UNIVERSITY OF CALIFORNIA,
IRVINE

Creation and Detection of Small Bubble Clusters

THESIS

submitted in partial satisfaction of the requirements
for the degree of

MASTER OF SCIENCE

in Physics

by

Nathaniel See

Thesis Committee:
Professor Michael Dennin, Chair
Assistant Professor Jun Allard
Professor Zuzanna Siwy

2017

TABLE OF CONTENTS

Page

LIST OF FIGURES **iv**

LIST OF TABLES **v**

ACKNOWLEDGMENTS **vi**

ABSTRACT OF THE THESIS **vii**

1 Introduction **1**
- 1.0.1 Soft Condensed Matter . . . 1
- 1.0.2 Foams . . . 2
- 1.0.3 Bubble Clusters . . . 4

2 Methods **5**
- 2.1 Apparatus . . . 5
  - 2.1.1 Trough . . . 5
  - 2.1.2 Bubble Solution . . . 5
  - 2.1.3 Bubble Production . . . 7
  - 2.1.4 Mixing Plate . . . 7
  - 2.1.5 Fan . . . 8
  - 2.1.6 Camera . . . 8
  - 2.1.7 Bubble Removal . . . 9
  - 2.1.8 Barriers . . . 9
- 2.2 Software . . . 9
  - 2.2.1 Fan and Camera . . . 9
  - 2.2.2 Image Processing . . . 10
- 2.3 Procedures . . . 14

3 Results **16**
- 3.1 Data . . . 16
  - 3.1.1 Data Location and Storage . . . 23
- 3.2 Other Notes and Observations . . . 24
  - 3.2.1 Bubble Production . . . 24
  - 3.2.2 Failure Modes . . . 25

4 Discussion **27**
4.1 Future work 29
4.1.1 Mixing Plate 29
4.1.2 Improving the Stability of the Experimental Setup 29
4.1.3 General Cluster Recognition 30
4.1.4 Video Tracking Script 31

Bibliography **35**

A Image Processing Code **36**

LIST OF FIGURES

Page

2.1 Apparatus 6
2.2 Enumeration of stable six-bubble states [1] 12

3.1 Runs 1 and 2: 4% Glycerol, 5% Detergent 17
3.2 Run 3: 20% Glycerol, 5% Detergent 17
3.3 Run 4: 36% Glycerol, 5% Detergent 17
3.4 Run 5: Modified mixing plate, 4% Glycerol, 5% Detergent 18
3.5 Run 6: Modified mixing plate, 20% Glycerol, 5% Detergent 18
3.6 Run 7: Modified mixing plate, 30% Glycerol, 5% Detergent 19
3.7 Run 8: Modified mixing plate, 20% Glycerol, 10% Detergent 19
3.8 Run 9: Modified mixing plate, 20% Glycerol, 15% Detergent 19
3.9 Run 9: Modified mixing plate, 20% Glycerol, 20% Detergent 20
3.10 Precursor Clusters 21

4.1 Mixing Plate Regions 28
4.2 Contacts Between Like Bubbles 29
4.3 Solution Reservoir 30

LIST OF TABLES

Page

2.1 Vertex codes for valid combinations of edges 12
2.2 Sequences corresponding to cluster states 13

3.1 Formation of 6-bubble Clusters 22
3.2 Data File Locations 23

ACKNOWLEDGMENTS

I would like to thank Professor Michael Dennin and Dr. Chin-Chang Kuo for their help.

ABSTRACT OF THE THESIS

Creation and Detection of Small Bubble Clusters

By

Nathaniel See

Master of Science in Physics

University of California, Irvine, 2017

Professor Michael Dennin, Chair

I describe a process for creating small bubble clusters and an image processing script for recognizing them. Bubbles are created by flowing nitrogen through submerged needles and driven forward by a fan to an imaging area. Clusters are recognized by identifying contact points between bubbles in a cluster, representing the cluster as a contact graph, and checking for isomorphism with graphs of known configurations. Frequency distributions of clusters identified by this method are shown.

Chapter 1

Introduction

1.0.1 Soft Condensed Matter

The field of soft matter studies materials that are easily deformed, typically by energies of order $k_B T$. Often, such materials exhibit properties intermediate between those of solids and liquids.

A solid is a material whose constituent particles tend to remain in the neighborhood of their initial positions when a stress is applied. Solids have a nonzero shear modulus, which means that a constant applied stress produces a finite strain, provided the applied stress is not too large. In the ideal case of a Hookean solid, this strain is exactly proportional to the stress [2, 3].

Conversely, the shear modulus of a liquid is zero, which means that particles may move arbitrarily far from their initial locations when a shear is applied. In response to an applied shear, a liquid flows with a strain rate that depends on the applied shear stress. In the ideal case of a Newtonian fluid, this strain rate is exactly proportional to the shear stress.

Many soft matter systems respond to stresses in ways that lie between these two extremes. Typically, this property is the result of structural components that are much larger than individual atoms but much smaller than the system in question [2]. Such systems tend to be disordered – that is, their components are not organized in a periodic lattice [4]. Examples of soft materials includes polymer solutions and melts, granular media, gels, colloids, and liquid crystals.

Although there exists a great variety of soft materials, many soft and disordered systems can be described by theoretical models that are insensitive to the specific nature of the systems' interparticle interactions. Often, they describe large scale motion in terms of local regions that are particularly susceptible to deformation. One such model focuses on rearrangements of small groups of particles (typically fewer then ten) located in *shear transformation zones* (STZs), which are created and annihilated as a material is sheared [5]. Other theories based on local rearrangements include weak-zone and soft spot-theories.

1.0.2 Foams

Foams – materials consisting of pockets of gas separated by solid or liquid barriers – form another class of soft matter systems. Foams come in many varieties – those with solid barriers are referred to as "solid foams", and may be either 'open-cell' or 'closed-cell'. A closed-cell foam is one in which bubble walls fully enclose regions of gas, while an open-cell foam is one in which those walls may contain holes [6].

In contrast, liquid foams always have closed cells. The structure of such foams depends on amount of gas present, as a fraction of the system's total volume. In the dry foam limit, where there is much less liquid than gas, cells are polyhedral and bounded by thin, curved films. If the amount of liquid present is raised past a certain point, a foam undergoes a melting transition into a bubbly liquid, where cells are spherical and rarely touch each other.

Near this transition point lie wet foams, in which cells do touch but remain roughly spherical except for deformations at contact points [7].

Three-dimensional foams tend to be opaque and thus difficult to probe experimentally. As a result, experiments on and simulations of two-dimensional foams are vital to our understanding of foams in general. A two-dimensional foam is one in which the positions and motion of bubbles are largely confined to a plane [6]. Example of two-dimensional foams include bubble rafts [8, 9], langmuir monolayer foams [10], and layers of bubbles sandwiched between solid plates [11, 12]. In two-dimensional foams, a gas area fraction ϕ is used instead of a gas volume fraction, with a melting transition occurring at $\phi \approx 0.841$ [11].

Foams, both two and three-dimensional, exhibit complex behavior when sheared. For small shears, average stress throughout a foam is approximately proportional to shear strain as is the case in a solid. Individual cells are elastically deformed but not rearranged, and if the applied strain is removed the foam returns to its original state [7]. Above a certain *yield strain*, irreversible plastic deformation occurs as cells rearrange around each other. Finally, above an even higher *critical stress*, the foam begins to flow.

Unlike ordinary fluids, foams do not flow homogeneously. Rather, stress accumulates throughout the system before being released in short bursts of rapid rearrangements known as *avalanches* or *stress drops* [8, 13]. This occurs repeatedly, causing shear stress to fluctuate about some average value as the foam flows.

There exist a number of models that describe these avalanches. However, their specific predictions differ. For dry foams ($\phi \approx 1$), Okuzono and Kawasaki's *vertex model* simulations predict that the energy released in avalanches follows a power law with exponent $-3/2$, with the largest avalanches spanning the entire system [13]. In contrast, Jiang et al. found no indication of a power law or system-spanning events in their large-Q Potts model for dry

foams [14]. Likewise, Hutzler, Weaire, and Bolton observed no evidence of a power law for dry foams in their the quasistatic model [15].

However, they did find evidence of large avalanches and power law behavior in the wet limit ($\phi \approx 0.84$). In that same limit, Durian's bubble model predicts a power law with an exponential cutoffs which would preclude the occurrence of large events [16].

Experimental results vary, with Dennin and Knobler finding no evidence of a power law in langmuir monlayer foams [10] of varying gas fraction, Pratt and Dennin finding evidence of a power law with exponential cutoff in wet bubble rafts [9], and Kader and Earnshaw observing large rearrangement events in foams between glass plates [12].

1.0.3 Bubble Clusters

As mentioned in section 1.0.1, rearrangements of small groups of particles are vital to the description of flow in soft and disordered materials. In a series of experiments and simulations, Lundberg et al. characterized flow in bubble rafts in terms of rearrangements of groups of three and four particles [17].

The experiments done in this thesis is part of a series of experiments and simulations meant to probe the strength and range of interactions between bubbles by examining small, stable bubble clusters. This work focuses on bidisperse clusters, each containing three small and three large bubbles. It describes an apparatus used for generating these clusters, a means of recognizing different configurations of six bubbles by computer image processing, and measurements of the frequencies with which these different configurations occur in solutions of varying viscosity and surface tension.

Chapter 2

Methods

2.1 Apparatus

2.1.1 Trough

The experiments described in this thesis were conducted in a trough constructed by a former member of the group. While the trough was initially T-shaped, the head of the T was sealed off so as to decrease the volume of fluid needed to fill it. What remains is a rectangular section, 40 cm long and 20 cm wide. The bottom of the trough is flat and transparent, below which is located a white LED pad used for illumination. The entire setup is shown in Figure 2.1

2.1.2 Bubble Solution

Bubbles are created in a solution of water, glycerol, and detergent. Depending on the run, the amount of glycerol used varies between 4% and 36%, by volume of unmixed components,

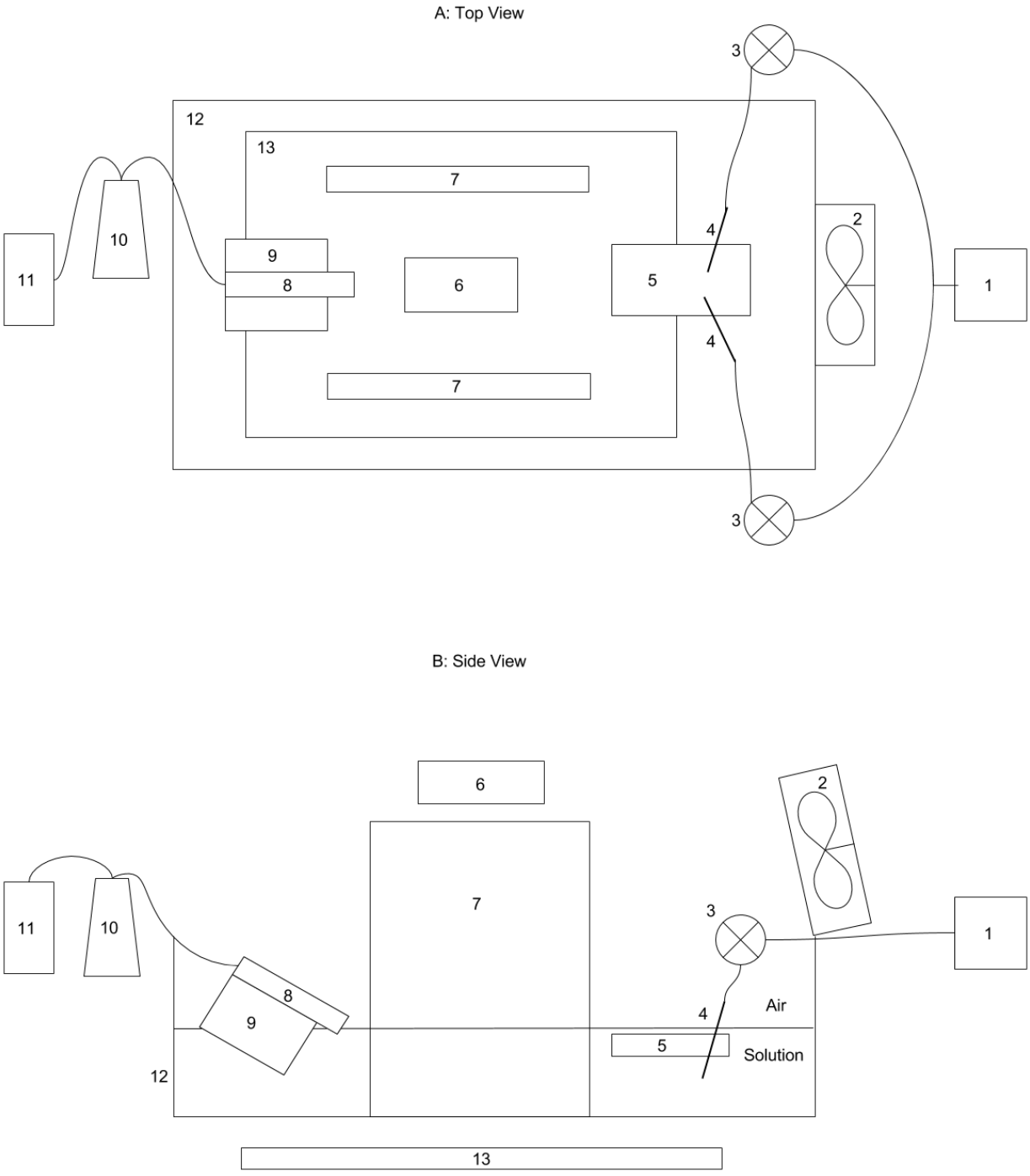

Figure 2.1: 1 - Compressed nitrogen cylinder; 2 - Fan; 3 - Valve; 4 - Needle; 5 - Mixing plate; 6 - Camera; 7 - Barrier; 8 - Bubble removal nozzle; 9 - Boat; 10 - Erlenmeyer flask; 11 - Vacuum pump; 12 - Trough walls; 13 LED pad.

while the amount of detergent used varies between 5% and 20%. In all cases, the volumes of the components sum to 2100 mL, corresponding to a depth of approximately 2.6 cm in the trough. Since the different components of the solution evaporate at different rates, the trough's contents are replaced completely before each run.

2.1.3 Bubble Production

Bubbles of two different sizes are produced by flowing nitrogen gas through a pair of submerged hollow needles with different inner diameters (0.603 mm and 0.377 mm). The regulator valve of a compressed gas cylinder is connected to a plastic tube which branches at a y-shaped connector. Each branch is connected to a separately adjustable valve, which controls the flow of gas through its corresponding needle.

Each needle is connected to its valve by a plastic tube, and to its tube by a rigid connector [picture of needles and valves]. To hold the needle in place, the connector is tied to a straightened steel paper-clip by copper wire. The paper-clips are themselves held in place by binder clips clamped to the walls of the trough, creating a stable yet adjustable support.

Bubble sizes are determined by a combination of needle diameter, gas flow rate, and needle depth. These parameters are tuned so that the ratio of diameters between large and small bubbles is approximately 1.4.

2.1.4 Mixing Plate

A 2.5 cm wide glass microscope slide, referred to as a "mixing plate" is used to merge the two streams of bubbles produced at the needles into a single stream of clusters. The slide is glued to a large block of plastic for stability, leaving a 2.5 cm by 1.4 cm section protruding. The block is positioned below the water's surface such that bubbles rising from the needles

collect beneath the plate. A pair of 3 mm-wide parallel ridges on two sides of the plate prevent bubbles from slipping out sideways. Subsequent bubbles push those already present to the front of the plate, where they detach, rise to the surface, and merge into clusters. Clusters that form in the region immediately outside the mixing plate tend to contain four bubbles or fewer. Most larger clusters are formed downstream when these small clusters merge.

Some runs were conducted with a modified mixing plate, where a layer of silicone glue has been added over the ridges to make them steeper. Bubbles near the modified ridges tend to stick in a consistent manner, decreasing the effective width of the plate to 1.2 cm. Runs conducted with this modified plate are labeled accordingly.

2.1.5 Fan

We use a computer-controlled fan located on the trough's wall above and behind the mixing plate to drive the bubble clusters forward. The effective speed of the fan is controlled by switching it on and off several times per second. The speed of the fan is a major factor in determining the size of created clusters, higher speeds resulting in clusters being blown forward before they can accumulate a large number of bubbles.

2.1.6 Camera

A downward-facing Logitech C920 webcam is suspended above the trough. A Matlab script uses it to take pictures at regular intervals, which are chosen to be slightly longer than the time needed for a cluster to pass from one end of the camera's field of view to the other.

2.1.7 Bubble Removal

Past the camera, bubbles are removed from the trough via a plastic tube. The tube is connected to one of the two openings in a rubber stopper on an Erlenmeyer flask. From the second opening runs another plastic tube which connects to a central vacuum system.

The tube in the trough is supported by a "boat" consisting of a pair of small, sealed, empty plastic bottles. The boat is adjusted so that the tube's opening lies just above the water's surface. The tube is angled so that suction from the vacuum is enough to lift bubbles up the tube's slope to be collected in the flask, but not enough to otherwise draw significant amounts of liquid from the trough.

2.1.8 Barriers

A pair of parallel barriers are glued to the bottom of the trough, lengthwise. Their purpose is to direct the flow of fluid in the trough under the camera and toward the bubble removal tube. Before these barriers were added, flow in the trough tended to be much more erratic.

2.2 Software

2.2.1 Fan and Camera

The fan is controlled by a Matlab script written by Chin-Chang Kuo, another group member. The script runs the fan in cycles, each cycle consisting of two phases of separately adjustable length, one in which the fan is on and one in which the fan is off.

Initially, the phases were intended to last several seconds each and the camera was synchro-

nized with the fan so that it took a picture at the end of every "off" phase, so as to produce the clearest image by photographing bubbles as they were moving the slowest. However, this synchronization proved to be unnecessary for accurate bubble detection, and the large variations in fan speed produced unpredictable currents that made bubble removal difficult.

As a result, I modified the program to take a picture once every n cycles. The cycles themselves were made much shorter, each phase lasting a fraction of a second. This resulted in an effectively constant velocity field in the trough, slow enough that the camera could take clear pictures. Images taken by the camera are saved to a directory for analysis by the image processing script.

2.2.2 Image Processing

The image processing script consists of three basic stages, and is itself a modification of another script written by Ching-Chang Kuo. In the first stage, clusters of bubbles are identified within an image. In the second, individual bubbles are identified, located, and measured within each cluster. The third stage uses the measurements obtained in the second stage to determine if the cluster's state corresponds to one of the 24 configurations listed in Figure 2.2.

To begin the first stage, the images are converted to black and white. Bubbles within a cluster appear as black rings, usually separated by narrow white regions. A morphological close followed by a morphological open operation is then performed on the image to bridge these gaps and close any apparent openings in the bubble walls. This turns clusters into connected components of black pixels. These connected components are detected using the function bwconncomp from Matlab's image processing toolbox. Each connected component is cropped and isolated for separate analysis in the second stage.

The second and third stages assume that all bubbles fall within one of two size ranges, referred to as "large" and "small". These size ranges are adjusted for each run to match the sizes of the bubbles produced at each needle.

In the second stage, the interiors of bubbles correspond to connected components made of white pixels surrounded by black, which are identified in the same way as the previous stage. Components that do not fall into the two size ranges are discarded. The number, sizes, and positions of the components that remain are recorded for use in the third stage.

The third stage examines only clusters with three large bubbles and three small bubbles: $n_L = n_S = 3$. A cluster can be represented by a graph, where each node on the graph corresponds to a bubble, and an edge exists between two nodes if the corresponding bubbles are touching. Nodes corresponding to large and small bubbles are labeled accordingly. Identifying the state of a cluster amounts to establishing isomorphism between the cluster's graph and the graph of one of the 24 known stable configurations.

A simple method for distinguishing between the stable configurations for $n_L = n_S = 3$ begins as follows. Define three different types of edges – short edges, which are connections between two small bubbles; medium edges, between a small and a large bubble; and long edges, between two large bubbles.

Since there are three of each type of bubble in a cluster, a vertex can have between 0 and 2 long or short edges incident on it, and between 0 and 3 medium edges. We can assign a unique number (referred to hereafter as a "vertex code") to every valid combination of short, medium, and long edges that may be incident on a vertex. The set of assignments we use is listed in Table 2.1.

Any cluster can be associated with a sequence of six vertex codes. The sequences associated with the 24 stable states, using the labeling scheme described in Table 2.1, are listed in Table 2.2. It can be seen that all 24 sequences are distinct. Although it is possible that some of

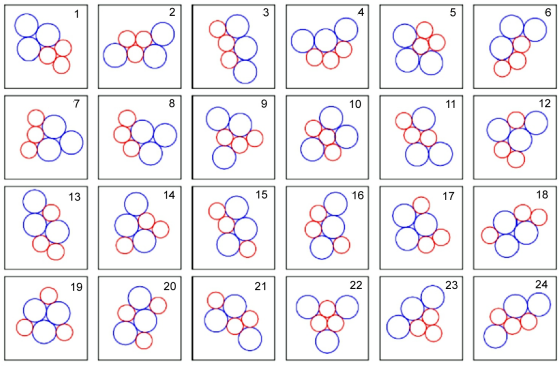

Figure 2.2: Enumeration of stable six-bubble states [1]

Vertex Code	Large	Medium	Small
1	0	0	1
2	0	0	2
4	0	1	0
5	0	1	1
6	0	1	2
8	0	2	0
9	0	2	1
10	0	2	2
12	0	3	0
13	0	3	1
14	0	3	2
16	1	0	0
20	1	1	0
24	1	2	0
28	1	3	0
32	2	0	0
36	2	1	0
40	2	2	0
44	2	3	0

Table 2.1: Vertex codes for valid combinations of edges

the 24 states share sequences with configurations not within the set of 24, we have never observed a cluster in which this is the case. Thus, in practice, computing the sequence for a cluster allows us to reliably identify it.

Implementing the method described above, bubbles are determined to be touching if the distance between their centers is equal to the sum of the bubbles' radii, within a certain

State Index	Sequence
1	2, 6, 10, 32, 36, 40
2	6, 8, 10, 10, 20, 24
3	5, 9, 10, 20, 24, 40
4	6, 9, 9, 20, 20, 44
5	5, 5, 14, 24, 24, 36
6	5, 6, 13, 20, 28, 36
7	5, 5, 10, 32, 40, 40
8	5, 6, 9, 32, 36, 44
9	5, 8, 10, 13, 20, 28
10	9, 9, 12, 14, 24, 24
11	5, 10, 12, 13, 20, 24
12	5, 9, 12, 20, 24, 44
13	5, 9, 12, 20, 28, 40
14	5, 8, 13, 20, 28, 40
15	5, 8, 13, 24, 24, 40
16	8, 9, 9, 20, 24, 44
17	5, 8, 9, 36, 36, 44
18	5, 8, 9, 36, 40, 40
19	8, 8, 8, 40, 40, 40
20	8, 8, 12, 24, 24, 44
21	8, 8, 9, 13, 24, 28
22	8, 8, 8, 10, 10, 10
23	2, 10, 10, 20, 20, 40
24	8, 9, 9, 10, 20, 28
25[1]	6, 6, 14, 24, 24, 36

[1] Variant of state 5 where each small bubble touches the other two.

Table 2.2: Sequences corresponding to cluster states

margin of error. Two different margins of error are used – a "loose" one equal to $\bar{r}_S$, the mean radius of small bubbles within the cluster; and a "strict" one equal to min($\bar{r}_S$,$\bar{r}_L$ - $\bar{r}_S$), where $\bar{r}_L$ is the mean radius of large bubbles within the cluster. Pairs of bubbles which fall within either margin are considered connected, but those which satisfy only the loose condition cause the cluster to be flagged for human review.

Vertex codes in the image processing script take form of three-digit numbers in base 4, where the first digit is the number of short edges incident on a vertex, the second digit the number

of medium edges, and the third digit the number of long edges. These are the numbers that (converted back into decimal) appear in Table 2.1.

When processing a cluster, all vertex codes are initially set to 0. Every pair of vertices is checked for a connection as described two paragraphs prior. Whenever a connection is found, the codes for the vertices at its endpoints are incremented by the appropriate amount – 1 for short edges, 4 for medium edges, and 16 for long edges. The six vertex codes associated with the cluster are complete once all pairs of bubbles have been checked for connections. The codes are then sorted from least to greatest and compared with the sequences in Table 2.2. This completes identification of the cluster.

For each cluster with $n_L = n_S = 3$ detected, the script makes a copy of the image and highlights the associated cluster. Copies that have been identified as one of the 25 states listed in Table 2.2 are saved to directories corresponding to their state. Those that have previously been flagged for review are stored in a subdirectory within their state's directory. Finally, clusters which have not been identified as corresponding to any known stable state are sent to a directory of their own.

2.3 Procedures

Before each run, a total of 2100 mL of deionized water, glycerol, and detergent are prepared. The amount of each component in an experimental variable. The water is measured and held in a beaker, while the glycerol and detergent are measured in graduated cylinders before being poured in with the water. The contents of the beaker are mixed with a stirring stick until all of the glycerol and detergent is dissolved.

If the trough still contains contents from the previous run, those are removed using the vacuum system described in the "Bubble Removal" subsection of Section 2.1. The tube

connected to the boat is detached and submerged, drawing liquid into the flask. The entire trough is emptied in this way. The contents of the flask are discarded.

The new solution is poured into the trough. Any foam produced is removed, again using the bubble removal tube. This time, the tube's opening is held at the surface, so as to remove bubbles while drawing in as little liquid as possible. Once all foam is removed, any liquid in the flask is poured back into the trough and the removal tube is reattached to the boat.

Next, the fan is turned on and the compressed gas cylinder's outlet valve is opened, beginning bubble production. The position of the two needles, the settings on their corresponding valves, and the setting on the cylinder's regulator are kept fixed between runs. The speed of the fan is adjusted so that bubbles take six seconds to clear the camera's field of view. The experiment is then left to run for several hours.

Once the run is complete, the pictures produced are given to the image processing script. The size thresholds for large and small bubbles are adjusted to correspond to the sizes of bubbles produced in the run. Once the image processing script has completed its work, all images flagged for human review are checked. Those with clusters that correspond to stable configurations are moved to their corresponding directories; the rest are discarded. Finally, the contents of each configuration's directory are counted and tabulated.

Chapter 3

Results

3.1 Data

The first four runs were conducted using the original mixing plate. Each run used a solution containing 5% detergent by volume. The amount of glycerol used was varied between runs to determine if the solution's viscosity had any effect on the distribution of bubble states. The first two runs used 4 % glycerol by volume, the third used 20%, and the fourth used 36%.

In each run, the number of clusters of each type are counted and recorded as described in Chapter 2. We considered 24 different configurations of six bubble states, enumerated as shown in figure 2.2.

A total of 271 six-bubble clusters were detected in the first two runs, 429 in the third run, and 163 in the fourth. The distribution of clusters over these 24 states are shown in figures 3.2 - 3.4.

Another six runs were conducted using the modified mixing plate described in section 2.1.4.

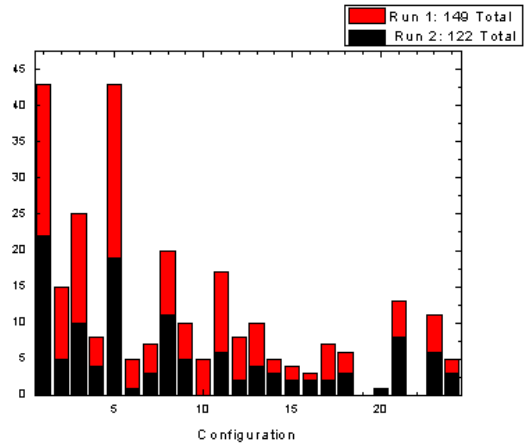

Figure 3.1: Runs 1 and 2: 4% Glycerol, 5% Detergent

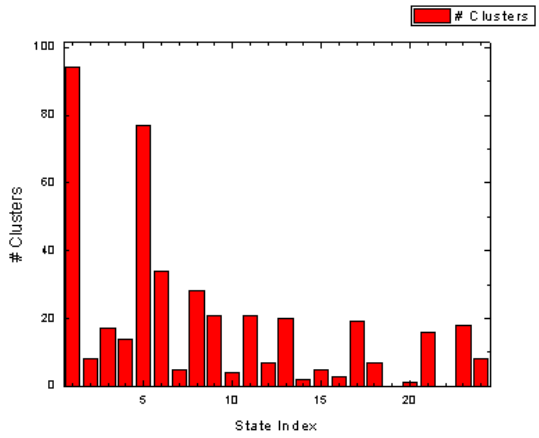

Figure 3.2: Run 3: 20% Glycerol, 5% Detergent

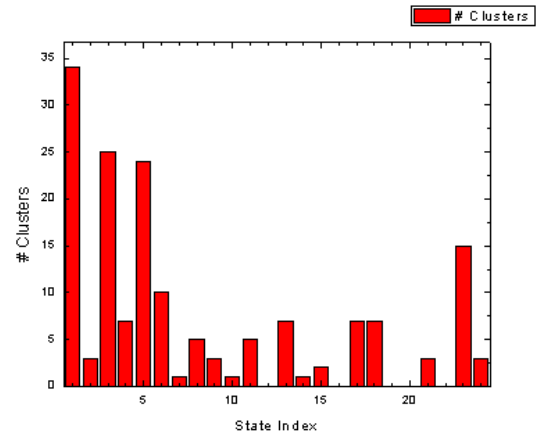

Figure 3.3: Run 4: 36% Glycerol, 5% Detergent

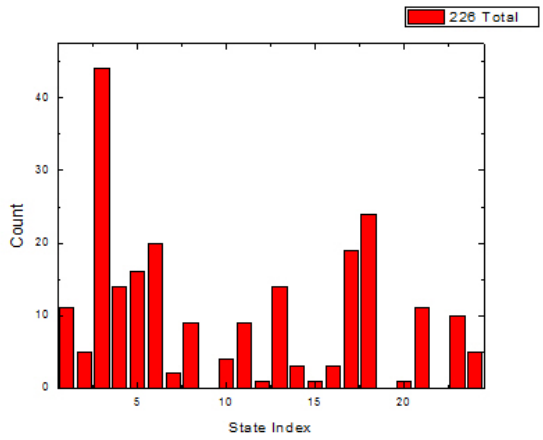

Figure 3.4: Run 5: Modified mixing plate, 4% Glycerol, 5% Detergent

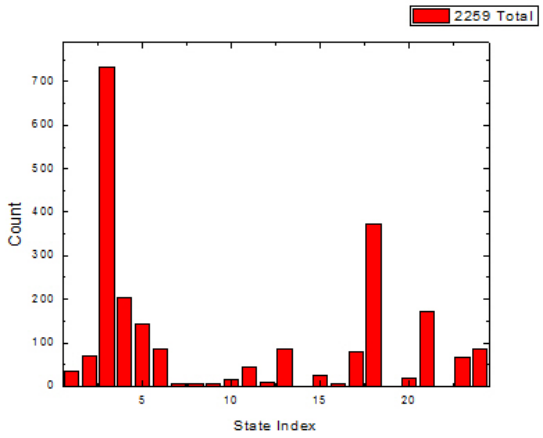

Figure 3.5: Run 6: Modified mixing plate, 20% Glycerol, 5% Detergent

Runs 5-7, like runs 1-4, used solutions containing 5% detergent, with glycerol concentrations of 4%, 20%, and 30% respectively.

Runs 8-10 used a solution containing 20% glycerol (the same amount as in run 6), with varying detergent concentrations of 10%, 15%, and 20% respectively.

Clusters containing six bubbles almost never emerge directly from the mixing plate. Rather, the vast majority of bubbles leave the mixing plate in clusters containing three bubbles or fewer. Isolated bubbles are the most common, followed by two and then three-bubble clusters. Upon separating from the plate, these bubbles are immediately attracted to each

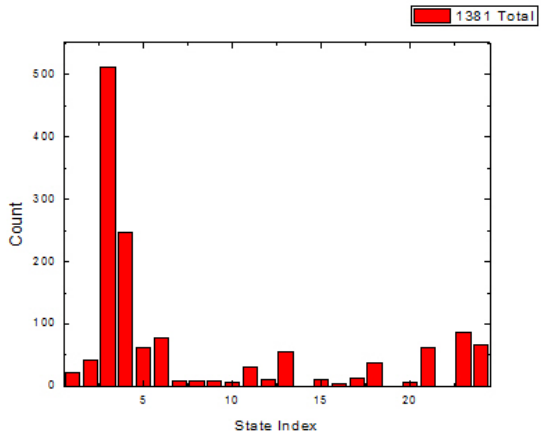

Figure 3.6: Run 7: Modified mixing plate, 30% Glycerol, 5% Detergent

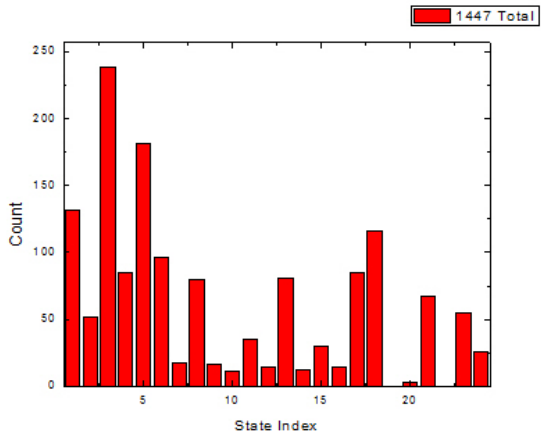

Figure 3.7: Run 8: Modified mixing plate, 20% Glycerol, 10% Detergent

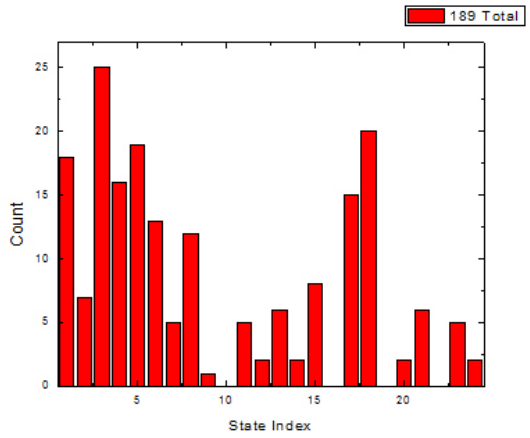

Figure 3.8: Run 9: Modified mixing plate, 20% Glycerol, 15% Detergent

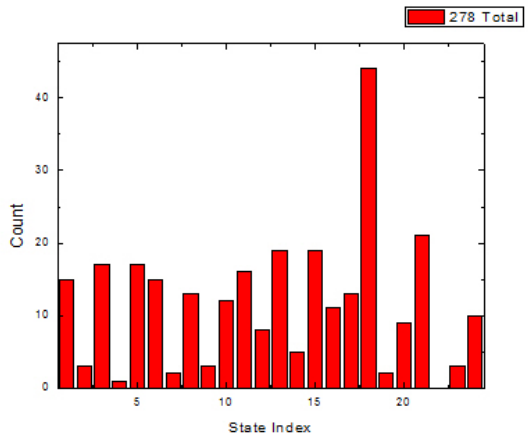

Figure 3.9: Run 9: Modified mixing plate, 20% Glycerol, 20% Detergent

other and combine to form larger clusters. Figure 3.11 lists out all stable clusters with five bubbles or fewer that contain no more than three bubbles of either type (large or small).

In order to study the process by which six-bubble clusters are formed, approximately 40 minutes of video were recorded from run 8. Events in which two or more smaller clusters merged to form one of the states in Figure 2.2 were identified by hand. We refer to smaller clusters as "precursor clusters".

Table 3.1 lists out the first 94 such events observed in the run. Events are described based on the initial and final bubble states involved (e.g. the event "$4.7 + 2.2 \rightarrow 6.3$" is one in which a four-bubble cluster in state 4.7 combines with a two-bubble cluster in state 2.2 to form a six-bubble cluster in state 6.3). These are sorted from most to least frequently occurring.

Note that Table 3.1 only tracks events that lead directly to the formation of a six-bubble cluster. Many of the initial clusters involved were themselves the result of mergers between even smaller clusters.

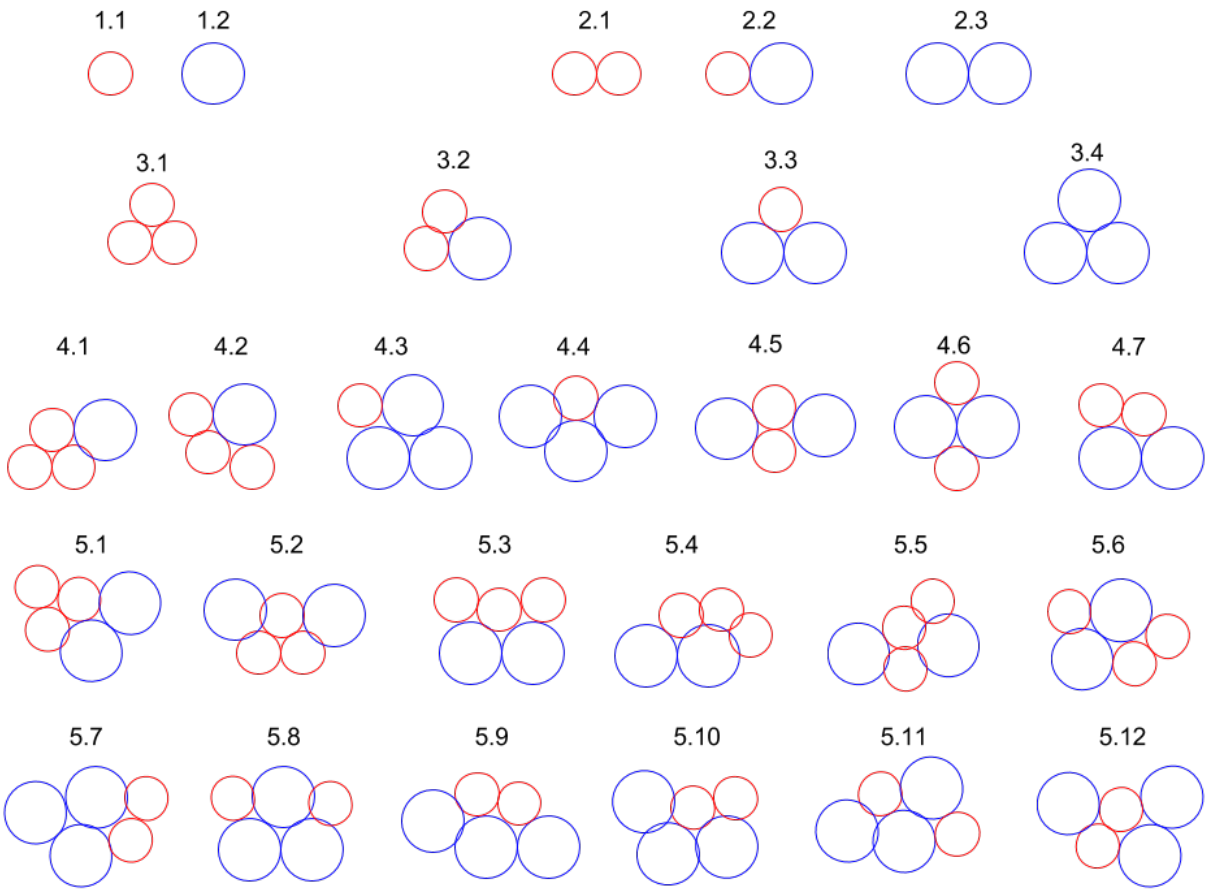

Figure 3.10: Precursor Clusters

Table 3.1: Formation of 6-bubble Clusters

Process	Count
$4.7 + 2.2 \rightarrow 6.3$	14
$3.3 + 3.2 \rightarrow 6.18$	12
$3.3 + 3.2 \rightarrow 6.3$	11
$4.7 + 2.2 \rightarrow 6.17$	7
$5.4 + 1.2 \rightarrow 6.4$	6
$4.7 + 2.2 \rightarrow 6.18$	5
$4.7 + 2.2 \rightarrow 6.4$	5
$3.3 + 2.2 \rightarrow 6.6$	4
$5.1 + 1.2 \rightarrow 6.23$	3
$4.7 + 2.2 \rightarrow 6.5$	3
$5.9 + 1.1 \rightarrow 6.23$	2
$5.8 + 1.1 \rightarrow 6.18$	2
$5.3 + 1.2 \rightarrow 6.3$	2
$5.9 + 1.1 \rightarrow 6.3$	1
$5.6 + 1.2 \rightarrow 6.21$	1
$5.6 + 1.2 \rightarrow 6.3$	1
$5.12 + 1.1 \rightarrow 6.21$	1
$5.10 + 1.1 \rightarrow 6.6$	1
$5.6 + 1.2 \rightarrow 6.14$	1
$4.7 + 2.2 \rightarrow 6.15$	1
$4.6 + 2.2 \rightarrow 6.21$	1
$4.5 + 2.2 \rightarrow 6.21$	1
$5.1 + 1.2 \rightarrow 6.1$	1
$3.3 + 3.2 \rightarrow 6.21$	1
$4.7 + 1.2 + 1.1 \rightarrow 6.6$	1
$4.7 + 1.2 + 1.1 \rightarrow 6.3$	1
$3.2 + 2.2 + 1.2 \rightarrow 6.18$	1
$3.2 + 2.2 + 1.1 \rightarrow 6.13$	1
$2.2 + 2.2 + 2.2 \rightarrow 6.20$	1
$3.3 + 1.2 + 1.1 + 1.1 \rightarrow 6.3$	1
$2.2 + 2.1 + 1.2 + 1.1 \rightarrow 6.5$	1

Table 3.2: Data File Locations

Run	Data Location
1	D:\4PercentDataRun1
2	D:\4PercentDataRun2
3	D:\20PercentDataRun
4	D:\36PercentDataRun
5	D:\4Glyc5Det
6	D:\20Glyc5Det
7	D:\30Glyc5Det
8	D:\20Glyc10Det
9	D:\20Glyc15Det
10	D:\20Glyc20Det

3.1.1 Data Location and Storage

All raw data from runs is stored on the computer named named "Vyre", in folders located directly within the Data (D:) drive. Each folder is created by the image-taking script at the start of a run. The names of these directories are listed in Table 3.2. Within each folder are the raw images taken from the corresponding run.

Additionally, each folder contains subfolders whose names are the numbers 0 through 25, created by the image processing script. In these subfolders are images corresponding to identified six-bubble clusters. Folders 1 through 25 correspond to the states listed in Table 2.2. Folder 0 contains all identified six bubble clusters that do not match any of these. Furthermore, each of these numbered folders contains a folder labeled "questionable", where clusters that meet only the script's "loose" criterion are stored. These images are reviewed by hand and deleted (in the case of false positives) or moved into numbered folders as appropriate.

The state counts that became the histograms above are stored in Origin files on Vyre in the directory "Documents\origin user files".

3.2 Other Notes and Observations

3.2.1 Bubble Production

The size and rate at which bubbles are produced is determined in part by the depth at which the needles are located below the air-solution interface, with deeper needles producing smaller bubbles at a slower rate. The needles tend to produce bubbles of uniform size if kept stationary. However, if a needle's opening is low enough to touch the bottom of the trough or high enough to touch the mixing plate, bubble sizes tend to be much more erratic.

Since large and small bubbles are produced at two different needles, the two types of bubbles are not evenly distributed below the mixing plate. Although some rearrangement occurs beneath the plate, bubbles of a given type tend to accumulate on the side of the plate corresponding to the needle from which they were produced. This happens even if the tips of the needles are placed close to each other.

Bubbles that detach from the mixing plate and rise to the surface are accelerated forward by airflow from the fan above. This means that bubbles move most slowly in the region directly in front of the plate and fastest a few centimeters forward. Beyond that point, bubbles slow down again as the air current from the fan weakens. Since faster moving bubbles must be spread farther apart along the direction of motion, average bubble spacing follows a similar pattern. Because of this, fan speed is a major factor in determining average cluster size. Faster fan speeds result in larger spacings between bubbles, which decrease the frequency of mergers, resulting in smaller cluster sizes.

Although the barriers in the trough help to regulate the motion of bubbles, flow still tends to be unpredictable. The angle and positioning of the fan often needs to be adjusted between runs, as does the location of the bubble removal nozzle. Generally, it is not possible to keep the flow straight. Rather, bubbles tend to veer to one side. Suction from the removal

tube is unable to noticeably influence the motion of bubbles further than approximately a centimeter from the nozzle. As a result, the nozzle must be placed directly in the path of the stream of bubbles.

3.2.2 Failure Modes

On all runs depicted in the histograms above, data was taken until it was no longer possible due to a failure in the experimental apparatus or software. Before the mixing plate was modified, bubbles would slip out the sides and stick to the needles. These bubbles attracted yet more bubbles, eventually filling the area completely and preventing new bubbles from leaving. The two added ridges on the modified mixing plate solved this problem and indeed runs using the modified mixing plate were able to run for much longer, as indicated by the larger event counts in some of the histograms above.

However, the system is still somewhat unstable and usually unable to run over the course of an entire night. There are three common ways in which a run might come to an end. First, and least often, an accumulation of bubbles may obstruct flow as before. This usually happens when the path of the bubble stream changes over the course of the run, causing bubbles to veer into the barriers. Alternatively, bubbles may miss the removal nozzle and either accumulate behind it, circulate around the trough and collect in other places, or attach to the bubble removal boat itself.

The second way a run might end is by depletion of the solution in the trough. Generally, this involves large amounts of fluid being drawn into the collection flask. Although a small amount of liquid is drawn away whenever a bubble is removed, this should occur at a consistent rate. However, the amount of time it takes for fluid in the trough to drop far enough to stop the apparatus from working (if it happens at all over the course of a run) varies wildly. Although I have never been present to observe this, I suspect that a small drop in fluid level causes

the bubble removal boat (which is supported at one end by tube leading to the flask) to tilt, which in turn causes the bubble removal nozzle to submerge more than it should. At this point, fluid is pulled directly from the trough into the flask.

The last way that runs end is due to the image taking software failing. This typically occurs after a few hours due to what Matlab refers to as a "low level graphics error". Forcing Matlab to use the software implementation of OpenGL seems to prevent his. However, doing so slows down the program considerably. Most problematically, this makes the timing of the computer-controlled fan inconsistent.

Chapter 4

Discussion

From the histograms in Chapter 3, it appears that changes in the width of the mixing plate have a much larger effect on the distribution of bubble states than changes in viscosity or surfactant concentration. Although no exact numerical data was recorded, there were a few failed runs in which a blockage made the effective mixing plate much narrower. In these runs, thousands of six bubble clusters in state 3 formed, compared to at most dozens of clusters in each of the 23 other states. Most of these state 3 clusters were produced by the merging of three state 2.2 clusters, which were themselves produced in large numbers.

Figure 4.1 depicts the tip of the mixing plate. As described in Chapter 3, large bubbles tend to accumulate on one side of the mixing plate (region a) and small bubbles on the other (region b). Different parts of the mixing plate will therefore tend to produce different types of clusters. Clusters that detach from the plate in regions a or b will tend to consist primarily of large or small bubbles respectively. However, those that emerge in region b will often contain a mix.

This explains the excess of state 3 clusters in the failed runs mentioned above. As the effective width of the mixing plate decreases, region b becomes larger relative to regions a

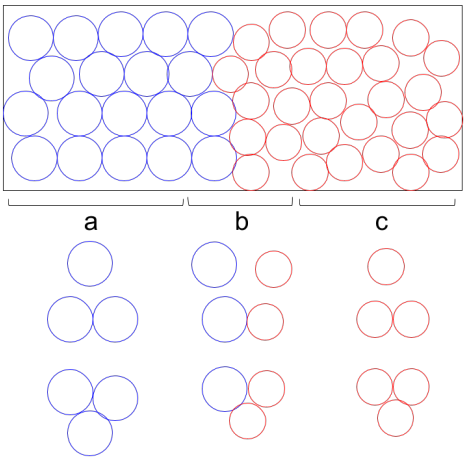

Figure 4.1: Mixing Plate Regions

and c, meaning that a larger portion of two-bubble clusters are created in state 2.2. This may also be the reason that runs conducted with the modified mixing plate produce peaks at state 3. Conversely, increasing the size of the mixing region increases the size of regions a and c relative to b. This should result in larger numbers of precursor clusters composed of a single type of bubble.

One might thus expect that a larger mixing plate would increase the frequency of clusters in which bubbles of the same type are grouped together. This 'groupedness' can be crudely measured by counting the number of contact points between bubbles of the same size in a cluster, and dividing this by the total number of contact points.

These ratios are plotted in Figure 4.2, which bears a striking resemblance to the histograms corresponding to runs using the original mixing plate. In fact, counts for each state in Figures 3.2 - 3.4 (runs 1 and 2 combined, 3, and 4) are strongly correlated with that state's ratio in Figure 4.2, with correlation coefficients R = 0.60, 0.62, and 0.67 respectively. A correlation is still present, albeit much weaker, for runs using the modified mixing plate (e.g. R = 0.33, 0.13, and 0.16 for runs 5, 6, and 7 respectively).

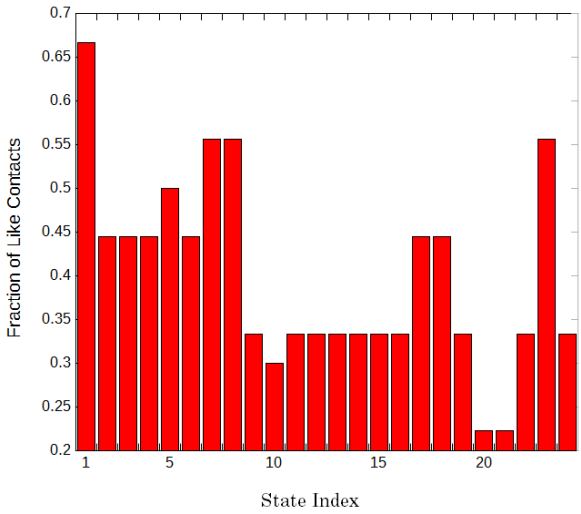

Figure 4.2: Contacts Between Like Bubbles

4.1 Future work

4.1.1 Mixing Plate

It seems reasonable that an even larger mixing plate should produce a higher degree of segregation between the two types of bubbles. A plastic mixing plate approximately 3 cm across was attempted. However, there were large regions of stationary bubbles, resulting in an effective width comparable to the original plate. It may be that bubbles are more prone to sticking to plastic than to glass. A working mixing plate with an adjustable width (i.e. one with movable ridges) would be useful for studying the effects of different cluster creation protocols.

4.1.2 Improving the Stability of the Experimental Setup

Fluid depletion might be prevented by adding a secondary reservoir as depicted in Figure 4.3. A sealed container partially filled with solution (perhaps another stoppered Erlenmeyer

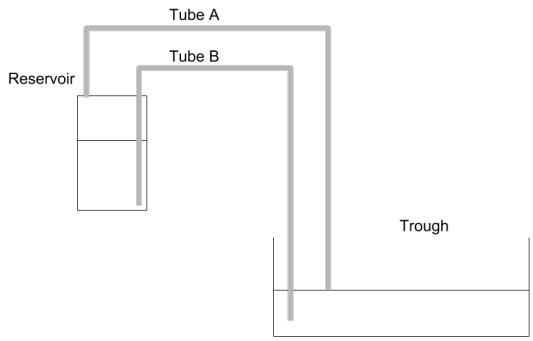

Figure 4.3: Solution Reservoir

Flask) is kept elevated above the trough. Tube A runs from the top of the container to a point in the trough at the desired fluid level. Tube B runs from near the bottom of the container to a point near the bottom of the trough. If the liquid in the trough falls below the opening of tube A, air is allowed to enter the container, which allows tube B to act as a siphon, pulling liquid from the container to the trough. When the liquid level rises above the opening of tube A, tube A is sealed again which in turn prevents flow through tube B.

Issues with Matlab's low-level graphics errors might be resolved by completely separating the part of the script responsible for image taking from the part responsible for controlling the fan. Ideally, this would involve running the two new scripts on two different computers.

4.1.3 General Cluster Recognition

The method used to identify clusters in Chapter 1 is not easily generalized to clusters with a different number of bubbles, or even to six bubble clusters. It turns out that matlab's bioinformatics toolbox contains a function, "graphisomorphism", specifically for detecting isomorphism between graphs. Although this function is unable to distinguish between dif-

ferent types of nodes (e.g. large and small bubbles), it is able to accept directed graphs.

It can quite easily be shown that the sizes of bubbles in a cluster can be fully encoded by edges in a directed graph. As before, let vertices and edges represent bubbles and contacts between bubbles respectively. When two bubbles of different size touch, this contact is represented by a directed edge pointing from the larger bubble to the smaller. When two bubbles of the same size touch, a bidirectional edge is used instead.

This convention means that a directed edge uniquely determines the size of the bubbles at both of its endpoints. Likewise, if the size at one endpoint of a bidirectional edge is known, the other is determined as well. Since clusters are, by definition, connected graphs, ever vertex can be reached from any other vertex by some chain of edges. Fixing any vertex thus determines the bubble size at every other vertex.

If a cluster contains both large and small bubbles, there must exist a location where a small bubble touches a large one, and thus there must exist at least one directed edge in the associated graph. Otherwise, all bubbles in the cluster are the same size. In either case, a graph always contains at least one vertex of known size. Combined with the previous paragraphs, this means that a directed graph with interchangeable edges maps uniquely maps to a cluster of large and small bubbles, and vise versa.

4.1.4 Video Tracking Script

Being able to track bubbles as they move in a video would be useful for studying the formation of larger clusters. A video tracking script might work as follows:

1. On each frame, locate every bubble using the methods described in Chapter 2. This may be made more robust by explicitly testing for roundness (the current script assumes that all features of the appropriate size are bubbles, which works most of the time).

A simple way to test for roundness is to divide the prospective bubble's area by its perimeter squared and check that this is equal to $1/4\pi$ to within a certain margin of error. Record the size and position of every bubble in an array.

2. For each bubble, find the nearest bubble (as determined by the distance between their centers) of the same size in the previous frame and check if this distance falls below a small threshold value.

   (a) If it does not, the bubble is considered "new" and it is assigned a unique identification number. Identification numbers are assigned in ascending order. Numbers corresponding to bubbles that no longer exist are not reused.

   (b) If it does, the new bubble is assumed to be the same as the bubble from the previous frame and is assigned the old bubble's identification number.

3. Find all contacts between bubbles. The current image processing script determines if two bubbles are touching by checking if the distance between the two centers is approximately equal to the sum of their radii. A better way to do this would be to check all (binarized) pixels on a line between the centers of the two bubbles. Starting from one center, move along that line and note the color of the pixel at each point. Count the number of times this changes. If there are exactly two changes (i.e. moving along the line brings you from white to black to white again), then the two bubbles share a wall and thus are touching.

4. From this list of contacts, create an (undirected) adjacency matrix for the entire frame. The rows and columns are sorted by identification number, from least to greatest. Rows and columns corresponding to bubbles that existed in the previous frame but not in the present are maintained as temporary padding (filled with zeros as nonexistent bubbles cannot connect to anything). The convention that identification numbers are assigned in ascending order means that any new rows and columns are added to the bottom

and right of the matrix. For comparison purposes, the matrix from the previous frame is also padded to the bottom and right by the same number of rows and columns.

5. Find all events (merging, splitting, or rearrangement of a cluster, or appearance or disappearance of a bubble) occurring between the previous frame and this. If the adjacency matrix and ID number list is the same as the matrix and list from the previous frame, then no event has occurred.

   (a) Any change in the matrix consists of places where a zero has changed to a one or vise versa. That is, places where contact is made or broken between a pair of bubbles. The following information is recorded for each event:

      i. The type of event (rearrangement, merge, split, appearance or disappearance)

      ii. The specific bubble or pair of bubbles that define the event.

      iii. A list of all ID numbers that were part of either bubble's connected component in the frame before the event. A corresponding list of bubble sizes.

      iv. (for rearrangements, merges, and splits) The submatrix of that frame's adjacency matrix corresponding to the list of ID numbers above.

      v. (for rearrangements, merges, and splits) A similar list and submatrix from after the event.

   (b) If a zero has changed to a one, there is now contact between two bubbles that were not touching before. If the two bubbles were not part of the same connected component before the event, then two clusters have merged. Else, the two bubbles were part of a rearrangement within the same cluster

   (c) Conversely, if a one changes to a zero, contact has been broken between a pair of bubbles. If the two bubbles are still part of the same connected component after the event, then a rearrangement has occurred. Else, two clusters have split apart (this will probably never happen).

(d) Bubble appearance and disappearance events are the simplest, corresponding to additions or subtractions from the list of ID numbers.

It is this set of of events occurring in each frame that is permanently recorded. Optionally, the position of each bubble in the frame may be recorded as well, which would allow one to trace each bubble's path over its lifetime. From the event data, the general cluster recognition method described above can be used to identify all clusters involved.

Bibliography

[1] K. Zhang, C. Kuo, N. See, C. OHern, and M. Dennin, Unpublished.

[2] R. Jones, *Soft Condensed Matter* ((Oxford Press, New York, NY, 2002)).

[3] J. Selinger, *Introduction to the Theory of Soft Matter* (Springer, New York, NY, 2016).

[4] D. Weaire and N. Rivier, Contemporary Physics **25:1**, 59 (1984).

[5] M. L. Falk and J. Langer, Physical Review E **57**, 7192 (1998).

[6] I. Cantat, S. Cohen-Addad, F. Elias, F. Graner, R. Hohler, O. Pitois, and A. S.-J. F. Rouyer, *Foams: Structure and Dynamics* (Oxford University Press,Oxford, England, 2013).

[7] S. Hutzler and D.Weaire, *The Physics of Foams* (Oxford University Press,Oxford, England, 1999).

[8] J. Lauridsen, M. Twardos, and M. Dennin, Physical Review Letters **89**, 9 (2002).

[9] E. Pratt and M. Dennin, Physical Review E **67**, 051402 (2003).

[10] M. Dennin and C. M. Knobler, Physical Review Letters **78**, 2485 (1997).

[11] D. J. Durian, Physical Review Letters **75**, 4780 (1995).

[12] A. A. el Kader and J. C. Earnshaw, Physical Review Letters **82**, 2610 (1999).

[13] T. Okuzono and K. Kawasaki, Physical Review E **51**, 1246 (1995).

[14] Y. Jiang, P. J. Swart, A. Saxena, M. Asipauskas, and J. Glazier, Physical Review E **59**, 5819 (1999).

[15] S. Hutzler, D. Weaire, and F. Bolton, Philosophical Magazine Part B **71:3**, 277 (1995).

[16] D. DUrian, Physical Review E **55**, 1739 (1997).

[17] M. Lundberg, K. Krishan, N. Xu, C. S. OHern, , and M. Dennin, Physical Review E **77**, 041505 (2008).

Appendix A

Image Processing Code

```
function improcessing6bubble_basic(data_dir,dsize,sizemin,t_s_num,
    t_b_num)
%improcessing6bubble_basic('D:\newsetup_test6\New folder (2)
    ',450,100,3,3)
    configurations = [
                        2,6,10,32,36,40;
                        6,8,10,10,20,24;
                        5,9,10,20,24,40;
                        6,9,9,20,20,44;
                        5,5,14,24,24,36;
                        5,6,13,20,28,36;
                        5,5,10,32,40,40;
                        5,6,9,32,36,44;
                        5,8,10,13,20,28;
                        9,9,12,14,24,24;
                        5,10,12,13,20,24;
```

```
                    5,9,12,20,24,44;
                    5,9,12,20,28,40;
                    5,8,13,20,28,40;
                    5,8,13,24,24,40;
                    8,9,9,20,24,44;
                    5,8,9,36,36,44;
                    5,8,9,36,40,40;
                    8,8,8,40,40,40;
                    8,8,12,24,24,44;
                    8,8,9,13,24,28;
                    8,8,8,10,10,10;
                    2,10,10,20,20,40;
                    8,9,9,10,20,28;
                    6,6,14,24,24,36;];
    for n = 0:size(configurations);
                mkdir(data_dir, num2str(n));
                mkdir(strcat(data_dir,'\',num2str(n)), '
                    questionable');

    end
files = dir(fullfile(data_dir,'6bubble*'));
num_sets = size(files, 1);
disp(num_sets);
%num_sets=100;
outputcount=1;
outputcountclustersize=1;
```

```
h = waitbar(0,'Please wait...');
hw=findobj(h,'Type','Patch');
set(hw,'EdgeColor',[0 0 0],'FaceColor',[0 1 0]) % changes the
    color to green
s_count=1;
b_count=1;

for kk = 1:num_sets
    waitbar( kk/ num_sets);
    img = fullfile(data_dir,files(kk).name);
    %b0=imread('6bubbleimage_05272015_0001.bmp');
    disp(img);
    b0=imread(img);
    if kk==1;
        imshow(b0);
        text(1, 1, ['Crop bubbles for ratio detection:'], '
            FontSize', 24, 'Color', 'Y','VerticalAlignment','top');
        [b0, rect_0] = imcrop;
%           b1=im2double(b0);
%           I3 = imadjust(b1);
%           level = graythresh(I3);
    else
        b0 = imcrop(b0 , rect_0);
    end
    b1=im2double(b0);
    %I3 = imadjust(b1);
    level = graythresh(b1);
```

```
%level= 0.8;
bw = im2bw(b1,level);
%-----------------------------------------------------
%se1 = strel('disk',8);
%erodedBW = imerode(bw,se1);
%dilateBW = imdilate(erodedBW,se2);
%-----------------------------------------------------

 inv_bw=~bw;
 se = strel('disk',3);
 inv_bw = imdilate(inv_bw,se);
 filled_bw = imfill(inv_bw,'holes');
 filled_bw = imerode(filled_bw,se);
 filled_bw2 = bwareaopen(filled_bw, sizemin);

 CC = bwconncomp(filled_bw2);
 STATS = regionprops(CC, 'basic');
 num_cluster=length(STATS);
 h_fig=imshow(b0);
 [b0y b0x ]=size(b0);
 hold on

 for j=1:num_cluster;
%          if num_cluster>10;
%              num_cluster
%          end
       if num_cluster<10 ...
```

```
&& STATS(j,1).Area < 20000 ...
&& STATS(j,1).BoundingBox(1,1) > 1 ...
&& STATS(j,1).BoundingBox(1,2) > 1 ...
&& STATS(j,1).BoundingBox(1,1)+0.5+ STATS(j,1).
    BoundingBox(1,3) < b0x ...
&& STATS(j,1).BoundingBox(1,2)+0.5+ STATS(j,1).
    BoundingBox(1,4) < b0y ...

   %v.1 use the same global loccal binary image;
       separaes in
   %08142015
   %[croped_inv_bw, rect] = imcrop(inv_bw, STATS(j,1).
       BoundingBox);
   [croped_b0, rect] = imcrop(b0, STATS(j,1).
       BoundingBox);
   level = graythresh(croped_b0);
   croped_bw = im2bw(croped_b0,level*1.3);
   croped_inv_bw =~ croped_bw;

   %more code can be put here to fix the bubble
       boundary breaking
   %issue.
   croped_inv_bw = bwmorph(croped_inv_bw,'bridge');

   croped_inv_bw=imfill(croped_inv_bw,[1 1]);
   croped_inv_bw=imfill(croped_inv_bw,[rect(4) 1]);
```

```
croped_inv_bw=imfill(croped_inv_bw,[rect(4) rect(3)
    ]);
croped_inv_bw=imfill(croped_inv_bw,[1 rect(3)]);
croped_bw=~croped_inv_bw;

croped_bw2 = bwareaopen(croped_bw, sizemin);
CC_temp = bwconncomp(croped_bw2);
i=0;
Bcount_0=CC_temp.NumObjects;
Bcount_1=Bcount_0;
  if Bcount_0 >1 ;
      while Bcount_0==Bcount_1
          i=i+1;
          se = strel('disk',i);
          croped_bw2expand = imdilate(croped_bw2,se)
              ;
          CC_temp = bwconncomp(croped_bw2expand);
          Bcount_1=CC_temp.NumObjects;
      end
  end
croped_bw3=croped_bw2;
  if i>1;
      se = strel('disk',i-1);
      croped_bw3 = imdilate(croped_bw2,se);
  end
CC2 = bwconncomp(croped_bw3);
STATS2 = regionprops(CC2, 'basic');
```

```
    bubble_num=length(STATS2);
    area_index=zeros(bubble_num,1);
        for i=1:bubble_num
            area_index(i,1)=STATS2(i,1).Area;
        end
%s_area_index=sort(area_index);
 b_num=0;
 s_num=0;
 s_list=find(area_index<dsize);
 b_list=find(area_index>dsize & area_index<2500);

 s_num=length(s_list);
 b_num=length(b_list);

 clustersize_list(outputcountclustersize,1)=s_num;
 clustersize_list(outputcountclustersize,2)=b_num;
 outputcountclustersize=outputcountclustersize+1;

 s_x_y=zeros(s_num,2);
      for i=1:s_num
      s_x_y(i,1)=STATS2(s_list(i)).Centroid(1);
      s_x_y(i,2)=STATS2(s_list(i)).Centroid(2);
      plot(s_x_y(i,1)+rect(1),s_x_y(i,2)+rect(2),'ro','
          MarkerSize',3,'MarkerFaceColor','r')
      end
```

```
b_x_y=zeros(b_num,2);
    for i=1:b_num
    b_x_y(i,1)=STATS2(b_list(i)).Centroid(1);
    b_x_y(i,2)=STATS2(b_list(i)).Centroid(2);
    plot(b_x_y(i,1)+rect(1),b_x_y(i,2)+rect(2),'bo','
        MarkerSize',3,'MarkerFaceColor','b')
    end

    if s_num==t_s_num && b_num==t_b_num;

        for k=1:s_num
            s_area(k)=STATS2(s_list(k)).Area;
            s_b_diameter_list(s_count,1)=2*sqrt(s_area
                (k)/pi);
            s_count=s_count+1;
        end
        s_mean_area=mean(s_area);
        s_diameter=2*sqrt(s_mean_area/pi);
        s_3_area=polyarea(s_x_y(:,1),s_x_y(:,2));
        n_s_3_area=(polyarea(s_x_y(:,1),s_x_y(:,2)))/(
            s_diameter^2);

        for k=1:b_num
            b_area(k)=STATS2(b_list(k)).Area;
            s_b_diameter_list(b_count,2)=2*sqrt(b_area
                (k)/pi);
            b_count=b_count+1;
```

```
        end
        b_mean_area=mean(b_area);
        b_diameter=2*sqrt(b_mean_area/pi);
        b_3_area=polyarea(b_x_y(:,1),b_x_y(:,2));
        n_b_3_area=(polyarea(b_x_y(:,1),b_x_y(:,2)))/(
            s_diameter^2);

    end
```

```
rectangle('Position',rect,'EdgeColor','g');
%%text(rect(1), rect(2)+rect(4), num2str(j), 'FontSize
    ', 16, 'Color', 'k','VerticalAlignment','top');
text(rect(1), rect(2), num2str(s_num), 'FontSize', 16,
     'Color', 'r','VerticalAlignment','bottom');
text(rect(1)+rect(3)/2, rect(2), num2str(b_num), '
    FontSize', 16, 'Color', 'b','VerticalAlignment','
    bottom');
```

```
%
```

```
if b_num==t_b_num && s_num==t_s_num;
    %text(rect(1), rect(2)+rect(4), ['A_S+A_L '
        num2str(n_s_3_area+n_b_3_area)], 'FontSize',
        16, 'Color', 'y','VerticalAlignment','top');
    d_s_b(1,:)=mean(s_x_y,1);
    d_s_b(2,:)=mean(b_x_y,1);

    if b_num == 3 && s_num == 3;
    all_x_y = cat(1, s_x_y, b_x_y);
    alldist = zeros(6);
    connections = [0,0,0,0,0,0];
    errormarginloose = s_diameter/2;
    errormargin = min([ s_diameter/2, (b_diameter -
       s_diameter)/2]) ;
    errorflag = 0;
    numfound = 0;
    foundconfig = 0;



    disp('yes');
    for i = 2:6;
        for j = 1:(i-1);
            alldist(i,j)= sqrt((all_x_y(i,1) - all_x_y
                (j,1))^2 + (all_x_y(i,2) - all_x_y(j,2)
                )^2)
```

```
if i <= 3;
    if abs(alldist(i,j) - s_diameter) <
       errormarginloose;
        connections(i) = connections(i) +
           1;
        connections(j) = connections(j) +
           1;
        if mod(connections(i),4) == 0 ||
           abs(alldist(i,j) - s_diameter)
           > errormargin ;
             errorflag = 1
        end
    end
elseif j <= 3;
    if abs(alldist(i,j) - (b_diameter +
       s_diameter)/2) < errormarginloose;
        connections(i) = connections(i) +
           4;
        connections(j) = connections(j) +
           4;
        if mod(connections(i),16) == 0 ||
           abs(alldist(i,j) - (b_diameter
           + s_diameter)/2) > errormargin;
             errorflag = 1
        end
    end
else
```

```
                if abs(alldist(i,j) - b_diameter) <
                    errormarginloose;
                  connections(i) = connections(i) +
                      16;
                  connections(j) = connections(j) +
                      16;
                  if mod(connections(i),64) == 0 ||
                      abs(alldist(i,j) - b_diameter)
                      > errormargin;
                        errorflag = 1
                  end
              end
          end


    end
end
connections = sort(connections);
disp(connections);
disp(isequal(connections, configurations(16,:)));
numconfigs = size(configurations)


    for k = 1:numconfigs;
        if isequal(configurations(k,:),connections
            )
        disp(k);
        numfound = numfound +1;
        foundconfig = k;
```

```
                end
            end
            if numfound == 1 && errorflag == 0;
           text(rect(1)+rect(3), rect(2), num2str(
              foundconfig), 'FontSize', 16, 'Color', 'g','
              VerticalAlignment','bottom');
            else
                text(rect(1)+rect(3), rect(2), strcat(
                   num2str(foundconfig),'?'), 'FontSize',
                   16, 'Color', 'g','VerticalAlignment','
                   bottom');
            end
        end
%
  ------------------------------------------------------------

%
  ------------------------------------------------------------

%                  eigen_state = eig(contact_table);
%                  text(20, 200, [num2str(eigen_state)], 'FontSize
  ', 8, 'Color', 'g','VerticalAlignment','top');
%
```

```
%-------------------------------------------------------------------
%text(rect(1), rect(2)+rect(4)+50, ['d_{sL}= '
    num2str(pdist(d_s_b)/s_diameter)], 'FontSize',
    16, 'Color', 'y','VerticalAlignment','top');
%text(rect(1), rect(2)+rect(4)+50, ['d_state5 '
    num2str(s5_distance/s_true_d)], 'FontSize', 16,
     'Color', 'y','VerticalAlignment','top');
%text(rect(1), rect(2)+rect(4)+100, ['d_L/d_s '
    num2str(b_diameter/s_diameter)], 'FontSize',
    16, 'Color', 'y','VerticalAlignment','top');
%text(rect(1), rect(2)+rect(4)+150, ['mr^2 '
    num2str(all_mr_qurt/s_diameter*s_diameter)], '
    FontSize', 16, 'Color', 'y','VerticalAlignment
    ','top');
outputA_SA_L__dsl(outputcount,1)=s_3_area+b_3_area
    ;
outputA_SA_L__dsl(outputcount,2)=pdist(d_s_b);
outputA_SA_L__dsl(outputcount,3)=n_s_3_area+
    n_b_3_area;
outputA_SA_L__dsl(outputcount,4)=pdist(d_s_b)/
    s_diameter;
outputA_SA_L__dsl(outputcount,5)=b_diameter/
    s_diameter;

%output_mr_qurt(outputcount,1)=all_mr_qurt/
    s_diameter*s_diameter;
```

```
%------------------------for state
    5-----------------
%output_d_state5(outputcount,1)=b_true_d/s_true_d;
%output_d_state5(outputcount,2)=s5_distance/
    s_true_d;
%
    ------------------------------------------------------------------------

                if errorflag ==0;
                pathjpg = strcat(data_dir, '\',num2str(foundconfig
                    ),'\','imageindex',num2str(outputcount),'_',
                    num2str(files(kk).name(22:26)),'.jpg');
                else
                    pathjpg = strcat(data_dir, '\',num2str(
                        foundconfig),'\questionable\','imageindex',
                        num2str(outputcount),'_',num2str(files(kk).
                        name(22:26)),'.jpg');
                end
                print('-f1', '-djpeg', '-r300', pathjpg);
                outputcount=outputcount+1;
                %pause(1);
            end
        end
    end
```

```
    %reply = input('Do you want more? Y/N [Y]: ', 's');
    %-----------------------------------
    %cleared_bw=imclearborder(filled_bw);

    %BW2 = bwareaopen(BW, 50); remove objects smaller than 50
        pixels


    %
        ------------------------------------------------------------

    % [centers, radii, metric] = imfindcircles((inv_bw,[15 30]);
    % centersStrong5 = centers(1:5,:);
    % radiiStrong5 = radii(1:5);
    % metricStrong5 = metric(1:5);
    hold off
end
pathone = strcat(data_dir, '\','SL_area__dsl','.dat');
pathtwo = strcat(data_dir, '\','s_b_diameter','.dat');
paththree = strcat(data_dir, '\','cluster_size','.dat');
%pathfour = strcat(data_dir, '\','mofinertia','.dat');
%pathfive = strcat(data_dir, '\','d_state5','.dat');

csvwrite(pathone,outputA_SA_L__dsl);
csvwrite(pathtwo,s_b_diameter_list);
csvwrite(paththree,clustersize_list);
```

```
%csvwrite(pathfour,output_mr_qurt);
%csvwrite(pathfive,output_d_state5);


%%%
%[croped_I3, rect] = imcrop(I3, STATS(j,1).BoundingBox);
```