

AN EXPERIMENTAL METHOD FOR VISUALIZING UNDRAINED SHEARING FAILURE IN A  
TRANSPARENT SOFT CLAY SURROGATE

BY

CHRISTOPHER MATTHEW CHINI

THESIS

Submitted in partial fulfillment of the requirements  
for the degree of Master of Science in Civil Engineering  
in the Graduate College of the  
University of Illinois at Urbana-Champaign, 2015

Urbana, Illinois

Advisers:

Research Assistant Professor Joshua M. Peschel  
Assistant Professor Cassandra J. Rutherford

## ABSTRACT

This research investigated the use of non-invasive testing procedures to visualize the failure surface of four laboratory shear-testing techniques in a soft clay surrogate, Laponite RD, at shallow depths with minimal boundary effects. The thesis developed a procedure for tracking deformation in Laponite RD as a soft clay surrogate for the miniature shear vane and the T-bar, ball, and cone penetrometers and provided the first non-invasive *in situ* images of deformation patterns. Previous attempts to visualize the shear failure surfaces of a rotating vane or penetrometers included significant disturbance of the soil sample or methods of shearing along a transparent surface. Additional existing methods involved seeding of the soil with various markers creating a non-homogenous soil mixture, adding potential sources for error. None of the previous attempts to characterize the deformation patterns surrounding the undrained shear strength testing devices provided an effective means of visualizing the deformations occurring *in situ* while minimizing potential boundary effects.

The testing set-up utilized a helium-neon laser, a laser line-generator lens, a camera, and a computer to capture and analyze resulting displacement of the soil from the various testing devices. Within the transparent soil surrogate, the laser illuminated a plane with a camera all deformation for particle tracking purposes. A series of experimental tests used two sizes of rectangular lab vanes and three types of penetrometer tests (T-Bar, ball, and cone). Miniature shear vanes and the T-Bar penetrometer each had five planes of analysis with three tests taken at each plane, resulting in 45 tests in total. The symmetry of the ball and cone penetrometers required only one plane of analysis for each device with six different tests on the plane. In all, the research included 57 independent tests using open source particle tracking algorithms and additional processing to create a repeatable methodology for future studies.

Contrary to previous studies that used particle image velocimetry (PIV) and digital image correlation (DIC), individual particle-tracking, algorithms were necessary and feasible due to the relatively sparse and random illuminated particle displays within the Laponite RD soft soil surrogate. The study provides a detailed description of the algorithms and the assumptions made in the process of tracking particles and plotting them for analysis. The results of the experiment yielded the first displacement plots of the failure surface without intrusively damaging the soil

structure due to *in situ* shear strength measuring devices. Additionally, the similarities between tests at the same planes of analysis for each testing device supported the methodology as consistent and repeatable.

## TABLE OF CONTENTS

<b>1. Introduction .....</b>	<b>1</b>
<b>2. Background .....</b>	<b>3</b>
<b>2.1 Existing Practices .....</b>	<b>3</b>
<b>2.2 Transparent Soils .....</b>	<b>4</b>
<b>2.3 Miniature Vane Shear Test.....</b>	<b>5</b>
<b>2.4 Full Flow Penetrometer Tests.....</b>	<b>6</b>
<b>2.5 Cone Penetrometer Test.....</b>	<b>6</b>
<b>3. Methodology.....</b>	<b>8</b>
<b>3.1 Testing Setup and Procedure .....</b>	<b>8</b>
3.1.1 Test Frame .....	9
3.1.2 Transparent Soil Surrogate Preparation .....	9
<b>3.2 Shear Strength Devices Experimental Methodologies .....</b>	<b>10</b>
3.2.1 Miniature Shear Vane Methodology.....	10
3.2.2 T-Bar Penetrometer Methodology .....	13
3.2.3 Ball Penetrometer Methodology .....	16
3.2.4 Cone Penetrometer Methodology .....	17
<b>3.3 Computer Vision and Particle Tracking Algorithms.....</b>	<b>18</b>

3.3.1	Miniature Shear Vane Tracking Algorithms.....	19
3.3.2	T-Bar Penetrometer Tracking Algorithm.....	23
3.3.3	Ball Penetrometer Tracking Algorithm.....	25
3.3.4	Cone Penetrometer Tracking Algorithm .....	28
<b>4.</b>	<b>Experimental Results.....</b>	<b>31</b>
4.1	Miniature Shear Vane Results.....	31
4.2	T-Bar Penetrometer Results.....	34
4.3	Ball Penetrometer Results .....	37
4.4	Cone Penetrometer Results .....	39
<b>5.</b>	<b>Discussion .....</b>	<b>41</b>
<b>6.</b>	<b>Conclusions.....</b>	<b>43</b>
<b>7.</b>	<b>References.....</b>	<b>45</b>
	<b>Appendix A – Miniature Shear Vane .....</b>	<b>48</b>
	<b>Appendix B – T-Bar Penetrometer .....</b>	<b>59</b>
	<b>Appendix C – Ball Penetrometer .....</b>	<b>67</b>
	<b>Appendix D – Cone Penetrometer .....</b>	<b>78</b>

## 1. INTRODUCTION

This thesis presents an experimental approach for visualizing four different methods of measuring undrained shear strength as conducted using a transparent soft clay surrogate. The approach provides a nonintrusive way to evaluate the deformations within a transparent soil surrogate for four commonly used methods of determining undrained shear strength: i) miniature shear vanes, ii) T-Bar penetrometer, iii) ball penetrometer, and iv) cone penetrometer. The study utilized a transparent soft clay soil surrogate, Laponite RD, based on its material property similarities to a soft clay illuminated by a laser. Laponite RD is fully transparent in normal lighting conditions allowing for unobstructed views of any testing device within the soil surrogate. The thesis developed a method to track and visualize the illuminated particle movements when subjected to the miniature shear vane, T-Bar penetrometer (at shallow depths), ball penetrometer (at shallow depths), and the cone penetrometer. The visualization of these *in situ* measurement devices is important as shear strength measurements from these measurements rely upon an understanding and quantification of the failure surface within the soil.

Various means of determining the deformation patterns of each *in situ* measurement device have previously included either numerical modeling, intrusive methods of evaluation, or methods with significant boundary affects (Section 2.1). Existing practices in transparent soils include using digital image correlation (DIC) or particle image velocimetry (PIV) (Iskander 2010). The research presents a consistent and reliable method for the display of previously unobtainable visualizations of particle deformations due to *in situ* shear strength measuring devices. Additionally, an alternative computer vision approach to conventional DIC or PIV methods that tracks individual particles obtains accurate visualizations under the given conditions.

The thesis provides a discussion of existing techniques of visualization followed by a detailed methodology for each *in situ* measurement device and accompanying experimental results. Chapter 2 describes the existing state of particle deformation techniques and current uses of transparent soils in geotechnical engineering practices. Chapter 3 provides a detailed

methodology section including testing set up and procedure (Section 3.1), experimental procedures (Section 3.2), and computer vision and particle tracking (Section 3.3). Both the experimental procedures and particle tracking sections discuss methodologies specific to each of the *in situ* measurement devices (miniature shear vane, T-bar penetrometer, ball penetrometer, cone penetrometer). Chapter 4 discusses experimental results and provides the resultant visualizations of particle displacement within the soil surrogate for each of the *in situ* measurement devices. For clarification, the study does not necessitate actual undrained shear strength calculations and comparisons between devices; instead, the study provides a means of visualizing general geometries of particle displacements. Wallace and Rutherford (2015) provide a quantitative understanding of the undrained shear strength and other properties of Laponite RD. Chini, et al. (2015), Wallace, et al. (2015a), and Wallace, et al. (2015b) provide further comparative discussions and analytical results using the methodology defined within the thesis. Discussion (Chapter 5) and Conclusion (Chapter 6) sections conclude the main body of the text. Provided appendices include the documented Matlab code developed within the scope of the thesis for the miniature shear vane (Appendix A), T-bar penetrometer (Appendix B), ball penetrometer (Appendix C), and cone penetrometer (Appendix D).

## 2. BACKGROUND

The undrained shear strength of a soil relies on an understanding of the failure surface and displacement mechanisms around the device during the testing process. This section describes previous attempts to quantify failure surfaces around *in situ* measurement devices including experimental correlations and numerical simulation, transparent soils, and the uses of the *in situ* shear strength, measurement devices.

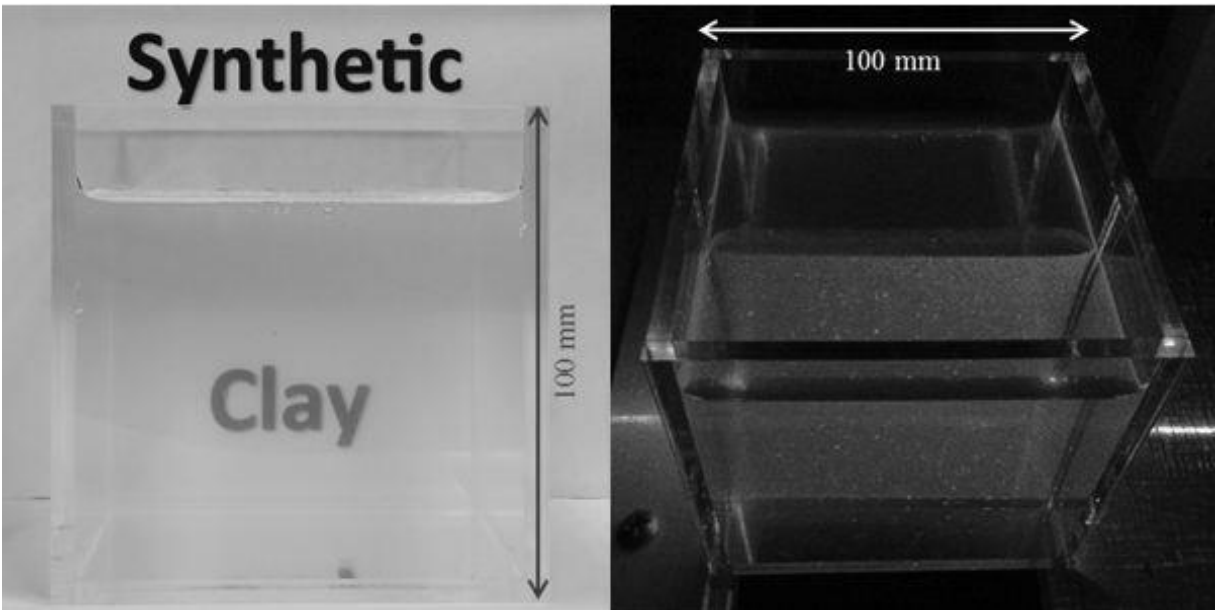
### 2.1 Existing Practices

Current studies have not visualized the deformations without disturbance of the soil. Previous techniques to record soil movement used X-rays for both a penetrometer (Francescon 1983) and a miniature shear vane (Arman et al. 1975). However, these methods either relied on intrusive techniques or seeding the soil with lead, creating a non-homogenous soil. Additionally, Gill and Lehane (2001) used transparent soils and 2 mm diameter black beads as targets to study the displacement patterns around a 12.7 mm diameter penetrometer with a flat tip.

Previous studies evaluated the soil deformations from both full flow penetrometers (T-bar and ball) and cone penetrometers through theoretical, numerical, or experimental methods. Theoretical and numerical models predicted the shallow heave type mechanism for the T-bar (Lu et al. 2000, Barbosa-Cruz and Randolph 2005, Zhou et al. 2008, White et al. 2010, Tian et al. 2011) and physical models utilized a cylindrical object placed tightly up against a transparent boundary (Dingle et al. 2007, Zhou et al. 2008). Previous assessments of the flow around a ball penetrometer included numerical analyses in which the ball was assumed to be pre-embedded within the soil, neglecting the push rod (Lu et al. 2000, Zhou and Randolph 2007, Zhou et al. 2008). Multiple studies evaluated deformations due to cone penetration in clays using theoretical solutions, numerical simulations, and experimental methods (Randolph et al. 1979, Gue 1984, Lehane and Gill 2004, Ni et al. 2010).

## 2.2 Transparent Soils

Laponite RD is an amorphous silica powder consisting of ultra-fine particles with individual diameters approximately 25 nm in width (Wallace and Rutherford 2015). The particles are hydroscopic and become transparent when mixed with distilled de-aired water. Wallace and Rutherford (2015) investigated geotechnical properties of the transparent soft clay surrogate. A visual inspection of the soil surrogate proved the applicability of the term transparency for a small sample size (Figure 1). Iskander (2010) also used the term transparent to discuss similar soil surrogates at small sample sizes. Additionally, Figure 1 shows the speckle pattern of illuminated soil particles within the sample. Table 1 provides a brief summary of the geotechnical properties for Laponite RD.



**Figure 1: Transparent Clay Soil Surrogate (left) normal light conditions showing transparency and (right) illuminated by laser to show typical observable speckle pattern.**

**Table 1: Summary of Geotechnical Properties for Laponite RD from Chini et al. (2015).**

<b>Parameter</b>	<b>Value</b>
<b>Single Crystal Dimension <sup>b</sup>, nm</b>	25 x 0.92
<b>Specific Gravity <sup>a</sup></b>	2.53
<b>Liquid Limit <sup>b</sup>, LL</b>	1150
<b>Plasticity Index <sup>b</sup>, PI</b>	910
<b>S<sub>u</sub>,miniature vane shear <sup>b</sup>, kPa</b>	0.40
<b>K <sup>b</sup>, m/s</b>	1*10 <sup>-9</sup>
<b>C<sub>c</sub> <sup>b</sup></b>	20.6
<b>C<sub>r</sub>/C<sub>c</sub> <sup>b</sup></b>	0.40
<b>C<sub>α</sub>/C<sub>c</sub> <sup>b</sup></b>	0.033
<b>C<sub>v</sub>,compression <sup>b</sup>, m<sup>2</sup>/yr</b>	0.010
<b>C<sub>v</sub>,recompression <sup>b</sup>, m<sup>2</sup>/yr</b>	0.020

<sup>a</sup> Specified by the manufacturer<sup>b</sup> Wallace and Rutherford (2015)

Previous studies utilized transparent soils, specifically, Laponite RD, to evaluate slope stability, drag anchors, liquefaction mitigation (Gidley and Grozic 2008, Beemer 2011, Bobet et al. 2012). The structure of the clay is similar to that of the natural clay mineral hectorite and consists of a sheet of octahedrally coordinated magnesium oxide between two sheets of tetrahedrally coordinated silica. Full hydration of the Laponite powder with distilled water resulted in a transparent slurry that, when consolidated, provided an appropriate medium for model testing for soft clays. Other studies utilized transparent materials to visualize non-intrusive deformations of soft soils or the flow through porous media (Mannheimer and Oswald 1993, Iskander et al. 1994, Welker et al. 1999, Gill and Lehane 2001, Song et al. 2009, Lo et al. 2010, Ni et al. 2010, Iskander and Liu 2010, Peters et al. 2011).

### **2.3 Miniature Vane Shear Test**

Offshore geotechnics utilize the laboratory vane shear test or miniature vane shear test (MV) to determine the undrained shear strength of clays. The device involves inserting a four bladed vane to the desired depth within the sample, rotating the vane, and measuring the resulting torque in order to calculate the undrained shear strength. While minor variations between the laboratory vane and the field vane shear testing procedure do exist (ASTM D2573, ASTM 4648), the method for estimating the undrained shear strength is identical.

## **2.4 Full Flow Penetrometer Tests**

The T-Bar penetrometer test is one of two full flow penetrometers used to characterize the peak and remolded undrained shear strength of soft clays with depth. Full flow penetrometers force the soft clay to flow in a viscous like manner around the instrument itself. First used to characterize soft clays in centrifuge applications (Stewart and Randolph 1991), subsequent work developed the T-Bar for field use (Stewart and Randolph 1994), specifically offshore site investigations in soft clay deposits.

Additionally, the ball penetrometer, the second of two full flow penetrometers, allows for the determination of undrained shear strength properties of soft clays with depth. Originally developed for the T-Bar, the concept of full flow penetrometers extended to implementation with a spherical penetrometer to reduce the tendency for load cell bending and allow for smaller diameter testing (DeJong et al. 2004, Randolph 2004). As opposed to the T-Bar, which assumes plane strain condition, the ball penetrometer is axisymmetric in its flow characteristics.

The failure mechanism in clay due to T-Bar or ball penetrometer tests divides into two categories based on the depth of the penetrometer: deep full flow failure and shallow upward heave type failure. Initial penetration of the penetrometer causes heave of the soil surface immediately adjacent to the penetrometer with a gap opening above the penetrometer. During shallow upward heave type failure there is still some radial deformation around the penetrometer indicating flow around the bar or ball heading towards the next stage of failure, full flow. The gap in the upward heave failure mechanism remains open until a given depth where it closes and the failure mechanism transitions to the full flow type. By definition, full flow does not occur until the gap closes above either the ball or bar.

## **2.5 Cone Penetrometer Test**

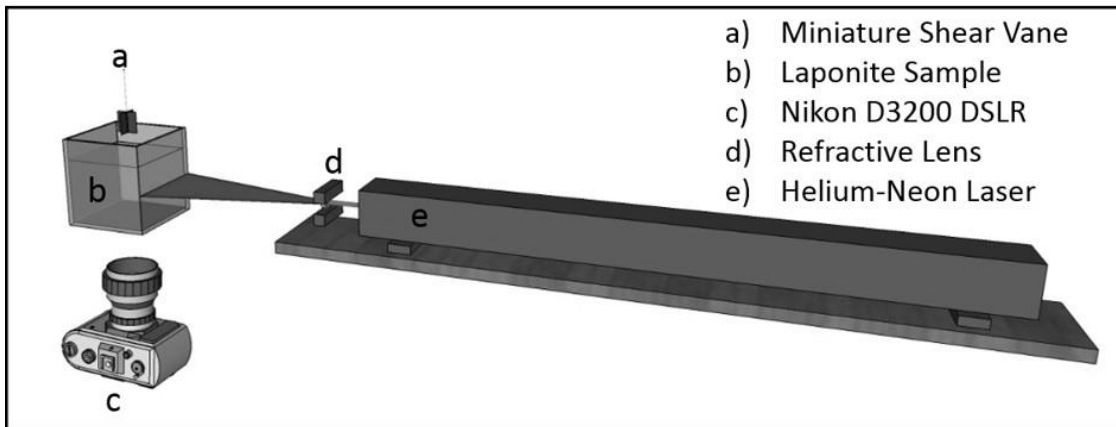
The cone penetrometer test (CPT) is an *in situ* site investigation tool used to characterize a wide range of soils. The penetrometer consists of a 60° conical tip followed by a cylindrical sleeve to measure tip and sleeve resistance respectively. The acquired measurements for soil classification using soil behavior charts such as those developed by Robertson and

Campanella (1983), Robertson, et al. (1986), and Robertson (1990) while geotechnical properties including strength and compressibility parameters, shear wave velocity, and modulus can be obtained via correlations. During penetration, the soil must displace outward to account for the volume of soil displaced by the cone and push rods. The stresses measured by the cone are a result of the induced deformations. Various theoretical and experimental studies evaluated the deformations resulting from cone penetration in clays. Previous researchers have visualized the soil deformations around a cylindrical penetrometer in transparent material; however, this work only tracked the movement of target beads rather than the movement of the particles around the penetrometer during shearing (Gill and Lehane 2001).

### 3. METHODOLOGY

#### 3.1 Testing Setup and Procedure

The experimental setup consisted of a digital camera and computer to record deformation illuminated at a specific plane within the sample using a red light emitting helium-neon laser and a line-generating lens shown in Figure 2. The vertical laser planes required the camera perpendicular to the side of the sample. The camera is a Nikon DSLR D3200 supported by a standard tripod positioned perpendicular to the side face of the sample with a view parallel to the ground for the penetrometer tests. For the miniature shear vane tests, the camera is positioned directly underneath the sample facing upwards to record the radial motion of the shearing failure. A manually focused 18-35 mm lens on the camera records video to ensure the highest clarity of the individual particles illuminated via the laser. An HDMI cable connected the camera to a computer with a game capture device as the intermediary between the camera and computer. The game capture device, Elgato Game Capture HD, included proprietary software used to record, do simple edits, and export the video feed from the camera. The proprietary software on the computer controlled the recording functions of the camera to avoid any disturbances or vibrations that may occur by interacting with the camera directly.



**Figure 2: Schematic of experimental setup for tracking deformations with bottom view camera for horizontal laser plane (miniature shear vane).**

The laser was a helium-neon laser that emitted a beam of red light not fully visible by the camera in full lighting conditions. Therefore, all lights in the laboratory were off during the

testing procedure. Two tripods with extendable supports via a hand crank allowed for manual height and location adjustment of the laser and laser plane. Leveling of the laser occurred prior to any testing and ensuring the laser is perpendicular to the face of the sample. Additionally, the laser set-up included a line lens that refracts the laser beam into a straight line of light. The line lens created either a horizontal or vertical laser plane that intersects the transparent soft clay surrogate sample. The use of a level ensured the line lens was accurate in either orientation.

### **3.1.1 Test Frame**

A wooden frame and table frame supported and provided the mount for the miniature shear vane, testing device. The miniature shear vane-testing device was vertically adjusted using threads connected to a hand crank. The motor on the device rotated the miniature shear vane at a rate of  $1^\circ/\text{sec}$ . However, this rate was not the actual rotation of the vane in the transparent soil surrogate. A calibrated spring transmitted the torque from the motor to the rotating vane, which then registered the point of which peak strength. The penetrometer tests required removal of the threads and hand crank of the miniature shear vane, testing device. In lieu of the miniature shear vane, the operators attached each penetrometer to the testing device. This modified device allowed for the manual lowering of the penetrometer into the transparent soft clay surrogate. Black foam board reduced the reflectivity of the frame and the surrounding walls from the laser and reduced ambient light from entering the testing system.

### **3.1.2 Transparent Soil Surrogate Preparation**

Hydrating of the transparent soft clay surrogate, Laponite RD, used a distilled water pore fluid at an initial concentration of 4.5% by weight and underwent self-weight consolidation for seven days prior to testing. High rotational velocity mixers hydrated the powdered, dry soft clay surrogate for twenty minutes per manufacturer's instructions. These samples self-consolidated in 100 mm x 100 mm clear plastic cubes. Plastic wrap secured with a rubber band covered each sample to avoid evaporation and a subsequent decrease in water content. Wallace and Rutherford (2015) previously had identified that a 4.5% by weight solution of

Laponite and water resulted in the highest shear strength while retaining its transparent qualities. Therefore, all samples were prepared with a 4.5% concentration, by weight, of Laponite RD powder with distilled, de-aired water. A higher concentration resulted in the mixture thickening too quickly in the mixture, not allowing for full hydration of the particles, and the transparency of the soil decreased. A lower concentration of Laponite RD resulted in lower shear strengths of the transparent soil.

During sample preparation, it was important that trapped air bubbles be nonexistent within the sample prior to testing. Air bubbles caused increase diffraction of the laser and subsequent loss of definition in particle identification. The air bubbles tend to light up larger portions of the sample outside the targeted plane, which distorts the results of the tracking algorithms. Air bubbles tended to saturate whole samples at random and, when encountered, necessitated discarding of the sample without testing.

### **3.2 Shear Strength Devices Experimental Methodologies**

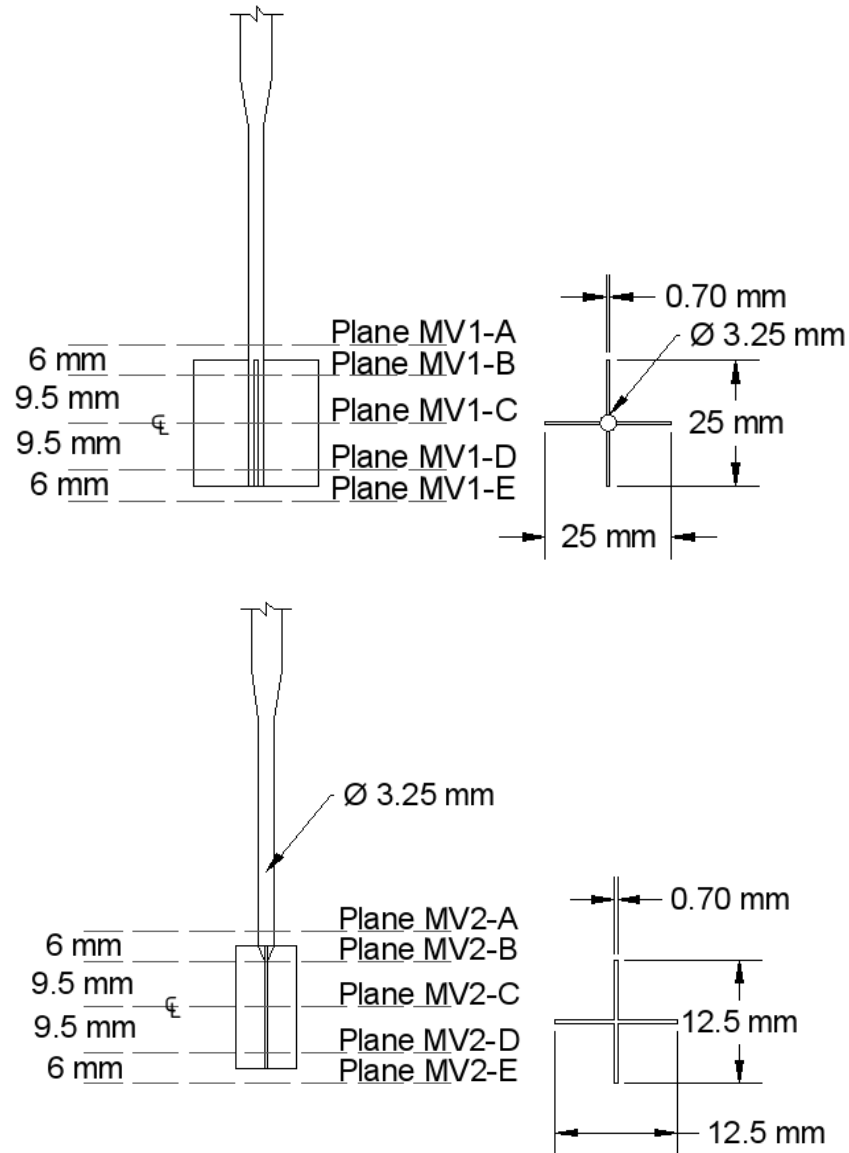
The study recorded video with multiple trials at discrete planes either horizontally for the cylindrical failure of the miniature shear vanes or vertically for the penetrometer test for the entire failure process. Video recording for each sample was in high definition at 720p and 30 frames/second (fps). The camera was set to an ISO of 3200 with an aperture setting of f5.0. Trial and error determined these settings to be the best combination to capture the proper amount of light and reduce noise in the recording. Each of the following sections further specifies the testing methodology for each respective test.

#### **3.2.1 *Miniature Shear Vane Methodology***

The camera recorded tests using a horizontal laser plane at five discrete locations along the miniature vane. The five planes included three planes on the miniature vane shear, one plane directly above, and one plane directly below while video recording occurred from the bottom of the sample looking upwards. Insertion of the miniature vane shear occurred at the middle of the 80 mm high sample at the radial center of the cube. A standard laboratory-testing device sheared the sample while operators noted the time to peak strength. The pre-peak strength and post-peak strength behavior of the soil was found to behave differently based

on Wallace, et al. (2015b). A two person testing team notated the time to peak strength with verbal communication used to start both the video and stopwatch at the same time. The stopwatch operator watched the miniature shear vane testing apparatus to determine when the dial gauge stopped moving, thus reaching peak strength. The two timing points most likely resulted in small amounts of error due to reaction time. However, comparing times to video footage yielded minimal discrepancies. A difference in a few hundredths of a second is an acceptable margin for error due to the relatively slow rotation of the vane.

The time to peak strength split the video into two distinct parts for analysis, start to peak and peak to 90° rotation. Testing included the use of two different miniature shear vanes, each at 25 mm tall with one at 25 mm in diameter and the other 12.5 mm in diameter. Figure 3 shows the discrete locations of testing that occurred along both miniature vane types. At each location, the completion of at least three tests occurred to ensure that the resultant overlays of particle displacements showed enough particles to describe adequately the movement of the soil sample.



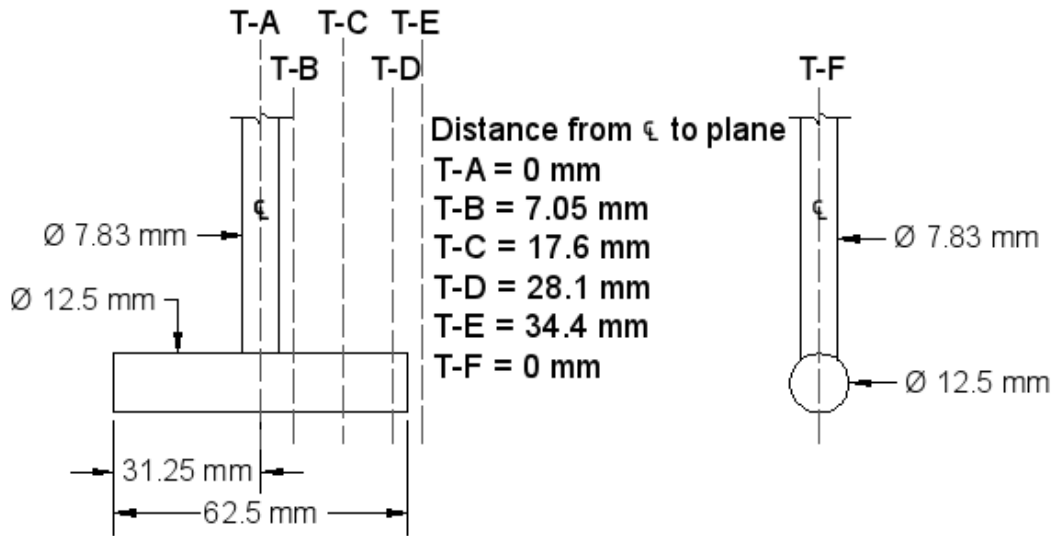
**Figure 3: Diagram and dimensions of the two miniature shear vanes used in the study shown with the locations of the vertical laser planes (MV1-A to MV1-E and MV2-A to MV2-E) along which deformation tracking occurred.**

One issue that occurred with testing at Plane MV-A, included bubbles that would form at the top of the vane due to drag as the vane was pushed into the Laponite sample. A thin layer of water on top of the clear soil surrogate prior to testing remedied the bubble formation issue. Distilled water would then occupy the voids created by the vane instead of air, resulting in less diffraction of the laser. The extra water was on the sample for a very short amount of time not allowing for permeation into the sample. Shear strength measurements from the

miniature shear vane were not affected using this method when compared to the no water method.

### **3.2.2 *T-Bar Penetrometer Methodology***

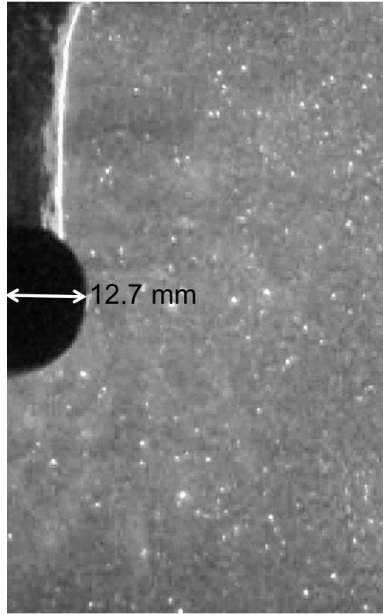
The digital camera recorded video with multiple trials at five discrete planes along one-half of the T-Bar penetrometer, assuming symmetry on the other half, while operators manually inserted the T-Bar into the soft clay surrogate. The study also consisted of a sixth test plane along the longitudinal axis of the T-Bar. The monotonic testing procedure and the equipment used to push the T-Bar penetrometer are similar to that of the cone penetration test, but rather than a cone tip, the apparatus consists of a horizontal cylindrical bar attached perpendicularly to the end of the push rod as illustrated in Figure 4. The T-Bar penetrometer measured 63.5 mm long with a diameter of 12.7 mm. The camera recorded video for the entire failure process from initial insertion until the penetrometer reached the bottom of the sample. However, to track the shallow failure mechanism, analysis required only recordings for the upper section of the sample. Figure 4 shows the discrete locations and the dimensions of the penetrometer. For each discrete location, the methodology necessitated three tests to ensure proper video quality, sample quality, and testing procedure. The methodology resulted in the testing of 18 samples.



**Figure 4: Diagram and dimensions of the T-Bar penetrometer used in the study shown with the locations of the vertical laser planes (Plane T-A to Plane T-F) along which deformation tracking occurred.**

Shallow failure occurs from the point of initial insertion of the T-Bar in the soil to where full flow occurs around the bar. Full flow begins at the point in time and space where the gap above the bar closes. Initial tests in the 100 mm deep cubes showed that full flow began to occur towards the bottom of the cube. Figure 5 shows a typical example of the shallow depth embedment gap observed during the T-Bar testing. Figure 5 shows the gap at an approximate depth of two diameters below the surface of the sample. The gap at this stage is still uniform and exhibits near vertical walls within the soft clay surrogate.

The sixth location at which video was recorded (T-F) was not included as part of any particle tracking algorithms. Visual inspection of the video was sufficient to gain insight to the deformation processes in a qualitative sense for this case.



**Figure 5: Image showing typical gap formation above the T-Bar penetrometer during shallow depth embedment in the soft clay surrogate.**

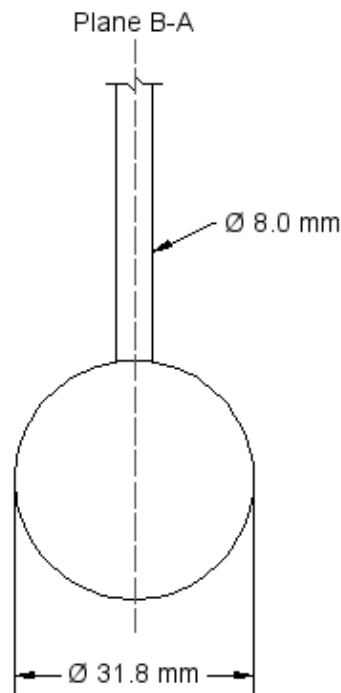
Testing in 100 mm tall cubes showed full flow to start forming towards the bottom of the sample without completely reaching full flow. Further tests in 17.8 cm tall samples determined the depth of full flow in order to negate possible boundary effects of full flow versus shallow embedment. Multiple tests confirmed that shallow depth embedment and failure governs until the center of the bar reaches a depth of about 6.32 cm or approximately five diameters below the surface of the sample. Since samples are approximately 80 mm tall, evaluating only the upper portion of the test (until a depth of 4.5 radii) was well within the shallow flow embedment type. Table 2 shows the data and individual test results from the determination of full flow depths in the 17.8 cm tall samples.

**Table 2: Determination of Full Flow Depths for T-Bar Penetrometer.**

Test	Depth to Full Flow (cm)	Depth to Full Flow (Diameters)
1	6.35	5.0
2	6.60	5.2
3	5.72	4.5
4	6.35	5.0
5	6.60	5.2
<b>Averages</b>	<b>6.32</b>	<b>4.98</b>

### 3.2.3 Ball Penetrometer Methodology

Similar to the T-Bar penetration test, a camera recorded video of the ball penetrometer tests from the side of the sample in conjunction with a vertical laser orientation. The inherent symmetry of the 31.8 mm spherical ball required only one location, the middle of the ball, for video recording with six different trials. Each test included inserting the ball vertically into the sample at a rate of approximately 6.4 mm/sec. Video recording occurred from initial insertion of the ball penetrometer to the full depth of the sample, approximately 79.4 mm. Figure 6 shows the discrete location of analysis and the dimensions of the penetrometer.



**Figure 6: Diagram and dimensions of the ball penetrometer shown with the location of the vertical laser plane (Plane B-A) along which deformation tracking occurred.**

Shallow failure occurs from the point of initial insertion of the ball in the soil to where full flow occurs around the bar. Full flow begins at the point in time and space where the gap above the bar closes. Initial tests in the 100 mm deep cubes showed that the gap above the ball penetrometer begins to converge towards the bottom of the cube. In order to ensure tracking of only shallow embedment failure, analysis included only the upper section of the sample, to a depth of three radii (1.5 diameters) below the sample surface. Additionally, tests

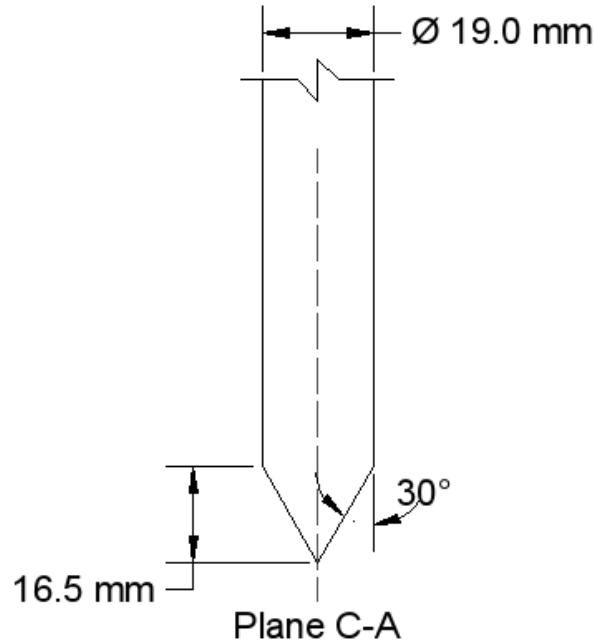
were conducted in the transparent soil surrogate on samples with a height of 17.8 cm. These ball penetrometer tests verified the depth at which shallow flow embedment failure stops and full flow begins. The governing principles for ball penetrometer full flows are the same as for the T-Bar penetrometer. Table 3 shows the results of the tall sample tests. The results indicated that shallow embedment failure governs until the center of the ball reaches a depth of approximately 8 cm or 2.5 diameters (5 radii) below the surface of the soil.

**Table 3: Determination of Full Flow Depths for Ball Penetrometer.**

<b>Test</b>	<b>Depth to Full Flow (cm)</b>	<b>Depth to Full Flow (Diameters)</b>
<b>1</b>	8.64	2.72
<b>2</b>	7.62	2.4
<b>3</b>	7.62	2.4
<b>4</b>	8.13	2.56
<b>5</b>	8.26	2.6
<b>Averages</b>	<b>8.05</b>	<b>2.53</b>

### **3.2.4 Cone Penetrometer Methodology**

Testing of the sample required a vertical orientation of the laser and recording from the side of the sample. The cone penetrometer consisted of a 19 mm diameter rod with a tip angled at 60° from the horizontal. The inherent symmetry of the cone penetrometer resulted in only one location for analysis. The testing device aided in lowering the penetrometer into the sample at a rate of 6 mm/sec. A video camera recorded the test from initial insertion until the penetrometer reached the bottom of the sample. Figure 7 shows the discrete location of analysis and the dimension of the penetrometer.



**Figure 7: Diagram and dimensions of the cone penetrometer shown with the locations of the vertical laser plane (Plane C-A) along which deformation tracking occurred.**

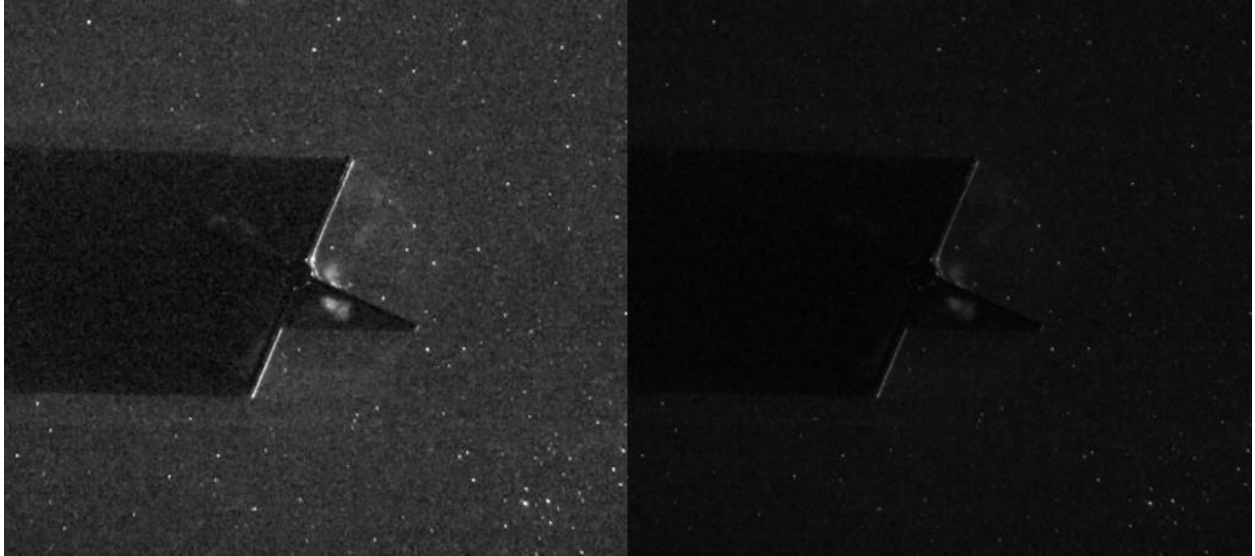
### **3.3 Computer Vision and Particle Tracking Algorithms**

Open source algorithms identified intensity peaks in individual frames for the purposes of particle tracking, which, when correlated with peaks in the next successive frame, identified particle displacement over time. The inherent speckle pattern within Laponite was relatively sparse. The particles visible with the laser were due to refractive index between the pore fluid (water) and either small pockets of air or unsaturated Laponite particles. The random particles provided the means to track the Laponite naturally, without adding a seeding particle. However, the randomness of the speckle pattern did not allow use of particle image velocimetry (PIV) or digital image correlation (DIC) techniques. The particle density was not sufficient for open source PIV or DIC algorithms to work reliably. The relatively large spacing between particles was useful for the tracking algorithm used, but PIV algorithms focused on the noise in the video capture system in excess. Additionally, there were no seeding particles placed within the surrogate, further nullifying the use of PIV and DIC (Stanier, et al. 2012). Instead, a computer vision approach tracked each particle individually.

The tracking algorithm was open source and Matlab compatible. The specific tracking algorithm was from Blair and Dufresne (2008), track.m, and accompanying files. Initial tests on known particle movements determined the accuracy of the algorithm. The accuracy tests were successful under the given conditions, speaking to the accuracy of the tracking algorithm. No speckle patterns were the same across any of the multiple tests performed for each device. Therefore, repeatability was reliant upon the general observed failure geometries of multiple tests. With respect to this qualitative analysis, the tests were repeatable using the algorithm. As lighting conditions were constant for all tests, subsequent analysis used similar inputs and parameters. The constant conditions and accurate tracking algorithm created a reliable methodology to determine failure geometries. In conclusion, the randomness and inherent nature of the Laponite soil surrogate did not allow for the application of PIV and DIC tracking algorithms. Additionally, qualitative analysis of reliability and repeatability through multiple tests supported the proposed methodology within this study, while accuracy of the algorithm was independently verified using known movements of particles. The following subsections provide specific algorithms for each of the analyzed devices (miniature shear vanes, T-bar, ball, and cone penetrometers).

### ***3.3.1 Miniature Shear Vane Tracking Algorithms***

The tracking algorithm for the miniature shear vane involved a significant amount of preparatory work prior to the final tracking function. The first step in the video analysis process required the exporting of frames from the video. The times for peak strength recorded in the testing stage of the sample provided the basis for exporting the video. Exportation of the video included with only the red band for the gray scale image and cropped based on a visual inspection of the raw video. Figure 8 shows the difference in resolution that resulted between traditional gray scale images (a weighted average of all three bands) (right) and just the red band of the image (left).



**Figure 8: Comparison of Exporting Just the Red Band of Image (left) versus a grayscale image (right).**

A Sobel edge detection function highlighted major contrast differences in the gray scale cropped video. This function outlined both particles and the edges of the miniature shear vane. Following the edge detection algorithm, a real-space band pass function filtered the resultant image. The function filled in the outlined edges of the particles to create a uniform black and white image with clearly identified particles. Use of the real space band pass filter and edge detection methodology cleaned up the image and filtered out noise from the video. Figure 9 shows a typical exported miniature shear vane image next to its gray scale frame counterpart.



**Figure 9: Typical image of miniature shear vane gray scale (left) and after frame exporting (right).**

In addition to exporting frames, it was necessary to obtain a scale and an origin point for the frames. The miniature shear vane rotated around the push rod at the center of the vane with each flange of the vane being the same length. Therefore, a scale that is the length of the radius of the vane and an origin point at the center of the vane was the logical decision. The center of the vane remained constant during the shearing process. To obtain the scale and center of the miniature shear vane, user input was determined to be the most accurate means of analysis. For each test, a graphical user interface presented to the user displayed an image of the miniature shear vane from one of the frames, prompting the user to click on three of the ends of three blades of the miniature shear vane in any order. From the three points, calculation of a unique circle occurred with its center located at the origin point and its radius is equal to the length of the miniature shear vane's blades.

The next step in the process involved implementing the tracking algorithms on all frames. Each exported frame received a Hough lines function. Since the exported frames have both the vane itself and the particles highlighted, it is important to eliminate the vane itself from the frame to avoid any interference in the results. The Hough lines algorithm analyzed the image and determined if there exists a configuration of pixels forming a line (a blade of the

miniature shear vane). From the determined start and ends of these lines, the image is altered by blacking out all pixels within a given buffer area of the lines. This creates an image with just the particles themselves highlighted.

Following this process, a function determines intensity peaks within the image. The function requires certain particle sizes and determines the locations in terms of coordinates for all the particles within the image. When combined across all frames noting the frame number, these peaks provide the necessary inputs for the tracking algorithm. The frame number, in this method acted as a time value. The tracking function on the determined array of points and time values returned a sorted list of points, times, and particle identification numbers. Both the peak finding algorithm (pkfind.m) and tracking algorithm (track.m) were from open source code banks (Blair and Dufresne 2008).

The tracking algorithm's results were the final input necessary prior to data plotting. In order to display the data, a function cycled through all the returned points from the tracking algorithm, sorting the data based on particle identification numbers. A buffer array stored points of like particle numbers before exporting to a second function. The second function took all points for the given particle and fit a one-dimensional line to the data. Each particle that experienced movement resulted in the exportation of the start and ends of a fitted vector. To help eliminate noise in the data, a minimum displacement requirement helped remove particles that seemed to vibrate back and forth during the video noise from the results and subsequent analysis.

Normalization of data occurred based on the coordinates of the center of the miniature shear vane and the radius of the blades. The point at which both the x and y coordinates equal zero was set to be the center of the miniature shear vane. Arrow vectors showed direction of the particle and distance traveled. A semicircle on the graph shows the boundary of the assumed cylindrical failure plane that lands on the edge of the miniature shear vane's blades.

### 3.3.2 T-Bar Penetrometer Tracking Algorithm

The T-Bar penetrometer tracking algorithm involved multiple steps including preparing and exporting the video, tracking the particles on individual frames, and displaying the results. The first step in the video analysis progress required the exporting of frames from the video. Exporting of video frames began when the center of the T-Bar penetrometer is located at the top of the sample. This frame was the zero location of the T-Bar in the y-direction. Frame exports occurred for the red band of the raw video, cropped based on visual inspection. A Sobel edge detection function highlighted major contrast differences in the gray scale cropped video. This function outlined both particles and the edges of the miniature shear vane. Following the edge detection algorithm, a real-space band pass function filtered the resultant image. The function filled in the outlined edges of the particles to create a uniform black and white image with clearly identified particles. Use of the real space band pass filter and edge detection methodology cleaned up the image and filtered out noise from the video. Figure 10 shows a typical resultant image following processing next to its gray scale counterpart.

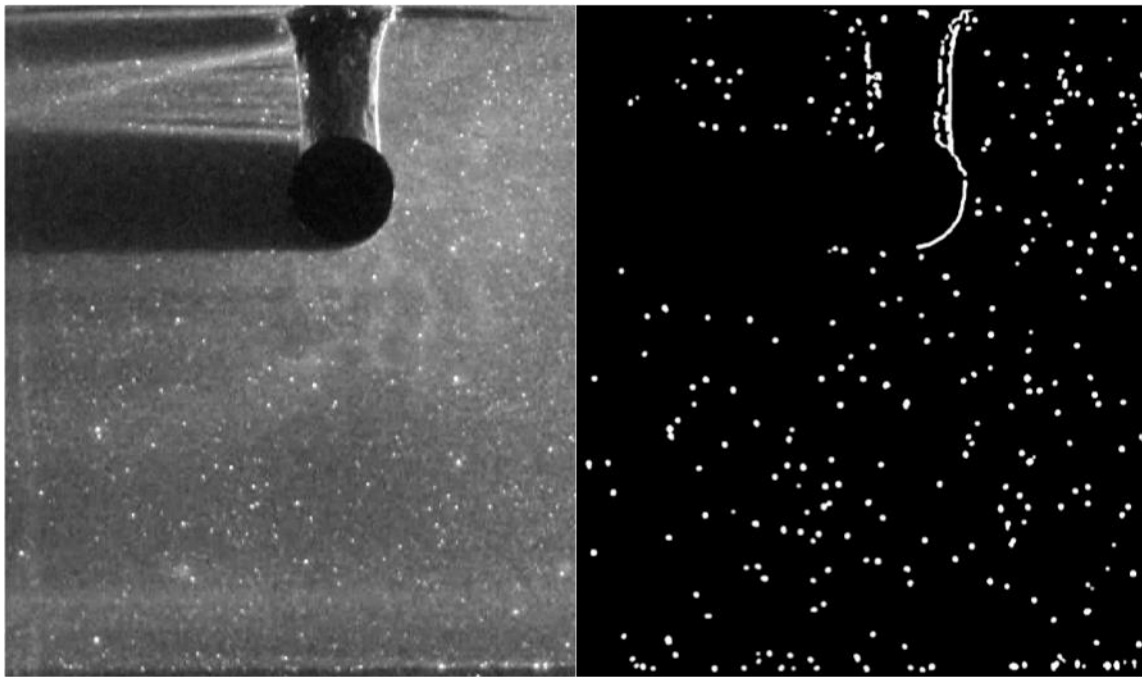


Figure 10: Typical frame of T-Bar penetrometer gray scale (left) and after frame exporting (right).

In addition to exporting frames, it was necessary to obtain a scale and an origin point for the frames and to track the T-Bar at every frame for the purposes of displaying the results correctly in plots. Additionally, normalizing the data in a generalized plot required the radius of the T-Bar penetrometer. To track the T-Bar penetrometer, conversion of the gray scale frame to a black and white image occurred using an assessed pixel index value for intensity contrasts. All pixels less than that value were zero (black) with all values greater than that value changed to 255 (white). Using this step each frame was black where the T-Bar was located including its shadow and white everywhere else. Trial and error determined the index value with visual displays to support the appropriateness of the value. From the resultant image, the sum of pixels in each row was determined with the lowest values corresponding to the location of the T-Bar penetrometer. The width of the trough in the summation data as well as its locations provided an accurate determination of the penetrometer's location and size at every frame. An average of the size and horizontal location of the penetrometer determined values for radius and x location, respectively.

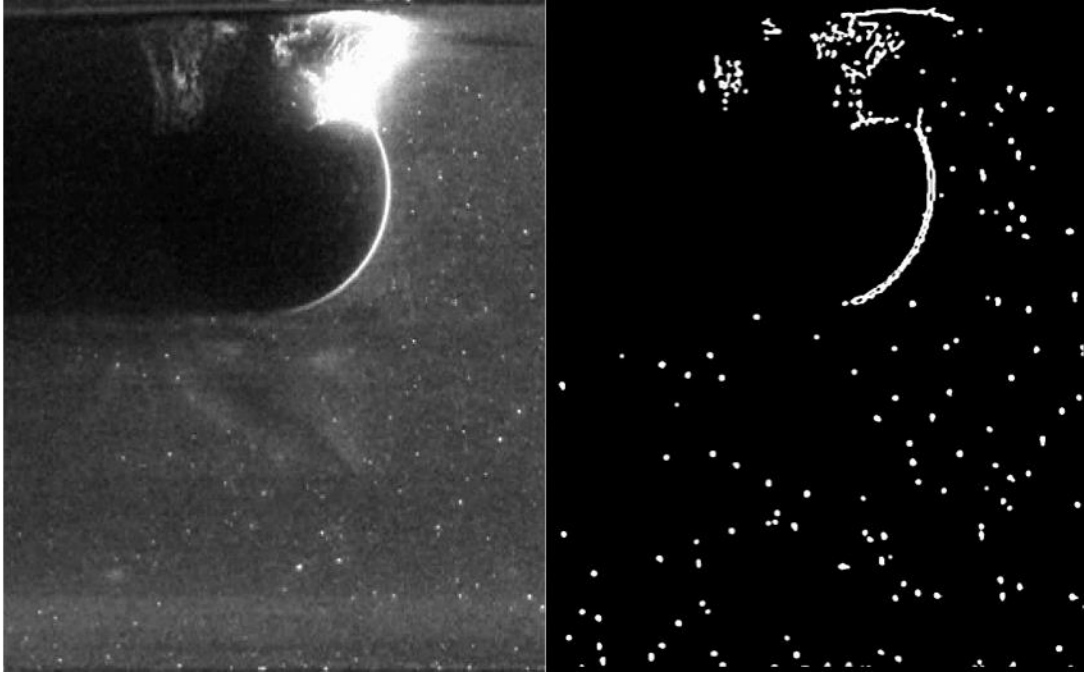
The next step in the process involved implementing the tracking algorithms on all frames. Open source algorithms determined intensity peaks in each frame, combined these peaks with time values, and then tracked each particle throughout time (Blair and Dufresne 2008). The resultant from the functions was an array of values that included x and y coordinates in terms of pixels, time values, and particle identification numbers.

The tracking algorithm's results were the final input necessary prior to data plotting. In order to display the data, a function cycled through all the returned points from the tracking algorithm, sorting the data based on particle identification numbers. A buffer array stored points of like particle numbers before exporting to a second function. The second function took all points for the given particle and fit a one-dimensional line to the data. Each particle that experienced movement resulted in the exportation of the start and ends of a fitted vector. To help eliminate noise in the data, a minimum displacement requirement helped remove particles that seemed to vibrate back and forth during the video noise from the results and subsequent analysis.

The horizontal location of the T-Bar and the radius of the penetrometer normalized the data. The point at which both the x and y coordinates equal zero equated to the top of the sample at the horizontal location that the T-Bar entered the transparent soil surrogate. Arrows showed direction of the particle and distance traveled. Three graphs divided the resultant particle displacements based on the determined vertical location of the T-Bar at every given frame. Two semicircles on each graph show the start (dashed line) and end (solid line) location of the T-Bar for each respective condition.

### **3.3.3 Ball Penetrometer Tracking Algorithm**

The ball penetrometer tracking algorithm involved multiple steps including preparing and exporting the video, tracking the particles on individual frames, and displaying the results. The first step in the video analysis progress required the exporting of frames from the video. The first exported frame of the video is the time at which the center of the ball penetrometer is located at the top of the sample. This location is the zero location in the y-direction. Frame exports occurred for the red band of the raw video, cropped based on visual inspection. A Sobel edge detection function highlighted major contrast differences in the gray scale cropped video. This function outlined both particles and the edges of the miniature shear vane. Following the edge detection algorithm, a real-space band pass function filtered the resultant image. The function filled in the outlined edges of the particles to create a uniform black and white image with clearly identified particles. Use of the real space band pass filter and edge detection methodology cleaned up the image and filtered out noise from the video. Figure 11 shows a typical gray scale frame and the resultant exported frame from the ball penetrometer tests.



**Figure 11: Typical frame from ball penetrometer video, gray scale image (right) and after exporting frame (left).**

In addition to exporting frames, it was necessary to obtain a scale and an origin point for the frames by tracking the ball at every frame for the purposes of displaying plots. The radius of the ball penetrometer normalized the data into a generalized plot. Prior to tracking, the horizontal location and scale of the image were determined based upon user input. A graphical user interface displayed with three different gray scale frames of the video. For each of these three frames, the user clicked three points along the edge of the ball penetrometer. From these three points, the algorithm generated a circle with the radius corresponding to the radius of the ball, in pixels, and the horizontal location of the center corresponding to the horizontal location of the ball penetrometer. This graphical user interface resulted in three circles averaged together to estimate the constants of horizontal location and radius to be used throughout the tracking process.

To track the vertical location of the ball penetrometer, the algorithm converted the gray scale frame into a black and white image using an estimated pixel index value based on image intensity. All values less than that value became zero (black) with all values greater than the value changed to 255 (white). Using this methodology, each frame was black where the ball

was located including its shadow and white everywhere else. The index value was determined through trial and error with visual displays to determine the accuracy of the value. From the image, the sum of pixels in each row could be determined with the lowest values corresponding to the location of the ball penetrometer. Searching from the bottom to the top of the image by rows allowed the program to determine at which row the bottom of the ball penetrometer began. The established radius could then be included into the equation resulting in the vertical location of the center of the ball penetrometer. Factors such as the index value and a vertical shift were included as inputs to the exporting function with a visual display to provide a means of verifying the trial and error approach to these values.

The next step in the process involved implementing the tracking algorithms on all frames. Matlab scripts determined intensity peaks in each frame then combined peaks with time values before tracking applying tracking algorithms (Blair and Dufresne 2008). The resultant array included x and y coordinates in terms of pixels, time values, and particle identification numbers.

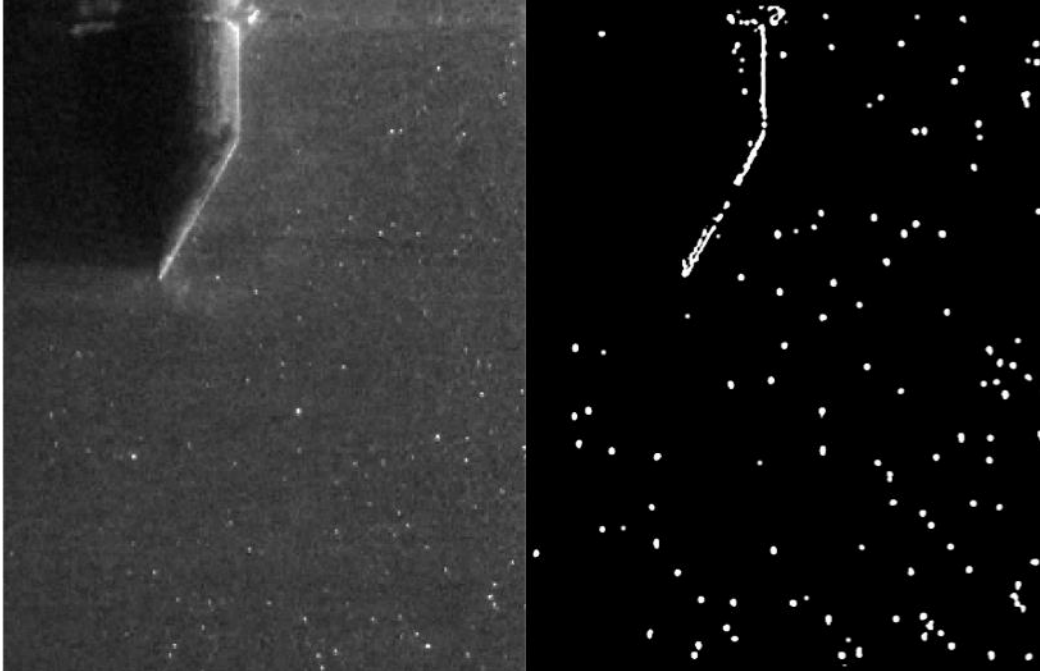
The tracking algorithm's results were the final input necessary prior to data plotting. In order to display the data, a function cycled through all the returned points from the tracking algorithm, sorting the data based on particle identification numbers. A buffer array stored points of like particle numbers before exporting to a second function. The second function took all points for the given particle and fit a one-dimensional line to the data. Each particle that experienced movement resulted in the exportation of the start and ends of a fitted vector. To help eliminate noise in the data, a minimum displacement requirement helped remove particles that seemed to vibrate back and forth during the video noise from the results and subsequent analysis.

The algorithm then normalized the data based on the horizontal location and the radius of the ball penetrometer. The point at which both the x and y coordinates equal zero equated to the top of the sample at the horizontal location that the ball penetrometer entered the transparent soil surrogate. Arrows show direction of the particle and distance traveled. Two plots divide the results based on the determined vertical location of the ball penetrometer at

every given frame. Two semicircles on each graph show the start and end location of the ball penetrometer for each respective condition.

#### **3.3.4 Cone Penetrometer Tracking Algorithm**

The cone penetrometer tracking algorithm involved multiple steps including preparing and exporting the video, tracking the particles on individual frames, and displaying the results. The first step in the video analysis progress required the exporting of frames from the video. The first exported frame of the video occurred at the time when the angled portion of the cone was fully inserted into the sample (i.e. the tip of the cone is 16.5 mm below the top of the sample). This location was the zero location in the y-direction. The algorithm exported only the cropped red band of the video. A Sobel edge detection function highlighted major contrast differences in the gray scale cropped video. This function outlined both particles and the edges of the miniature shear vane. Following the edge detection algorithm, a real-space band pass function filtered the resultant image. The function filled in the outlined edges of the particles to create a uniform black and white image with clearly identified particles. Use of the real space band pass filter and edge detection methodology cleaned up the image and filtered out noise from the video. Figure 12 shows a typical gray scale frame and the resultant exported frame from the cone penetrometer tests.



**Figure 12: Typical frame from cone penetrometer test gray scale (right) and after exporting (left).**

As one can observe in Figure 12, the algorithm clearly shows a defined vertical interface of the cone penetrometer with the soil surrogate. This interface is useful for obtaining a scale and origin point for each frame. A Hough lines transform detected the longest line in the last frame of analysis corresponding to the horizontal location of the penetrometer. Additionally, it was necessary to determine a constant scale for each video. A graphical user interface that allowed the user to select the diagonal portion of the cone penetrometer in multiple frames determined the scale of the test. The length of the user-inputted lines converted to a radius value, in pixels, for the cone penetrometer.

The next task involved tracking, vertically, the tip of the cone in the sample. A pixel intensity value based on trial and error created a black and white image. All values less than the intensity value became zero (black), while greater values changed to 255 (white). The result was a black and white image where the cone penetrometer and its cast shadow were black with all other values white. Summing the pixels in each row corresponded to the lowest values equal to the location of the cone penetrometer and shadow. Searching from the bottom to the top of the image allowed the program to determine the vertical location of the cone tip. Factors such as the index value and a vertical shift were included as inputs to the

exporting function with a visual display to provide a means of verifying the trial and error approach to these values.

The next step in the process involved implementing the tracking algorithms on all frames. Algorithms determined intensity peaks in each frame, combined peaks with time values, and tracked each particle throughout time (Blair and Dufresne 2008). The resultant array included x and y coordinates of particles in terms of pixels, time values, and particle identification numbers.

The tracking algorithm's results were the final input necessary prior to data plotting. In order to display the data, a function cycled through all the returned points from the tracking algorithm, sorting the data based on particle identification numbers. A buffer array stored points of like particle numbers before exporting to a second function. The second function took all points for the given particle and fit a one-dimensional line to the data. Each particle that experienced movement resulted in the exportation of the start and ends of a fitted vector. To help eliminate noise in the data, a minimum displacement requirement helped remove particles that seemed to vibrate back and forth during the video noise from the results and subsequent analysis.

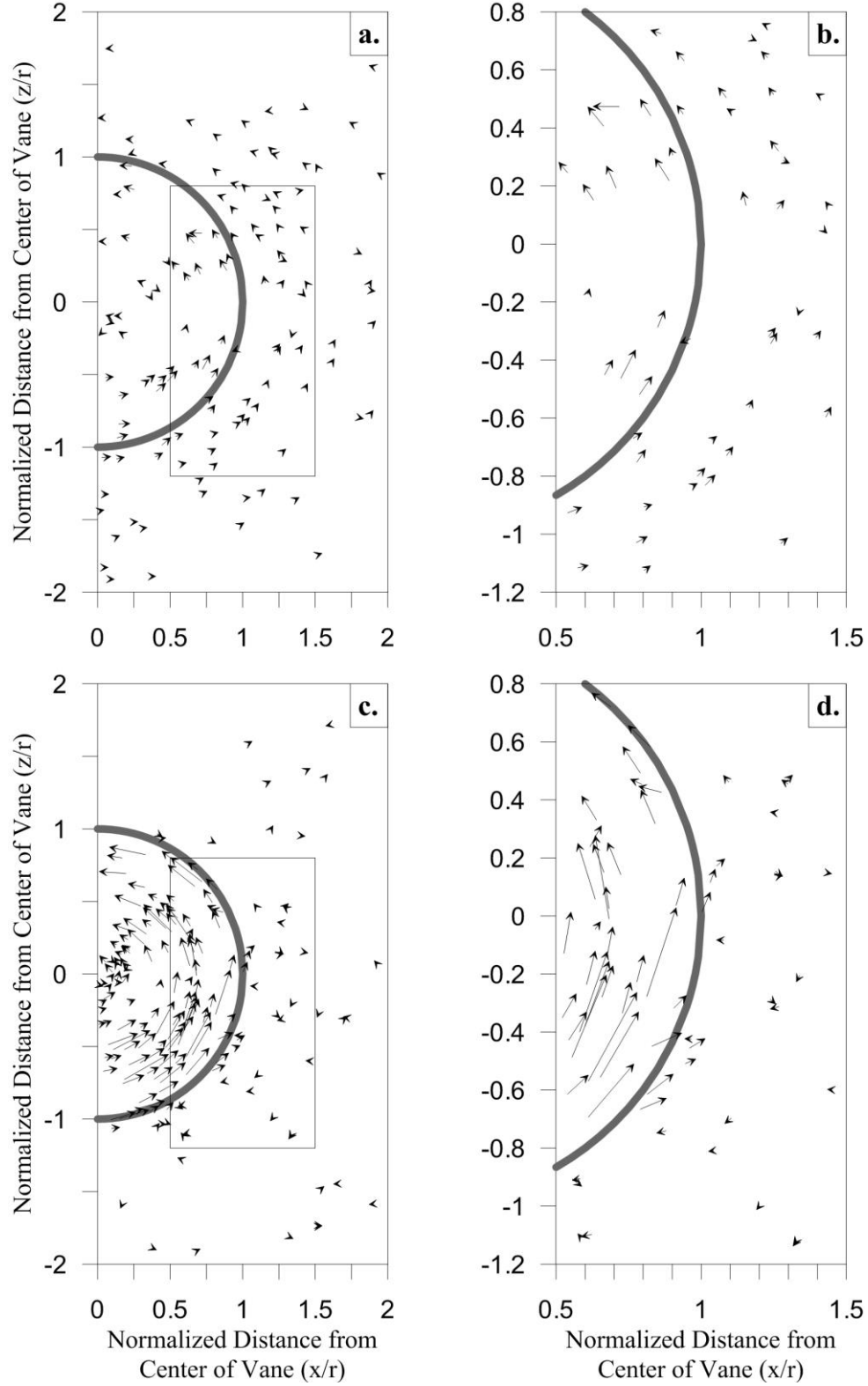
The algorithm normalized data based on the horizontal location and the radius of the cone penetrometer. The origin point for the frame equates to the top of the sample and at the center of the cone penetrometer. Arrows show direction of the particle and distance traveled. The vertical location of the cone penetrometer divided the results into three graphs. Dashed and solid lines indicate the initial and final positions of the cone, respectively.

## **4. EXPERIMENTAL RESULTS**

### **4.1 Miniature Shear Vane Results**

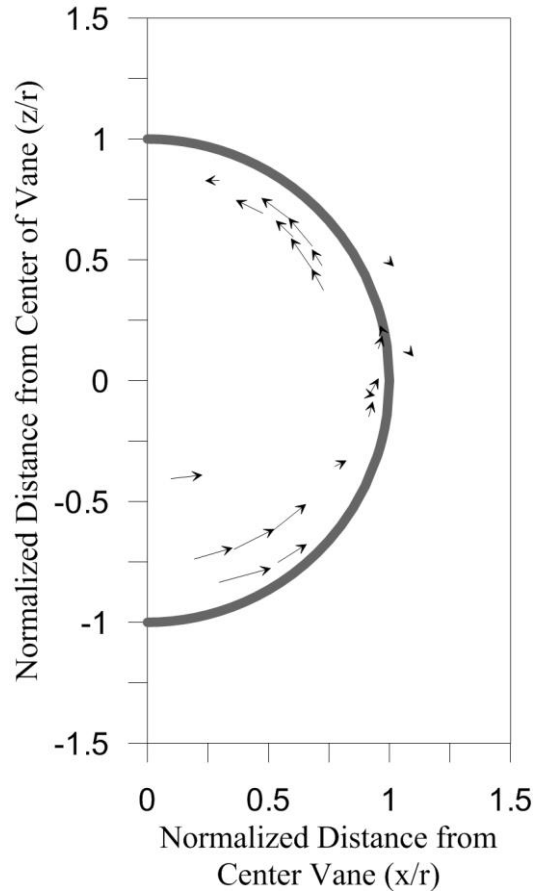
Analysis of particle displacement included results on a plane-by-plane basis as well as based on the time to reach peak undrained shear strength (peak strength). It was determined that no motion occurred at Plane MV1-A or Plane MV1-E (also MV2-A and MV2-E) vertically outside the miniature shear vane, which was evident based on a lack of particle movement in the particle-tracking algorithm at each location. The finding was consistent with the assumption of a cylindrical failure surface that is tangent to the top, bottom, and edges of the miniature shear vane.

Rotation of the miniature shear vane is broken into two distinct phases in the analysis, start to peak and post peak displacement. The vector arrows indicate the distance and direction of the particles displacement during the shearing process. The generalized displacement of the particle uses a linear interpolation of the particle's path across the relevant frames. A linear interpolation is valid based on the relatively small displacements of particles. For particles tracked over a larger distance, their trajectories were broken into smaller sections then linearly interpolated into the vector arrows. The right of each main figure provides a higher level of detail for individual particles in an enlarged section of the main figure. The boxes in the larger figures indicate the viewing frame of the respective zoomed in figure to the right of the initial diagram. The gray arc in all graphs represents the edge of the miniature shear vane as it rotates within the visualized plane.



**Figure 13: Results of tracking algorithm on the 25 mm by 25 mm miniature shear vane at Plane MV1-C**  
a) displacement from start to peak, b) displacement from start to peak zoomed in, c) displacement post peak, and d) displacement post peak zoomed in (Chini et al. 2015).

Figure 14 shows the plotted deformation of Plane MV2-C for the 25 mm by 12.5mm miniature shear vane from peak strength to the end of rotation



**Figure 14: Results of tracking algorithm on the 25 mm by 12.5 mm miniature shear vane at Plane MV2-C for post peak motion.**

In the case of the larger miniature shear vane, time to reach peak strength from the start of the test is approximately 15 seconds, corresponding to a rotation of about 4 degrees with the vane. However, the smaller miniature shear vane reaches peak strength in about 5 seconds. This lower time results in a very small window of motion, which, when combined with the smaller area covered by the wings of the vane, results in a low density of particles to track for a small distance. Video tracking of the small window of time with a low particle density resulted in no plotting of points for the start to peak strength motion. Figure 14 combines three different tests, yet still has a lower particle density than its counterpart in Figure 13. However, it is still apparent that particle motion occurs predominantly within the radius of

the miniature shear vane. Additionally, slight deformation outside the radius of the miniature shear vane in the opposite direction of vane rotation accompanies the predominant circular particle motion within the radius of the miniature shear vane.

Aside from the discrepancies in particle density, it is clear that displacement paths are consistent between the two miniature vanes. Both miniature shear vanes exhibit a radially symmetric displacement pattern with a majority of displacement occurring within the assumed cylindrical failure surface.

#### **4.2 T-Bar Penetrometer Results**

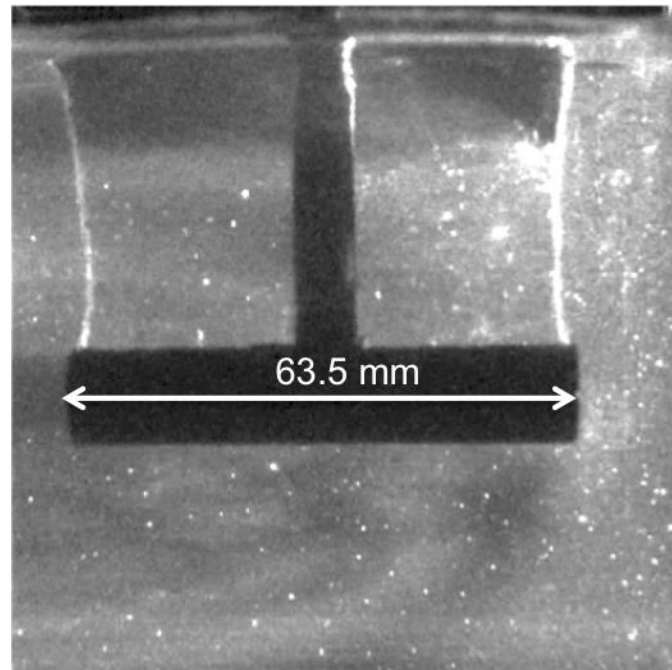
Figure 16 shows the typical case for particle deformation due to the T-Bar penetrometer, Plane T-D, since Planes T-B, T-C, T-D are very similar. The depths of embedment analyzed are between 1.75 and 4.75 diameters below the surface relative to the center of the T-Bar. The depths are below those previously visualized in laboratory models relative to pipeline embedment (Dingle et al. 2007, Zhou et al. 2008). Due to reflection off the push rod, the particle tracking algorithm was inconclusive for Plane T-A. Analysis detected no movement outside the end of the T-Bar Plane T-E). Figure 15 shows the closing of the gap observed along the length of the bar, Plane T-F.

Figure 16 is broken into three graphs of displacement based on the location of the T-Bar penetrometer in relation to the top surface of the sample. The vector arrow indicates direction of displacement and the relative magnitude of displacement, scaled by a factor of 2.5 to make the motion more visible. The zero coordinate on the y-axis correlates to the top of the transparent soil surrogate sample. The zero value on the x-axis represents the center of the T-Bar throughout its penetration into the sample.

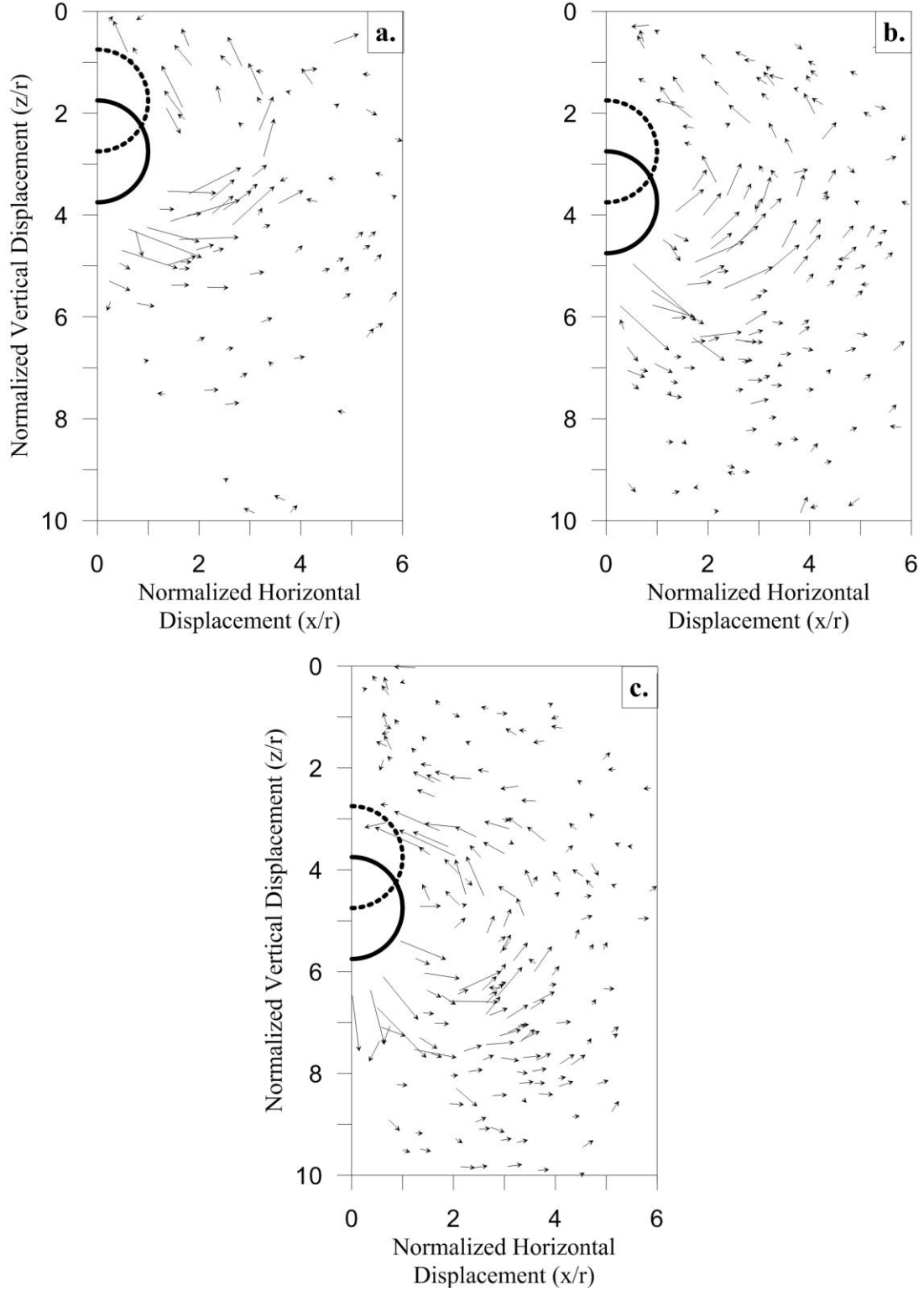
Below the horizontal midline of the T-Bar, the observed deformations are similar along the entire length of the bar. However, above the horizontal midline the deformations are similar along the length of the bar except at the pushrod location. At the pushrod, the gap formed above the bar requires less horizontal deformation to close. This observation at the pushrod was discovered based on visual analysis of the deformations recorded during penetration,

however no particle tracking at Plane T-A was able to occur due to reflection of the laser off the pushrod, causing inconclusive results in the analysis. The reflection of the laser off the pushrod illuminated large sections of the transparent clay surrogate, creating a difficult to discern distinction between individual clay particles located close to the T-Bar and pushrod.

Deformation of the transparent soil surrogate was also recorded perpendicular to the end of the T-Bar penetrometer at shallow penetration depths (Plane T-F), showing non-plane strain behavior near the ends of the bar. As seen in Figure 15, there is inward movement along the visualized plane T-F that forms a curvilinear arc from the top of the bar to the surface, which indicates that gap closure at shallow depth embedment is not isolated to a single direction around the bar.



**Figure 15: Image of T-Bar penetrometer along Plane T-F showing gap closure occurring around the cylindrical ends of the T-Bar (Wallace, et al. 2015b).**



**Figure 16: T-Bar particle tracking results at Plate T-D broken into displacement steps a) 1.75 to 2.75 radii, b) 2.75 to 7.75 radii, and c) 3.75 to 4.75 radii below the surface; displacement scaled by a factor of 2.5x (Wallace, et al. 2015b).**

### 4.3 Ball Penetrometer Results

The results of the ball penetrometer tests include the overlay of six individual tests broken up into two displacement steps. The first step shows penetration depth from 0 to 1.5 radii. The second plot shows displacement of the penetrometer from 1.5 to 3 radii below the surface of the sample. The two plots are graphs of displacement based on the location of the ball penetrometer in relation to the top surface of the sample, Figure 17. The vector arrows on each plot represent the path of a tracked particle interpolated into an assumed linear displacement. A linear interpolation of the particle path was determined to be acceptable based on the relatively small amount of motion indicated in each of the respected graphical results. The vector arrow indicates direction of displacement and the relative magnitude of displacement. The zero coordinate on the y-axis correlates to the top of the transparent soil surrogate sample. The zero value on the x-axis represents the center of the ball penetrometer throughout its penetration into the sample.

The accompanying image for the plots illustrates the typical gap formed above the bar during the shallow embedment testing. Figure 17b shows a typical gap formation at a depth of 1.5 radii below the surface of the sample, with Figure 17d displaying the gap at a depth of approximately 2.5 radii. The formed gap does not allow full flow failure to occur which would show particles flowing completely around the penetrometer before angling towards the top of the ball. The gap formation did not enable completion of tracking above the ball penetrometer.

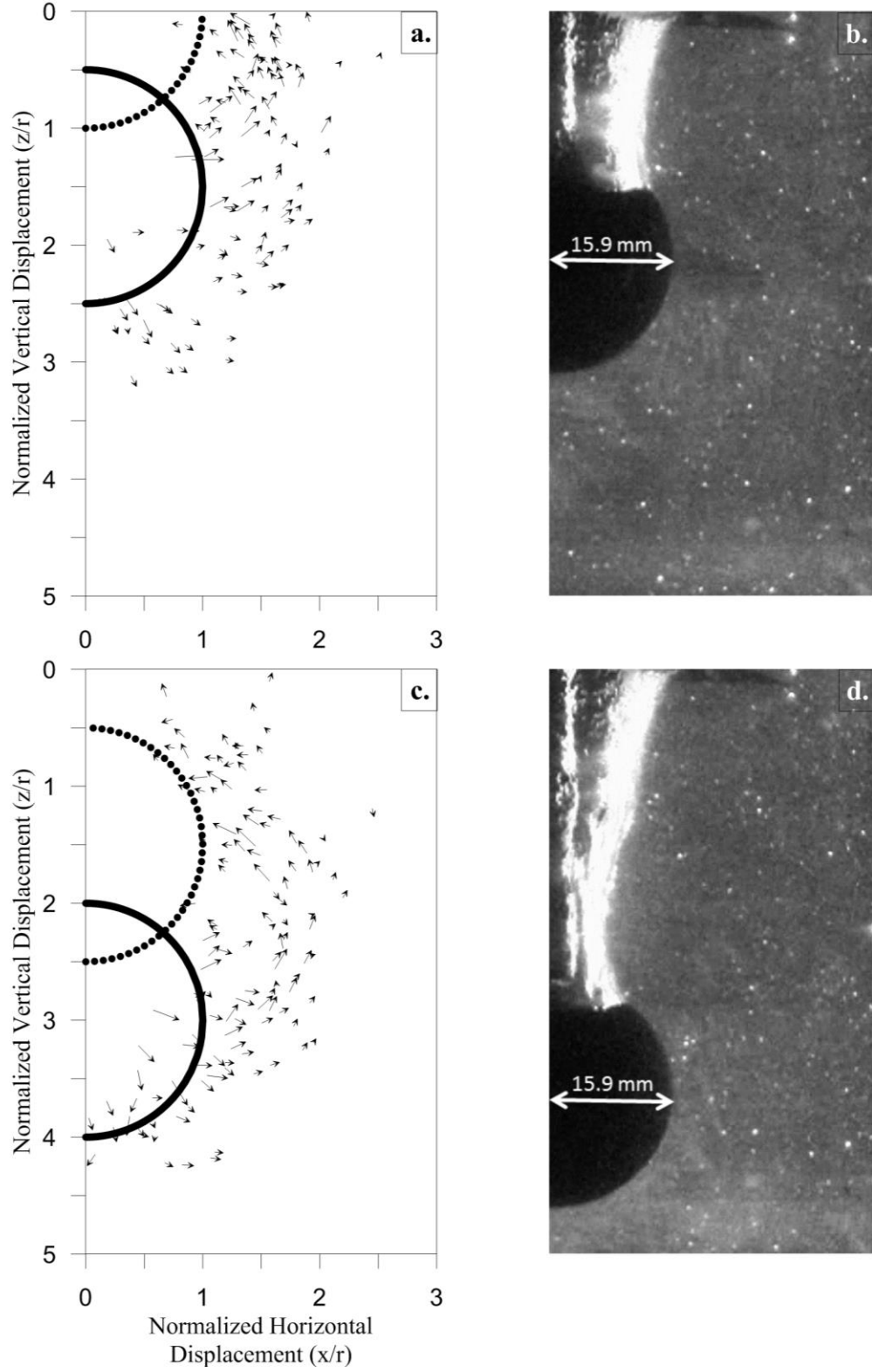


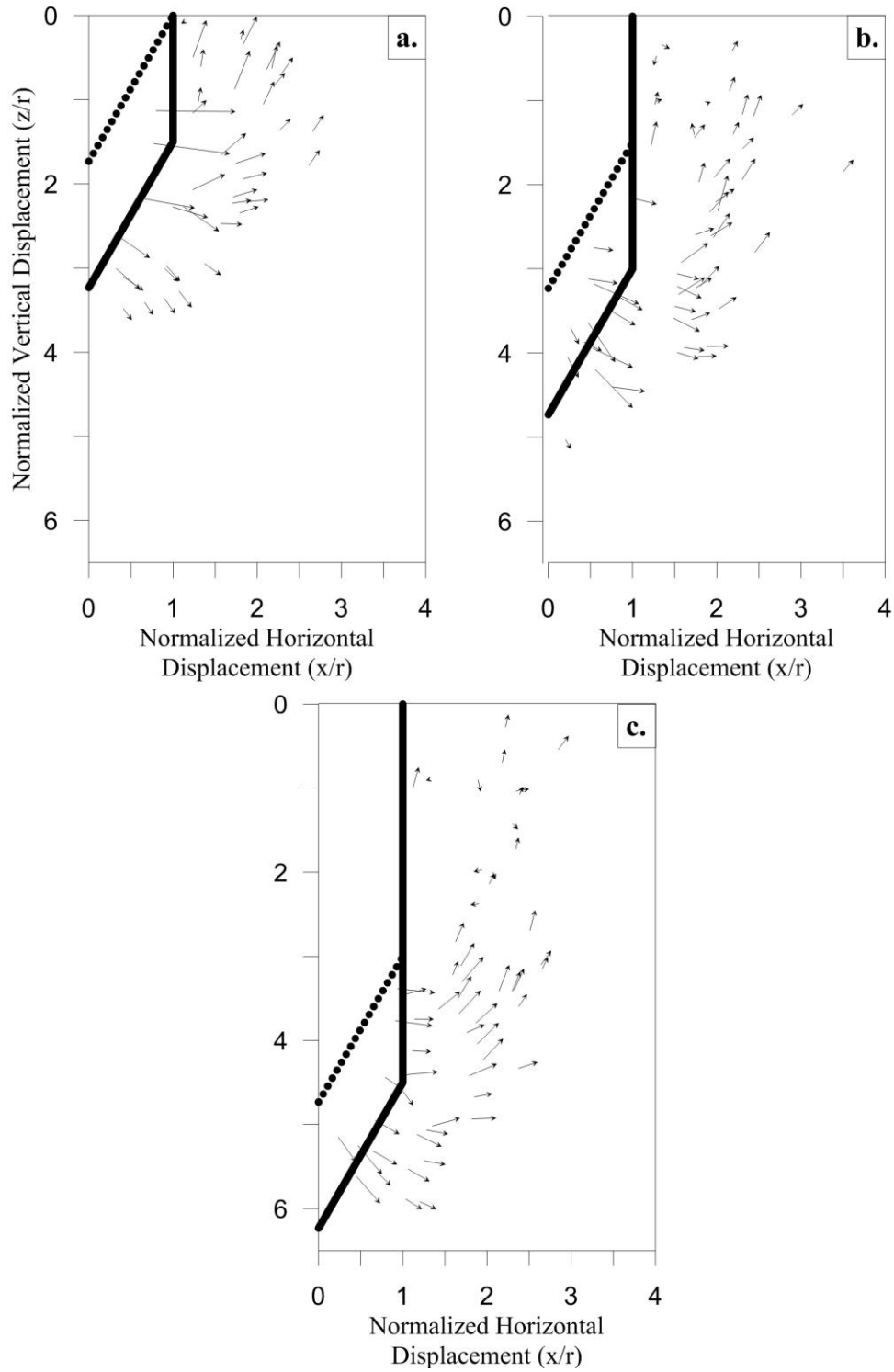
Figure 17: Ball particle tracking results at Plane B-A broken into two displacement steps a) 0 to 1.5 radii embedment and c) 1.5 to 3 radii embedment; b) a(b) and (d) show typical gap that occurs during shallow embedment failure for depths of 1.5 and 3 radii, respectively (Chini et al. 2015).

As shown in Figure 17, the failure mechanism begins to change to more of a full flow failure mechanism as the depth increases. In Figure 17a, the direction of arrows in the upper left of the plot angle up and to the left. Figure 17c shows the displacement arrows at a lower depth with the vectors in a similar location become more horizontal, pointing to the left. This change in displacement direction represents an approach from shallow upward heave failure to a full flow condition. A true full flow failure condition would have the vector arrows pointing down and to the left as the particles move in to replace the soil displaced by the ball penetrometer.

#### **4.4 Cone Penetrometer Results**

Initial insertion of the cone penetrometer into the transparent clay surrogate illuminated an interesting pattern of particle motion corresponding to a failure surface. Particles directly underneath the cone penetrometer would move away from the cone mimicking a log-spiral deformation until clearing the tip of the cone. Following clearing the tip of the cone, the particle would then stop following the log-spiral deformation and move vertically upwards parallel to the push rod of the cone penetrometer. The same particle displacement was evident in the cycling of the penetrometer within the clear soil surrogate. Figure 18 shows an overlay of five trials using the cone penetrometer and the plotted deformation from particle tracking algorithms.

Figure 18 shows the particle displacement due to cone embedment broken down into three steps. An embedment depth of 1 radii (9.5 mm) corresponds to the fully cylindrical portion of the cone being embedded 9.5 mm within the sample. With a 60° angle on the cone, a 1 radii embedment equates to the tip of the cone being a depth of 26 mm below the surface of the sample. The thick black solid lines on the plots indicate the initial position of the cone, whereas the thick dashed lines indicate the final position of the cone penetrometer for the given plot. The displacement vectors are scaled by a factor of two to better show direction of displacement.



**Figure 18: Cone particle tracking results at Plane C-A broken into displacement steps a) 0 to 1.5 radii embedment, b) 1.5 to 3 radii embedment, and c) 3 to 4.5 radii embedment below the surface; displacement is scaled by a factor of 2x (Chini, et al. 2015).**

## 5. DISCUSSION

This section discusses the major assumptions and the potential sources of error within the testing structure and algorithms. The deformation patterns associated with undrained shear strength testing devices are important as they directly correlate to the corresponding shear strength measurements. This thesis presented the algorithms and necessary steps necessary to extract displacement paths from each of the four laboratory undrained shear strength testing equipment. Within the testing structure, there were assumptions necessary to display the data. These assumptions and the inherent variability of the transparent soil speckle pattern created a potential for error within the processing of the video and displaying of results.

The first assumption pertains to the display of particles in their respective plots. Particle displacement paths were estimated based on their relatively small paths of motion, to follow linear trajectories. This simplification of particle motion is necessary to avoid excessive noise in the results. Because of this and the overlaying of multiple tests, some particle paths appear to overlap. Additionally, trial and error determined many inputs to the algorithms. The estimated values included intensity cut-offs for determining locations of penetrometers as well as adjustment values for penetrometer locations. Visual inspection of the resultant values applied to the test and intermediate output steps verified the use of the. For future replication of results, it is important that the code be modified based on user preferences to enable intermediate verification steps within the process. Other assumptions include the linear uniformity of the laser throughout the entire sample, minimal distortion of the camera lens, and minimal reflectivity of the penetrometer or miniature shear vane to distort any visualization. Any error associated with the assumptions is minimal and inconsequential in terms of results.

The testing and analysis process introduced additional error in several ways. These areas include the inherent variability of the speckle pattern within the transparent soil and the testing equipment. Multiple laboratory tests helped minimize the effect of error, human or otherwise. The ability of multiple tests to yield similar results speaks to the relatively small

impact of error and overall repeatability of the tests. The speckle pattern within the transparent soil surrogate is a result of reflectivity index mismatches between the pore fluid and either air pockets or unsaturated particles. The natural impurities in the soil surrogate enable the deformation due to shear to be tracked using computer vision algorithms. However, these impurities have an ability to compress, rotate, or fade in and out of the plane of analysis, inducing potential to track the same particle as multiple particles while it fades in and out of the frame. In the results, a large number of short displacement vectors neighboring larger displacement arrows indicate this induced tracking error. As the goal of analysis is to show the geometry of the failure surface and not necessarily the magnitude of the displacements, the error was determined to be minimal within the results. Additionally, there is inherently some error involved with the camera, computer vision tracking algorithms, and the corresponding conversion from pixel to displacement values. However, since the display of all motions is relative to the individual test, the error is minimal. The inputs for Sobel edge detection algorithms and tracking algorithms were determined using a trial and error method to minimize noise in the video. In conclusion, the sources of error were determined to be minor. Minor adjustments of particles for the induced error do not affect the result of displacement flow geometries.

## 6. CONCLUSIONS

The thesis presented a new methodology for non-destructive, visual determination of particle displacement paths due to four undrained shear strength, laboratory measurement devices (miniature shear vane, T-bar penetrometer, ball penetrometer, and cone penetrometer). The study resulted in three outcomes: i) a repeatable and reproducible method for analyzing shear deformations, ii) the first *in situ* visualizations of shear strength testing devices, and iii) previous methods for evaluating deformations in transparent soil, PIV and DIC, are not suitable for Laponite under the experimental conditions evaluated.

Each of the four differing *in situ* testing devices resulted in consistent data for the same planes of analysis regarding displacement paths across each test. The miniature shear vanes, T-Bar penetrometer, and ball penetrometer each utilized three tests at each plane of analysis. The cone penetrometer and ball penetrometer devices, considering only the single plane of analysis, used six overlapping tests for analysis. Similar displacements between tests combined with the inherent randomness of particle displacement within the transparent soil surrogate further support the repeatability and reproducibility of the methodology. The finding supports the further use of computer vision particle tracking algorithms for investigation of Laponite or similar transparent soils by materials or geotechnical researchers.

The images from the new methodology provided the first *in situ* visualizations of particle displacements from undrained shear strength, testing devices. The graphical displays of particle movement provide an opportunity to evaluate existing assumptions and validate existing numerical models of each of the *in situ* testing devices. The contribution of these images has applications in multiple facets of laboratory geotechnical research and investigations. While the algorithms currently do not allow for curvilinear displacement paths in display, the undisturbed *in situ* displacement paths provide a clear geometrical pattern for evaluation and comparison with the existing understanding of the testing device. The visualizations of the particle displacement paths have implications in future research on transparent soils and other *in situ* testing devices.

This research found that existing methods to track displacement in transparent soil surrogates such as particle image velocimetry (PIV) and digital image correlation (DIC) algorithms are not applicable to the specific transparent soil medium used in the study under the given conditions. The randomness and relatively low density of illuminated particles within the transparent soil surrogate does not allow implementation of PIV and DIC algorithms. The algorithms tend to detect noise at a high level and the relatively large spacing between illuminated particles does not allow for accurate interpolation of displacement vectors between points. The relatively small density of particles with high degrees of contrast creates an ideal medium for individual particle tracking algorithms. The transparent soil surrogate utilized contains natural random distributions of particles illuminated by the laser and allows small samples to be tested *in situ*. Larger samples, however, will tend to distort the laser illumination line due to reflectivity index mismatches and, therefore, are not suitable for previously utilized particle tracking methodologies (PIV and DIC). The use of particle tracking algorithms in lieu of PIV and DIC have benefits for further research of transparent soils similar to Laponite.

## 7. REFERENCES

- ASTM Standard D2573. 2001. Standard Test Method for Field Vane Shear Test in Cohesive Soil. West Conshohocken, PA, USA: ASTM International.
- ASTM Standard D4648. 2000. Standard Test Method for Laboratory Miniature Vane Shear Test for Saturated Fine- Grained Clayey Soil. West Conshohocken, PA, USA: ASTM International.
- Arman, A., Poplin, J. & Ahmad, N, 1975, "Study of the Vane Shear," *Proceedings of conference on in situ measurement of soil properties, Special Conference of the Geotechnical Engineering Division, ASCE*, Reston, VA, ASCE, 1: 93–120.
- Barbosa-Cruz, E.R. and Randolph, M.F., 2005, "Bearing Capacity and Large Penetration of a Cylindrical Object at Shallow Embedment." *Proc. Int. Symposium on Frontiers in Offshore Geotechnics - ISFOG 2005*, Perth, Australia: 615-621.
- Beemer, R. D., 2011, "Analytical and Experimental Studies of Drag Embedment Anchors and Suction Caissons." *M.S. Thesis*, Texas A&M University, 93 p.
- Blair, D., and Dufresne, E., 2008, The Matlab Particle Tracking Code Repository. *Matlab Particle Tracking*. Retrieved May 9, 2014, from <http://physics.georgetown.edu/matlab/>
- Bobet, A., El Howayek, A., Johnston, C., Ochoa, F., Santagata, M., and Sinfield, J., 2012, "Engineering the Pore Fluid of Sands with Highly Plastic Nano-Particles for Liquefaction Prevention." Poster. NEEShub.
- Chini, C.M., Wallace, J.F., Rutherford, C.J., and Pescehel, J.M., 2015, "Shearing Failure Visualization via Particle Tracking in Soft Clay using a Transparent Soil," *Physical Modeling with Transparent Soils, ASTM STP*, M. Iskander, and R. Bathurst, Eds., ASTM International, West Conshohocken, PA.
- DeJong, J. T., Yafrate, N. J., DeGroot, D. J., & Jakubowski, J., 2004, "Evaluation of the undrained shear strength profile in soft layered clay using full-flow probes." *Proceedings ISC-2 2nd International Conference on Site Characterization*, Rotterdam, The Netherlands, pp. 679-686.
- Dingle, H.R. C., White, D.J., and Gaudin, C., 2007, "Mechanisms of Pipe Embedment and Lateral Breakout on Soft Clay." *Can. Geotech. Journal*, Vol. 47, No. 6, pp. 636-652.
- Francescon, M., 1983, "Model Pile Tests in Clay. Stresses and Displacements Due to Installation and Axial Loading." Ph.D. Thesis, University of Cambridge.
- Gidley, I. D. and Grozic, J. L. H., 2008, "Gas Hydrate Dissociation Structures in Submarine Slopes." *4th Canadian Conference on Geohazards: From Causes to Management*, Presses de l'Universite Laval, pp. 20-24.

Gill, D.R. and Lehane, B.M., 2001, "An Optical Technique for Investigating Soil Displacement Patterns." *Geotech. Testing Journal*, Vol. 24, No. 3, pp. 324–329.

Iskander, M., 2010, *Modelling with Transparent Soils*. Springer.

Iskander, M. and Liu, J., 2010, "Spatial Deformation Measurement Using Transparent Soil." *Geotechnical Testing Journal*, Vol. 33, No. 4, pp. 314-321.

Iskander, M., Lai, J., Oswald, C., and Mannheimer, R., 1994, "Development of a Transparent Material to Model the Geotechnical Properties of Soils." *Geotechnical Testing Journal*, Vol. 17, No. 4, pp. 425–433.

Lo, H., Tabe, K., Iskander, M., and Yoon, S., 2010, "Modeling of 2D Multiphase Flow and Surfactant Flushing Using Transparent Aquabeads," *ASTM Geotechnical Testing Journal*, Vol. 33, No. 1, pp. 1-13.

Lu, Q., Hu, Y., and Randolph, M. H., 2000, "FE Analysis for T-bar and Spherical Penetrometers in Cohesive Soil." *Proceedings of the Tenth (2000) International Offshore and Polar Engineering Conference*, ISOPE, Seattle, WA, pp. 617-623.

Mannheimer, R. and Oswald, C., 1993, "Development of Transparent Porous Media with Permeabilities and Porosities Comparable to Soils, Aquifers, and Petroleum Reservoirs." *Ground water*, Vol. 31, No. 5, pp. 781–788.

Ni, Q., Hird, C., and Guymer, I., 2010, "Physical modeling of pile penetration in Clay using transparent soil and particle image velocimetry." *Geotechnique*, Vol. 60, No. 2, pp. 121–132.

Peters, S.B., Siemens, G.A., and Take, W.A., 2011, "Characterization of Transparent Soil for Unsaturated Applications," *Geotechnical Testing Journal*, Vol. 34, No. 5, pp. 445-456.

Randolph, M. F., 2004, "Characterisation of Soft Sediments for Offshore Applications," *Proc. 2nd Int. Conf. on Site Characterisation, Porto*, Vol. 1, pp. 209-231.

Robertson, P. K., & Campanella, R. G., 1983, "Interpretation of Cone Penetration Tests. Part II: Clay," *Canadian Geotechnical Journal*, Vol. 20, No. 4, pp. 734-745.

Robertson, P.K. Campanella, R.G., Gillespie, D., and Greig, J., 1986, "Use of Piezometer Cone Data," *Proceedings of ASCE Specialty Conference In Situ'86: Use of Insitu Tests in Geotechnical Engineering*, Blacksburg, Va., ASCE, New York, pp. 1263-1280.

Robertson, P.K., 1990, "Soil Classification Using the Cone Penetration Test," *Canadian Geotechnical Journal*, Vol. 27, No. 1, pp. 151-158.

Song, Z., Hu, Y., O'Loughlin, C., and Randolph, M., 2009, "Loss of Anchor Embedment during Plate Anchor Keying in Clay." *J. Geotechnical and Geoenvironmental Engrg.* Vol. 135, No. 10, pp. 1475–1485.

Stanier, S. A., Black, J. A., and Hird, C. C., 2012, "Enhancing Accuracy and Precision of Transparent Synthetic Soil Modelling." *International Journal of Physical Modelling in Geotechnics*, Vol. 12 No. 4, pp. 162-175.

Stewart, D.P. and Randolph, M.F., 1991, "A New Site Investigation Tool for the Centrifuge." *Proc. Centrifuge*, Vol. 91, pp. 531-538.

Stewart, D.P. and Randolph, M.F., 1994, "T-bar Penetration Testing in Soft Clay." *J. Geotech. Engineering*, Vol. 120, No. 12, pp. 2230-2235.

Tian, Y., Wang, D., and Cassidy, M.J., 2011, "Large Deformation Finite Element Analysis of Offshore Geotechnical Penetration Tests." *Proceedings of the 2<sup>nd</sup> International Symposium on Computational Geomechanics (ComGeo II)*, Cavtat-Dubrovnik, Croatia, pp. 925-933.

Wallace, J.F. and Rutherford, C.J., 2015, "Geotechnical Properties of Laponite RD," *Physical Modeling with Transparent Soils, ASTM STP*, M. Iskander, and R. Bathurst, Eds., ASTM International, West Conshohocken, PA, 2015.

Wallace, J.F., Chini, C.M., Rutherford, C.J., and Peschel, J.M., 2015a, "Visualizing the failure surface of a laboratory vane shear in soft clay using transparent surrogate soil." International Symposium on Frontiers in Offshore Geotechnics, Oslo, Norway.

Wallace, J.F., Chini, C.M., Rutherford, C.J., and Peschel, J.M., 2015b, "Visualizing the Shallow Failure Mechanism of the T-bar Penetrometer." IFCEE Conference, San Antonio, Texas.

Welker, A. L., Bowders, J.J., and Gilbert, R.B., 1999, "Applied Research using a Transparent Material with Hydraulic Properties Similar to Soil." *Geotechnical Testing Journal*, Vol. 22, No. 3, pp. 266-270.

White, D.J., Gaudin, C., Boylan, N., and Zhou, H., 2010, "Interpretation of T-bar Penetrometer Tests at Shallow Embedment and in Very Soft Soils." *Can. Geotech. J.* Vol. 47, No. 2, pp. 218-229.

Zhou, H., White, D. J., Randolph, M.F., 2008, "Physical and Numerical Simulation of Shallow Penetration of a Cylindrical Object into Soft Clay." *Proceedings of the ASCE GeoCongress 2008*, GSP 179, ASCE, New Orleans, La, pp. 108-117.

## APPENDIX A – MINIATURE SHEAR VANE

The following appendix contains all of the relevant Matlab code that was used to implement the algorithm and produce plots for the miniature shear vane tests. Not included within the appendix is the script file used to run all functions, due to its specificity to file name and location as well as video quality, size, etc. Enough discussion and comments are provided within the functions and the appendix to reproduce results as desired.

### Contents of Appendix

**exportMV.m:** The function reads in a video file, exports gray scale frames, applies filters, and determines the location of the miniature shear vane within each frame. Exporting of the video file is completed for two different locations relative to the time of reaching peak strength. This is the first function that needs to be run for miniature shear vane tracking.

**trackMV.m:** Using the open source tracking algorithm, this function will return a list of particles and their locations in space and time.

**plotMV.m:** The function will take the trackMV data, convert it to linear vectors, and plot each displacement vector on the appropriate graph broken up by peak strength.

## ***exportMV.m***

```
function [centerX, centerY, width] = exportMV( inputFile, ...
    output_location, times, cropDim, edgeDetect, rsbp, speed, fig)
%This function takes in videos, extracts only the red band, crops, and
%exports the video into two different sets of images (start to peak and
%peak to 90 degree rotation). The images are run through a sobel edge
%detection method and then run against a real-space band pass filter to
%help eliminate noise. Following the run, the center of the minivane along
%with the pixel width of the vane radius is determined for normalization
%purposes
%
%
% Syntax:  [] = exportMV(ParaIn,ParaIn2,...)
%
% Input parameters:
%   inputFile      - String showing where the video file is located
%   output_location - String indicating where the frames should be
%                       exported
%   times          - array of three times indicating start, peak, and
%                       end times for the video
%   cropDim        - array of decimals that indicate the amount to
%                       crop the left, right, top, and bottom of the
%                       video frame. Must be in that specific order.
%   edgeDetect     - intensity value to be used in the sobel method
%   rsbp           - vector of three values that are used to within
%                       real-space band pass method
%   speed          - Value used to determine how often frames should
%                       be exported. Variable denotes that every
%                       nth frame should be exported
%
% Output parameters:
%   centerX        - Determined center of the miniature vane in terms
%                       of horizontal pixels
%   centerY        - Determined center of the miniature vane in terms
%                       of vertical pixels
%   width          - width or radius in pixels of the miniature vane
%
% Other m-files required: CropImage.m
% Subfunctions: none
% MAT-files required: none
%
% See also: trackMV.m, plotMV.m
%
% Project:  Miniature Shear Vane Tracking
% Authors:  Chris Chini
% History:  06.16.2014  file created
%           full description at the top
%           06.20.2014  added interactive determination of center and
%           radius of the miniature shear vane

%%Export Frames from the given video file with appropriate transformations

%Reads Video File
video=VideoReader(inputFile);
```

```

%Define Start and End Frames for Analysis
startT=times(1);
peakT=times(2);
endT=times(3);

if startT>endT || startT>peakT || peakT>endT
    error('Something is wrong with your times');
end;

startFrame=startT*video.FrameRate;
peakFrame=peakT*video.FrameRate;
endFrame=endT*video.FrameRate;

%Defines the number of Frames for Output from Start to Peak
peakFrames=peakFrame-startFrame+1;
peakFrameOutput=round(peakFrames/speed)-1;

%Defines the number of Frames for Output from Start to Peak
residualFrames=endFrame-peakFrame+1;
residualFrameOutput=round(residualFrames/speed)-1;

%Declares output Location for Frames to Peak
outPeak=strcat(output_location, '_peak/FRAME');

%Declares output Location for Frames from Peak to End
outResidual=strcat(output_location, '_residual/FRAME');

%Export start to Peak Frames
for i=1:peakFrameOutput
    frame=read(video, i*speed+startFrame);
    cropped=CropImage(frame, cropDim(1), cropDim(2), cropDim(3), cropDim(4));
    red=cropped(:, :, 1);
    edgeImage=edge(red, 'sobel', edgeDetect);
    imwrite(edgeImage, 'temp1.tiff', 'TIFF');
    image1=imread('temp1.tiff');
    b=bpss(image1(:, :, 1), rsbp(1), rsbp(2));
    b=b*rsbp(3);
    file=strcat(outPeak, num2str(i), '.tiff');
    imwrite(b, file, 'TIFF');
end;

%Export Peak to End frames (residual)
for i=1:residualFrameOutput
    frame=read(video, i*speed+peakFrame);
    cropped=CropImage(frame, cropDim(1), cropDim(2), cropDim(3), cropDim(4));
    red=cropped(:, :, 1);
    edgeImage=edge(red, 'sobel', edgeDetect);
    imwrite(edgeImage, 'temp2.tiff', 'TIFF');
    image2=imread('temp2.tiff');
    b=bpss(image2(:, :, 1), rsbp(1), rsbp(2));
    b=b*rsbp(3);
    file=strcat(outResidual, num2str(i), '.tiff');
    imwrite(b, file, 'TIFF');
end;

```

```

%% Find the center of the mini vane
%The center and radius of the miniature shear vane will be determined
%interactively through the gline matlab command

%Get the initial image
frame=read(video, speed+startFrame);
cropped=CropImage(frame, cropDim(1), cropDim(2), cropDim(3), cropDim(4));
red=cropped(:,:,1);

figure(fig)
hold on
imagesc(red);
colormap(gray);
axis equal;

[point1,point2, point3]=rbtriangle();

[center, radius]=calcCircle(point1, point2, point3);

centerX=center(1,1);
centerY=center(1,2);

width=radius;

circle(centerX, centerY, width);

end

```

## ***trackMV.m***

```
function [ pointListPeak, pointListResidual ] = trackMV(file_location,...
    percentMax, particleSize, maxDistance, memory, good)
%% Tracks Particles based on the exported images in a specified file set
% The function will read in the number of files in the specified file
% location for both the peak and residual exported frames. It will return a
% point list with time or both conditions (to peak, and to final residual)
%
%
% Syntax: [pointListPeak, pointListResidual] = trackMV(file_location)
%
% Input parameters:
%   file_location      - location of files not including _peak or _residual
%                       extension
%   percentMax         - percent of the maximum intensity that will be
%                       considered a particle
%   particleSize       - minimim pixel size of particle to be analyzed
%   maxDistance       - maximum distance a particle travels between frames
%   memory            - number of frames a particle can disappear before
%                       reappearing and being considered the same particle
%   good              - variable that determines the amount of play in
%                       particle movement
%
%
% Output parameters:
%   pointList         - a list of all the points that were found in the set of
%                       images. The first column equates to the x location in pixels, the
%                       second column equates to the y location in terms of pixels. The third
%                       column references the frame number. The fourth and final column
%                       represents the ID of the particle
%
%
% Other m-files required: pkfnd.m, track.m
% Subfunctions: none
% MAT-files required: none
%
% See also: exportMV.m, plotMV.m
%
% Project:  Miniature Shear Vane Tracking
% Authors:  Chris Chini
% History:  06.17.2014  file created
%           full description at the top
%           XX.XX.2014  future iterations with change description

%% First track Start To Peak
%Determine number of frames to be analyzed
folderName1=strcat(file_location, '_peak/');
D1=dir([folderName1]);%There are three random files that get added into
%the directory so just subtract them from total
num1=length(D1)-3;

for index=1:num1
    frameName=strcat(folderName1, 'FRAME', num2str(index), '.tiff');
```

```

bp=imread(frameName);

%Eliminate all unacceptable points. Unacceptable points include points
%that are the mini-vane itself
%Logically if points are dense enough that a line can be formed, they
%should be discarded. The dense linear formation of lines should be
%attributed to the minivane itself and not the particles. The
%likelihood of multiple particles aligning to form a dense linear array
%is highly unlikely.

%This is done by finding all points within a buffer area around a line
%and setting them equal to zero
[H,T,R] = hough(bp);
P = houghpeaks(H,5,'threshold',ceil(0.3*max(H(:)))));
lines = houghlines(bp,T,R,P,'FillGap',5,'MinLength',20);
for k = 1:length(lines)
    xy = [lines(k).point1; lines(k).point2];

    x1=xy(1,1);
    x2=xy(2,1);
    y1=xy(1,2);
    y2=xy(2,2);

    m=abs((y2-y1)/(x2-x1));

    if m>=1
        A=[x1-5,y1];
        B=[x1+5,y1];
        C=[x2+5,y2];
        D=[x2-5,y2];
    else
        A=[x1,y1-5];
        B=[x1,y1+5];
        C=[x2,y2+5];
        D=[x2,y2-5];
    end;

    X=[A(1),B(1),C(1),D(1),A(1)];
    Y=[A(2),B(2),C(2),D(2),A(2)];

    minx=min([x1,x2]);
    maxx=max([x1,x2]);

    miny=min([y1,y2]);
    maxy=max([y1,y2]);

    for col=minx:maxx
        for row=miny:maxy
            if inpolygon(col, row, X, Y)==1
                bp(row, col)=0;
            end;
        end;
    end;
end;

```

```

maxVal=max(max(bp));
threshold=percentMax*maxVal;
peak=pkfnd(bp, threshold, particleSize);
peak(:,3)=index;

if index==1
    points=peak;
else
    ex=length(points(:,1));
    new=length(peak(:,1));
    points(ex+1:ex+new,:)=peak;
end;

end;

%Declare Variables for Tracking Algorithm
param.mem=memory;
param.dim=2;
param.good=good;
param.quiet=1;

%Track Particles
tr=track(points, maxDistance, param);

pointListPeak=tr;

%% Second Track Particles from Peak to 90 degree residual condition
%Determine number of frames to be analyzed
folderName2=strcat(file_location, '_residual/');
D2=dir([folderName2]);%There are two random files that get added into
                        %This directory so just subtract them from total
num2=length(D2)-3;
for index=1:num2
    frameName=strcat(folderName2, 'FRAME', num2str(index), '.tiff');
    bp=imread(frameName);

    %Eliminate all unacceptable points. Unacceptable points include points
    %that are the mini-vane itself
    %Logically if points are dense enough that a line can be formed, they
    %should be discarded. The dense linear formation of lines should be
    %attributed to the minivane itself and not the particles. The
    %likelihood of multiple particles aligning to form a dense linear array
    %is highly unlikely.

    %This is done by finding all points within a buffer area around a line
    %and setting them equal to zero
    [H,T,R] = hough(bp);
    P = houghpeaks(H,5,'threshold',ceil(0.3*max(H(:)))));
    lines = houghlines(bp,T,R,P,'FillGap',5,'MinLength',20);
    for k = 1:length(lines)
        xy = [lines(k).point1; lines(k).point2];

        x1=xy(1,1);
        x2=xy(2,1);
        y1=xy(1,2);

```

```

y2=xy(2,2);

m=abs((y2-y1)/(x2-x1));

if m>=1
    A=[x1-5,y1];
    B=[x1+5,y1];
    C=[x2+5,y2];
    D=[x2-5,y2];
else
    A=[x1,y1-5];
    B=[x1,y1+5];
    C=[x2,y2+5];
    D=[x2,y2-5];
end;

X=[A(1),B(1),C(1),D(1),A(1)];
Y=[A(2),B(2),C(2),D(2),A(2)];

minx=min([x1,x2]);
maxx=max([x1,x2]);

miny=min([y1,y2]);
maxy=max([y1,y2]);

for col=minx:maxx
    for row=miny:maxy
        if inpolygon(col, row, X, Y)==1
            bp(row, col)=0;
        end;
    end;
end;

maxVal=max(max(bp));
threshold=percentMax*maxVal;
peak=pkfnd(bp, threshold, particleSize);
peak(:,3)=index;

if index==1
    points=peak;
else
    ex=length(points(:,1));
    new=length(peak(:,1));
    points(ex+1:ex+new,:)=peak;
end;

end;

%Declare Variables for Tracking Algorithm
param.mem=memory;
param.dim=2;
param.good=good;
param.quiet=1;

```

```
%Track Particles
tr=track(points, maxDistance, param);

pointListResidual=tr;

end
```

## ***plotMV.m***

```
function [lineSegments ] = plotMV( track, figureNum )
%The function takes in the tracking data table and plots it on the
%appropriate graph corresponding to the input figure number
%
%
% Syntax:  [] = plotMV(ParaIn,ParaIn2,..)
%
% Input parameters:
%   track          - list of points in time and space along with IDs
%                   of points that correspond to the tracking
%                   algorithm completed in trackMV.m
%   figureNum      - figure ID to plot
%
% Output parameters: none
%
% Other m-files required:
% Subfunctions: none
% MAT-files required: none
%
% See also: trackMV.m
%
% Project:  Miniature Shear Vane Tracking
% Authors:  Chris Chini, Jeff Wallace, Cassandra Rutherford, Joshua Peschel
% History:  06.18.2014  file created
%           full description at the top
%           XX.XX.2014  future iterations with change description

figure(figureNum)
hold on;

idPrevious=track(1,4);
xc=track(1,1);
yc=track(1,2);
buffer=[xc yc];
%Buffer is used to track path of specific particles before plotting

numPoints =length(track(:,1));
%Tracks Particles Assuming tr is sorted based on point id

lineSegments=zeros(1,4);
lineCount=1;

for k=2:numPoints
    idCurrent=track(k,4);
    bufferSize=length(buffer(:,1));
    if idPrevious==idCurrent && bufferSize<50
        xc=track(k,1);
        yc=track(k,2);
        buffer(bufferSize+1,1:2)=[xc yc];
    else
        x=buffer(:,1);
```

```

y=buffer(:,2);

x0=x(1);
y0=y(1);
xf=x(end);
yf=y(end);

disp=sqrt((x0-xf)^2+(y0-yf)^2);

startLocation=sqrt(x0^2+y0^2);

if disp>0.02 && startLocation >0.1%Varies between 0 and 0.05
depending on size of MV
    [x1, y1, x2, y2]=plotMVFitted(x,y,1,figureNum);

    lineSegments(lineCount,:)= [x1, y1, x2, y2];
    lineCount=lineCount+1;
end;

xc=track(k,1);
yc=track(k,2);
buffer=[xc yc];
idPrevious=idCurrent;
end;
end;

end

```

## APPENDIX B – T-BAR PENETROMETER

The following appendix contains all of the relevant Matlab code that was used to implement the algorithm and produce plots for the T-Bar penetrometer tests. Not included within this appendix is the script file used to run all functions, due to its specificity to file name and location as well as video quality, size, etc. Enough discussion and comments are provided within the functions and the appendix to reproduce results as desired.

### Contents of Appendix

**prepareTBarVideo.m:** The function reads in a video file, exports gray scale frames, applies filters, and determines the location of the T-Bar within each frame. This is the first function that needs to be run for T-Bar tracking.

**findBar.m:** This function takes in an image input and determines the vertical location of the T-Bar penetrometer within the frame. The function is called within the prepareTBarVideo function.

**TrackTBarParticles.m:** Using the open source tracking algorithm, the function will return a list of particles and their locations in space and time.

**plotTBarDeformation.m:** This function will take the TrackTBarParticles data, convert it to linear vectors, and separate displacement based on location of the T-Bar in each frame for visualization.

## ***prepareTBarVideo.m***

```
function [barLocations] = prepareTBarVideo(inputFile, output_location,
startT, endT, left, right, top, bottom, speed)
%This function takes in videos, extracts only the red band, crops, and
exports
%the edges using 'sobel' method then runs a band pass filter over the sobel
%edge detection
%
%The result of this function is a set of output of noise images
%
%
% Syntax: [barLocations] = prepareTBarVideo(ParaIn,ParaIn2,..)
%
% Input parameters:
%     inputFile           - String showing where the video file is located
%     output_location     - String indicating where the framses should be
%                           exported
%     startT              - Start time for analysis on video
%     endT                - End time for analysis on video
%     left                - Crop in decimal form for the left side of the
%                           video as a percentage of the whole video width
%     right               - Crop in decimal form for the right side of the
%                           video as a percentage of the whole video width
%     top                 - Crop in decimal form for the top side of the
%                           video as a percentage of the whole video width
%     bottom              - Crop in decimal form for the bottom side of the
%                           video as a percentage of the whole video width
%     speed               - Value used to determine how often frames should
%                           be exported. Variable denotes that every
%                           nth frame should be exported
%
% Output parameters:
%     barLocations        - a matrix of locations of the bar where the row
%                           corresponds to the frame, the first column is the x coordinate, the
%                           second is the y coordinate and the final column is the computed radius
%                           of the frame
%
%
% Other m-files required: findBar.m
% Subfunctions: none
% MAT-files required: none
%
% See also: trackTBarParicles.m, trackTBarScript.m

% Project:  T-Bar Tracking Algorithms
% Authors:  Chris Chini
% History:  04.24.2014   file created
%                               full description at the top
%           xx.xx.2014   description of modification

%Reads Video File
video=VideoReader(inputFile);

%Define Start and End Frames for Analysis
```

```

if startT>endT
    error('Start Time must be before End Time');
end;

startFrame=startT*video.FrameRate;
endFrame=endT*video.FrameRate;

%Defines the number of Frames for Output
numFrames=endFrame-startFrame+1;
frameOutput=round(numFrames/speed)-1;

%Declares output Location for Frames
out=strcat(output_location, 'FRAME');

%create matrix for x,y, and radius values of the tbar for each frame that
%is being process
tbarLocate=zeros(frameOutput,3);

for i=1:frameOutput
    frame=read(video, i*speed+startFrame);
    cropped=CropImage(frame, left, right, top, bottom);
    red=cropped(:, :, 1);
    [x,y,r]=findBar(red);
    tbarLocate(i,1:3)=[x,y,r];
    edgeImage=edge(red, 'sobel', 0.12);
    imwrite(edgeImage, 'temp.tiff', 'TIFF');
    image=imread('temp.tiff');
    b=bpass(image(:, :, 1), 1, 5);
    b=b*20;
    file=strcat(out, num2str(i), '.tiff');
    imwrite(b, file, 'TIFF');
end;
figure(2)
hold on;
imagesc(red);
circle(x,y,r);
colormap(gray);

barLocations=tbarLocate;

end

```

## ***findBar.m***

```
function [ x,y,radius ] = findBar( image )
%The function takes in a frame of the T-Bar test and returns the x,y
%coordinates in terms of pixels. Additionally, the radius of the bar (in
%pixels) is returned for the purposes of scale.
%
%The function assumes that the frame is of one dimension (greyscale).
%Additionally, it assumes that the bar is a black circle. The function then
%goes and sets all values that are not black (<20) to black (pixel=0),
%making all the black pixels white (pixel=255)
%
%
% Syntax:  [x, y, radius] = findBar(image)
%
% Input parameters:
%   image      - m x n matrix of pixel values that are in the range of 0-255
%
%
% Output parameters:
%   x          - x location in pixels of the center of the bar
%   y          - y location in pixels of the center of the bar
%   radius     - radius in pixels of the bar
%
%
% Other m-files required: none
% Subfunctions: none
% MAT-files required: none
%
% See also: prepareTBarVideo.m, trackTBarScript.m, TrackTBarParticles.m

% Project:  T-Bar Tracking Algorithms
% Authors:  Chris Chini
% History:  04.24.2014    file created
%              full description at the top
%              xx.xx.2014    description of modification

[row, col, band]=size(image);

frame=image;

if band~=1
    error('Image must be of only one band');
end;

%Set all pixels to their appropriate color
for r=1:row
    for c=1:col
        pixel=frame(r,c);
        if pixel<=10
            frame(r,c)=255;
        else
            frame(r,c)=0;
        end;
    end;
end;
```

```

end;

%The center of the T-bar should be where the highest sum of pixels occurs
%in each row and column
%Sum each row and each column into two separate vector
rowVector=zeros(row,1);
colVector=zeros(1, col);

top=0;
bottom=0;
left=0;
right=0;

for r=1:row
    rowVector(r)=sum(frame(r,:));
    if top==0 && rowVector(r)>1000
        top=r;
    end;
end;
for r=row:-1:1
    if bottom==0 && rowVector(r)>1000
        bottom=r;
        break;
    end;
end;

for c=1:col
    colVector(c)=sum(frame(:,c));
    if left==0 && colVector(c)>1000
        left=c;
    end;
end;
for c=col:-1:1
    if right==0 && colVector(c)>1000
        right=c;
        break;
    end;
end;

%Radius is the average of the distance between peaks of the sums of rows
%and columns
vt=bottom-top;
hz=right-left;

radius=round((vt+hz)/4);

x=round((left+right)/2);
y=round((top+bottom)/2);

end

```

## ***TrackTBarParticles.m***

```
function [pointList] = TrackTBarParticles(tbar, outFramesLoc)
%% Tracks the particles through space and time of a set of frames
%% that have previously been exported from a video
% This is a follow up function to prepareTBarVideo.m. This function reads
% in the previously created frames, finds particles that are outside the
% bar itself, and tracks them through space and time. All locations of
% particles are normalized based on the location of the bar at any given
% time and also are normalized based on the radius of the bar. A value of 2
% equals a particle that is two radii outside the center of the T-bar
%
%
% Syntax: [pointList] = TrackTBarParticles(tbar, outFramesLoc)
%
% Input parameters:
%     tbar          - matrix of coordinates in pixels and radii of the bar
%                   at each frame to be analyzed. See findBar.m
%
%     outFramesLoc  - String containing the location for the frames to be
%                   analyzed
%
%
% Output parameters:
%     pointList     - a list of all the points that were found in the set of
%                   images. The first column equates to the x location in terms of the
%                   radius and location of the bar, the second column equates to the
%                   y location in terms of the radius and location of the bar. The third
%                   column references the frame number. The fourth and final column
%                   represents the id of the particle
%
%
% Other m-files required: pkfnd.m, track.m
% Subfunctions: none
% MAT-files required: none
%
% See also: prepareTBarVideo.m, trackTBarScript.m

% Project:  T-Bar Tracking Algorithms
% Authors:  Chris Chini
% History:  04.24.2014   file created
%                   full description at the top
%           xx.xx.2014   description of modification

%Declare Variables for Find Peak
percentMax=0.25;%What percent of the max will be the threshold
particleSize=10;

[frameNum, temp]=size(tbar);

averageRadius=sum(tbar(:,3))/frameNum;

pointNum=2;
```

```

%Determine Peak Points in Each Frame
for index=1:frameNum
    frameName=strcat(outFramesLoc, 'FRAME', num2str(index), '.tiff');
    bp=imread(frameName);
    maxVal=max(max(bp));
    threshold=percentMax*maxVal;
    peak=pkfnd(bp, threshold, particleSize);
    peak(:,3)=index;

    [rowPeak]=length(peak(:,1));

    %Eliminate all unacceptable points. Unacceptable points include points
    %that fall within the T-Bar Itself
    for p=1:rowPeak
        particleDistance=sqrt((peak(p,1)-tbar(index,1))^2+...
            (peak(p, 2)-tbar(index,2))^2);

        if particleDistance>=1.1
            positionList(pointNum,1)=peak(p,1)/averageRadius;
            positionList(pointNum,2)=peak(p,2)/averageRadius;
            positionList(pointNum,3)=peak(p,3);
            pointNum=pointNum+1;
        end;
    end;
end;

points=positionList(2:end,1:end);

%Declare Variables for Tracking Algorithm
maxDistance=3/averageRadius;
param.mem=2;
param.dim=2;
param.good=12;
param.quiet=1;

%Track Particles
tr=track(points, maxDistance, param);

pointList=tr;

end

```

### ***plotTBarDeformation.m***

```
function [] = plotTBarDeformation(track, color, fig)
%Plots deformation based on input list of points
%Plots best line of points for each particle
%Color is a string for output color of plot

figure(fig)
hold on;

idPrevious=track(1,4);
xc=track(1,1);
yc=track(1,2);
buffer=[xc yc];
%Buffer is used to track path of specific particles before plotting

numPoints =length(track(:,1));
%Tracks Particles Assuming tr is sorted based on point id
for k=2:numPoints
    idCurrent=track(k,4);
    if idPrevious==idCurrent
        bufferSize=length(buffer(:,1));
        xc=track(k,1);
        yc=track(k,2);
        buffer(bufferSize+1,1:2)=[xc yc];
    else
        x=buffer(:,1);
        y=buffer(:,2);
        plot(x, y, 'r-', 'LineWidth', 1, 'Color', color);
        if length(x)>1
            plot(x(1),y(1), '.', 'LineWidth',1, 'Color', color);
        end;
        xc=track(k,1);
        yc=track(k,2);
        buffer=[xc yc];
        idPrevious=idCurrent;
    end;
end;

end

end
```

## APPENDIX C – BALL PENETROMETER

The following appendix contains all of the relevant Matlab code that was used to implement the algorithm and produce plots for the ball penetrometer tests. Not included within this appendix is the script file used to run all functions, due to its specificity to file name and location as well as video quality, size, etc. Enough discussion and comments are provided within the functions and the appendix to reproduce results as desired.

### Contents of Appendix

**exportBall.m:** The function reads in a video file, exports gray scale frames, applies filters, and determines the location of the ball within each frame. This is the first function that needs to be run for ball tracking.

**findBallLocation.m:** The function takes in an image input and determines the vertical location of the ball penetrometer within the frame. The function is called within the exportBall function.

**trackBall.m:** Using the open source tracking algorithm, the function will return a list of particles and their locations in space and time.

**plotBall.m:** This function will take the trackBall data, convert it to linear vectors, and separate displacement based on location of the ball in each frame for visualization.

## ***exportBall.m***

```
function [ballLocation, avgRadius] = exportBall( inputFile, ...
    output_location, times, cropDim, edgeDetect, rsbp, speed, trackVar)
%This function takes in videos, extracts only the red band, crops, and
%exports the video to *.tiff images that are run through edge detection and
%a real space bandpass filter. The function exports the frames and returns
%the location of the ball for every exported frame
%
%
% Syntax:  [] = exportBall(ParaIn,ParaIn2,..)
%
% Input parameters:
%   inputFile      - String showing where the video file is located
%   output_location - String indicating where the frames should be
%                     exported
%   times          - array of two times indicating start and
%                     end times for the video
%   cropDim        - array of decimals that indicate the amount to
%                     crop the left, right, top, and bottom of the
%                     video frame. Must be in that specific order.
%   edgeDetect     - intensity value to be used in the sobel method
%   rsbp           - vector of three values that are used to within
%                     real-space band pass method
%   speed          - Value used to determine how often frames should
%                     be exported. Variable denotes that every
%                     nth frame should be exported
%   trackVar       - user inputted variables for tracking calibration
%
% Output parameters:
%   ballLocation   - Array of x and y locations for each frame of the
%                     interpreted location of the ball
%   avgRadius      - Single Integer corresponding to number of pixels
%                     on average the ball is in radius
%
% Other m-files required: CropImage.m, findBallLocation.m
% Subfunctions: none
% MAT-files required: none
%
% See also:
%
% Project:  Ball Penetrometer Tracking
% Authors:  Chris Chini
% History:  07.13.2014  file created
%           full description at the top
%           XX.XX.2014  future iterations with change description

%% Read Video and determine main evaluation frame

%Reads Video File
video=VideoReader(inputFile);

%Define Start and End Frames for Analysis
startT=times(1);
endT=times(2);
```

```

if startT>endT
    error('Something is wrong with your times');
end;

startFrame=startT*video.FrameRate;
endFrame=endT*video.FrameRate;

%Defines the number of Frames for Output
numFrames=endFrame-startFrame+1;
numFrameOutput=round(numFrames/speed)-1;

%% Display three frames and get user input for x location and radius of ball

%Get three frames
index1=round(numFrameOutput/3+startFrame);
frame1=read(video, index1);
cropped=CropImage(frame1, cropDim(1), cropDim(2), cropDim(3), cropDim(4));
red1=cropped(:,:,1);

index2=round(numFrameOutput*2/3+startFrame);
frame2=read(video, index2);
cropped=CropImage(frame2, cropDim(1), cropDim(2), cropDim(3), cropDim(4));
red2=cropped(:,:,1);

frame3=read(video, endFrame);
cropped=CropImage(frame3, cropDim(1), cropDim(2), cropDim(3), cropDim(4));
red3=cropped(:,:,1);

%Display each frame and ask for user input to get ball information
figure(5)
hold on
imagesc(red1);
colormap(gray);
axis equal;
title('Pick three points on the edge of the ball');

[point1,point2, point3]=rbtriangle();

[center1, rad1]=calcCircle(point1, point2, point3);

close(5);

figure(5)
hold on
imagesc(red2);
colormap(gray);
axis equal;
title('Pick three points on the edge of the ball');

[point1,point2, point3]=rbtriangle();

[center2, rad2]=calcCircle(point1, point2, point3);

```

```

close(5);

figure(5)
hold on
imagesc(red3);
colormap(gray);
axis equal;
title('Pick three points on the edge of the ball');

[point1,point2, point3]=rbtriangle();

[center3, rad3]=calcCircle(point1, point2, point3);

close(5);

%From the three circles, find the average radius and the average x location
%of all interpolations to define as overall averages

ballLocation=zeros(numFrameOutput,2);

avgX=(center3(1)+center2(1)+center1(1))/3;
avgRadius=(rad1+rad2+rad3)/3;

ballLocation(:,1)=avgX;

%% Declare output location and export frames
%Declares output Location for Frames forFiles
outLocate=strcat(output_location, '_frames/FRAME');

%Export Frames
for i=1:numFrameOutput

    figure(1)
    hold on;

    %Read in vide frame, crop image, and extract red band
    frame=read(video, i*speed+startFrame);
    cropped=CropImage(frame, cropDim(1), cropDim(2), cropDim(3), cropDim(4));
    red=cropped(:,:,1);

    %determine location of ball
    ballLocation(i,2)=findBallLocation(red, trackVar(1), trackVar(2))-
    avgRadius;

    %run edge detection algorithm on grayscale image
    edgeImage=edge(red, 'sobel', edgeDetect);
    imwrite(edgeImage, 'temp1.tiff', 'TIFF'); %create temp image file
    imagel=imread('temp1.tiff'); %read in temp input file
    b=bpss(imagel(:,:,1),rsbp(1),rsbp(2));%run real space bandpass filter
    b=b*rsbp(3);%increase contrast
    file=strcat(outLocate, num2str(i), '.tiff');
    imwrite(b, file, 'TIFF');%write exported frame to file

```

```
%Show image and determined circle for visual check on algorithm
imagesc(red);
colormap(gray);
circle(avgX, ballLocation(i,2),avgRadius);
hold off;

end;

end
```

## ***findBallLocation.m***

```
function [yLocation] = findBallLocation( image , shadowValue, verticalOffset)
%This function takes in an image and a scale factor and determines where
%the bottom of the ball is for the frame. The function is to be called
%from exportBall.m
%
%
% Syntax:  [] = findBallLocation(ParaIn,ParaIn2,..)
%
% Input parameters:
%   image           - a cropped image that has already had edge
%                     detection run on it
%   shadowValue      - cutoff value for determining edge of shadow in
%                     image
%   verticalOffset   - the observed offset value (approx) for ball
%                     penetrometer
%
% Output parameters:
%   yLocation        - Determined y location of the bottom of the ball
%
% Other m-files required: none
% Subfunctions: none
% MAT-files required: none
%
% See also: exportBall.m
%
% Project:  Cone Tracking
% Authors:  Chris Chini
% History:  07.13.2014  file created
%           full description at the top
%           XX.XX.2014  updates

[row, col]=size(image);

frame=image;

%Convert Image to two-tone image with input as an arbitrary index value
for r=1:row
    for c=1:col
        if image(r,c)>shadowValue
            frame(r,c)=255;
        else
            frame(r,c)=0;
        end;
    end;
end;

maxValue=255*col;

%Find row closest to the bottom with the lowest row sum, which will
%correlate to the bottom of the ball based on shadowing.
```

```

for r=row:-1:1
    sumRow=sum(frame(r,:));
    if sumRow<0.85*maxValue
        break;
    end;
end;

yLocation=r+verticalOffset;%Add Observed Calibration Factor

end

```

## ***trackBall.m***

```
function [ trackedParticles ] = trackBall(file_location, ballCoordinates, ...
    percentMax, particleSize, maxDistance, memory, good)
%% Tracks Particles based on the exported images in a specified file set
% The function will read in the number of files in the specified file
% location for both the peak and residual exported frames. It will return a
% point list with time excluding all points that fall within the cone (or
% behind the cone)
%
%
% Syntax: [trackedParticles] = trackBall(file_location, ...)
%
% Input parameters:
%   file_location      - location of files
%   ballCoordinates    - coordinates of the ball for every frame
%   percentMax         - percent of the maximum intensity that will be
%                       considered a particle
%   particleSize       - minimim pixel size of particle to be analyzed
%   maxDistance        - maximum distance a particle travels between frames
%   memory             - number of frames a particle can disappear before
%                       reappearing and being considered the same particle
%   good               - variable that determines the amount of play in
%                       particle movement
%
%
% Output parameters:
%   pointList          - a list of all the points that were found in the set of
%                       images. The first column equates to the x location in pixels, the
%                       second column equates to the y location in terms of pixels. The third
%                       column references the frame number. The fourth and final column
%                       represents the ID of the particle
%
%
% Other m-files required: pkfnd.m, track.m
% Subfunctions: none
% MAT-files required: none
%
% See also: exportBall.m, plotBall.m
%
% Project:   Ball Penetrometer Tracking
% Authors:   Chris Chini
% History:   07.13.2014   file created
%                               full description at the top
%               XX.XX.2014   future iterations with change description

%% Track Particles using open source algorithm
%Determine number of frames to be analyzed
numFrames=length(ballCoordinates(:,1));

%Cycle through all frames
for index=1:numFrames
    frameName=strcat(file_location, '_frames/FRAME', num2str(index),
        '.tiff');
```

```

image=imread(frameName);

%Find all peaks in the image
maxVal=max(max(image));
threshold=percentMax*maxVal;
peak=pkfind(image, threshold, particleSize);
peak(:,3)=index;

if index==1
    points=peak;
else
    ex=length(points(:,1));
    new=length(peak(:,1));
    points(ex+1:ex+new,:)=peak;
end;

end;

%Declare Variables for Tracking Algorithm
param.mem=memory;
param.dim=2;
param.good=good;
param.quiet=1;

%Track Particles
tr=track(points, maxDistance, param);

trackedParticles=tr;

end

```

## ***plotBall.m***

```
function [vectors] = plotBall( track, ballLocation, step )
%This function takes in the tracking data table and plots it
%
%
% Syntax:  [] = plotBall(ParaIn,ParaIn2,...)
%
% Input parameters:
%   track          - list of points in time and space along with IDs
%                   of points that correspond to the tracking
%                   algorithm completed in trackBall.m
%   ballLocation    - list of ball locations normalized in radians and
%                   normalized by the top of the sample
%   step            - step length for plotting in radians
%
% Output parameters: none
%
% Other m-files required: none
% Subfunctions: none
% MAT-files required: none
%
% See also: trackBall.m
%
% Project:  Ball Penetrometer Tracking
% Authors:  Chris Chini, Jeff Wallace, Cassandra Rutherford, Joshua Peschel
% History:  07.13.2014  file created
%           full description at the top
%           XX.XX.2014  future iterations with change description

hold on;

idPrevious=track(1,4);
currentTime=track(1,3);
previousPlot=getPlotNumber(ballLocation(currentTime,2),step);
xc=track(1,1);
yc=track(1,2);
buffer=[xc yc];
%Buffer is used to track path of specific particles before plotting

numPoints =length(track(:,1));
%Tracks Particles Assuming tr is sorted based on point id

lineSegments=zeros(1,5);
lineCount=1;

for k=2:numPoints
    idCurrent=track(k,4);
    currentTime=track(k,3);
    currentPlot=getPlotNumber(ballLocation(currentTime,2),step);
    bufferSize=length(buffer(:,1));
    if idPrevious==idCurrent && previousPlot==currentPlot && bufferSize <20
        xc=track(k,1);
```

```

        yc=track(k,2);
        buffer(bufferSize+1,1:2)=[xc yc];
    else
        x=buffer(:,1);
        y=buffer(:,2);

        x0=x(1);
        y0=y(1);
        xf=x(end);
        yf=y(end);

        disp=sqrt((x0-xf)^2+(y0-yf)^2);

        if disp>0.07
            [x1, y1, x2, y2]=plotBallFitted(x,y,previousPlot);

            lineSegments(lineCount,:)=[x1, y1, x2, y2, previousPlot];
            lineCount=lineCount+1;
        end;

        xc=track(k,1);
        yc=track(k,2);
        buffer=[xc yc];
        idPrevious=idCurrent;
        previousPlot=currentPlot;
    end;
end;

vectors=lineSegments;

end

```

## APPENDIX D – CONE PENETROMETER

The following appendix contains all of the relevant Matlab code that was used to implement the algorithm and produce plots for the cone penetrometer tests. Not included within this appendix is the script file used to run all functions, due to its specificity to file name and location as well as video quality, size, etc. Enough discussion and comments are provided within the functions and this appendix to reproduce results as desired.

### Contents of Appendix

**exportCone.m:** This is the initial function that takes in raw video with other parameters and exports the required frames. The function also returns the location of the cone for each frame to be utilized later.

**findConeTip.m:** The function is called within the exportCone function and will determine the vertical location of the cone within a given frame using intensity contrasts of the image.

**trackCone.m:** Using the open source tracking algorithm, the function will return a list of particles and their locations in space and time.

**plotCone.m:** This function will take the trackCone data, convert it to linear vectors, and separate displacement based on location of the cone in each frame for visualization.

**plotConeFitted.m:** The function is called within the plotCone function and determines a best fit linear approximation of motion for an individually tracked particle.

## ***exportCone.m***

```
function [coneCoordinates, scale] = exportCone( inputFile, ...
    output_location, times, cropDim, edgeDetect, rsbp, speed, trackVar)
%The function takes in videos, extracts only the red band, crops, and
%exports the video to *.tiff images that are run through edge detection and
%a real space bandpass filter. The function exports the frames and returns
%the location of the cone for every exported frame
%
%
% Syntax:  [] = exportCone(ParaIn,ParaIn2,..)
%
% Input parameters:
%   inputFile       - String showing where the video file is located
%   output_location - String indicating where the frames should be
%                       exported
%   times           - array of three times indicating start, peak, and
%                       end times for the video
%   cropDim         - array of decimals that indicate the amount to
%                       crop the left, right, top, and bottom of the
%                       video frame. Must be in that specific order.
%   edgeDetect      - intensity value to be used in the sobel method
%   rsbp            - vector of three values that are used to within
%                       real-space band pass method
%   speed           - Value used to determine how often frames should
%                       be exported. Variable denotes that every
%                       nth frame should be exported
%   trackVar        - Sequence of three variables that govern the
%                       tracking of the cone. First variable is the cut
%                       off value for shadow finding. The second variable
%                       is the adjustment factor for vertical offset in
%                       the cone tip. The final factor is the offset
%                       horizontally for the vertical face of the cone.
%
% Output parameters:
%   coneCoordinates - xy coordinate of cone tip and outer edge of
%                       angled cone line. tx4 matrix where first two
%                       columns correspond to the cone tip
%                       the last two columns correspond to the outside
%                       edge of the cone
%
% Other m-files required: CropImage.m, findConeTip.m
% Subfunctions: none
% MAT-files required: none
%
% See also:
%
% Project:   Cone Tracking
% Authors:   Chris Chini
% History:   06.24.2014  file created
%             full description at the top
%             07.09.2014  updated algorithm to export all edges of cone

%% Read Video and determine main evaluation frame
```

```

%Reads Video File
video=VideoReader(inputFile);

%Define Start and End Frames for Analysis
startT=times(1);
endT=times(2);

if startT>endT
    error('Something is wrong with your times');
end;

startFrame=startT*video.FrameRate;
endFrame=endT*video.FrameRate;

%Defines the number of Frames for Output from Start to Peak
numFrames=endFrame-startFrame+1;
numFrameOutput=round(numFrames/speed)-1;

frame=read(video, endFrame);
cropped=CropImage(frame, cropDim(1), cropDim(2), cropDim(3), cropDim(4));
red=cropped(:, :, 1);
finalEdge=edge(red, 'sobel', edgeDetect);

%% Find the right edge of the cone based on the final edgeImage
%The outside edge of the cone can be found using hough lines to determine
%the length of the longest line in the image. The longest line should be
%the vertical interface of the cone and laponite

%First run hough algorithm

[H,T,R] = hough(finalEdge);

P = houghpeaks(H,5,'threshold',ceil(0.2*max(H(:)))));

% Find lines for peak condition
lines = houghlines(finalEdge,T,R,P,'FillGap',5,'MinLength',20);

max_len = 0;
for k = 1:length(lines)
    xy = [lines(k).point1; lines(k).point2];

    % Determine the endpoints of the longest line segment
    len = norm(lines(k).point1 - lines(k).point2);
    if ( len > max_len)
        max_len = len;
        xy_long = xy;
    end
end

x1=xy_long(1,1);

x2=xy_long(2,1);

```

```

verticalInterface=round((x1+x2)/2-trackVar(3));%Add Observed Calibration
factor

%% Export frames and find location of cone tip

figure(5)
hold on
imagesc(red);
colormap(gray);
axis equal;
title('Draw Line from cone tip to outer edge for scale');

[point1,point2]=rbline();

x1=point1(1,1);
y1=point1(1,2);
x2=point2(1,1);
y2=point2(1,2);

close(5);

%This is the distance from the tip of the cone along the slanted edge to
%the edge of the rod. Correlates to 0.75 inches
disp=sqrt((x1-x2)^2+(y1-y2)^2);
%The distance also correlates to the diameter of the cone
%The scale factor will correlate to the radius of the cone, so 1/2*diam
scale=disp/2;

%Declares output Location for Frames forFiles
outLocate=strcat(output_location, '_frames/FRAME');

%Store tip location for each frame
edgeCoordinates=zeros(numFrameOutput,4);

angle=degtorad(60);

edgeCoordinates(:,3)=verticalInterface;
edgeCoordinates(:,1)=round(verticalInterface-disp*cos(angle));

%Export Frames
for i=1:numFrameOutput

    figure(1)
    hold on;
    title(inputFile);

    frame=read(video, i*speed+startFrame);%read in frame
    %crop frame and extract red band
    cropped=CropImage(frame, cropDim(1), cropDim(2), cropDim(3), cropDim(4));
    red=cropped(:,:,1);

    edgeImage=edge(red, 'sobel', edgeDetect); %edge detect image.

```

```

imwrite(edgeImage, 'temp1.tiff', 'TIFF');%write temp file
image1=imread('temp1.tiff');%read in temp file
b=bpss(image1(:,:,1),rsbp(1),rsbp(2)); %run real space filter
b=b*rsbp(3); %increase contrast
file=strcat(outLocate, num2str(i), '.tiff');
imwrite(b, file, 'TIFF');%export file

[yTip]=findConeTip(red, trackVar(1), trackVar(2)); %find tip of cone
edgeCoordinates(i,2)=yTip;

%plot tracked cone tip for visual confirmation of accuracy
imagesc(red);
colormap(gray);
plot([verticalInterface-disp*cos(angle), verticalInterface], ...
     [yTip, yTip-disp*sin(angle)], 'red');
end;

edgeCoordinates(:,4)=round(edgeCoordinates(:,2)-disp*sin(angle));

coneCoordinates=edgeCoordinates;

close(1);

end

```

## ***findConeTip.m***

```
function [yTip] = findConeTip( image , shadowValue, verticalOffset)
%This function takes in an image and a scale factor and determines where
%the tip of the cone is for the frame. The function is to be called
%from exportCone.m
%
%
% Syntax:  [] = exportMV(ParaIn,ParaIn2,...)
%
% Input parameters:
%   image           - a cropped image that has already had edge
%                     detection run on it
%   shadowValue      - cutoff value for determining edge of shadow in
%                     image
%   verticalOffset    - the observed offset value (approx) for vertical
%                     cone tip
%
% Output parameters:
%   yTip             - Determined y location of the cone tip
%
% Other m-files required: none
% Subfunctions: none
% MAT-files required: none
%
% See also: exportCone.m
%
% Project:  Cone Tracking
% Authors:  Chris Chini
% History:  06.24.2014  file created
%           07.09.2014  full description at the top
%           07.09.2014  updated tracking algorithm for cone tip
%           07.10.2014  updated input values for function

[row, col]=size(image);

frame=image;

%Convert Image to two-tone image with 30 as an arbitrary index value
for r=1:row
    for c=1:col
        if image(r,c)>shadowValue
            frame(r,c)=255;
        else
            frame(r,c)=0;
        end;
    end;
end;

maxValue=255*col;

%Find row closest to the bottom with the lowest row sum, which will
%correlate to the tip of the cone based on shadowing.
```

```

for r=row:-1:1
    sumRow=sum(frame(r,:));
    if sumRow<0.85*maxValue
        break;
    end;
end;

yTip=r+verticalOffset;%Add Observed Calibration Factor

end

```

## ***trackCone.m***

```
function [ trackedParticles ] = trackCone(file_location, coneCoordinates, ...
    percentMax, particleSize, maxDistance, memory, good)
%% Tracks Particles based on the exported images in a specified file set
% The function will read in the number of files in the specified file
% location for both the peak and residual exported frames. It will return a
% point list with time excluding all points that fall within the cone (or
% behind the cone)
%
%
% Syntax: [trackedParticles] = trackMV(file_location)
%
% Input parameters:
%   file_location      - location of files
%   coneCoordinates    - coordinates of the cone for every frame
%   percentMax         - percent of the maximum intensity that will be
%                       considered a particle
%   particleSize       - minimim pixel size of particle to be analyzed
%   maxDistance        - maximum distance a particle travels between frames
%   memory             - number of frames a particle can disappear before
%                       reappearing and being considered the same particle
%   good               - variable that determines the amount of play in
%                       particle movement
%
%
% Output parameters:
%   pointList          - a list of all the points that were found in the set of
%                       images. The first column equates to the x location in pixels, the
%                       second column equates to the y location in terms of pixels. The third
%                       column references the frame number. The fourth and final column
%                       represents the ID of the particle
%
%
% Other m-files required: pkfnd.m, track.m
% Subfunctions: none
% MAT-files required: none
%
% See also: exportMV.m, plotMV.m
%
% Project:   Cone Tracking
% Authors:   Chris Chini
% History:   07.09.2014   file created
%                               full description at the top
%               XX.XX.2014   future iterations with change description

%% Track Particles using open source algorithm
%Determine number of frames to be analyzed
numFrames=length(coneCoordinates(:,1));

%Cycle through all frames
for index=1:numFrames
    frameName=strcat(file_location, '_frames/FRAME', num2str(index),
        '.tiff');
```

```

image=imread(frameName);

%Eliminate all unacceptable points. Unacceptable points include points
%that are the illuminated edge of the cone or in the shadowed area
%The regions are determined based on the cone coordinates in each
%frame
xTip=coneCoordinates(index, 1);
yTip=coneCoordinates(index, 2);
xEdge=coneCoordinates(index, 3);
yEdge=coneCoordinates(index, 4);

image(1:yTip, 1:xTip)=0;
image(1:yEdge, xTip:xEdge)=0;

X=[xTip, xEdge, xTip, xTip];
Y=[yTip, yEdge, yEdge, yTip];

for row=yEdge:yTip
    for col=xTip:xEdge
        if inpolygon(col, row, X, Y)==1
            image(row, col)=0;
        end;
    end;
end;

%Now find all peaks in the leftover image
maxVal=max(max(image));
threshold=percentMax*maxVal;
peak=pkfind(image, threshold, particleSize);
peak(:,3)=index;

if index==1
    points=peak;
else
    ex=length(points(:,1));
    new=length(peak(:,1));
    points(ex+1:ex+new,:)=peak;
end;

end;

%Declare Variables for Tracking Algorithm
param.mem=memory;
param.dim=2;
param.good=good;
param.quiet=1;

%Track Particles
tr=track(points, maxDistance, param);

trackedParticles=tr;

end

```

## ***plotCone.m***

```
function [vectors] = plotCone( track, coneLocation, scale, step )
%This function takes in the tracking data table and plots it
%
%
% Syntax:  [] = plotCone(ParaIn,ParaIn2,..)
%
% Input parameters:
%   track          - list of points in time and space along with IDs
%                   of points that correspond to the tracking
%                   algorithm completed in trackCone.m
%   coneLocation    - list of vertical cone locations for each time
%                   step
%   scale           - pixel to radii scale for test
%   step            - length in radii of the displacement step for
%                   visualization on each plot
%
% Output parameters:
%   vectors         - list of vectors for each plot in terms of start
%                   and end coordinates per particle (x, y)
%
% Other m-files required: plotConeFitted.m
% Subfunctions: none
% MAT-files required: none
%
% See also: trackCone.m
%
% Project:  Cone Tracking
% Authors:  Chris Chini
% History:  07.09.2014  file created
%           full description at the top
%           XX.XX.2014  future iterations with change description

hold on;

minYCone=sqrt(3);

tipYLocations=(coneLocation(:,2)-coneLocation(1,2))/scale;

idPrevious=track(1,4);
currentTime=track(1,3);
previousPlot=getPlotNumber(tipYLocations(currentTime),step);
xc=track(1,1);
yc=track(1,2);
buffer=[xc yc];
%Buffer is used to track path of specific particles before plotting

numPoints =length(track(:,1));
%Tracks Particles Assuming tr is sorted based on point id

lineSegments=zeros(1,5);
lineCount=1;
```

```

for k=2:numPoints
    idCurrent=track(k,4);
    currentTime=track(k,3);
    currentPlot=getPlotNumber(tipYLocations(currentTime),step);
    %rely on a buffer array to read in points for each particle
    bufferSize=length(buffer(:,1));
    %add values to buffer if they are the same particle, will lie on the
    %same plot and does not exceed maximum buffer size, for visualization
    if idPrevious==idCurrent && previousPlot==currentPlot && bufferSize <20
        xc=track(k,1);
        yc=track(k,2);
        buffer(bufferSize+1,1:2)=[xc yc];
    else
        x=buffer(:,1);
        y=buffer(:,2);

        %find start and end coordinates of particle
        x0=x(1);
        y0=y(1);
        xf=x(end);
        yf=y(end);

        %determine length of line
        disp=sqrt((x0-xf)^2+(y0-yf)^2);

        %line length must be greater than 0.1 to be plotted
        if disp>0.1
            %plot linear vector
            [x1, y1, x2, y2]=plotConeFitted(x,y,previousPlot);

            lineSegments(lineCount,:)=[x1, y1, x2, y2, previousPlot];
            lineCount=lineCount+1;
        end;

        xc=track(k,1);
        yc=track(k,2);
        buffer=[xc yc];
        idPrevious=idCurrent;
        previousPlot=currentPlot;
    end;
end;

vectors=lineSegments;

end

```

### ***plotConeFitted.m***

```
function [x1,y1,x2,y2] = plotConeFitted(x,y, fig)
%Plots a scaled arrow from the start of deformation to the end of the
%deformation.

%Scale is internally changed
scale=2;

%find linear fit for data
p=polyfit(x,y,1);

startx=x(1);

%draw line between xmin and xmax at given slope, p
low=min(x);
high=max(x);

dif=high-low;
step=dif/5;

difLow=startx-low;
difHigh=high-startx;

if difLow<=difHigh
    high=low+dif*scale;
    xp=low:step:(high);
else
    low=high-dif*scale;
    xp=high:-step:(low);
end;

yp=p(1)*xp+p(2);

figure(fig);

if length(xp)>1 %Ensure that there is more than one point in vector
    plot(xp, yp, 'LineWidth', 1, 'Color', 'black');

    plot(xp(1),yp(1),'.','LineWidth', 1, 'Color', 'black');

    x1=xp(1);
    y1=yp(1);
    x2=xp(end);
    y2=yp(end);
else
    x1=NaN;
    y1=NaN;
    x2=NaN;
    y2=NaN;
end;

end
```