:
12) $A = B$	a (99.2% right)
13) $A > B$	e (99.6% right)
14) $A \equiv B$	c (86% right)
15) $A \propto B$	l (39% right)
16) $A \approx B$	b (86% right)

a. *A* is equal to *B*.
b. *A* is roughly equal to *B*.
c. *A* is defined to be exactly equal to *B*.
d. *A* is the average of all *B* values.
e. *A* is greater than *B*.
f. *A* is less than *B*.
g. *A* is 10 times greater than *B*.
h. *A* is 10 times less than *B*.
i. *A* is a subset of *B* values.
j. *A* is the inverse of *B*.
k. *A* is the absolute value of *B*.
l. *A* is proportional to *B*.

Figure 28: Mathematics Brush-up designed by Sarah Keller, showing correct answers and percent of correct responses.

have poor math backgrounds [Tapia 2009]. By increasing the math ability of all students, we hope to increase the numbers of STEM majors from diverse backgrounds. Using an adaptive homework program such as ALEKS may allow students with underdeveloped mathematical skills to individually practice the mathematics required along with the chemistry with which it is paired.

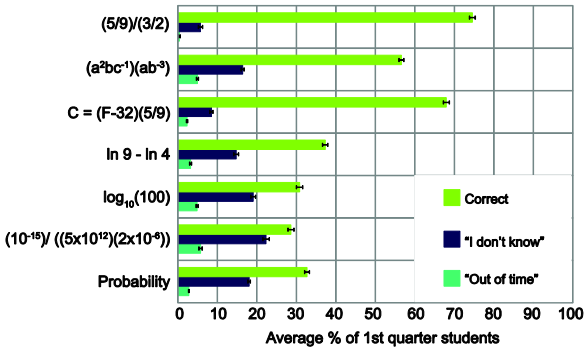

Figure 29: Percentage of 1st quarter General Chemistry students who answered the selected questions correctly, reported not knowing the answer, or reported being out of time.

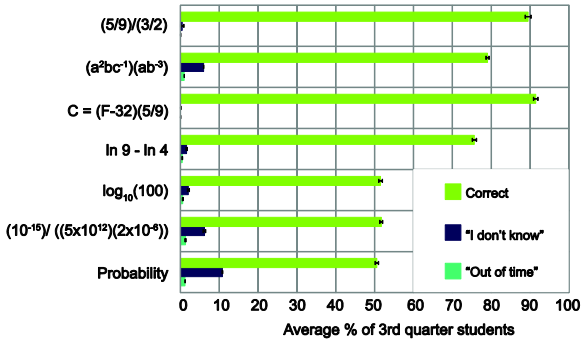

Figure 30: Percentage of third quarter General Chemistry students who answered the selected questions correctly, reported not knowing the answer, or reported being out of time.

II. Details of Studies

First year chemistry mathematics quizzes

Leopold and Edgar showed that second semester General Chemistry students have low proficiency in pre-calculus. They gave a 30 minute, surprise, calculator-free quiz of 20 multiple choice questions to 360 students (Table 5). Our first study, in 2008, involved administering a similar mathematics quiz to two populations of students to determine if our students had similar mathematics aptitudes to those in Leopold & Edgar's study (Figure 28). The quiz was given during the first TA-led recitation session ("quiz section") in Autumn of 2009 to 1st quarter General Chemistry students (Figure 29) and 3rd quarter General Chemistry students (Figure 30). Students in both groups appear to have little retention of logs, scientific notation, and probability. In 2009, we gave a similar mathematics quiz to ~240 first quarter Advanced General

Table 6: Selected questions from the mathematics assessment given to first and second quarter advanced General Chemistry students.

Question text	Correct Answer	% Students Correct in 1st quarter	% Students Correct in 2nd quarter
$(a^2bc^{-1})(ab^{-3})$ = ?	$a^3b^{-2}c^{-1}$	80%	87%
Express as one term: ln7 – ln 4 = ?	ln 7/4	64%	85%
A∝B means what? (multiple choice)	A is proportional to B	39%	84%

Chemistry (Chem 144) students (Table 6). Our results were not very different from what Leopold and Edgar found at the start of 2nd semester, showing that students have problems in areas of logarithms, exponents, and algebra. A second, similar math quiz was given to the same group of students at the start of 2nd quarter advanced General Chemistry (Chem 154). Although the average number of students with the correct answer increased, the percentage of students with correct answers was around 85% in all areas. This indicated that during the term in which students took General Chemistry they either learned or remembered some of the math needed for chemistry, but that they were still not as proficient in areas of mathematics required for full mastery of chemical concepts like pH, proportionality, and probability.

Senior year chemistry math refreshers

In 2010, 105 students in senior level Thermodynamics for Biochemists (Chem 452) were given an untimed, self-graded, calculator-free homework assignment in order to assess their ability to complete pre-calculus and calculus math problems. The students tried a set of problems and reported whether they did the problem correctly by looking at an answer key and a description of the correct way to finish the problem. The students then tried questions similar to the original problems. Second attempt scores were higher than first attempt scores for all questions (Figure 31). The pre-requisites for Chem 452 are second quarter General Chemistry, second quarter Calculus, and second quarter Physics. This shows that students are forgetting the mathematics required to master senior level Physical Chemistry. Many 4th year students have not taken a math course in at least 2 years. This suggests that expecting students to enter Physical Chemistry courses with a high level of mathematics proficiency is not plausible.

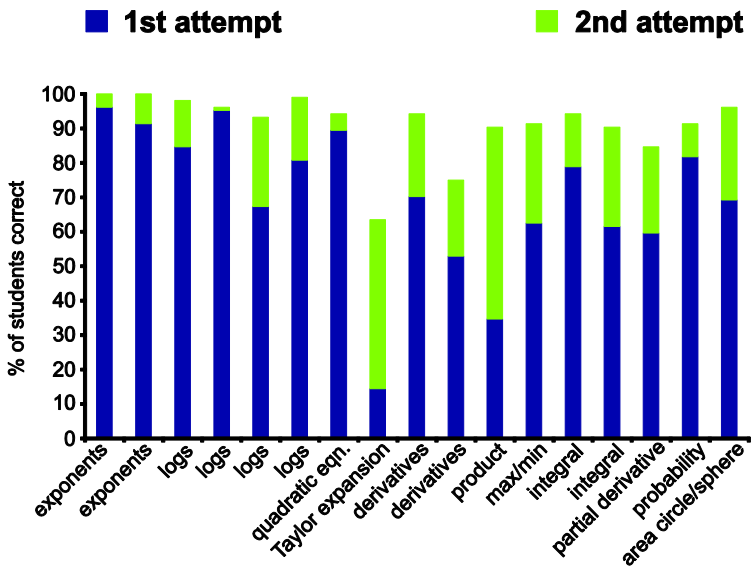

Figure 31: Mathematics brush-up results for senior level Thermodynamics for Biochemists (Chem 452) students. Blue bars show the percentage of students who got the questions correct. Green bars shows the additional percentage of students who answered the question correctly the second time.

In 2009, we gave an online mathematics assessment to students in first quarter Physical Chemistry for Biochemists (Chem 452). These students reported similar results as those who had completed paper assessments (data not shown). Students were asked to self-report whether they got the answer correct, were on the right path, or got the answer wrong. Interestingly, many students incorrectly inflated their own self-grading results even though full credit was given simply for completing the assignment. Several students had incorrect answers recorded, yet reported they did it correctly. Students were told that their responses would help the professor assess how to teach best. Students who inflated scores encouraged the professor to skip remedial mathematics material that those students needed. Even more interesting were students who had entirely incorrect answers and yet reported that they were on the right path. This raises questions regarding whether students did not want to admit being wrong, or if the students did not realize how different their answers were from the actual answer. This also raises issues of self-reporting among students. It would be interesting to follow up on self-reporting cases by interviewing students to assess why they reported what they did.

SUPER-QUICK SURVEY CHEM453 Name : *Not needed! Anonymous!*

Please help us improve Chem 453 by taking this survey. The same survey was recently published in an education journal, and we're trying to discern if the results in the paper are relevant to this class.

Statement		Strongly Agree 1	2	3	4	Strongly Disagree 5
(a) I feel confident applying my math skills to chemistry.	a)					
(b) I feel that I have sufficient math to succeed in this course.	b)					
(c) I enjoy math problems in chemistry.	c)					
(d) I think I should brush up on my math for this course.	d)					
(e) I think this class will have too much math in it.	e)					
(f) I feel comfortable seeking help from others to clarify difficult material.	f)					
(g) I feel comfortable explaining how I solve chemical problems to others.	g)					
(h) I feel that the amount of math in this course could lower my grade.	h)					
(i) I enjoy challenges.	i)					
(j) I am only taking this course because it is required.	j)					
(k) The grade I receive in this course is less important than what I learn.	k)					
(l) I expect to learn a lot in this course	l)					

… And now, a completely unrelated question that will help us tailor course subject matter to you:

After graduating, do you hope/intend to have a career in health care, that involves directly caring for patients?

YES _____ NO _____

Figure32: Survey given to students in Thermodynamics for Biochemists (Chem 452).

Senior year chemistry opinion survey

In 2009, ~200 students completed anonymous surveys at the start of senior-level Thermodynamics for Biochemists (Chem 452) concerning what they expected of the math content in the course. The survey was a series of questions students answered on a Likert scale, with 5 corresponding to "strongly agree" and 1 corresponding to "strongly disagree" (Figure 32). Most students seemed worried about the mathematics they expected to see in the course (Figure 33), which suggests they recognized the challenge of the course material. The most telling data is the disagreement between what the professors predicted that students would say and what the students actually reported. Professors overestimated the confidence students would have about their mathematics proficiency and overestimated the level of engagement the students expect to have in the course, which corresponds to the level of agreement reported with the statement "I enjoy math problems in chemistry".

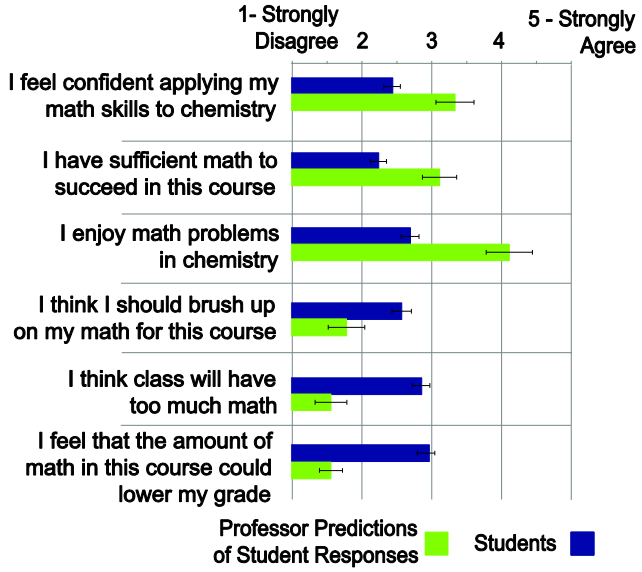

Figure 33: A survey given to first quarter Thermodynamics for Biochemists students (blue) to assess how students feel about mathematics in the course. The same survey was given to professors (green) to predict what students would say.

SAT and final GPA correlations

Plots of mathematics brush-up scores versus SAT scores contain significant scatter, although there is a slight positive correlation between how well a student did on the mathematics portion of the SAT and how well a student did on the brush-up (Figure 34). This suggests that doing well on the SAT does not necessarily mean a student has a high retention of

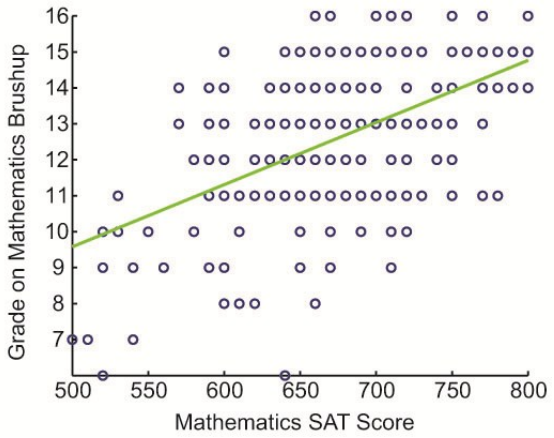

Figure 34: General Chemistry mathematics brush-up score correlated with Mathematics SAT scores.

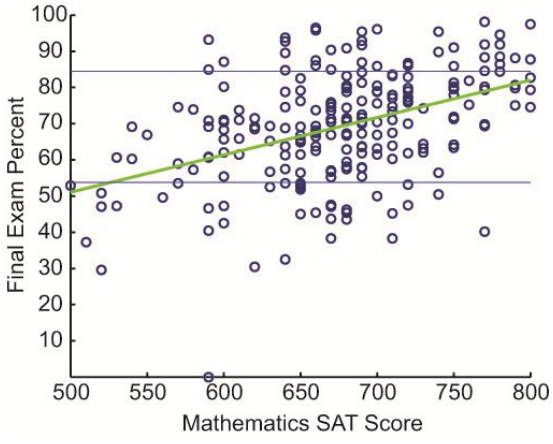

Figure 35: Final exam percent earned in General Chemistry correlated with Mathematics SAT scores.

math skills. There is also significant scatter in plots of final exam scores versus SAT scores, with a slight correlation between doing well on the SAT and doing well on the final exam (Figure 35). This once again suggests that doing well on the SAT does not guarantee that a student will do well in Chemistry. Andrews and Andrews (1979) found similar results showing that chemistry grades correlate positively to SAT scores, but with significant scatter. They concluded that a high SAT score in mathematics does not necessarily guarantee a student a high GPA in General Chemistry, whereas a low SAT score is more predictive of doing poorly in General Chemistry.

ALEKS data from Colleen Craig

For the past few years, at the University of Washington homework for the large General Chemistry courses was conducted via the website Webassign. Webassign is an online homework platform that mirrors traditional homework, and consists of about 10 homework questions chosen by the instructor for the students to complete weekly. During the past few quarters, online homework has been switched to ALEKS. ALEKS is an adaptive program that assesses how well the student performs on an objective and modifies subsequent questions accordingly. In "learning mode", the students have access to tutorials for help with the

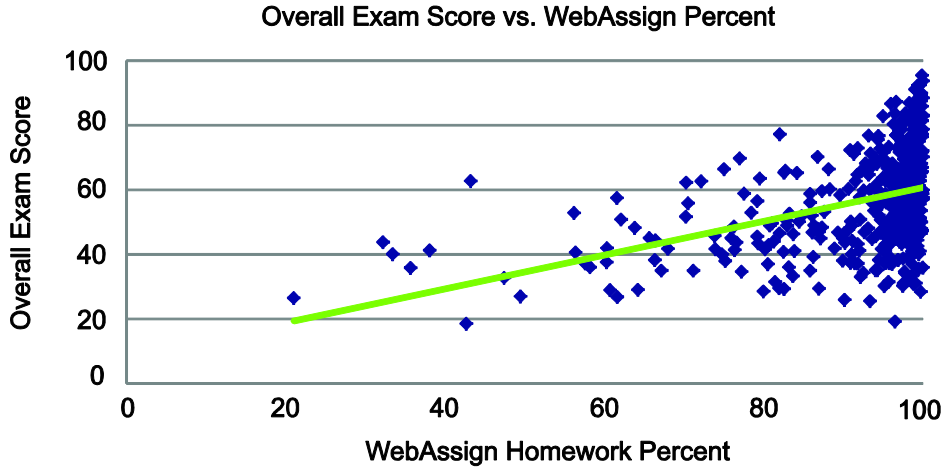

Figure 36: Overall exam scores correlated with Webassign homework percentage for first quarter General Chemistry (Chem 142) students.

questions. Once a student works through the objectives in learning mode, ALEKS switches to "assessment mode". The assessment mode measures what a student can do, cannot do, and what the student should learn next. This type of homework allows a student to learn at his/her own pace, and it brings up old topics to remind students how to solve questions they have not seen recently. This repetition continues until the student shows mastery in the topic. Data taken in 2010 shows that final course grades correlate weakly with Webassign grades (Figure 36, unpublished data of Colleen Craig). Data taken in 2011 with the same instructor shows grades correlate more strongly with mastery in ALEKS assessment (Figure 37, unpublished data of Colleen Craig). This is not to say that students perform better using ALEKS than with Webassign, but doing well on ALEKS is a better indicator of how students will perform on the final exam. Therefore, ALEKS provides more meaningful feedback to students on their mastery of the material. The main difference between the data sets taken in the two quarters lies in the scatter. Webassign grades are more scattered with respect to final exam grades, whereas the ALEKS grades are more tightly correlated around the average line.

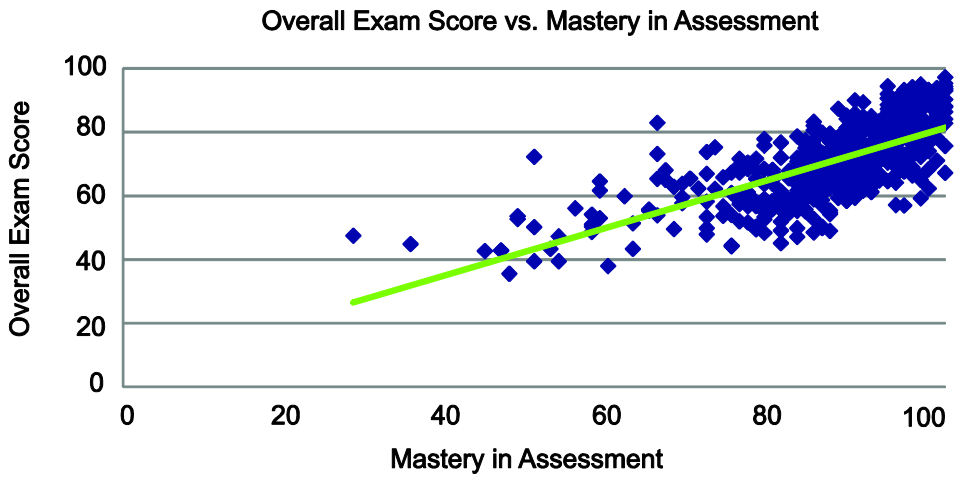

Figure 37: Overall exam score correlated with mastery in assessment in ALEKS for first quarter General Chemistry (Chem 142) students. Mastery in ALEKS assessment is measured by the overall progress in assessment mode that the student made through the quarter.

III. Discussion and conclusions

Foremost, I observed that students had lower mathematics aptitudes than I expected given the time that had elapsed between students' required mathematics courses and science courses. Perhaps the lack of a strong correlation between SAT scores and their scores on the mathematics refresher (Figure 34) could be due to the fact that some students took math immediately before the SAT while in high school, and then did not take math again, or at least until they completed General Chemistry. It would be interesting to compare math refresher scores and the time that elapsed between General Chemistry and students' last math to see if there is a correlation. There is evidence that students need reminders after delays between math courses. In the case of the senior level Chemistry student scores on the mathematics refresher, we know that students have previously learned the mathematics they need because they have passed prerequisites, but that they have forgotten it (Figure 31). Once students are reminded of the process of working through mathematical problems, the number of correct answers always increased. Students report that they know their math is not as good as it needs to be (Figure 33), but professors may not be aware of students' lack of confidence. Students who lack confidence in their own ability do not do as well as students who have high confidence in chemistry [Lewis et al. 2009]. Reminding the students of what they know may be an effective tactic to increase their mathematics confidence so that they perform better in chemistry classes.

Given my results above, one avenue of research that I think would be valuable would be to design an intervention of math questions tailored to individual homework sets, much as in Koopman et al. 2008. This intervention could be applied both to General Chemistry and to Physical Chemistry for Biochemists (Chem 452) through an adaptive homework program such as ALEKS. This procedure would remind students of forgotten math without taking class time.

One aspect of the intervention that will require careful design is to ensure that students are encouraged to continue in chemistry even if they perform poorly on a math quiz. Students

with a positive opinion of their own abilities perform better in General Chemistry [Lewis et al.]. In our Chem 144 study in the Autumn quarter of 2009, students with more than 5 out of 16 questions incorrect were sent a friendly letter encouraging them to get help through several resources available on campus. We learned through anecdotal student feedback to teaching assistants that (at minimum) several students contemplated dropping their Chemistry class out of fear of failure because of poor performance on the math quiz, but that no students did (at least in this small set). It would be helpful to hold interviews with students in order to ascertain how to best motivate them to review the required math without scaring them away from majoring in STEM fields.

My data suggests that a fruitful avenue of research may lie in optimizing the sequence used to teach math and science. If we want to make students better scientists, they require a firm grasp of certain mathematical concepts. It may help students if math and science education is far more integrated than it is right now. A short term goal would be to require students to complete a mathematics assessment that highlights to each student what math they will need to review for the course. Then, throughout the course, students should practice important mathematical skills required for success. This procedure can be easily added to an adaptive program such as ALEKS, since it is already designed to remind students of topics they have already learned throughout the term. In this way we can help students practice a skill that is essential to success in Chemistry, but has been atrophied by disuse.

References

Aliaskarisohi, S., Tierno, P., Dhar, P., Khattari, Z., Blaszczynski, M., & Fischer, T. M. (2010). On the diffusion of circular domains on a spherical vesicle. *J. Fluid Mech., 654*, 417 - 451.

Andrews, M., & Andrews, L. (1979). First-year chemistry grades and SAT math scores. *J. Chem. Educ., 56*, 231-232.

Camley, B. A., & Brown, F. L. (2010). Dynamic simulatoins of multicomponent lipid membranes over long length and time scales. *Phys. Rev. Lett., 105*(14), 148102.

Camley, B. A., & Brown, F. L. (2011). Dynamic scaling in phase separation kinetics for quasi-two-dimensional membranes. *J. Chem. Phys., 135*, 225106.

Camley, B. A., Esposito, C., Baumgart, T., & Brown, F. L. (2010). Lipid bilayer domain fluctuations as a probe of membrane viscosity. *Biophys. J., 99*, L44 - L46.

Cicuta, P., Keller, S. L., & Veatch, S. L. (2007). Diffusion of liquid domains in lipid bilayer membranes. *J. Phys. Chem. B, 111*, 3329-3331.

De Koker, R. (1996). *Ph. D. thesis.* Stanford University.

Dimova, R., Dietrich, C., Hadjisky, A., Danov, K., & Pouligny, B. (1999). Falling ball viscometry of giant unilamellar vesicle membranes: Finite-size effects. *Eur. Phys. J. B., 12*, 589 - 598.

Esposito, C., Tian, A., Melamed, S., Johnson, C., Tee, S. Y., & Baumgart, T. (2007). Flicker spectroscopy of thermal lipid bilayer domain boundary fluctuations. *Biophys. J., 93*, 3169 - 3181.

Fan, J., Han, T., & Haataja, M. (2010). Hydrodynamic effects on spinodal decomposition kinetics in lipid bilayer membranes. *J. Chem. Phys., 133*, 235101.

Garcia-Saez, A. J., & Schwille, P. (2008). Fluorescence correlation spectroscopy for the study of membrane dynamics and protein/lipid interactions. *Methods*, 116-122.

Gomez, J., Sagues, F., & Reigada, R. (2008). Use of an enhanced bulk diffusion-based algorithm for phase separation of ternary mixture. *J. Chem. Phys.*, 184115-1-184115-9.

Hahn, K. E., & Polik, W. F. (2004). Factors influencing success in physical chemistry. *J. Chem. Educ., 81*(4), 567-572.

Honerkamp-Smith, A. R., Cicuta, P., Collins, M. D., Veatch, S. L., den Nijs, M., Schick, M., & Keller, S. L. (2008). Line tensions, correlation lenths, and critical exponents in lipid membranes near critical points. *Biophys. J., 95*, 236-246.

Honerkamp-Smith, A. R., Machta, B. B., & Keller, S. L. (2012). Experimental observations of dynamic critical phenomena in a lipid membrane. *Phys. Rev. Lett.*, In Press.

Honigmann, A., Walter, C., Erdmann, F., Eggeling, C., & Wagner, R. (2010). Characterization of horizonal lipid bilayers as a model system to study lipid phase separation. *Biophys. J., 98*, 2886 - 2894.

Hughes, B. D., Pailthorpe, B. A., & White, L. R. (1981). The translational and rotational drag on a cylinder moving in a membrane. *J. Fluid Mech., 110*, 349-372.

Kahya, N., Scherfeld, D., & Schwille, P. (2005). Differential lipid packing abilities and dynamics in giant unilamellar vesicles composed of short-chain saturated glycerol-phospholipids, sphingomyelin and cholesterol. *Chem. Phys. Lipids, 135*, 169 - 180.

Kestin, J., Sokolov, M., & Wakeham, W. A. (1978). Viscosity of liquid water in the range -8 C to 150 C. *J. Phys. Chem. Ref. Data, 7*(3), 941 - 948.

Koopam, L., Brouwer, N., & Heck, A. (2008). Remedial mathematics for quantum chemistry. *J. Chem. Educ., 85*(9), 1233-1236.

Laradji, M., & Sunil Kumar, P. B. (2004). Dynamics of domain growth in self-assembled fluid vesicles. *Phys. Rev. Lett., 93*(19), 198105.

Laradji, M., & Sunil Kumar, P. B. (2005). Domain growth, budding, and fission in phase-separating self-assembled fluid bilayers. *J. Chem. Phys., 123*, 224902.

Leopold, D. G., & Edgar, B. (2008). Degree of mathematics fluency and success in second-semester introductory chemistry. *J. Chem. Educ., 85*(5), 724.

Lewis, S. E., Shaw, J. L., & Heitz, J. O. (2009). Attitude counts: self-concept and success in general chemistry. *J. Chem. Educ., 86*(6), 744-749.

Liang, X., Li, L., Qiu, F., & Yang, Y. (2010). Domain growth dynamics in multicomponent vesicles composed of BSM/DOPC/Cholesterol. *Physica A, 389*, 3965 - 3971.

Lindblom, G., & Oradd, G. (2009). Lipid lateral diffusion and membrane heterogeneity. *Biochim. Biophys. Acta, 1788*, 234 - 244.

Malcom, S. M. (2010). Written testimony before the committee on science and technology subcommittee on research and science education. *Washington, DC: House of Representatives.*

Otsu, N. (1979). A threshold selection method from gray-level histograms. *IEEE Sys. Man. Cybern., 9*(1), 62-66.

Ozsogomoyan, A., & Loftus, D. (1979). Predictors of general chemistry grades. *J. Chem. Educ., 56*(3), 173-175.

Petrov, E. P., & Schwille, P. (2008). Translational diffusion in lipid membranes beyond the Saffman-Delbruck approximation. *Biophys. J., 94*(5), L41-L43.

Pike, L. J. (2006). Rafts defined: a report on the Keystone symposium on lipid rafts and cell function. *J. Lipid Res., 47*, 1597.

Pitaevskii, L. P., & Lifshitz, E. M. (1961). *Physical Kinetics.* Oxford, England: Pergamon Press.

Przybylo, M., Sykora, J., Humpolickova, J., Benda, A., Zan, A., & Hof, M. (2006). Lipid diffusion in giant unilamellar vesicles is more than 2 times faster than in supported phospholipid bilayers under indentical conditions. *Langmuir, 22*, 9096-9099.

Ramachandran, S., Komura, S., & Gompper, G. (2010). Effects of an embedding bulk fluid on phase separation dynamics in a thin liquid film. *Europhys. Lett., 89*, 56001 - 56006.

Ramachandran, S., Laradji, M., & Sunil Kumar, P. B. (2009). Lateral organization of lipids in multi-component liposomes. *J. Phys. Soc. JPN., 78*(4), 041006.

Rixes, J. S., & Pickering, M. (1985). Freshman chemistry as a predictor of future academic success. *J. Chem. Educ., 62*(4), 313-315.

Saeki, D., Hamada, T., & Yoshikawa, K. (2006). Domain-Growth Kinetics in a Cell-Sized Liposome. *J. Phys. Soc. Japan*, 013602-1-013602-3.

Saffman, P., & Delbruck, M. (1975). Brownian Motion in biological membranes. *Proc. Natl. Acad. Acad. Sci.*, 3111-3113.

Seki, K., Ramachandran, S., & Komura, S. (2011). Diffusion coefficient of an inclusion in a liquid membrane supported by a solvent of arbitrary thickness. *Phys. Rev. E, 84*(2), 021905.

Spencer, H. E. (1996). Mathematical SAT test scores and college chemistry grades. *J. Chem. Educ., 73*(12), 1150-1153.

Stottrup, B. L., Veatch, S. L., & Keller, S. L. (2004). Nonequilibrium behavior in supported lipid membranes containing cholesterol. *Biophys. J., 86*(5), 2942 - 2950.

Taniguchi, T. (1996). Shape deformation and phase separation dynamics of two-component vesicles. *Phys. Rev. Lett., 76*, 4444-4447.

Tapia, R. A. (2009). Minority students and research universities: How to overcome the 'mismatch'. *Chron. Higher Ed., 55*(29), A72.

Tian, A., Johnson, C., Wang, W., & Baumgart, T. (2007). Line tension at fluid membrane domain boundaries measured by micropipette aspiration. *Phys. Rev. Lett., 98*, 208102 - 208104.

Turner, M. S., Sens, P., & Socci, N. D. (2005). Nonequilibrium Raftlike Membrane Domains under Continuous Recycling. *Phys. Rev. Lett., 95*, 168301.

Ursell, T. S., Klug, W. S., & Phillips, R. (2009). Morphology and interaction between lipid domains. *Proc. Natl. Acad. Sci. USA, 106*(32), 13301-13306.

Veatch, S. L., & Keller, S. L. (2003). Separation of liquid phases in giant vesicles of ternary mixtures of phospholipids and cholesterol. *Biophys. J., 85*, 3074-3083.

Veatch, S. L., Cicuta, P., Sengupta, P., Honerkamp-Smith, A. R., Holowaka, D., & Baird, B. (2008). Critical fluctuations in plasma membrane vesicles. *ACS Chem. Bio., 3*(5), 287-293.

Vind-Kezunovic, D., Neilsen, C. H., Wojewodzka, U., & Gniadecki, R. (2008). Line tension at lipid phase boundaries regulates formation of membrane vesicles in living cells. *Biochim. Biophys. Acta, 1778*, 2480-2486.

Wintersmith, J. R., Zou, L., Bernoff, A. J., Alexander, J. C., Mann Jr., J. A., Koojiman, E. E., & Mann, E. K. (2007). Determination of interphase line tension in Langmuir films. *Phys. Rev. E., 75*, 061605.

Yanagisawa, M., Imai, M., Masui, T., Komura, S., & Ohta, T. (2007). Growth Dynamics of Domains in Ternary Fluid Vesicles. *Biophys. J.*, 115-125.

Appendix A: Matlab code

Track_vesicle

```
function track_vesicle(fileName, diskrad)
% This m-file is for tracking and centering a tif stack of vesicle images
%CENTER CROP CALLED AT END!!!! You need to run this as a function, but
%beforehand you need to input the folder and home. You can leave diskrad
%blank if you are unsure of the radius of the in-focus area of the vesicle.
%You will want to change the disk thickness depending on your bright ring
%around the in-focus area of the vesicle. See line 50
%Needs Avesiclemask2.m, center_crop.m
%Program originally written by Aurelia R. Honerkamp-Smith and modified by Cynthia A. Stanich
%It includes a 12-line section written by Matthew Blosser that
% finds the radius of the in-focus area
FOLDER = ('C:\Program Files (x86)\MATLAB\R2007a Student\work\diffusion\Ostwald\Water\2-16-10\v2')%Where
the movies are
HOME = ('C:\Program Files (x86)\MATLAB\R2007a Student\work'); %Where work files are
cd(FOLDER) %Change the directory to where the movies are
file = [fileName];
%%%THIS IS FOR TIFFS - It should not be required to change anything below this line
fileinfo=imfinfo(file);
moviesize=fileinfo.Width;
lengthmovie=max(size(fileinfo));
in=zeros(moviesize,moviesize,lengthmovie,'uint16');
        for k=1:lengthmovie;
in(:,:,k)=imread(file,k);
        end
%Have loaded in the movie.  Now, take each frame and find the center of the vesicle.
%Choose a radius, then find the center.
      figure(200)
      firstone = in(:,:,1);
      imshow(firstone)
      hold on
%make starting guess
       start = ginput(1);  %this line lets you click on the first picture to choose the center.
         if nargin < 2 %If you didn't know the radius of the in-focus area of the vesicle
            %(which is almost always the case), then this will ask you to do a second click.
            %Click the edge of the bright ring.
         edge = ginput(1);
         diskrad = ((start(1) - edge(1))^2 + (start(2) - edge(2))^2)^(1/2); %Calculates the radius of the in-focus area of
the vesicle.
         end
         for k=1:max(size(fileinfo))
         clear s pic
         pic = in(:,:,k);
         s = size(pic);
         s1 = double(pic);
          %figure(11)                  %**remove this figure to run faster**
          %imagesc(pic), axis square
          %hold on
 %%%%Start Matt Blosser's code that finds the radius of the in-focus area%%%%%%
```

```
        diskwidth = 10;
         vars(1) = start(1); %Your first click, center of vesicle
         vars(2) = start(2); %second click, if needed, edge of vesicle.
         if nargin == 1
            vars(3) = diskrad;
           testFcn = @(vars)Avesiclemask2(vars, s1, diskwidth); %Calls a program written by Aurelia
           fit = fminsearch(testFcn, vars);
         else
           testFcn = @(vars)maskFixedRad(vars, s1, diskrad, diskwidth);
           fit(3) = diskrad;
           fit = fminsearch(testFcn, vars);
         end %End Matt's code

         if nargin > 1
            fit(3) = diskrad;
         end
        if (fit(1)<=0) | (fit(2) <= 0) | fit(1) >= s(1) | fit(2) >= s(2)
           ERROR = 1
             xxx %crash program if center is outside picture
        end
        figure(200)
        plot(fit(1), fit(2), 'or', 'LineWidth', 3)
        hold on
        savefit(k,:) = fit;%round(fit); %Matt had the round function but you shouldn't use it.
        picsize(k,:) = s;
        k
        start = fit;
       % saverad(k)=diskrad;
      end
    save([ file '_center.dat'], 'savefit', '-ascii', '-tabs', '-double');  %Saves the center coordinates
    save([ file 'diskrad.dat'], 'savefit', '-ascii', '-tabs', '-double');  %Saves the radius per frame
%clear savefit center picsize final
   close all
  figure (42)
   plot(savefit(:,3))

   cd(HOME)
 center_crop  %calls the center_crop file to create the centered and cropped movie file.
```

For manual_track.m, the code to run the program is very similar. The input for the function is the full address of the movie.

Param_difco

```
%%This is a typical example of a param file for running programs to measure diffusion constants.
%Written by Cynthia A. Stanich.
close all
clear
firstgo=0; % set this to 1 to select the region of interest and check domains on first frame.
TEST=0; %if  TEST=1, then program will stop (crashing) at the right point to
%show/check feature detection.
useframes=10; %We tested 5 frames, 10 frames, and 20 frames at a time.
%Useframes is important because it chooses how many frames are used in the
%diffusion constant calculation of mean squared displacement/time.
% It turns out that 10 frames and 5 frames
% give similar results, whereas 20 starts to deviate. Since 10 frames
%causes less work, we used 10 in all of my work.
 cut=16;   % this is the  distance in pixels that will be accepted as maximum
%movement in successive frames
maxdiam=400; % this is the maximum area allowed for an object. used to
%through away any large patches from image analysis.
WHITEDOMAINS=0; % set =1 if domains are bright on dark background in original movie file, set =0 otherwise
%The next two variables are set to zero because we center the movies before we measure.
inputgrossDY= 0;%108 ;  DY is the column (so horizontal movement)  negative if image is moving to right
inputgrossDX= 0;%-142 ;  DX is the row (so vertical movement) negative if image is moving downwards
cd('C:\Program Files (x86)\MATLAB\R2007a Student\work\')
hd=cd;
traj={'q2Thesis'}; %Name your output folders.
file='q2_00.tif'; %The name of the movie you are using.
% the movie above must be in the folder below.
filedirectory=['C:\Program Files (x86)\MATLAB\R2007a Student\work\diffusion\Comp1\Water\11-10-
09\v2\q2\movies'];
%NeedsAureliaGradient=1;  %uncomment this line if the movie has an uneven background.
using_previous_choice_of_domains=0; % make = 1 if you already clicked the domains for these 10 frames.
threshold =1; %The otsu method used in the tracker_diffusion program sometimes estimates the
%threshold as too high or too low. You may need to adjust some multiplier to move the threshold to something
better.
initial=[85  ]; %First frame for the set of 10 frames.
smallf=0.035; %This is the minimum end of the length scale for the filters.
%If whitedomains=1, increasing smallf will increase the detail on the frame
bigf=0.2; %This is the maximum end of the length scale for the filters.
%If whitedomains =1, increasing bigf will increase the detail on the frame
cd(filedirectory)
AAA=imfinfo(file); %imfinfo will read all kinds of information about your movie.
RECT(:,1)=[1   1  AAA.Width  AAA.Height]'; %left column   %top row   %right column   %bottom row
FILT=1; %Strange counter needed for tracker_Diffusion, just leave as 1.
FFF=1; %Again, counting is no longer used, but will crash tracker_diffusion without this
%Below this line is the tracker program you choose to use
tracker_diffusion
%You can have different outputs displayed if you want.
OUTGOOD(FFF)=Nfeg;
OUTX(:,FFF)=meandevx2;
OUTY(:,FFF)=meandevy2;
```

Tracker_diffusion

```
%Tracker_diffusion program for measuring diffusion coefficients.
% SAVES Eccentricity, diameters, slopes
% Has a geometric fix, does not delete overall center of mass movement
%This program was originally written by Pietro Cicuta and modified by Aurelia R. Honerkamp-Smith and Cynthia A. Stanich.
% It calls “sfigure” written by Chris Warth.
%
skip = 0; %When rerunning all of the movies for reprocessing on automatic, sometimes there is no domain to track
%through the ten frame interval. This gives an empty d* folder and needs to skip the code below. This skip must
%be here and will be changed to a 1 below if needed.

cd(filedirectory) %Change directory to the \movies folder
cd .. %goes up one directory to create the d* folder
dos(['md ' char(traj)]) %create d* folder
cd(char(traj)) %open d* folder

thisdatahere=cd;  %defines a variable as the d* folder
cd(filedirectory) %opens the d* folder
datadir=thisdatahere; %some later code written by others uses datadir instead of thisdatahere
%  for the d* folder. % I have defined both to be the d* folder


clear in %in is the array of grey scale image matrix (Changing the tif stack to Matlab variables)
first=initial(FFF); last= first+useframes-1; %defining the first and last frames

%%%%%%%%%%%%%%%%%%%%READ IN MOVIES
if exist('fileisavi')==0, % this means that the file is not avi, and is a tiff stack

  for k=1:useframes, %useframes is given from the param file
  in(:,:,k)=imread(file,initial+k-1); %turning the tif stack to an array of
                              %grey scale images
  end %end reading in movie

end %end the if tif cycle

if exist('fileisavi')==1, % this means that the file is an avi

   MOV = aviread(file,first:last); %This is how you read in an avi
   aviinfo(file) %will print the avi info onto the command line

   for k=1:useframes, %changing the avi info into a Matlab variable
     in(:,:,k)=MOV(k).cdata;

   end %end avi info to Matlab variable

end %end the if avi cycle


numframes=useframes; %numframes is used for useframes in code written by others
```

```
% This defines the area of interest. Most param files set this as the size of the movie. You can choose smaller areas
if desired
minwidth=RECT(1,FFF); %left column
minheight=RECT(2,FFF);  %top row
maxwidth=RECT(3,FFF); %right column
maxheight=RECT(4,FFF);%bottom row

%First go is set in the param files. You can set it to 1 and choose the
%area of interest manually.
if firstgo==1,
   sfigure(50); %sfigure was written by Chris Warth. It allows for use of Matlab even while figures are
%popping up. If you use "figure" instead,the figure is always on top.
   imshow(in(:,:,1)) %show the first image
   hold on
   [rex,rey] = ginput(2); %You will have to make two clicks on the image;
   %one on the top left and one on bottom right corners.
    RECT(:,1)=round([rex(1) rey(1) rex(2)  rey(2)]'); %must be integers
    minwidth=RECT(1,FFF); %left column
   minheight=RECT(2,FFF);  %top row
   maxwidth=RECT(3,FFF); %right column
   maxheight=RECT(4,FFF);%bottom row

   %The plot below will show up on figure (50) as colored lines denoting
   %the area of interest you chose

   plot( [ minwidth  maxwidth   ], [  minheight minheight ], '-r' ) %top side
   plot([ minwidth  maxwidth   ], [maxheight maxheight   ],  '-g' ) %bottom side
   plot([minwidth minwidth   ], [ minheight maxheight    ],  '-b' )  %left side
   plot([ maxwidth  maxwidth   ], [minheight maxheight    ],  '-c' ) %right side
   RECT'   %prints the values for the vector

end %ends manual selection of the area of interest

%I have commented out the lines below because my area of interest is always
%the size of the movie. If you intend to input an area of interest without
%manual clicking, but it is an area smaller than the size of the movie, you
%must uncomment these lines so you can see the outline of the area of
%interest. This is only necessary if you want to check your work, otherwise
%it saves time to remove it.

% if firstgo~=1,
%    sfigure(50);
%    imshow(in(:,:,1))
%    hold on
%    plot( [ minwidth  maxwidth   ], [  minheight minheight ], '-r' ) %top side
%    plot([ minwidth  maxwidth   ], [maxheight maxheight   ],  '-g' ) %bottom side
%    plot([minwidth minwidth   ], [ minheight maxheight    ],  '-b' )  %left side
%    plot([ maxwidth  maxwidth   ], [minheight maxheight    ],  '-c' ) %right side
% end
```

```
for k=1:useframes, %Doing initial analysis of the 10 frame interval

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%          LOAD IMAGES AND SELECT REGION OF INTEREST         % Written by Pietro Cicuta
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   im=in(:,:,k); %k-th frame
   imgray=(im(minheight:maxheight,minwidth:maxwidth,1));  %region of interest
   h = sfigure(1);
   set(h,'units','normalized','position',[0.1 0.1 0.3 0.3]);
   if k==1, imgrayfirst=imgray; end  % keep the first image in memory
  averagelight=mean(mean(imgrayfirst));
   imshow(imgray) %shows the first frame in figure 1
   pause(0.01);


 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 %               NOW CLEAN UP THE IMAGE,                  %
 %          TO GET THE BEST POSSIBLE BLACK&WHITE           % Written by Pietro Cicuta
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   I=imgray;
   %  figure, imshow(I)
   % Set up spatial frequency bandpass filter
   if (k==1)
       [f1,f2] = freqspace(81,'meshgrid');
       Hd = ones(81);
       radius = sqrt(f1.^2 + f2.^2);
       Hd((radius<smallf)|(radius>bigf)) = 0;  % remember in frequency space
       %figure, mesh(f1,f2,Hd)
        win = fspecial('gaussian',81,10);
       win = win ./ max(win(:));  % Make the maximum window value be 1.
       %figure, mesh(win)
       filter = fwind2(Hd,win);
       %figure, freqz2(filter)
       %figure, freqz2( filter(1:2:length(filter(:,1))  ,1:2:length(filter(1,:)) )  )
    end

   %filter with it and threshold
   out = imfilter(I,filter,'replicate');
   out2 = imadjust(out,stretchlim(out),[0,1]);

   %figure, imshow(imadjust(I,stretchlim(I),[0,1]));
%NeedsAureliaGradient filter written by Aurelia R. Honerkamp-Smith and modified for use
%here by Cynthia A. Stanich
% It is an Otsu method done over a grid
   if exist('NeedsAureliaGradient')==1 %%Allows for files done before 9/3/09 to work with this
% tracker program without having a "needsAureliaGradient" value.
      if NeedsAureliaGradient==1 %NeedsAureliaGradient is good for images of a vesicle that
%is next to a bright vesicle and so has a gradient of illumination across it.
         eye=double(out2); %change the values of filtered image to double
         H = fspecial('disk',100);
         J = imfilter(out2, H, 'replicate');
```

```
        J=double(J);
        %figure(100)
        % imshow(out2, []);
        % figure(200)
        % imshow(out2, [])
        % figure(300)
        % imshow(eye-J, []);
        FI = eye - J; %excess brightness
        addittoFI=min(min(FI));
        out2=FI-addittoFI; %remove excess brightness
      end %end NeedsAurliaGradient = 1
    end

    %figure, imshow(out2);
    %The line below sets the threshold value for black and white choices
    thresholdvalue = threshold*graythresh(out2)*(max(max(out2))); %threshold is set in the param files
    if WHITEDOMAINS==0, %set in the param file, if 0: dark domains
       bw2 = (out2 < thresholdvalue);  % this  processes images that have dark domains
    end

    if WHITEDOMAINS==1, %set in the param file, if 1: bright domains
       bw2 = (out2 > thresholdvalue);  % this  processes images that have bright domains
    end

%We know the domains are uniform, but sometimes light differences across the domains cause the domain to
%appear to have domains inside. The line below fills in the perimeter of the domain to ensure the domain is
% uniform. However, you should comment the line below out if your domains are not uniform.
    bw2 = imfill(bw2, 'holes');
%%%Commented out for speed
%     sfigure(2);
%     imshow(bw2)
%%%%%%DO Geometric Fix here%%%%%%%%%
R=load('R.dat'); %This is measured beforehand by track_vesicle
bw3=geometrical_correction(bw2,R);
[bw4,num]=bwlabel(bw3);
bw2 = bw4;

    sfigure(3); %This shows the goodness of the thresholding code

    pat0(:,:,2)=0.99*double(imgray)/65536;
    %pat0(:,:,2)=double(imgray)/255;
    pat0(:,:,3)=0.99*double(imgray)/65536;%double(imgray)/255;
    %pat0(:,:,3)=double(imgray)/255;
    pat0(:,:,1)=0.99*double(imgray)/65536;%double(imgray)/255;
    %pat0(:,:,1)=double(imgray)/255;
    pat=pat0;
    pat(:,:,2)=1;
    dbw = bw2;
    pat(:,:,3)=1-0.5*(dbw);
    pat(:,:,1)=1-0.5*(dbw);
    merged = immerge( pat0 , pat, 0.5);
    imshow(merged)
    hold on
```

```
if (k==1) %I want to click the processed image
   clickthisone=merged;
end


 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 %                NOW LABEL EACH FEATURE                  % Originally written by Pietro Cicuta and
 %modified by Cynthia A. Stanich
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   %label connected regions, find mean coordinates of each region
   fprintf('frame %d\n', k);
   ima=bw2;
   [cim, num] = bwlabel(ima);  % label each connected region
 %Defile the measurements desired:
   stats = regionprops(cim,'Centroid','Eccentricity','EquivDiameter','Area','Perimeter');
   %%%%Change the line below if you are looking at circular domains.
   eccthreshold = 1; %This is useful to cause the program to ignore merging domains.
   cc = 1; %set counter to 1
   xm=[]; ym=[]; siz=[]; rad=[]; memo=[]; ecc=[]; %create empty matricies
   %loop over the boundaries
   for c = 1:num

      metric = stats(c).Eccentricity;
      centroid = stats(c).Centroid;
      totalarea=stats(c).Area;
      perimeter=stats(c).Perimeter;
      diameter  = stats(c).EquivDiameter;
      circumradius=perimeter*(diameter/2);
      topdoublearea=totalarea*2*1.15;
      bottomdoublearea=totalarea*2*0.85;

      plot(centroid(1),centroid(2),'xr')
      %ignore particles at edges of image and non-circular ones and small ones (prob. dirt)
      if (diameter > 4)&(diameter < maxdiam) %very small
       if (circumradius > bottomdoublearea)&(circumradius < topdoublearea) %better measurement than
%setting an eccentricity threshold. (Cynthia likes this more)
         if metric < eccthreshold
centerx=centroid(2);
centery=centroid(1);
distancex=centerx+(diameter*(3/5));
distancey=centery+(diameter*(3/5));
ifneartopleftcornerx=centerx-(diameter*(3/5));
ifneartopleftcornery=centery-(diameter*(3/5));
ifnearbottomrightcornerx=centerx+(diameter*(3/5));
ifnearbottomrightcornery=centery+(diameter*(3/5));
if (ifneartopleftcornerx >= 1)&(ifneartopleftcornery >= 1) %This ignores domains on the top and left edge of the
% image.
   if (ifnearbottomrightcornerx <= maxwidth)&(ifnearbottomrightcornery <=maxheight)
      %Ignores domains on the bottom and right edge of the image.
            xm(cc) = centroid(2);
            ym(cc) = centroid(1);
            siz(cc)= stats(c).Area;
```

```
            rad(cc)= diameter;
            memo(cc)= c;
            ecc (cc) = metric;
            cc = cc + 1;
             plot(centroid(1),centroid(2),'xg') %shows domains that "pass" inspection
          end
         end
        end
         end
        end
    end

    if TEST==1,
       xxx  % uncomment this xxx to cause the program to stop. Use for
       % finding the best region of interest and threshold value on the first frame
    end
     pause(0.01) %comment out for speed
    %close(figure(3))
    %for each image, generate a matrix with info on the features
if size(xm) ~=0  %This could be zero if there are no measurable domains in the frame
    xm=xm'; ym=ym'; siz=siz'; rad=rad'; memo=memo'; ecc=ecc';
%This creates information for each domain in each frame
    eval([ 'FEA' num2str(k) '(:,1)' ' = xm ;']);
    eval([ 'FEA' num2str(k) '(:,2)' ' = ym ;']);
    eval([ 'FEA' num2str(k) '(:,3)' ' = siz ;']);
    eval([ 'FEA' num2str(k) '(:,4)' ' = rad ;']);
    eval([ 'FEA' num2str(k) '(:,5)' ' = memo ;']);
    eval([ 'FEA' num2str(k) '(:,6)' ' = ecc ;']);
    % cd(thisdatahere)

    if k==1,  %save the list of object radii obtained from the first image of each time series
       save([datadir filesep char(traj(1)) 'radius' char(num2str(FFF)) '.dat'], 'rad', '-ascii', '-tabs', '-double');
       save([datadir filesep char(traj(1)) 'eccentricity' char(num2str(FFF)) '.dat'], 'ecc', '-ascii', '-tabs', '-double');
    end
    clear siz; clear xm; clear ym; clear rad; clear memo; clear ecc;
    %save the labelled matrix cim, in which the numbers correspond to the numbers in memo(cc)
    save([datadir filesep char(traj) '_' file '_temp_labelled_' num2str(k) '_' char(num2str(FFF)) '.dat'],'cim','-ascii','-
double')

  onesinbw2=nonzeros(bw2);
  [l,w]=size(bw2);
  totalsizebw2=l*w;
  allones=length(onesinbw2);
  areafraction(k)=allones/totalsizebw2;  %Calculates area fraction of each frame
else
skip=1;
end  % this end finishes the first cycle through the time series (k)
end
%we now have all the important info summarized in the vectors vxm, vym,etc...  positions (cc) along these vectors
%refer to the feature of numbers memo(cc) in the  file saved from the matrix cim.
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 %   NOW MATCH UP CORRESPONDING FEATURES IN TIME SERIES    % Written by Pietro Cicuta
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%take the list of features from successive pairs of images and match them up
if skip ~=1
for k=1:useframes-1;

   fprintf('frame %d \n', k);
   safe=0; like=0;
   ini=   eval([ 'FEA' num2str(k)  ]);  %check syntax
   fin=   eval([ 'FEA' num2str(k+1)]);
    roughXM(k)=mean(ini(:,1));
   roughYM(k)=mean(ini(:,2));
   roughXMf(k)=mean(fin(:,1));
   roughYMf(k)=mean(fin(:,2));
  % roughDX(k)=roughXM(k)-roughXMf(k); roughDY(k)=roughYM(k)-roughYMf(k);
   roughDX(k)=0; roughDY(k)=0;
   %roughDX(k)=inputgrossDX; roughDY(k)=inputgrossDY;

   distmatrix=zeros(size(ini,1),size(fin,1));
   transmatrix=zeros(size(ini,1),size(fin,1));

   fprintf('size=%d\n', size(ini,1));

   %build the distance matrix
   for ii=1:size(ini,1),
      dist=sqrt(     ( ini(ii,1)-fin(:,1)-roughDX(k) ).^2   +  ( ini(ii,2)-fin(:,2)-roughDY(k) ).^2         );
      distmatrix(ii,:)=dist';
    end
    for ii=1:size(ini,1),   %cycle down the rows of the matrix
      [val pos]=  min(  distmatrix(ii,:)   );
      [val2 pos2] = min(  distmatrix(:,pos)   );
          if (val<cut)&(pos2==ii),    %condition of forwards/backwards identity
            transmatrix(ii,pos)=1; distmatrix(ii,pos)=1000; safe=safe+1; %found safe matching
         end
   end

   for ii=1:size(ini,1),   %cycle down the rows of the matrix
      [val pos]=  min(  distmatrix(ii,:)   );
      [val2 pos2] = min(  distmatrix(:,pos)   );
      if (val<cut),    %condition of forwards/backwards identity
         transmatrix(ii,pos)=2;  like=like+1;%found likely matching
      end
   end

   fprintf('safe=%d      like=%d\n', safe, like);

   %now save to file the coordinates of the 1s and 2s
   [un1 un2]=find(transmatrix==1); [du1 du2]=find(transmatrix==2);
   if isempty(un1)~=1,
      uni(:,1)=un1; uni(:,2)=un2;
      else  uni=[0 0];
   end
   if isempty(du1)~=1,
      dui(:,1)=du1; dui(:,2)=du2;
      else  dui=[0 0];
```

```
    end
     eval([ 'UNI' num2str(k)  ' = uni ;']);
    eval([ 'DUI' num2str(k)  ' = uni ;']);
      %tran(1)=trans2(k+1);   tran(2)=trans1(k+1);
    %eval([ 'TRA' num2str(k)  ' = uni ;']);
    clear uni; clear dui;
 end
end

  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  %   NOW TRACK THROUGH THE SEQUENCE TO PRODUCE TRAJECTORY DATA   %
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
tottrans=zeros(2,useframes);
 %attempt to follow through the particles from the first image,
%making a trajectory matrix
if skip ~=1
Nfe= length(eval([ 'FEA' num2str(1) '(:,1)']))
xcord=zeros(  useframes,   Nfe        );  %rows are increasing time, columns are each feature
ycord=xcord;
nfet=zeros(  useframes,   Nfe        );

 for ff=1:Nfe,
    k=1;
    xcord(1,ff)=eval([ 'FEA' num2str(k) '(ff,1)']);
    ycord(1,ff)=eval([ 'FEA' num2str(k) '(ff,2)']);
    nfet(1,ff)=ff;
end

for ff=1:Nfe, %fixes feature in initial image, to track where it goes
    for k=1:useframes-1,
        temfea=nfet(k,ff); %this is the feature in the start image of each couple
         if temfea~=0,
            uui=eval(   [ 'UNI' num2str(k) '(:,1)']              );
            uuu=find(uui==temfea);
            ddi=eval(   [ 'DUI' num2str(k) '(:,1)' ]             );
            ddd=find(ddi==temfea);
            if isempty(ddd)~=1,
                ddf=eval(   [ 'DUI' num2str(k) '(ddd,2)']              );
               xcord(k+1,ff)=eval([ 'FEA' num2str(k+1) '(ddf,1)'])-tottrans(1,k+1);
               ycord(k+1,ff)=eval([ 'FEA' num2str(k+1) '(ddf,2)'])-tottrans(2,k+1);
               nfet(k+1,ff)= ddf;
             end
            if isempty(uuu)~=1,
               uuf=eval(   [ 'UNI' num2str(k) '(uuu,2)']              );
               xcord(k+1,ff)=eval([ 'FEA' num2str(k+1) '(uuf,1)'])-tottrans(1,k+1);
               ycord(k+1,ff)=eval([ 'FEA' num2str(k+1) '(uuf,2)'])-tottrans(2,k+1);
               nfet(k+1,ff)= uuf;
             end
            if (isempty(uuu)==1)&(isempty(ddd)==1),   nfet(k+1,ff)= 0; end  %added 19 nov 04
        end  % ends if temfea~=0,
   end % ends k=1:useframes-1,
end % ends cycle on features
```

```
 %now for all the objects in first frame, save the list of object radii, for
%reference. These are saved in FEA 4th column.

listofradiuses=[];
listofeccen=[];
for ff=1:Nfe,
listofradiuses=[listofradiuses FEA1(ff,4)];
listofeccen = [listofeccen FEA1(ff,6)];
end
save([datadir filesep char(traj) 'set' char(num2str(FFF)) 'listofradiuses' '.dat'], 'listofradiuses', '-ascii', '-tabs', '-
double');
save([datadir filesep char(traj) 'set' char(num2str(FFF)) 'xcord' '.dat'], 'xcord', '-ascii', '-tabs', '-double');
save([datadir filesep char(traj) 'set' char(num2str(FFF)) 'ycord' '.dat'], 'ycord', '-ascii', '-tabs', '-double');


 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 %   MAKE SOME FIGURES TO SEE IF TRACKING WAS OK    % Written by Pietro Cicuta
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

figure(10)
imshow(imgrayfirst); hold on;
%plot the successful trajectories
for ff=1:Nfe,
   if nfet(useframes,ff)~=0
      plot(  ycord(:,ff), xcord(:,ff), '-g', 'linewidth',0.5 );
      hold on
   end
    if nfet(useframes,ff)==0
      plot(  ycord(1,ff), xcord(1,ff), 'or', 'linewidth',0.5, 'markersize',2  );
      hold on
   end
 end

plot(roughYM,roughXM,'-xy')
%plot(roughYM-roughDY,roughXM-roughDX,'-xw')

 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 %       NOW TIDY UP DATA TO PRESENT TRACKING RESULTS          %
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Nfe=eval('size(nfet(1,:))'); Nfe=Nfe(2);
indi=1; goodx=[]; goody=[]; goodf=[];

%make new lists with only the successful ones, for later ease of analysis
for ff=1:Nfe,
    if nfet(useframes,ff)~=0
      goodx(:,indi)=xcord(:,ff); goody(:,indi)=ycord(:,ff);    goodf(:,indi)=nfet(:,ff);
      goodrad(indi) = listofradiuses(ff);
      indi=indi+1;
    end
end
[sizegoodf,sizegoodff]=size(goodf);
if sizegoodf ~= 0
```

```
Nfeg=eval('size(goodf(1,:))'); Nfeg=Nfeg(2);   %this is the number of successful trajectories
meanxtime=mean(goodx,2);  meanytime=mean(goody,2); % these are the mean x and y of all the features at each
time
meanxtime0=meanxtime(1);  meanytime0=meanytime(1);
newgoodx=[];   newgoody=[];
  for tt=1:useframes,
    if (Nfe == 1)
      newgoodx(tt,:) = goodx(tt,:); newgoody(tt,:) = goody(tt,:);
    else
    newgoodx(tt,:)= goodx(tt,:);%-meanxtime(tt)+meanxtime0;%Comment out after goodx(tt,:)
%to stop deleting collective center of mass movement
    newgoody(tt,:)= goody(tt,:);%-meanytime(tt)+meanytime0;%Comment out after goodx(tt,:)
%to stop deleting collective center of mass movement
    end
  end

%sfigure(20);
%imshow(imgrayfirst); hold on;
%imshow(clickthisone); hold on;
%sizeimgrayfirst=size(imgrayfirst);
sizeimgrayfirst=size(clickthisone);
for ff=1:Nfeg,
  plot(  newgoody(:,ff), newgoodx(:,ff), '-m', 'linewidth',0.5 );
end
for ff=1:Nfe,
  if nfet(useframes,ff)==0
    plot(  ycord(1,ff), xcord(1,ff), 'oy', 'linewidth',0.5, 'markersize',2  );
    hold on
  end
end
 print([datadir filesep file 'temp' char(num2str(FFF)) '.tif'],'-dtiff')

%make the the mean square displacements for each object:
devx2=[]; devy2=[];
for tt=1:useframes,
  devx2(tt,:)= (newgoodx(tt,:)-newgoodx(1,:)).^2;   devy2(tt,:)= (newgoody(tt,:)-newgoody(1,:)).^2;
end
meandevx2= mean(devx2,2); meandevy2= mean(devy2,2);  %these are the averages over all
%features

 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 %  NOW SELECT THE GOOD DOMAINS TO BE ANALYZED
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 if using_previous_choice_of_domains~=1, % graphical input for choice of domains
  badclick=0;
sfigure(20);
imshow(clickthisone); hold on;
for ff=1:Nfe,
   if nfet(useframes,ff)~=0
    plot(  ycord(:,ff), xcord(:,ff), '-m', 'linewidth',0.5 );
    hold on
   end
```

```
   if nfet(useframes,ff)==0
      plot(  ycord(1,ff), xcord(1,ff), 'or', 'linewidth',0.5, 'markersize',2  );
      hold on
   end
end
   disp('Choose some dots!');
   [k_x, k_y] = myginput; % these are coordinates of points inside the features that we want to keep.
   disp('Thank you, that was delicious.');

   save([datadir filesep char(traj) 'k_x'  '.dat'], 'k_x', '-ascii', '-tabs', '-double');
   save([datadir filesep char(traj) 'k_y'  '.dat'], 'k_y', '-ascii', '-tabs', '-double');
 else % load the info from saved file
   k_x=load([datadir filesep char(traj) 'k_x'  '.dat'], 'k_x');
   k_y=load([datadir filesep char(traj) 'k_y'  '.dat'], 'k_x');

%%%%This code below is an attempt to map existing dots onto the geometric fixed image. This works most of the
%time. Figure 20 at the end will let you know if it doesn't.
d1k_x=int16(k_x);
d1k_y=int16(k_y);
matrixsize=max(size(clickthisone));
olddotmatrix=zeros(RECT(3));
for i=1:length(d1k_x);
%for j = 1:length(d1k_y)
colindex=d1k_x(i);
rowindex=d1k_y(i);
olddotmatrix(rowindex,colindex)=1;
%end
end
% figure (1)
% imshow(olddotmatrix)
% title('Old Dot Placement')

%R=load('C:\Program Files (x86)\MATLAB\R2007a Student\work\diffusion\Comp1\Water\9-4-
%09\v1\q2\movies\R.dat')
newdotmatrix=geometrical_correction(olddotmatrix,R);
% figure (2)
% imshow(newdotmatrix)
% title('New Dot Placement')
threshclick = 0.001;

bwdotmatrix=im2bw(newdotmatrix,threshclick);
[clickcim, clicknum] = bwlabel(bwdotmatrix);
clickstats = regionprops(clickcim,'Centroid');
 for c = 1:clicknum
clickcent(c,:) = clickstats(c).Centroid;
 end

 k_x = round(clickcent(:,1));
 k_y = round(clickcent(:,2));
% newdotmatrix=int16(newdotmatrix);
% [row,column]=find(newdotmatrix == 1);%column are the x axis, row is the y
% %row=matrixsize-row;
% k_x=column;
```

```
% k_y=row;
numclicks=length(k_x);
if Nfeg ~= numclicks,
   badclick=1;
else
   badclick=0;
  % numclicks=Nfeg
end
%%%%%%%%%%%end remaping
end % ending if USINGPREVIOUS~=1,
 numclicks=length(k_x);
k_x=round(k_x);  k_y=round(k_y);
  % AIM: figure out which is the feature in "cim" corresponding to each of the
  % mouse clicks on sfigure(20). Remember there is a cim matrix for each
  % timestep, and the matching feature has a different numner in cim at
  % each time !

  clear num
for ff=1:numclicks,
   dist=(   (k_y(ff)-newgoodx(1,:)).^2 + (k_x(ff)-newgoody(1,:)).^2  );
   [mi po]=min(dist);
   num(ff)=po; % this contains the index in newgoodx and newgoody of the clicked feature ff
end
% now go back to the information that is in matrix "cim"
clear  num_c
% It is better to use mouse click positions in cim only on first frame, as domains might
%move, (especially the small ones).
%For the geometrically fixed images, some clicks are done on domains that touch the edge.
k=1;
cim = load([datadir filesep char(traj) '_' file '_temp_labelled_' num2str(k) '_' char(num2str(FFF)) '.dat']);
for ff=1:numclicks,
   num_c(k,ff)=cim( k_y(ff), k_x(ff) ); % this is now the number in the matrix "cim" for 1st frame
%(will not be the same for all k)
end
% so better strategy after 1st frame to use the center of mass of  the domain

   for k=2:numframes, %cycle through time again
      cim=   load([datadir filesep char(traj) '_' file '_temp_labelled_' num2str(k) '_' char(num2str(FFF)) '.dat']);
      for ff=1:numclicks,
         num_c(k,ff)=cim( round(goodx(k, num(ff))), round(goody(k, num(ff))) ); % this is now the
%number in the matrix "cim" for 1st frame (will not be the same for all k)
      end
   end % end of cycle through time
% num(k,ff)=po; % this contains the index in newgoodx and newgoody of the clicked feature ff
%(should be the same for all k)

  %check we got the proper ones
sfigure(20);
imshow(clickthisone); hold on;

for k=1:numframes,
   for ff=1:numclicks,
      sfigure(20); hold on
```

```
      plot(newgoody(k,num(ff)), newgoodx(k,num(ff)), 'oc', 'linewidth',0.5, 'markersize',3  );
      hold on
   end
end

clear cell_row cell_col cm_row cm_col areafea

for ff=1:numclicks,
   for k=1:numframes,
      % cd(thisdatahere)
      cim=   load([datadir filesep char(traj) '_' file '_temp_labelled_' num2str(k) '_' char(num2str(FFF)) '.dat']);
      % cd(here)
      clear bw_single BW1_p BW2_p xee yee
      bw_single = (cim == num_c(k,ff));
      BW1_p = bwmorph(bw_single,'fill');
      BW2_p = bwperim(BW1_p);
      areafea(ff,k)=sum(sum(bw_single));
      double_single = double(bw_single);
      singlestat = regionprops(double_single,'Perimeter','Eccentricity');
      perim(ff,k) = singlestat.Perimeter;
      eccen(ff,k) = singlestat.Eccentricity;
      [xee yee]=find(BW2_p == 1);
      cell_row(ff,k) = {[xee]};
      cell_col(ff,k) = {[yee]};
      cm_row(ff,k)=mean([xee]);
      cm_col(ff,k)=mean([yee]);
%We no longer subtract the average center of mass movements.
      cell_row{ff,k}= cell_row{ff,k} + cm_row(ff,1)  - cm_row(ff,k) ; %taking away center of mass
%motion for the figure
      cell_col{ff,k}= cell_col{ff,k} + cm_col(ff,1)  - cm_col(ff,k) ;
%Commented out to try to fix out of bounds problem
% can put back to see what happens
%       sfigure(20);
%       plot(cell_col{ff,k}, cell_row{ff,k}, 'oy', 'markersize', 1)
%       hold on
   end
end
if badclick == 1
   %My first thought was to find the change in the areas, but this doesn't
   %work very well if the mistaken domains have similar sizes. Instead I
   %am trying the difference in x direction placement
         %A=mean(areafea,2);
         %B=abs(A-areafea(:,1));
         diffdomains=clicknum - Nfeg;
         A = abs(cm_row(:,1)-cm_row(:,2));
         for z=1:diffdomains  %need to remove the bad domains from the list
         C=find(A==max(A));
         A(C,:)=[];
         cm_row(C,:)=[];
         cm_col(C,:)=[];
      areafea(C,:)=[];
      perim(C,:) = [];
      eccen(C,:) = [];
```

```
       cell_row(C,:)= [];
       cell_col(C,:)= [] ;
          end
end

figure (20)
delete(get(0,'CurrentFigure'))
figure(20)
imshow(clickthisone);, hold on
[endff, frames]= size(cell_col);
for ff=1:endff,
   for k=1:numframes,
plot(cell_col{ff,k}, cell_row{ff,k}, 'oy', 'markersize', 1)
   end
end

  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  %  NOW ANALYZE DIFFUSION IN THE SELECTED DOMAINS           %
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 %I normally do not use color and lines to look at figure 30, but uncomment the next two lines if desired (Note: if
there are too many domains, this will crash your program when it reaches the end.
 %color = {'or' 'oy' 'og' 'oc' 'ob' 'ok' 'om' 'vr' 'vy' 'vg' 'vc' 'vb' 'vk' 'vm' 'or' 'oy' 'og' 'oc' 'ob' 'ok'};
 %line =  {'-r' '-y' '-g' '-c' '-b' '-k' '-m' ':r' ':y' ':g' ':c' ':b' ':k' ':m' '-r' '-y' '-g' '-c' '-b' '-k' '-m'};

figure(30)
hold on
x = 0:k-1;
for ff = 1:numclicks,
   fnum = num(ff);

%     rmsd = sqrt(devx2(:,fnum) + devy2(:,fnum));
%     msq(ff,:) = rmsd;

%From Cicuta, Keller and Veatch paper, "the average of vertical and horizontal mean
%square displacements (MSD) is linear with time t and fit to <x^2> = 2nDr(t)
   msd = devx2(:,fnum) + devy2(:,fnum);
%msd=uint16(msd);
   dia(ff) = goodrad(fnum);
   %PP=polyfit([1:k],msd',1);
  % P=PP(1);
   %[P,intercept,aerror,corcoe]=curveFitting([1:k],msd);
   [P] = lsqcurvefit('Alinear', 1, [1:k], msd'); %fitting time
  % slope(ff) = P(1);
  slope(ff)=P/4; %n = 2
%I normally do not use color and lines to look at figure 30, but uncomment the next two lines if desired (Note: if
there are too many domains, this will crash your program when it reaches the end.
%     plot(x, rmsd, char(color(ff)))
%     plot(x, P*x, char(line(ff)))
%If you uncommented the two lines above, you will want to comment the two lines below.
    plot(x, msd)
    plot(x, P*x)

end
```

```
xlabel('frames')
ylabel('msd (pixels squared)')

figure(35)
plot(dia, slope, 'ob')
xlabel('diameter (pixels)')
ylabel('slope of msd fit (pixels squared per frame')

%Now save the diameter and slope information to read in later
saveslopes = vertcat(dia, slope);
save([char(traj) 'slopes' '.dat'], 'saveslopes', '-ascii', '-tabs', '-double');
save([char(traj) 'areafraction' '.dat'], 'areafraction', '-ascii', '-tabs', '-double');
save([char(traj) 'AREA' '.dat'], 'areafea', '-ascii', '-tabs', '-double');
times=[initial,useframes];
save([char(traj) 'times' '.dat'],'times','-ascii', '-tabs', '-double');
save([char(traj) 'Perimeter' '.dat'], 'perim', '-ascii', '-tabs', '-double');
save([char(traj) 'Eccentricity' '.dat'], 'eccen', '-ascii', '-tabs', '-double');
end
end
%The lines below are for old param files so they do not crash. It simply
%resets values of cleared variables for looping perposes.
Nfeg=1;
meandevx2=1;
meandevy2=1;
char(traj) %I like to print the traj folder upon completion so if I get a crash,
%I know where to start processing.
```

Run_all_difco

```
%This is a typical run_all file to analyze data from tracker_diffusion.
%Written by Cynthia A. Stanich
clear all
close all
 fileplace='C:\Program Files (x86)\MATLAB\R2007a student\work\diffusion\halfandhalf\water\9-13-10\v1\movies';
cd(fileplace)

%This part loads in all the data
files = dir('d*slopes.dat')  %A * is a space filler and this will find any file with d and slopes.dat in the name.
filenames = char(files.name);

%Read in the radiuses (in pixels, column 1) and slopes (in pixels per
%frame, column 2) of domains from a movie
s = size(filenames); %This is how many files there are and how many characters in the names.
s=s(1); %This is how many files there are.

for i = 1:s %I use a loop here to keep the files in order. This is not so important for the
   %diffusion constant calculation, but it is important for anything time
   %dependent.
   j = char(num2str(i))
file = strcat('d',j,'slopes.dat');
%    file = filenames(i, :);
   thisdata = load(file)';
if i==1 %This starts the alldata string. You can also start it with alldata=[] without the if/then.
   alldata=thisdata;
else
   alldata = vertcat(alldata, thisdata);
 end
end
[row,column]=size(alldata);
%now load up later movies with 1s/frame data

diameter = alldata(:,1)*.18; %this should convert diameter to microns.
slope = [alldata(:,2) * (.18)^2 * (2)]; %this should convert to microns squared per second

figure(1), hold on
 plot(diameter/2, slope,'*')% plotskies(m,:))
xlabel('domain radius in microns')
ylabel('D in microns squared per second')
%legend(compositions)

%now bin in bins of one micron (This is useful for diameter, but to plot it
%is better to use bins/2. This will convert to radius and allow for half micron binning.

nbins = 30; %Should be the same as your max diameter
for i = 1:nbins,
   lowint = find(diameter <= i);
   highint = find(diameter >= (i-1));
   int = intersect(lowint, highint);
   slbin(i) = mean(slope(int));
```

```
   devbin(i) = std(slope(int));
   clear lowint highint int
end

%Make a figure with errorbars
figure(3), hold on
errorbar((1:nbins)/2, slbin, devbin,'*')%, plotskies(m,:))
xlabel('domain radius in microns')
ylabel('D in microns squared per second')
%legend(compositions);

%But you can't get a nice log-log plot with the errorbar function above. So do it the way below:
figure (4), hold on
plot((1:nbins)/2,slbin,'*')%(m,:),plotskies(m,:))

for x=1:nbins;
    plot([x/2, x/2], [slbin(x)-(devbin(x)/2), slbin(x) + (devbin(x)/2)])%, colorskies(m,:) )
end
%end

%Now plot the expected values from HPW's Diffusion Constant equation. There
%are no free parameters, use values from the actual data.
%EDIT: Ediffco was changed to the new equation by de Koker 1996 and Seki 2011 to assume the inclusion is fluid
with the same viscosity as the membrane.
T=38+273.15; %Use the temperature from the quench
r=(1:0.1:20)*10^(-6);
eida=.0007;
D=Ediffco(T,eida,r);
r=r*10^6;
D=D*10^12;
plot(r,D)

%This is for getting rid of imaginary numbers from the diffusion constants.
%Mostly this is for if the radii and slopes are binned after taking the
%log. However, I use this as a final filtering step.
xva=find(~isnan(slbin));
for a=1:length(xva);
   newslbin(a)=slbin(xva(a));
   newdevbin(a)=devbin(xva(a));
end

%Plot final plot for log-log manipulation and visualization
figure (42), hold on
plot(xva/2,newslbin,'*')
for x=1:length(xva);
    plot([xva(x)/2, xva(x)/2], [newslbin(x)-(newdevbin(x)/2), newslbin(x) + (newdevbin(x)/2)]) end
plot(r,D)

%Save data with lines below if you plan to use this data for figures for
%papers using another run_all file later.
% save(['allradii.dat'], 'radius', '-ascii', '-tabs', '-double');
% save(['alldifco.dat'], 'slope','-ascii','-tabs','-double'
```

Ediffco

```
%Written by Cynthia A. Stanich
function D=Ediffco(T,eida,r)
%D=(1.3807*10^(-23)).*T./(16.*eida.*r); %HPW
D=(2.*1.3807*10^(-23)).*T./(3.*(pi^2).*eida.*r); %de Koker 1996, Seki 2011
end
%This results in D in m^2/s
%Run for D with these values. THEN you multiply r by 10^6 to put back into
%microns and multiply D by 10^12. Then plot.
```

PetrovSchwille

```
%Written by Cynthia A. Stanich
function [Dum] = PetrovSchwille(a,T,eida,mu1) %calculates the diffusion constant of radii
%according to Petrov & Schwille
%a = set of domain radii in microns
%T = temperature of quench in Celcius
%eida = viscosity of the membrane
%in P&S e=a/l and is called the reduced radius.
if nargin <4
   mu1 = 0.0006535
end
if nargin < 3
   eida = 4*10^-9; %units of Pa s m
end
if nargin < 2
   T = 40.29; %C
end
if nargin < 1 %This is set up to work with Cynthia's data on her laptop. If you would like to
   %have an auto-loading data file, you will need to create one.
nbins=45; %This is to create the plot that I want to fit with the equation below. These are the x
%values
slbin=load('slbin.dat'); %This will load the y values of the plot you are fitting with the equation below.
figure (12)
plot(2:nbins,slbin(2:nbins),'o')
title('Diffusion Constants versus radii of domains')
xlabel('Radii (microns)')
ylabel('Diffusion constants (micron^2 / s)')
end
% %%%
% %%%%%%%%%%%%%%%%%%
% %This loads the actual radii of the domains so that you have actual values
% %of data to work with. These values are in microns
% allradii=load('radii.dat');
%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%
%%%Below here is anything you might want to change. You should check units
%%%of the values I have below and make sure they all make sense
k = 1.3806503*10^(-23); %(m^2kg/s^2K)
T = 273.15+T; %K
avagadro = 6.0221415*10^(23); %molecules/mole
%%%%%%%%%%%%%%%%%%
%%%%eida is the viscosity of the membrane. It is the number you should try
%%%%changing once you make sure all the units make sense
iis=0.0008:0.0001:0.1;
colorset=varycolor(length(iis));
m=1;
c1 = 0.73761; %Unitless constant
c2 = 0.52119; %Unitless constant
b1 = 2.74819; %Unitless constant
b2 = 0.61465; %Unitless constant
gamma = 0.577215; %Unitless euler's constant
```

```
mu2 = mu1;
a=a*10^(-6);% changes radii from microns to meters
%epsilon will change as you change eida. Please make sure the units
%actually cancel. Where is the thickness of membrane? What does that do to
%units?????
epsilon = (a.*(mu1+mu2))./(eida); %unitless  epsilon is also a/L, where L = thickness of membrane * viscosity of membrane / (2 * viscosity of water).
%%%%%%%%%%%%
%%%%%%%%%%%Don't change the stuff in the next few lines%%%%%%%%%%%%%%%%%%%%%
part1 = log(2./epsilon)-gamma+(4.*epsilon./pi)-((epsilon.^2)/2).*log(2./epsilon); %unitless
part2 = 1-(((epsilon.^3)/pi).*log(2./epsilon))+((c1*epsilon.^b1)./(1+c2*epsilon.^b2)); %unitless
D=(k*T/(4*pi*eida)).*part1./part2; %m^2/s
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%
aum=a*10^6; %in microns
Dum=D*(10^6)*(10^6); %in microns^2/s
%This figure will plot the values you created. Does it look right?
figure (103), hold on
plot(aum,Dum,'.k')
title('Diffusion Constants versus radii of domains')
xlabel('Radii (microns)')
ylabel('Diffusion constants (micron^2 / s)')
```

Param_growth

```
%Written by Cynthia A. Stanich
%%%GROWTH%%%
%Parameter file for using with all versions of the growthexponent measurements.
%Be sure to comment out the correct lines.
clear all
close all
firstgo=0; %Set to 1 if you want to choose RECT
warning off all
WHITEDOMAINS=0; % set =1 if domains are bright on dark background in original avi file, set =0 otherwise
hd='C:\Program Files (x86)\MATLAB\R2007a Student\work\diffusion\Comp2\Water\2-1-
10\water_sucrose\v1\q1\movies';
cd(hd)
homedirect='C:\Program Files (x86)\MATLAB\R2007a Student\work'; %This is where the program files are.
filedirectory=[hd];
FILT=1; %counter

%%%growthexponent required lines: YOU MUST COMMENT THIS OUT FOR growthexponent_startanywhere and
%growthexponent_choose
%Below is a loop through all of the movies of a single quench
initial = 1;
filelist=['q1_00.tif';
'q1_01.tif';
'q1_02.tif';
'q1_03.tif';
'q1_04.tif';
'q1_05.tif']; %you can use as many movies at a time as you want with growthexponent.m
TimeStart=[5,4,14;
5,8,16;
5,10,23;
5,12,09;
5,13,59;
5,16,32]; %Hour,Minute,Seconds TIME WHEN THE MOVIE STARTS
traj='q1Grow'; %Make folder for data
%%%%End growthexponent.m required lines

%%%growthexponent_startanywhere required lines: COMMENT THIS OUT FOR growthexponent and for
%growthexponent_choose, you will not be able to do a loop. You should
%only give one value in firstframelist, useframeslist, perfectrun, and fillon
filelist=['v2001.tif']; %For growthexponent_startanywhere you can only use 1 movie.
TimeStart=[1,00,00]; %Hour,Minute,Seconds TIME WHEN THE MOVIE STARTS
traj='v2001Growth'; %Make folder for data, must be named by movie
firstframelist=[1:10:150];%actual movie is 159 frames, 15 runs
useframeslist=[ones(1,14)*20,159-max(firstframelist)];
perfectrun=[zeros(1,15)]; %You can have the code skip parts of movies if it ran successfully before.
fillon=zeros(1,15); %It is possible to turn off fill. This is important for anything not circular.
[row2,column2]=size(firstframelist);

%These lists are necessary for all growthexponent programs.
%If using growthexponent: These are the filters you need for each movie.
%If using growthexponent_startanywhere and growthexponent_choose: These are the filters you will use
```

```
%for interval designated by the initial and final frames.
smallflist=[0.04,0.04,0.04,0.04,0.04,0.04,0.04,0.04,0.035,0.035,0.035,0.035,0.035,0.035,0.04];%,0.04,0.04];
bigflist=[0.17,0.14,0.14,0.13,0.13,0.12,0.10,0.10,0.13,0.12,0.11,0.11,0.10,0.09,0.08];%,0.11,0.10];
thresholdlist=[0.9,0.9,0.9,0.9,0.9,0.9,0.9,0.9,0.85,0.85,0.8,0.8,0.8,0.85,0.85];%,0.8,0.8];
NeedsAureliaGradientList=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];%,0,0];

%%%%%The following lines are necessary if you are using
%%%%%growthexponent_startanywhere. These lines are not necessary to comment out.
doyou= exist (strcat([char(hd) filesep char(traj) filesep char(traj) 'perimetersum' char(filelist) '.dat']));
if (doyou == 2)
 perimetersum=load([char(hd) filesep char(traj) filesep char(traj) 'perimetersum' char(filelist) '.dat']);
 areafraction=load([char(hd) filesep char(traj) filesep char(traj) 'areafraction' char(filelist) '.dat']);
 normalizedperimeter=load([char(hd) filesep char(traj) filesep char(traj) 'normalizedperimeter' char(filelist) '.dat']);
 blackarea=load([char(hd) filesep char(traj) filesep char(traj) 'blackarea' char(filelist) '.dat']);
end
%%%%%%%%%%End growthexponent_startanywhere code

%
dos(['md ' char(traj)])
[row,column]=size(filelist);%Get the number of movies from "row"
for loop=1:row; %This loops over all of the movies (row)
    smallf=smallflist(loop);
   bigf=bigflist(loop);
   threshold=thresholdlist(loop);
   NeedsAureliaGradient=NeedsAureliaGradientList(loop);
file=filelist(loop,:);%calls up the movie
AAA=imfinfo(file);
[little,big]=size(AAA);
final=big;
useframes=final-initial+1;
framelist(loop)=useframes;
cd(homedirect)
growthexponent %Measures all movies in a quench (MULTIPLE MOVIES)
growthexponent_startanywhere %Measures intervals IN ONE MOVIE
growthexponent_choose %For measuring specific domains

times=0.5*(0:useframes-1); %times in seconds (for figures in growthexponent_v4.m)

for www=1:useframes;
   clear (['FEA' num2str(www)])
   clear (['DUI' num2str(www)])
end
clear imgray
clear MOV
clear cim
clear ddi
clear diam
clear distmatrix
clear ecclist
clear f1
clear f2
clear fin
clear goodf
```

```
clear goodx
clear goody
clear histed
clear histme
clear im
clear ima
clear imgrayfirst
clear ini
clear listofradiuses
clear lograd
clear logtime
clear meandiam
clear merged
clear micronmeanradius
clear nfet
clear nome
clear onesinbw2
clear out
clear out2
clear perim
clear radius
clear roughDX
clear roughDY
clear roughXM
clear roughXMf
clear roughYM
clear roughYMF
clear tottrans
clear transmatrix
clear win
clear xcord
clear ycord
close all
end
cd(thisdatahere)

for loop=1:row
 hour=TimeStart(loop,1);
 minute=TimeStart(loop,2);
 second=TimeStart(loop,3);
 sec(1,loop)=second+(60*minute)+(3600*hour);

if (loop==1)
time=[0.5:0.5:framelist(1)/2];%sets initial time
else if (loop~=1)
   nextstart=sec(loop)-sec(1);
   nextend=nextstart+(((framelist(loop))-1)/2);
   sequence=nextstart:0.5:nextend;
   time=[time,sequence];
   end
end
end
save([char(traj) 'TimeList' '.dat'],'time','-ascii','-tabs','-double')
```

```
for loop=1:row; %This loops over all of the movies (row)
file=filelist(loop,:);%calls up the movie
newareafraction=load ([char(traj) 'areafraction' char(file) '.dat']);
newperimetersum=load ([char(traj) 'perimetersum' char(file) '.dat']);
newnormperim=load([char(traj) 'normalizedperimeter' char(file) '.dat']);
newdiams=load([char(traj) 'diam' char(file) '.dat']);
neweccs=load([char(traj) 'eccentricity' char(file) '.dat']);
[rows,columns]=size(neweccs);
for n=1:columns; %Process over each frame of the each movie
   indexlist=find(newdiams(:,n)); %This finds the indicies that are not zeros
   aa=min(indexlist); %Starting (always 1)
   bb=max(indexlist); %Ending (last index before the first zero)
   for k=aa:bb;
   %a(k,n)=sqrt(newareas(k,n)/(pi*sqrt(-(neweccs(k,n)^2-1))));
   %b(k,n)=sqrt((a(k,n))^2-(((a(k,n))^2)*neweccs(k,n)));
   %r(k,n)=sqrt(a(k,n)*b(k,n));
   r(k,n)=newdiams(k,n)/2;
   end
meanr(n)=mean(r(aa:bb,n));
end
clear r
if loop==1
   perimList=newperimetersum;
   normperimList=newnormperim;
   AFList=newareafraction;
   meanrList=meanr;
else AFList=[AFList,newareafraction];
   normperimList=[normperimList,newnormperim];
   perimList=[perimList,newperimetersum];
   meanrList=[meanrList,meanr];
end
clear meanr
end
close all

figure (1)
plot(time,AFList)
title('The fraction of dark to all versus time')
xlabel('Time (seconds)')
ylabel('Fraction of dark to all')

figure (4)
plot(time,perimList)
title('Total Perimeter vs Time')
xlabel('Time')
ylabel('Perimeter (pixels)')

figure (5)
plot(time,normperimList)
title('Normalized Perimeter vs Time')
xlabel('Time')
ylabel('Normalizeed Perimeter (Perimeter/area of movie)')
```

Growth_exponent

```
%%%GROWTH%%%
% You can use this program to measure R on multiple movies.
% Saves area fraction, total area, total perimeter, and time
%LINE 338 - holes
%VERSION 6 - Rewriting the way k is counted. Need a "while" instead of a "for"
%VERSION 5 - Streamlined growthexponent code.
%Saves total perimeter
%Includes geometric fix.
%saves Area and Eccentricity for a better calculation of
%diameter. Ensures that the area chosen is square.
%This program contains parts of tracker_diffusion that were originally
%written by Pietro Cicuta and modified by Cynthia A. Stanich


cd(filedirectory) %makes directory hd
cd(char(traj)) %Changes directory to new directory
thisdatahere=cd; %Tells the code to put all the data in the new directory

%figure(5)%First go (firstgo=1,test=1) this will be blank
set(gcf,'paperunits','centimeters')
set(gcf,'paperposition',[5 5 10 8]) %left bottom  totalwidth totalheight
axes('position',[0.2 0.2 0.6 0.6])   %left bottom  relativewidth relativeheight

cd(filedirectory)%makes directory hd from param*.m
here=cd;
cd(hd)
 %This next part allows for choosing the area to study on figure 50.
if firstgo==1,
sfigure(50)
   imshow(in(:,:,1)) %This part takes the bitdepth values in the in array and plots them on figure 50
   hold on
   [rex,rey] = ginput(2);%graphical input from mouse. This gets 2 points from
   %the current axes and returns the x- and y-coordinates in length 2 vectors rex and rey
   RECT(:,1)=round([rex(1) rey(1) rex(2)  rey(2)]');
  % CIRC(:,1)=round([rex(3) rey(3) rex(4) rey(4)]');
    minwidth=RECT(1,FFF); %left column
   minheight=RECT(2,FFF);  %top row
   maxwidth=RECT(3,FFF); %right column
   maxheight=RECT(4,FFF);%bottom row
   diffies=[maxwidth-minwidth,maxheight-minheight]; %Our measurement requires that the
   %area of interest is square. This forces it to be so.
 square=max(diffies)-min(diffies);
 if diffies(1)>=diffies(2)
   maxheight=maxheight+square;
  else maxwidth=maxwidth+square;
 end

   %%draws the colored square around the vesicle seen on figure 50
   plot( [ minwidth  maxwidth   ], [  minheight minheight ], '-r' ) %top side
   plot([ minwidth  maxwidth   ], [maxheight maxheight   ],  '-g' ) %bottom side
```

```
    plot([minwidth minwidth   ], [ minheight maxheight    ],  '-b' )  %left side
    plot([ maxwidth  maxwidth   ], [minheight maxheight    ],  '-c' ) %right side

        %RECT' %prints the chosen numbers
end %Ends first go

if firstgo~=1, %aka 0
%in this case all the Rect values are known. No need to choose
sizes=[AAA.Width, AAA.Height];
sides=min(sizes);
   RECT(1,:)=[1 1 sides sides];
    minwidth=RECT(1); %left column
    minheight=RECT(2);  %top row
    maxwidth=RECT(3); %right column
    maxheight=RECT(4);%bottom row

 %You can comment this figure out to speed up processing.
   sfigure(50)
    imshow(in(:,:,1))
    hold on
    plot( [ minwidth  maxwidth   ], [  minheight minheight ], '-r' ) %top side
    plot([ minwidth  maxwidth   ], [maxheight maxheight   ],  '-g' ) %bottom side
    plot([minwidth minwidth   ], [ minheight maxheight    ],  '-b' )  %left side
    plot([ maxwidth  maxwidth   ], [minheight maxheight    ],  '-c' ) %right side

end %Ends first go

  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  %          LOAD IMAGES AND SELECT REGION OF INTEREST% Originally written by Pietro Cicuta
  %   %%Modified heavily by Cynthia A. Stanich
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Want to count through k so that each frame set just continues with the next k in the line.
counter=1;
for loop=1:column2; %This loops over all of the framesets (row) and reads in
    %all of the values in the parameter file for each movie.
    smallf=smallflist(loop);
    bigf=bigflist(loop);
    threshold=thresholdlist(loop);
    NeedsAureliaGradient=NeedsAureliaGradientList(loop);
    lastframe=firstframelist(loop)+useframeslist(loop)-1;
    useframes=useframeslist(loop);
    k = firstframelist(loop);
    if perfectrun(loop) == 0
while k <= lastframe
    im=in(:,:,k);
    imgray=(im(minheight:maxheight,minwidth:maxwidth,1));  % select the best part of the frame
clear im
    if k==firstframelist(loop), imgrayfirst=imgray;  end% keep the first image in memory
    averagelight=mean(mean(imgrayfirst));
       h =sfigure(1);
       set(h,'units','normalized','position',[0.1 0.1 0.3 0.3]);
       imshow(imgray)
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                NOW CLEAN UP THE IMAGE,                  %
%          TO GET THE BEST POSSIBLE BLACK&WHITE   % Written by Pietro Cicuta
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    I=imgray;
    % Set up spatial frequency bandpass filter
  if (k==firstframelist(loop))
      [f1,f2] = freqspace(81,'meshgrid');
      Hd = ones(81);
      radius = sqrt(f1.^2 + f2.^2);
      Hd((radius<smallf)|(radius>bigf)) = 0;  % remember in frequency space
      %figure, mesh(f1,f2,Hd)
      win = fspecial('gaussian',81,10);
      win = win ./ max(win(:));  % Make the maximum window value be 1.
      %figure, mesh(win)
       filter = fwind2(Hd,win);
        end
   %filter with it and threshold
   out = imfilter(I,filter,'replicate');
   out2 = imadjust(out,stretchlim(out),[0,1]);
   %figure, imshow(imadjust(I,stretchlim(I),[0,1]));

%NeedsAureliaGradient filter written by Aurelia R. Honerkamp-Smith and modified for use
%here by Cynthia A. Stanich
% It is an Otsu method done over a grid
  if exist('NeedsAureliaGradient')==1 %%Allows for files done before 9/3/09 to work with this tracker program
without having a "needsAureliaGradient" value.
      if NeedsAureliaGradient==1 %Needs Aurelia Gradient is good for images of a vesicle that are next to a bright
vesicle and so have a gradient of illumination across it.
         eye=double(out2); %change the values of filtered image to double
         H = fspecial('disk',100);
         J = imfilter(out2, H, 'replicate');
         J=double(J);
         %figure(100)
         % imshow(out2, []);
         % figure(200)
         % imshow(out2, [])
         % figure(300)
         % imshow(eye-J, []);
         FI = eye - J; %excess brightness
         addittoFI=min(min(FI));
         out2=FI-addittoFI; %remove excess brightness
      end %end NeedsAurliaGradient = 1
   end

   %figure, imshow(out2);
 %The line below sets the threshold value for black and white choices
   thresholdvalue = threshold*graythresh(out2)*(max(max(out2)));
  if WHITEDOMAINS==0, %set in the param file, if 0: dark domains
      bw2 = (out2 < thresholdvalue);  % this  processes images that have dark domains
   end
```

```
   if WHITEDOMAINS==1, %set in the param file, if 1: bright domains
      bw2 = (out2 > thresholdvalue);  % this  processes images that have bright domains
   end
   if fillon(loop) == 1 %We do not know the domains are uniform. The param file
      %sets whether or not you should fill in the holes.
   bw2 = imfill(bw2, 'holes');
   end
   clear FI

%Commented out for speed
%       sfigure(2)
%       imshow(bw2)

clear pat0
clear pat
clear merged
clear dbw
        sfigure(3) %comment out for speed

       pat0(:,:,2)=0.99*double(imgray)/65536;
   %pat0(:,:,2)=double(imgray)/255;
   pat0(:,:,3)=0.99*double(imgray)/65536;%double(imgray)/255;
   %pat0(:,:,3)=double(imgray)/255;
   pat0(:,:,1)=0.99*double(imgray)/65536;%double(imgray)/255;
   %pat0(:,:,1)=double(imgray)/255;
   pat=pat0;

   pat(:,:,2)=1;

       clear imgray
       %aurelia intervention to change bw2 type!
       dbw = double(bw2);
       pat(:,:,3)=1-0.5*(dbw);
       pat(:,:,1)=1-0.5*(dbw);
cd(hd)
      merged =immerge(pat0,pat,0.5);%uses immerge.m(explanation included in that file)
     hold on
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  %               NOW LABEL EACH FEATURE                  %
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%label connected regions, find mean coordinates of each region
  ima=bw2;
  [cim, num] = bwlabel(ima);   % label each connected region
  %%%%%THIS IS THE NEW CODE FOR GEOMETRICAL CORRECTIONS%%%%%%%%%
R=load('R.dat');
   bw3=geometrical_correction(cim,R); %Uses Sarah Veatch's geometric fix program
[bw4,num]=bwlabel(bw3);
%%%END CODE FOR GEOMETRICAL CORRECTIONS

 close(figure(3))
 cd(thisdatahere)
save([char(traj) char(file) 'CorrectedImage' char(num2str(counter)) '.dat'], 'bw4', '-ascii', '-tabs', '-double')
```

```
   %Change bw2 to bw4 below
BW2=bwperim(bw4,4); %finds the boarder between the phases
sfigure (2), imshow(bw4)
sfigure (5), imshow(BW2)

totalperim=sum(sum(BW2)); %The boarder pixels are designated as a 1 in the
%matrix. This adds them all up to find the length of the total boarder.
[rp,cp]=size(BW2);
%The lines below delete the edges of the matrix since matlab outlines
%domains touching the edge on the edge of the image.
edgeperim=sum(BW2(1,:))+sum(BW2(rp,:))+sum(BW2(:,1))+sum(BW2(:,cp));
perimetersum(counter)=totalperim-edgeperim;

 pause(0.02)
 %adding up the area fraction
   onesinbw4=nonzeros(bw4);
   [l,w]=size(bw4);
   totalsizebw4=l*w;
   allones=length(onesinbw4);
   blackarea(counter)=allones;
   areafraction(counter)=allones/totalsizebw4;
   timelist(counter)=k;
   counter=counter+1;
 clear siz; clear xm; clear ym; clear rad; clear memo;% clear perimeters;
clear bw4; clear BW2; clear cim; clear bw2; clear FEA; clear DUI;

 cd(here)
   k=k+1;
end% this end finishes the first cycle through the time series (k)
    end %ends perfectrun if
end
  % we now have all the important info summarized in the vectors vxm, vym, etc...
%positions (cc) along these vecors refer to the
  % feature of numbers memo(cc) in the  file saved from the matrix cim.
sfigure (5000)
plot(areafraction)
title('areafraction')

cd(thisdatahere)
areaofmovie=RECT(3)*RECT(4);
normalizedperimeter=blackarea./perimetersum; % REALLY R from Fan,Han,Haataja 2010

    save([char(traj) 'perimetersum' char(file) '.dat'], 'perimetersum', '-ascii', '-tabs', '-double');
    save([char(traj) 'areafraction' char(file) '.dat'], 'areafraction', '-ascii', '-tabs', '-double');
    save([char(traj) 'normalizedperimeter' char(file) '.dat'], 'normalizedperimeter', '-ascii', '-tabs', '-double');
    save([char(traj) 'blackarea' char(file) '.dat'], 'blackarea','-ascii', '-tabs', '-double');
    save([char(traj) 'timelist' char(file) '.dat'], 'timelist','-ascii', '-tabs', '-double');
    cd(hd)

   clear arealist
  clear diam
  clear ecclist
  clear perim
```

```
clear areafraction
clear m1
clear m2
clear m3
clear mmm
clear thresholdvaluem1
clear thresholdvaluem1m2
clear thresholdvaluem1m3
clear thresholdvaluem2m1
clear thresholdvaluem2
clear thresholdvaluem2m3
clear thresholdvaluem3m1
clear thresholdvaluem3m2
clear thresholdvaluem3
clear BW2
clear perimetersum
clear blackarea
clear in
```

Two additional programs, growthexponet_startanywhere.m and growthexponent_choose.m were written and are extremely similar to code above. The program growthexponent_startanywhere.m is a modified form of growthexponent.m and is different for the method by which it loops through movies and by which it saves and names output files. The program growthexponent_choose.m is similar growthexponent.m but retains some tracking code from tracker_diffusion.m.

Run_all_growth

```
%Run all for growth
%This is for getting the growth exponent and will be added to the run_all* files upon completion.
%This run_all file will plot several measurements in time.
%Written by Cynthia A. Stanich
clear all
%close all
sets='C:\Program Files (x86)\MATLAB\R2007a Student\work\diffusion\halfandhalf\water\9-13-
10\v1\movies'; %The movies you are interested in
setz=1;
%Need to calculate the seconds.
TimeStart=[4,02,15;
4,07,15;
4,11,14;
4,15,57;
4,20,19;
4,25,14];

cd('C:\Program Files (x86)\MATLAB\R2007a Student\work\diffusion\halfandhalf\water\9-13-
10\v1\movies\v1Grow') %you will need to put the traj directory from the param file here.
[row,column]=size(TimeStart);
framelist=[389,121,121,121,121,121]; %Lengths in frames of all the movies
firstmovielength=framelist(1);
adjfirstmovieindex=firstmovielength-(121*2); %You only want the last 121 frames of the first movie, or the first
movie weights the average too much.
traj='v1Grow'; %Traj
hour = TimeStart(1,1);
minute = TimeStart(1,2);
second = TimeStart(1,3);
sec(1,1)=second+(60*minute)+(3600*hour);
timelist=[0.5:0.5:framelist(1)/2];
for loop=2:row
 hour=TimeStart(loop,1);
 minute=TimeStart(loop,2);
 second=TimeStart(loop,3);
 sec(1,loop)=second+(60*minute)+(3600*hour);
%  if (loop==1)
% timelist=[0.5:0.5:framelist(1)/2];%sets initial time
% else if (loop~=1)
    nextstart=sec(loop)-sec(1);
    nextend=nextstart+(((framelist(loop))-1));
    sequence=nextstart:nextend;
    timelist=[timelist,sequence];
  %  end
  %  end
end
save([char(traj) 'TimeList' '.dat'],'timelist','-ascii','-tabs','-double')

%You will need different files depending on the growth exponent measurement
%you are doing. This is for a sequence of movies in the same quench.
filelist=['v1Growperimetersumv1001.tif.dat';
```

```
        'v1Growperimetersumv1002.tif.dat';
        'v1Growperimetersumv1003.tif.dat';
        'v1Growperimetersumv1004.tif.dat';
        'v1Growperimetersumv1005.tif.dat';
        'v1Growperimetersumv1006.tif.dat'];
filelist2=['v1Growareafractionv1001.tif.dat';
         'v1Growareafractionv1002.tif.dat';
         'v1Growareafractionv1003.tif.dat';
         'v1Growareafractionv1004.tif.dat';
         'v1Growareafractionv1005.tif.dat';
         'v1Growareafractionv1006.tif.dat'];
filelist3=['v1Grownormalizedperimeterv1001.tif.dat';
        'v1Grownormalizedperimeterv1002.tif.dat';
        'v1Grownormalizedperimeterv1003.tif.dat';
        'v1Grownormalizedperimeterv1004.tif.dat';
        'v1Grownormalizedperimeterv1005.tif.dat';
        'v1Grownormalizedperimeterv1006.tif.dat'];
filelist4=['v1Growblackareav1001.tif.dat';
        'v1Growblackareav1002.tif.dat';
        'v1Growblackareav1003.tif.dat';
        'v1Growblackareav1004.tif.dat';
        'v1Growblackareav1005.tif.dat';
        'v1Growblackareav1006.tif.dat'];

for i=1:row;
nextperiminline=load(filelist(i,:));
nextafinline=load(filelist2(i,:));
nextRinline=load(filelist3(i,:));
nextblackarea=load(filelist4(i,:));
if  i==1
   periminorder=nextperiminline;
   afinorder=nextafinline;
   Rinorder=nextRinline;
   bainorder=nextblackarea;
else
   periminorder=[periminorder,nextperiminline];
   afinorder=[afinorder,nextafinline];
   Rinorder=[Rinorder,nextRinline]; %Basically the most important: R = area/perimeter
   bainorder=[bainorder,nextblackarea];
end
end
timelist=load('v1GrowTimeList.dat');

figure (16) %Perimeter vs time
plot(timelist,periminorder)
title('Total Perimeter vs time q5' )
xlabel('Time')
ylabel('Perimeter')

%Best fit for log(perim) vs log(time). CurveFitting.m is a program I wrote
%to report more statistics than Matlab has avaiable
%ae is the error in the average
%r^2 is the sum of the square of the residuals
```

```
[a_long,b_long,ae_long,r2_long]=curveFitting(log(timelist(adjfirstmovieindex:length(timelist))),log(periminorder(adjfirstmovieindex:length(timelist))))

x=log(max(timelist)+3);
y=0;
b=b_long;
idealfity=((-1/3)*log(timelist))+b; %Creates a trend line

%This is Sarah's Idea to relate perimeter with area fraction
afideal=ones(1,length(timelist))*afinorder(adjfirstmovieindex);
perimadj=(afideal(1).*periminorder)./afinorder;

[sarah_a_long,sarah_b_long,sarah_ae_long,sarah_r2_long]=curveFitting(log(timelist(adjfirstmovieindex:length(timelist))),log(perimadj(adjfirstmovieindex:length(timelist))))
b=sarah_b_long;
idealfity=((-1/3)*log(timelist))+b; %creates a trendline

figure (31), hold on %plots log(perimeter) vs log(time)
plot(log(timelist(1:adjfirstmovieindex)),log(perimadj(1:adjfirstmovieindex)),'Color',[0.5 0.5 0.5])
plot(log(timelist(adjfirstmovieindex:length(timelist))),log(perimadj(adjfirstmovieindex:length(timelist))),'k')
%plot(log(timelist),(sarah_a_short*log(timelist))+sarah_b_short,'--c')%early
plot(log(timelist),(sarah_a_long*log(timelist))+sarah_b_long,'--m')%late
%plot([log(45.5),log(45.5)],[-5,20],'r')
plot([log(timelist(adjfirstmovieindex)),log(timelist(adjfirstmovieindex))],[-5,20],'g')
plot([log(max(timelist)),log(max(timelist))],[-5,20],'b')
plot(log(timelist),idealfity,'--c')
legend('Early Perimeter Data','Fitable Perimeter Data','fit','time = Start','time = End','y = -1/3 x + b')
title('log-log Perimeter vs Time - P/AF=Padj/AFideal (Sarah) q5','fontsize',14)
xlabel('log Time', 'fontsize',14)
ylabel('log Perimeter','fontsize',14)


%This is Aurelia's idea to relate perimeter with area fraction
afsqrt=sqrt(afinorder*100);
perimadjahs=periminorder./afsqrt;
[aurelia_a_long,aurelia_b_long,aurelia_ae_long,aurelia_r2_long]=curveFitting(log(timelist(adjfirstmovieindex:length(timelist))),log(perimadjahs(adjfirstmovieindex:length(timelist))))
%[aurelia_a_short,aurelia_b_short,aurelia_ae_short,aurelia_r2_short]=curveFitting(log(timelist(91:201)),log(perimadjahs(91:201)))
b=aurelia_b_long-0.2;
idealfity=((-1/3)*log(timelist))+b;


figure (51), hold on %Plots area fraction in time
plot(timelist(1:length(timelist)),afinorder(1:length(timelist)),'k')
plot(timelist(1:length(timelist)),afideal(1:length(timelist)),'r')
title('Area Fraction vs Time q5', 'fontsize',14)
xlabel('Time', 'fontsize',14)
ylabel('Area Fraction','fontsize',14)
legend('Actual Area Fraction', 'Ideal Area Fraction')
%perimadjnolog=(afideal(1).*periminorder)./afinorder;
starttime=timelist(adjfirstmovieindex)
endtime=max(timelist)
```

```
figure (61), hold on %Plots R in time
plot(timelist(1:length(timelist)),Rinorder(1:length(timelist)),'k')
title('R = (black area/total perimeter) vs Time q5', 'fontsize',14)
xlabel('Time', 'fontsize',14)
ylabel('R pixels','fontsize',14)
[Ra_long,Rb_long,Rae_long,Rr2_long]=curveFitting(log(timelist(adjfirstmovieindex:length(timelist))),log(Rinorder(a
djfirstmovieindex:length(timelist))))

figure (71), hold on  % Plots the minority phase in time
plot(timelist(1:length(timelist)),bainorder(1:length(timelist)),'k')
title('Black Area vs Time q5', 'fontsize',14)
xlabel('Time', 'fontsize',14)
ylabel('Black Area pixels^2','fontsize',14)
timelistadj=timelist(adjfirstmovieindex:length(timelist));
Rinorderadj=Rinorder(adjfirstmovieindex:length(timelist));
save(['TimeListadj' '.dat'],'timelistadj','-ascii','-tabs','-double')
save(['Rinorderadj' '.dat'],'Rinorderadj','-ascii','-tabs','-double')
```

CurveFitting

```
%%% DIFFUSION AND GROWTH - Curve Fitting %%%
% Written by Cynthia A. Stanich
%This function fits a linear line to data. It returns four values. a is the
%slope of the line y=ax+b and b is the intercept. aerror is the error in
%the slope due to the fit to the data. corcoe is the correlation coefficient
%(aka R^2).
%clear all
%close all
function [a,b,aerror,corcoe]=curveFitting(xi,yi)
%Curve fitting tool given to me by Aurelia is below
%Uses the statistics toolbox and optimization toolbox, which I have!
%lsqcurvefit: Find coefficients x that best fit the equation shown in help
%given input data xdata, and the observed output ydata,
%where xdata and ydata are vectors of length m and F(x, xdata) is a
%vector-valued function.
%[P,resnorm,residual,EXITFLAG,OUTPUT,LAMBDA,Jacobian] = lsqcurvefit('exponential', Po, x, y);
%CI = nlparci(P,residual,'jacobian',Jacobian, 'alpha', .34);
%dP = .5*(CI(:, 2)-CI(:, 1));
%Start my curve fitting tool below this line
n=length(yi);
sumxy=sum(xi.*yi);
sumx=sum(xi);
sumy=sum(yi);
sumxsqrd=sum(xi.^2);
sumysqrd=sum(yi.^2);
a=((n*sumxy)-(sumx*sumy))/((n*sumxsqrd)-sumx^2);
b=((sumxsqrd*sumy)-(sumx*sumxy))/((n*sumxsqrd)-sumx^2);
sumeqn=yi-(a*xi)-b;
S=sqrt(sum(sumeqn.^2)/(n-2));
aerror=S*sqrt(n/((n*sumxsqrd)-sumx^2));
corcoe=(n*sumxy-sumx*sumy)/(sqrt(n*sumxsqrd-sumx^2)*sqrt(n*sumysqrd-sumy^2));
```

Run_all_Ost

```
%This is to run the Ostwald ripening data output files generated from growthexponent_choose and a
%param_growth file.
%%Written by Cynthia A. Stanich
clear all
close all
%FILES GO INTO THE SAME PLACE AS *slopes.dat and *areafraction.dat Which is
%the movies folder
sets='C:\Program Files (x86)\MATLAB\R2007a Student\work\diffusion\Ostwald\5% Dextran\12-7-
09\v2\ostwald2nd2\movies';
setz=1; %There can be more than one directory of data.
%for n=1:setz
n=1;
cd(sets(n,:))
numberofdomains=9; % It is easier to count the number of domains while tracking them by hand
using %growthexponent_choose.m
ColorSet=varycolor(numberofdomains); %You should set this for the number of domains there are.
letters=['a';'b';'c';'d';'e';'f';'g';'h';'I';'j';'k';'l';'m';'n';'o';'p';'q';'r';'s';'t';'u';'v';'w';'x';'y';'z'];
%clr=colorset(n,:);
for i=1:numberofdomains;%Can set this to stop after a known number of domains, otherwise the ColorSet will not
work.
   %Go through all letters
   filetoload=dir(strcat(letters(i),'*','AREA.dat'));
  timetoload=dir(strcat(letters(i),'*','times.dat'));
  s=size(filetoload);
  filename=char(filetoload.name);
  timename=char(timetoload.name);
  %I need to be able to distinguish if I am using domain A or B or C
  %etc...

  for j=1:s(1); %Once you do this you can comment out and use the loop after the disk radius code
     newarea=load(filename(j,:));
     %newarea=newarea';
     newtime=load(timename(j,:));
     newtimelist=([newtime(1):newtime(1)+newtime(2)-1]*0.5)-0.5;
     if j==1 %Have to start the lists, or you can create a blank one at the beginning of the code
       area=newarea;
      timelist=newtimelist;
    else
       area=horzcat(area,newarea);
       timelist=horzcat(timelist,newtimelist);
     end
    radius=sqrt(area/pi); %Assumes circle (I believe it is safe to assume since we are only tracking “good” domains.)
    radius=radius*.18; %convert to microns
    %area = area *.18^2; %microns squared
    boxcarave=moving_average(radius,2,2); % smooths the data
    Aboxy=Aboxend(radius, 2); %%Aboxend written by Aurelia R. Honerkamp-Smith
    Aboxmed=Aboxmedian(radius,2); %%Aboxmedian written by Aurelia R. Honerkamp-Smith
    timelisttothird=timelist.^(1/3);
save([letters(i) 'domainradius' '.dat'],'radius','-ascii','-tabs','-double')
save([letters(i) 'domaindata' '.dat'],'area','-ascii','-tabs','-double')
```

```
save([letters(i) 'domaintime' '.dat'],'timelist','-ascii','-tabs','-double')
%This creates a list of all the data found for each domain.

figure (1), hold on % area versus time
set(gca,'YGrid', 'on')
area2 = area*.18^2;
plot(timelist,area2,'Color',ColorSet(i,:))
title('Area of Domains vs Time')
xlabel('Time (s)')
ylabel('Area (micron^2)')

figure (2), hold on  %radius versus time
set(gca,'YGrid','on')
plot(timelist,radius,'Color',ColorSet(i,:))
title('Radius of Domains vs Time')
xlabel('Time (s)')
ylabel('Radius (micron)')

figure (3), hold on  % using moving_average on radius
set(gca,'YGrid','on')
plot(timelist,boxcarave,'Color',ColorSet(i,:))
title('Boxcar Average of Domains vs Time')
xlabel('Time (s)')
ylabel('Radius (mircon)')

figure (4), hold on  % using Aboxend on radius
set(gca,'YGrid','on')
plot(timelist,Aboxy,'Color',ColorSet(i,:))
title('Boxcar Average With Aboxend of Domains vs Time')
xlabel('Time (s)')
ylabel('Radius (mircon)')

figure (5), hold on  % using Aboxmedian on radius
set(gca,'YGrid','on')
plot(timelist,Aboxmed,'Color',ColorSet(i,:))
title('Boxcar Median With Aboxmedian of Domains vs Time')
xlabel('Time (s)')
ylabel('Radius (mircon)')

figure (6), hold on  % plotting radius versus  t^(1-3)
set(gca,'YGrid','on')
plot(timelisttothird,radius,'Color',ColorSet(i,:))
title('Radius of Domains vs Time')
xlabel('Time (s)')
ylabel('Radius (micron)')
  end
  clear area
  clear timelist
  clear radius
end

% For Ostwald ripening measurement, we are looking at very small changes in domain radius. So we wanted to
```

```
%take into account the movement of the vesicle in the z direction. Track_vesicle saves the disk radius from
the %centering program. Load it here:
diskrad=load('C:\Program Files (x86)\MATLAB\R2007a Student\work\diffusion\Ostwald\5% Dextran\12-7-
09\v2\ostwald2nd2_2.tifdiskrad.dat');
diskrad=diskrad*.18; %change to microns

%smooth diskrad
Aboxmed2=Aboxmedian(diskrad(:,3),2);
Aboxmed2Area = pi*(Aboxmed2.^2)
%diskarea = pi * (diskrad.^2)
i=1;
% while i < length(diskrad)
% diskarearatio (i) =Aboxmed2Area(i+1)/Aboxmed2Area (i);
diskarearatio = Aboxmed2Area/Aboxmed2Area(1);
diskarea = pi * (diskrad(:,3).^2);
diskareadiff = diskarea - diskarea(1);
diskraddiff = diskrad(:,3) - diskrad(1,3);
Aboxmed3 = Aboxmedian(diskraddiff,2);
Aboxmed3Area = pi * (Aboxmed3.^2);
% i=i+1;
% end

for j = 1:numberofdomains %This is for after you do the loop above the disk radius code.
filetoload=load(strcat(letters(j),'domainradius.dat'));
timetoload=load(strcat(letters(j),'domaintime.dat'));
areatoload=load(strcat(letters(j),'domaindata.dat'));
boxcarave=moving_average(filetoload,2,2);
    Aboxy=Aboxend(filetoload, 2);
    Aboxmed=Aboxmedian(filetoload,2);
aboxcarave=Aboxmedian(areatoload,2);%pixels
aboxcaravemic = aboxcarave * (.18^2);
m=1;
domainarea(1)=aboxcaravemic(1);

while m < length(timetoload)
  index = (timetoload(m) * 2) +1;
  domainarea(m+1) = diskarearatio (index) * aboxcaravemic(m+1)
  m=m+1;
end
Aboxdomainarea = domainarea;
%aboxdomainarea = Aboxdomainarea * (.18^2);

 figure (8), hold on %  Plots a line for each domain for radius versus time
set(gca,'YGrid', 'on')
plot(timetoload,filetoload,'Color',ColorSet(j,:))
title('Radius of Domains vs Time')
xlabel('Time (s)')
ylabel('Radius (micron^2)')

figure (9), hold on  % Plots a line for each domain using moving_average for radius versis time
set(gca,'YGrid','on')
plot(timetoload,boxcarave,'Color',ColorSet(j,:))
title('Boxcar Average of Domains vs Time')
```

```
xlabel('Time (s)')
ylabel('Radius (mircon)')

figure (10), hold on% Plots a line for each domain using Aboxend for radius versis time
set(gca,'YGrid','on')
plot(timetoload,Aboxy,'Color',ColorSet(j,:))
title('Boxcar Average With Aboxend of Domains vs Time')
xlabel('Time (s)')
ylabel('Radius (mircon)')

figure (11), hold on % Plots a line for each domain using Aboxmedian for radius versis time
set(gca,'YGrid','on')
subplot(2,1,2),plot(timetoload,Aboxmed,'Color',ColorSet(j,:))
title('Boxcar Median With Aboxmedian of Domains vs Time')
xlabel('Time (s)')
ylabel('Radius (mircon)')

figure (12), hold on % Aboxmedian is the best smoother we have. We used that one to change radius to microns
%and plot radius versus time
set(gca,'YGrid','on')
%subplot(2,1,2),
plot(timetoload,aboxcaravemic,'Color',ColorSet(j,:))
%subplot(2,1,2),plot(timetoload,domainarea,'Color',ColorSet(j,:))

figure (13), hold on % Plots the individual domain areas versus time.
set (gca,'YGrid','on')
subplot(2,1,2),plot(timetoload,Aboxdomainarea,'Color',ColorSet(j,:))

clear domainarea
clear timetoload
clear filetoload
end


figure (11), hold on
subplot(2,1,1),plot([1:481]*0.5, Aboxmed2,'Color', [.5,.5,.5]) %Plots the smoothed disk radius

figure (13), hold on
subplot(2,1,1),plot([1:481]*0.5,Aboxmed2Area(1)*ones(1,481)) %Plots the smoothed disk area
```

Mkparamfiles

```
% Written by Chris Warth to generate param files originally written by Cynthia Stanich
%Automatically make diffusion parameter files for movies.
function mkparamfiles(name_template, filelist)
  % You can input a list of files or just one file name. The file list is the list of movies. The name template here is
  %your choice of how you want to name your param files. I used ‘param_(growth or diffco)_(movie name and
  %quench)’.
  %generate a list of file from the file specification,
   % e.g. '*.tif' might generate 'foo1.tif bar.tif xxx.tif'
   %
   foo=dir(filelist);
   bar = {foo(:).name};

   % outlist will be filled with the names of all the param files that are
   % generated.  This is used to make a single script that will run all
   % the param files.
   %
   outlist = [];
   traj = 1; %I used traj to say what the first number of the param sequence should be.
   for i = 1:numel(bar),
      fname = bar{i};
      disp(fname);

      [nFiles outnames] = mkparamfile(fname, traj, name_template);
      traj = traj + nFiles;

      outlist = [outlist outnames];
   end
    mkrunall(outlist);
  end
 % create an individual param file.
% arguments are
%   - name of the .tif movie file.
%   - a template for the name of the param file, e.g. 'param_9_7_09_v1_q1'
%   - traj, the starting number for the param file.
%
%  This function will generate a bunch of param files for the tif movie
%  file, one param file for every 10 frames of the movie.
%  The param files will be named according to the filename template,
%  appended with a string like '_d1.m' with the number sequentially
%  increasing for every 10 frames.
function [nFiles, outnames] = mkparamfile(fname, traj, name_template)

   iFrame = 10;  % frame increment.
   [pathstr, basename, ext, versn] = fileparts(fname);
   imgInfo = imfinfo(fname);
   tFrames = numel(imgInfo);
   w = imgInfo.Width;
   h = imgInfo.Height;
   w = min(w,h);
   h = min(w,h);
```

```
% save this for later so we can see how many param files we generated
% for this movie.
initial_traj = traj;

outnames = [];
for i=1:ceil(tFrames/iFrame)

   nFrame =((i-1) * iFrame)+1;

   if (nFrame + iFrame >= tFrames)
      if (tFrames - nFrame <= 7)
         break;
      end
      iFrame = tFrames - nFrame;
   end

   [outname] = sprintf([ name_template '_d%d'], traj);
   fid = fopen([ outname '.m'], 'w');
   %Below is the content for the param files
   fprintf(fid, '\n\n');
   fprintf(fid, 'function %s(use_previous)\n', outname);
   fprintf(fid, '\n');
   fprintf(fid, 'using_previous_choice_of_domains=1;\n');
   fprintf(fid, 'if (nargin ~= 0)\n');
   fprintf(fid, '\tusing_previous_choice_of_domains = str2num(use_previous);\n');
   fprintf(fid, 'end\n');
   fprintf(fid, '\n\n');
   fprintf(fid, 'close all\n');
   fprintf(fid, 'firstgo=0;\n');
   fprintf(fid, 'TEST=0;\n');
   fprintf(fid, 'useframes=%d;\n', iFrame);
   fprintf(fid, 'cut=16;  \n');
   fprintf(fid, 'maxdiam=400; \n');
   fprintf(fid, 'WHITEDOMAINS=1;\n');
   fprintf(fid, 'inputgrossDY= 0;\n');
   fprintf(fid, 'inputgrossDX= 0;\n');
   fprintf(fid, 'filedirectory=[''%s''];\n', cd);
   workdir = fullfile(matlabroot, 'work');
   fprintf(fid, '%% cd(''%s'')\n', workdir);
   fprintf(fid, 'hd = [''%s''];\n', workdir );
   fprintf(fid, 'traj={''d%d''};\n', traj);
   fprintf(fid, 'file=''%s''; \n', fname);
   fprintf(fid, '% using_previous_choice_of_domains=1;\n');
   fprintf(fid, 'threshold =1\n');
   fprintf(fid, 'initial=[%d]; %% %d frames total\n', nFrame, tFrames);
   fprintf(fid, 'RECT(:,1)=[1   1   %d  %d]''; \n', w, h);
   fprintf(fid, '    smallf=0.035; bigf=0.13;   \n\n');
   fprintf(fid, 'NeedsAureliaGradient=0;\n');
   fprintf(fid, 'FILT=1;\n\n');
   fprintf(fid, 'for FFF=1:length(initial),\n');
   fprintf(fid, '    tracker_diffusion2\n\n');
   %fprintf(fid, '    track_diffusion\n\n');
%       fprintf(fid, '    OUTGOOD(FFF) = Nfeg;\n');
```

```
%       fprintf(fid, '    OUTX(:,FFF) = meandevx2;\n');
%       fprintf(fid, '    OUTY(:,FFF) = meandevy2;\n');
      fprintf(fid, 'end\n');
      fprintf(fid, 'end\n');
      fclose(fid);
      outnames = [outnames cellstr(outname)];
      traj = traj + 1;
    end

   nFiles = traj - initial_traj;
end

function mkrunall(outlist)
   % write the 'runall.m' file that collects all the generated param files
   % in one place.
   fid = fopen('runall.m', 'w');
   fprintf(fid, 'function runall(use_previous)\n');
   fprintf(fid, '\n');
   fprintf(fid, 'using_previous_choice_of_domains=1;\n');
   fprintf(fid, 'if (nargin ~= 0)\n');
   fprintf(fid, '\tusing_previous_choice_of_domains = str2num(use_previous);\n');
   fprintf(fid, 'end\n');


   for i = 1:numel(outlist),
      fprintf(fid, '%s using_previous_choice_of_domains\n', outlist{i});
   end
   fprintf(fid, 'end\n');
   fclose(fid);
end
```

Curriculum Vitae

Cynthia Ann Stanich

Education

Bachelor of Arts in Chemistry 2006
Michigan State University
Advisor: Professor James Harrison

Bachelor of Science in Physiology 2007
Michigan State University
Advisor: Professor Robert W. Wiseman

Bachelor of Arts in East Asian Languages and Cultures 2007
Michigan State University

Master of Science in Chemistry 2009
University of Washington

Ph.D. in Physical Chemistry 2012
University of Washington
Advisor: Professor Sarah L. Keller

Honors and Awards

Honen Fellowship, University of Washington 2007-8
Travel Award, Chemistry Education Research Grad. Student Conference, Miami Univ., OH 2009
Travel Award, Biophysical Society Annual Conference, San Francisco, CA 2010
Northwest POGIL Conference, 1 of 50 chosen participants 2010
Outstanding Teaching Award, Department of Chemistry, University of Washington, Seattle 2010-11
Biophysical Society Student Research Achievement Award, Second Place, Baltimore, MD 2011
UW - Chemistry Travel Award, to Biophysical Society Annual Conference, Baltimore, MD 2011
Travel Award, TRUSE Conference, University of St. Thomas in St. Paul, MN 2012
Biophysical Society Student Research Achievement Award, First Place, San Diego, CA 2012
Travel Award, TRUSE Conference, University of St. Thomas in St. Paul, MN 2012

Memberships in Professional Organizations

Biophysical Society
ACS Division of Chemical Education

Contributed presentations at National Conferences

Chemistry Education Research Graduate Student Conference poster presentation. 2009
"Undergraduate students' proficiency in math and attitudes about math."

Biophysical Society poster presentation. 2010
"Direct measurement of time-dependent domain coarsening in giant unilamellar vesicles."

Biophysical Society poster presentation. 2nd place Student Research Achievement Award. 2011
"Measurement of late stage coarsening of domains in lipid membranes."

Biophysical Society poster presentation. 2012
"Math preparation of undergraduates in general chemistry, a gatekeeper course for biophysicists."

Biophysical Society poster presentation. 1st place Student Research Achievement Award. 2012
"Coarsening dynamics of domains in lipid membranes."

Conference on Transforming Research in Undergraduate STEM Education (TRUSE). 2012
"Math preparation of undergraduates in general chemistry, a gatekeeper course for biophysicists."

Contributed presentations at the University of Washington

UW Teaching and Learning Symposium. 2012
"Math preparation of undergraduates in general chemistry, a gatekeeper course for biophysicists."

Publication

"Schrodinger Cats in Double Well Bose Condensates: Modeling Their Collapse and Detection Via Quantum State Diffusion" William P. Reinhardt, Cynthia A. Stanich, Cory D. Schillaci, *Applied Mathematics and Information Sciences,* 3(3), 273-299, 2009

In preparation

"Late Stage Coarsening Dynamics of Domains in Lipid Membranes" Cynthia A. Stanich, Aurelia R. Honerkamp-Smith, Gregory Garbès Putzel, Christopher S. Warth, Andrea K. Lamprecht, Pritam Mandal, Elizabeth Mann, Thien-An D. Hua, and Sarah L. Keller