Method and Evaluation for Minimization of Mechanical Effects from Impact Velocity for the Optimization of Freezing Quality of Metal Mirror Impact Freezers

A Thesis Presented

by

Nector Ritzakis

to

The Department of Mechanical Engineering
for the degree of

Master of Science

in

Mechanical Engineering

Northeastern University
Boston, Massachusetts

December 2011
ABSTRACT

Introduction: Preparation of samples for electron microscopy typically balances the need to “fix” the sample against the necessary generation of artifacts which influence the interpretation of observed structures. Ultrarapid freezing by impact of a specimen with a thermally conductive metal surface, cooled to at least liquid nitrogen (LN) temperatures, has long been thought of as the gold standard for minimizing cryofixation artifact. However, there has been a long standing debate about whether or not impact with the metal surface introduces a mechanical deformation to the sample before it freezes. Though it has been argued that the mechanical deformation wave is preceded over short distances by the freezing front, no quantitative investigation has been attempted to verify this view. The principal objective of this study was to determine if impact freezing is likely to create distortion artifact. Prior to this work, there was no method, device or fixation assay which permitted an objective, controlled study of the effect of impact velocity on the fixation of a specimen. To perform the investigation (1) a recently constructed “touch” freezer was modified; (2) a dual-phase velocity profile was developed for controlling the approach of the sample to the cooled metal mirror; (3) a new assay was introduced, capable of quantifying mechanical deformation effects produced during sample impact with a cooled copper mirror; and (4) the hypothesis that mechanical distortion of the sample by impact freezing is proportional to impact velocity was tested. Methods: The distortion assay was produced using a polystyrene microsphere-infused Laponite clay-based, shear-thinning gel (90% water) which was transparent enough to be scanned with an inverted light microscope to determine pre- and post-impact bead position. Effect of impact velocity on the gel structure were examined by impact freezing the gel using a dual-phase velocity profile which initially accelerates the sample to 4.5 m s\(^{-1}\) to avoid premature sample cooling, and decelerates it to 2.5, 5, 10, or 30 cm s\(^{-1}\)for the final 0.1 seconds before making contact with the mirror. Samples were z-scanned using differential interference contrast microscopy pre- and post-impact freezing. Three dimensional point clouds of the microspheres were generated from the images and analyzed using a series of custom written MATLAB algorithms. The analysis yields the strain of the 3-dimensional point cloud of the beads which reflects the effect of the impact of the gel with the mirror. Results: Strain comparisons showed an increasing trend in sample distortion with increasing impact velocity. Specifically, the 2.5, 5, and 10 cm s\(^{-1}\) impact velocity strains were statistically different (\(P = 0.011, 0.014, \) and 0.022) from those of the 30 cm s\(^{-1}\) impact velocity. Attempts to use a standard commercial slam freezer (Delaware Diamond Knives’ Cryogun: impact velocity 2.3 m s\(^{-1}\)) produced distortions so great that the point clouds could not be correlated. Conclusions: The results from this study support the hypothesis approach velocity influences sample distortion in impact freezing. Further, we have demonstrated the efficacy of a new method (dual-phase velocity profile) for impact freezing that minimizes sample distortion incurred during contact with a cooled metal surface. Finally, we have generated a new assay capable of evaluating other impact freezing designs based on the impact distortion they introduce.
ACKNOWLEDGEMENTS

First, I would like to thank Dr. Jeffrey Ruberti for allowing me to pursue my master’s degree in his laboratory. This project began for me as a senior for my undergraduate project. I was able to take a prototype my team constructed in 2008, and, under his guidance, transform it into a complete and robust system. The whole journey offered me a learning experience that I doubt will ever be matched.

I would also like to thank the rest of Extracellular Matrix Engineering Laboratory for all their help and support. Dr. Graham Tilburey, thank you for your help in designing the setup for viewing the sample on the microscope. Jeff Paten, thank you for constantly making me think about what I was doing and why I was doing it. Ramin Zareian, thank you for your initial help with the microscope. Finally, thank you to Mehdi Abedi for his help with using the high speed camera.

This would not be a complete acknowledgement section if I did not include certain people that were crucial to my success outside of the laboratory setting. Above all, I must thank my parents. Words cannot describe the love and gratitude I have for you and what you have done for me. Σας αγαπώ! Finally, thank you to all my friends and family who supported me from near and far. You will always be in my heart.
# TABLE OF CONTENTS

1 INTRODUCTION ................................................................................................................ 10

1.1 CRYOPRESERVATION ............................................................................................... 10

1.2 IMPACT FREEZING ................................................................................................. 13

1.2.1 Background ............................................................................................................. 13
1.2.2 Metals and Cryogens Used ..................................................................................... 13
1.2.3 Impact Velocity ....................................................................................................... 14
1.2.4 Bounce .................................................................................................................... 14
1.2.5 Advantages of Impact Freezing .............................................................................. 15

1.3 IMPACT FREEZING OBSTACLES ............................................................................. 16

1.3.1 Sample Travel Distance .......................................................................................... 16
1.3.2 Impact Distortion .................................................................................................... 17

2 TOUCH FREEZER .............................................................................................................. 18

2.1 BACKGROUND: PHASE 1 TOUCH FREEZER DESIGN ......................................... 18

2.2 PHASE 1 COMPONENTS AND METHODS INCLUDED IN PHASE 2 DESIGN ... 19

2.2.1 Linear Motor ........................................................................................................... 19
2.2.2 Wide Beam Laser .................................................................................................... 21
2.2.3 Cooling Materials .................................................................................................... 21
2.2.4 Operation ................................................................................................................. 22
2.2.5 Control Hardware and Software ............................................................................. 24
2.2.6 Discussion ............................................................................................................... 24

2.3 MODIFIED DESIGN (PHASE 2) ................................................................................. 25

2.3.1 Overview ............................................................................................................... 25
2.3.2 Mirror Module ........................................................................................................ 27
2.3.3 Support Braces ....................................................................................................... 29
2.3.4 Hard Stop ................................................................................................................ 30
2.3.5 Main Frame ........................................................................................................... 33
2.3.6 Switch Panel .......................................................................................................... 34

3 DUAL-PHASE VELOCITY PROFILE ............................................................................... 36

3.1 OVERVIEW ................................................................................................................. 36

3.2 PROGRAMMING THE DUAL-PHASE VELOCITY PROFILE .................................. 37
LIST OF FIGURES

Figure 1: Original “touch” freezer design [77]. ................................................................. 19

Figure 2: LinMot magnetic induction linear motor (www.linmot-usa.com). ...................... 20

Figure 3: Keyence wide beam laser sensor (www.keyence.com). ........................................ 21

Figure 4: Description of how touch freezing sequence distance, X_{TF}, is measured through the combination of a linear motor and laser sensor in the original design [77]. This procedure was kept for the phase 2 design. The measurements are calculated on the active mirror. See text for full description. ........................................................................................................................................ 23

Figure 5: Modified touch freezing device with (a) a simplified main frame, (b) a paired motor/mirror module assembly, and (c) wide beam laser assembly.................................................. 25

Figure 6: Side view of modified touch freezer device. Two mounting plates (a) are fastened to both sides of the motor flange, which holds the motor stator and slider. The opposite ends of the mounting plates are fastened around an angle bracket which is mounted onto the main frame. The angle bracket keeps the angles the motor at 60 degrees from the horizontal. A central brace (b) is fastened onto the back face of the motor flange. Two pillow block linear ball bearings (c) are fastened near the ends of the brace. These allow for a precision shaft (d) to be aligned parallel to the motor slider. The precision shaft holds the new mirror holder. It is comprised of a four armed mirror holder (e) which is fastened to an aluminum shaft (f). The aluminum shaft is connected to the precision shaft via a ceramic standoff (g). A gear locking mechanism (h) is fastened to the opposite end of the central brace for mirror interchange. A hard stop (i) is fastened onto the mounting plates as a safety precaution to prevent the motor slider from advancing further than programmed. A wide beam laser emitter and receiver (j) are mounted onto two metal plates which are fastened to the main frame. Two stabilization braces (Section 2.3.3) are mounted to the main frame and fastened onto the central brace. ........................................................... 26

Figure 7: Four-armed mirror module. Arrow shows set screw location for securing the copper mirror. ........................................................................................................................................ 27

Figure 8: Description of order of events (a-e) for rotating between mirrors using the gear locking mechanism for modified mirror module. ......................................................................................... 29

Figure 9: Support brackets used to stabilize the central brace. Arrows indicate orientation of brackets. ........................................................................................................................................... 30

Figure 10: Safety hard stop design for modified touch freezer. See text for detailed description. 32

Figure 11: Hard stop. (a) Position of the hard stop fastener as the slider approaches the mirror. (b) Position of the hard stop fastener when the motor has reached the mirror and stopped. Note the gap between the metallic and neoprene washers. (c) Worst case scenario. The motor travels past calculated distance. The two washers make contact and prevent the motor from continuing its motion. ................................................................................................................................. 33

Figure 12: Main frame of modified touch freezer. .................................................................. 34
Figure 13: Custom made switch panel for controlling motor sequence commands. ........................... 36

Figure 14: (a) Motor accuracy verification test setup. (b) Caliper probe backed off to avoid impact from motor trajectory. (c) Caliper probe reapplied to the motor slider to measure position where it stopped. ........................................................................................................................... 39

Figure 15: Motor accuracy result plots for various combinations of first phase accelerations (Acc) and decelerations (Dec). (a) Acc = 100 m/s², Dec = 100 m/s² (b) Acc = 100 m/s², Dec = 150 m/s², (c) Acc = 150 m/s², Dec = 100 m/s², (d) Acc = 150 m/s², Dec = 150 m/s². .......................... 40

Figure 16: Dual-phase velocity profile evaluation for second-phase velocities of (a) 2.5 cm/s, (b) 5 cm/s, (c) 10 cm/s, and (d) 30 cm/s). ............................................................................................ 41

Figure 17: Second phase duration verification. Overlaid curves without (a) and with (b) 30 cm/s profile........................................................................................................................................ 42

Figure 18: (a) Laponite particle. (b) “House of cards” structure. (Image from http://www.scprod.com/pdfs/Laponite%20brochure%20EN.pdf) ................................................................. 43

Figure 19: Example of contrast enhancement with DIC. Center of microsphere (arrow) is on (a) frame 17, below focal plane, (b) frame 30, at focal plane, and (c) frame 48, above focal plane. Frame interval = 0.5 µm. Microsphere diameter = 3 µm. Bar = 5um................................................................. 45

Figure 20: Plot of correlation values for a microsphere. Values increase as bead approaches focal plane during z-scan, and decrease as it passes it, reaching a peak when it is in focus. The local maximum is defined as the center of the bead. Frame intervals = 0.5 µm. Microsphere diameter = 3µm. .............................................................................................................................................. 47

Figure 21: XY position plot of captured beads............................................................................ 48

Figure 22: (a) Pre- and (b) post-frozen region imaged with DIC at 20x magnification. Bead patterns (circled region) are identifiable. Arrows indicate similar bead pattern below the focal plane. Bar 25um. .......................................................................................................................... 48

Figure 23: Description of spherical coordinate system. ................................................................ 50

Figure 24: Verification of accuracy for microsphere center calculation with MATLAB algorithm. ....................................................................................................................................................... 51

Figure 25: (a) Overhead and side view of sample on glass slide. (b) Actual sample image. ....... 53

Figure 26: Sample setup (a) From left to right: FCS2 chamber base, lower layer of silicon, sample on slide with O-ring, upper silicon layer with thicker layer (orange) and glass cover creating the space for the sample to occupy, and FCS2 chamber locking cover. (b) All pieces of (a) assembled together, in same order.................................................................................................................. 54

Figure 27: Example DIC images of (a) pre- and (b) post-frozen sample. The glass fiber pattern is used to find the same region of interest. Bar = 40 µm. ........................................................................ 55
Figure 28: (a) Stack for mounting sample onto the motor slider. (b) Sample attached to stack and mounted onto motor slider ............................................................................................................ 56

Figure 29: Second phase duration verification plots. Overlaid second phase curves of (a) all and (b) without the 30 cm/s impact velocity runs............................................................................................................ 58

Figure 30: (a) Second phase curves and (b) final 2 mm of travel for the 2.5 cm/s impact velocity runs. Arrow points to approximate position of sample during impact. ........................................ 59

Figure 31: (a) Second phase curves and (b) final 2 mm of travel for the 5 cm/s impact velocity runs. Arrow points to approximate position of sample during impact. ................................. 60

Figure 32: (a) Second phase curves and (b) final 2 mm of travel for the 10 cm/s impact velocity runs. Arrow points to approximate position of sample during impact. ............................ 61

Figure 33: (a) Second phase curves and (b) final 2 mm of travel for the 30 cm/s impact velocity runs. Arrow points to approximate position of sample during impact. ............................... 62

Figure 34: Plot of average strain versus impact velocity for control and 2.5, 5, 10, and 30 cm/s impact velocity runs. Data is from Table 4.................................................................................. 64

Figure 35: Glass fiber distortion between (a) pre- and (b) post-frozen samples following impact freezing with the DDK Cryogun (impact velocity = 2.3 m/s). The fibers are greatly distorted. Bar = 40µm. ......................................................................................................................................... 71

Figure 36: Glass fiber distortion between (a) pre- and (b) post-frozen samples following impact freezing at .......................................................................................................................... 72

Figure 37: Glass fiber distortion between (a) pre- and (b) post-frozen samples following impact freezing at .......................................................................................................................... 73

Figure 38: Glass fiber distortion between (a) pre- and (b) post-frozen samples following impact freezing at .......................................................................................................................... 74

Figure 39: Glass fiber distortion between (a) pre- and (b) post-frozen samples following impact freezing at 2.5 cm/s. Significantly less fiber distortion compared to the Cryogun (Figure 35). Bar = 40µm. ......................................................................................................................................... 75

Figure 40: Glass fiber distortion between (a) pre- and (b) post-imaging of control................. 76
LIST OF TABLES

Table 1 - Peak amplitudes during the latter half of the second phase test profiles.................. 42

Table 2 Parameters used for programming the dual-phase profile (see Section 7.1). ................. 52

Table 3 – Peak amplitudes during the latter half of the second phase profiles. *This run produced
to shifted sets of curves. One set had 1.5 cm/s is amplitudes, and the other had 1.0 m/s. (see
Section 9.2 for details).................................................................................................................. 63

Table 4: Averages and standard deviations of strain (SR), absolute change in theta (Abs Theta),
and absolute change in phi (Abs Phi) for all impact velocities and control. ................................. 63

Table 5: Statistical difference results (P-values) between impact velocities and control.......... 64
Appearances are often deceiving – Aesop

Accuracy of observation is the equivalent of accuracy of thinking. - Wallace Stevens

1 INTRODUCTION

Today, electron microscopy (EM) offers better than 50 pm resolution [1]. Observing biological samples with EM is common for ultrastructural and immunolabeling studies, and capturing rapid biological processes [2–4]. The ultimate aim for these methods is to preserve the specimen and its features such that when it is viewed by EM, an in vivo representation is observed. Ideally, no new features (artifacts) should be introduced during specimen preparation for EM observation.

In reality, however, this is not the case. Artifacts are always introduced to the sample throughout the preservation process as well as during EM observation itself (see Bozzola and Russell, [5]). The burden to discern artifact from in vivo features falls on the observer. This task is not trivial, and can lead to inaccurate observations and conclusions about the process or structure being studied. Although artifact cannot be completely prevented, it can be minimized by selecting the appropriate preparation methods for the specific sample to be observed. This could make recognizing artifact less trivial, and lead to more accurate observations.

1.1 CRYOPRESERVATION

A common technique for preserving biological specimens is through cryopreservation. The sample, which typically ranges from a cell monolayer to a whole tissue section, is fixed by cooling to very low temperatures, at least 77 K (-196°C), boiling point of liquid nitrogen (LN).
All biological and biochemical activity is effectively halted at these sub-zero temperatures (see Chapter 2 in Grout and Morris for details, [6]).

When cooling the sample to such temperatures, it is natural for water in the sample to undergo a phase change from liquid to solid (ice). Details on ice formation are explained elsewhere [6-9]. Briefly, when water is cooled below its freezing point, it is referred to as being supercooled. Ice forms in supercooled liquids through the growth of ice crystals in an event called nucleation. In homogenous nucleation, the foundation from which ice crystals form is called a nucleus, a small cluster of water molecules. As supercooled liquid water cools below 0°C, various sized clusters spontaneously form from the natural movement of water molecules due to thermal fluctuations. A majority of these clusters dissipate, however, because they succumb to surface tension acting on them by the surrounding. For a cluster to survive, it must be big enough to overcome this surface tension. Once a cluster of ‘critical size’ is formed, it survives and forms a nucleus to which more molecules can attach to, enlarging the crystal. As the temperature of the water decreases, the surface tension also decreases. Consequently, the critical size for a cluster to become a nucleus decreases, which in turn, increases the number of nuclei in the liquid. Near -40°C, the critical size is approximately equal to the typical cluster size. Homogenous nucleation must occur at this point. The water freezing at this temperature would essentially consist entirely of ice crystal nuclei. Hence, the water would freeze without significant ice crystal growth.

The greatest obstacle with freezing biological samples is avoiding the damage caused by ice formation. Ice damage can affect samples in three ways. First, as water in an aqueous solution transitions to ice, the remaining liquid water increases the concentration of the solute. As more liquid crystallizes, the concentration increases. These high concentrations can be toxic
to cells [10]. Second, as water cools, it can form ice in the extracellular space. This can draw water from intracellular regions and lead to cellular dehydration and cell membrane damage [11, 12]. Finally, intracellular ice formation can be equally damaging for cell membranes, and is taken to indicate cell death [12-14]. These events can introduce artifacts to the sample which deviate it from the true in vivo state (see Gilkey and Staehelin for recognizing ice artifact, [15]).

Destructive ice crystals can be minimized by cryoprotecting the sample with an anti-freezing agent (e.g. glycerol or sucrose) prior to freezing [16], by high pressure freezing (HPF) [17], or by freezing the sample at extremely high cooling rates [18]. All these methods attempt to freeze the sample before ice crystals grow to a damaging size.

Of these methods, the favored is freezing at high cooling rates, referred to as ultrarapid (UR) freezing. It was thought that these cooling rates freeze water before it can crystallize. When this happens, the ice is said to be amorphous, or vitrified. This would eliminate the need to use cryoprotectants prior to freezing. Rasmussen points out, however, that although theoretically the water vitrifies at these cooling rates, it is unlikely that all the water has vitrified. Rather, it began to crystallize, but froze before the ice crystals could get large enough to (i) be seen and (ii) to cause ultrastructural damage [19]. This idea is incorporated into Gilkey and Staehelin’s definition of UR freezing: “a region of an unfixed, nonchemically cryoprotected sample has been ultrarapidly frozen if heat has been withdrawn so rapidly that no ice crystal damage is visible at the [electron microscope] level” [15].

Common UR freezing techniques include plunge, spray, jet, and cold metal impact freezing. Reviews of all these methods and their applications are described and discussed elsewhere [7, 15, 20, 21]. Impact freezing is a preferred UR technique for TEM sample preparation, as it offers the highest cooling rates of all the UR freezing methods.
1.2 IMPACT FREEZING

1.2.1 Background

Impact freezing, also referred to as slam freezing (SF) or metal block freezing, involves bringing the sample into contact with a polished, ultrapure metal mirror (typically copper or silver) cooled to liquid nitrogen (LN) or liquid helium (LHe) temperatures, 77K or 4K respectively. The cooling rates provided by impact freezing (>10⁴ K/s) have allowed biological samples to be frozen, with no noticeable ice crystal growth or damage, up to 30µm below the frozen surface [3, 20, 22-36].

Although the idea of freezing through contact with a cold block was first attempted by Simpson [37], Eränkö first proposed that this technique freezes samples quicker than immersion cooling, and used it for light microscopy observation [38]. Van Harreveld and Crowell first developed the method for use in EM observation [33], which was later modified and improved upon by Heuser, Escaig, Heath, Phillips, Boyne, and Coulter [25, 39-43].

1.2.2 Metals and Cryogens Used

Impact freezing typically uses metals such as copper or silver for the slamming surface. Although these metals have nearly the same thermal capacity as some organic liquids used in other techniques, their thermal conductivity is much higher. Consequently, the same quantity of heat is transferred nearly 10³ times faster through the metals than through organic liquids such as propane [7]. The faster the heat transfer, the quicker the freezing and fewer ice crystal formations, especially in specimens that were not cryoprotected prior to freezing [20, 39].

The cryogens commonly used to cool the metal mirror are either LN or LHe. Whatever the combination of metal and cryogen, the cooling rates provided are significantly greater than using a liquid cryogen alone. Terracio and Schwabe calculated the cooling rate of copper at LN
temperature (impact freezing) to be 123 percent faster than liquid propane alone (as used in plunge freezing) [44].

Bald used finite element analysis to calculate the optimum cooling block temperature for copper, silver, aluminum and sapphire. He concluded that (i) the optimum freezing temperature for copper is 18K, but provides the fastest cooling rates of all other metals for the first 30µm when cooled to LN temperatures, (ii) the optimum freezing temperature for silver is 16K, and its cooling rate drops 50 percent when cooled to LHe temperatures, because of the metal’s temperature-dependent properties [35]. In another study, he calculated the there is little difference between the cooling rates of silver at LHe temperature and copper at LN temperature [45]. Despite this, there still seems to be a debate as to which cryogen is better (see Echlin and Sitte et al. for more details) [7, 20].

1.2.3 Impact Velocity

There have been a wide range of velocities reported for various impact freezing designs. Common velocities are 3 to 5 m/s [7, 20, 26, 36, 39, 46, 47]. Other designs have incorporated relatively slower travel velocities (20 to 100 cm/s) [33, 48-50]. These high velocities are required to eliminate precooling of the sample by allowing it to rapidly pass through the cold cryogen gas layers above the mirror surface [15, 39]. Precooling lowers the difference in temperature between the mirror and sample. The heat flux through the sample into the mirror is minimized and the sample freezes at a slower rate. Consequently, larger distorting ice crystals form.

1.2.4 Bounce

In earlier designs, it was noticed that the sample was unable to maintain initial contact with the mirror. Even millisecond bounces can significantly reduce the cooling rate and lead to distorting ice crystal formation and lead to ultrastructure artifact [43]. Subsequent impacts can
even damage the sample [42]. Both Boyne and Heuser designed impact freezers to suppress bounce. Heuser used strong retention magnets to maintain initial contact [25], and Boyne incorporated a hydraulic damping system into his design [42]. In another design, Coulter achieved the same result by using a toggle linkage mechanism which decelerated the sample as it approached the mirror surface, and kept contact with the mirror once it was made [43].

1.2.5 Advantages of Impact Freezing

Before embedding for ultrathin sections, impact frozen samples are dehydrated by either freeze substitution (FS) or freeze drying (FD). These methods and their applications have been reviewed and described [7, 20, 24, 29, 44, 51-54]. Impact freezing, in combination with FS or FD, has shown that it better preserves both ultrastructure [3, 30, 33, 54-58] and antigenic sites for immunolabeling study [3, 24, 32, 51, 57, 59] better than both conventional chemical fixation and cryopreservation methods, involving chemical cryoprotection.

Impact freezing has also been used in combination with freeze fracture and deep etch replication (quick-freeze deep-etch, QFDE) [2, 23, 60-66]. Specifically, work by Heuser et al. in 1979 allowed QFDE to become a favored technique in biological specimen preparation since it (i) better preserves the ultrastructure of samples compared to conventional transmission electron microscopy (TEM) methods (ii) gives higher resolution images and (iii) eliminates the need for prefixation and cryoprotective agents [4].

The high cooling rates provided by impact freezing have been calculated to freeze the first 10µm of the sample, theoretically, in less than 1ms [35]. Experimentally, van Harreveld et al. calculated less than 8.5ms [50], Escaig calculated less than 1.5ms [39], and Heuser et al. calculated approximately 1ms [4]. This has allowed the arrest of rapid physiological events [4, 50, 60, 67, 68]. This cannot be done with other preservation methods. Chemical fixation can take
several seconds to immobilize structural features [15, 56]. Although HPF can freeze samples to a
greater depth (~200 mm), it takes up to 30ms. This is not fast enough to capture events like exo-
and endocytosis at the frog neuromuscular junction, which have been captured in the single digit
millisecond range [4, 67].

1.3 IMPACT FREEZING OBSTACLES

1.3.1 Sample Travel Distance

Impact freezer designs originally accelerated the sample towards the mirror surface by
gravity, air pressure, or tension springs. Newer designs incorporate electromagnetic motors [47].
Regardless of the propelling method, a mechanical hard stop is used for stopping the sample
once contact with the mirror is made [25, 33, 39, 40, 69]. Consequently, the sample makes
contact with the metal block at the high velocities required to prevent precooling.

Other designs allow for an offset stop distance from the mirror to be set, thus preventing
the sample from getting squashed [43, 70, 71]. With such designs it is not possible to effectively
know the distance from the leading edge of the specimen to the cooled mirror surface. The
stopping position of the plunger that holds the sample is typically set on the device either
visually, empirically, or by approximation. Even commercial impact freezers have this
drawback. Leica’s MM80 (Leica Microsystems, Buffalo Grove, IL) adjustments are made by
setting thickness, speed and pressure parameters according to an arbitrary scale. For instance, a
low speed of “1” is recommended for sensitive specimens, while a max speed of “11” can be
used for more robust specimens. For Delaware Diamond Knives’ Cryogun (DDK Inc.,
Wilmington, DE) the stopping position is set by adjusting the threaded hard stop.
These inaccurate methods can lead to the sample not making intimate (or any) contact with the cold metal which can slow the rate of freezing and allow for ice crystal distortion, or too much contact being made resulting in great and damaging distortion from the impact. Many designs have sought to minimize making too much contact with the mirror by cushioning the sample with foam or small pieces of liver, and/or mechanical dampers [7, 25, 39, 42, 61, 72]. Coulter designed a device that incorporates a toggle linkage to slow the velocity of the sample as it approaches the mirror surface [43]. However, this still involves estimating the stop position of the plunger that holds the sample.

1.3.2 Impact Distortion

As mentioned previously, artifact is introduced in every preservation method. Impact freezing is no exception. Ice formation and bounce have already been discussed. Former approaches have well understood these potential artifact-inducing effects, and their designs reflect attempts to minimize (ice) and suppress (bounce) them. Mechanical deformation resulting from the impact itself, however, has yet to be quantitatively investigated.

Samples making contact with the mirror at up to 5 m/s must undergo some form shearing or thrust effect. When impact is made with the mirror, it has been suggested that two waves are formed. The first is a freezing wave resulting from the temperature gradient between the ambient temperature sample and the LN or LHe temperature mirror. The second is a shock (deformation) wave resulting from the contact with the mirror. Da Silva and Kachar explain that this shock wave propagates through the sample as increased hydrostatic pressure [73]. If the shock wave precedes the freezing wave, this will result in distortion of the sample’s ultrastructure prior to freezing. This still remains to be tested.
Previous investigators have reported on distortion effects resulting from the sample’s impact with the metal mirror [29, 68, 72-76]. However, to the author’s knowledge, no investigation has been attempted which quantifies the distortion effect as a function of impact velocity, thus enabling the development of a device which minimizes the mechanical distortion effects.

The purpose of this thesis is to (1) modify our current “touch” freezer; (2) introduce a dual-phase velocity profile for controlling the approach velocity of the sample to the cooled metal mirror; (3) introduce a new assay capable of quantifying mechanical deformation effects produced during sample impact with a cooled metal mirror; and (4) verify the hypothesis that mechanical distortion of the sample experienced through impact freezing are proportional to impact velocity.

2 TOUCH FREEZER

2.1 BACKGROUND: PHASE 1 TOUCH FREEZER DESIGN

The original work of a gentle impact (touch) freezer was designed and built by the author and co-workers as part of a senior undergraduate design project (Capstone) at Northeastern University (Boston, MA) in 2008 [77]. The goal of the project was to design an impact freezing apparatus that would: (i) cool a sample fast enough to form a ice-crystal-free region within the first 15-25µm depth, (ii) minimize sample deformation due to impact with the metal mirror, (iii) have the ability to cycle between four mirrors to freeze against, (iv) include a temporary storage area for already frozen samples, and (v) have a budget of $5,000 to manufacture the device. The final prototype is shown in Figure 1.
2.2 PHASE 1 COMPONENTS AND METHODS INCLUDED IN PHASE 2 DESIGN

2.2.1 Linear Motor

Of all the components of the design, that which advances the sample to the cooled mirror is the most important. It must have the ability to accelerate to relatively high velocities. This will allow the sample to travel through the liquid cryogen vapor layer, thereby minimizing precooling of the sample. It should also not cause any bounce effect once it decelerates. Bounce will cause the sample to make secondary contact with the mirror, which dramatically affects freezing quality. Finally, the component should have a high repeatability and accuracy to ensure that consistent contact with the mirror is always made.
An electromagnetic linear motor (LinMot Inc., Elkhorn, WI) was incorporated into the design (Figure 2). It uses electromagnetic force to create linear movement. Consisting of only three parts, a slider, stator, and a flange mount, the LinMot motor does not incorporate belts, gears, etc. which can wear and require replacement. The slider is a stainless steel tube containing neodymium magnets, which slips into and slides within the stator. The stator houses coils to provide the electromagnetic force which moves the slider.

The motor offers a 50 µm repeatability, which is an acceptable variation to have, since a typical freezing motion profile will travel an additional 1mm, at least (see Section 7.1). The magnetic force generated has the ability to accelerate the slider to velocities up to 5 m/s through the 270mm available slider travel distance.

The LinMot motor is a marriage of Heuser and Coulter’s novel ideas of using magnetic forces and slowing down the sample as it approaches the mirror, respectively, to minimize bounce (Section 1.2.4). The magnetic induction provides greater accelerating and decelerating
control than pure mechanically driven ultrarapid freezers. The combination of decelerating the sample as it approaches the mirror with the ability to keep the sample still once contact is made dramatically minimizes bounce effect.

2.2.2 Wide Beam Laser

To overcome the inability to measure the sample thickness (Section 1.3.1) a wide beam laser sensor (Keyence, Woodcliff Lake, NJ) was incorporated in the design (Figure 3). Integrating this sensor with the linear motor eliminates the need of needing to know the thickness of the sample to be frozen. Consequently, samples of any thickness can be brought right to the surface of the mirror and effectively frozen. A full description is included in subsequent sections.

![Figure 3: Keyence wide beam laser sensor (www.keyence.com).](image)

2.2.3 Cooling Materials

Copper was selected as the material for the mirrors, and LN was chosen as the cooling cryogen. Liquid nitrogen was chosen because it is more readily available and more cost effective than LHe. Since LN was shown by Bald to provide the fastest cooling rates when it was used to cool copper ([35]), it was the natural choice for the metal material.
2.2.4 Operation

The LinMot controller and software is used to program and execute motion profiles. After the inner basin has been filled with liquid nitrogen and has reached 77K (approximately 5 minutes), the mirror module is rotated and locked in place to expose a cooled mirror. The system can then be calibrated. This involves calculating the distance between the planar laser and the active mirror surface (Figure 4). This allows for samples of any thickness, known or unknown, to be accurately accelerated towards the mirror and stopped just as contact with the mirror is made, essentially touch freezing them. This process is described as follows.

Touch freezing a sample involves two sequences, termed calibration and freeze. The motor is switched on and the slider is initialized to its home position. Home is defined as the front face of the slider being 10mm below the front face of the stator. Since the slider will briefly make contact with the mirror during the calibration sequence, a magnetic adaptor is attached to the front face of the slider. It has an extruded semicircular lip that makes contact with part of the outer edge of the mirror to avoid contact with center of the mirror where the sample will make contact.

During the calibration sequence, the operator manually moves the slider, with the adaptor attachment, until it makes contact with the mirror, and records the distance (Figure 4, $X_M$) displayed by the software. The slider is then homed and a command is initiated to advance the slider at 1 mm/s until the first interference with the wide laser beam sensor is detected, upon which the motor stops. The user then records this distance as well (Figure 4, $X_L$). The difference (Figure 4, $X_{TF}$), which is the distance from the laser to the mirror, is entered into the program. A touch sequence profile is then created by the program for the motor to follow,
according to the user’s chosen velocity, acceleration and deceleration values. The calibration takes approximately 1 minute to complete.

Figure 4: Description of how touch freezing sequence distance, $X_{TF}$, is measured through the combination of a linear motor and laser sensor in the original design [77]. This procedure was kept for the phase 2 design. The measurements are calculated on the active mirror. See text for full description.
2.2.5 Control Hardware and Software

Control of the system was achieved by an E1100 Servo Controller (LinMot Inc., Elkhorn, WI). It communicated with a PC via RS232 connection. The servo controller allows the motor and laser to be integrated. The execution of the velocity profiles was controlled directly from a PC using LinMot Talk software (version 4.0). The software provides a digital oscilloscope. Remaining control hardware includes a voltage supply (72VDC) for the motor, a voltage supply (24VAC) for both the controller and laser sensor, and a control panel for initiating command sequences.

LinMot Talk was used to create the calibration and touch freezing sequences, upload them to the motor, and control the motor. Commands for each sequence were assigned to a command table input/output (IO) interface. The interface was used to be able to incorporate the switch panel to initialize sequences, and allow the laser sensor to control the motor. For instance, when the sample interrupts the laser beam, it sends a signal to through the switch panel to the controller for the motor to follow the command line with the touch freezing profile.

2.2.6 Discussion

Combining the motor with a planar laser also eliminated the problem of unknown sample thicknesses which led to deformations from the impact itself. By allowing the sample to travel through a known calculated distance, it can decelerate as it approaches the mirror, and effectively be touch frozen.

Despite the uniqueness of the device, there were areas that needed improvement. First, a redesign of the inner basin, which houses the mirror module, was required. Those components included a high volume of aluminum, which required over 15L of LN to cool to 77K. This is a
large quantity to use for one round of sample freezing. A solution was needed to minimize the amount of LN required to cool the mirrors. The oscilloscope feature needed to be utilized through the application software to measure sample velocity at impact. Redesigning the device, verifying motor control, development and testing of a dual-phase velocity profile are described in the next sections.

2.3 MODIFIED DESIGN (PHASE 2)

2.3.1 Overview

The new design is shown in Figure 5. It consists of an updated main frame (Figure 5, a) which supports the paired motor/mirror module assembly (Figure 5, b) and a wide beam laser assembly (Figure 5, c). Refer to Figure 6 for details.

Figure 5: Modified touch freezing device with (a) a simplified main frame, (b) a paired motor/mirror module assembly, and (c) wide beam laser assembly.
The linking together of the mirror module with the linear motor module allows for the basin assembly from the original design to be replaced by a standard polystyrene foam box. The main frame was modified slightly to increase the new design’s sturdiness.

Figure 6: Side view of modified touch freezer device. Two mounting plates (a) are fastened to both sides of the motor flange, which holds the motor stator and slider. The opposite ends of the mounting plates are fastened around an angle bracket which is mounted onto the main frame. The angle bracket keeps the angles the motor at 60 degrees from the horizontal. A central brace (b) is fastened onto the back face of the motor flange. Two pillow block linear ball bearings (c) are fastened near the ends of the brace. These allow for a precision shaft (d) to be aligned parallel to the motor slider. The precision shaft holds the new mirror holder. It is comprised of a four armed mirror holder (e) which is fastened to an aluminum shaft (f). The aluminum shaft is connected to the precision shaft via a ceramic standoff (g). A gear locking mechanism (h) is fastened to the opposite end of the central brace for mirror interchange. A hard stop (i) is fastened onto the mounting plates as a safety precaution to prevent the motor slider from advancing further than programmed. A wide
beam laser emitter and receiver (j) are mounted onto two metal plates which are fastened to the main frame. Two stabilization braces (Section 2.3.3) are mounted to the main frame and fastened onto the central brace.

2.3.2 Mirror Module

The critical feature in need of redesigning was the mirror module. The original module had to be large enough to house four thermocouples for mirror temperature monitoring. These were ultimately not used, however. Furthermore, the module also needed to rotate between four mirrors. In order to keep the module from moving, and ensure the active mirror would be positioned and kept perpendicular to the motor slider, it had to be fastened onto the inner basin. This combination led to a bulky module which, along with the inner basin plates, required over 15L of liquid nitrogen to bring to temperature.

Consequently, a smaller and simpler mirror module could be used. This would significantly minimize the amount of liquid nitrogen needed to cool the module. As a result, a four armed aluminum module (Figure 7) was designed and machined to hold the copper mirrors. The mirrors are inserted into through-holes on the module and secured via set screws.

Figure 7: Four-armed mirror module. Arrow shows set screw location for securing the copper mirror.
The mirror holder is attached to a 25 inch precision shaft. The shaft is guided and aligned parallel to the motor slider by two pillow block linear ball bearings. The bearings are fastened onto a central brace which is mounted onto the motor flange mount. Each bearing allows a maximum misalignment of 0.5 degrees. Given the length of the shaft, two bearings were mounted at the extreme ends of the central brace to minimize any misalignment that could result from having one bearing for the specific shaft length. A basic two-armed handle is fastened to the top end of the shaft for ease of rotating between mirrors.

Rotating between mirrors is accomplished by a gear locking mechanism (Figure 8). A four notched gear is secured onto the mirror module shaft via set screw. A platform with a mating tooth is fastened above the upper bearing onto the central brace. The platform has a hole for the shaft to fit through. The gear notch slip fits around the tooth to constrain rotation of the shaft, and rests on the platform to keep the shaft from advancing further under its own weight. To rotate between mirrors, the shaft is raised off the platform by the handle. Once the gear disengages from, and clears the tooth, the module can be rotated in 90 degree increments to expose a new mirror. The module is then lowered to reengage the tooth and lock it in place.
2.3.3 Support Braces

The central brace extends 5 inches below the lower edge of the motor flange mount. Designing the brace longer than the flange mount allowed for the bearings to be placed further apart. However, the free end of the bract acts as a cantilever. Consequently, that portion would vibrate from the acceleration and deceleration of the motor, which was transferred to the mirror module. Any motion on the mirror module, especially during the slider’s deceleration, can cause the sample to lose contact with the mirror once initial contact is made. This can compromise the freezing quality of the sample. Therefore, a support brace was required to secure the free end of the central brace and minimize these vibration effects.

Since the precision shaft of the mirror module is aligned parallel to the longitudinal axis of the central brace, two support braces (Figure 9) were designed to fasten on either side of the
shaft. The braces have a rectangular cross section to resist any moment caused through the vibration. Both support braces were fastened to the central brace and mounted onto the main frame.

![Support brackets used to stabilize the central brace. Arrows indicate orientation of brackets.](image)

**Figure 9: Support brackets used to stabilize the central brace. Arrows indicate orientation of brackets.**

2.3.4 Hard Stop

The design of the touch freezer allows for a sample to be brought to the active mirror quickly, and stop just as contact is made. It is imperative that the calibration sequence is performed correctly, to ensure the motor slider that holds the sample does not go too far. Any miscalculation or typo entering the travel distance into the software can lead to a dangerous and potentially catastrophic impact with the mirror.

If the motor slider travels too far, the impact with the mirror module would cause a moment on the module's active arm. If the moment is great enough, it will lead to the failure of
the weakest component within the mirror module. In this device, it is most likely that the threads of the ceramic standoff would get stripped. Regardless of whether or not this occurs, there is a chance of liquid nitrogen getting splashed on a user, and/or debris from the impact projecting into the surrounding area. In order to best avoid this, a hard stop was designed to ensure the motor slider can only go a set distance.

The hard stop assembly (Figure 10) consists of an aluminum platform that is fastened to two support arms (Figure 10, a & b). Both arms are stepped for the platform to rest on and ensure the edges are aligned before it is fastened. The support arms are in turn fastened to the motor flange through the mounting arms plates. The platform assembly is positioned over the motor stator. A counterbored through hole on the platform allows the motor slider to travel through it. A 1/8 inch thick neoprene washer is placed on the counterbore hole in the platform.

To complete the assembly, a 2-1/4 inch fastener ((Figure 10, d) was inserted into the threaded end of the slider which travels through the hole on the platform. Two 9/16 inch washers are sandwiched between the head of the bolt and a 1-1/8 inch washer. All the washers are secured onto the fastener by a 1-1/4 inch spacer and nut. The fastener and washers are positioned above the platform and are fastened through the platform’s through hole into the slider’s threaded end.
The workings of the hard stop assembly are described as follows. Before the calibration sequence, the user manually moves the slider to the approximate position it would be when the sample is frozen. The hard stop fastener can then be tightened or loosened on the motor slider until the gap between its washer and the neoprene washer on the platform is approximately 1mm. The gap size represents the slider overshoot that will be allowed before the hard stop engages. The device can then be calibrated and run. In the event of an overshoot of the slider, the fastener’s washer will be pulled toward the platform’s through hole. Since the washer cannot fit through the hole, it will interfere and make contact with the platform’s neoprene washer (Figure 10, c), thereby being constrained from further travel (see Figure 11).

Two different diameter washers are secured onto the fastener to distribute the load more uniformly from the large washer to the fastener head. The neoprene washer serves to dampen
the impact once the hard stop engages. In cases where more or less travel distance needs to be used, a different length fastener can be used to for the desired stop position. As an additional safety parameter, the motor has been programmed to cut power to the slider if half the maximum current is drawn. This would prevent the motor from continuing to advance the slider once the hard stop has engaged. The hard stop is not intended to be used frequently. It is a safety precaution in the event of a user mistype of the travel distance of the slider, or miscalculation during the calibration sequence.

Figure 11: Hard stop. (a) Position of the hard stop fastener as the slider approaches the mirror. (b) Position of the hard stop fastener when the motor has reached the mirror and stopped. Note the gap between the metallic and neoprene washers. (c) Worst case scenario. The motor travels past calculated distance. The two washers make contact and prevent the motor from continuing its motion.

2.3.5 Main Frame

Minor modifications were made to the mainframe for the phase 2 design. In the original design, the basin was fastened to the frame, which stabilized the overall system. Removing the basin meant that the frame was now less stable. The thrust generated by the linear motor would make the system more susceptible to rocking.
To minimize this effect, each of the frame’s side legs was fastened to the front ends of two base pieces of the extruded aluminum, forming an ell. Another piece was connected at a 60 degree angle from the horizontal to connect the back end of the base piece to the top end of the side legs. This created a triangular frame that constrained the frame from any cantilever effects (Figure 12). Furthermore, the base pieces were fastened to the lab’s optical table. This, combined with the triangular frame, secured and prevented any rocking effect.

![Figure 12: Main frame of modified touch freezer.](image)

2.3.6 Switch Panel

With the original design, not much emphasis went into improving how efficiently the user interacts with the LinMot Talk control panel. This is where all the motor commands are controlled. Most of these commands required manual activation through the control panel window. This can interrupt the flow of an experiment. The user also has to turn from the device to the computer to manually click the appropriate checkboxes, when their attention should be with the device and the sample. In order to give the user easier accessibility to, and making the interaction with, LinMot Talk more efficient, a new switch panel was designed to complement the phase II device.
The original switch panel was designed from sheet metal with only a push button for initiating a sequence and one to act as an emergency stop. The new switch panel (Figure 13) consists of both push buttons and toggle switches mounted on a new control box. In addition to the original ‘Go’ and ‘E-Stop’ buttons, the new features include three push buttons (Initialize Home, Home, and Error Acknowledge) and one toggle switch (Enable Motor).

A user interacts with the switch panel as follows. To power the motor, the ‘Enable Motor’ switch is toggled on. Since this command is now controlled externally via the toggle switch, there is no need to subsequently switch off and back on again, as with the manual method described above. The ‘Initialize Home’ push button is held down until the slider initialization is completed, and then released. If an error is triggered, the ‘Error Acknowledge’ button can be held down until the error clears. The ‘Home’ button is pressed to bring the slider to the home position at any time after the initialization has been completed. An extra toggle switch and push button were added, for potential future use.

All the wiring from the 24VAC power supply and laser controller was fed through the switch panel as well. Additionally, the laser controller itself was mounted onto the side of the control box. This allowed for a single device that housed all the controlling components. The new buttons and toggle switch were wired to a digital I/O port on the Linmot controller. These, like the original push buttons and laser controller, are triggered by 24VAC.
3 DUAL-PHASE VELOCITY PROFILE

3.1 OVERVIEW

The combination of a magnetic induction linear motor and wide beam laser sensor allows for accurate measurement of the distance the sample is to travel before making contact with the mirror. Once the distance from the laser sensor to the mirror surface is known, a dual-phase velocity profile can be programmed into the control software for the motor to follow.

When the sample interrupts the laser sensor, the first phase is triggered and it is accelerated through the cold nitrogen vapor at a velocity up to 5 m/s (maximum achievable velocity of this model/controller). The second phase decelerates the sample to a velocity as low as 2.5 cm/s for the final 0.1 seconds of the profile, whereupon contact with the mirror is made.

The duration of the second phase was chosen to ensure that all samples, at every impact velocity tested, get the same second phase exposure time. However, too much time gives the sample a longer exposure time in the cold cryogen vapor region, thereby increasing the chance of precooling.
The slow impact velocities tested were 2.5, 5, and 10 cm/s. These are compared to 30 cm/s. Although this is one of the slowest velocities in the literature (second slowest to 20 cm/s reported by Morgenstern and Edelmann [48]), it was assumed that if this minor increase in impact velocity shows an increase in resulting mechanical distortion, then a more significant increase in impact velocity will result in a more significant increase in mechanical distortion.

3.2 PROGRAMMING THE DUAL-PHASE VELOCITY PROFILE

Before programming the dual-phase velocity profile, the touch freezer is calibrated as described in Section 2.2.4. Once the measured distance from the laser to the mirror, $X_{TF}$, is measured, it is separated into two distances, $d_1$ and $d_2$. The former is the distance of the first phase, where the sample travels at 4.5 m/s to avoid precooling. The latter is the distance of the second phase, where the sample travels at the impact velocity and makes contact with the mirror.

Distance $d_2$ is defined by

$$d_2 = v_2 t_2$$

With time $t_2$ of the second phase already defined to be 0.1 s, second phase impact velocities, $v_2$, of 2.5, 5, 10, and 30 cm/s give a second phase distances, $d_2$, of 2.5, 5, 10, and 30 mm, respectively. Knowing $d_2$, $d_1$ is defined as

$$d_1 = X_{TF} - d_2$$

The distances, and velocities for both phases are now defined. However, a few more parameters need to be defined and optimized to have an effective dual-phase velocity profile.

3.3 MOTOR ACCURACY EVALUATION

Programming a velocity profile requires distance, acceleration, deceleration, and velocity parameters for both phases. The distance is measured and the velocities are defined, but the other
parameters can be any value within the capabilities of the motor. Therefore, accuracy testing was performed to find the optimum parameters to use in testing the dual-phase profiles.

The parameters tested were 100 and 150 m/s$^2$ for acceleration and deceleration, and 4, 4.5, and 5 m/s for velocity of the first phase. Every combination (12 total) of these parameters was tested ($n = 5$) for each of the slow impact velocities.

The test setup used (Figure 14) incorporated a set of digital calipers being secured on the bench top such that its depth probe is lined up with the motor slider. A distance representing a typical distance to the mirror was selected and with the slider at that position, the caliper probe was brought into contact with its leading face, zeroed, and backed off a few millimeters. Once the parameters being tested programmed into the controller, the dual-phase profile was run. The caliper probe was brought back to the surface of the slider and the measurement was recorded.
Figure 14: (a) Motor accuracy verification test setup. (b) Caliper probe backed off to avoid impact from motor trajectory.
(c) Caliper probe reapplied to the motor slider to measure position where it stopped.

The runs of each set were averaged and are plotted in (Figure 15). The parameters that were the most consistent across all three secondary phases were 100 m/s\(^2\), 150 m/s\(^2\), and 4.5 m/s for acceleration, deceleration, and maximum velocity, respectively. Equally good results were given with the same acceleration and deceleration for the 5 m/s maximum velocity. However the motor did not consistently reach 5 m/s so it was not chosen. The results show that the LinMot overshot for all the runs, but not more than 110 µm. For the parameters chosen, the overshoot was approximately 70 µm. This is tolerable for this investigation. The repeatability for all the dual-phase runs was less than 20 µm, which is well within the advertised 50 µm repeatability of the motor.
Figure 15: Motor accuracy result plots for various combinations of first phase accelerations (Acc) and decelerations (Dec). (a) Acc = 100 m/s², Dec = 100 m/s², (b) Acc = 100 m/s², Dec = 150 m/s², (c) Acc = 150 m/s², Dec = 100 m/s², (d) Acc = 150 m/s², Dec = 150 m/s².

3.4 PERFORMANCE EVALUATION

Once the parameters to be used were selected, the profiles they generated had to be verified. An example run of each of the four impact velocities is shown in Figure 16. The data was captured by the LinMot Talk’s digital oscilloscope. Each profile reaches both its first and second phase velocities, as shown in the main plots and respective insets. The second phases, however, are fairly oscillatory. Although the oscillatory effect of the 2.5 and 30 cm/s curves settles during the latter half of the second phase, it is not the case with the 5 and 10 cm/s curves.
Figure 16: Dual-phase velocity profile evaluation for second-phase velocities of (a) 2.5 cm/s, (b) 5 cm/s, (c) 10 cm/s, and (d) 30 cm/s).

The impact velocity profiles were overlaid against each other (Figure 17) to check whether or not any oscillatory character of one profile interferes with another profile. Figure 17a shows the three slow impact velocity profiles. All three impact profiles are distinct. However, those of 2.5 and 5 cm/s nearly intersect at a few locations. These locations of concern are in the first half of the two profiles’ second phases. Actual impact with the sample, however, will occur in the later half (see Section 7.1) where the profiles do not intersect. This is rechecked with the experimental profiles (see Section 9.2).
Figure 17: Second phase duration verification. Overlaid curves without (a) and with (b) 30 cm/s profile.

The peak amplitudes of the curves in the latter half of the second phase are shown in Table 1. This shows that at the worst case for any velocity’s second phase (i.e. the contact velocity for each profile is at the peak amplitude) the other velocity profiles will not intersect with it, for the latter half, at least. Consequently there is no chance of two different velocity profiles making contact at the same velocity.

Table 1 - Peak amplitudes during the latter half of the second phase test profiles.

<table>
<thead>
<tr>
<th>Impact Velocity (cm/s)</th>
<th>Peak Amplitude (cm/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5</td>
<td>± 0.5</td>
</tr>
<tr>
<td>5</td>
<td>± 1.0</td>
</tr>
<tr>
<td>10</td>
<td>± 1.0</td>
</tr>
<tr>
<td>30</td>
<td>± 1.0</td>
</tr>
</tbody>
</table>

4 LAPONITE

Laponite is a synthetic crystalline layered silicate colloid [78]. When dispersed in water to form a gel, the disc-shaped crystals develop a negative charge on the faces and small positive charge along the edge. As a result, the edge of one crystal becomes attracted to the face of another. This creates a “house of cards” structure (Figure 18).
Laponite gel exhibits thixotropic rheology, meaning it is shear-thinning. The house of cards structure is broken the instant a shear load is applied. Under continuous shear, the gel exhibit fluid properties (i.e. very low resistance to flow and low viscosity). When the shear load is removed, the structure reforms.

The shear-thinning properties of Laponite gel make it an ideal material on which to test the mechanical effects of impact velocity. By infusing the gel with microspheres, their displacements resulting from impact with the mirror can be measured, since the beads will displace with the flowing gel.

5 LAPONITE IMAGING – DIC MICROSCOPY

Imaging of the microsphere-infused Laponite gel was done by Differential Interference Contrast (DIC) microscopy. DIC is an optical microscopy method that allows imaging of transparent samples (Laponite gel). Sample contrast is enhanced through a beam-shearing interference method. Briefly, a polarized light beam is sheared into two rays and is focused on the sample plane of interest. The sheared rays go through the sample plane and get absorbed,
refracted and scattered by different amounts, depending on their individual optical path. This interaction with the sample leads to a phase change of one ray with respect to the other. The rays are then recombined, with their interference leading to brightening or darkening of the image of the sample plane according to the rays’ optical path difference [79].

The advantage of this technique is that the contrast of an object is only enhanced at a very narrow focal plane. Once an object is out of the focal plane, contrast is lost and it blends into the background. An example of this effect is seen in Figure 19.

The microscope used for imaging (Nikon TE2000E, Tokyo, Japan) is equipped with a 10 and 20x magnification objectives and a digital camera for capturing images (CoolSNAPHQ2 1394; Photometric, Pleasanton, CA). The microscope has a stage that translates in the X, Y, and Z directions. The ability to translate in the Z direction allows the Laponite sample to be imaged at various Z (depth) planes. This makes it a desirable method for imaging the three dimensional positions of the microspheres within the gel.

To take a set of images along the depth of a sample (z-scan), the top and bottom boundaries are programmed into the microscope’s image software (NIS Elements; Nikon Instruments, Inc., Tokyo, Japan). The top is defined as the focal plane just above the top of the sample. This will ensure that any beads at the surface of the sample will be registered by the analysis algorithm (see Section 6.1). The section of interest in this study is the top 30 µm of the Laponite sample. This includes all reported depths of acceptable impact freezing (see Section 1.2.1). As with defining the top surface, to fully capture all the beads at the bottom end of the sample, it is defined as 50 µm, from the top surface.
6 MATLAB IMAGE ANALYSIS

6.1 MATLAB ALGORITHM OVERVIEW

Using image analysis, the relative pre- and post-freezing positions of the microsphere in the Laponite gel can be calculated and compared. The analysis is accomplished by two sets of custom MATLAB (MathWorks, Natick, MA) algorithms written by the author. A brief description of the algorithms is discussed here.

The first set of algorithms inputs a set of images taken at different Z planes. The depth interval between two consecutive images should be much less than the diameter of the microspheres being used. This will provide a more precise measurement of the center of a microsphere. Here, a 0.5 µm interval was chosen for a 3 µm diameter microsphere.

A two dimensional cross-correlation on each image is performed, with a user defined bead template. The template is a bead-sized image (approximately 15x15 pixels) of a focused

Figure 19: Example of contrast enhancement with DIC. Center of microsphere (arrow) is on (a) frame 17, below focal plane, (b) frame 30, at focal plane, and (c) frame 48, above focal plane. Frame interval = 0.5 µm. Microsphere diameter = 3 µm. Bar = 5um.
bead. The more a bead in the image matches the template, the higher a correlation value it receives on a scale of 0 to 1, where 1 is a perfect match.

Matches with a correlation greater than 0.8 from each frame are compared with the matches of all the other sets. If a bead from image A was in the same position, within a set pixel tolerance, as a bead from image B, the bead is defined as being the same. The sorted bead positions were written to an Excel file. Once a specific bead is registered (correlation is at least 0.8) its correlation value increases with each consecutive image until a maximum value is reached. This is because the center of the bead is moving closer to the microscopes focal plane (see Figure 19a). When the two match up, the bead is in focus (Figure 19b). The center of the bead then continues past the focal plane, going out of focus and disappearing (Figure 19c). The correlation values then decrease until the bead is not registered (correlation drops below 0.8).

The correlation values for each bead in the image set are plotted against the frames in which it was registered, and a second order line is fitted to the points. The inflection point is defined as the center frame of the bead. An example is shown in Figure 20.
Figure 20: Plot of correlation values for a microsphere. Values increase as bead approaches focal plane during z-scan, and decrease as it passes it, reaching a peak when it is in focus. The local maximum is defined as the center of the bead. Frame intervals = 0.5 μm. Microsphere diameter = 3μm.

When all the bead centers are calculated, the 3D coordinates (X, Y, Z-frame) of each bead are written to an Excel file. Two plots are generated from the calculated data. The first is a scatter plot of the XY positions of each bead projected onto the plot. The second includes the same information but for the top 60 frames (30 μm) from the sample surface (Figure 21).
The user uses these plots along with the image sequences to correspond beads between the pre- and post-frozen sets. Care must be taken to ensure that bead A from the pre-frozen set is indeed the same one as bead B from the post-frozen set. Typically, bead patterns can be recognized which aid in finding corresponding beads. An example is shown in Figure 22.
The user inputs the 3D coordinate data of the correlated pre- and post-frozen beads into the second set of MATLAB algorithms. The positions of the beads are converted from a Cartesian (X, Y, Z) to a spherical (r, theta, phi) coordinate system. The relative distance, \( r_{jk} \), from one bead, \( j \), to every other bead, \( k \), in each set is calculated. \( \text{Theta} \), defined as the angle about the Z-axis formed by the projection of \( r_{jk} \) on the XY plane and measured from the X-axis, and \( \text{phi} \), defined as the vertical angle from the XY plane formed by \( r_{jk} \) (see Figure 23). For \( n \) beads, the number, \( m_{jk} \), of such relative distances, \( r_{jk} \), between beads \( j \) and \( k \) is given by

\[
m_{jk} = \frac{n(n - 1)}{2}
\]

For all \( m \) combinations, \( r \), \( \text{theta} \), and \( \text{phi} \) differences (\( \Delta \)) of beads \( j \) and \( k \) for both sets are calculated. This gives an idea of how beads \( j \) and \( k \) changed relative to each other through impact with the cooled copper mirror. \( \Delta \text{Theta} \) gives a measure of any twist motion resulting from the impact. It was assumed that no significant rotational effects parallel to the mirror surface should be induced. \( \Delta \text{Phi} \) gives a measure of compression in the Z-direction. For instance, if the original \( \text{phi} \) angle was 45 degrees, and after impact it is now 0 degrees, \( \Delta \text{phi} \) will be -45 degrees. This would indicate that two beads were at different Z-planes, but the impact brought them to the same plane. This was assumed to happen, although perhaps not to such a dramatic extent. It is important to note that neither \( \text{theta} \) nor \( \text{phi} \) give information regarding distance between the points. This is given by \( r \). The difference between post- and pre-frozen \( r \) gives a change in relative distance between the two beads. Furthermore, the strain can also be calculated. The strain, \( SR \), is defined as

\[
SR = \frac{r_{jk}^{\text{post}} - r_{jk}^{\text{pre}}}{r_{jk}^{\text{pre}}}
\]
In the above equation, the numerator is $\Delta r$. This quantity is divided by the original relative distance between the two beads to give a percent stretch induced from the impact. All these parameters are calculated for all $m$ combinations of beads in the run. Average $\Delta \theta$, $\Delta \phi$, and $SR$ are also calculated for each run. All respective averages within a set are, in turn, averaged to get an overall quantifiable measure for each impact set.

Figure 23: Description of spherical coordinate system.

6.2 VERIFICATION OF AUTOMATED BEAD CENTER CALCULATION

Most of the algorithm developed to analyze the images relies on the calculation of the bead centers. If this calculation is not accurate, the results will not reflect the sample’s true response to the impact. The accuracy of the algorithm’s ability to calculate the center frame of a bead was tested against a human’s ability. A volunteer looked through a set of images, and was asked to pick the center frames of ten random beads. This was performed a total of three times for the same ten beads, and the results averaged. The algorithm was also run on the same image
set, and the center frames calculated were compared to those of the volunteer. The results were converted to microns for better quantifying, and are shown in Figure 24.

![Figure 24: Verification of accuracy for microsphere center calculation with MATLAB algorithm.](image)

The algorithm consistently calculated a bead’s center an average of 1.75 (± 0.7) µm higher (closer to the surface) than the volunteer. This verifies that the algorithm is capable to calculate centers of beads at accuracy levels comparable to a human.
7 METHODS

7.1 CALIBRATION AND DUAL-PHASE VELOCITY PROFILE

The touch freezer mirror module was cooled by filling the polystyrene basin with LN. The mirror module was left to cool to 77K (approximately 5 min). The mirror module was then rotated, exposing the mirror to be used so it could be calibrated. The dual-phase velocity profile was then programmed (with the impact velocity to be tested) into LinMot Talk as described in Section 3.1, with one modification. Namely, a distance, $d^*$, was added to the first phase distance, $d_1$. This would ensure complete and intimate contact with the mirror was made during the second phase velocity, rather than during the final deceleration. Table 2 shows a summary of parameters used for the impact freezing testing.

<table>
<thead>
<tr>
<th>Impact Vel (m/s)</th>
<th>Max Vel (m/s)</th>
<th>Accel (m/s²)</th>
<th>Decel (m/s²)</th>
<th>d* (mm)</th>
<th>Accel (m/s²)</th>
<th>Decel (m/s²)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5</td>
<td>4.5</td>
<td>100</td>
<td>150</td>
<td>1</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4.5</td>
<td>100</td>
<td>150</td>
<td>1</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>4.5</td>
<td>100</td>
<td>150</td>
<td>1</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>4.5</td>
<td>100</td>
<td>150</td>
<td>1.5</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>

After calibration, the mirror module was rotated, resubmerging the mirror in LN to keep it cool until the sample was prepared.

7.2 SAMPLE PREPARATION

Laponite powder (Southern Clay Products, Gonzales, TX) was added to deionized water at a 10% (w/v) concentration. Before mixing, a 5 µL/ml concentration of 3 µm diameter polystyrene microspheres (Polysciences, Inc., Warrington, PA) was also added. The solution was
agitated continuously for 10 minutes to ensure the powder fully wets out without any clump formation.

A column of the prepared Laponite gel (approximately 5 mm diameter by 2 mm height) was formed onto a circular glass slide (Figure 25). The gel column sat on the slide within a small plastic O-ring. The O-ring was secured onto the slide with epoxy. It acted as a hard stop, ensuring that all the samples get compressed to a consistent thickness.

The slide and sample fit in a self locking base of a FSC2 microchamber (Bioptechs, Butler, PA). They were sandwiched between two annular layers of silicon (Figure 26). The top layer has a thicker annular silicon piece glued on top of it, and a piece of glass that covers the top surface. This formed a cover that fit over the O-ring and sample. When the chamber was covered and locked, the silicon rings were pressed against the slide, completely sealing the sample within this tight space. This minimizes the volume of air the sample was exposed to and consequently, any resulting dehydration effects.
Figure 26: Sample setup (a) From left to right: FCS2 chamber base, lower layer of silicon, sample on slide with O-ring, upper silicon layer with thicker layer (orange) and glass cover creating the space for the sample to occupy, and FCS2 chamber locking cover. (b) All pieces of (a) assembled together, in same order.

Before the sample/slide assembly was sealed in the chamber, glass fiber particles were scattered with a wooden stick. These functioned as markers for finding a region to take images of. The sealed chamber was placed on the DIC microscope. The top surface of the gel (opposite the glass surface) was scanned at 10x magnification to find a region of interest. Once a region was found, a picture was taken to be used as a reference for finding the same region on the post frozen sample. Figure 27a shows a sample region before impact freezing. The region of interest was scanned at 20x magnification to find an adequate region for taking images. A set of 50 images were taken of the region at 0.5 µm depth intervals.
Figure 27 Example DIC images of (a) pre- and (b) post-frozen sample. The glass fiber pattern is used to find the same region of interest. Bar = 40 µm.

7.3 TOUCH FREEZING

Once the image scan was complete, the sample was carefully taken out of the chamber, and quickly transferred to the touch freezer to minimize sample dehydration. It was attached to a sample stack comprising of the magnetic mount, double sided tape, and foam (Figure 28a). The sample was pressed onto the exposed tape surface, and the whole stack mounted onto the touch freezer (Figure 28b).

The command to initiate the touch freeze sequence was given via pushbutton on the control panel, whereupon the sample begins to creep at 1 cm/s towards the laser sensor. Once the beam is interrupted, the motor followed the dual-phase velocity profile and brought the sample into complete contact with the cool copper mirror. The digital oscilloscope was used to record position and velocity data for verifying the impact velocity on contact.

Contact with the mirror was held for 7 seconds to ensure the whole sample gets frozen. The motor then backed the sample 20 mm away from the mirror for 10 seconds to allow the user to remove the stack from the motor slider. The slider then went to its home position, and the mirror could be resubmerged into the LN.
Figure 28: (a) Stack for mounting sample onto the motor slider. (b) Sample attached to stack and mounted onto motor slider

7.4 REIMAGING

Once the sample was frozen, it was allowed to come to room temperature for 4 minutes before being replaced into the sealed chamber and the DIC microscope for imaging. The sample was scanned at 10x magnification looking for the same glass fiber pattern (Figure 27). Once the region was found, a confirmation reference picture was taken.

The sample was then viewed at 20x magnification and the same region was found. The same imaging steps as described in Section 7.2 were followed for capturing another Z-scan image set.

7.5 POST-PROCESSING

After both image sets were captured, they were analyzed using the set of custom algorithms described in Section 6.1 using MATLAB.
7.6 CONTROL

A control sample was used to measure the thrust effects of the first phase of the velocity profile on the sample. To measure this, the sample was programmed to follow only the first phase of the dual-phase velocity profile, as the acceleration and deceleration achieved in this phase were significantly greater than those of the second phase. To restrict the measurement to only thrust effects, the sample did not make contact with the mirror.

8 RESULTS

8.1 SECOND PHASE AND IMPACT VELOCITY CURVES

Using the data from the digital oscilloscope, curves were generated to verify the 0.1 second duration of the second phase. Figure 29 shows a compilation of the second phase duration for all the tested secondary velocities sets. Curves of second phase duration for individual sets are shown in Figure 30a to Figure 33a, for 2.5, 5, 10, and 30 cm/s respectively.
Figure 29: Second phase duration verification plots. Overlaid second phase curves of (a) all and (b) without the 30 cm/s impact velocity runs.

Curves were also generated to verify the impact velocities of each set. The final 2 mm travel distance before the sample comes to a stop for each set is shown in Figure 30b to Figure 33b, for 2.5, 5, 10, and 30 cm/s respectively. The red arrow indicates the approximate region where impact occurs, namely, $d^*$ mm from the end of the second phase. This is based on distance, $d^*$, which was added to the first phase distance, $d_f$ (see Section 7.1).
Figure 30: (a) Second phase curves and (b) final 2 mm of travel for the 2.5 cm/s impact velocity runs. Arrow points to approximate position of sample during impact.
Figure 31: (a) Second phase curves and (b) final 2 mm of travel for the 5 cm/s impact velocity runs. Arrow points to approximate position of sample during impact.
Figure 32: (a) Second phase curves and (b) final 2 mm of travel for the 10 cm/s impact velocity runs. Arrow points to approximate position of sample during impact.
Figure 33: (a) Second phase curves and (b) final 2 mm of travel for the 30 cm/s impact velocity runs. Arrow points to approximate position of sample during impact.

Oscillatory peak amplitude for the latter half of the second phase each profile are shown in Table 3.
Table 3 – Peak amplitudes during the latter half of the second phase profiles. *This run produced shifted sets of curves. One set had 1.5 cm/s amplitudes, and the other had 1.0 m/s. (see Section 9.2 for details)

<table>
<thead>
<tr>
<th>Impact Velocity cm/s</th>
<th>Peak Amplitude cm/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5</td>
<td>± 0.5</td>
</tr>
<tr>
<td>5</td>
<td>± 1.5 *</td>
</tr>
<tr>
<td>10</td>
<td>± 1.0</td>
</tr>
<tr>
<td>30</td>
<td>± 1.0</td>
</tr>
</tbody>
</table>

Results from the MATLAB analysis, namely averaged SR, absolute change in theta, and absolute change in phi, for each impact velocity, along with their respective standard deviations for each impact velocity tested and the control are summarized in Table 4.

Table 4: Averages and standard deviations of strain (SR), absolute change in theta (Abs Theta), and absolute change in phi (Abs Phi) for all impact velocities and control.

<table>
<thead>
<tr>
<th>Impact Velocity</th>
<th>SR</th>
<th>Abs Theta (deg)</th>
<th>Abs Phi (deg)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5 mm</td>
<td>0.247</td>
<td>4.014</td>
<td>3.172</td>
</tr>
<tr>
<td>Average</td>
<td>0.061</td>
<td>0.986</td>
<td>0.967</td>
</tr>
<tr>
<td>5 mm</td>
<td>0.292</td>
<td>7.947</td>
<td>4.084</td>
</tr>
<tr>
<td>Average</td>
<td>0.096</td>
<td>3.213</td>
<td>1.200</td>
</tr>
<tr>
<td>10 mm</td>
<td>0.342</td>
<td>6.929</td>
<td>4.773</td>
</tr>
<tr>
<td>Average</td>
<td>0.130</td>
<td>3.152</td>
<td>0.978</td>
</tr>
<tr>
<td>30 mm</td>
<td>0.714</td>
<td>8.600</td>
<td>6.466</td>
</tr>
<tr>
<td>Average</td>
<td>0.241</td>
<td>3.154</td>
<td>2.333</td>
</tr>
<tr>
<td>Control</td>
<td>0.003</td>
<td>0.336</td>
<td>0.360</td>
</tr>
<tr>
<td>Average</td>
<td>0.001</td>
<td>0.187</td>
<td>0.041</td>
</tr>
</tbody>
</table>

Figure 34 shows the averaged SR for the control and each impact velocity from Table 4. Standard deviation of average SR for each set is shown in the y-axis error bars. The control’s error bars are too small to be visible in this scale.
A two-tailed, unequal variance t-test was used to evaluate the significant difference in average SR with respect to the impact velocities tested. The results of the t-test are shown in Table 5.

Table 5: Statistical difference results (P-values) between impact velocities and control.

(T-test: two-tailed, unequal variance)

<table>
<thead>
<tr>
<th>(cm/s)</th>
<th>2.5</th>
<th>5</th>
<th>10</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.365</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0.152</td>
<td>0.468</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>0.011</td>
<td>0.014</td>
<td>0.022</td>
<td></td>
</tr>
<tr>
<td>Control</td>
<td>0.00086</td>
<td>0.000706</td>
<td>0.001389</td>
<td>0.002739</td>
</tr>
</tbody>
</table>
9 DISCUSSION

9.1 TOUCH FREEZING

The modifications made to the original touch freezer did not affect its freezing performance, but rather, improved the overall use of the apparatus. The simpler mirror module significantly minimized the amount of LN required to bring the mirror module to 77K (from 15 L to 2 L). The custom control panel made running experiments faster and easier. Finally, the hard stop created the sense of safety with operating the device.

9.2 RESULTS

9.2.1 Motor Accuracy

Motor testing with various first phase parameter combinations for the dual-phase velocity profiles showed the most repeatable results at 4.5 m/s, 100 m/s², and 150 m/s² for peak velocity, acceleration, and deceleration, respectively. Specifically, accuracy and repeatability for these parameters at were measured to be 70 and 17 µm, respectively. These values were well within the advertised ranges of the device, and acceptable for this investigation.

9.2.2 Dual-Phase Velocity Profile

Overlaying the curves for the second phases of the experiment runs (Figure 29a) shows the 10 and 30 cm/s velocities clearly distinguishable from each other. Omitting the 30 cm/s curves (Figure 29b) shows that the 2.5 and 5 cm/s curves intersect during the first half of the second phase, but not during the latter half, when impact occurs. The near intersection in the latter half, however, leads to the conclusion that the 2.5 cm/s velocity difference seems to be approaching the resolving capabilities of the LinMot, for low velocities, at least.
Figure 29a allows qualitative comparison of the four sets of curves since they are all presented on the same scale. All the curves exhibit an oscillatory period, but only the 2.5 and 30 cm/s ones settles after the during the latter half of the second phase. The 5 and 10 cm/s curves seem to keep their oscillatory effect throughout the whole secondary phase. This was also the case for the initial velocity profile evaluation (see Section 3.4). A closer look at the 5 cm/s curves (Figure 31) reveals that they are two separate sets of curves separated by a 90° phase shift. Observation of the impact velocity (red arrow) shows that contact with the mirror was made at a “node” where the two sets of curves intersect. This is coincidental.

The peak amplitudes from the experiments (Table 3) match those of the initial evaluation (Table 1). One set of the identified 5 cm/s curves has a peak amplitude of 1.5 cm/s, however. Neither case will lead to different programmed velocity profiles making contact with the mirror at the same velocity. This is in agreement with the initial velocity evaluation (Section 3.4).

The contact velocities of the runs (Figure 30b - Figure 33b) are all within the amplitude boundaries. The 2.5 and 5 cm/s sets are fairly close, however. It is concluded, therefore, that 2.5 cm/s seems to approach the resolving ability of the LinMot. This is not discouraging. LinMot motors are not designed for high precision velocity control. They are meant for applications where position control is required. Thus, incorporating a motor with higher quality velocity control could allow for even lower impact velocities, assuming they can also reach the higher velocities.

9.2.3 MATLAB Analysis

The algorithm written by the author to calculate bead centers was verified to be comparable to a human in accuracy and superior in time efficiency. Figure 24 shows that the
algorithm calculated bead centers an average of 1.75 (± 0.7) µm closer to the surface than the volunteer. There was no need to adjust for the offset, as this was going to affect every set.

Finding corresponding beads between pre- and post-frozen sets was done manually. This process requires caution to ensure that truly corresponding beads were matched. Typically, beads were identified by patterns that were present in both sets (refer to Figure 22). If there was any doubt as to whether a bead was the same as one from the other set, it was not included.

9.2.4 Sample

The micron-infused Laponite sample gel was an ideal assay for testing impact effects. First, the sample was transparent, which permitted it to be imaged with DIC microscopy. Second, the shear-thinning property of the Laponite makes it a very sensitive method for measuring distortion as a function of increasing velocities, since even the slightest shear will cause the gel to flow. Consequently, a larger shear will result in a larger flow, which will translate into a larger deformation.

Third, both the freezing and impact effects needed to be captured, as both are present when impact freezing biological samples. The Laponite incurred both, impact effects from contact with the mirror, and freezing effects from the contact with a cooler metal. However, once the impact force stopped, the Laponite restructured in its distorted configuration. This allowed it to thaw (so it can turn transparent again) while maintaining its frozen and distorted structure. This also avoided introduction of new artifact (e.g. from preparation for EM observation) since the sample was viewed immediately after freezing and thawing.

Maintaining the impact effects until observation is not easy to achieve with other natural and biological samples. Several biological samples, for instance, are not transparent, resulting in the need for EM observation, which again, can introduce other distortion that might not
necessarily be distinguishable from impact artifact. Those that are transparent, and other elastic
samples, might indeed freeze normally. However, in order to avoid EM preparation artifact, the
samples would have to be thawed like the Laponite. During the thawing process, though, they
would recover their original structure.

Other samples, still, do not have any uniform or repeating pattern on the microscopic
level which can be measured. Bennett used the lattice spacing between thick and thin muscle
fibers to estimate compression during impact [72]. Although a measureable pattern exists, the
sample required EM observation, which, as discussed, has its own drawbacks.

Sample dehydration was also minimized with the incorporation of the sealed FCS2
chamber. Any dehydration that occurred would have done so primarily during placement of the
sample onto the motor to freeze it and the time to thaw. Both processes were kept consistent to
expose every sample to the same dehydration effects. The samples were placed on the touch
freezer and frozen within 15 seconds of removing from the FCS2 chamber, and then allowed to
thaw for 4 minutes.

9.2.5 Strain

The shear-thinning property of the Laponite translated to a displacement of the original
microsphere positions within the sample when a force is applied. This, in turn was converted to a
change in distance between all bead combinations. Calculating the percent elongation or
compression (SR) gave a quantitative metric of impact freezing distortion that could be
compared with various impact velocities.

The three slowest impact velocities did not show a statistically significant difference
between themselves. However, their averages did exhibit an increasing trend. The lack of
statistical significance suggests that the assay developed is not sensitive enough resolve deformations at such small impact velocity differences. Perhaps different concentrations of Laponite or different size microspheres can increase this sensitivity.

The 30 cm/s impact velocity, however, was statistically significantly different from all three other velocities and the control. It also followed the increasing linear relationship ($R^2 = 0.993$) of the other impact velocities. This verifies the hypothesis that mechanical deformation is proportional to impact velocity.

The control showed a significantly smaller strain and was well below the trend of the other impact velocities. This indicates that no mechanical deformation is induced by the acceleration and deceleration provided by the first phase (100 and 150 m/s$^2$, respectively).

9.3 HIGHER IMPACT VELOCITIES

Although the assay developed may have not been sensitive enough for the slower velocities, it was in fact too sensitive for higher impact velocities. Impact velocities greater than 30 cm/s were too difficult to test. The slower velocities (2.5, 5, and 10 cm/s) were originally going to be compared to the DDK Cryogun (pneumatically driven impact freezer), but distortions so great were produced with the device, that corresponding matches between sets could not be found. Even the glass fibers would be so significantly rearranged from the impact, that in some cases, even the same region of interest could not be found. Figure 35 shows an example of a pre- and post-frozen region of interest from a run with the DDK. Similar examples are shown in Figure 36-Figure 40 for the impact velocities tested, as well as the control. The same trend is seen as with the point cloud analysis, namely, that the degree of glass fiber distortion is proportional to the impact velocity. The control, for example, shows no visual difference in the fiber pattern.
The velocity of the Cryogun plunger was investigated, as a result of such great distortions it caused even in the glass fibers. A high speed camera (500 fps) was set up to capture its plunger in motion. With the suggested air pressure (60 psi), the Cryogun trigger was pulled. This established a connection between two contacts mounted on the device which triggered the camera to capture the images. By finding displacements of the plunger between a set of frames, and knowing the frame rate, a velocity could be calculated. At 60 psi, the Cryogun’s impact velocity was calculated to be 2.3 m/s. This falls within the range of conventional impact freezer velocities.
Figure 35: Glass fiber distortion between (a) pre- and (b) post-frozen samples following impact freezing with the DDK Cryogun (impact velocity = 2.3 m/s). The fibers are greatly distorted. Bar = 40µm.
Figure 36: Glass fiber distortion between (a) pre- and (b) post-frozen samples following impact freezing at 30 cm/s. Bar = 40µm.
Figure 37: Glass fiber distortion between (a) pre- and (b) post-frozen samples following impact freezing at 10 cm/s. Circled fiber is the same in the images. Bar = 40µm.
Figure 38: Glass fiber distortion between (a) pre- and (b) post-frozen samples following impact freezing at 5 cm/s. Bar = 40µm.
Figure 39: Glass fiber distortion between (a) pre- and (b) post-frozen samples following impact freezing at 2.5 cm/s.

Significantly less fiber distortion compared to the Cryogun (Figure 35). Bar = 40μm.
Figure 40: Glass fiber distortion between (a) pre- and (b) post-imaging of control.

No bead distortion is discernable. Bar = 40µm.
9.4 IMPLICATIONS

A quantifiable assay and method for testing impact velocity effects on mechanical distortions, specific for impact freezing has not been developed before. Verifying the hypothesis that impact velocity is indeed proportional to mechanical distortion sheds some light on impact freezing. Although this doesn’t necessarily imply that all previous work with impact freezing is void, it does indicate not only the importance of proper design of impact freezers, but also the caution with which EM observation of impact frozen samples should be conducted. Impact freezing is used as cryofixation technique for freeze-etching, freeze-fracturing, freeze substitution, and cryoEM. Many areas of research incorporate impact freezing into their protocols. Such an assay, dual-phase profile, and analysis method can be used to test and evaluate other impact freezers based on the mechanical deformation they introduce. This can aid in minimizing one artifact-inducing step in EM, and lead to more accurate observations.

9.5 FUTURE WORK

Touch freezing of biological samples has yet to be tested with the dual-phase velocity profile. However, it is necessary to ensure that the sample does not precool during the second phase approach. Therefore, freezing of biological tissue with the dual-phase velocity profile needs to be performed. Freeze substitution or etching and rotary replication following freeze-fracture can be used to dehydrated the frozen sample in preparation for EM observation to verify freezing quality.

9.6 CONCLUSION

It was stated in the Introduction that artifact cannot be completely, but rather, only minimized. This thesis aimed at minimizing artifact induced through impact freezing.
A recently designed “touch” freezer was modified. The modifications made to the original touch freezer permitted the testing of a dual-phase velocity profile to measure the effects of mirror impact with respect to resulting mechanical distortion. The principle and unforeseeable advantage of the device is the combination of its ability to modulate approach and contact velocities to match profiles known to minimize distortion of the sample.

A dual-phase velocity profile has been introduced for bringing a sample into contact with a metal mirror, in a controlled manner, for impact freezing. It can accelerate a sample at 4.5 m/s through the cold cryogen vapor region, and then decelerate to as slow as 2.5 cm/s for the final 0.1 s of travel, before making gentle, controlled impact with the mirror surface.

An assay has been developed to quantify the mechanical distortion introduced to a sample through contact with a metal mirror during impact freezing. The shear-thinning property of Laponite makes it a very sensitive method for measuring distortion. It also maintains its deformed structure, allowing for direct observation instead of further processing for EM observation.

The hypothesis that mechanical distortion of a sample frozen by impact freezing is proportional to impact velocity was verified. This can be used to evaluate other designs based on the mechanical deformation they introduce, which can aid in the minimization of artifact introduction into the EM preparation process.
REFERENCES


APPENDIX

Loading Gel Onto Glass Slide .................................................................................................... - 1 -
Touch Freezer Parts List ............................................................................................................. - 2 -
Touch Freezer Detailed Drawings .............................................................................................. - 3 -
MATLAB Algorithms .............................................................................................................. - 20 -
    Master File ........................................................................................................................ - 20 -
    Create New Bead Template ............................................................................................... - 24 -
    create_raw_matrix ............................................................................................................. - 26 -
    remove xcorr2_duplicates ............................................................................................... - 28 -
    saveRawMatrixFile .......................................................................................................... - 31 -
    sort_matrix ...................................................................................................................... - 33 -
    transpose_matrix ............................................................................................................. - 36 -
    DeleteEmptyExcelSheets ................................................................................................. - 37 -
    calculate_center_frame ................................................................................................. - 39 -
    get_final_beads ............................................................................................................... - 39 -
    get_matching_beads_v4 ................................................................................................. - 42 -
    convert_to_spherical ....................................................................................................... - 48 -
Loading Gel Onto Glass Slide

Figure A1: Loading the (Laponite) gel onto the glass slide. (a) Requires a 1-1.5 mm thick piece of polydimethylsiloxane (PDMS) with a 5 mm hole in the center. (b) Place PDMS on the O-ring that is epoxied onto the glass slide, ensuring that the holes are centered. Hold PDMS in place with a pair of forceps. (c) While holding down the PDMS, press gel into its hole. Care must be taken not to pack too much gel into the hole, otherwise it will spread out below the PDMS surface. (d) While holding down one corner of the PDMS, smoothly scrape across the top surface, creating a clean and even surface. (e) Carefully grasp one corner of PDMS and gently remove with a peeling action. This will leave a column of gel on the glass slide centered within the O-ring. (f) Dip wooden stick into container of glass fibers and then sprinkle them on the top surface of the gel sample. Best results were noticed when stick was held 1-2 inches over the sample and tapped to dislodge fibers from the wooden surface.
**Touch Freezer Parts List**

<table>
<thead>
<tr>
<th>Vendor</th>
<th>Name</th>
<th>Description</th>
<th>Qty Needed</th>
<th>Model #</th>
<th>P/N</th>
<th>Price/Qty</th>
<th>Qty Order</th>
<th>Total Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>High performance stator</td>
<td>Stator with male 9-pin conn</td>
<td>1</td>
<td>PS01-32X460F-HP-R</td>
<td>0150-1214</td>
<td>$699.70</td>
<td>1</td>
<td>$699.70</td>
</tr>
<tr>
<td>L</td>
<td>High performance slider</td>
<td>Slider, high performance</td>
<td>1</td>
<td>PLU1-12480/440-HP</td>
<td>0150-1524</td>
<td>$507.70</td>
<td>1</td>
<td>$507.70</td>
</tr>
<tr>
<td>M</td>
<td>Flange mount</td>
<td>Flange 2 x 170 mm</td>
<td>1</td>
<td>PF22-234170</td>
<td>0150-2117</td>
<td>$51.65</td>
<td>1</td>
<td>$51.65</td>
</tr>
<tr>
<td>M</td>
<td>Controller</td>
<td>Controller (7xV/1xA)</td>
<td>1</td>
<td>E1200-GP-HC</td>
<td>0150-1668</td>
<td>$989.80</td>
<td>1</td>
<td>$989.80</td>
</tr>
<tr>
<td>M</td>
<td>Extension cable W/ 2 m</td>
<td>1</td>
<td>0150-2119</td>
<td>$149.40</td>
<td>1</td>
<td>$149.40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>Power supply</td>
<td>Power supply (72V/300W)</td>
<td>1</td>
<td>PF1-122300</td>
<td>0150-1920</td>
<td>$388.30</td>
<td>1</td>
<td>$388.30</td>
</tr>
<tr>
<td>M</td>
<td>Configuration cable</td>
<td>PC to servo config cable</td>
<td>1</td>
<td>RS232 PC cable</td>
<td>0150-3009</td>
<td>$10.70</td>
<td>1</td>
<td>$10.70</td>
</tr>
<tr>
<td>M</td>
<td>Lubricant</td>
<td>Linear motor lubricant (50g)</td>
<td>1</td>
<td>LUB2-50</td>
<td>0150-1954</td>
<td>$21.50</td>
<td>1</td>
<td>$21.50</td>
</tr>
</tbody>
</table>

**Keyence**

<table>
<thead>
<tr>
<th>Vendor</th>
<th>Name</th>
<th>Description</th>
<th>Qty Needed</th>
<th>Model #</th>
<th>P/N</th>
<th>Price/Qty</th>
<th>Qty Order</th>
<th>Total Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>Laser Sensor Head</td>
<td>30 mm Wide Beam</td>
<td>1</td>
<td>LV-H300</td>
<td>$299.00</td>
<td>1</td>
<td>$299.00</td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>Laser Amplifier</td>
<td></td>
<td>1</td>
<td>LV-S1M</td>
<td>$299.00</td>
<td>1</td>
<td>$299.00</td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>Mounting Brackets</td>
<td>Height-adjustable brackets, 2 pcs</td>
<td>1</td>
<td>LV-8302</td>
<td>$13.00</td>
<td>1</td>
<td>$13.00</td>
<td></td>
</tr>
</tbody>
</table>

**McMaster**

<table>
<thead>
<tr>
<th>Vendor</th>
<th>Name</th>
<th>Description</th>
<th>Qty Needed</th>
<th>Model #</th>
<th>P/N</th>
<th>Price/Qty</th>
<th>Qty Order</th>
<th>Total Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>Bearing 3/8&quot; OD, 3/8&quot; ID, 1/8&quot; thick</td>
<td>2</td>
<td>897415</td>
<td>$18.66</td>
<td>1</td>
<td>$37.32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>Bearing Housing 3/8&quot; ID</td>
<td>2</td>
<td>98042</td>
<td>$29.33</td>
<td>1</td>
<td>$58.66</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>Shift 25\°, 24&quot; Travel</td>
<td>1</td>
<td>8350718</td>
<td>$55.52</td>
<td>1</td>
<td>$55.52</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>Ceramic Standoff 1/4-20, 1.5\°</td>
<td>1</td>
<td>94335A152</td>
<td>$4.88</td>
<td>1</td>
<td>$24.40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>Handle 1/4-20 Thread</td>
<td>1</td>
<td>5417968</td>
<td>$1.76</td>
<td>1</td>
<td>$1.76</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>Hex Cap Screws 1/4-20, 3/8\°</td>
<td>4</td>
<td>92240A355</td>
<td>$5.96</td>
<td>50</td>
<td>$298.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>Phillips Screws 6-32, 7/16\°</td>
<td>8</td>
<td>91772A149</td>
<td>$4.13</td>
<td>100</td>
<td>$413.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>Nuts 6-32, 7/64\°</td>
<td>8</td>
<td>94600A100</td>
<td>$7.13</td>
<td>50</td>
<td>$356.50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>Flathead Screws M4 x 30mm</td>
<td>3</td>
<td>91420A235</td>
<td>$5.20</td>
<td>100</td>
<td>$520.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>Nuts M4 x 3.2\°</td>
<td>3</td>
<td>94150A355</td>
<td>$2.85</td>
<td>50</td>
<td>$142.50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>Socket Head Screw M4 x 45mm</td>
<td>2</td>
<td>91292A133</td>
<td>$7.17</td>
<td>50</td>
<td>$358.50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>Socket Head Screw M5 x 45mm</td>
<td>2</td>
<td>91292A135</td>
<td>$6.59</td>
<td>25</td>
<td>$164.75</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>Washer M5</td>
<td>2</td>
<td>93475A240</td>
<td>$2.81</td>
<td>100</td>
<td>$281.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>Socket Head Screw 1/4-20, 1.75\°</td>
<td>3</td>
<td>92198A446</td>
<td>$6.51</td>
<td>25</td>
<td>$162.75</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>Nuts 1/4-20 x 7/32\°</td>
<td>3</td>
<td>91840A029</td>
<td>$4.28</td>
<td>100</td>
<td>$428.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>Washer 1/4\°</td>
<td>3</td>
<td>95229A029</td>
<td>$2.94</td>
<td>100</td>
<td>$294.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>Flathead Socket Screws 1/4-20 x 5/8\°</td>
<td>3</td>
<td>92210A539</td>
<td>$7.49</td>
<td>50</td>
<td>$374.50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>Soft Tip Screw M4</td>
<td>4</td>
<td>90418A223</td>
<td>$9.26</td>
<td>5</td>
<td>$46.30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>Cone Point Set Screw M4</td>
<td>1</td>
<td>93212A205</td>
<td>$1.50</td>
<td>25</td>
<td>$37.50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>Cup Point Set Screw M4</td>
<td>1</td>
<td>91380A010</td>
<td>$3.91</td>
<td>100</td>
<td>$391.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>Aluminum Screw 1/4-20 x 3/4\°</td>
<td>1</td>
<td>93143A540</td>
<td>$6.78</td>
<td>25</td>
<td>$169.50</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**80/20**

<table>
<thead>
<tr>
<th>Vendor</th>
<th>Name</th>
<th>Description</th>
<th>Qty Needed</th>
<th>Model #</th>
<th>P/N</th>
<th>Price/Qty</th>
<th>Qty Order</th>
<th>Total Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>1 inch square 80/20</td>
<td>128 inches</td>
<td>1015</td>
<td>$0.23</td>
<td>128</td>
<td>$29.44</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>1 inch square Machining of 80/20</td>
<td>14 sections</td>
<td>1</td>
<td>$1.95</td>
<td>14</td>
<td>$27.30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>Tri-corner connector</td>
<td></td>
<td>4</td>
<td>Frame</td>
<td>4042</td>
<td>$17.85</td>
<td>1</td>
<td>$17.85</td>
</tr>
<tr>
<td>T</td>
<td>hardware connector supporting hardware</td>
<td>12</td>
<td>Frame</td>
<td>3017</td>
<td>$0.53</td>
<td>1</td>
<td>$0.53</td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>90 degree inner bracket</td>
<td>28</td>
<td>Frame</td>
<td>4119</td>
<td>$2.90</td>
<td>1</td>
<td>$2.90</td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>hardware bracket supporting hardware</td>
<td>56</td>
<td>Frame</td>
<td>3393</td>
<td>$0.40</td>
<td>1</td>
<td>$0.40</td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>3-hole joining strip</td>
<td>4</td>
<td>Frame</td>
<td>4118</td>
<td>$4.30</td>
<td>1</td>
<td>$4.30</td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>hardware strip supporting hardware</td>
<td>12</td>
<td>Frame</td>
<td>3321</td>
<td>$0.50</td>
<td>1</td>
<td>$0.50</td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>60 degree bracket</td>
<td>4</td>
<td>Frame</td>
<td>4143</td>
<td>$4.90</td>
<td>1</td>
<td>$4.90</td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>hardware bracket supporting hardware</td>
<td>16</td>
<td>Frame</td>
<td>3321</td>
<td>$0.50</td>
<td>1</td>
<td>$0.50</td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>30-degree bracket</td>
<td>4</td>
<td>Frame</td>
<td>4164</td>
<td>$5.10</td>
<td>1</td>
<td>$5.10</td>
<td></td>
</tr>
</tbody>
</table>

**80/20 Total** $313.80

**Grand Total** $3,875.52
Touch Freezer Detailed Drawings

This section includes the Touch Freezer parts that were modeled and machined.

Figure A2: Overview of designed and machined parts for Touch Freezer. The support brace is also included, but not pictured.
Motor Mount Base
MATLAB Algorithms

Master File

%%% Master File
%%% Nector Ritzakis
%%% 20 June 2011
%%% Final Rev 20 July 2011

Cleanup
clear all
close all
clc

Set Directories for Gettin Raw Images From
% User defined original directory
fprintf('Select Original Folder
')
original_folder_path = uigetdir;
clc
% User defined destination directory
fprintf('Select Destination Folder
')
destination_folder_path = uigetdir;
clc

Define Image Details
% User enters secondary speed of run
distance = input... ('Enter secondary distance value ONLY - no decimals\n(or DDK/Thrust)\n','s');
distance = ['_' distance];
clc
% User defined run
run = input('Enter run: 1, 2, or 3\n','s');
run = ['mm_run' run];
clc
% User defined either before or after set, and set that as title
b_or_a = input... ('Is this a before or after set?\n[1] = Before\n[2] = After\n');
switch b_or_a
    case(1)
        set_title = 'before';
    case(2)
        set_title = 'after';
    otherwise
        error('Invalid Entry\n')
end
clc
% Define image extention type
original_ext = 'tif';
% Remove "mm" heading from title if DDK or THRUST run
same = strcmp(distance,'_DDK');
if same == 1
    run(1:2) = [];
end
same = strcmp(distance,'_Thrust');
if same == 1
    run(1:2) = [];
end
clc
% Define rename filetype same as original
ext = 'tif';

Compile Details to Define Names and Titles
% Compile parts and define wild name
Define Bead Template

% User inputs whether bead template already exists
beadExist = input('
'...\n'Does a bead template already exist?\n[y] - yes or [n] - no\n','s');
clc
ok = 5;
if beadExist == 'y'
[bead_title,bead_path] = uigetfile...
({'*.tif;*.jpg;*.jpeg;*.png;*.gif','All Image Files'},...
'Select bead image to use as template',destination_folder_path);
bead_path = [bead_path bead_title];
bead = imread(bead_path);
elseif beadExist == 'n'
bead = create_new_bead_template(ok,original_image_name_wild,...
original_folder_path,image_title,destination_folder_path);
else
error('Invalid entry');
end

Convert Raw Images to Gray and Move to Destination Folder

% Find all files with wild name in original folder
original_image_file = dir(original_image_name_wild);
% Create folder for raw images in destination folder
raw_folder_name = ['raw_' image_title];
mkdir(destination_folder_path,raw_folder_name);
% Define raw image folder path
raw_folder_path = fullfile(destination_folder_path,raw_folder_name);
% Define x for reversing image order
x = numel(original_image_file);
% Loop to convert original images to gray and move to raw folder
for k = 1:x
original_image_path = fullfile(original_folder_path,...
original_image_file(k).name);
original_image = imread(original_image_path);
image_info = imfinfo(original_image_path);
colorType = (image_info.ColorType == 'grayscale');
if all(colorType) ~= 1
original_image = rgb2gray(original_image);
end
original_image = im2double(original_image);
y = x - (k-1);
y_string = sprintf('%03.0f',y);
raw_image_path = [raw_folder_path '\ set_title y_string '.' ext];
imwrite(original_image,raw_image_path);
end
% Clear original image from workspace
clear original_image;
% Define wild name for search in raw folder
raw_wild_name = fullfile(raw_folder_path,raw_wild_name);

Rename Images

% Find all images with wild name in raw folder
raw_image_file = dir(raw_wild_name);
% Create folder for renamed images in destination folder
renamed_folder_name = ['renamed_' image_title];
mkdir(destination_folder_path,renamed_folder_name);
% Define renamed image folder path
renamed_folder_path = fullfile(destination_folder_path,renamed_folder_name);
% Loop to rename images and move them to renamed folder
for k = 1:numel(raw_image_file)
    raw_image_path = fullfile(raw_folder_path,raw_image_file(k).name);
    raw_image = imread(raw_image_path);
    k_string = sprintf('%03.0f',k);
    renamed_image_name = [image_title '_' k_string '.' ext];
    renamed_image_path = fullfile(renamed_folder_path,renamed_image_name);
    imwrite(raw_image, renamed_image_path);
end
% Clear raw image from workspace
clear raw_image;
% Define wild name for search in renamed folder
renamed_wild_name = fullfile(renamed_folder_path,wild_name);

Perform 2D Cross-Correlation to Find Bead Centroids in XY Plane
% Find all images with wild name in renamed folder
renamed_image_file = dir(renamed_wild_name);
% Create folder for cross-correlated Images in destination folder
correlated_folder_name = ['correlated_' image_title];
mkdir(destination_folder_path,correlated_folder_name);
% Define cross-correlated image folder path
correlated_folder_path = fullfile(destination_folder_path,...
correlated_folder_name);
% Create folder for outlined images in destination folder
outlined_folder_name = ['outlined_' image_title];
mkdir(destination_folder_path,outlined_folder_name);
% Define outlined image folder path
outlined_folder_path = fullfile(destination_folder_path,outlined_folder_name);
% Generate raw centroid matrix
rawCentroidMatrix = create_raw_matrix(bead,renamed_image_file,...
renamed_folder_path,image_title,ext,correlated_folder_path,...
outlined_folder_path);

Save Matrix Files
[finalVerticalMatrix,matrix_file_path,verticalMatrix] = saveRawMatrixFile...
(destination_folder_path,image_title,rawCentroidMatrix);

Calculate Center Frame
Create folder for plots in destination folder
plot_folder_name = ['plot_' image_title];
mkdir(destination_folder_path,plot_folder_name);
% Define plot image folder path
plot_folder_path = fullfile(destination_folder_path,plot_folder_name);
% Calculate center frame
resultMatrix = calculate_center_frame(verticalMatrix,matrix_file_path,...
plot_folder_path,b_or_a,image_title);
close all
clc
% Display center frame for each bead for user to select top surface
frame = resultMatrix(3:3:end).
fprintf('View calculated center frames for each bead,
')
fprintf('and decide what frame should be considered as "top"
')
% pause
% clc
% User inputs frame representing top of sample
fprintf('Enter frame number considered the top of the sample
')
top = input(' (ROUND DOWN FROM FRAME SELECTED)
');
top = 101-top;
clc

Filter Out Unusable Beads
[beadMatrix,beadMatrix_60frames] = get_final_beads(resultMatrix,...
finalVerticalMatrix,matrix_file_path,top);
% beadMatrix = cell2mat(beadMatrix);
Convert to matrix to plot data
beadMatrix_60frames = cell2mat(beadMatrix_60frames);

**Plot All Final Beads**
x = beadMatrix(3,:); y = beadMatrix(2,:);
plot(x,y,'.r')
axis([0 1392 0 1040])
xlabel('X-Pixel (Col)');
ylabel('Y-Pixel (Row)');
if b_or_a == 1
title_string = ['B' image_title(2:end) ' Final Beads - Full Scan'];
title(title_string,'Interpreter','none')
else
title_string = ['A' image_title(2:end) ' Final Beads - Full Scan'];
title(title_string,'Interpreter','none')
end
axis ij
hold on
for i = 1:size(beadMatrix,2)
    beadLabel = num2str(beadMatrix(1,i));
    text(beadMatrix(3,i),beadMatrix(2,i),beadLabel,'color','blue',...
         'VerticalAlignment','bottom','FontSize',7,'FontWeight','light')
end
hold off
final_bead_path = [destination_folder_path '\Full Scan Final Beads Plot '...
                   image_title '.jpeg'];
saveas(gcf,final_bead_path);
close all

**Plot Final Beads (Top 60 Frames)**
x = beadMatrix_60frames(3,:); y = beadMatrix_60frames(2,:);
plot(x,y,'.r')
axis([0 1392 0 1040])
xlabel('X-Pixel (Col)');
ylabel('Y-Pixel (Row)');
if b_or_a == 1
    title_string = ['B' image_title(2:end) ' Final Beads - Top 60 Frames(30um)'];
title(title_string,'Interpreter','none')
else
    title_string = ['A' image_title(2:end) ' Final Beads - Top 60 Frames(30um)'];
title(title_string,'Interpreter','none')
end
axis ij
hold on
for i = 1:size(beadMatrix_60frames,2)
    beadLabel = num2str(beadMatrix_60frames(1,i));
    text(beadMatrix_60frames(3,i),beadMatrix_60frames(2,i),beadLabel,...
         'color','blue','VerticalAlignment','bottom','FontSize',7,...
         'FontWeight','light')
end
hold off
final_bead_path = [destination_folder_path '\Top 60fr Final Beads Plot '...
                   image_title '.jpeg'];
saveas(gcf,final_bead_path);
close all

**Save Workspace to Destination Folder**
save([destination_folder_path '\Workspace_' image_title '.mat']);

Published with MATLAB® 7.10
Create New Bead Template

Nector Ritzakis Created: 12-Sep-2011

```matlab
function bead = create_new_bead_template(ok,original_image_name_wild,...
    original_folder_path,image_title,destination_folder_path)
    clc
    % User input image for bead selection
    bead_image_number = input...
        ('Input ORIGINAL IMAGE NUMBER for selecting bead template:\n');
    clc
    % Find all files with wild name in original folder
    bead_image_file = dir(original_image_name_wild);
    % Find user specified image
    bead_image_title = bead_image_file(bead_image_number).name;
    % Create image filepath
    bead_image_path = fullfile(original_folder_path,bead_image_title);
    % Load input image and convert to grayscale
    bead_image = imread(bead_image_path);
    image_info = imfinfo(bead_image_path);
    colorType = (image_info.ColorType == 'grayscale');
    if all(colorType) ~= 1
        bead_image = rgb2gray(bead_image);
    end
    clc
    % Display image and wait for user to zoom into appropriate bead
    figure(1), imshow(bead_image)
    fprintf('Zoom to the bead and press enter\n');
    pause
    clc
    % User selects opposite corners of bead
    fprintf('Select UPPER LEFT and LOWER RIGHT corners of bead\n');
    [x,y] = getpts;
    % Round to get pixels selected
    x = round(x);
    y = round(y);
    % Create bead image defined by user
    bead = bead_image(y(1):y(2),x(1):x(2));
    figure(2), imshow(bead)
    % User input to accept or redo bead selection
    ok = input('Enter 2 to accept bead image, or 3 to redo:\n');
    % While loop to keep repeating bead selection until accepted
    while ok ~= 2 | isempty(ok) == 1
        close(2)
        clc
        fprintf('Select UPPER LEFT and LOWER RIGHT corners of bead\n');
        [x,y] = getpts;
        x = round(x);
        y = round(y);
        bead = bead_image(y(1):y(2),x(1):x(2));
        figure(2), imshow(bead)
        ok = input('Enter 2 to accept bead image, or 3 to redo:\n');
    end
    clc
    close 1 2
    clear bead_image

Save Template
```

first_bead = input...
    ('Is this the first bead template of this set? [y]-yes or [n]-no?\n', 's');
    if first_bead ~= 'y'
        clc
        nth_bead = input('Which bead is it (2, 3, etc)\n', 's');
        clc
        bead_title = ['bead' nth_bead '_' image_title '.tif'];
else
clc
bead_title = ['bead_' image_title '.tif'];
end
% Define path and save bead image
bead_path = [destination_folder_path '\ bead_title];
imwrite(bead, bead_path);

Published with MATLAB® 7.10
CREATE_RAW_MATRIX performs a normalized 2D cross-correlation on a set of images using a template. The results are filtered to keep matches greater than a defined threshold value (e.g. 0.8 - where 1.0 is a perfect match). The filtered matches are then sent to REMOVE_XCORR2_DUPLICATES to remove any immediate neighboring matches. Each match is then plotted onto each original image. The images are saved, and the results are stored in RAWCENTROIDMATRIX where each column triplet includes the X and Y coordinates, and correlation value (greater than threshold value) for each match in one frame.

```matlab
function rawCentroidMatrix = create_raw_matrix(bead, renamed_image_file, renamed_folder_path, image_title, ext, correlated_folder_path, outlined_folder_path)
    format shortG;
    % Loop to perform 2D cross-correlation on each image and get bead centroids
    for k = 1:numel(renamed_image_file)
        % Define path for reading renamed images
        renamed_image_path = fullfile(renamed_folder_path, renamed_image_file(k).name);
        renamed_image = imread(renamed_image_path);
        image = renamed_image;
        % Define template image for cross-correlation
        % bead = imread('bead.tif');
        % Perform 2D cross-correlation
        correlated_image = normxcorr2(bead, image);
        % Define image number, name, and path to write to file
        k_string = sprintf('%03.0f',k);
        correlated_image_name = ['correlated_' image_title '_' k_string '.' ext];
        correlated_image_path = fullfile(correlated_folder_path, correlated_image_name);
        imwrite(correlated_image, correlated_image_path);
        % Define template image dimensions and get centroid coordinates
        [height, width] = size(bead);
        centerHeight = floor(height/2);
        centerWidth = floor(width/2);
        % Get coordinates of matches greater than 0.8 (1.0 means perfect match)
        [row, col] = find(correlated_image > 0.8);
        % Define matrix for placing filtered correlation matches
        correlation_values = zeros(length(row),1);
        for n = 1:length(row)
            correlation_values(n) = correlated_image(row(n), col(n));
        end
        REMOVE CORRELATED DUPLICATES
        Call function to remove duplicate matches (neighboring pixels)
        filtered_frame_centroids = remove_xcorr2_duplicates(row, col,...
        correlation_values);
        % Subtract Template centroid coordinates (for plotting)
        filtered_frame_centroids(:,1) = filtered_frame_centroids(:,1) - centerHeight;
        filtered_frame_centroids(:,2) = filtered_frame_centroids(:,2) - centerWidth;
        % Recall renamed image for plotting centroid outline
        imshow(image)
        hold on
        if isempty(filtered_frame_centroids) == 1
            hold on
            outlined_image = getframe(gcf); % If no matches, save plain image
        else
            for i = 1:size(filtered_frame_centroids,1)
                plot(filtered_frame_centroids(i,2), filtered_frame_centroids(i,1), 'or')
            end
        end
    end
```
outlined_image = getframe(gcf);
end
hold off
% Define name and path of outlined images for writing to file
outlined_image_name = ['outlined_' image_title '_' k_string '.' ext];
outlined_image_path = fullfile(outlined_folder_path,outlined_image_name);
imwrite(outlined_image.cdata,outlined_image_path);
% Loop for creating centroid matrix
for n = 1:size(filtered_frame_centroids,1)
c1 = ((k-1)*3+1);
c2 = ((k-1)*3+2);
c3 = ((k-1)*3+3);
rawCentroidMatrix(n,c1) = filtered_frame_centroids(n,1);
rawCentroidMatrix(n,c2) = filtered_frame_centroids(n,2);
rawCentroidMatrix(n,c3) = filtered_frame_centroids(n,3);
if rawCentroidMatrix(n,c1) < 0
rawCentroidMatrix(n,c1) = 0;
rawCentroidMatrix(n,c2) = 0;
rawCentroidMatrix(n,c2) = 0;
end
end
end

Published with MATLAB® 7.10
remove xcorr2_duplicates

By Nector Ritzakis Created 11 July 2011 Final Rev 18 July 2011

REMOVE_XCORR2_DUPLICATES takes the coordinate pair results of a normalized 2D crosscorrelation and filters out any matching coordinate pairs within a defined pixel tolerance. The vectors ROW and COL are the row and column coordinate indeces, respectively, of the correlation matches. CORRELATION_VALUES are the normalized correlation values (where 1.0 is a perfect match) for each coordinate pair. Matching row indeces (if any) are compared with matching column indeces (if any) to find any duplicate coordinate pairs (within a defined tolerance). If more than one coordinate pair is found, the indeces are averaged, and rounded. The maximum correlation value of the matching coordinate pairs is chosen. The final coordinate pairs, along with their respective maximum correlation values are placed in FILTERED_FRAME_CENTROIDS.

\[
\text{function filtered_frame_centroids = remove_xcorr2_duplicates(row,col,}\ldots\text{correlation_values)}
\]

\[
\text{clc}
\]

% Define dummy matrices of cross-correlation match coordinates
dummyRow = row;
dummyCol = col;
End = length(row);

\[
\text{centroidRow = zeros(End,1);
centroidCol = zeros(End,1);
maxCorrelationValue = zeros(End,1);
}\]

% Define tolerance value for finding duplicate cross-correlation matches
tol = 5;
% Define counter for placing centroid value in end matrix
centroidIndex = 0;

% Loop for finding duplicate cross-correlation matches and removing them
for i = 1:End
\[
\text{if dummyRow(i) == 0}
\]
% Skip if no matches
continue % Skip if no matches
if dummyRow(i) == 0
\text{else}
\text{ Advance counter}
\text{ centroidIndex = centroidIndex + 1;}
% Define row coordinate of cross-correlation match
rowSearchElement = dummyRow(i);
% Search dummy row matrix for matching (within tolerance)
rowSearchResult = find(dummyRow(:)<(rowSearchElement + tol) &...dummyRow(:)>(rowSearchElement - tol));
% Define column coordinate of cross-correlation match
colSearchElement = dummyCol(i);
% Search dummy column matrix for matches (within tolerance)
colSearchResult = find(dummyCol(:)<(colSearchElement + tol) &...dummyCol(:)>(colSearchElement - tol));
% Perform logic test whether row match vector is smaller
smallerLength = length(rowSearchResult) <= colLength;
% Define index matrix for matching index matches
matchIndex = zeros(End,1);
% Define counter for placing index of matching index matches
place = 0;
\text{end}
\]
% Use match vector with fewer elements
switch smallerLength
    case(0) % Column match vector has fewer elements
        for j = 1:colLength
            % Advance counter
            place = place + 1;
            % Define match column index to search row vector
            matchSearchElement = colSearchResult(j);
            matchSearchResult = find(rowSearchResult == ... matchSearchElement,1);
            % Check if any match was found
            emptyTest = isempty(matchSearchResult);
            if emptyTest == 0
                % If match found, place its index in index matrix
                matchIndex(place) = matchSearchElement;
            else
                % If no match, decrease counter
                place = place - 1;
            end
        end
    end
    % Delete zeros from index matrix
    matchIndex(matchIndex == 0) = [];
    case(1) % Row match vector has fewer elements
        for j = 1:rowLength
            % Advance counter
            place = place + 1;
            % Define match row index to search column vector
            matchSearchElement = rowSearchResult(j);
            matchSearchResult = find(colSearchResult == ... matchSearchElement,1);
            % Check if any match was found
            emptyTest = isempty(matchSearchResult);
            if emptyTest == 0
                % If match found, place its index in index matrix
                matchIndex(place) = matchSearchElement;
            else
                % If no match, decrease counter
                place = place - 1;
            end
        end
    end
    % Delete zeros from index matrix
    matchIndex(matchIndex == 0) = [];
end
% Average matching row/col coordinates and round
averageRowCentroid = round(mean(dummyRow(matchIndex)));
averageColCentroid = round(mean(dummyCol(matchIndex)));
% Place averaged coordinate in row/col vectors...
centroidRow(centroidIndex) = averageRowCentroid;
centroidCol(centroidIndex) = averageColCentroid;
% and max correlation value of matches in the correlation vector
maxCorrelationValue(centroidIndex) = ...
    max(correlation_values(matchIndex));
% Delete matches so they are not searched for again
dummyRow(matchIndex) = 0;
dummyCol(matchIndex) = 0;
end
end
% Delete zeros from final vectors
centroidRow(centroidRow == 0) = [];
centroidCol(centroidCol == 0) = [];
maxCorrelationValue(maxCorrelationValue == 0) = [];
% If no centroids, mark as zero
if isempty(centroidRow) == 1
    centroidRow = 0;
end
if isempty(centroidCol) == 1
    centroidCol = 0;
end
if isempty(maxCorrelationValue) == 1
    maxCorrelationValue = 0;
end
% Combine vectors into final output matrix
filtered_frame_centroids = [centroidRow, centroidCol, maxCorrelationValue];
end

Published with MATLAB® 7.10
saveRawMatrixFile

By Nector Ritzakis Created 13 July 2011 Final Rev 18 July 2011

function [finalVerticalMatrix, matrix_file_path, verticalMatrix] = ...
saveRawMatrixFile(destination_folder_path, image_title, ...
rawCentroidMatrix)

Save Raw Matrix
% Disable excel warning
warning off MATLAB:xlswrite:AddSheet
% Define name and path of raw matrix
matrix_file_name = ['CentroidMatrix_' image_title '.xlsx'];
matrix_file_path = fullfile(destination_folder_path, matrix_file_name);
% matrix_file_path = 'C:\Users\Nector\Desktop\testsheet2.xlsx';
% Save raw matrix in sheet 1 of excel file
xlswrite(matrix_file_path, rawCentroidMatrix, 'Raw', 'A1');

Sort Raw Matrix
clc
% Call function to sort raw matrix
sortedMatrix = sort_matrix(rawCentroidMatrix);
% Save sorted matrix in sheet 2 of excel file
xlswrite(matrix_file_path, sortedMatrix, 'Sorted', 'A1');

Transpose Sorted Centroid Matrix
clc
% Call function to transpose sorted matrix
verticalMatrix = transpose_matrix(sortedMatrix);

Find max correlation value for each bead and what frame (row) it is in.
% Define index counter for storing max correlation value
place = 0;
% Define zeros vector for row index of max correlation value
row = zeros(1, size(verticalMatrix, 2)/3);
% Define zeros vector for maximum correlation value
maximum = zeros(1, length(row));
% Define zero matrix for combining above two vectors
maxLocation = zeros(2, size(verticalMatrix, 2));
for j = 3:3:size(verticalMatrix, 2)
    % Advance counter
    place = place + 1;
    % Find max value and its respective index
    [maxim, r] = max(verticalMatrix(:, j));
    % Store row index and max value
    row(place) = r;
    maximum(place) = maxim;
end
% Combine row index and max value vectors
result = [row, maximum];
% Space results to every third column
maxLocation(:, 3:3:end) = result;
maxLocationMatrix = num2cell(maxLocation);
% Define zeros vector for calculated max
maxCalculated = zeros(1, size(verticalMatrix, 2));
maxCalculatedMatrix = num2cell(maxCalculated);
% Get size of vertical matrix
[frameRow, frameCol] = size(verticalMatrix);
% Define number of beads (column triplets) in vertical matrix
beadMax = frameCol/3;
% Define numeric range for bead heading numbers
beadNumbers = 1:1:beadMax;
% Convert to cell array
beadNumbers = num2cell(beadNumbers);
% Define heading cells
beadHeader = {'Bead'};
frameHeader = {'Frame'};
maxRowHeader = {'MaxRow'};
maxValueHeader = {'MaxValue'};
maxCalcHeader = {'CalcMax'};
rowHeader = {'Row'};
colHeader = {'Col'};
corrValueHeader = {'CorrValue'};
% Define numeric range for frame heading numbers
frame = (1:1:frameRow).';
% Combine frame number with vertical matrix and convert to cell array
frameMatrix = [frame verticalMatrix];
frameCol = frameCol + 1; % To account for frame column added
frameRow = frameRow + 5; % To account for header rows added
% Assemble final vertical matrix
finalVerticalMatrix(1,1) = beadHeader;
finalVerticalMatrix(2,1) = maxRowHeader; %
finalVerticalMatrix(3,1) = maxValueHeader; %
finalVerticalMatrix(4,1) = maxCalcHeader; %
finalVerticalMatrix(5,1) = frameHeader; %
finalVerticalMatrix(1:3:3:frameCol) = beadNumbers;
finalVerticalMatrix(2:3:2:frameCol) = maxLocationMatrix; %
finalVerticalMatrix(4,2:frameCol) = maxCalculatedMatrix; %
finalVerticalMatrix(5,2:3:frameCol) = rowHeader; %
finalVerticalMatrix(5,3:3:frameCol) = colHeader; %
finalVerticalMatrix(5,4:3:frameCol) = corrValueHeader; %
finalVerticalMatrix(6:frameRow,:) = frameMatrix;
% Save vertical matrix in sheet 3 of excel file
xlswrite(matrix_file_path,finalVerticalMatrix,'Vertical','A1');

Delete Empty Sheets in Excel File
% Define excel filename
fileName = matrix_file_path;
% Delete empty excel sheets
DeleteEmptyExcelSheets(fileName);
end

Published with MATLAB® 7.10
sort_matrix

By Nector Ritzakis
Created 5-July-2011

SORTEDCENTROIDMATRIX = SORT_CENTROIDS(RAWCENTROIDMATRIX) takes
RAWCENTROIDMATRIX,
a matrix of triplet columns of X and Y centroid coordinates as well as area of
beads calculated using RAW_TO_BW and BW_FILTER functions. Each triplet of columns represents
all the beads found in one frame. SORT_CENTROIDS creates a new matrix, SORTEDCENTROIDMATRIX,
where each row includes the centroid and area data for one bead, for each frame it was found
in.

function sortedMatrix = sort_matrix(rawCentroidMatrix)
% clearvars -except rawCentroidMatrix
% clc

Start Here

% Get size of centroid matrix
[rowEnd,colEnd] = size(rawCentroidMatrix);
% Define tolerance for search
xTol = 10;
yTol = 10;
% Define sorted matrix
sortedMatrix = zeros(rowEnd,colEnd);
% Define dummy matrix
dummyCentroidMatrix = rawCentroidMatrix;

sortRow = 0;

for col = 1:3:colEnd
    for row = 1:rowEnd
        sortRow = sortRow + 1;

        % Ensures empty cells aren't used in the search
        if dummyCentroidMatrix(row,col) == 0
            sortRow = sortRow - 1;
            continue
        else

            % Define X centroid to search for in other frames
            xSearchElement = rawCentroidMatrix(row,col);
            % Search other X centroid columns for matching values, and get
            % their indices
            [xSearchResultRow,xSearchResultCol] = find(...
                (dummyCentroidMatrix(:,(col+3):3:colEnd)<...
                (xSearchElement + xTol) & dummyCentroidMatrix...
               (:,(col+3):3:colEnd)>(xSearchElement-xTol));

            % Combine row/column indices of matching X centroids
            xSearchResultRowCol = [xSearchResultRow,xSearchResultCol];
            % Calculate how many matching X centroids were found
            xSearchResultRowColSize = size(xSearchResultRowCol,1);
            % If there are no X centroid matches...
            if isempty(xSearchResultRow) == 1
                sortedMatrix(sortRow,(col:(col+2))) = rawCentroidMatrix...
                    (row,(col:(col+2)));
                dummyCentroidMatrix(row,(col:(col+2))) = 0;
                continue
            end

            % Define associated Y centroid to search for in other frames
            ySearchElement = rawCentroidMatrix(row,(col+1));
            % Search other Y centroid columns for matching values, and get
            % their indices
            [ySearchResultRow,ySearchResultCol] = find(...
                (dummyCentroidMatrix(:,(col+4):3:colEnd)<...
                (ySearchElement + yTol) & dummyCentroidMatrix...

            % Calculate how many matching Y centroids were found
            ySearchResultRowColSize = size(ySearchResultRowCol,1);
            % If there are no Y centroid matches...
            if isempty(ySearchResultRow) == 1
                sortedMatrix(sortRow,(col:(col+2))) = rawCentroidMatrix...
                    (row,(col:(col+2)));
                dummyCentroidMatrix(row,(col:(col+2))) = 0;
                continue
            end

            % Define centroid to search for in other frames
            zSearchElement = rawCentroidMatrix(row,(col+2));
            % Search other Z centroid columns for matching values, and get
            % their indices
            [zSearchResultRow,zSearchResultCol] = find(...
                (dummyCentroidMatrix(:,(col+5):3:colEnd)<...
                (zSearchElement + zTol) & dummyCentroidMatrix...
               (:,(col+5):3:colEnd)>(zSearchElement-zTol));

            % Calculate how many matching Z centroids were found
            zSearchResultRowColSize = size(zSearchResultRowCol,1);
            % If there are no Z centroid matches...
            if isempty(zSearchResultRow) == 1
                sortedMatrix(sortRow,(col:(col+2))) = rawCentroidMatrix...
                    (row,(col:(col+2)));
                dummyCentroidMatrix(row,(col:(col+2))) = 0;
                continue
            end

            % Define location of centroid to search for in other frames
            locSearchElement = rawCentroidMatrix(row,col);
            % Search other location columns for matching values, and get
            % their indices
            [locSearchResultRow,locSearchResultCol] = find(...
                (dummyCentroidMatrix(:,(col+6):3:colEnd)<...
                (locSearchElement + locTol) & dummyCentroidMatrix...

            % Calculate how many matching location centroids were found
            locSearchResultRowColSize = size(locSearchResultRowCol,1);
            % If there are no location centroid matches...
            if isempty(locSearchResultRow) == 1
                sortedMatrix(sortRow,(col:(col+2))) = rawCentroidMatrix...
                    (row,(col:(col+2)));
                dummyCentroidMatrix(row,(col:(col+2))) = 0;
                continue
            end

            % Define location of centroid to search for in other frames
            locSearchElement = rawCentroidMatrix(row,col);
            % Search other location columns for matching values, and get
            % their indices
            [locSearchResultRow,locSearchResultCol] = find(...
                (dummyCentroidMatrix(:,(col+7):3:colEnd)<...
                (locSearchElement + locTol) & dummyCentroidMatrix...

            % Calculate how many matching location centroids were found
            locSearchResultRowColSize = size(locSearchResultRowCol,1);
            % If there are no location centroid matches...
            if isempty(locSearchResultRow) == 1
                sortedMatrix(sortRow,(col:(col+2))) = rawCentroidMatrix...
                    (row,(col:(col+2)));
                dummyCentroidMatrix(row,(col:(col+2))) = 0;
                continue
            end

            % Define location of centroid to search for in other frames
            locSearchElement = rawCentroidMatrix(row,col);
            % Search other location columns for matching values, and get
            % their indices
            [locSearchResultRow,locSearchResultCol] = find(...
                (dummyCentroidMatrix(:,(col+8):3:colEnd)<...
                (locSearchElement + locTol) & dummyCentroidMatrix...
               (:,(col+8):3:colEnd)>(locSearchElement-locTol));

            % Calculate how many matching location centroids were found
            locSearchResultRowColSize = size(locSearchResultRowCol,1);
            % If there are no location centroid matches...
            if isempty(locSearchResultRow) == 1
                sortedMatrix(sortRow,(col:(col+2))) = rawCentroidMatrix...
                    (row,(col:(col+2)));
                dummyCentroidMatrix(row,(col:(col+2))) = 0;
                continue
            end

            % Define location of centroid to search for in other frames
            locSearchElement = rawCentroidMatrix(row,col);
            % Search other location columns for matching values, and get
            % their indices
            [locSearchResultRow,locSearchResultCol] = find(...
                (dummyCentroidMatrix(:,(col+9):3:colEnd)<...
                (locSearchElement + locTol) & dummyCentroidMatrix...

            % Calculate how many matching location centroids were found
            locSearchResultRowColSize = size(locSearchResultRowCol,1);
            % If there are no location centroid matches...
            if isempty(locSearchResultRow) == 1
                sortedMatrix(sortRow,(col:(col+2))) = rawCentroidMatrix...
                    (row,(col:(col+2)));
                dummyCentroidMatrix(row,(col:(col+2))) = 0;
                continue
            end

            % Define location of centroid to search for in other frames
            locSearchElement = rawCentroidMatrix(row,col);
            % Search other location columns for matching values, and get
            % their indices
            [locSearchResultRow,locSearchResultCol] = find(...
                (dummyCentroidMatrix(:,(col+10):3:colEnd)<...
                (locSearchElement + locTol) & dummyCentroidMatrix...

            % Calculate how many matching location centroids were found
            locSearchResultRowColSize = size(locSearchResultRowCol,1);
            % If there are no location centroid matches...
            if isempty(locSearchResultRow) == 1
                sortedMatrix(sortRow,(col:(col+2))) = rawCentroidMatrix...
                    (row,(col:(col+2)));
                dummyCentroidMatrix(row,(col:(col+2))) = 0;
                continue
            end

            % Define location of centroid to search for in other frames
            locSearchElement = rawCentroidMatrix(row,col);
            % Search other location columns for matching values, and get
            % their indices
            [locSearchResultRow,locSearchResultCol] = find(...
                (dummyCentroidMatrix(:,(col+11):3:colEnd)<...
                (locSearchElement + locTol) & dummyCentroidMatrix...

            % Calculate how many matching location centroids were found
            locSearchResultRowColSize = size(locSearchResultRowCol,1);
            % If there are no location centroid matches...
            if isempty(locSearchResultRow) == 1
                sortedMatrix(sortRow,(col:(col+2))) = rawCentroidMatrix...
                    (row,(col:(col+2)));
                dummyCentroidMatrix(row,(col:(col+2))) = 0;
                continue
            end

            % Define location of centroid to search for in other frames
            locSearchElement = rawCentroidMatrix(row,col);
            % Search other location columns for matching values, and get
            % their indices
            [locSearchResultRow,locSearchResultCol] = find(...
                (dummyCentroidMatrix(:,(col+12):3:colEnd)<...
                (locSearchElement + locTol) & dummyCentroidMatrix...
% Combine row/colum indices of matching Y centroids
ySearchResultRowCol = [ySearchResultRow,ySearchResultCol];
% Calculate how many matching Y centroids were found
ySearchResultRowColSize = size(ySearchResultRowCol,1);
% Find which set of centroids (X or Y) is smaller
smallerSearchResultSize = xSearchResultRowColSize <= ySearchResultRowColSize;
% Use smaller set to find matches
switch smallerSearchResultSize
case(0) % Y is smaller
  % Add original X-Y centroid and area in same indeces in % sorted matrix
  sortedMatrix(sortRow,(col:(col+2))) = ....
  rawCentroidMatrix(row,(col:(col+2))) = ....
  % Delete original centroid values from dummy matrix
  dummyCentroidMatrix(row,(col:(col+2))) = 0;
  for i = 1:ySearchResultRowColSize
    % Define row index to search X centroid set
    rowMatchSearch = ySearchResultRowCol(i,1);
    % Define column index to search X centroid set
    colMatchSearch = ySearchResultRowCol(i,2);
    % Search X centroid set for row/col index matches
    matchResult = find(xSearchResultRowCol(:,1) == rowMatchSearch & xSearchResultRowCol(:,2) == colMatchSearch);
    % Define matching X centroid row in terms of % original centroid matrix
    matchRow = xSearchResultRowCol(matchResult,1);
    % Define matching X centroid column in terms of % original centroid matrix columns
    matchCol = 3*(xSearchResultRowCol(matchResult,2)) + col;
    % Add matching X-Y centroid(s) and area(s) in same row % as original, and same column(s) on sorted matrix
    sortedMatrix(sortRow,(matchCol:(matchCol+2))) = ....
    rawCentroidMatrix(matchRow,(matchCol:(matchCol+2))) = ....
    dummyCentroidMatrix(matchRow,(matchCol+2)) = 0;
  end
case(1) % X is smaller
  % Add original X-Y centroid and area in same indeces in % sorted matrix
  sortedMatrix(sortRow,(col:(col+2))) = ....
  rawCentroidMatrix(row,(col:(col+2))) = ....
  % Delete original centroid values from dummy matrix
  dummyCentroidMatrix(row,(col:(col+2))) = 0;
  for i = 1:xSearchResultRowColSize
    % Define row index to search Y centroid set
    rowMatchSearch = xSearchResultRowCol(i,1);
    % Define column index to search Y centroid set
    colMatchSearch = xSearchResultRowCol(i,2);
    % Search Y centroid set for row/col index matches
    matchResult = find(ySearchResultRowCol(:,1) == rowMatchSearch & ySearchResultRowCol(:,2) == colMatchSearch);
    % Define matching Y centroid row in terms of % original centroid matrix
    matchRow = ySearchResultRowCol(matchResult,1);
    % Define matching Y centroid column in terms of % original centroid matrix columns% adjusts to % actual dummyCentroidMatrix column
    matchCol = 3*(ySearchResultRowCol(matchResult,2))...
+ col;
% Add matching X-Y centroid(s) and area(s) in same
% row as original, and same column(s) on sorted
% matrix
sortedMatrix(sortRow, (matchCol:(matchCol+2))) = ... 
rawCentroidMatrix(matchRow, ... 
(matchCol:(matchCol+2))); 
% Delete matching centroid values from dummy matrix
dummyCentroidMatrix...
(matchRow, (matchCol:(matchCol+2))) = 0;
end
end
end
end
end
Published with MATLAB® 7.10
transpose_matrix

By Nector Ritzakis Created 11 July 2011 Final Rev 18 July 2011

function verticalSortedCentroidMatrix = transpose_matrix(sortedMatrix)
clc
% Get size of sorted matrix
[rowEnd, colEnd] = size(sortedMatrix);
% Increase number of columns by a factor of three
newColEnd = rowEnd*3;
% Define matrix with all X centroid coordinates
xCentroids = sortedMatrix(:,1:3:colEnd);
% Define matrix with all Y centroid coordinates
yCentroids = sortedMatrix(:,2:3:colEnd);
% Define matrix with all correlation values
correlationValue = sortedMatrix(:,3:3:colEnd);
% Transpose each matrix
xCentroids = xCentroids.';
yCentroids = yCentroids.';
correlationValue = correlationValue.';
% Recombine into vertical sorted centroid matrix
verticalSortedCentroidMatrix(:,1:3:newColEnd) = xCentroids( :, :);
verticalSortedCentroidMatrix(:,2:3:newColEnd) = yCentroids( :, :);
verticalSortedCentroidMatrix(:,3:3:newColEnd) = correlationValue( :, :);
end

Published with MATLAB® 7.10
DeleteEmptyExcelSheets

% DeleteEmptyExcelSheets: deletes all empty sheets in an xls-file
%==========================================================================
% Version : 1.0
% Author : hnagel
% Date : 27/04/2007
% Tested : 02/05/2007 (DR)
%==========================================================================
% This function looped through all sheets and deletes those sheets that are
% empty. Can be used to clean a newly created xls-file after all results
% have been saved in it.
% References: Torsten Jacobsen, "delete standard excel sheet"
%---------------------------------------------------------------------
%
% Input:
% fileName: name of xls file
%---------------------------------------------------------------------
%
% Output:
% none
%---------------------------------------------------------------------
%
% See also XLSWRITE
%---------------------------------------------------------------------
%
% Changes
%---------------------------------------------------------------------
%
% Name :
% Date :
% Description:
% Indicated :

function DeleteEmptyExcelSheets(fileName)

% Check whether the file exists
% if ~exist(fileName,'file')
  error([fileName ' does not exist !']);
% else
% Check whether it is an Excel file
% typ = xlsfinfo(fileName);
% if ~strcmp(typ,'Microsoft Excel Spreadsheet')
  error([fileName ' not an Excel sheet !']);
% end
% end
%
% If fileName does not contain a \\
% the name of the current path is added
% to fileName. The reason for this is that the full path is required for
% the command "excelObj.workbooks.Open(fileName)" to work properly.
% if isempty(strfind(fileName,'\'))
  fileName = [cd '\' fileName];
% end

excelObj = actxserver('Excel.Application');
excelWorkbook = excelObj.workbooks.Open(fileName);
worksheets = excelObj.sheets;
sheetIdx = 1;
sheetIdx2 = 1;
numSheets = worksheets.Count;
% Prevent beeps from sounding if we try to delete a non-empty worksheet.
excelObj.EnableSound = false;
% Loop over all sheets
while sheetIdx2 <= numSheets
% Saves the current number of sheets in the workbook
temp = worksheets.count;
% Check whether the current worksheet is the last one. As there always
% need to be at least one worksheet in an xls-file the last sheet must
% not be deleted.
if or(sheetIdx>1,numSheets-sheetIdx2>0)
% worksheets.Item(sheetIdx).UsedRange.Count is the number of used cells.
% This will be 1 for an empty sheet. It may also be one for certain other
% cases but in those cases, it will beep and not actually delete the sheet.
if worksheets.Item(sheetIdx).UsedRange.Count == 1
worksheets.Item(sheetIdx).Delete;
end
end
% Check whether the number of sheets has changed. If this is not the
% case the counter "sheetIdx" is increased by one.
if temp == worksheets.count;
sheetIdx = sheetIdx + 1;
end
sheetIdx2 = sheetIdx2 + 1; % prevent endless loop...
end
excelObj.EnableSound = true;
excelWorkbook.Save;
excelWorkbook.Close(false);
excelObj.Quit;
delete(excelObj);
return;
end

Published with MATLAB® 7.10
calculate_center_frame

By Nector Ritzakis Created 14 July 2011 Final Rev 20 July 2011

```matlab
function resultMatrix = calculate_center_frame(verticalMatrix,...
    matrix_file_path,plot_folder_path,b_or_a,image_title)
    referenceMatrix = verticalMatrix(1:end-2,:);
    [rowEnd,rowCol] = size(referenceMatrix);
    resultMatrix = zeros(1,rowCol);
    plot_number = 1;
    % Plot correlation values for each bead, fit line, get local maximum to
    % define as bead center
    for i = 3:3:rowCol
        dummyMatrix = zeros(rowEnd,2);
        dummyMatrix(:,1) = 1:rowEnd;
        dummyMatrix(:,2) = referenceMatrix(:,i);
        [r,~] = find(dummyMatrix(:,2) == 0);
        dummyMatrix(r,:) = [];
        avg = mean(dummyMatrix(:,2));
        stdev = std(dummyMatrix(:,2));
        outliers = find(dummyMatrix(:,2) > (2*stdev + avg) | dummyMatrix(:,2)...
            < (2*stdev - avg));
        dummyMatrix(outliers,:) = [];
        x = dummyMatrix(:,1);
        y = dummyMatrix(:,2);
        p = polyfit(x,y,2);
        y2 = polyval(p,x);
        der = polyder(p);
        root = roots(der);
        if isempty(root) == 1
            root = 0;
        end
        resultMatrix(i) = root;
        plot_number_string = sprintf('%03.0f',plot_number);
        plot(x,y,'o',x,y2,'-');
        xlabel('Frame')
        ylabel('Correlation Value (Max = 1.0)')
        if b_or_a == 1
            title_string = ['B' image_title(2:end) ' Bead' plot_number_string...
                ' Polynomial Fit'];
            title(title_string,'Interpreter','none')
        else
            title_string = ['A' image_title(2:end) ' Bead' plot_number_string...
                ' Polynomial Fit'];
            title(title_string,'Interpreter','none')
        end
        plot_image_path = [plot_folder_path '\Bead' plot_number_string '_...
            image_title '.jpg'];
        saveas(gcf,plot_image_path);
        plot_number = plot_number + 1;
    end
    resultMatrix = num2cell(resultMatrix);
    % Save vertical matrix in sheet 1 of excel file
    xlswrite(matrix_file_path,resultMatrix,'Vertical','B4');
end
```

get_final_beads

By Nector Ritzakis Created 19 July 2011 Final Rev 20 July 2011

```matlab
function [beadMatrix,beadMatrix_60frames] = get_final_beads(resultMatrix,...
    finalVerticalMatrix,matrix_file_path,top)
    Adjust Vertical Sorted Matrix Such That Top Row Represents Top of Sample
```
close all
clc
verticalMatrix = finalVerticalMatrix(6:end,:);
verticalMatrix = cell2mat(verticalMatrix);
verticalMatrix(:,1) = [];
originalTop = size(verticalMatrix,1);
% Convert original top frame to new numbering
newTop = (originalTop+1) - top;
Define new vertical matrix without frames above "top"
adjustedVerticalMatrix = verticalMatrix;
adjustedVerticalMatrix(1:(newTop-1),:) = [];
Adjust max location row to reflect "top" change
maxLocation = finalVerticalMatrix(2,2:end);
maxLocation = cell2mat(maxLocation);
maxLocation(1,3:3:end) = maxLocation(1,3:3:end) + 1 - newTop;
maxLocation = num2cell(maxLocation);
Adjust result matrix to reflect "top" change
resultMatrix = cell2mat(resultMatrix);
resultMatrix(3:3:end) = resultMatrix(3:3:end) + 1 - newTop;
resultMatrix = num2cell(resultMatrix);
Define new frame number range
frame = (1:1:top).';
[frameRow,frameCol] = size(adjustedVerticalMatrix);
frameCol = frameCol + 1; % To account for frame column added
frameRow = frameRow + 5; % To account for header rows
Assemble final adjusted vertical matrix
frameMatrix = [frame adjustedVerticalMatrix];
frameMatrix = num2cell(frameMatrix);
finalAdjustedVerticalMatrix(6:frameRow,1:frameCol) = frameMatrix;
finalAdjustedVerticalMatrix(1,1:frameCol) = finalVerticalMatrix(1,:);
finalAdjustedVerticalMatrix(2,1) = finalVerticalMatrix(2,1);
finalAdjustedVerticalMatrix(2,2:frameCol) = maxLocation;
finalAdjustedVerticalMatrix(3:5,1:frameCol) = finalVerticalMatrix(3:5,:);
finalAdjustedVerticalMatrix(4,2:frameCol) = resultMatrix;
Save top adjusted matrix in excel file
xlswrite(matrix_file_path,finalAdjustedVerticalMatrix,'TopAdjusted','A1');
Get Final Beads From Top Adjusted Matrix Along the Whole Scan
clc
beadMatrix = resultMatrix;
beadMatrix = cell2mat(beadMatrix);
beadMatrix = beadMatrix(3:3:end);
maxBeadNumber = length(beadMatrix);
beadNumber = 1:maxBeadNumber;
beadMatrix = [beadNumber; beadMatrix];
[~,c] = find(beadMatrix(2,:) <= 0);
beadMatrix(:,c) = [];
for i = 1:length(c)
    adjustedVerticalMatrix(:,((c(i)-1)*3+1:(c(i)-1)*3+3)) = [];
c = c-1;
end
numberOfFrames = size(adjustedVerticalMatrix,1);
[~,c] = find(beadMatrix(2,:) > numberOfFrames);
beadMatrix(:,c) = [];
for i = 1:length(c)
    adjustedVerticalMatrix(:,((c(i)-1)*3+1:(c(i)-1)*3+3)) = [];
    c = c-1;
end
beadMatrix = round(beadMatrix);
[~,c] = find(beadMatrix(2,:) < 0);
beadMatrix(:,c) = [];
for i = 1:length(c)
    adjustedVerticalMatrix(:,((c(i)-1)*3+1:(c(i)-1)*3+3)) = [];
    c = c-1;
end
% Change any frames labeled 0 to 1
[-,c] = find(beadMatrix(2,:) == 0);
beadMatrix(2,c) = 1;
numberOfBeads = size(beadMatrix,2);
beadCentroidMatrix = zeros(2,numberOfBeads);
% (1,:) = centroid row coordinate, (2,:) = centroid col coordinate
for i = 1:numberOfBeads
% Define calculated center frame to get coordinates from
frameCoordinate = beadMatrix(2,i);
% If frame is "0" (i.e. top frame) move down one frame for coordinates
if frameCoordinate == 0
frameCoordinate = 1;
end
if frameCoordinate > size(adjustedVerticalMatrix,1)
continue
end
beadRowLocation = (i-1)*3+1;
beadColLocation = (i-1)*3+2;
beadCentroidMatrix(1,i) = adjustedVerticalMatrix(frameCoordinate,...
        beadRowLocation);
beadCentroidMatrix(2,i) = adjustedVerticalMatrix(frameCoordinate,...
        beadColLocation);
end
beadMatrix = [beadMatrix;beadCentroidMatrix];
beadMatrix([2 3 4],:) = beadMatrix([3 4 2],:);
[-,c] = find(beadMatrix(2,:) == 0 & beadMatrix(3,:) == 0);
beadMatrix(:,c) = [];
% Calculate original frame
col = size(beadMatrix,2);
originalFrame = zeros(1,col);
for i = 1:col
originalFrame(i) = top - beadMatrix(4,i);
end
beadHeader = {'Bead'};
rowHeader = {'Row'};
colHeader = {'Col'};
frameHeader = {'Frame'};
originalFrameHeader = {'OrigFrame'};
fullBeadMatrix = [beadMatrix;originalFrame];
fullBeadMatrix = num2cell(fullBeadMatrix);
fullFinalBeadMatrix(1:5,1) = [beadHeader;rowHeader;colHeader;frameHeader;...
originalFrameHeader];
fullFinalBeadMatrix(:,2:size(fullBeadMatrix,2)+1) = fullBeadMatrix;
xlswrite(matrix_file_path,fullFinalBeadMatrix,'FullScanFinalBeads','A1');

Get Final Beads for Only Top 60 Frames of Sample (60 Frames = 30um)
beadMatrix_60frames = [beadMatrix;originalFrame];
[-,c] = find(beadMatrix(4,:) > 60);
beadMatrix_60frames(:,c) = [];
beadMatrix_60frames = num2cell(beadMatrix_60frames);
finalBeadMatrix(1:5,1) = [beadHeader;rowHeader;colHeader;frameHeader;...
originalFrameHeader];
finalBeadMatrix(:,2:size(beadMatrix_60frames,2)+1) = beadMatrix_60frames;
xlswrite(matrix_file_path,finalBeadMatrix,'60FramesFinalBeads','A1');
end
Published with MATLAB® 7.10

close all
clear all
clc
% Disable excel warning
warning off MATLAB:xlswrite:AddSheet
% Set folder
fprintf('Select folder BOTH before AND after folders are located\n')
path = uigetdir;
clc

Get Before Data
beforeFolderSearchPath = [path '\*before*'];
beforeFolder = dir(beforeFolderSearchPath);
umFolders = numel(beforeFolder);
if numFolders > 1
fprintf('Have found %d folders with BEFORE title\n\n',numFolders)
for i = 1:numFolders
fprintf('Is the folder you want to use:\n\n%s\n\n','beforeFolder(i).name)
y_or_n = input('Enter [y]-yes or [n]-no\n','s');
while y_or_n ~= 'y' && y_or_n ~= 'n'
clc
fprintf('Not a valid entry\n\n')
fprintf('Is the folder you want to use:\n\n%s\n\n','beforeFolder(i).name)
y_or_n = input('Enter [y]-yes or [n]-no\n','s');
end
clc
if y_or_n == 'y'
beforeFolder = beforeFolder(i).name;
break
end
if y_or_n == 'n' && i == numFolders
error('Did not select any folder')
end
else
beforeFolder = beforeFolder.name;
end
beforeFolderPath = fullfile(path,beforeFolder);
matSearchPath = [beforeFolderPath '\*before*.mat'];
matFile = dir(matSearchPath);
matFilePath = [beforeFolderPath ' ' matFile.name];
load(matFilePath);
beforeTitle = image_title;
beforeTitle = beforeTitle(8:end);
clearvars -except beadMatrix_60frames beforeTitle path
beforeBeadMatrix = beadMatrix_60frames;
clear beadMatrix_60frames

Get After Data
afterFolderSearchPath = [path '\*after*'];
afterFolder = dir(afterFolderSearchPath);
umFolders = numel(afterFolder);
if numFolders > 1
fprintf('Have found %d folders with AFTER title\n\n',numFolders)
for i = 1:numFolders
fprintf('Is the folder you want to use:\n\n%s\n\n','afterFolder(i).name)
y_or_n = input('Enter [y]-yes or [n]-no\n','s');
while y_or_n ~= 'y' && y_or_n ~= 'n'
clc
fprintf('Not a valid entry\n\n')
fprintf('Is the folder you want to use:\n\n%s\n\n','afterFolder(i).name)
y_or_n = input('Enter [y]-yes or [n]-no\n','s');
end
clc
if y_or_n == 'y'
afterFolder = afterFolder(i).name;
break
end
if y_or_n == 'n' && i == numFolders
error('Did not select any folder')
end
else
afterFolder = afterFolder.name;
end
afterFolderPath = fullfile(path,afterFolder);
matSearchPath = [afterFolderPath '\*after*.mat'];
matFile = dir(matSearchPath);
matFilePath = [afterFolderPath ' ' matFile.name];
load(matFilePath);
afterTitle = image_title;
afterTitle = afterTitle(8:end);
clearvars -except beadMatrix_60frames afterTitle path
afterBeadMatrix = beadMatrix_60frames;
clear beadMatrix_60frames
while y_or_n ~= 'y' && y_or_n ~= 'n'
    clc
    fprintf('Not a valid entry\n\n')
    fprintf('Is the folder you want to use:\n\n%s\n\n','afterFolder(i).name)
    y_or_n = input('[y]-yes or [n]-no\n','s');
end
    clc
    if y_or_n == 'y'
        afterFolder = afterFolder(i).name;
        break
    end
    if y_or_n == 'n' && i == numFolders
        error('Did not select any folder')
    end
end
elseif numFolders < 1
    error('No folders with AFTER title were found\n')
else
    afterFolder = afterFolder.name;
end
afterFolderPath = fullfile(path,afterFolder);
matSearchPath = [afterFolderPath '\*after*.mat'];
matFile = dir(matSearchPath);
matFilePath = [afterFolderPath '\' matFile.name];
load(matFilePath);
afterTitle = image_title;
afterTitle = afterTitle(7:end);
clearvars -except beadMatrix_60frames beforeTitle
clearvars -except afterTitle path beforeBeadMatrix
afterBeadMatrix = beadMatrix_60frames;
clear beadMatrix_60frames
clc

Check That Corresponding Sets Were Selected
if beforeTitle ~= afterTitle
    error('ERROR!! Before and after distances different.\n')
end
image_title = beforeTitle;
clear beforeTitle afterTitle

Input Before and After Bead Matches
beadMatchNumber = input('Enter number of bead matches found\n');
clc
beforeMatchedBeadNumber = zeros(1,beadMatchNumber);
afterMatchedBeadNumber = zeros(1,beadMatchNumber);
beforeMatchedBeadMatrix = zeros(5,beadMatchNumber);
afterMatchedBeadMatrix = zeros(5,beadMatchNumber);
rawMatchedBeforeMatrix = zeros(5,beadMatchNumber+1);
rawMatchedAfterMatrix = zeros(5,beadMatchNumber+1);
check = 3;
while check ~= 2
    for i = 1:beadMatchNumber
        beforeMatchedBeadNumber(i) = input...
        ('Enter one bead number of BEFORE set at a time, followed by ENTER\n');
    end
    clc
    for i = 1:beadMatchNumber
        fprintf('Enter one bead number of AFTER set at a time,\n')
        afterMatchedBeadNumber(i) = input...
        (' IN THE SAME SEQUENCE, followed by ENTER\n');
    end
    [beforeMatchedBeadNumber;afterMatchedBeadNumber]
    check = input...
end
clc

Save Workspace to Folder
save([path '\CorrBeadsWorkspace_' image_title '.mat']);

Compile Before and After Match Matrices
for i = 1:beadMatchNumber
    beadSearchElement = beforeMatchedBeadNumber(i);
    [~,c] = find(beforeBeadMatrix(1,:) == beadSearchElement);
    beforeMatchedBeadMatrix(:,i) = beforeBeadMatrix(:,c);
    beadSearchElement = afterMatchedBeadNumber(i);
    [~,c] = find(afterBeadMatrix(1,:) == beadSearchElement);
    afterMatchedBeadMatrix(:,i) = afterBeadMatrix(:,c);
end

% Define headers for excel file
beadHeader = {'Bead No'};
rowHeader = {'Row'};
co1Header = {'Col'};
frameHeader = {'Frame'};
originalFrameHeader = {'OrigFrame'};
beforeHeader = {'Before Set'};
afterHeader = {'After Set'};

% Place bead match matrix into final (raw) matrix
rawMatchedBeforeMatrix(:,2:end) = beforeMatchedBeadMatrix;
rawMatchedAfterMatrix(:,2:end) = afterMatchedBeadMatrix;

% Convert to cells
rawMatchedBeforeMatrix = num2cell(rawMatchedBeforeMatrix);
rawMatchedAfterMatrix = num2cell(rawMatchedAfterMatrix);

% Insert row headers
rawMatchedBeforeMatrix(1:5,1) = [beadHeader;rowHeader;co1Header;frameHeader;originalFrameHeader];
rawMatchedAfterMatrix(1:5,1) = [beadHeader;rowHeader;co1Header;frameHeader;originalFrameHeader];

% Define path to save in folder
folder_file_path = [path '\FinalMatchedBeads' image_title '.xlsx'];

% Save raw matrix as excel file
xlswrite(folder_file_path,beforeHeader,'rawMatches','A1');
xlswrite(folder_file_path,rawMatchedBeforeMatrix,'rawMatches','A2');
xlswrite(folder_file_path,afterHeader,'rawMatches','A8');
xlswrite(folder_file_path,rawMatchedAfterMatrix,'rawMatches','A9');

Define Path for Saving All Runs of Each Set

% Define title distance
distTitle = image_title(1:end-5);
set_path = ['C:\Users\Nector\Desktop\' distTitle '.xlsx'];

Define Run, Headers and Ranges for Saving

Raw Matrix
% Get run to define which excel row to save in
run = image_title(end);
run = str2num(run);
runTitle = sprintf('RUN%d',run);
runHeader = {runTitle};
% Define which excel row to save raw matrix
rowCell = (run-1)*14+1; % For run title and before matrix
beforeCell = rowCell+1; % For before matrix
afterCell = rowCell+7; % For after title
afterCell = rowCell+8; % For after matrix
% Define cell location to save run title and raw matrix
runRange = sprintf('%d',rowCell);
beforeTitleRange = sprintf('%d',rowCell);
beforeMatrixRange = sprintf('%d',beforeCell);
afterTitleRange = sprintf('%d',afterCell);
afterMatrixRange = sprintf('%d',afterCell);

Save Raw Matrix to Set File
% Save raw matrix files in set file
Sort Beads

Get bead coordinates, frames, and original frames to sort

toSortMatrix = [...
    [beforeMatchedBeadMatrix(1:5,:);afterMatchedBeadMatrix(1:5,:)]
];
% Sort according to column first, then row
sortedMatrix = (sortrows(toSortMatrix,'4')).';
sortedCellMatrix = num2cell(sortedMatrix);
% Define cell matrix for sorted beads
finalSortedMatchedBeforeMatrix = cell(5,beadMatchNumber+1);
finalSortedMatchedAfterMatrix = cell(5,beadMatchNumber+1);
% Assemble matrix
finalSortedMatchedBeforeMatrix(:,2:end) = sortedCellMatrix(1:5,:);
finalSortedMatchedAfterMatrix(:,2:end) = sortedCellMatrix(6:10,:);
% Insert heading
finalSortedMatchedBeforeMatrix(1:5,1) = [...
    {'Sorted Bead'};rowHeader;colHeader;frameHeader;originalFrameHeader;
];
finalSortedMatchedAfterMatrix(1:5,1) = [...
    {'Sorted Bead'};rowHeader;colHeader;frameHeader;originalFrameHeader;
];
% Save sorted matrix in excel file in destination folder
xlswrite(folder_file_path,beforeHeader,'sortedMatches','A1');
xlswrite(folder_file_path,beforeHeader,'sortedMatches','A2');
xlswrite(folder_file_path,afterHeader,'sortedMatches','A8');
xlswrite(folder_file_path,afterHeader,'sortedMatches','A9');
% Insert sorted matrix to set file
xlswrite(set_path,runHeader,'sortMatches',runRange);
xlswrite(set_path,beforeHeader,'sortMatches',beforeTitleRange);
xlswrite(set_path,finalSortedMatchedBeforeMatrix,'sortMatches',...beforeMatrixRange);
xlswrite(set_path,afterHeader,'sortMatches',afterTitleRange);
xlswrite(set_path,finalSortedMatchedAfterMatrix,'sortMatches',...afterMatrixRange);

Convert to Microns

Define matrix to convert to microns
convertedMatchedBeforeMatrix = ...
    cell2mat(finalSortedMatchedBeforeMatrix(1:5,2:end));
convertedMatchedAfterMatrix = ...
    cell2mat(finalSortedMatchedAfterMatrix(1:5,2:end));
format shortG
% Perform conversion
convertedMatchedBeforeMatrix(2:5,:) = ...
    round([convertedMatchedBeforeMatrix(2:3,:).*0.322;...
    convertedMatchedBeforeMatrix(4:5,:).*0.5].*100)./100;
convertedMatchedAfterMatrix(2:5,:) = ...
    round([convertedMatchedAfterMatrix(2:3,:).*0.322;...
    convertedMatchedAfterMatrix(4:5,:).*0.5].*100)./100;
% Convert to cell matrix
convertedMatchedBeforeMatrix = num2cell(convertedMatchedBeforeMatrix);
convertedMatchedAfterMatrix = num2cell(convertedMatchedAfterMatrix);
% Define final converted matrix size
finalConvertedMatchedBeforeMatrix = cell(5,beadMatchNumber+1);
finalConvertedMatchedAfterMatrix = cell(5,beadMatchNumber+1);
% Insert converted data into final matrix
finalConvertedMatchedBeforeMatrix(:,2:end) = convertedMatchedBeforeMatrix;
finalConvertedMatchedAfterMatrix(:,2:end) = convertedMatchedAfterMatrix;
% Insert headers
finalConvertedMatchedBeforeMatrix(1:5,1) = [{'Converted Bead'};...]
rowHeader;colHeader;frameHeader;originalFrameHeader];
finalConvertedMatchedAfterMatrix(1:5,1) = {{'Converted Bead'};...}
rowHeader;colHeader;frameHeader;originalFrameHeader];
% Save converted matrix in excel file in destination folder
xlswrite(folder_file_path,beforeHeader,'convertedMatches','A1');
xlswrite(folder_file_path,finalConvertedMatchedBeforeMatrix,...
'convertedMatchGs','A2');
xlswrite(folder_file_path,afterHeader,'convertedMatches','A8');
Get Before and After Matching Beads
6
xlswrite(folder_file_path,finalConvertedMatchedAfterMatrix,...
'convertedMatchGs','A9');
% Save converted matrix in set file
xlswrite(set_path,runHeader,'convertMatches',runRange);
xlswrite(set_path,beforeHeader,'convertMatches',beforeTitleRange);
xlswrite(set_path,finalConvertedMatchedBeforeMatrix,'convertMatches',...
beforeMatrixRange);
xlswrite(set_path,afterHeader,'convertMatches',afterTitleRange);
xlswrite(set_path,finalConvertedMatchedAfterMatrix,'convertMatches',...
afterMatrixRange);

Delete Empty Sheets in Excel Files
fileName = folder_file_path;
% Delete empty excel sheets
DeleteEmptyExcelSheets(fileName);
% Define filename
fileName = set_path;
% Delete empty excel sheets
DeleteEmptyExcelSheets(fileName);

Save Workspace to Folder
save([path '\FinalBeadWorkspace_' image_title '.mat']);

Plot
SortedMatchedBeforeMatrix = finalSortedMatchedBeforeMatrix(2:4,2:end);
SortedMatchedAfterMatrix = cell2mat(SortedMatchedBeforeMatrix);
SortedMatchedBeforeMatrix = cell2mat(SortedMatchedAfterMatrix);
ConvertedMatchedBeforeMatrix = finalConvertedMatchedBeforeMatrix(2:4,2:end);
ConvertedMatchedAfterMatrix = cell2mat(ConvertedMatchedBeforeMatrix);
beforeX = ConvertedMatchedBeforeMatrix(2,:);
beforeY = ConvertedMatchedBeforeMatrix(1,:);
beforeZ = ConvertedMatchedBeforeMatrix(3,:);
afterX = ConvertedMatchedAfterMatrix(2,:);
afterY = ConvertedMatchedAfterMatrix(1,:);
afterZ = ConvertedMatchedAfterMatrix(3,:);
plot(beforeX,beforeY,'.b')
hold on
plot(afterX,afterY,'.r')
axis([0 450 0 335])
xlabel('\mum');
ylabel('\mum');
title_string =...
{'Before and After Final Matching Beads (X-Y Projection);Nector Ritzakis'};
title(title_string,'Interpreter','none');
axis ij
legend('Before','After','Location','EastOutside')
hold off
final_match_path = [path '\Final Bead Match ' image_title '.jpeg'];
saveas(gcf,final_match_path);
close all

Save Workspace to Folder
save([path '\RegistrationResultsWorkspace_' image_title '.mat']);
convert_to_spherical

By: Nector Ritzakis Created: 27-Oct-2011

clear all
close all
clc
% Disable excel warning
warning off MATLAB:xlswrite:AddSheet

Import Cartesian Point Cloud
inputfile = 'C:\Users\Nector\Desktop\Point Cloud Analysis.xlsx';
clc
distance = input...
('Enter secondary distance WITHOUT decimal\n(or DDK/Thrust)\n','s');
check25 = strcmp(distance,'25');
check5 = strcmp(distance,'5');
check10 = strcmp(distance,'10');
check30 = strcmp(distance,'30');
checkThrust = strcmp(distance,'Thrust');
checkDDK = strcmp(distance,'DDK');
if check25 == 1
distance = '2.5mm';
elseif check5 == 1
distance = '5mm';
elseif check10 == 1
distance = '10mm';
elseif check30 == 1
distance = '30mm';
elseif checkThrust == 1;
distance = 'Thrust';
elseif checkDDK == 1;
distance = 'DDK';
else
1
derror('INVALID ENTRY')
end
clc
run = input('Enter run\n','s');
clc
fprintf('Enter UPPER LEFT cell row of range of BEFORE SET\n')
topRange = input(' (e.g. if cell is C2...enter 2\n','s');
bottomRange = input...
('Enter LOWER RIGHT cell of range of SAME BEFORE SET\n','s');
clc
beforeRange = ['D' topRange ':' 'F' bottomRange];
afterRange = ['H' topRange ':' 'J' bottomRange];
beforeCartesian = xlsread(inputfile,distance,beforeRange);
afterCartesian = xlsread(inputfile,distance,afterRange);

Get Lengths
% Calculate number of lengths, m
n = size(beforeCartesian,1);
m = n*(n-1)/2;
% Get length indeces
index1 = zeros(m,1);
index2 = zeros(m,1);
begin = 2;
k = 1;
for i = begin-1:n-1
for j = begin:n
index1(k) = i;
index2(k) = j;
k = k+1;
end
end
begin = begin+1;
end
% Define before and after length matrices
beforeLengths = zeros(m,3);
afterLengths = zeros(m,3);
% Calculate lengths
for i = 1:m
    beforeLengths(i,1) = beforeCartesian(index2(i),1) - ...
    beforeCartesian(index1(i),1); % Y
    beforeLengths(i,2) = beforeCartesian(index2(i),2) - ...
    beforeCartesian(index1(i),2); % X
    beforeLengths(i,3) = beforeCartesian(index2(i),3) - ...
    beforeCartesian(index1(i),3); % Z
    afterLengths(i,1) = afterCartesian(index2(i),1) - ...
    afterCartesian(index1(i),1); % Y
    afterLengths(i,2) = afterCartesian(index2(i),2) - ...
    afterCartesian(index1(i),2); % X
    afterLengths(i,3) = afterCartesian(index2(i),3) - ...
    afterCartesian(index1(i),3); % Z
end
Convert to Spherical
% Define before and after spherical matrices
beforeSpherical = zeros(m,3);
afterSpherical = zeros(m,3);
% Convert lengths to spherical
for i = 1:m
    [theta,phi,r] = cart2sph(beforeLengths(i,2),beforeLengths(i,1),...
    beforeLengths(i,3));
    theta = theta*180/pi;
    phi = phi*180/pi;
    beforeSpherical(i,:) = [r,theta,phi];
    [theta,phi,r] = cart2sph(afterLengths(i,2),afterLengths(i,1),...
    afterLengths(i,3));
    theta = theta*180/pi;
    phi = phi*180/pi;
    afterSpherical(i,:) = [r,theta,phi];
end
Calculate Deltas
% Define delta matrix
delta = zeros(m,3);
for i = 1:m
    deltaR = afterSpherical(i,1) - beforeSpherical(i,1);
    deltaTheta = afterSpherical(i,2) - beforeSpherical(i,2);
    % Check angle change isn’t greater than 180 degrees
    if abs(deltaTheta) > 180
        if deltaTheta < 0
            deltaTheta = 360 - abs(deltaTheta);
        else
            deltaTheta = -360 + abs(deltaTheta);
        end
    end
    deltaPhi = afterSpherical(i,3) - beforeSpherical(i,3);
    % Check angle change isn’t greater than 180 degrees
    if abs(deltaPhi) > 180
        if deltaTheta < 0
            deltaTheta = 360 - abs(deltaPhi);
        else
            deltaTheta = -360 + abs(deltaPhi);
        end
    end
    delta(i,:) = [deltaR,deltaTheta,deltaPhi];
end
Calculate Stretch and Absolutes
stretchAbsolute = zeros(m,3);
for i = 1:m
    rStretch = abs(delta(i,1)/beforeSpherical(i,1));
    absTheta = abs(delta(i,2));
    absPhi = abs(delta(i,3));
    stretchAbsolute(i,:) = [rStretch,absTheta,absPhi];
end

Create Final Matrix
% Define final matrix
finalSphericalMatrix = [index1,index2,beforeLengths,afterLengths,...
beforeSpherical,afterSpherical,delta,stretchAbsolute];
% Add 2 top rows for headers
finalSphericalCell = [zeros(2,20);finalSphericalMatrix];
% Convert matrix to cells
finalSphericalCell = num2cell(finalSphericalCell);
% Define headers
lengthHeader = {'Length'};
yHeader = {'Y(um)'};
xHeader = {'X(um)'};
zHeader = {'Z(um)'};
rHeader = {'r(um)'};
thetaHeader = {'Theta(deg)'};
phiHeader = {'Phi(deg)'};
stretchHeader = {'Stretch'};
absThetaHeader = {'abs(Theta)'};
absPhiHeader = {'abs(Phi)'};
beforeHeader = {'Before'};
afterHeader = {'After'};
deltaHeader = {'Delta'};
% Insert headers
finalSphericalCell(2,1) = lengthHeader;
finalSphericalCell(2,3:3:6) = yHeader;
finalSphericalCell(2,4:3:7) = xHeader;
finalSphericalCell(2,5:3:8) = zHeader;
finalSphericalCell(2,9:3:15) = rHeader;
finalSphericalCell(2,10:3:16) = thetaHeader;
finalSphericalCell(2,11:3:17) = phiHeader;
finalSphericalCell(2,12:3:17) = stretchHeader;
finalSphericalCell(2,14:3:17) = absThetaHeader;
finalSphericalCell(2,15:3:17) = absPhiHeader;
finalSphericalCell(2,16) = beforeHeader;
finalSphericalCell(2,17) = afterHeader;
finalSphericalCell(2,18) = deltaHeader;
% Delete zeros in first 2 rows
cellDelete = [1 2 3 5 6 8 9 11 12 14 15 17 18 19 20];
for i = 1:15
    finalSphericalCell{1,cellDelete(i)} = [];
end
finalSphericalCell{2,2} = [];

Save to Excel File
filename = ['C:\Users\Nector\Desktop\' distance 'Spherical.xlsx'];
sheet = ['Run' run];
range = 'A1';
xlswrite(filename,finalSphericalCell,sheet,range);
% Delete empty excel sheets
DeleteEmptyExcelSheets(filename);

Calculate Averages and Standard Deviations
% Calculate ave and std
rAve = mean(stretchAbsolute(:,1));
rStd = std(stretchAbsolute(:,1));
thetaAve = mean(stretchAbsolute(:,2));
thetaStd = std(stretchAbsolute(:,2));
phiAve = mean(stretchAbsolute(:,3));
phiStd = std(stretchAbsolute(:,3));
% Compile matrix
aveStd = [rAve thetaAve phiAve;rStd thetaStd phiStd];
% Add header row
aveStd = [zeros(1,3);aveStd];
% Convert to cell matrix
aveStdCell = num2cell(aveStd);
% Insert headers
aveStdCell(1,1) = {'stretch ave'};
aveStdCell(1,2) = {'theta ave'};
aveStdCell(1,3) = {'phi ave'};
% Save to excel file
xlswrite(filename,aveStdCell,sheet,'U2');

**Save Workspace**
workspacePath = ['C:\Users\Nector\Desktop\Spherical Workspace\' distance...'...
'Run' run '.mat'];
save(workspacePath);

*Published with MATLAB® 7.10*