Black Dots: Reference-Free Traction Force Microscopy for Measuring Single-Cell Forces

Kevin M. Beussman

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2022

Reading Committee:
Nathan J. Sniadecki, Chair
Ashley Emery
Juan Carlos del Álamo

Program Authorized to Offer Degree:
Mechanical Engineering

University of Washington

Abstract

Black Dots: Reference-Free Traction Force Microscopy for Measuring Single-Cell Forces

Kevin M. Beussman

Chair of the Supervisory Committee:
Professor Nathan J. Sniadecki
Mechanical Engineering

Cells generate cytoskeletal forces for a many purposes including cell locomotion and tissue functions like heart beating and wound healing. The amount of force produced by a cell depends on many things including its cell type, size, and environment. Measuring the forces produced by cells can provide insight into their behavior and physiological function, or can be used to study the effect of drugs and disease. To isolate their behavior, forces are measured in single cells, typically on a 2D surface for simplicity, yielding "traction forces" as a cell pulls parallel to the 2D plane. This principle has been used to develop techniques such as membrane wrinkling, microposts, and traction force microscopy among many others. However, existing traction force methods have several drawbacks including the limited number of cells that can be measured per experiment or inadvertent impact on cell functions by strictly constraining cell size and shape.

In this work, a novel reference-free traction force microscopy technique is developed to overcome limitations from existing methods. The technique, named "black dots", microcontact prints a fluorescent micropattern onto a flexible substrate to measure cellular traction forces without constraining cell shape or needing to detach the cells. **Chapter 2** describes the theory and development behind the black dots approach. Several techniques are combined including microcontact printing and sacrificial films to deposit a fluorescent pattern on very soft PDMS with high fidelity. The pattern is characterized and an example data set is

generated to study the sensitivity of the black dots approach. In **Chapter 3** the technique is demonstrated by assessing forces in human platelets, which are the smallest cells in the human body and can generate very large forces for their size. Because the black dots approach is reference-free, fixation and immunofluorescent staining can be applied along with geometric measurements to accompany the traction force data. In this study, platelets that exert more force tend to have more spread area, are more circular, and have more uniformly distributed F-actin filaments. As a result of the high yield of data obtainable by the black dots approach, a clustering analysis and a multivariate mixed effects model with interaction terms can be used to identify clusters and trends between force, area, circularity, and F-actin dispersion. Finally, **Chapter 4** explores the use of black dots in measuring forces in live, dynamically beating cardiomyocytes, including a protocol and special considerations for using black dots with cardiomyocytes to ultimately measure both isometric and twitch forces over time. These forces as well as the twitch dynamics are analyzed for a small set of example cells that showcase the potential for new discoveries enabled black dots.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would first like to thank everyone who contributed to black dots over the many years of development: Andrea Leonard for sparking the initial ideas and prototypes, Molly Mollica for helping develop the final protocols, Robin Yan and Kenia Diaz for troubleshooting the method, Zizhen Song for contributing to the analysis code, and Priti Mulimani and Deema Alroweilly for learning to use black dots providing constructive feedback. You were all vital for the success of this project and this project would not have been possible without you all.

I would like to thank my committee chair and PI Nathan Sniadecki, whose guidance made me who I am today and pushed me to always learn new things and venture out of my comfort zone. I will always be grateful for the wide variety of experiences in your lab that ultimately allowed me to find my true passion. And for the rest of my committee, Ashley Emery, Jen Davis, Wendy Thomas, and Juan Carlos del Álamo, thank you for your advice and your help over the years to lead me to successes and bringing me back from failures.

To all the members of the Cell Biomechanics Lab past and present, thank you for providing a wonderfully inviting community to foster great research while maintaining a fun, casual environment. I am proud to have been a member of this group and my experiences here will never be forgotten.

To all of the other graduate students I have interacted with, to all the undergraduate researchers who often don't get enough credit, and to all the students that I tought and TAed, thank you for making these years so profoundly engaging.

And finally, to my family for whom I owe everything. My parents, for always fostering my love of science and math. My late brother who I looked up to and who I know would be proud. My wife for always pushing me to pursue greater things, and for sticking with

me this long. And my children, for giving me so much joy and brightening even the darkest days. Thank you with all my heart.

DEDICATION

To my wife, Xue, and my children, Leona and Elias.

Chapter 1
INTRODUCTION

1.1 Cells generate useful traction forces

Cells use forces to locomote, contract, and probe their environment [30, 104]. To generate these forces, cells use interactions between actin and myosin to create cytoskeletal tension. These tensile cytoskeletal forces are then transmitted through adhesions to the extracellular matrix, allowing cells to interact with their surroundings dynamically. While the structure and protein isoforms differ between cell types, the mechanisms of force generation and transmission remain largely the same.

Different types of cells use force in different ways. Platelets, for example, generate large contractile forces in order to seal wounds. On the other hand, cardiomyocytes contract rhythmically to collectively push blood through the body. Other cells may instead use their traction forces to propel themselves forward to migrate or hunt for contaminants. Regardless of their use, measuring these forces becomes important to understand how cells develop and behave.

1.2 Measuring cell traction forces

Cell forces can change as cells develop, in response to therapeutic drugs, or as cells become diseased. Measuring these traction forces is therefore necessary to fully understand cell development, and can also be useful for testing new therapies for diseases. However, quantification of cellular forces can be challenging because force production varies by cell type, size, and shape, and because cellular forces can be inadvertently affected by environmental factors such as substrate stiffness, substrate topography, or extracellular matrix (ECM) type [35, 61, 77, 81, 83].

When cells produce force, they deform their environmental substrate, allowing for force calculation from the substrate deformation (more deformation indicates more cell force). Substrate deformation can be ultimately be quantified by comparing the undeformed, zero-displacement reference state of the substrate to the deformed, cell-displaced state. The traction forces are then inferred from these measured deformations. This principle has been used to develop techniques such as membrane wrinkling [40], traction force microscopy [23, 51, 86], and microposts [8, 88, 90], among many others to measure single-cell forces [68, 74, 78, 80]. Of these approaches, traction force microscopy (TFM) has been one of the most widely adopted for measuring single-cell forces. Traction force microscopy (TFM) was one of the first techniques used to quantify cell forces, where fiducial beads are randomly embedded in a flexible substrate and used to measure substrate deformation. TFM has been thoroughly developed by many authors and serves as the basis for this work.

1.3 A brief history of traction force microscopy

The earliest demonstrations of traction force measurements by Harris *et al.* utilized wrinkling membranes to detect substrate deformation [40]. As the cells pulled on their environment, wrinkles formed in the soft silicone rubber substrate which were visible using a microscope. These ruffles were a bonafide demonstration and proof of cells pulling and probing their environment. While pioneering for the field of force microscopy, wrinkling membranes were limited due to difficulties in quantifying the magnitude and direction of the forces. Lee *et al.* built upon this technique by introducing latex beads to the surface of the silicone substrate, allowing for direct measurement of the surface deflections due to migrating keratocytes [51]. These crude measurements led to theories about how these traction forces may be distributed throughout the cell, and how these force distributions might change during cell function.

Dembo and Wang improved upon the method by using polyacrylamide sheets with fluorescent latex beads distributed throughout [23]. The larger number of beads allowed for subcellular measurements of displacement underneath 3T3 cells. In addition, Dembo and Wang used advanced mathematics including the Boussinesq equations, using entropy max-

imization and global force and torque balances to accurately estimate the traction forces from the displacement image. The improved analysis and force resolution allowed for direct observation of the force distribution throughout the cell, highlighting the large forces at lamellopodia and the force disparities between anterior and posterior regions.

Recent traction force microscopy (TFM) techniques rely heavily on the work from Dembo and Wang, making incremental adjustments and optimizations to help improve traction force measurements. Franck *et al.* used digital volume correlation and confocal imaging to obtain traction force measurements in 3D, allowing for both in-plane and out-of-plane traction force measurements [31, 67]. Del Álamo *et al.* accounted for the real thickness of the substrate instead of assuming an infinitely thick material, showing that traction forces were likely underestimated using previous methods[21]. Sabass *et al.* incorporated the use of two differently colored nanobeads to greatly improve the resolution of the traction force measurements and offered improvements to the different analysis methods, comparing boundary element method (BEM), Fourier transform traction cytometry (FTTC), and traction reconstruction with point forces (TRPF) [84]. Other improvements to the standard TFM approach include polymerizing the gels upside-down to promote fiducial marker accumulation in-plane at the surface [37], or introducing advanced denoising and Bayesian solution methods [45].

1.4 Reference-free TFM

In TFM, the deformation of the beads is typically determined by first imaging the cell-displaced state of the beads, and then removing the cells to obtain the zero-displacement state. Cell removal is inherently incompatible with further processing such as fixation and immunofluorescent staining. An alternative to this approach is to image the substrate with live microscopy, observing the beads before a cell has adhered. However, this is time consuming and limits the overall data yield. Still, this is incompatible with fixation as it is often too difficult to identify the same fixed cells that were captured by live imaging.

To overcome this primary limitation of TFM, the requirement of a reference image, reference-free methods have been developed. Reference-free approaches aim to deposit the

fiducial markers in a well-known pattern rather than randomly distributed. In this way, the reference state of each fiducial marker can be estimated by extrapolating the position of undeformed markers, usually far away from cells. Because the reference state can be estimated without cell removal or live imaging, reference-free methods are able to measure forces of fixed cells. Several researchers have developed their own reference-free methods built upon this principle. Balaban *et al.* first demonstrated this by embedding fluorescent photoresist into a flexible substrate [1]. Others have used microcontact printing to deposit fluorescent ECM onto flexible substrates in a grid pattern [15, 33, 75]. Bergert and Lendenmann *et al.* developed Cellogram, a tool that utilizes an ink-jet like printing process to deposit clusters of quantum dots in a grid [5, 52]. Two-photon laser scanning lithography can be used to create a 3D array of fiducial markers within a flexible substrate, facilitating traction force measurements even deep within the substrate [2]. Others have taken a different approach, instead developing larger patterns to measure whole-cell forces [66, 76].

While these techniques offer great improvements to TFM and are compatible with fixation, they are often difficult to manufacture, are low-yield, or adversely affect cell spreading. As such, these reference-free methods have not been widely adopted for quantifying cell forces.

1.5 Black dots: a new method to fill a gap in single-cell techniques

Existing 2D single-cell force techniques have several drawbacks including the limited number of cells that can be measured per experiment or inadvertent impact on cell functions by strictly constraining cell size and shape. To overcome these drawbacks, the black dots method was invented. Rather than deposit fluorescent ECM in a grid pattern, which constrains cell spreading, the black dots method decouples the fluorescent grid from the adhesive ECM.

The black dots technique offers several advantages over existing TFM and other reference-free methods for measuring single-cell forces: 1) it is high-yield due to the ability to measure force with a single image, 2) it is compatible with immunofluorescent staining so that traction forces can be measured alongside analysis of structure and/or protein localization, and 3) it

does not constrain cell shape and size due to the substrate containing a contiguous adhesive protein. This dissertation will describe in detail the development of the black dots approach and its use in measuring single-cell forces.

Chapter 2

DEVELOPMENT OF THE BLACK DOTS APPROACH: MANUFACTURING, THEORY, AND ANALYSIS

The black dots platform is developed as a 2D single-cell force measurement approach that is reference-free, capable of measuring forces in both fixed and stained samples. By using fixed samples, forces can be easily combined with immunofluorescent staining of important proteins, allowing for direct correlation between structure and force. Here, I describe the development of the black dots approach, including the manufacturing procedure, the mathematical theory, and the image analysis required to estimate forces.

2.1 Manufacturing black dots substrates

In order to measure forces with the black dots method, cells are seeded onto a soft PDMS surface that has a pattern of fluorescent bovine serum albumin (BSA) on the surface. The black dots pattern is created using a microcontact-printing approach modified with sacrificial film technology. Typical physiological stiffness of biological materials is around 1-100 kPa [24], which poses problems when using traditional microcontact printing to deposit patterns of protein. Researchers have developed techniques using sacrificial films to transfer the protein gently to the surface [95, 103]. Here, a modified poly(vinyl alcohol) (PVA) film approach is used to transfer protein from a patterned stamp to the soft PDMS substrate. Finally, the pattern is coated with ECM protein before cells are attached.

2.1.1 Microfabrication of patterned stamp

First, a silicon master mold with an array of vertical pillars is created with the desired pattern size by photolithography as described previously [29]. Photoresist is spun onto

a silicon wafer and either an e-beam lithography system or traditional photolithography approaches are used to pattern circles of the desired diameter and center-to-center. The photoresist is then developed and etched to create a master containing an array of vertical silicon pillars. For typical cells, a pattern with diameter of 2-3 μm and center-center spacing of 6-9 μm is sufficient. In the case of traditional photolithography, a chrome mask is placed over the photoresist and light is shone through the mask to cure the photoresist. Appendix A.5 contains example code to automatically generate the geometry files needed for the mask writer, both for black dots as well as micropost designs. The height of the pattern is not vital to the process, but the pillars used here typically have a height of 5 μm. For cells as small as platelets, a pattern with a diameter of 850 nm, center-to-center spacing of 2 μm, and height of 3.5 μm is used. Larger pattern sizes are easier to image but result in a lower spatial resolution of traction forces.

To generate the stamps for patterning the fluorescent protein, PDMS (Sylgard 184, Dow Corning) at a 10:1 base to curing agent ratio is poured onto the master mold and cured in a 110 °C oven for 20 minutes. The cured PDMS is peeled from the master mold, revealing a negative version of the original pattern: a grid of holes instead of pillars. Edges of the stamp are trimmed with sharp razors and stored in enclosed petri dishes prior to use.

The negative PDMS stamp created here may encounter some shrinkage during the proceeding manufacturing steps. A master mold with an actual spacing between dots of 1.99 μm (measured by widefield microscopy, data not shown here) resulted in a final fluorescent pattern of black dots with a mean spacing of 1.96 μm. This shrinkage may need to be accounted for in the design of the master mold if the pattern dimensions are crucial. At the very least, pattern dimensions should be measured from the final flexible substrate instead of relying upon the master mold dimensions when calculating traction forces.

2.1.2 Sacrificial PVA film production

Poly(vinyl alcohol) (PVA) films for transferring the fluorescent pattern are made following previously described protocols with some modifications [57, 103]. A mixture of 0.55 g PVA

powder (Sigma) is mixed with 15 mL DI water and heated for 30 minutes at 110 °C until the powder fully dissolved. The PVA powder will only form a solution when heated; it is preferred to heat the mixture in a 110 °C oven rather than using a microwave, as other literature suggests, in order to prevent boiling and loss of liquid. Filtering the liquid PVA may also be useful for ensuring a homogeneous and flat film, although it is not entirely necessary.

To create the thin PVA film, the liquid PVA is poured into a petri dish and the liquid is allowed to evaporate. First, a standard 10 cm Petri dish is plasma treated for 10 seconds to help the final film remain attached to the dish. Plasma-treating the petri dish before pouring in the liquid PVA ensures that the film remains attached to the petri dish and does not lift off prematurely. After cooling down to room temperature, the liquid PVA mixture is poured into the plasma-treated dish. The dish is left uncovered in a 65 °C oven overnight to allow the liquid to completely evaporate. The next day, the dish is removed from the oven revealing a thin, dried PVA film loosely attached to the bottom of the Petri dish. The film is then cut into appropriate sized pieces and used as needed, or the dish can be covered and sealed with parafilm for long-term storage.

2.1.3 Soft PDMS substrate preparation

Flexible substrates are manufactured as previously published [71, 82]. Soft PDMS (Sylgard 527, 1:1 ratio of parts A and B, Dow Corning) and normal PDMS (Sylgard 184, 10:1 ratio of base to curing agent) are first prepared separately and allowed to degas for at least 20 minutes under vacuum. The two types of PDMS are then mixed to form a mixture of Sylgard 184 and Sylgard 527. The final substrate stiffness can be controlled by varying the precise mixture of these two types of PDMS. For example, a mixture of 5% Sylgard 184 and 95% Sylgard 527 by weight results in a final substrate stiffness of 13.5 kPa. The mixture is degassed for 20 minutes under vacuum to remove any bubbles.

Round glass coverslips (25 mm diameter, #1 thickness, VWR) are plasma treated for 30 seconds (Plasma Prep II, SPI Supplies) and a 100-130 μL droplet of the PDMS mixture is

placed onto each of the plasma-treated glass coverslips. The PDMS droplets are allowed to spread across the glass coverslips on a level counter top for at least 30 minutes, resulting in a PDMS layer that is approximately 250 μm in height. Keeping the platform flat to ensure even spreading of PDMS is important to reduce tilt when imaging. Placing the coverslips onto the *inside* of the top lid of a petri dish, rather than the bottom, provides a more consistent flat surface. The PDMS-coated coverslips are degassed for 30 minutes before transferring to a 65 °C oven overnight to cure. The following day, the PDMS substrates are removed from the oven and cooled at room temperature.

To extract unpolymerized monomers, the PDMS substrates were submerged in 100% ethanol for at least 3 hours, followed by multiple rinses with DI water before drying in a 65 °C oven overnight. The substrates should be used as soon as possible or sealed with parafilm for long-term storage. It is very important that the soft PDMS substrates remain as clean as possible after creation; keeping them stored in a parafilm-closed petri dish and only opening the dish in a TC hood are useful for preventing dust buildup.

Standard tensile testing apparatus are not sensitive enough to measure the material properties of these low stiffness samples, so a first-principles approach was used. Dog bone-shaped samples of mixtures of PDMS (Sylgard 527 and 184) were created to measure their stiffness. Mixtures of 2.5%, 5%, and 10% ratios of Sylgard 184 to Sylgard 527 were poured into molds and cured at 65 °C overnight. Two spots used for measuring strain were drawn in the neck region of the sample. Several loads were hanged from a sample and imaged with a phone camera to evaluate the resulting engineering stress and strain for each load (Fig. 2.1A). Stress was calculated by dividing the applied weight by the minimum cross-sectional area of the sample (measured by analog calipers), and strain was calculated by measuring the distance between the two spots drawn previously. When plotted, the data points form a straight line where the slope of the line is the stiffness or Young's modulus (Fig. 2.1B). Three separate samples were created and tested for each PDMS mixture. The initial unstretched distance between the two spots was not measured, so the intercept of the linear stress-strain curves is not accurate; however, the slope of these lines is not affected by this oversight.

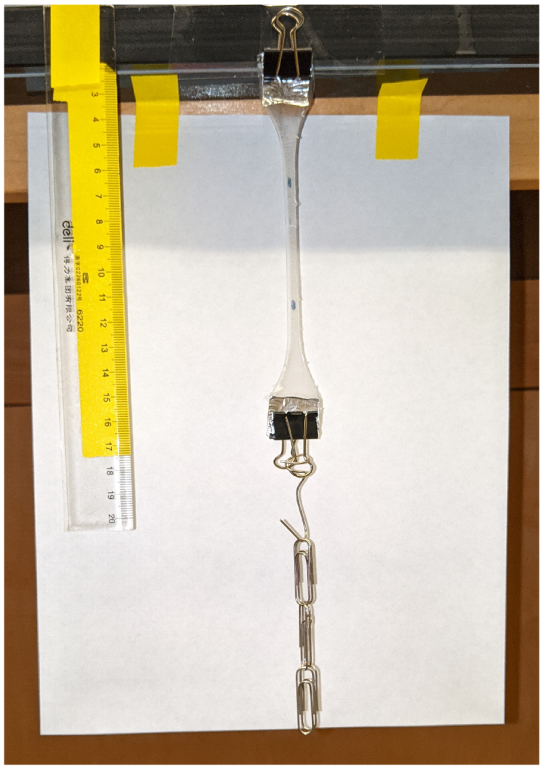
These measurements found that mixtures containing 2.5%, 5%, and 10% Sylgard 184 results in stiffnesses or Young's modulus of 7.7, 13.5, and 46.7 kPa. These measurements agree closely with prior literature [71]. Mixtures of pure Sylgard 527 are too difficult to accurately measure with this technique, so literature values must be relied upon (typically 5 kPa).

2.1.4 Patterning black dots on soft PDMS substrates

The patterned PDMS stamps and PVA film were used to deposit a layer of fluorescent protein onto the flexible PDMS substrates similar to previously published techniques with some modifications (Fig. 2.2B) [57, 103]. All steps were performed at room temperature and preferably in a standard tissue culture hood. First, Alexa Fluor 488, 594, or 647-conjugated-bovine serum albumin (BSA) (5 mg/mL, Life Technologies) is diluted 1:2000 in PBS (1X without calcium or magnesium, Life Technologies), and a 400 μL droplet is gently placed onto a patterned stamp (about 1 cm^2 area) within a petri dish. The droplet is left on the stamp for 30 minutes to allow the fluorescent BSA to adsorb onto the surface. If the concentration of BSA is too high, the pattern features may get filled in with fluorescence and result in a poor-looking pattern when transferred. Avoid too small features or too high concentration of BSA. Fresh PBS is slowly added to the petri dish until the liquid level rose above the stamp. The stamp is removed from the PBS and rinsed 3 times in fresh PBS dishes by gently submerging the stamp. Special care should be taken when rinsing the stamp with PBS. Remove the stamp from the PBS slowly and at an angle to ensure that as much liquid as possible is removed from the stamp. When PBS dries, it tends to leave behind salt crystals which can destroy the pattern if it is allowed to dry on top of the stamp. After the final rinse, the stamp is thoroughly dried with a gentle stream of nitrogen gas. A low velocity stream of nitrogen gas is preferred as the fluorescent BSA layer can be fragile.

Next, the PVA film is used to transfer the fluorescent pattern from the stamp to the flexible substrate. A PVA film is trimmed to a size slightly larger than the stamp. The film is plasma treated for 60 seconds to facilitate protein transfer from the stamp. Using a pair of tweezers, the film is then lowered onto the dried stamp. The film is gently pressed

A



B

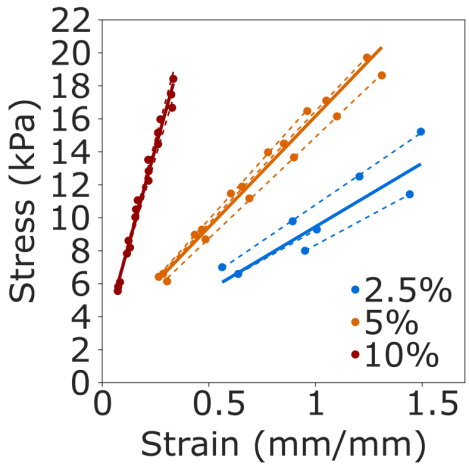


Figure 2.1: Measurement of soft PDMS stiffness. (A) PDMS tensile testing apparatus to measure the Young's modulus of each mixture of PDMS. (B) Measurements of stress and strain from the tensile test. Dotted lines indicate lines of best fit for each experiment. Solid lines are the average slope and intercept for the dashed lines for each given PDMS mixture.

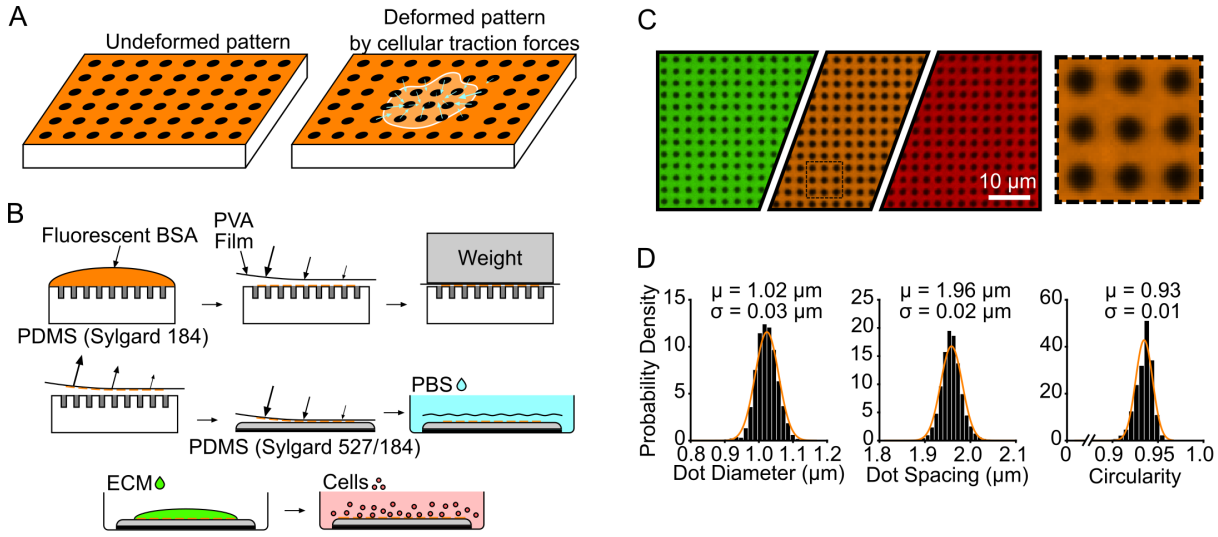


Figure 2.2: Black dots overview, manufacturing, and characterization. (A) Principle of the black dots platform, where tension from an adhered cell causes the pattern of dots to displace. (B) Manufacturing black dots substrates using microcontact printing and a sacrificial PVA film to transfer an array of fiducial markers from a patterned stamp to a soft substrate. (C) Example of final manufactured substrate that can be made in the desired fluorescent channel using different fluorescent BSA such as BSA-Alexa Fluor 488 (green), BSA-Alexa Fluor 594 (orange), and BSA-Alexa Fluor 647 (red). The black dotted line area is shown on the right, scaled up 4X larger. (D) Characterization of diameter, center-center spacing, and circularity of the black dots pattern. μ = mean, σ = standard deviation. Data from 25,081 individual dots from 2 substrates. Y-axis is Probability Density for all three plots. Normal Gaussian probability density functions are overlayed.

onto the stamp using rounded-tip tweezers from the center outwards to remove any gaps, and a thin piece of glass slide is placed on top of the film. A 50-gram weight is placed onto the glass slide to maintain close contact between the film and stamp. After 20 minutes, the weight and glass slide are removed and the PVA film is gently peeled from the stamp and transferred to the flexible PDMS substrate. Again, rounded-tip tweezers are used to gently press the film onto the substrate and remove any air gaps. The film is left on the flexible PDMS substrate for 20 minutes. The substrate is then submerged in PBS for up to 5 minutes, causing the film to rehydrate and float away from the surface where it can be discarded. The final substrate containing the pattern of fluorescent BSA is stored in PBS overnight at 4 °C before cell seeding and can be stored for at least 1 week.

Once the black dots technique is optimized, it provides a consistent pattern and can be tailored to suit the nature of the cells. Black dots substrates have been manufactured using BSA conjugated with Alexa Fluor 488, 594, and 647 to demonstrate the versatility in the fluorescent coatings that are possible (Fig. 2.2C). The pattern uniformity is critical for obtaining accurate results from image analysis and force calculations. Through quantitative image analysis, the black dots pattern was found to be uniform in size (1.02 ± 0.03 μm diameter), spacing (1.96 ± 0.02 μm center-to-center), and shape (0.93 ± 0.01 circularity) (Fig. 2.2D).

Using this protocol, areas with black dots pattern are reliably printed up to 1 cm^2; but areas up to 6 cm^2 have been manufactured with a larger PDMS stamp (Fig. 2.3). The black dots approach could potentially be scaled to larger culture dishes for even higher throughput in measurements. Overall, the microcontact printing and sacrificial film technique can deposit fluorescent BSA patterns of black dots with regular size, spacing, and shape that cover a large surface area for experiments with cells.

2.1.5 Functionalization of black dots substrates for cell adhesion

While cells may adhere directly to the fluorescent BSA, a more controlled adhesion can be achieved by placing a layer of relevant ECM over the black dots pattern. Generally, a droplet

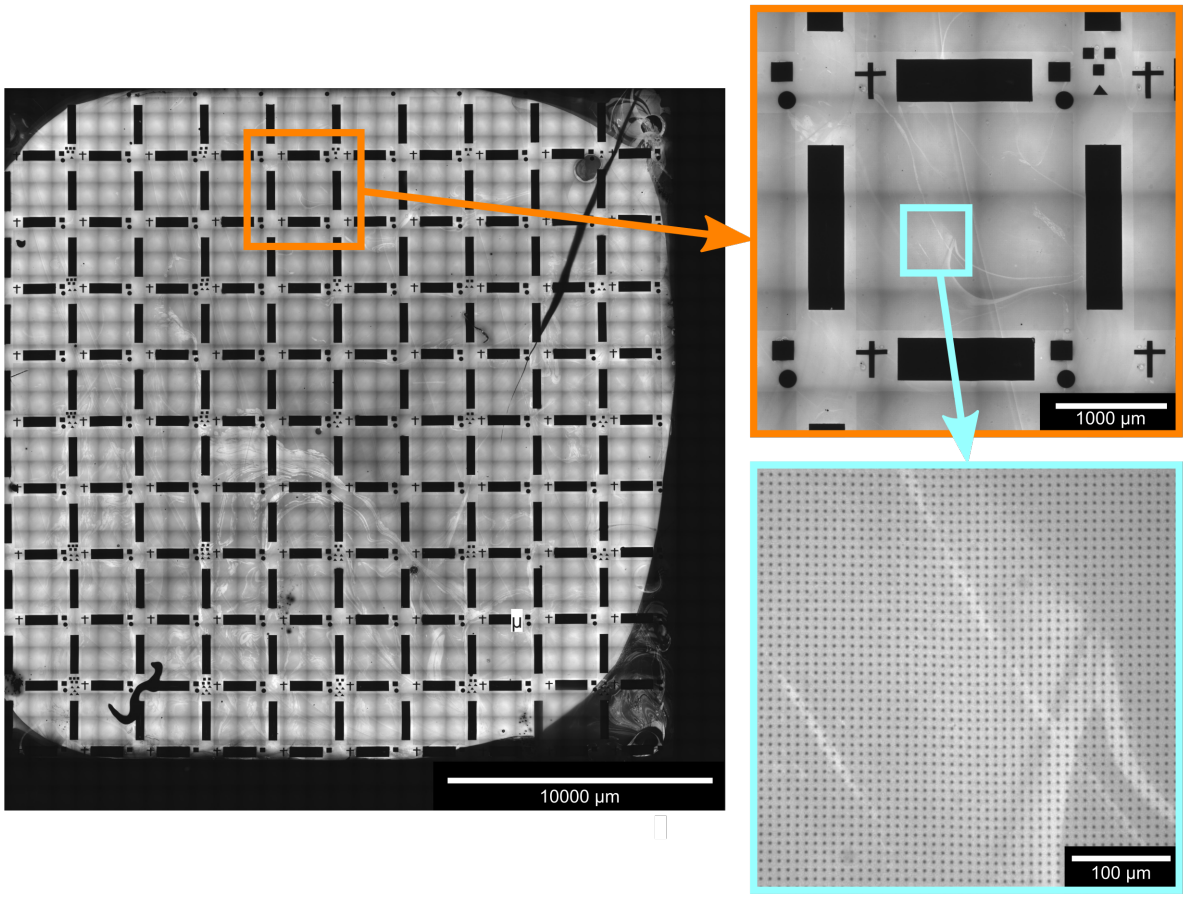


Figure 2.3: Example of black dots pattern printed over larger areas. In this example, the black dots pattern has a 9 µm center-to-center spacing and covers a total physical area of 6 cm^2.

of ECM protein diluted between 5-150 μg/mL is gently placed on top of the finished black dots substrate surface. The droplet is allowed to remain on the surface for 1-2 hours until the surface is saturated with adsorbed protein. To encourage droplet spreading over the black dot surface, a glass coverslip may be gently placed on top of the droplet. To block the surface, the substrate can be submerged in a 0.2% Pluronic F-127 (BASF) in PBS for 30 minutes. The substrate is finally submerged into PBS and stored until cell seeding. Substrates that have been functionalized should be used within 1 week to ensure the ECM protein is fresh.

Black dots substrates have been functionalized with fibrinogen, laminin, and von Willebrand Factor (VWF). Fibrinogen from human plasma that is conjugated to Alexa Fluor-488 (Life Technologies) was diluted to 150 μg/mL and incubated on the black dots substrate surface for 1 hour. Similarly, mouse laminin (Life Technologies, 23017015) was diluted to 10 μg/mL and VWF (Haematological Technologies) was diluted in PBS to 5 μg/mL and incubated on black dots substrate surface for 1 hour. Depending on the protein, different adsorption patterns were observed (Fig. 2.4); some proteins preferentially adhered to areas with fluorescent BSA, although protein quantification revealed that some protein did adhere to the dots themselves. Optimization may be required for other proteins not discussed here.

2.1.6 Other considerations and findings

Throughout the development of this approach, several processes were optimized to obtain robust, repeatable substrate manufacturing. A summary of the results from these optimizations is given here:

Fluorescent protein Alexa Fluor-conjugated BSA (AF-BSA) and goat-anti-mouse antibody (AF-GaM) were used to create black dots substrates. No major difference was found, but BSA was chosen due to its common use as a blocking agent and the risk of an antibody reacting with cells.

Protein adsorption time The adsorption time of the fluorescent protein was tested for

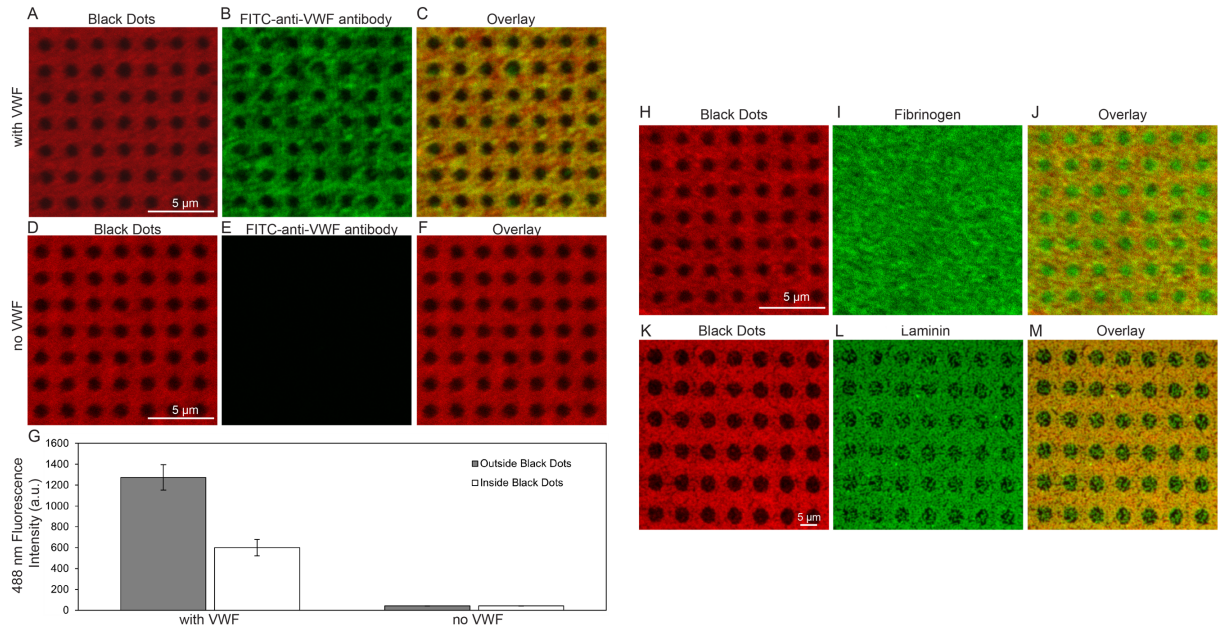


Figure 2.4: (A-C) VWF-treated black dots substrates with fluorescent BSA shown in red (A), VWF labeled with a FITC-anti-VWF antibody shown in green (B), and an overlay (C). Qualitatively, VWF is bound to both the fluorescent-BSA (orange in C) as well as the non-fluorescent portion of the black dots pattern (green in C). (D-F) Black dots substrates without VWF treatment with fluorescent BSA shown in red (D), VWF labeled with a FITC-anti-VWF antibody shown in green (E), and an overlay (F). As expected, the FITC-anti-VWF antibody does not bind to the substrate without VWF treatment (E). (G) Quantitatively, substrates with VWF have VWF bound both outside the black dots (gray bar) and inside the black dots (white bar), with higher fluorescent intensity outside the black dots. As expected, substrates without VWF have negligible 488 fluorescence both inside and outside the black dots. (H-I) Black dots substrates can also be treated with fluorescent-fibrinogen, which binds uniformly across the surface, or (K-M) laminin, which has some preference for binding outside the black dots. Note that this panel shows a black dots pattern with a larger diameter.

AF-GaM, from 1 to 120 minutes. The fluorescence intensity of protein on the stamp and on the final substrate were calculated from images. While longer time was associated with higher fluorescence, a time of 30-60 minutes was used for most studies, which had more than enough fluorescence for fixed images where exposure time can easily be modified. For videos, longer adsorption time may be required.

Protein mixing and spinning The amount of mixing or centrifuge spinning of the fluorescent protein in PBS was tested, but no effect on the final substrate quality was found. It is recommended to spin the stock solution of fluorescent protein to pull any large particles to the bottom where they can be avoided.

Surface treatment Several surface treatments of the final soft PDMS substrate were tested for use with direct stamping, including bare, NaOH treatment (5M for 30 minutes), plasma/UV treatment, or additional curing at 110 °C. Of these, NaOH offered no improvement and plasma/UV treatment was discarded due to the surface hardening effect. Bare surface was used for further experiments.

Protein solvent Deionized water or PBS were used to dilute the fluorescent protein. Since most proteins are pH sensitive, this will depend on the fluorescent protein used. For AF-BSA or AF-GaM, PBS should be used to dilute the protein.

Protein droplet removal To remove the droplet of fluorescent protein from the PDMS stamp, two methods were tried: using a gentle stream of nitrogen gas, or using PBS to submerge the stamp/droplet. Using the nitrogen gas resulted in wildly inconsistent pattern, often containing many filled-in dots or areas with highly differential fluorescence intensity. Submerging the stamp in PBS followed by several steps of washing in PBS was found to yield the most consistent pattern.

Stamping process Two methods for transferring the protein from the stamp to the substrate were tried: direct microcontact printing and the sacrificial PVA film method.

Direct stamping does not work well without plasma/UV treatment of the substrate, which causes surface hardening rendering it useless. The sacrificial PVA film method was used with great success. It is recommended to heat up the PVA powder slowly in DI water, rather than use a microwave as suggested by the original inventors [103].

Protein adsorption on PDMS Both Sylgard 184 and Sylgard 527 were tested throughout the process for use with the stamp or the final substrate. Notably, it was difficult to adsorb fluorescent protein directly onto Sylgard 527 (soft PDMS), while it was relatively easy to adsorb onto Sylgard 184.

2.2 Imaging and analysis

Black dots substrates are amenable to imaging with both widefield and confocal microscopy. The working distance of the objective should be large enough to account for the thickness of the substrate (at least 250 μm). Images should be taken with a large enough field of view to ensure several black dots with no deformation surround each cell, with high enough image resolution such that each black dot has a diameter of at least 10 pixels. If an image contains multiple cells, it should be cropped to only include one cell for the image analysis (Fig. 2.5A).

2.2.1 Image analysis to extract black dot positions

To estimate the centroids of the black dots (Fig. 2.5B), two approaches were taken: 1) a fully automated approach using a modified object tracking code, and 2) a more manual approach by estimating the grid position and analyzing each dot separately. The core code for the image analysis is included in Appendix A.3.

Automated approach

An existing object tracking code was modified for black dots analysis [9]. This code automatically finds objects of a certain size, given as inputs by the user. The fluorescent image

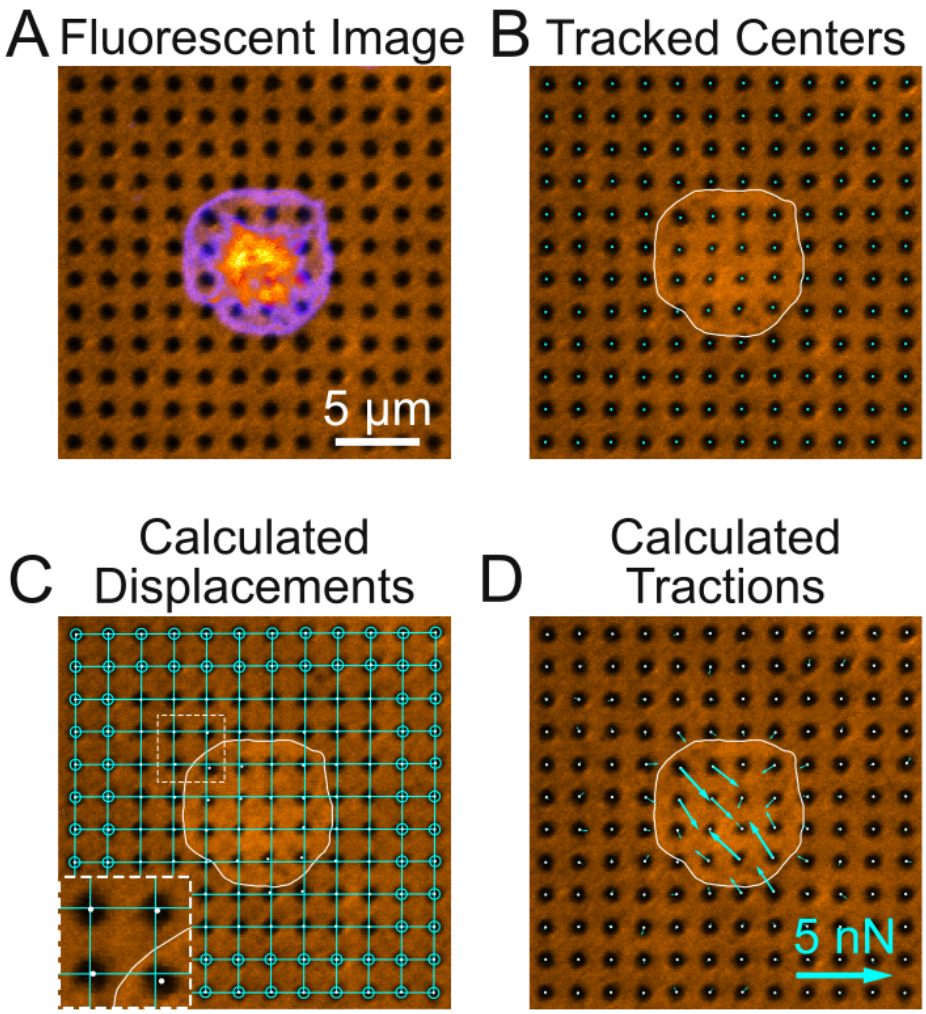


Figure 2.5: (A) A fluorescence image of deformed substrate with platelet stained for F-actin (pixel intensity scaled from purple to yellow). (B) The black dots pattern is binarized and the centroid of each dot is found using automated detection. (C) Undeformed dots near the edges (circled) are used to fit horizontal and vertical lines throughout the entire field of view. The intersection of these lines marks the zero-displacement state of each dot. Inset is 2x magnified. (D) Forces are calculated from the displacement of each dot relative to its zero-displacement (undeformed) location.

of the black dots substrate is first run through a spatial bandpass filter, followed by a peak finding algorithm to find the centroid of each dot to subpixel accuracy. While not perfect, it generally finds all the black dots very quickly and with ease. Poor looking black dots may include deformities that can get picked up by this code; these extra dots are removed manually before proceeding with the analysis.

The particle tracking code yields a list of centroid positions (one for each dot). These positions are fit into a 2D rectangular grid using a novel approach developed here. Starting with the dot closest to the center of the image, the position of the adjacent dots left and right are extrapolated from. This is continued until the edge of the image is reached. These dots are grouped as a "line" of the grid. Once all the horizontal lines are found, the process is repeated for the vertical lines. In this way, the centroid positions are organized onto a 2D grid, ready for the reference position of each dot to be estimated.

Manual approach

In this approach, a grid of the appropriate size is created where each grid point corresponds to a single black dot. First, the black dots image is denoised with a Gaussian filter and binarized. Next, a proximity mask is created such that each pixel is given a value for how far away it is from the nearest black dot edge; pixels located inside black dots have a negative value, while pixels between black dots have a positive value. This creates a hilly landscape of sorts where a ball dropped at any location will fall towards the center of the nearest black dot. The nominal dot size and spacing is used to overlay an estimate grid onto this hilly landscape. For each estimated grid point, an iterative process finds the approximate centroid of the nearest black dot: the estimate is repeatedly updated to be the lowest value of the surrounding 3x3 region of pixels. This iterative process stops when the estimate is already the minimum of the surrounding 3x3 region, which corresponds to the approximate centroid of the black dot. Finally, the actual centroid is calculated from the binarized black dots image using the approximate centroid. This process results in an extremely fast centroid measurement that is fairly robust, assuming that the initial black dot estimate locations

are reasonable (because the estimate will simply fall towards the nearest black dot). The advantage of this process over the more automated approach is that the data will already be organized into a 2D grid.

2.2.2 Estimation of the reference state

To calculate the displacement of each dot, the reference state of each dot must be determined. The dots in the image are organized into a 2D rectangular array of rows and columns. For each row and column, a line is fit through four of the dots near the edges of the image (Fig. 2.5C). The dots near the edge of the image are assumed to have little or no displacement and can therefore be used as reference for the highly displaced dots near the cell. The lines fit through the rows intersect with lines fit through the columns, and the intersection points are used as the reference state of each dot. From here, the displacement of each dot can be calculated by subtracting the reference position from the deformed position. Due to the image rotation, some grid intersections may lie off the edge of the image. For these positions, a "fake" dot is created and the displacement is set to zero.

2.3 Mathematical theory of traction force estimation from displacements

2.3.1 Assumptions

To estimate the forces from black dot displacements, Fourier Transform Traction Cytometry (FTTC) with regularization is used. FTTC was first described by Butler *et al.* [13], and regularization was later introduced by Sabass *et al.* [84]. In these formulations, several assumptions must be made:

1. The domain is an infinite half-space; the material thickness is much larger than the cell and its displacements

2. The material is linearly elastic, with Young's modulus E and Poisson's ratio ν

3. The displacements are measured directly at the surface of the half-space

4. The displacements and traction forces are assumed to occur in-plane on the surface; out-of-plane forces are considered negligible

For assumption 1, the apparent stiffness at the surface of the underlying PDMS substrate is a function of the thickness; thinner substrates may appear stiffer to cells than thicker ones even if the bulk stiffness of the PDMS is the same. The critical thickness for which cells can sense the thickness is debated [58]. However, other work has shown that cells with lateral dimension of ∼100 μm were not adversely affected by a substrate thickness greater than ∼200 μm [55]. A more rigorous mathematical analysis found that the apparent stiffness is affected when the dominant wavelength in the deformation field (typically equal to the cell width) is greater than 3 times the thickness of the substrate [22]. In our case, the thickness of our PDMS substrate (∼250 μm) is much larger than the length of typical cells (∼20 to 100 μm), so the effect of substrate thickness on the apparent stiffness should be negligible. However, care should be taken if larger cells or monolayers of cells, which may affect the dominant wavelength in the deformation field, are used.

Assumption 2 is reasonable for the Sylgard 527/Sylgard 184 PDMS mixtures used for black dots substrates [71]. Dogbone samples were created and a simple tensile test was performed that also shows the highly linear nature of these PDMS mixtures. A Poisson's ratio of 0.5 is assumed for the PDMS mixtures used here, although it has been shown that traction force errors are sensitive to small changes in Poisson's ratio around the incompressible limit for thinner substrates [22].

For standard TFM that employs fluorescent beads, corrections need to be made to account for the distance between the beads and the surface. This can be mitigated by performing extra processing steps (e.g., centrifugation) to pull the beads close to the surface. The black dots approach avoids this entirely by directly depositing the fluorescent markers onto the surface; therefore, assumption 3 is guaranteed.

The last assumption 4 is made to keep the system as simple as possible and reduce mathematical complexity. Under typical use, black dots patterns do not displace out-of-

plane enough to be detectable by widefield microscopy.

2.3.2 Mathematical formulation

Here, the mathematical equations of FTTC and its use to calculate traction forces will be detailed. First, the displacements $\mathbf{u}(\mathbf{r})$ measured at points $\mathbf{r} = (x, y)$ are related to the traction forces $\mathbf{t}(\mathbf{r}')$ measured at points $\mathbf{r}' = (x', y')$ in the domain Ω on the surface:

$$\mathbf{u}(\mathbf{r}) = \int G(\mathbf{r}, \mathbf{r}')\mathbf{t}(\mathbf{r}')d\Omega \tag{2.1}$$

where $\mathbf{G}(\mathbf{r}, \mathbf{r}')$ are the Green's functions and the displacements $\mathbf{u} = (u_x, u_y)$. In this case, the simplified Boussinesq Green's functions are used:

$$\mathbf{G}(\mathbf{r}, \mathbf{r}') = \frac{(1+\nu)}{\pi E|\mathbf{r}-\mathbf{r}'|^3}\begin{bmatrix}(1-\nu)|\mathbf{r}-\mathbf{r}'|^2 + \nu u_x^2 & \nu u_x u_y \\ \nu u_x u_y & (1-\nu)|\mathbf{r}-\mathbf{r}'|^2 + \nu u_y^2\end{bmatrix} \tag{2.2}$$

where E and ν are the Young's modulus and Poisson's ratio of the underlying substrate. It is important to note that here, a singularity occurs if $|\mathbf{r} - \mathbf{r}'| = 0$. Effectively, this means that traction forces cannot be measured at the exact same location where displacement has been measured.

Since the displacements are measured directly from images, the traction forces can be solved for by inverting equation 2.1. However, the convolution integral complicates this inversion; equation 2.1 describes how all of the forces are related to all of the displacements, resulting a large ill-conditioned interaction matrix that is difficult to invert, especially when working with the large sets of displacements needed for sufficient spatial resolution of force. To overcome this, equation 2.1 can instead be solved in the Fourier domain by using a Fourier Transform (FT), causing the integral to disappear:

$$\tilde{\mathbf{u}}(\mathbf{k}) = \tilde{\mathbf{G}}(\mathbf{k})\tilde{\mathbf{t}}(\mathbf{k}) \tag{2.3}$$

where now $\mathbf{k}$ are the 2D wave vectors $\mathbf{k} = (k_x, k_y)$; equation 2.3 measures quantities in a frequency domain instead of a spatial domain. Applying the Fourier Transform to the

Boussinesq Green's functions yields:

$$\tilde{\mathbf{G}}(\mathbf{k}) = FT\left[\mathbf{G}(\mathbf{r},\mathbf{r}')\right] = \frac{2(1+\nu)}{E|\mathbf{k}|^3}\begin{bmatrix}(1-\nu)|\mathbf{k}|^2+\nu k_x^2 & -\nu k_x k_y \\ -\nu k_x k_y & (1-\nu)|\mathbf{k}|^2+\nu k_y^2\end{bmatrix} \tag{2.4}$$

In the Fourier domain, equation 2.3 now shows that traction force at one wave vector is directly related to displacement at that wave vector by a single 2×2 matrix (Equation 2.4). In this way, a large interaction matrix is avoided and computational effort is significantly reduced. Now, traction forces in the Fourier domain $\tilde{\mathbf{t}}(\mathbf{k})$ are found by inverting equation 2.3:

$$\tilde{\mathbf{t}} = \tilde{\mathbf{G}}^{-1}\tilde{\mathbf{u}} \tag{2.5}$$

However, in most cases a regularization scheme should be employed to account for noise in the data. Regularization is performed by replacing equation 2.5 with:

$$\tilde{\mathbf{t}} = \left(\tilde{\mathbf{G}}^T\tilde{\mathbf{G}} - \lambda^2\tilde{\mathbf{H}}\right)^{-1}\tilde{\mathbf{G}}^T\tilde{\mathbf{u}} \tag{2.6}$$

Where λ is the regularization coefficient and $\tilde{\mathbf{H}}$ is the regularization tensor. Here, choose $\tilde{\mathbf{H}}$ to be the identity matrix, resulting in 0^{th}-order Tikhonov regularization. Regularization generally allows for smoothing of the solution to reduce errors due to noise in the measurements. The choice of λ is not unique and is discussed in section 2.3.3.

The right hand side of equation 2.6 can be directly computed using equation 2.4. Finally, the traction forces in the spatial domain $\mathbf{t}(\mathbf{r}')$ are calculated by applying an inverse Fourier Transform:

$$\mathbf{t}(\mathbf{r}') = FT^{-1}(\tilde{\mathbf{t}}(\mathbf{k})) \tag{2.7}$$

In practice, equations 2.1-2.7 are performed with data stored in matrices. For example, black dots substrates have m rows and n columns of dots where displacements (u_x, u_y) are measured. Here, m and n are assumed to be even. These displacements are stored in a single matrix for ease of handling; u_x and u_y will each be matrices of size $m \times n$. Next, the wave vectors (k_x, k_y) are calculated and stored in similar sized matrices. For instance, k_x is

a matrix of size $m \times n$, where each *column* is identical and is equal to:

$$\text{Cols of } k_x = \frac{2\pi}{md}\begin{bmatrix} 0 & 1 & \cdots & \left(\frac{m}{2}-1\right) & -\left(\frac{m}{2}\right) & -\left(\frac{m}{2}-1\right) & \cdots & -1 \end{bmatrix}^T \tag{2.8}$$

where d is the grid spacing which is assumed to be the same between rows and between columns. Similarly, k_y is also a matrix of size $m \times n$, where each *row* is identical and equal to:

$$\text{Rows of } k_y = \frac{2\pi}{nd}\begin{bmatrix} 0 & 1 & \cdots & \left(\frac{n}{2}-1\right) & -\left(\frac{n}{2}\right) & -\left(\frac{n}{2}-1\right) & \cdots & -1 \end{bmatrix} \tag{2.9}$$

Note that the position of the dots is not used when calculating the wave vectors, only the dot spacing. In this way, each element of these matrices contains a unique wave vector where displacement and force are directly related via equation 2.4.

Rather than calculate it every time, it is more efficient to work out everything in front of $\tilde{\mathbf{u}}$ in equation 2.6. Since the Green's function is itself a 2×2 matrix as shown in equation 2.4, this quantity will also be a 2×2 matrix.

$$\left(\tilde{\mathbf{G}}^T\tilde{\mathbf{G}} - \lambda^2\tilde{\mathbf{H}}\right)^{-1}\tilde{\mathbf{G}}^T = \begin{bmatrix} \tilde{G}_{xx}^{-1} & \tilde{G}_{xy}^{-1} \\ \tilde{G}_{xy}^{-1} & \tilde{G}_{yy}^{-1} \end{bmatrix} \tag{2.10}$$

The values here can be directly determined by performing laborious matrix calculations in equation 2.6. For completeness, the full forms are shown here:

$$\tilde{G}_{xx}^{-1} = \left[\frac{\begin{array}{c} 2E\left(\nu+1\right)\left(\lambda^2E^2{k_x}^4 + \lambda^2E^2{k_y}^4 - \lambda^2E^2{k_x}^4\nu \right. \\ -\lambda^2E^2{k_x}^2{k_y}^2\nu + 2\lambda^2E^2{k_x}^2{k_y}^2 - 4{k_x}^2\nu^3 - 4{k_x}^2\nu^2 \\ \left. +4{k_x}^2\nu + 4{k_x}^2 + 4{k_y}^2\nu^4 - 8{k_y}^2\nu^2 + 4{k_y}^2\right) \end{array}}{\begin{array}{l} 16(\nu-1)^2(\nu+1)^4\sqrt{{k_x}^2+{k_y}^2} \\ \quad + E^2\lambda^2\left({k_x}^2+{k_y}^2\right)^{3/2}\left(\lambda^2E^2{k_x}^2 + \lambda^2E^2{k_y}^2 + 4\nu^4 - 4\nu^2 + 8\nu + 8\right) \end{array}}\right] \tag{2.11}$$

$$\tilde{G}_{yy}^{-1} = \left[\frac{\begin{array}{c} 2E\left(\nu+1\right)\left(\lambda^2E^2{k_x}^4 + \lambda^2E^2{k_y}^4 - \lambda^2E^2{k_y}^4\nu \right. \\ -\lambda^2E^2{k_x}^2{k_y}^2\nu + 2\lambda^2E^2{k_x}^2{k_y}^2 - 4{k_y}^2\nu^3 - 4{k_y}^2\nu^2 \\ \left. +4{k_y}^2\nu + 4{k_y}^2 + 4{k_x}^2\nu^4 - 8{k_x}^2\nu^2 + 4{k_x}^2\right) \end{array}}{\begin{array}{l} 16(\nu-1)^2(\nu+1)^4\sqrt{{k_x}^2+{k_y}^2} \\ \quad + E^2\lambda^2\left({k_x}^2+{k_y}^2\right)^{3/2}\left(\lambda^2E^2{k_x}^2 + \lambda^2E^2{k_y}^2 + 4\nu^4 - 4\nu^2 + 8\nu + 8\right) \end{array}}\right] \tag{2.12}$$

$$\tilde{G}_{xy}^{-1} = \left[\begin{array}{l} \dfrac{-2Ek_xk_y\nu\left(\nu+1\right)\left(\lambda^2E^2{k_x}^2+\lambda^2E^2{k_y}^2+4\nu^3+4\nu^2-4\nu-4\right)}{16(\nu-1)^2(\nu+1)^4\sqrt{{k_x}^2+{k_y}^2}} \\ \quad +E^2\lambda^2\left({k_x}^2+{k_y}^2\right)^{3/2}\left(\lambda^2E^2{k_x}^2+\lambda^2E^2{k_y}^2+4\nu^4-4\nu^2+8\nu+8\right) \end{array} \right] \tag{2.13}$$

Each of these quantities will be size $m \times n$ due to the size of k_x and k_y. When performing operations within these formulations for $\tilde{\mathbf{G}}^{-1}$, element-wise operations should be used (i.e., .* and .^ in MATLAB) because k_x and k_y are simply storage matrices for the wave vectors. Divergence can occur when $k_x = k_y = 0$ and is avoided by artificially setting $\tilde{G}_{xx}^{-1}(0,0) = \tilde{G}_{yy}^{-1}(0,0) = \tilde{G}_{xy}^{-1}(0,0) = 0$.

The traction forces in the Fourier domain can now be calculated by expanding equations 2.5 or 2.6:

$$\begin{aligned} \tilde{t}_x &= \tilde{G}_{xx}^{-1}\tilde{u}_x + \tilde{G}_{xy}^{-1}\tilde{u}_y \\ \tilde{t}_y &= \tilde{G}_{xy}^{-1}\tilde{u}_x + \tilde{G}_{yy}^{-1}\tilde{u}_y \end{aligned} \tag{2.14}$$

Finally, the traction forces in the spatial domain are calculated using equation 2.7.

2.3.3 Regularization discussion

Because the inverse problem is ill-posed, regularization is necessary to obtain a reasonable force estimate [85]. The value for the regularization parameter λ in equations 2.6 and 2.11-2.13 is not well defined and therefore there is no single unique solution. It is up to the user to decide what value to use for λ. There are some approaches that can guide towards an "optimal" value for λ, such as L-curve approaches [39]. However, these approaches were not developed for use with the FTTC method in mind. In the case of the black dots approach which uses the FTTC method, the L-curve criterion does not work well. Figure 2.6 shows a representative L-curve for regularization using FTTC for black dots analysis (black line) and an idealized L-curve (red line). The idealized L-curve contains a corner or "knee" that is typically chosen as the optimal value for λ. However, in the representative L-curve from black dots analysis, no such corner exists. Therefore, the choice for λ is decided simply

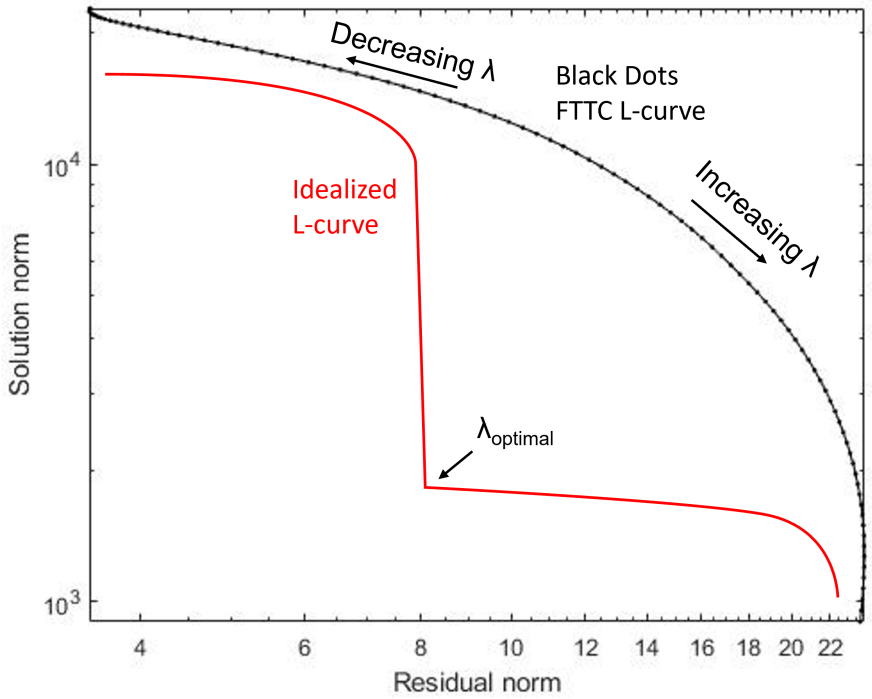


Figure 2.6: Example of an L-curve for regularization using FTTC for black dots analysis (black line) and an idealized L-curve (red line). The example L-curve does not contain a discernible corner to select an optimal value for λ.

by trial-and-error; a "good" value for λ will smooth out unwanted noise without disrupting major traction forces. It can be helpful to try and select a λ that minimizes the traction forces far away from the cell, but λ should not be so large as to completely smooth out the traction force gradient inside the cell.

2.4 Demonstration of the black dots approach with HS-27A cells

To demonstrate the black dots technique, HS-27A cells were seeded onto the substrates before fixing, staining, and measuring forces. HS-27A cells are a bone marrow-derived mesenchymal stromal cell line and were chosen here because they are relatively easy to culture and adhere to well to the black dots substrates even without added ECM. As a preliminary experiment,

HS-27A cells were seeded onto two different stiffness substrates to observe how these cells respond to stiffness and to investigate the sensitivity of the black dots technique.

2.4.1 Cell culture and immunocytochemistry

Black dots substrates were created with 5% and 10% Sylgard 184 (13.5 and 46.7 kPa). Substrates were placed in the bottom of a 6-well plate to facilitate cell seeding. No ECM was adsorbed because the cells were able to adhere well without added ECM, likely secreting their own. HS-27A cells were seeded onto black dots substrates at low density to ensure mostly single cells. A density of 20,000 cells in a single well of a 6-well plate was sufficient to prevent clusters of cells from forming.

After 24 hours, the substrates were removed from the 6-well plate and labeled with immunocytochemistry. Briefly, cells were fixed with 4% paraformaldehyde for 20 minutes, followed by permeabilization with 0.5% Triton X-100 for 10 minutes. Samples were then blocked for 1 hour in 10% goat serum, followed by labeling in primary and secondary antibodies for 1 hour each. The nucleus and focal adhesions were observed by staining with Hoechst 33342 and an anti-vinculin antibody.

2.4.2 Results

Total contractile force was measured from the black dots deformation and plotted against cell area (Fig. 2.7). Total force was calculated by summing the force magnitudes for all black dots beneath and immediately surrounding the cell. In these plots, a linear force-area relationship is observed for both stiffnesses which is a well-known phenomenon for many cell types [14, 37, 90, 94].

Upon visual inspection of the black dots pattern in Figure 2.7, it is not clear that the dots are being displaced for 13.5 kPa substrates. The histograms in Figure 2.8A-B show the displacements of all black dots measured from all cells in this study. The dots were categorized as “Outside Cell” or “Inside Cell” in order to determine if the dots inside the cell were being displaced. In addition, displacements were measured from several images on

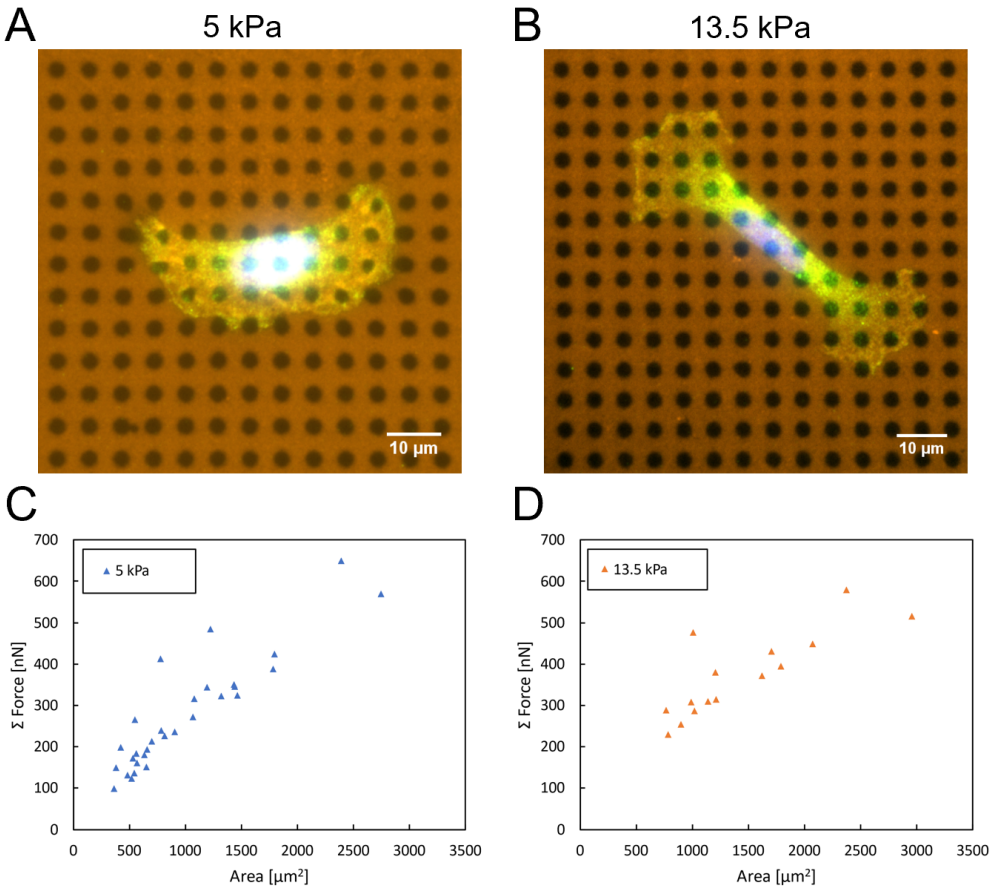


Figure 2.7: Example of an HS-27A cell deforming the black dots pattern for (A) 5 kPa and (B) 13.5 kPa substrates. Below, total force vs area of several HS-27A cells on (C) 5 kPa and (D) 13.5 kPa substrates.

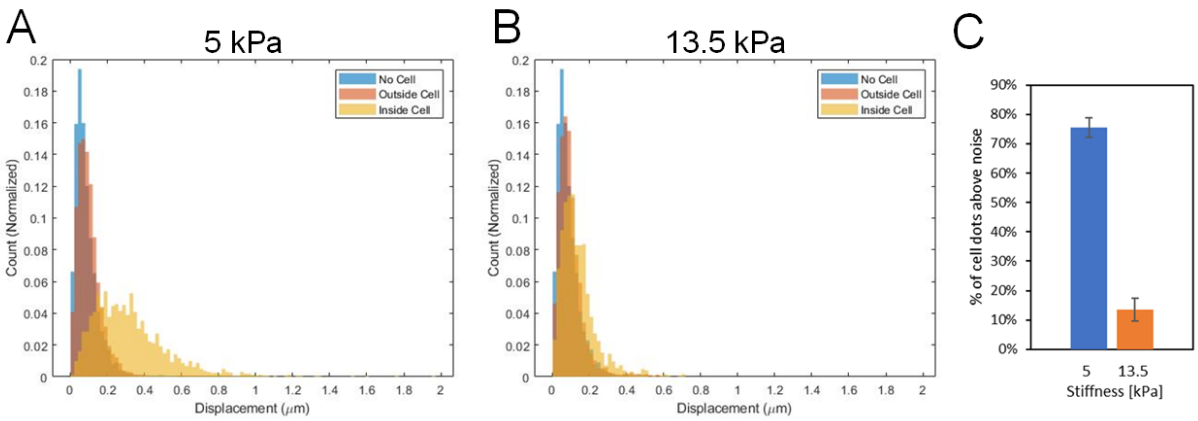


Figure 2.8: (A-B) Histograms for measured black dot displacements on 5 kPa and 13.5 kPa substrates. A noise threshold is chosen such that 95% of dot displacements measured outside the cell are below the threshold (approximately 0.22 μm for both 5 kPa and 13.5 kPa). (C) A majority of dots inside the cell boundary have displacements above the noise threshold on 5 kPa substrates, while very few of the dots displace above noise for 13.5 kPa substrates.

substrates without any cells (labeled "No Cell"). It is clear from the histogram on 5 kPa substrates that the dots inside cells are significantly more deformed than dots outside the cell or dots without cells. The histogram for dots inside cells on 13.5 kPa substrates, however, overlaps with the other histograms, indicating that these dots are not being displaced more than dots outside the cell. A threshold was set based on the "Outside Cell" histogram, such that 95% of dot displacements measured outside the cell are below the threshold. For both 5 kPa and 13.5 kPa substrates, the threshold was approximately 0.22 μm. Figure 2.8C shows that a majority of dots inside the cell on 5 kPa substrates are displaced above this threshold, but not for 13.5 kPa, suggesting that the black dots technique is not sensitive enough to measure cell tractions on 13.5 kPa substrates for this cell type.

By setting all displacements below the noise threshold to zero, this noise can be accounted for when calculating traction forces. Figure 2.9 shows the force-area relationship for HS-27A cells before and after the data has been denoised in this way. The data for HS-27A cells on

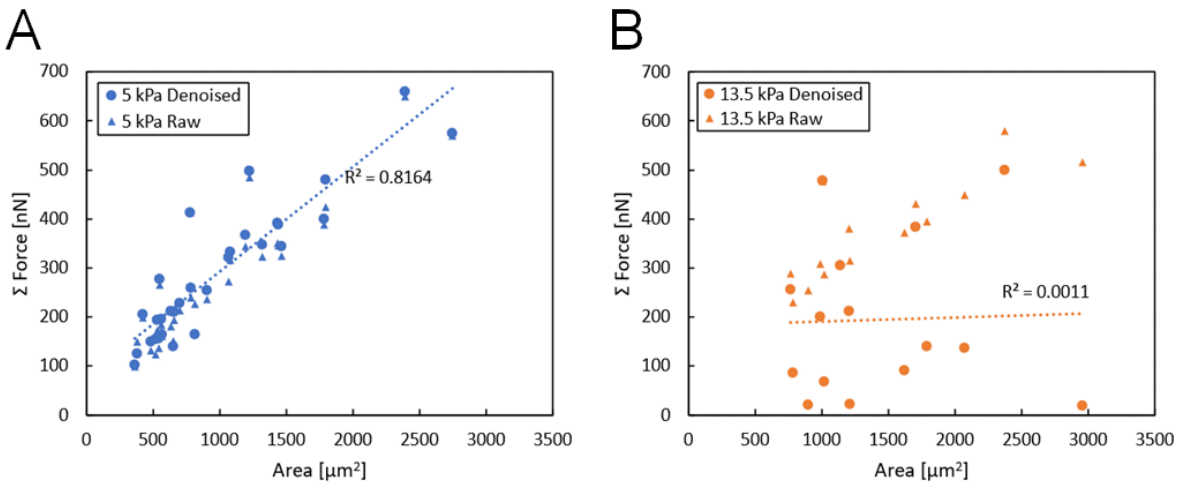


Figure 2.9: Total force vs area of HS-27A cells on (A) 5 kPa and (B) 13.5 kPa substrates before and after applying the noise threshold. The trend for 5 kPa substrates remains largely the same, while the forces on 13.5 kPa substrates are dramatically different because the displacements measured fell within the background noise.

5 kPa substrates remains largely unchanged after denoising, while the relationship on 13.5 kPa substrates disappears because the displacements measured were not significantly above the background noise. In addition to this sensitivity analysis, the effect of dot spacing on force estimation in platelets is shown in Chapter 3.2.11.

2.5 Conclusion

The black dots approach is a high-yield single-cell force measurement platform that is compatible with fixed cells without constraining cell shape and size. It relies on microcontact printing and sacrificial PVA film technology to deposit a grid of fluorescent protein that is used to measure substrate displacements. Regularized FTTC is used to calculate traction forces from these measured displacements to measure traction forces in single cells. The black dots method is highly tunable and potentially scalable due to the use of microcontact printing. By changing the substrate stiffness, ECM protein, and BSA fluorescence, the black

dots method may be used for measuring traction forces in many cell types.

Chapter 3

IDENTIFYING ROLES OF PLATELET SIZE, SHAPE, AND STRUCTURE IN GENERATING FORCE USING THE BLACK DOTS PLATFORM

To demonstrate the use of the black dots approach, the forces of many platelets were measured on black dots substrates and compared against platelet size, shape, and structure. The black dots approach facilitated the collection of a large data set of healthy human platelets, and helped to identify the individual and cooperative contributions of spread area, circularity, and dispersion of F-actin on platelet forces. This work was published in *Acta Biomaterilaia* [7].

3.1 Background on platelet force measurements

Platelets use their traction forces to adhere and form a hemostatic plug that stops bleeding [69, 96, 99]. During this process, the actin cytoskeleton of a platelet drives its shape change, spreading, and production of traction forces. Measuring these forces for individual platelets is challenging due to their small size (2-5 μm in diameter) [98], their ability to produce strong forces [43], and their sensitivity to collection and handling techniques [42]. It has been shown that the spread area of platelets correlated with the overall magnitude of their traction forces [37, 38]. While time-dependent changes in platelet shape and cytoskeletal structure have also been observed [70, 93], it is not known how these factors impact traction forces in platelets. Moreover, these factors may be interrelated because the actin cytoskeleton underlies changes in shape and spreading.

Previous measurements of platelet forces have used atomic force microscopy [50], classical TFM [37, 38, 43], reference-free TFM [66], and nanoposts [29] to elucidate properties of

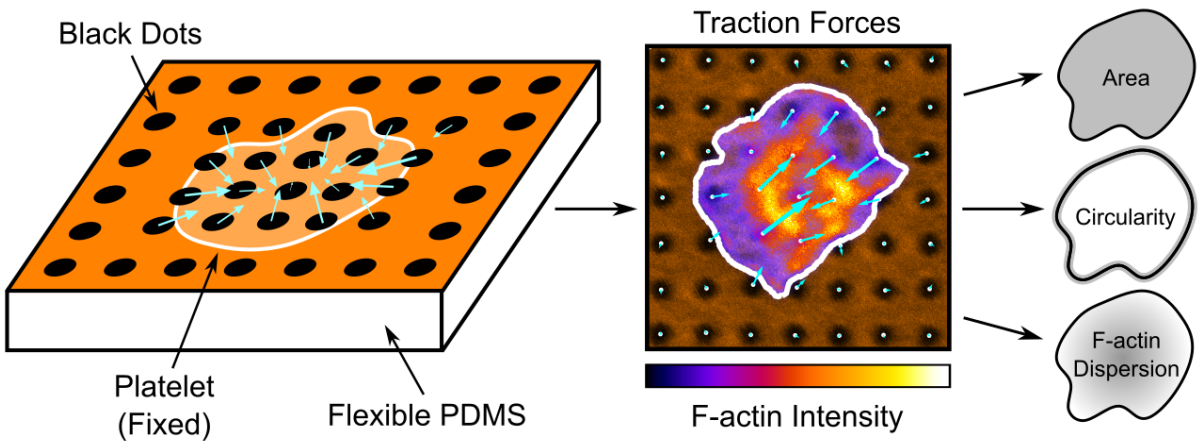


Figure 3.1: The black dots approach is used to measure displacements at the surface of flexible PDMS due to a contracting platelet. The displacements are used to calculate traction forces. Traction forces are compared against platelet area, circularity, and F-actin dispersion to assess how these factors influence or are influenced by force.

single platelets such as their temporal and directional contraction dynamics, the function of platelet mechanoreceptors, and the influence of biochemical and mechanical cues on platelet contraction. While these existing methods have allowed for understanding of important biophysical properties of platelets, they have been hampered by constraints on the shape or spreading of platelets and/or their low yield, often analyzing fewer than thirty platelets per condition.

Here, the black dots approach was used to characterized forces, cytoskeletal structures, and geometric properties of more than five hundred human platelets for linear mixed-effects modeling and K-means clustering, from which it was identified that platelet size, shape, and cytoskeletal structure have both independent and cooperative contributions to platelet force.

3.2 Materials and methods

Black dots substrates were prepared following the protocols from Chapter 2.1. Due to the small size of platelets, a pattern with a diameter of 850 nm, center-to-center spacing of

2 μm was used. Flexible 13.5 kPa substrates were manufactured using a mixture of 5% Sylgard 184 and 95% Sylgard 527 by weight. The soft PDMS mixture used here resulted in a substrate with a stiffness of 13.5 kPa and was selected because it was physiologically relevant for platelets [16, 18]. Softer and stiffer mixtures of Sylgard 527 and 184 were tested, but they were not optimal for traction force measurements with platelets because the resulting deformations of a subset of platelets were too large or too small to measure accurately (Fig. 3.2). For measurements with other cell types, the ratio of PDMS mixtures can be adjusted to match their level of contractility.

3.2.1 Functionalization of black dots substrates for platelet adhesion

In this work, VWF was chosen as the ECM to facilitate platelet adhesion. On the day of cell seeding, VWF (Haematological Technologies) was diluted in PBS to 5 μg/mL and was pipetted onto the black dots substrates. To encourage droplet spreading over the black dots surface, a glass coverslip was gently placed on top of the droplet. The von Willebrand Factor was incubated for 1-1.5 hours at room temperature before the coverslip was removed. To block the surface, the substrate was then submerged in a 0.2% Pluronic F-127 (BASF) in PBS for 30 minutes. The substrate was finally submerged into PBS and stored until platelet seeding.

To quantify VWF adsorption onto the surface, black dots substrates with and without VWF treatment were blocked with 10% goat serum (Life Technologies, diluted in PBS) for 1 hour and then incubated with a FITC-labeled anti-von Willebrand Factor antibody (Abcam, ab8822) for 1 hour). Substrates were mounted onto glass coverslips using Fluoromount-G mounting medium (Life Technologies) for confocal microscopy. Images collected with the same settings were quantified using MATLAB to characterize FITC fluorescence (and therefore VWF adsorption) on the fluorescent BSA and the non-fluorescent black dots for substrates with and without VWF treatment (Fig. 2.4A-G). VWF adsorption was visualized on the surface and was found to be contiguously adsorbed across the surface, with some preference for binding to the fluorescent BSA.

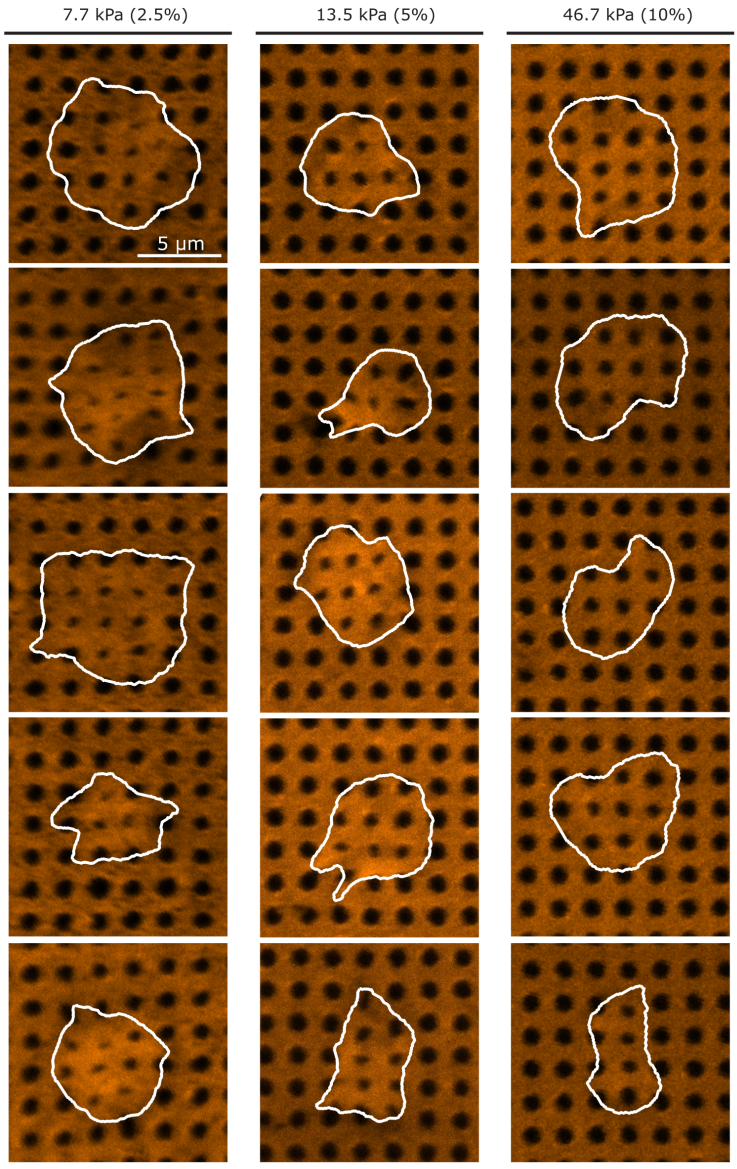


Figure 3.2: Selection of substrate stiffness. Examples of platelets attached to black dots substrates for the three different PDMS mixtures. Percentage indicates amount of Sylgard 184 in the PDMS mixture. Platelets on low stiffness (7.7 kPa) can distort the dots too much for image detection, while platelets on high stiffness (46.7 kPa) sometimes do not distort the dots beyond noise levels. The stiffness of 13.5 kPa was chosen for the platelet studies presented here.

3.2.2 Platelet isolation and seeding

Platelet-rich plasma (PRP) was collected from consenting research participants by plateletpheresis using the Trima Accel automated collection system. Research participants were healthy and not taking any platelet inhibiting medications. Platelets were isolated from plasma by platelet centrifugation washing modified from previously described protocols [42]. Platelets were pelleted at 1000 g and resuspended in HEN Buffer, pH 6.5 containing 10 mM HEPES (Sigma), 1 mM EDTA (Corning), and 150 mM NaCl (Fisher Scientific) and supplemented with 0.5 μm prostacyclin (PGI_2) (Sigma). To prevent activation, platelets were incubated for 10 minutes at room-temperature and then repeat treated with 0.5 μm PGI_2 and pelleted via centrifugation at 800 g. Platelets were resuspended and diluted to 3×10^8 platelets/mL with modified Tyrode's buffer, pH 7.3 containing 5 mM HEPES (Sigma), 137 mM NaCl (Fisher Scientific), 5.5 mM glucose (Fisher Scientific), 12 mM NaHCO3 (Sigma), 0.3 mM NaH_2PO_4 (Sigma), 2 mM KCl (JT Baker), 1 mM $MgCl_2$ (Sigma), and 2 mM $CaCl_2$ (Macron Fine Chemicals) and supplemented with 0.35% (w/v) human serum albumin and 0.02 U/mL apyrase.

Immediately before seeding the washed, isolated platelets onto black dots substrates, the platelets were further diluted to 2.5×10^7/mL in Tyrode's Buffer, pH 7.5 containing 10 mM HEPES (Fisher Scientific), 138 mM NaCl (JT Baker), 5.5 mM glucose (ACROS Organics), 12 mM $NaHCO_3$ (Sigma), 0.36 mM NaH_2PO_4 (Sigma), 2.9 mM KCl (VWR), 0.4 mM $MgCl_2$ (Fisher Scientific), and 0.8 mM $CaCl_2$ (VWR International). After dilution, 10 million platelets were seeded onto each black dot substrate. To allow time for initial platelet binding onto the black dots substrates, platelets were incubated at room-temperature for 10 minutes. To remove unattached platelets, the substrates were then gently dipped in PBS and then immediately submerged in fresh Tyrode Buffer. To allow time for platelet adhesion and contraction on the substrate, the platelets were incubated for an additional 30 minutes at room-temperature. Incubation times were selected to reduce temporal differences in platelet contraction by 1) preventing new platelet binding after 10 minutes, such that all platelets

were on the surface for 30-40 minutes and 2) allowing platelet binding and contraction for 30 minutes so that platelets reach a maximum contraction [37, 43, 50].

3.2.3 Immunocytochemistry

Platelets were fixed with 4% paraformaldehyde for 20 minutes, permeabilized with 0.1% Triton X-100 for 10 minutes at room temperature, and blocked with 10% goat serum (Life Technologies, diluted in PBS) for 1 hour. Platelet F-actin was labeled with phalloidin 488 (Life Technologies), and platelet GPIb was labeled with a CD42b monoclonal antibody, clone SZ2 (Life Technologies) and a goat anti-mouse IgG secondary antibody (Life Technologies). Substrates were mounted onto glass coverslips using Fluoromount-G mounting medium (Life Technologies) for confocal microscopy.

3.2.4 Imaging and image analysis

Fixed and stained platelets were imaged on a Nikon A1R or a Leica SP8 confocal microscope with a 60x oil objective (NA = 1.4). Images of platelets were taken with a large enough field of view to ensure several dots with no deformation surrounded each platelet. To quantify the deformation of the dots, an automated particle tracking approach was used (Chapter 2.2) [9]. The fluorescent image of the black dots pattern was first run through a spatial bandpass filter with a characteristic noise length scale of 1 pixel. The dots were identified with a peak finding algorithm using a peak threshold value of 0.15. For each dot, the centroid was then found to subpixel accuracy.

3.2.5 Force calculation

Traction forces are calculated from the surface displacements using regularized Fourier Transform Traction Cytometry (FTTC) described in Chapter 2.3 [36, 84]. FTTC requires a rectangular grid of displacements which is obtained trivially by the black dots pattern without any interpolation. Any missing data locations near the periphery or corners of the image

are filled in with imaginary data points assigned with zero displacement. A regularization parameter of 5×10^{-8} was used to smooth out noise in the traction forces.

To assess whole-cell contractility, total force and net force are calculated for each cell. Total force is calculated by summing the force magnitudes from each dot underneath the cell. Net force is similarly calculated by simply adding the force vectors from each dot together; the cell is assumed to be in static equilibrium, so the net force should tend towards 0. Because the cell can adhere to spaces between black dots, all black dots within the cell boundary and within 1 dot spacing outside the cell boundary are considered for these calculations.

3.2.6 Area and circularity calculations

The cell boundary and area were determined in MATLAB using a user-adjusted threshold and shape fill on the fluorescent F-actin image. Cell boundary was also determined using the GPIb fluorescent image and no difference in cell boundary was observed. Circularity was calculated from the cell boundary using: $C = 4\pi A/P^2$, where C is the circularity, A is the area, and P is the perimeter [73]. Circularity values can range from 0 to 1, where a value of 1 indicates a perfect circle.

3.2.7 F-actin dispersion calculation

For each cell, the stained F-actin image was normalized such that the fluorescent intensity within the cell boundary spanned values 0 to 1. A threshold was set at 0.1, and the F-actin dispersion was calculated as the percentage of cell pixels above this threshold. Based on this calculation, a cell with well-dispersed or uniform F-actin stain will receive a value close to 1, while a cell with localized F-actin intensity will receive a value close to 0. The threshold value of 0.1 was chosen because it resulted in both the largest spread of F-actin dispersion between cells and yielded a quantification most consistent with qualitative measures.

3.2.8 K-means clustering

A K-means clustering analysis was utilized to separate the data into clusters in an unbiased way. First, the area, circularity, and F-actin dispersion data were normalized. The built-in MATLAB function "kmeans" was used to cluster the data based on the area, circularity, and F-actin dispersion. Cell force was not included for purposes of clustering. To determine the optimal number of clusters, the built-in MATLAB function "evalclusters" was employed for up to 6 clusters using either Silhouette or Gap evaluation criteria with default settings. The optimal number of clusters is defined as the one with the highest Silhouette value or as the lowest number of clusters such that the mean Gap value for the next highest number of clusters falls within the standard error of the previous one. Silhouette and Gap values were evaluated for up to 6 clusters, and both criteria suggest that 2 clusters is optimal for this data set (Fig. 3.14).

3.2.9 Statistics

To compare donor to donor variability, a one-way ANOVA and Tukey's post-hoc test was used to determine whether differences in the means between donors were statistically significantly significant.

To examine effects of area, circularity, and F-actin dispersion on force, each covariate was centered at its mean and circularity and F-actin dispersion were transformed to a 0-100 scale. This transformation does not affect the results. A multivariate mixed effects model with random donor effects was used to analyze the centered data and determine the influence of individual covariates and interactions between covariates.

To determine if forces of platelets in cluster 1 and cluster 2 (as determined by the K-means clustering analysis) were significantly different from each other, a student's t-test was used. For all tests, significance is considered $p < 0.05$.

3.2.10 Cell exclusion considerations

To reduce systematic error in the data, several conditions for excluding cells from the analysis were considered. The analysis requires all black dots near the edge of the cell field of view to be undeformed; any cells within close proximity of each other will disrupt this requirement. Therefore, cells which are close in proximity to other cells are automatically disregarded from all analysis; close proximity is defined here as two neighboring cell boundaries coming within 2 μm of each other. Another exclusion criterion are high net forces which are indicative of irregular patterning or mounting issues. A cutoff of 10 nN was chosen here and all cells exhibiting net forces higher than this cutoff were excluded. Additionally, platelets that did not spread were removed from analysis by excluding platelets < 10 μm^2 that had no filopodial or lamellipodial protrusions. Finally, fluorescence from F-actin staining was observed to be highly variable in some cells; the exposure time was tuned but for some cells it was difficult to completely eliminate image saturation. Therefore, cells that had greater than 1% saturated pixels within the cell boundary were excluded from the analysis shown in Figures 3.6-3.13.

3.2.11 Technique sensitivity

The sensitivity of the black dots technique was investigated by measuring the effect of dot spacing on force estimation. Different pattern spacing was simulated by skipping rows and columns from the existing displacement measurements (Fig. 3.3A). For example, the pattern used in this study had a nominal dot spacing of 2 μm. By using only the displacements from every other dot, we effectively simulate a dot spacing of 4 μm. Using every third or every fourth dot, a spacing of 6 μm or 8 μm is achieved. The average number of dots within each platelet was 24.9, 6.2, 2.8, and 1.5 for dot spacings of 2, 4, 6, and 8 μm, respectively. For each of these simulated dot spacings, force was estimated with FTTC and total force was calculated for all platelets in this study (Fig. 3.3B). No significant difference was detected for estimated platelet force between 2, 4, and 6 μm. Force estimated from a dot spacing of 8 μm was significantly lower than the other tested dot spacings ($p < 0.05$ when tested with

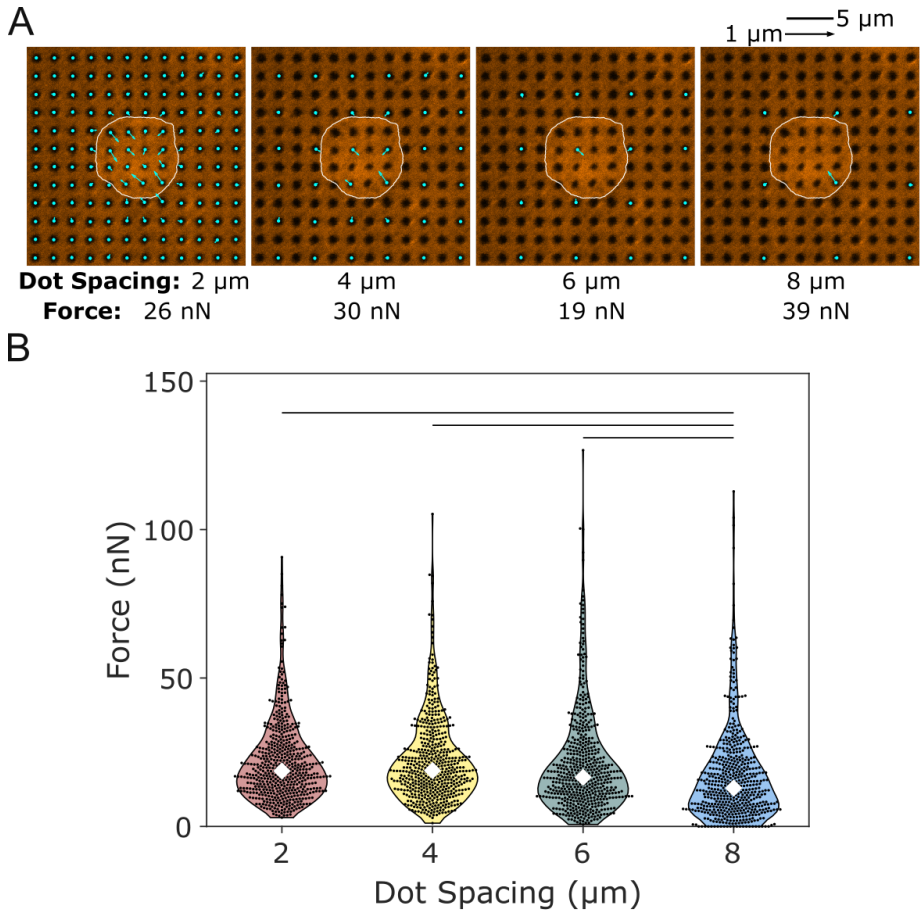


Figure 3.3: Effect of pattern size on force estimation. (A) Example of a platelet with displacement vectors with different pattern spacing simulated by skipping dots. For example, using data from every other dot simulates a pattern spacing of approximately 4 μm. Total force estimated for each pattern spacing is shown underneath each image. (B) Forces from all platelets for different dot spacing, n = 540. For dot spacing of 8 μm, several platelets exhibit no force likely because the dot spacing is too large. Lines indicate significant differences in forces ($p < 0.05$ when tested with paired t-tests).

paired t-tests). A dot spacing of 8 μm is likely too large for these cells as indicated by the large number of cells exhibiting no force; these cells had no dots beneath them to measure displacement or force. Overall, this data suggests that the 2 μm spacing used in this study is sufficiently small enough to accurately estimate force. While force on the 6 μm spacing is not significantly different than on the 2 μm spacing, the higher resolution in displacement measurements offered by lower spacing is important for ensuring each platelet has a sufficient number of dots to measure displacement.

3.3 Results and Discussion

3.3.1 Reference-free traction force microscopy with the black dots platform

To demonstrate the black dots approach, traction forces of human platelets were measured. Washed platelets were seeded onto VWF-coated black dots substrates for 10 minutes to allow them to adhere and then rinsed gently to remove unbound platelets. The platelets were allowed to to spread and contract for 30 minutes before fixing the samples. This timing for platelet binding and contraction was selected based on dynamics of platelet force generation [37, 43, 66, 70]. With immunofluorescence staining and confocal microscopy, many platelets can be captured in a single image (Fig. 3.4A and Fig. 3.5). The platelets had various shapes and sizes similar to previous observations on glass substrates [54, 70]. Platelets were also seeded onto black dots substrates without VWF where they did not bind and spread, demonstrating that the platelet adhesion is VWF-specific.

Individual platelets within an image are cropped and analyzed separately (Fig. 3.4B). The centroid of each black dot is identified using automated detection (Fig. 3.4C). Black dots at the edge of the region are used to form a grid of best-fit lines whose intersections denote the undeformed position of each black dot (Fig. 3.4D). For each black dot, the distance from its centroid to its respective intersection in the zero-displacement grid (Fig. 2D inset) is used to calculate the magnitude and direction of the forces using regularized Fourier Transform Traction Cytometry (FTTC) (Fig. 3.4E) [36, 84]. The black dots technique is suited well for

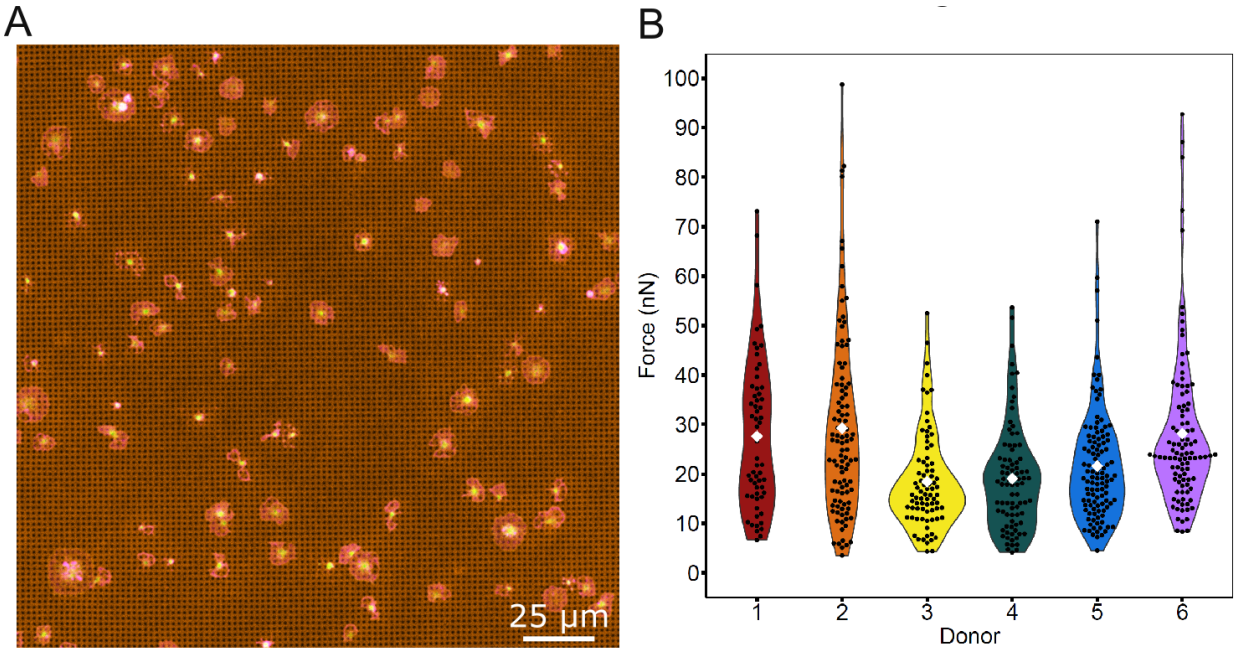


Figure 3.4: The black dots platform offers a higher yield way to measure forces. (A) Example of a field of view containing many platelets adhered to and contracting on a black dots substrate. Here, platelets are stained for both F-actin (green) and GPIb (magenta). (B) Forces from at least 100 platelets from 6 donors show high variability within each donor and between donors. Lines indicate significant differences in donor forces ($p < 0.05$ when tested with a one-way ANOVA and Tukey's post-hoc test). Number of cells analyzed for donors 1, 2, 3, 4, 5, and 6 are n = 111, 117, 100, 120, 112, and 100, respectively.

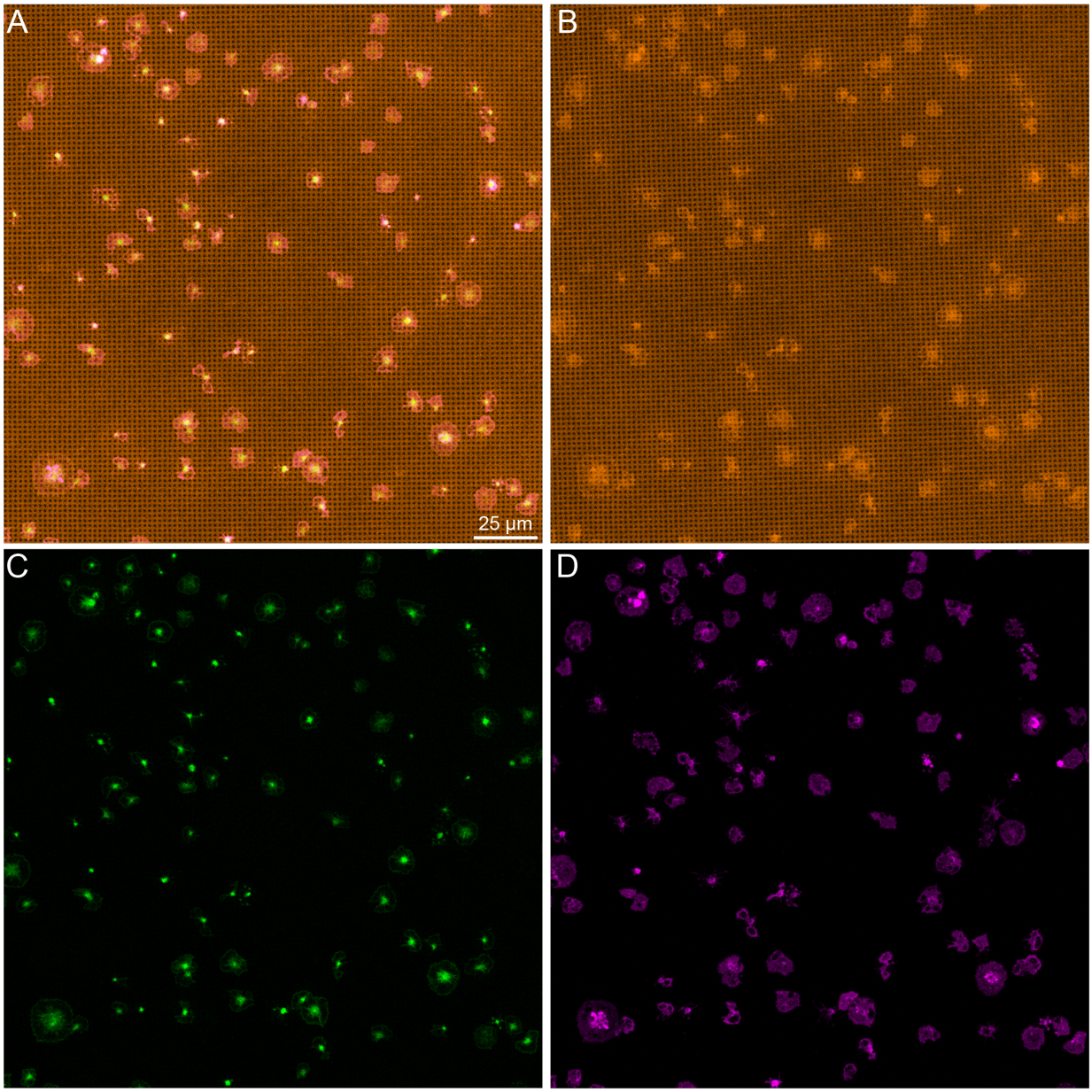


Figure 3.5: Example of a field of view containing many platelets adhered to and contracting on a black dots substrate. This figure shows the same image as Figure 3.4A, but with each channel shown separately. Here, (A) is an overlay of all channels, (B) is the black dots pattern in orange, (C) is F-actin in green, and (D) is GPIb in magenta.

FTTC, which requires the measured displacements to be on a regular grid. The total force for each platelet is calculated by summing the force magnitudes of each dot under the cell. All data plotted in this work is the total force of single platelets.

The total contractile forces of platelets from six healthy donors were analyzed (Fig. 3.4F). The mean force measured by the black dots approach was 24.1 nN, which is similar to other methods that have reported forces between 19 and 200 nN for individual platelets [29, 37, 50, 66]. The platelets exhibited a wide range of forces, from 3.5 to 98.7 nN (28-fold difference) with a standard deviation of 14.7 nN. This observation agrees with other studies; atomic force microscopy found that platelet forces varied from 1.5 to 79 nN [50] (more than 50-fold difference), and subsequent work using TFM and nanoposts have also observed heterogeneity in platelet forces [29, 37]. Heterogeneity was observed both within and between donors, including a standard deviation of 13.7 nN among platelets from the same donor as well as statistically significant differences between mean platelet force from different donors (lines in Fig. 3.4F). These results show that platelet forces measured with the black dots approach are similar in magnitude to previous measurements and indicate that populations of platelets produce a wide range of forces.

3.3.2 Platelet forces correlate with spread area, circularity, and F-actin dispersion

It is not clear whether the heterogeneity in total platelet forces could be attributed to their size, shape, and/or cytoskeletal structure. Platelets typically bind to a surface, increase their spread area approximately 5-fold over about 10 minutes, and then sustain their maximum spread area [37, 70, 93]. In experiments presented here, platelets were allowed to adhere and spread for 30-40 minutes after attachment to allow them to reach their maximum area. Spread area was examined as a factor influencing the overall magnitude of traction forces in platelets as it has been observed previously [37, 38] as well as in many other cell types [14, 90, 94]. The spread area of platelets ranged from 8.7 to 205.5 μm^2, with a mean and standard deviation of 43.5 $\pm$ 22.4 μm^2. A positive relationship between force and area was observed, having a best-fit slope of 0.53 nN/μm^2 ($R^2 = 0.49$) (Fig. 3.6A-C). This force-area

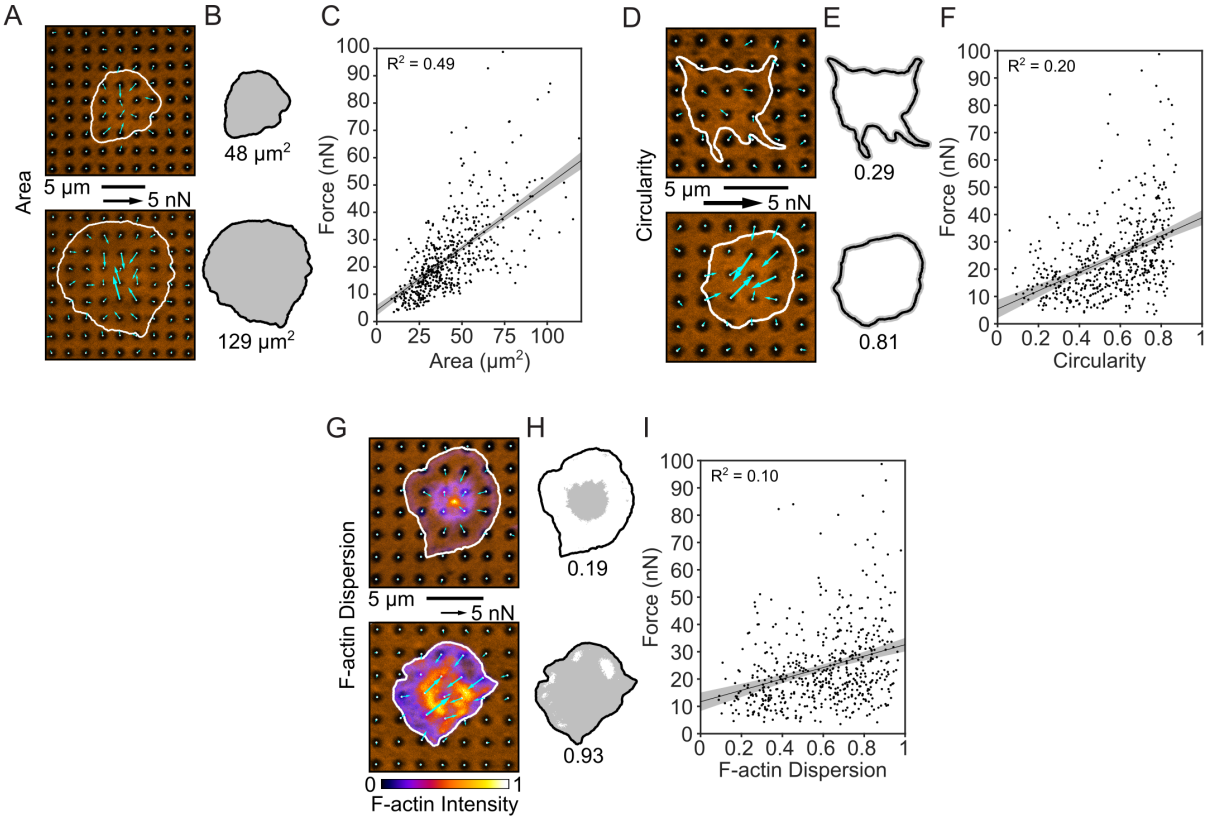


Figure 3.6: Platelet size, shape, and structure correlate with force. (A) Two examples of platelets with small and large areas. (B) Cell boundary and area measured from (A). (C) Platelet forces and area are linearly related. Note that in this panel, the x-axis maximum is zoomed to better view the data. Due to this axis zoom, two points (0.37% of the data) are not shown, but all points are included within all analyses (including the fit line calculation). (D) Two examples of platelets with low and high circularity. (E) Cell boundary and circularity measured from (D). (F) Platelet force and circularity show a moderate positive relationship. (G) Two examples of platelets with low and high F-actin dispersion. Color bar indicates fluorescence intensity which has been normalized to calculate F-actin dispersion. (H) Cell boundary and F-actin dispersion measured from (G). (I) F-actin dispersion is moderately positively correlated with platelet force. Shaded regions of fit lines indicate 95% prediction interval for the data.

relationship is maintained in all six donors, with some minor differences between them (Fig. 3.7). While these results indicate a strong force-area relationship in platelets, there is a degree of heterogeneity in the data. For example, platelets with a spread area of 50-55 μm^2 exerted forces from 14.3 to 71.0 nN, with a mean and standard deviation of 31.8 $\pm$ 13.3 nN. Although spread area has a strong correlation with platelet forces, it does not fully account for their contractile output.

Another aspect to consider is the dramatic shape change of platelets such as their transition from discoid to spherical shape upon activation and their extension of filopodial protrusions in the early stages of platelet spreading [4, 93]. Because these shape changes are important to platelet function, platelet shape and its correlation to force was investigated. Adherent platelets on the black dots substrates adopt a variety of different shapes, ranging from stellate to circular. This is quantitatively assessed for these different shapes using the circularity analysis metric (Fig. 3.6D-E). The more circular platelets generate larger forces than ones that are stellate and less circular (Fig. 3.6F). However, the best-fit slope of this relationship is 0.33 nN/0.01 circularity units (R^2 = 0.20) so it is not as correlative as the force-area relationship. All six donors showed similar force-circularity behavior (Fig. 3.8). These results indicate that circularity has a moderate correlation with force.

Due to the underlying role of actin remodeling in initiating platelet shape changes and generating cellular forces [4, 93], the arrangement of actin was quantified and compared with platelet forces. When the platelets were stained to view their F-actin network, a cortical actin ring was observed around the cell boundary of most platelets. However, there were some distinct differences in the F-actin structure in their interior, ranging from punctate to dispersed (Fig. 3.6G-H). The amount of F-actin dispersion was quantified and plotted against force. Platelets with more dispersed F-actin structure typically generated higher forces and the best-fit slope of this relationship is 0.21 nN/0.01 F-actin dispersion units (R^2 = 0.10) (Fig. 3.6I). All six donors exhibited a similar force-F-actin dispersion relationship (Fig. 3.9). Taken all together, area has a strong correlation with force while circularity and F-actin dispersion moderately correlate with force.

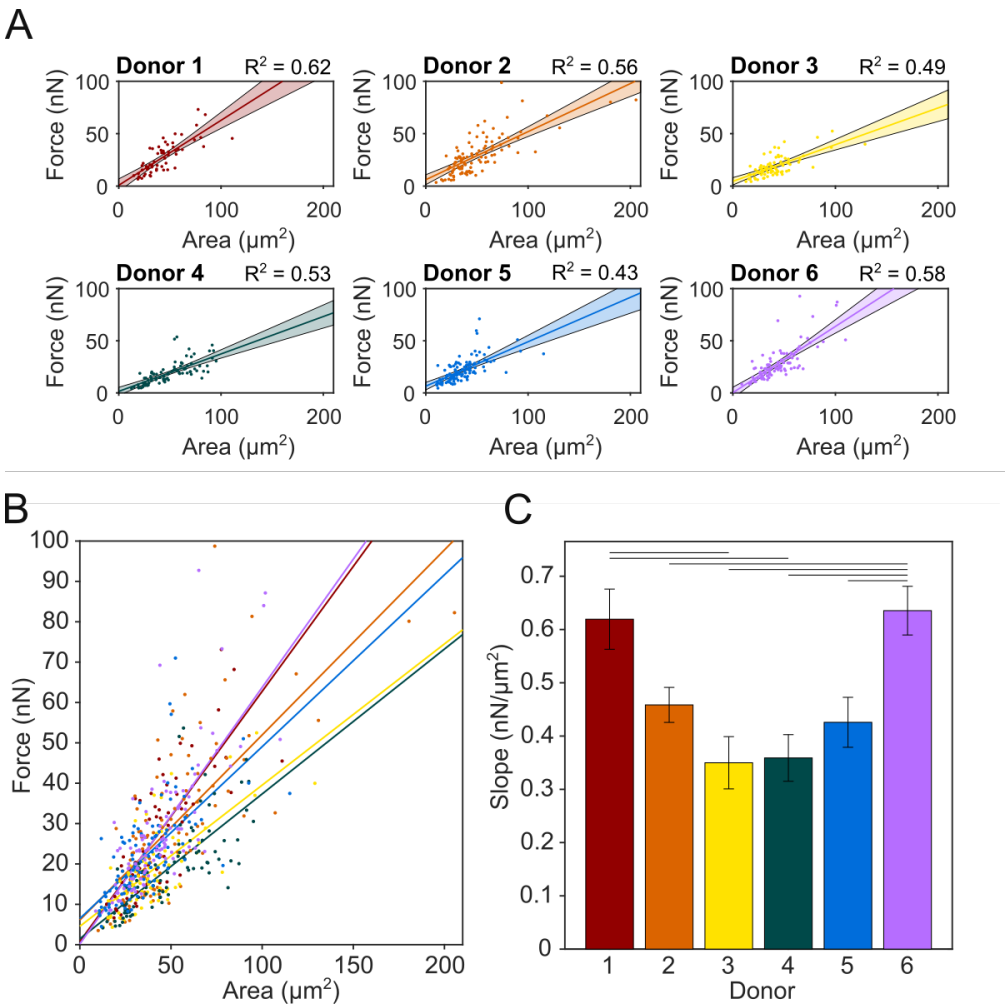


Figure 3.7: Force-area relationship by donor. (A) Force-area data plotted separately for each donor. Solid fit lines are displayed with shaded area that indicates 95% prediction interval for the fits. (B) Overlay of force-area data from all 6 donors with linear fit lines for each donor. (C) Slopes for the fit lines for each donor. Horizontal bars at the top indicate significance ($p < 0.05$ when tested with a one-way ANOCOVA and Tukey's post-hoc test).

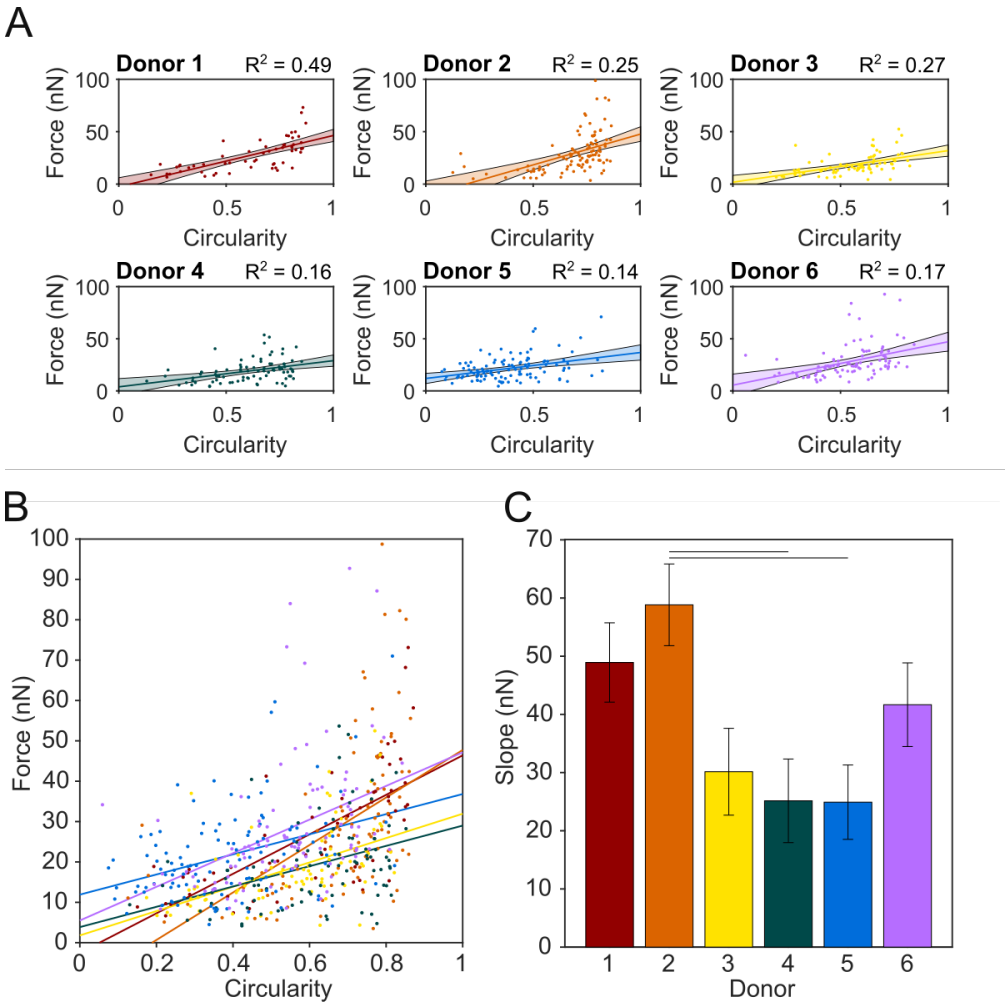


Figure 3.8: Force-circularity relationship by donor. (A) Force-circularity data plotted separately for each donor. Solid fit lines are displayed with shaded area that indicates 95% prediction interval for the fits. (B) Overlay of force-circularity data from all 6 donors with linear fit lines for each donor. (C) Slopes for the fit lines for each donor. Horizontal bars at the top indicate significance ($p < 0.05$ when tested with a one-way ANOCOVA and Tukey's post-hoc test).

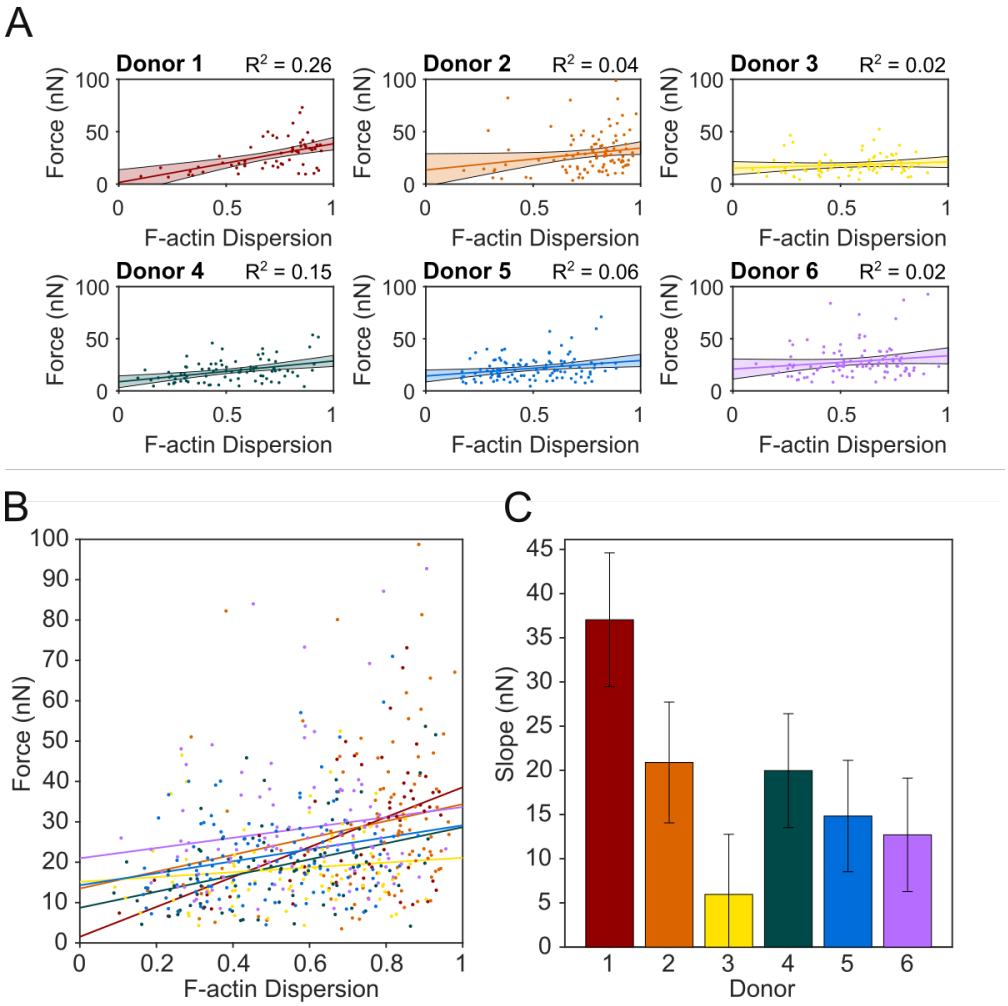


Figure 3.9: Force-F-actin dispersion relationship by donor. (A) Force-area data plotted separately for each donor. Solid fit lines are displayed with shaded area that indicates 95% prediction interval for the fits. (B) Overlay of force-area data from all 6 donors with linear fit lines for each donor. (C) Slopes for the fit lines for each donor. No significance was detected for the slopes between donors when tested with a one-way ANOCOVA and Tukey's post-hoc test.

3.3.3 Multivariate mixed effects modeling reveals cooperative effects between F-actin dispersion and circularity and between F-actin dispersion and spread area

Because platelet size, shape, and cytoskeletal organization change concurrently during platelet adhesion [93], correlations between area, circularity, F-actin dispersion were also investigated (Fig. 3.10A, D, H). For area and circularity, a moderate correlation was observed ($R^2 = 0.26$) (Fig. 3.10A). To visualize the combined relationship of circularity and area with increasing force, platelets were split into four equally sized groups by force (i.e., quartiles). Notably, low-force platelets had small areas, but also had a wide range of circularities. On the other hand, high-force platelets almost exclusively had high area and high circularity (Fig. 3.10B and Fig. 3.11 to see graphs with all points). By plotting the median of each quartile, circularity and area are shown to increase together with increasing force (Fig. 3.10C). Similarly, area and F-actin dispersion (Fig. 3.10D-F) as well as F-actin dispersion and circularity (Fig.3.10G-I) increase together with each force quartile. The particularly extreme shift observed in the circularity versus F-actin dispersion contour plots is somewhat surprising, given that each of these factors only moderately correlate with force. These results indicate that there are some correlations between platelet area, circularity, and F-actin dispersion and that together, they have strong effects on force.

To further investigate how F-actin dispersion, circularity, and area affect force in different but overlapping ways, R Studio was used to create a multivariate mixed effects model allowing for 2-way interactions between each of the factors (Table 3.1). The mixed effects model shows that across donors, the difference in force between two platelets that differ in area by 1 μm^2 (while other factors remain constant) is 0.41 nN (95% CI: 0.37, 0.45) on average, with the larger platelet generating more force (Table 3.1). Similarly, when holding other factors constant, two platelets that differ in circularity or F-actin dispersion by 0.01 will respectively differ in force by 0.069 nN (95% CI: 0.02, 0.11) and 0.15 nN (95% CI: 0.11, 0.19), on average. From estimates and standard errors in Table 3.1, p-values are calculated to determine what factors and interactions have a significant ($p < 0.05$) effect on force. All

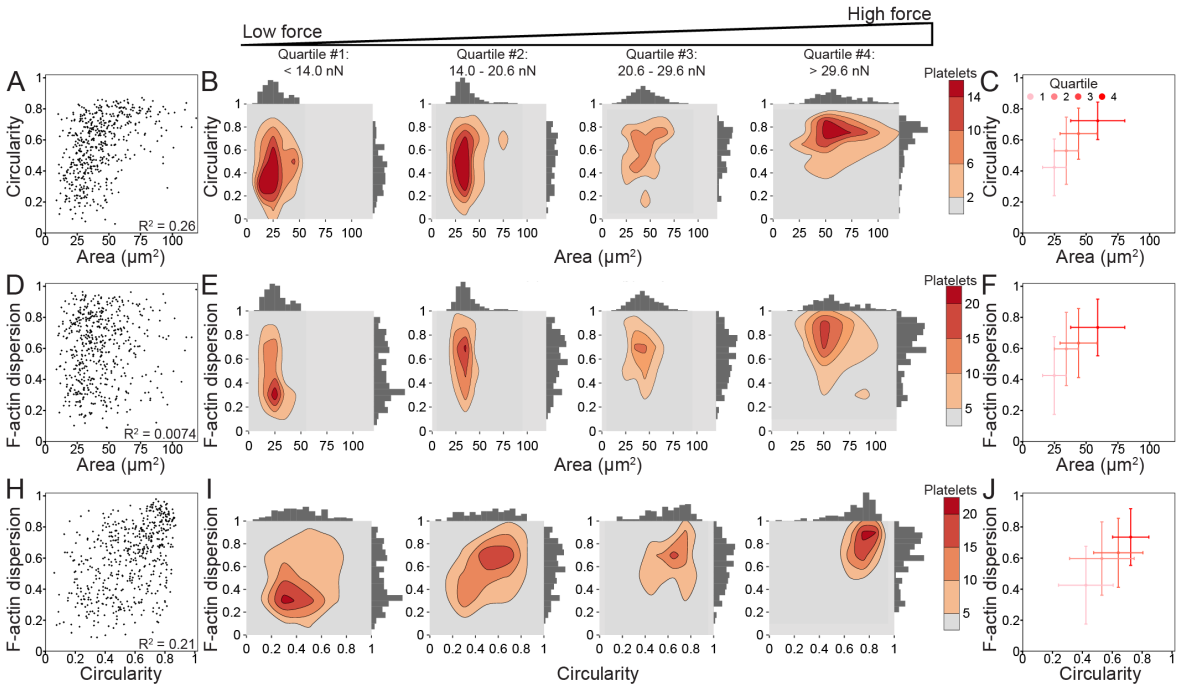


Figure 3.10: Platelet size, shape, and structure do not strongly correlate with each other, but increase together with force. (A) Area and circularity moderately correlate ($R^2 = 0.26$) when plotting platelets of all forces (n = 540). (B) To examine the relationship of circularity and area with increasing force, platelets are split into four quartiles (n = 135 in each quartile) by force, where the lowest force quartile (quartile #1) includes platelets generating less than 14.0 nN force, quartile #2 contains platelets generating 14.0-20.6 nN force, quartile #3 contains platelets generating 20.6-29.6 nN force, and the highest force quartile (quartile #4) contains platelets generating greater than 29.6 nN force. Contour density plots and histograms of area and circularity at each force quartile show that quartile #1 (low-force platelets) have low area and a large range of circularity, while quartile #4 (high-force platelets) tend to have both higher area and high circularity. (C) The median and median standard deviation show that circularity and area increase together with each force quartile. (D) F-actin dispersion and area do not correlate ($R^2 = 0.0074$), (E-F) but show a similar trend when examining force quartiles. (H) F-actin dispersion and circularity moderately correlate ($R^2 = 0.21$) and (I-J) show a shift from low-force platelets having large ranges of circularity and F-actin dispersion to high-force platelets have high circularity and high F-actin dispersion. Note that in A-F, the x-axis maximum is zoomed to better view the data. Due to this axis zoom, two points (0.37% of the data) are not shown, but all points are included within all analyses.

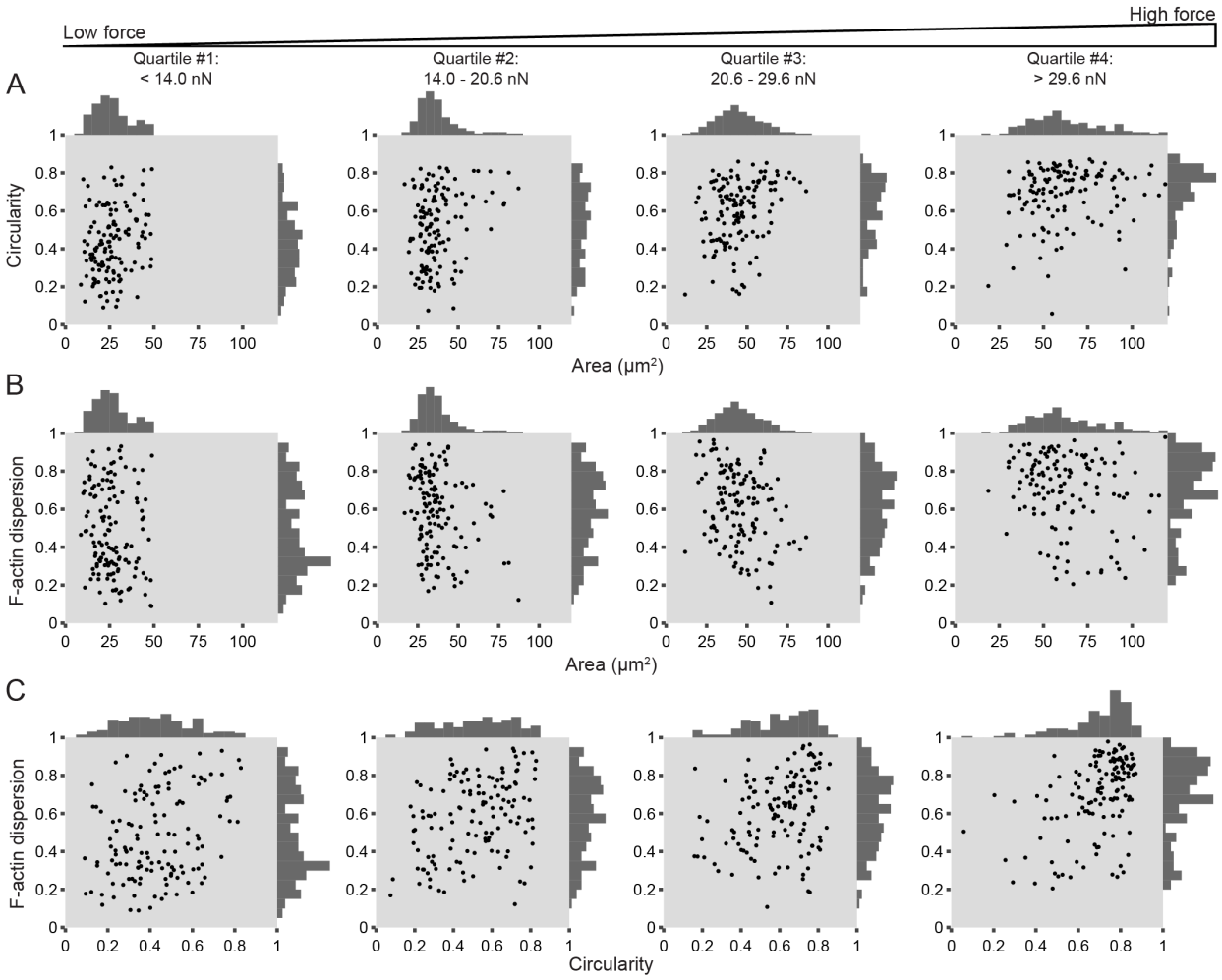


Figure 3.11: Platelet size, shape, and structure do not strongly correlate with each other, but increase together with force. This figure shows the same data as Figure 3.10B, E, and I, but with a scatter plot instead of a contour plot to display all points of data.

	Estimate	**Std. Error**	**p-value**
Area [nN/μm^2]	0.4100	0.0209	< 0.001
Circularity [nN/0.01 units]	0.0687	0.0247	0.0053
F-actin dispersion [nN/0.01 units]	0.1488	0.0187	< 0.001
Circularity : F-actin dispersion [nN/(0.01 units*0.01 units)]	0.0024	0.0010	0.0134
Area : F-actin dispersion [nN/(μm^2 *0.01 units)]	0.0024	0.0008	0.0028
Area : Circularity [nN/(μm^2 *0.01 units)]	-0.0012	0.0009	0.2079

Table 3.1: Multivariate mixed effects model summary shows significant interaction effects. The estimate column indicates the expected increase in force (in nN) if the variable in that row increases by 1 μm^2 (area) and/or 0.01 units of circularity or F-actin dispersion while all other factors remain constant. The standard error column indicates the error of that estimate. From these estimates and standard errors, p-values are calculated to determine what factors and interactions have a significant ($p < 0.05$) effect on force. In addition to these individual effects, pairs of effects were tested for interactions, where a positive value in the estimate column indicates a cooperative effect and a negative value an antagonistic effect.

individual factors (area, circularity, and F-actin dispersion) significantly contribute to force when controlling for the other factors. In addition to these main effects, two interaction terms were significant at the $p < 0.05$ level: F-actin dispersion interacting with area and F-actin dispersion interacting with circularity, each of which is a positive, cooperative effect. For example, when circularity is average, F-actin dispersion and force have a positive relationship with a slope of 0.069 nN/0.01 F-actin dispersion units.

Figure 3.12 shows the interaction plots that are output from the multivariate mixed effects model. They demonstrate the relationship between expected force (y-axis) and the x-axis variable change when another variable is average (green line), high (blue line), or low (red line). Slope differences in these green, blue, and red lines indicate how one variable changing alters the relationship between force and the x-axis variable. For example, when circularity is one standard deviation above average, the relationship between F-actin dispersion and circularity is stronger and has a slope of 0.12 nN/0.01 F-actin dispersion units (75% increase). Conversely, when circularity is one standard deviation below average, the relationship between F-actin dispersion and circularity is weaker and has a slope of 0.017 nN/0.01 F-actin dispersion units (75% decrease) (Fig. 3.12E for this for all other interaction plots). Area interacting with circularity has a p-value of 0.2079 and is not significant at the $p < 0.05$ level. All interaction plots are described in detail here:

Fig. 3.12A: Force-area for average circularity For average circularity (circularity = 0.557, green line), the relationship between expected force and area is positive with a slope of 0.410 nN/μm^2. If circularity is one standard deviation above average (circularity = 0.754, blue line), the force-area relationship has a slope of 0.387 nN/μm^2. If circularity is one standard deviation below average (circularity = 0.360, red line), the force-area relationship has a slope of 0.433 nN/μm^2. The interaction between area and circularity is not significant ($p = 0.2128$).

Fig. 3.12B: Force-circularity for average area For average area (area = 43.6 μm^2, green line), the relationship between expected force and circularity is positive with a slope of

6.87 nN/1 circularity unit. If area is one standard deviation above average (area = 66.0 μm^2, blue line), the force-circularity relationship has a slope of 4.25 nN/1 circularity unit. If area is one standard deviation below average (area = 21.1 μm^2, red line), the force-circularity relationship has a slope of 9.50 nN/1 circularity unit. The interaction between area and circularity is not significant (p = 0.2128).

Fig. 3.12C: Force-area for average F-actin dispersion For average F-actin dispersion (F-actin dispersion = 0.592, blue line), the relationship between expected force and area is positive with a slope of 0.410 nN/μm^2. If F-actin dispersion is one standard deviation above average (F-actin dispersion = 0.811, blue line), the force-area relationship has a slope of 0.464 nN/μm^2. If F-actin dispersion is one standard deviation below average (circularity = 0.372, red line), the force-area relationship has a slope of 0.356 nN/μm^2. The interaction between area and F-actin dispersion is significant (p = 0.0029).

Fig. 3.12D: Force-F-actin dispersion for average area For average area (area = 43.6 μm^2, green line), the relationship between expected force and F-actin dispersion is positive with a slope of 14.9 nN/1 F-actin dispersion unit (Fig. 3.12D). If area is one standard deviation above average (area = 66.0 μm^2, blue line), the force-F-actin dispersion relationship has a slope of 20.4 nN/1 circularity unit. If area is one standard deviation below average (area = 21.1 μm^2, red line), the force-F-actin dispersion relationship has a slope of 9.4 nN/1 F-actin dispersion unit. This interaction between area and F-actin dispersion is significant (p = 0.0029).

Fig. 3.12E: Force-circularity for average F-actin dispersion For average F-actin dispersion (F-actin dispersion = 0.592, blue line), the relationship between expected force and circularity is positive with a slope of 6.87 nN/1 circularity unit. If F-actin dispersion is one standard deviation above average (F-actin dispersion = 0.811, blue line), the force-circularity relationship has a slope of 12.1 nN/1 circularity unit. If F-actin dispersion is one standard deviation below average (circularity = 0.372, red line), the

force-circularity relationship has a slope of 1.7 nN/1 circularity unit. The interaction between circularity and F-actin dispersion is significant (p = 0.0150).

Fig. 3.12F: Force-F-actin dispersion for average circularity For average circularity (circularity = 0.557, green line), the relationship between expected force and F-actin dispersion is positive with a slope of 14.9 nN/1 F-actin dispersion unit. If circularity is one standard deviation above average (circularity = 0.754, blue line), the force-F-actin dispersion relationship has a slope of 19.5 nN/1 unity of F-actin dispersion. If circularity is one standard deviation below average (circularity = 0.360, red line), the force-F-actin dispersion relationship has a slope of 10.2 nN/1 unit of F-actin dispersion. The interaction between circularity and F-actin dispersion is significant (p = 0.0150).

This multivariate mixed effects model supports the contribution of area, circularity, and F-actin dispersion to force and suggests a complex relationship between these factors. Additionally, this analysis reveals significant cooperative effects between F-actin dispersion and circularity and between F-actin dispersion and area.

3.3.4 Unbiased clustering supports relationship between spread area, circularity, F-actin dispersion, and platelet force

Big data analyses are powerful tools that can help extract significant information in data sets that are large and unwieldy. In the population of platelets here, a large range of shapes, sizes, and structures were observed, so the possibility of clusters or subpopulations of platelets was investigated in this data set. An unbiased K-means clustering analysis was performed on platelet area, circularity, and F-actin dispersion to locate possible clusters, and to see if the relationships observed between these factors and force could be explained by distinct clusters or subpopulations of platelets. Two clusters arose from this analysis: cluster 1 is generally characterized by low spread area, circularity, and F-actin dispersion while cluster 2 is high spread area, circularity, and F-actin dispersion (Fig. 3.13A-D and Fig. 3.14A-D). The clustering analysis was also performed on each donor individually; each donor generally

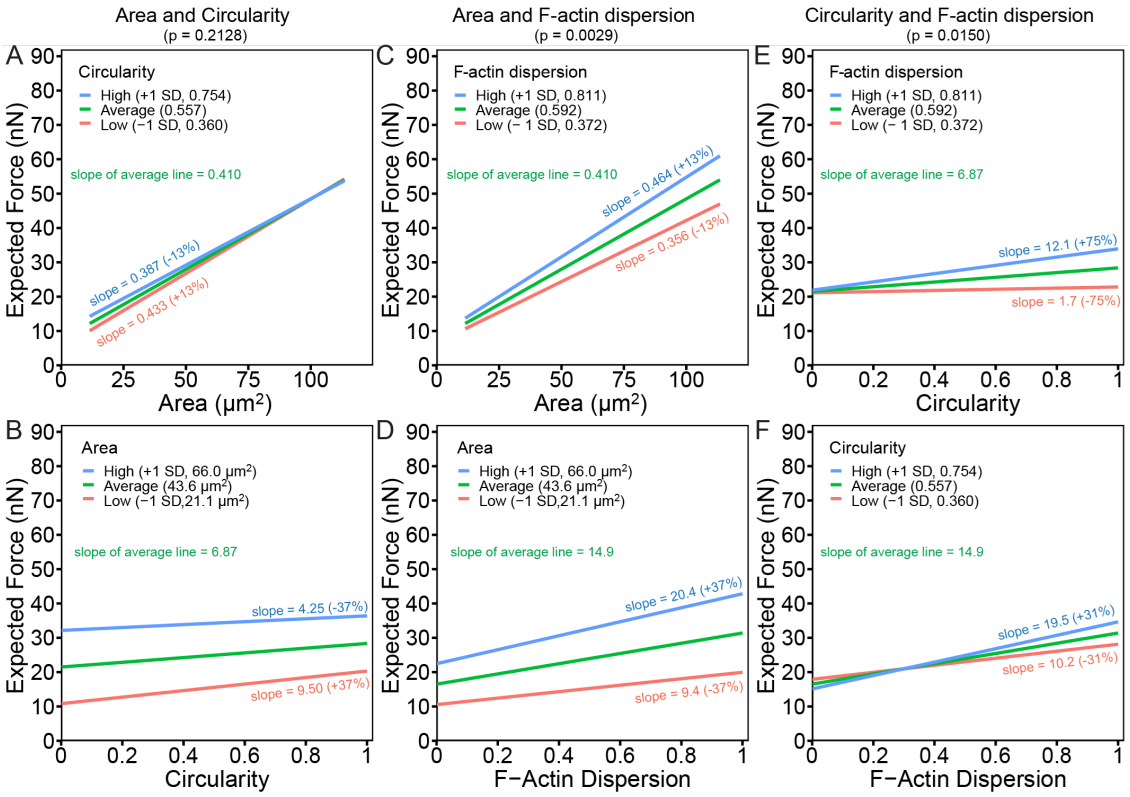


Figure 3.12: Interaction plots demonstrate cooperative interactions between F-actin dispersion and circularity as well as F-actin dispersion and area. These plots are output from the multivariate mixed effects model; they demonstrate the relationship between expected force (y-axis) and the x-axis variable change when another variable is average (green line), high (blue line), or low (red line). Slope differences in these green, blue, and red lines indicate how one variable changing alters the relationship between force and the x-axis variable. These plots detail the relationships between expected force and (A) area for average circularity, (B) circularity for average area, (C) area for average F-actin dispersion, (D) F-actin dispersion for average area, (E) circularity for average F-actin dispersion, and (F) F-actin dispersion for average circularity.

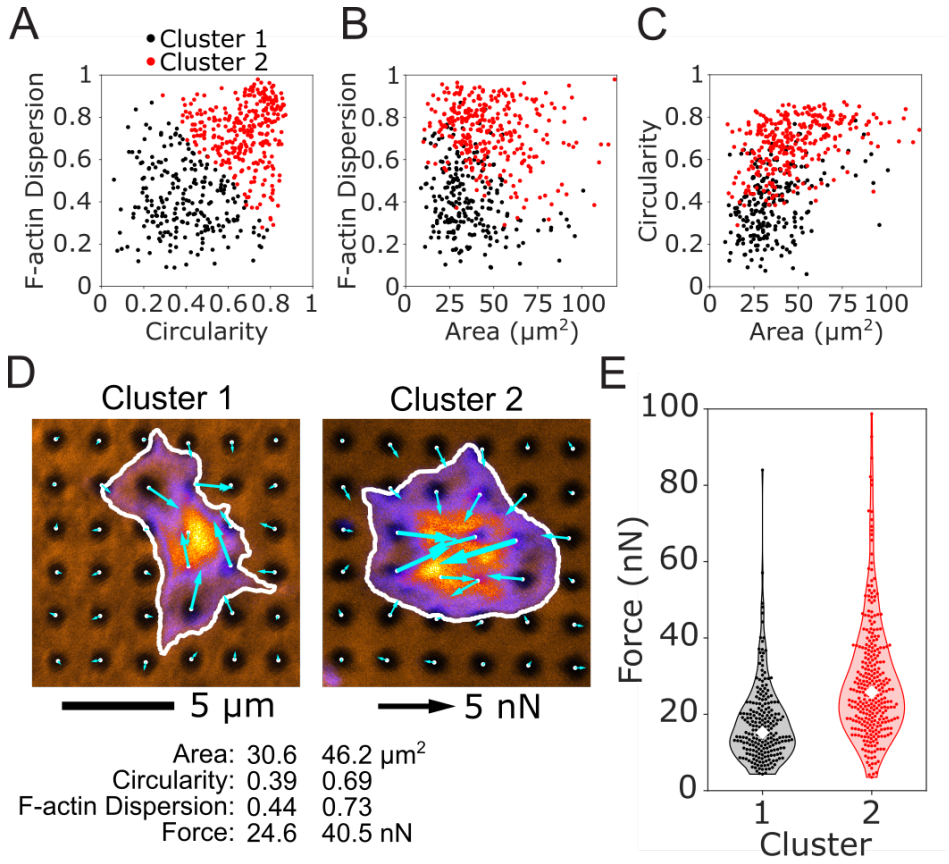


Figure 3.13: K-means clustering of platelet size, shape, and structure predict differences in force. An unbiased K-means clustering approach based on platelet area, circularity, and F-actin dispersion separated the population of platelets into 2 clusters. The two clusters are shown for (A) F-actin dispersion and circularity, (B) F-actin dispersion and area, and (C) circularity and area. (D) The most representative platelet from each cluster is shown. Platelets from cluster 1 have smaller area, lower circularity, and lower F-actin dispersion than cluster 2. (E) Forces from cluster 2 are significantly higher than cluster 1, even though force was not used to determine the clusters.

formed two clusters that were similar to the two clusters in the overall data set (Fig. 3.14E-J). For this clustering analysis, the force data was intentionally not included; despite this agnostic approach to platelet force, cluster 2 has significantly higher forces than cluster 1 using a Student's t-test (Fig. 3.13E), supporting earlier findings.

K-means clustering was chosen here due to its simplicity and widespread use, although other clustering methods may be more appropriate depending on the data set. By eye, the two clusters in this data set tend to lie on a continuum rather than distinct clusters with no overlap. This could indicate that the two clusters do not originate from distinct sources, but instead emerge from a single gradient such as platelet age in circulation, where older platelets tend to have less spread area, less circularity, and less dispersed F-actin. The physiological origin and significance of these clusters will be further investigated in future studies. Ultimately, this clustering analysis serves as a demonstration of big data analyses that are made possible by data from hundreds of cells collected with a high-yield method.

3.4 Concluding remarks

The black dots technique was used here to characterize the relationship of force with platelet size, shape, and cytoskeletal structure. Forces of more than 500 platelets were measured, which is five times more than previous studies [37, 38, 43] and is on par with existing high-yield methods that directly control cell shape and area [66]. The magnitude of forces from the black dots technique is similar to other methods that have reported forces for individual platelets [29, 37, 50, 66]. For the first time, platelet forces were correlated to platelet circularity and F-actin dispersion. This was only possible with the black dots technique because it does not constrain platelet shape or size and is compatible with the immunofluorescent techniques necessary to study the cytoskeleton. Significant associations were found between spread area, circularity, F-actin dispersion, and force, as well as interactions between these factors that significantly contribute to platelet force generation. When the independent effects are determined with a multivariate mixed effects model, F-actin dispersion associates more strongly with force than circularity, because it is less correlated with area. Moreover,

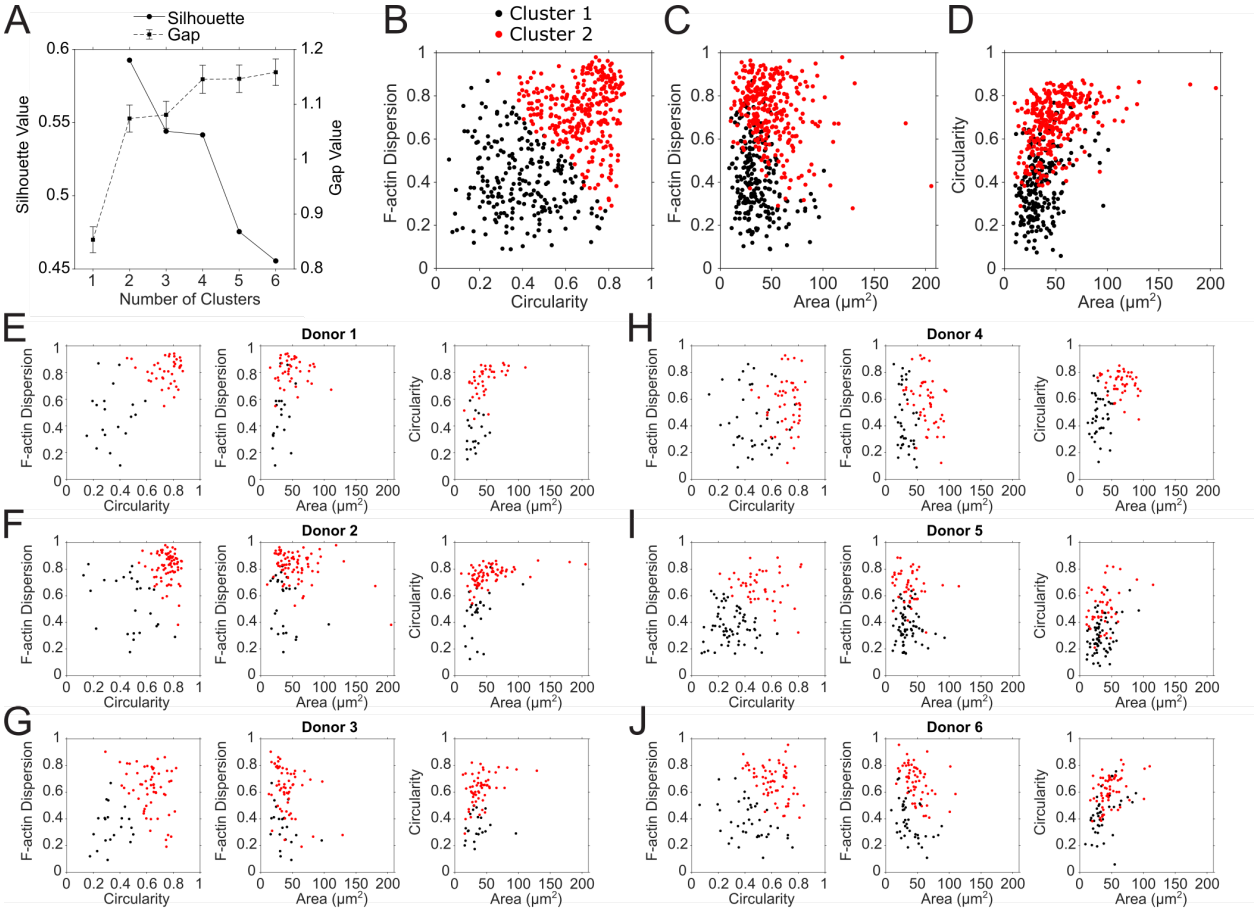


Figure 3.14: K-means clustering for each donor. (A) Silhouette and gap values used to determine the optimal number of clusters. For silhouette, the highest value determines the optimal number of clusters. For gap, the optimal number of clusters is the lowest number where the gap value of the next highest number is within the error of the previous one. In this case, both methods indicate that 2 clusters is optimal. (B, C, D) Same data as shown in the Figure 3.13A, B, and C, except the area axis shows all data points. (E-J) Relationships between area, circularity, and F-actin dispersion for all 6 donors. Here, K-means clustering was performed on each donor separately and generally shows that the platelets separate into two similar clusters for each donor.

cooperative interactions between F-actin dispersion and both area and circularity further highlight the importance of F-actin structure in generating contractile forces and provide new insight into the large heterogeneity of observed platelet forces.

Beyond the measures of area, circularity, and F-actin dispersion, the amount of contractile force a cell can generate likely depends on several factors that were not measured here, including activation of the actomyosin network by phosphorylation, amount and organization of the contractile fibers, genetic differences between donors, and disease states. The black dots platform may be used in conjunction with detailed fluorescent staining, western-blotting, or genetic screening to further enhance the understanding of force generation of platelets and other cells.

Chapter 4

STEM CELL-DERIVED CARDIOMYOCYTE CONTRACTILITY USING THE BLACK DOTS PLATFORM

The black dots method has successfully been used to measure static forces exerted by cells on the flexible substrate. However, many types of cells are dynamic and generate forces over time to perform their functions. Of specific interest are human induced pluripotent stem cell-derived cardiomyocytes (hiPSC-CMs). Cardiomyocytes, like other muscle cells, are characterized by rapid twitches in response to a stimulus that generate contractile forces which ultimately gives rise to bodily function (e.g., pumping blood, in the case of the heart). One of the most important functions of cardiomyocytes in the heart is to forcibly contract in order to pump blood through the body. Cardiomyocyte death and/or dysfunction are the underlying causes of many cardiac diseases [11]. Since cardiac tissue has a low capacity to regenerate or repair itself, hiPSC-CMs have the promise to restore function to the damaged regions of cardiac tissue. In addition to being used as a model to study heart maturation and disease [3, 19, 20, 32, 34], hiPSC-CMs have also been used to screen potential drug treatments [3, 34, 62] and to restore heart function [32, 48, 59, 60, 63, 65, 87]. Adult cardiomyocytes are packed with highly aligned myofibrils that are themselves composed of many contractile sarcomeres, the structures that contain the actin and myosin responsible for generating tension; on the other hand, stem cell-derived cardiomyocytes have fewer, more disorganized contractile structures which makes them generally weaker than their adult counterparts [79]. Understanding these forces in hiPSC-CMs is vital for their use as a potential therapy or as an assay for discovering new drugs. Here, the specific procedure for how the black dots platform can be used to measure forces in hiPSC-CMs will be discussed.

4.1 Background on stem cell-derived cardiomyocyte force measurements

Adult cardiomyocytes have a structure of actin and myosin filaments that is highly ordered, while hiPSC-CMs have a small, round morphology and generally disorganized sarcomeric structure (Fig. 4.1) [79]. The alignment of these contractile filaments allows cardiomyocytes to produce strong forces by concentrating the generation of force along one direction. Many cellular assays have been used to measure the contractility of single cardiomyocytes, including: magnetic beads [101], atomic force microscopy (AFM) [17, 25, 46, 56], traction force microscopy (TFM) [41, 44, 47], optical edge detection [12, 26], flexible cantilevers [49, 72, 102], microposts [8], and strain gauges [97]. These approaches have been used to study the biomechanics and mechanobiology of adult and neonatal cardiomyocytes, but AFM, microposts, and TFM are primarily used to measure forces from single human hiPSC-CMs [8, 41, 56]. Adult cardiomyocytes contract along a principle axis and for this reason, several methods measure force in a single direction [17, 25, 91, 92, 97]. However, immature cardiomyocytes like hiPSC-CMs do not have uniaxial contractions [100], and therefore are best suited for multidirectional force measurement methods like TFM. While twitch forces are readily measured by TFM, isometric or resting force is not obtainable without removing the cell from the substrate. Microposts have been used previously in the Sniadecki lab with some success, but their topology may adversely affect the spreading of cardiomyocytes which are particularly sensitive to their environment. Therefore, the black dots approach offers a unique advantage over existing techniques in measuring hiPSC-CM contractility, potentially unlocking new research directions.

4.2 Materials and methods

4.2.1 Generation of stem cell-derived cardiomyocytes

Human induced pluripotent stem cells (hiPSCs, WTC 11) were differentiated into cardiomyocytes using a small molecule WNT modulation protocol. First, 6-well plates were coated with 1x Matrigel at 4 °C overnight. On Day -2, hiPSCs were then plated onto the Matrigel-

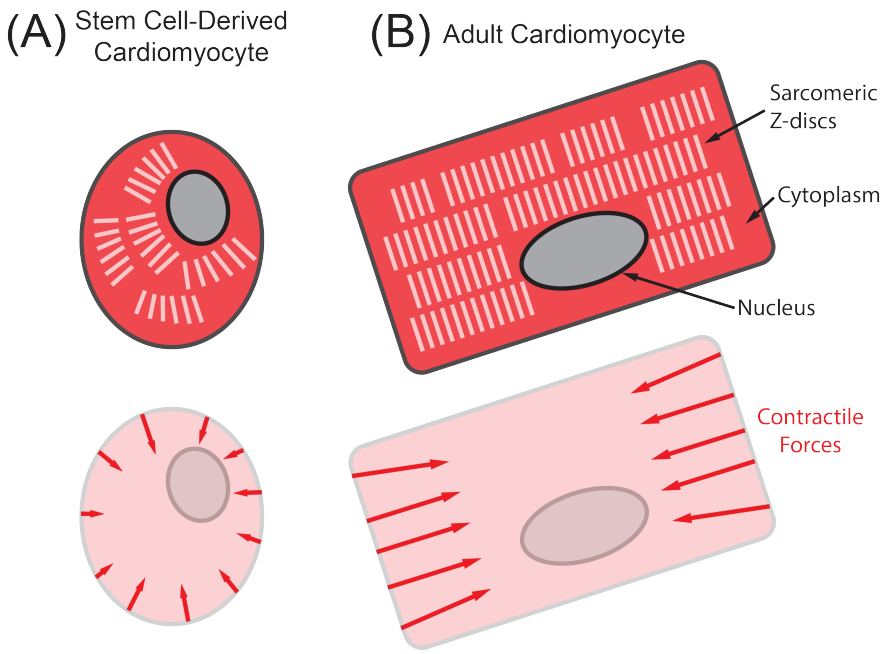


Figure 4.1: Structural and force differences in (A) immature hiPSC-CMs and (B) adult cardiomyocytes. In the top schematics, the cytoplasm is colored dark red, the sarcomeric Z-discs are white, and the nucleus is gray. The arrows in the bottom schematics indicate the primary directions of contractile forces. Stem cell-derived cardiomyocytes have a smaller, more round morphology, with unaligned myofibrils, resulting in less force generation than their adult counterparts.

coated plates at a concentration of 225 thousand cells in 4 mL of media (mTeSR+ with 10 μM Y-27632) per well. After 24 hours, on day -1, the media was changed to mTeSR with 1 μM Chiron 99021. On day 0, the cells were rinsed with DPBS and the media was changed to RPMI with 1x B-27 minus insulin and 5 μM Chiron 99021. The cells were incubated for 2 days and on day 2, the cells were washed with DPBS and the media was changed to RPMI with 1x B-27 minus insulin and 2 μM WNT C-59. After another 2 days, on day 4 the cells were washed with DPBS and the media was changed to just RPMI with 1x B-27 minus insulin. Finally, on day 6 onwards the cells were maintained with RPMI with 1x B-27 plus insulin (hereby denoted as B-27+). Penicillin-streptomycin at 1x concentration was added to all media to maintain sterility.

Beating cells are typically observed around day 7 and the cells are ready to use after day 14. The parameters used for differentiation here are subject to user variability and may need to be optimized for each user to achieve high purity; specifically the plating concentration of cells and concentrations of Chiron 99021 and WNT C-59 should be adjusted to optimal cardiomyocyte purity. If performed correctly, this protocol can achieve cardiomyocyte purity above 80%. However, low purity differentiations can be salvaged with lactate purification after day 14. To perform lactate purification, feed the cells with DMEM +Lglut -sodium pyruvate supplemented with 4 mM sodium lactate for 4 days. After the 4 day treatment, continue maintaining the cells with normal culture media (RPMI, B-27+, pen/strep).

4.2.2 Culture of cardiomyocytes on black dots substrates

Black dots substrates with pattern diameter of 3 μm and spacing of 6 μm were created with BSA-Alexa Fluor 488 on 6.25 kPa substrates (about 1% Sylgard 184 in Sylgard 527 by weight). The surface was functionalized with laminin by first depositing a 250 μL droplet of laminin diluted to 20 μg/mL in DPBS onto a parafilm surface and then gently lowering the black dot substrate face-down onto the droplet. This method ensures that the black dots substrate surface remains in contact with the liquid droplet and prevents the surface from drying which can damage the black dots pattern. After 1 hour at room temperature, the

substrates can be transferred to a 6-well plate (with the black dots pattern facing upwards) and stored in DPBS at 4 °C.

Once the hiPSC-CMs are robustly beating (typically 14-20 days after day 0 of differentiation), they can be lifted and seeded onto the black dots substrates. While older cells may be used (everything up to day 60 has been successful in preliminary experiments), younger cells typically have better attachment to the ECM, are quicker to recover, and produce more reliable twitches. Cardiomyocytes can be lifted from their original matrigel plate by rinsing with DPBS and incubating them in 0.25% trypsin-EDTA for 5 minutes at 37 °C. Once the cells are detaching, they can be triturated and collected in media (RPMI, B-27+, pen/strep) supplemented with 10% fetal bovine serum (FBS) which stops the enzymes in trypsin from destroying the cells. This cardiomyocytes can be safely pelleted by centrifuging (1000 rpm for 5 minutes using our tabletop centrifuge) and then resuspended in fresh media (same as above) with 10% FBS and 10 μM Y-27632 at a dilution of about 20,000 cells per 4 mL. Finally, the DBPS can be aspirated from the black dots substrates in the 6-well plate and 4 mL of cell suspension can be added, fully covering and submerging the substrates. Air bubbles may form underneath the substrates where the liquid has difficulty penetrating, so caution is advised to prevent substrates from floating; it can be helpful to briefly remove the substrate from the liquid entirely (using sterile tweezers), breaking the bubbles, then place the substrate back at the bottom of the well. After 24 hours, the media should be replaced with fresh media without FBS or Y-27632 (i.e., just RPMI, B-27+, pen/strep).

An alternative approach can be used to prevent the possibility of the black dots substrates drying out during this process: resuspended cardiomyocytes can be placed into an empty 6-well plate, followed by immediately transferring the black dots substrate from the laminin droplet into the suspension of cells (with the black dots pattern facing upwards). The substrate will fall to the bottom of the well, allowing media and cells to quickly cover the black dots substrate surface and maintain wetness. Cells trapped underneath the substrate should be minimal if performed relatively quickly, before cells have a chance to adhere to the plate. Care should be taken to ensure the substrate is fully submerged as surface tension

can cause the substrate to float at the top of the liquid.

4.2.3 Live microscopy

These twitches in hiPSC-CMs occur on the order of once every second, or 1 Hz, which imposes special imaging requirements to capture these dynamics. In addition to a high-speed camera capable of capturing many frames each second (often up to 100 Hz), a strong light source is needed to provide enough photons to create a clear image on the camera's detectors. However, cells are generally sensitive to this strong light, especially higher-energy fluorescent light, which can complicate the imaging process. This results in a challenging scenario where one experimental factor (the camera) demands more light, while another (the cells) demand less. In particular, with hiPSC-CMs this light can cause the cells to stop spontaneously beating, or even cause the cells to lift off and die. Special care should be taken to reduce the adverse effects of the bright, fluorescent light which can cause cell damage with prolonged exposure. Here, a specific protocol for imaging black dots substrates with living cells on an inverted microscope is detailed.

After the hiPSC-CMs have adhered to the black dots substrates and recovered (typically 2-3 days) they will begin to spontaneously beat, contracting the black dots pattern and causing visible displacement. The black dots substrate is transferred to an Attofluor chamber with fresh maintenance media (RPMI, B-27+, pen/strep). A widefield, inverted microscope (Nikon Ti-E) with temperature controlled environment set to 37 °C is used to image the substrate using a high-speed camera (Hamamatsu V3 Flash4.0). A 60x or 40x water-immersion objective is used by carefully balancing a water droplet on the inverted objective tip. Finally, an gas control box is placed over the Attofluor chamber to maintain a 5% CO_2 level while imaging.

This protocol results in video capture of at least 20 frames per second while preventing adverse effects to the cardiomyocytes. Oil objectives typically don't have enough working distance to see through the glass and PDMS substrate layers, while air objectives have poor performance in high-speed conditions, so water objectives are the best option. If the water

droplet does not balance easily on the objective, commercial products with similar optical density or index of refraction can be used in place of water (e.g., GenTeal eye drops). RPMI with no phenol red can be used here if there are concerns about fluorescent background. Preliminary experiments using an imaging buffer such as Tyrode's buffer resulted in cells that died within a few seconds of intense fluorescent light, and the entire substrate would stop spontaneously beating within 1 hour; by imaging in the normal culture media, cells were able to survive imaging for at least 10 seconds and remained spontaneously beating for at least 1.5 hours.

4.2.4 Electrical pacing

Electrical stimulus can be applied to cardiomyocytes on black dots substrates in a 6-well plate by using existing plate stimulators like the C-Pace system. To electrically stimulate cells while imaging, a stimulating cover can be fabricated from a 35 mm petri dish cover and carbon electrodes or platinum wire to deliver electrical signal to cells. While the effects of long-term electrical stimulation have not been investigated here, applying electrical pacing while imaging was extremely unreliable for single, isolated hiPSC-CMs; only very large cells or clumps of 2 or more cells would follow the electrical pace, and single cells would rarely follow pacing despite the voltage or frequency. Efforts to mature the cells or modifications to media ion concentrations (specifically calcium Ca^{2+}) may be required for single cells to respond to electrical pacing, although this has not been tested here.

4.2.5 Video analysis of black dots substrates

Analyzing the displacements of black dots in videos is largely similar to Chapter 2.2, with some modifications.

Automated approach

The open-source particle tracking code already has a built-in function to track particles in videos by comparing the locations of each found particle on each frame of the video, "joining" a particle on one frame to a particle on the next frame if it occupies roughly the same location [9]. The code even accounts for when particles disappear from one frame and reappear on the next due to noise or debris briefly obscuring the particle. For black dots analysis, this code is easily able to find and track the pattern and any tracked points that are not present on at least 75% of video frames are discarded. For the particles that are not tracked on every single frame, the gaps are filled in by using the particle's location on the nearest frame. The code can sometimes track random debris or pattern defects in addition to the pattern itself, so to clean this up a user may need to manually remove extraneous particles.

Manual approach

For video analysis, the manual approach described in Chapter 2.2 can be directly applied to each frame of the video, using the previous frame's dot locations as the initial dot estimate. The user only needs to manually draw the grid for a single frame. Occasionally, a row or column of dots may shift slightly out of the field of view, causing the number of dots in that row or column to go below the number required to fit a line; for example, if 2 dots from each end of the row are needed to fit a line, then there must be at least 4 dots in that row. If this happens, that row or column is discarded for all frames already analyzed and for all future frames. This manual approach does not require any further user input as the grid is initially estimated by the user, and can typically analyze a full 200-frame video within 1 minute. Figure 4.2A-B show the forces in a relaxed and contracted cardiomyocyte using this manual approach.

Filtering temporal noise

After finding each black dot position in each frame, the location of each black dot over time is passed through a butterworth lowpass filter with a 4 Hz cutoff to eliminate high frequency noise that may arise due to low image resolution of the black dots pattern. In MATLAB, this is performed using the "butter" function, passing in a cutoff frequency that is normalized with respect to the Nyquist frequency (one half of the video frame rate). The "filtfilt" function is used to apply the filter without phase shifting the location over time. To reduce any adverse effects of the filter, the data can be padded with a mirrored version on the front and back of the waveform.

4.2.6 Transient analysis for isometric and twitch force dynamics

The output of black dots analysis for videos is the total contractile force for each video frame: a force transient as shown in Figure 4.2C. Meaningful measurements to be extracted from the force transient include isometric (resting) force and twitch (active) force, as well as different temporal measurements including time to peak force and time to 50% or 90% decay. The black dots analysis code contains an automated approach for measuring these by finding the start, peak, and 50% and 90% relaxation time for each twitch in the force transient. The code first automatically finds the major peaks and troughs corresponding to highest and lowest force for each beat. Next, the onset of each twitch is found by looking between each trough and the next peak, using a threshold to determine when the force starts increasing rapidly. Finally the 50% and 90% decay are determined by simply finding the first force below 50% or 90% of the twitch amplitude for that beat. The code for this transient analysis is included in Appendix A.3.

4.3 Preliminary results and discussion

As proof of concept, five cardiomyocytes from the same experiment were analyzed to measure their contractile force and transient dynamics. Here, hiPSC-CMs were cultured as previously

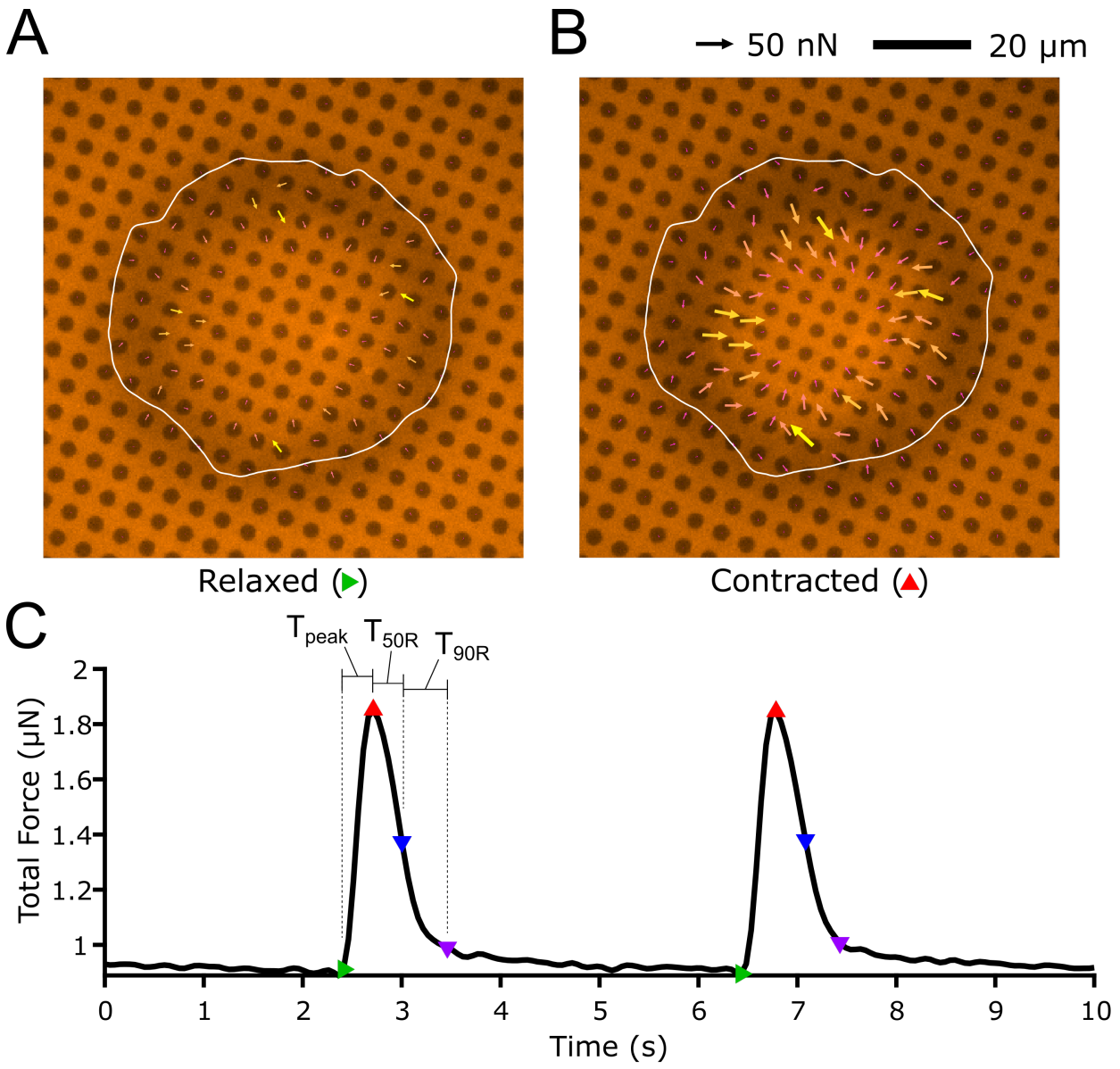


Figure 4.2: The black dots platform is used to measure forces in a twitching hiPSC-CM. Forces are shown in the (A) relaxed and (B) contracted portion of the twitch. Scale bars apply to both images. (C) The total force transient over the 200 frame video. Different temporal measurements such as time to peak T_{peak}, time to 50% relaxation T_{50R}, and time to 90% relaxation T_{90R} are indicated on the transient.

Area (μm^2)	Beat Rate (Hz)	Isometric Force (μN)	Twitch Force (μN)	T_{peak} (s)	T_{50R} (s)	T_{90R} (s)
3484.02	0.25	0.89	0.96	0.327	0.294	0.697
6944.82	0.35	2.15	2.13	0.352	0.473	0.975
2962	0.69	1.15	0.15	0.377	0.246	0.535
3694.27	0.25	1.1	0.54	0.302	0.246	0.728
16060.57	0.25	3.02	1.45	0.276	0.257	0.708

Table 4.1: Cell spread area, beat rate, force amplitude, and transient dynamics measured from hiPSC-CMs shown in Figure 4.3. Spread area was measured from the manually traced cell outline, while all other measurements were automatically calculated from the force transients.

described until day 14 when they were harvested and placed into cryostorage. Thawed hiPSC-CMs were maintained and seeded on black dots substrates coated with laminin on day 46, allowed to recover for a few days, and subsequently imaged on day 51. Traction forces for these five cardiomyocytes are shown in Figure 4.3A; these five cells were selected randomly across three separate substrates and show a wide variety in force amplitude and spontaneous twitch rate. Data extracted from these force transients is shown in Table 4.1. Due to the age of these cells, the spontaneous twitch rate was 0.25 - 0.7 Hz which resulted in only a handful of beats that could be recorded in a 10-second video. It is possible that a younger cell population could yield a quicker beat rate and more repeated measurements, although immature hiPSC-CMs can also have other beat irregularities due to their more immature calcium handling machinery.

For each these cardiomyocytes, 7 quantities were measured: spread area (traced by hand), beat rate, isometric force, twitch force, time to peak, time to 50% relaxation, and time to 90% relaxation. Figure 4.3B shows correlation plots for several relationships between these quantities. A specific focus is placed on isometric force since this a measurement that is not easily available for existing cardiomyocyte studies using TFM. While this is not enough data for any certainty, some interesting trends can be observed. Isometric force appears to correlate strongly with spread area, while twitch force is less clear. The time to peak force

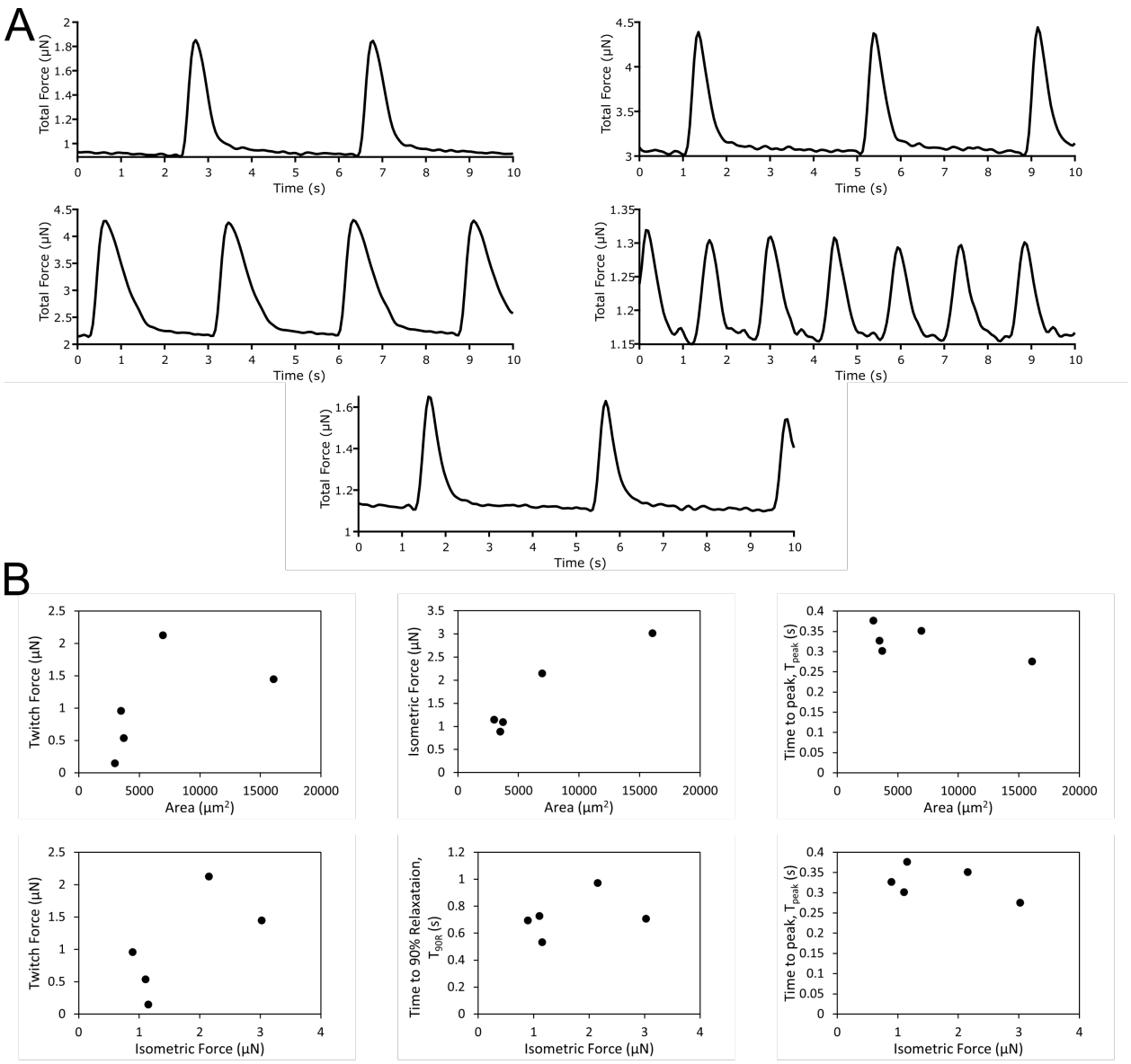


Figure 4.3: (A) Twitch force transients from five individual cardiomyocytes measured by the black dots platform. (B) Several comparison plots for different measurements from the analyzed force transients.

may actually be quicker for larger cells and cells with higher isometric force, as indicated by a slight negative correlation. Relaxation kinetics and beat rate (generally not shown on these plots) were not clearly correlated with force or area, although further data may be more revealing.

4.4 Concluding remarks

Immature hiPSC-derived cardiomyocytes pose a significant challenge in the field of traction force measurement; they are highly dynamic cells that produce large forces and are sensitive to their imaging environment, resulting in a uniquely difficult experience. The black dots platform was able to overcome these challenges and allow for a reference-free measurement of the dynamic forces from cardiomyocytes at high framerate, without significantly disturbing the cells during imaging. Data from several cardiomyocytes on black dots substrates were shown, revealing interesting trends between cell area, isometric force, and twitch dynamics. The protocols and methods described here will allow for larger more comprehensive investigations into the contractility in these immature cardiomyocytes.

While not shown here, the black dots platform offers some significant new avenues for research in hiPSC-CMs. The imaging process, if performed on an inverted microscope, is fully sterile and therefore is compatible with long-term studies that may reveal exciting new discoveries about cardiomyocyte development by tracking individual cell contractility over long periods of culture. Additionally, the reference-free nature of the black dots approach allows for fixing and staining which could allow for the analysis of protein structure in combination with twitch force dynamics; for example, many videos of cardiomyocytes beating on black dots substrates could be captured while noting the location of the cells on the substrate, then the cells could be fixed and stained for proteins like alpha-actinin that reveal the myofibril structure which could then be correlated with the isometric or twitch contractile forces. This type of study is not feasible on traditional TFM substrates and would require the use of live protein markers which come with their own challenges. Overall, the black dots platform provides a more streamlined, straightforward approach to measuring forces

in live, beating cardiomyocytes that is compatible with standard immunocytochemistry or other post-analysis procedures such as western blotting, and may pave the way for new discoveries.

Chapter 5
SUMMARY AND FUTURE DIRECTIONS

In this work, a novel reference-free traction force microscopy approach dubbed "black dots" was developed. The breakthrough of PVA film technology allowed for a robust way to deposit fluorescent protein onto the extremely soft PDMS substrates, which is difficult with traditional microcontact printing. The theory and analysis of the black dots approach was detailed to showcase how traction forces can be estimated from a single image, without the need for a second reference image, widening the range of applications to include fixed cells. The mathematics for estimating the traction forces from the displacements utilized the simplest form of Fourier transform traction cytometry, which naturally fits the rectangular grid of measured displacements from the black dots approach. A huge emphasis is put on the tunability of the black dots approach, facilitating its use for many cell types by modifying the substrate stiffness, fluorescent protein color, and extracellular matrix.

Using this method, forces in over 500 platelets were analyzed, a number that is unmatched in literature without constraining cell spreading. This large amount of data allowed for a deep dive into the different factors that might be influencing force generation: spread area, circularity, and F-actin dispersion. This data also highlighted newly uncovered cooperative effects between cell shape, size, and structure that affect force generation, and also facilitated machine learning approaches like k-means clustering which rely on large amounts of data. While this study only focused on some of the more obvious factors contributing to force, it is exciting to think about what other discoveries will be unlocked by staining for other proteins or even correlating subcellular force and protein structures in platelets.

Finally, the forces in dynamic, beating cardiomyocytes was investigated, further expanding the experimental domain of the black dots approach. The fluorescent protein deposited

on soft PDMS is bright enough to capture the dynamic beating of hiPSC-CMs at high framerates while maintaining sterility and cell health. A handful of cells were analyzed, providing a glimpse into what investigations will be possible with this approach, namely isometric force is difficult to obtain with other techniques and has not been well studied in hiPSC-CMs. Long-term culture is also possible for hiPSC-CMs on black dots substrates, which will allow for the monitoring of individual cell contractility over development and may help our understanding of immature cardiac development.

In its current state, the black dots approach is not compatible with additional microcontact printing to, for example, deposit aligned strips of ECM on top of the black dots pattern. This type of experiment would be useful to control cell spreading or, in the case of cardiomyocytes, to control their aspect ratio which can affect myofibril alignment and contractility. Still, the future is promising for the black dots approach, as it is highly tunable and can be used with many cell types. Traditional TFM which can be difficult to adopt by new labs, and this new method serves as an alternative that may be quicker to pick up, easier to analyze, and more amenable to future improvements. The black dots platform is compatible with fixed and live cells, which opens the door to study many interesting phenomena, and it will be exciting to watch how the platform grows and is used for new biological discoveries in the future.

BIBLIOGRAPHY

[1] Nathalie Q. Balaban, Ulrich S. Schwarz, Daniel Riveline, Polina Goichberg, Gila Tzur, Ilana Sabanay, Diana Mahalu, Sam Safran, Alexander Bershadsky, Lia Addadi, and Benjamin Geiger. Force and focal adhesion assembly: A close relationship studied using elastic micropatterned substrates. *Nature Cell Biology*, 3(5):466–472, 2001.

[2] Omar A Banda, Chandran R Sabanayagam, and John Slater. A Reference-Free Traction Force Microscopy Platform Fabricated via Two-Photon Laser Scanning Lithography Enables Facile Measurement of Cell-Generated Forces. *ACS Applied Materials & Interfaces*, page acsami.9b04362, 2019.

[3] Milena Bellin, Maria C. Marchetto, Fred H. Gage, and Christine L. Mummery. Induced pluripotent stem cells: The new patient? *Nature Reviews Molecular Cell Biology*, 13:713–726, 2012.

[4] Markus Bender and Raghavendra Palankar. Platelet Shape Changes during Thrombus Formation: Role of Actin-Based Protrusions. *Hamostaseologie*, 41(1):14–21, 2021.

[5] Martin Bergert, Tobias Lendenmann, Manuel Zündel, Alexander E. Ehret, Daniele Panozzo, Patrizia Richner, David K. Kim, Stephan J.P. Kress, David J. Norris, Olga Sorkine-Hornung, Edoardo Mazza, Dimos Poulikakos, and Aldo Ferrari. Confocal reference free traction force microscopy. *Nature Communications*, 7:12814, 2016.

[6] Alessandro Bertero, Paul A. Fields, Alec S.T. Smith, Andrea Leonard, Kevin Beussman, Nathan J. Sniadecki, Deok Ho Kim, Hung Fat Tse, Lil Pabon, Jay Shendure, William S. Noble, and Charles E. Murry. Chromatin compartment dynamics in a haploinsufficient model of cardiac laminopathy. *Journal of Cell Biology*, 218:2919–2944, 2019.

[7] Kevin M. Beussman, Molly Y. Mollica, Andrea Leonard, Jeffrey Miles, John Hocter, Zizhen Song, Moritz Stolla, Sangyoon J. Han, Ashley Emery, Wendy E. Thomas, and Nathan J. Sniadecki. Black Dots: Microcontact-Printed, Reference-Free Traction Force Microscopy. *bioRxiv*, page 2021.08.02.454500, 2021.

[8] Kevin M. Beussman, Marita L. Rodriguez, Andrea Leonard, Nikita Taparia, Curtis R. Thompson, and Nathan J. Sniadecki. Micropost arrays for measuring stem cell-derived cardiomyocyte contractility. *Methods*, 94:43–50, 2016.

[9] Daniel Blair and Eric Dufresne. The matlab particle tracking code repository. *Particle-tracking code available at http://physics.georgetown.edu/matlab*, 2008.

[10] Mara Brancaccio, Simona Guazzone, Nadia Menini, Elena Sibona, Emilio Hirsch, Marco De Andrea, Mariano Rocchi, Fiorella Altruda, Guido Tarone, and Lorenzo Silengo. Melusin is a new muscle-specific interactor for β1 integrin cytoplasmic domain. *Journal of Biological Chemistry*, 274:29282–29288, 1999.

[11] Eugene Braunwald and Robert O. Bonow. *Braunwald's heart disease : a textbook of cardiovascular medicine*. Saunders, 9th edition, 2012.

[12] Klara Brixius, Susanne Hoischen, Hannes Reuter, Kathrin Lasek, and Robert H.G. Schwinger. Force/shortening-frequency relationship in multicellular muscle strips and single cardiomyocytes of human failing and nonfailing hearts. *Journal of Cardiac Failure*, 7:335–341, 2001.

[13] James P. Butler, Iva Marija Tolić-Nørrelykke, Ben Fabry, and Jeffrey J. Fredberg. Traction fields, moments, and strain energy that cells exert on their surroundings. *American Journal of Physiology - Cell Physiology*, 282(3 51-3):C595–C605, 2002.

[14] Joseph P. Califano and Cynthia A. Reinhart-King. Substrate stiffness and cell area predict cellular traction stresses in single cells and cells in contact. *Cellular and Molecular Bioengineering*, 3(1):68–75, 2010.

[15] Elizabeth P. Canović, D. Thomas Seidl, Samuel R. Polio, Assad A. Oberai, Paul E. Barbone, Dimitrije Stamenović, and Michael L. Smith. Biomechanical imaging of cell stiffness and prestress with subcellular resolution. *Biomechanics and Modeling in Mechanobiology*, 13(3):665–678, 2014.

[16] M. E. Carr and S. L. Carr. Fibrin structure and concentration alter clot elastic modulus but do not alter platelet mediated force development. *Blood Coagulation and Fibrinolysis*, 6(1):79–86, 1995.

[17] Wei Tien Chang, David Yu, Yu Cheng Lai, Kuen You Lin, and Ian Liau. Characterization of the mechanodynamic response of cardiomyocytes with atomic force microscopy. *Analytical Chemistry*, 85:1395–1400, 2013.

[18] Zhaowei Chen, Jiankai Lu, Changjie Zhang, Isaac Hsia, Xinheng Yu, Leo Marecki, Eric Marecki, Mohammadnabi Asmani, Shilpa Jain, Sriram Neelamegham, and Ruogang Zhao. Microclot array elastometry for integrated measurement of thrombus formation and clot biomechanics under fluid shear. *Nature Communications*, 10(1):1–13, 2019.

[19] Cheryl Dambrot, Robert Passier, Douwe Atsma, and Christine L. Mummery. Cardiomyocyte differentiation of pluripotent stem cells and their use as cardiac disease models. *Biochemical Journal*, 434:25–35, 2011.

[20] Anjan Kumar Das and Rajarshi Pal. Induced pluripotent stem cells (ipscs): The emergence of a new champion in stem cell technology-driven biomedical applications. *Journal of Tissue Engineering and Regenerative Medicine*, 4:413–421, 2010.

[21] Juan C. Del Álamo, Ruedi Meili, Baldomero Alonso-Latorre, Javier Rodríguez-Rodríguez, Alberto Aliseda, Richard A. Firtel, and Juan C. Lasheras. Spatio-temporal analysis of eukaryotic cell motility by improved force cytometry. *Proceedings of the National Academy of Sciences of the United States of America*, 104(33):13343–13348, 2007.

[22] Juan C. Del Álamo, Ruedi Meili, Begoña Álvarez-González, Baldomero Alonso-Latorre, Effie Bastounis, Richard Firtel, and Juan C. Lasheras. Three-dimensional quantification of cellular traction forces and mechanosensing of thin substrata by fourier traction force microscopy. *PLOS ONE*, 8:e69850, 2013.

[23] Micah Dembo and Yu-Li Wang. Stresses at the cell-to-substrate interface during locomotion of fibroblasts. *Biophysical Journal*, 76(4):2307–2316, 1999.

[24] Dennis E. Discher, Paul Janmey, and Yu Li Wang. Tissue cells feel and respond to the stiffness of their substrate, 2005.

[25] Jan Domke, Wolfgang J. Parak, Michael George, Hermann E. Gaub, and Manfred Radmacher. Mapping the mechanical pulse of single cardiomyocytes with the atomic force microscope. *European Biophysics Journal*, 28:179–186, 1999.

[26] István Ferenc Édes, Dániel Czuriga, Gábor Csányi, Stefan Chłopicki, Fabio A. Recchia, Attila Borbély, Zoltán Galajda, István Ferenc Édes, Jolanda van der Velden, Ger J.M. M Stienen, and Zoltán Papp. Rate of tension redevelopment is not modulated by sarcomere length in permeabilized human, murine, and porcine cardiomyocytes. *American journal of physiology. Regulatory, integrative and comparative physiology*, 293:R20–9, 2007.

[27] Danny El-Nachef, Darrian Bugg, Kevin M. Beussman, Amy M. Martinson, Charles E. Murry, Nathan J. Sniadecki, Jennifer Davis, Sonette Steczina, Amy M. Martinson, Charles E. Murry, Nathan J. Sniadecki, and Jennifer Davis. Engrafted human induced pluripotent stem cell-derived cardiomyocytes undergo clonal expansion in vivo. *Circulation*, 143:1635–1638, 2021.

[28] Danny El-Nachef, Kevin Shi, Kevin M. Beussman, Refugio Martinez, Mary C. Regier, Guy W. Everett, Charles E. Murry, Kelly R. Stevens, Jessica E. Young, Nathan J. Sniadecki, and Jennifer Davis. A rainbow reporter tracks single cells and reveals het-

erogeneous cellular dynamics among pluripotent stem cells and their differentiated derivatives. *Stem Cell Reports*, 15:226–241, 2020.

[29] Shirin Feghhi, Adam D. Munday, Wes W. Tooley, Shreya Rajsekar, Adriane M. Fura, John D. Kulman, Jose A. López, and Nathan J. Sniadecki. Glycoprotein Ib-IX-V Complex Transmits Cytoskeletal Forces That Enhance Platelet Adhesion. *Biophysical Journal*, 111(3):601–608, 2016.

[30] Daniel A. Fletcher and R. Dyche Mullins. Cell mechanics and the cytoskeleton, 2010.

[31] C. Franck, S. Hong, S. A. Maskarinec, D. A. Tirrell, and G. Ravichandran. Three-dimensional Full-field Measurements of Large Deformations in Soft Materials Using Confocal Microscopy and Digital Volume Correlation. *Experimental Mechanics 2007 47:3*, 47(3):427–438, 2007.

[32] Christian Freund and Christine L. Mummery. Prospects for pluripotent stem cell-derived cardiomyocytes in cardiac cell therapy and as disease models. *Journal of Cellular Biochemistry*, 107:592–599, 2009.

[33] Brian P. Griffin, Christopher J. Largaespada, Nicole A. Rinaldi, and Christopher A. Lemmon. A novel method for quantifying traction forces on hexagonal micropatterned protein features on deformable poly-dimethyl siloxane sheets. *MethodsX*, 6:1343–1352, 2019.

[34] Marica Grskovic, Ashkan Javaherian, Berta Strulovici, and George Q. Daley. Induced pluripotent stem cells - opportunities for disease modelling and drug discovery. *Nature Reviews Drug Discovery 2011 10:12*, 10:915–929, 2011.

[35] Sangyoon J. Han, Kevin S. Bielawski, Lucas H. Ting, Marita L. Rodriguez, and Nathan J. Sniadecki. Decoupling Substrate Stiffness, Spread Area, and Micropost Density: A Close Spatial Relationship between Traction Forces and Focal Adhesions. *Biophysical Journal*, 103(4):640–648, 2012.

[36] Sangyoon J. Han, Youbean Oak, Alex Groisman, and Gaudenz Danuser. Traction microscopy to identify force modulation in subresolution adhesions. *Nature Methods*, 12(7):653–656, 2015.

[37] Jana Hanke, Dimitri Probst, Assaf Zemel, Ulrich S. Schwarz, and Sarah Köster. Dynamics of force generation by spreading platelets. *Soft Matter*, 14(31):6571–6581, 2018.

[38] Jana Hanke, Christiane Ranke, Eleonora Perego, and Sarah Köster. Human blood platelets contract in perpendicular direction to shear flow. *Soft Matter*, 15(9):2009–2019, 2019.

[39] Per Christian Hansen. Analysis of Discrete Ill-Posed Problems by Means of the L-Curve. *SIAM Review*, 34(4):561–580, 1992.

[40] Albert K. Harris, Patricia Wild, and David Stopak. Silicone rubber substrata: A new wrinkle in the study of cell locomotion. *Science*, 208(4440):177–179, 1980.

[41] Laurie B. Hazeltine, Chelsey S. Simmons, Max R. Salick, Xiaojun Lian, Mehmet G. Badur, Wenqing Han, Stephanie M. Delgado, Tetsuro Wakatsuki, Wendy C. Crone, Beth L. Pruitt, and Sean P. Palecek. Effects of substrate mechanics on contractility of cardiomyocytes generated from human pluripotent stem cells. *International Journal of Cell Biology*, 2012:508294, 2012.

[42] Béatrice Hechler, Arnaud Dupuis, Pierre H. Mangin, and Christian Gachet. Platelet preparation for function testing in the laboratory and clinic: Historical and practical aspects. *Research and Practice in Thrombosis and Haemostasis*, 3(4):615–625, 2019.

[43] Sarah Schwarz Henriques, Rabea Sandmann, Alexander Strate, and Sarah Köster. Force field evolution during human blood platelet activation. *Journal of Cell Science*, 125(16):3914–3920, 2012.

[44] Nils Hersch, Benjamin Wolters, Georg Dreissen, Ronald Springer, Norbert Kirchgeßner, Rudolf Merkel, Bernd Hoffmann, Bernd Hoffman, N. Kirchgessner, Rudolf Merkel, and

Bernd Hoffmann. The constant beat: cardiomyocytes adapt their forces by equal contraction upon environmental stiffening. *Biology Open*, 2:351–361, 2013.

[45] Yunfei Huang, Gerhard Gompper, and Benedikt Sabass. A Bayesian traction force microscopy method with automated denoising in a user-friendly software package. *Computer Physics Communications*, 256:107313, 2020.

[46] Jeffrey G. Jacot, Jody C. Martin, and Darlene L. Hunt. Mechanobiology of cardiomyocyte development. *Journal of Biomechanics*, 43:93–98, 2010.

[47] Jeffrey G. Jacot, Andrew D. McCulloch, and Jeffrey H. Omens. Substrate stiffness affects the functional maturation of neonatal rat ventricular myocytes. *Biophysical Journal*, 95:3479–3487, 2008.

[48] Masashi Kawamura, Shigeru Miyagawa, Kenji Miki, Atsuhiro Saito, Satsuki Fukushima, Takahiro Higuchi, Takuji Kawamura, Toru Kuratani, Takashi Daimon, Tatsuya Shimizu, Teruo Okano, and Yoshiki Sawa. Feasibility, safety, and therapeutic efficacy of human induced pluripotent stem cell-derived cardiomyocyte sheets in a porcine ischemic cardiomyopathy model. *Circulation*, 126:S29–37, 2012.

[49] Jinseok Kim, Jungyul Park, Kyounghwan Na, Sungwook Yang, Jeongeun Baek, Euisung Yoon, Sungsik Choi, Sangho Lee, Kukjin Chun, Jongoh Park, and Sukho Park. Quantitative evaluation of cardiomyocyte contractility in a 3d microenvironment. *Journal of Biomechanics*, 41:2396–2401, 2008.

[50] Wilbur A. Lam, Ovijit Chaudhuri, Ailey Crow, Kevin D. Webster, Tai De Li, Ashley Kita, James Huang, and Daniel A. Fletcher. Mechanics and contraction dynamics of single platelets and implications for clot stiffening. *Nature Materials*, 10(1):61–66, 2011.

[51] Juliet Lee, Michelle Leonard, Tim Oliver, Akira Ishihara, and Ken Jacobson. Traction

forces generated by locomoting keratocytes. *Journal of Cell Biology*, 127(6 II):1957–1964, 1994.

[52] Tobias Lendenmann, Teseo Schneider, Jérémie Dumas, Marco Tarini, Costanza Giampietro, Apratim Bajpai, Weiqiang Chen, Julia Gerber, Dimos Poulikakos, Aldo Ferrari, and Daniele Panozzo. Cellogram: On-the-Fly Traction Force Microscopy. *Nano Letters*, 19(10):6742–6750, 2019.

[53] Andrea Leonard, Alessandro Bertero, Joseph D. Powers, Kevin M. Beussman, Shiv Bhandari, Michael Regnier, Charles E. Murry, and Nathan J. Sniadecki. Afterload promotes maturation of human induced pluripotent stem cell derived cardiomyocytes in engineered heart tissues. *Journal of Molecular and Cellular Cardiology*, 118:147–158, 2018.

[54] Sebastian Lickert, Simona Sorrentino, Jan Dirk Studt, Ohad Medalia, Viola Vogel, and Ingmar Schoen. Morphometric analysis of spread platelets identifies integrin αiIbβ3-specific contractile phenotype. *Scientific Reports*, 8(1):5428, 2018.

[55] Yu Chun Lin, Dhananjay T. Tambe, Chan Young Park, Michael R. Wasserman, Xavier Trepat, Ramaswamy Krishnan, Guillaume Lenormand, Jeffrey J. Fredberg, and James P. Butler. Mechanosensing of substrate thickness. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 82:041918, 2010.

[56] Jianwei Liu, Ning Sun, Marc A. Bruce, Joseph C. Wu, and Manish J. Butte. Atomic force mechanobiology of pluripotent stem cell-derived cardiomyocytes. *PLoS ONE*, 7:e37559, 2012.

[57] Donald MacNearney, Bernard Mak, Grant Ongo, Timothy E. Kennedy, and David Juncker. Nanocontact Printing of Proteins on Physiologically Soft Substrates to Study Cell Haptotaxis. *Langmuir*, 32(50):13525–13533, 2016.

[58] John M. Maloney, Emily B. Walton, Christopher M. Bruce, and Krystyn J. Van Vliet. Influence of finite thickness and stiffness on cellular adhesion-induced deformation of compliant substrata. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 78(4):041923, 2008.

[59] Hidetoshi Masumoto, Takehiko Matsuo, Kohei Yamamizu, Hideki Uosaki, Genta Narazaki, Shiori Katayama, Akira Marui, Tatsuya Shimizu, Tadashi Ikeda, Teruo Okano, Ryuzo Sakata, and Jun K. Yamashita. Pluripotent stem cell-engineered cell sheets reassembled with defined cardiovascular populations ameliorate reduction in infarct heart function through cardiomyocyte-mediated neovascularization. *Stem Cells*, 30:1196–1205, 2012.

[60] Christina Mauritz, Andreas Martens, Sebastian V. Rojas, Tilman Schnick, Christian Rathert, Natalie Schecker, Sandra Menke, Silke Glage, Robert Zweigerdt, Axel Haverich, Ulrich Martin, and Ingo Kutschka. Induced pluripotent stem cell (ipsc)-derived flk-1 progenitor cells engraft, differentiate, and improve heart function in a mouse model of acute myocardial infarction. *European Heart Journal*, 32:2634–2641, 2011.

[61] Megan L. McCain, Hongyan Yuan, Francesco S. Pasqualini, Patrick H. Campbell, and Kevin Kit Parker. Matrix elasticity regulates the optimal cardiac myocyte shape for contractility. *AJP: Heart and Circulatory Physiology*, 306(11):H1525–H1539, 2014.

[62] Mark Mercola, Alexandre Colas, and Erik Willems. Induced pluripotent stem cells in cardiovascular drug discovery. *Circulation Research*, 112:534–548, 2013.

[63] Kenji Miki, Hisazumi Uenaka, Atsuhiro Saito, Shigeru Miyagawa, Taichi Sakaguchi, Takahiro Higuchi, Tatsuya Shimizu, Teruo Okano, Shinya Yamanaka, and Yoshiki Sawa. Bioengineered myocardium derived from induced pluripotent stem cells improves cardiac function and attenuates cardiac remodeling following chronic myocardial infarction in rats. *STEM CELLS Translational Medicine*, 1:430–437, 2012.

[64] Jason W. Miklas, Elisa Clark, Shiri Levy, Damien Detraux, Andrea Leonard, Kevin Beussman, Megan R. Showalter, Alec T. Smith, Peter Hofsteen, Xiulan Yang, Jesse Macadangdang, Tuula Manninen, Daniel Raftery, Anup Madan, Anu Suomalainen, Deok Ho Kim, Charles E. Murry, Oliver Fiehn, Nathan J. Sniadecki, Yuliang Wang, and Hannele Ruohola-Baker. Tfpa/hadha is required for fatty acid beta-oxidation and cardiolipin re-modeling in human cardiomyocytes. *Nature Communications*, 10, 2019.

[65] Federico Mosna, Francesco Annunziato, Giovanni Pizzolo, and Mauro Krampera. Cell therapy for cardiac regeneration after myocardial infarct: which cell is the best? *Cardiovascular & hematological agents in medicinal chemistry*, 8:227–243, 2010.

[66] David R. Myers, Yongzhi Qiu, Meredith E. Fay, Michael Tennenbaum, Daniel Chester, Jonas Cuadrado, Yumiko Sakurai, Jong Baek, Reginald Tran, Jordan C. Ciciliano, Byungwook Ahn, Robert G. Mannino, Silvia T. Bunting, Carolyn Bennett, Michael Briones, Alberto Fernandez-Nieves, Michael L. Smith, Ashley C. Brown, Todd Sulchek, and Wilbur A. Lam. Single-platelet nanomechanics measured by high-throughput cytometry. *Nature Materials*, 16(2):230–235, 2017.

[67] J. Notbohm, J.-H. H. Kim, A.R. R. Asthagiri, and G. Ravichandran. Three-dimensional analysis of the effect of epidermal growth factor on cell-cell adhesion in epithelial cell clusters. *Biophysical Journal*, 102(6):1323–1330, 2012.

[68] Ava M. Obenaus, Molly Y. Mollica, and Nathan J. Sniadecki. (De)form and Function: Measuring Cellular Forces with Deformable Materials and Deformable Structures. *Advanced Healthcare Materials*, 9(8):1–16, 2020.

[69] Akiko Ono, Erik Westein, Sarah Hsiao, Warwick S. Nesbitt, Justin R. Hamilton, Simone M. Schoenwaelder, and Shaun P. Jackson. Identification of a fibrin-independent platelet contractile mechanism regulating primary hemostasis and thrombus growth. *Blood*, 112(1):90–99, 2008.

[70] Aishwarya K. Paknikar, Benjamin Eltzner, and Sarah Köster. Direct characterization of cytoskeletal reorganization during blood platelet spreading. *Progress in Biophysics and Molecular Biology*, 144:166–176, 2019.

[71] Rachelle N. Palchesko, Ling Zhang, Yan Sun, and Adam W. Feinberg. Development of Polydimethylsiloxane Substrates with Tunable Elastic Modulus to Study Cell Mechanobiology in Muscle and Nerve. *PLoS ONE*, 7(12):e51499, 2012.

[72] Jungyul Park, Jaewook Ryu, Seung Kyu Choi, Eunseok Seo, Jae Min Cha, Seokchang Ryu, Jinseok Kim, Byungkyu Kim, and Sang Ho Lee. Real-time measurement of the contractile forces of self-organized cardiomyocytes on hybrid biopolymer microcantilevers. *Analytical Chemistry*, 77:6571–6580, 2005.

[73] Jeremy A. Pike, Victoria A. Simms, Christopher W. Smith, Neil V. Morgan, Abdullah O. Khan, Natalie S. Poulter, Iain B. Styles, and Steven G. Thomas. An adaptable analysis workflow for characterization of platelet spreading and morphology. *Platelets*, 2020.

[74] William J. Polacheck and Christopher S. Chen. Measuring cell-generated forces: A guide to the available tools. *Nature Methods*, 13(5):415–423, 2016.

[75] Samuel R. Polio, Katheryn E. Rothenberg, Dimitrije Stamenović, and Michael L. Smith. A micropatterning and image processing approach to simplify measurement of cellular traction forces. *Acta Biomaterialia*, 8(1):82–88, 2012.

[76] Ivan Pushkarsky, Peter Tseng, Dylan Black, Bryan France, Lyndon Warfe, Cynthia J. Koziol-White, William F. Jester, Ryan K. Trinh, Jonathan Lin, Philip O. Scumpia, Sherie L. Morrison, Reynold A. Panettieri, Robert Damoiseaux, and Dino Di Carlo. Elastomeric sensor surfaces for high-throughput single-cell force cytometry. *Nature Biomedical Engineering*, 2(2):1, 2018.

[77] Andrew D. Rape, Wei Hui Guo, and Yu Li Wang. The regulation of traction force in relation to cell shape and focal adhesions. *Biomaterials*, 32(8):2043–2051, 2011.

[78] Alexandre J.S. Ribeiro, Aleksandra K. Denisin, Robin E. Wilson, and Beth L. Pruitt. For whom the cells pull: Hydrogel and micropost devices for measuring traction forces. *Methods*, 94:51–64, 2016.

[79] Claire Robertson, David D. Tran, and Steven C. George. Concise review: Maturation phases of human pluripotent stem cell-derived cardiomyocytes. *Stem Cells*, 31:829–837, 2013.

[80] Pere Roca-Cusachs, Vito Conte, and Xavier Trepat. Quantifying forces in cell biology. *Nature Cell Biology*, 19(7):742–751, 2017.

[81] Anthony G. Rodriguez, Sangyoon J. Han, Michael Regnier, and Nathan J. Sniadecki. Substrate stiffness increases twitch power of neonatal cardiomyocytes in correlation with changes in myofibril structure and intracellular calcium. *Biophysical Journal*, 101(10):2455–2464, 2011.

[82] Marita L. Rodriguez, Kevin M. Beussman, Katherine S. Chun, Melissa S. Walzer, Xiulan Yang, Charles E. Murry, and Nathan J. Sniadecki. Substrate Stiffness, Cell Anisotropy, and Cell-Cell Contact Contribute to Enhanced Structural and Calcium Handling Properties of Human Embryonic Stem Cell-Derived Cardiomyocytes. *ACS Biomaterials Science & Engineering*, page acsbiomaterials.8b01256, 2019.

[83] Marita L. Rodriguez, Brandon T. Graham, Lil M. Pabon, Sangyoon J. Han, Charles E. Murry, and Nathan J. Sniadecki. Measuring the Contractile Forces of Human Induced Pluripotent Stem Cell-Derived Cardiomyocytes With Arrays of Microposts. *Journal of Biomechanical Engineering*, 136(5):051005, 2014.

[84] Benedikt Sabass, Margaret L. Gardel, Clare M. Waterman, and Ulrich S. Schwarz.

High Resolution Traction Force Microscopy Based on Experimental and Computational Advances. *Biophysical Journal*, 94(1):207–220, 2008.

[85] U. S. Schwarz, N. Q. Balaban, D. Riveline, A. Bershadsky, B. Geiger, and S. A. Safran. Calculation of forces at focal adhesions from elastic substrate data: The effect of localized force and the need for regularization. *Biophysical Journal*, 83(3):1380–1394, 2002.

[86] Ulrich S. Schwarz and Jérôme R.D. Soiné. Traction force microscopy on soft elastic substrates: A guide to recent computational advances. *Biochimica et Biophysica Acta - Molecular Cell Research*, 1853(11):3095–3104, 2015.

[87] Dinender K. Singla, Xilin Long, Carley Glass, Reetu D. Singla, and Binbin Yan. Induced pluripotent stem (ips) cells repair and regenerate infarcted myocardium. *Molecular Pharmaceutics*, 8:1573–1581, 2011.

[88] Nathan J. Sniadecki and Christopher S. Chen. Microfabricated Silicone Elastomeric Post Arrays for Measuring Traction Forces of Adherent Cells. *Methods in Cell Biology*, 83:313–328, 2007.

[89] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, and Jon Shlens. Rethinking the inception architecture for computer vision. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016.

[90] J. L. Tan, J. Tien, D. M. Pirone, D. S. Gray, K. Bhadriraju, and C. S. Chen. Cells lying on a bed of microneedles: An approach to isolate mechanical force. *Proceedings of the National Academy of Sciences*, 100(4):1484–1489, 2003.

[91] Yo Tanaka, Keisuke Morishima, Tatsuya Shimizu, Akihiko Kikuchi, Masayuki Yamato, Teruo Okano, and Takehiko Kitamori. Demonstration of a pdms-based bio-microactuator using cultured cardiomyocytes to drive polymer micropillars. *Lab on a Chip*, 6:230, 2006.

[92] Rebecca E. Taylor, Keekyoung Kim, Ning Sun, Sung Jin Park, Joo Yong Sim, Giovanni Fajardo, Daniel Bernstein, Joseph C. Wu, and Beth L. Pruitt. Sacrificial layer technique for axial force post assay of immature cardiomyocytes. *Biomedical Microdevices*, 15:171–181, 2013.

[93] Steven G. Thomas. *The structure of resting and activated platelets*. Academic Press, London, United Kingdom, 4 edition, 2019.

[94] Iva Marija Tolić-Nørrelykke and Ning Wang. Traction in smooth muscle cells varies with cell spreading. *Journal of Biomechanics*, 38(7):1405–1412, 2005.

[95] Peter Tseng, Ivan Pushkarsky, and Dino Di Carlo. Metallization and biopatterning on ultra-flexible substrates via dextran sacrificial layers. *PLoS ONE*, 9(8), 2014.

[96] Valerie Tutwiler, Rustem I. Litvinov, Andrey P. Lozhkin, Alina D. Peshkova, Tatiana Lebedeva, Fazoil I. Ataullakhanov, Kara L. Spiller, Douglas B. Cines, and John W. Weisel. Kinetics and mechanics of clot contraction are governed by the molecular and cellular composition of the blood. *Blood*, 127(1):149–159, 2016.

[97] Catherine Vannier, Hugues Chevassus, and Guy Vassort. Ca-dependence of isometric force kinetics in single skinned ventricular cardiomyocytes from rats. *Cardiovascular Research*, 32:580–586, 1996.

[98] James G. White. Platelet structure. In *Platelets*, chapter 3, pages 45–73. Academic Press, 2 edition, 2007.

[99] Evelyn Williams, Oluwamayokun Oshinowo, Abhijit Ravindran, Wilbur Lam, and David Myers. Feeling the Force: Measurements of Platelet Contraction and Their Diagnostic Implications. *Seminars in Thrombosis and Hemostasis*, 2018.

[100] Xiulan Yang, Lil Pabon, and Charles E. Murry. Engineering adolescence: Maturation of human pluripotent stem cell-derived cardiomyocytes. *Circulation Research*, 114:511–523, 2014.

[101] Shizhuo Yin, Xueqian Zhang, Chun Zhan, Juntao Wu, Jinchao Xu, and Joseph Cheung. Measuring single cardiac myocyte contractile force via moving a magnetic bead. *Biophysical Journal*, 88:1489–1495, 2005.

[102] Jin You, Hyowon Moon, Boo Yong Lee, Ju Young Jin, Zi Eun Chang, So Yeon Kim, Jungyul Park, Yu Shik Hwang, and Jinseok Kim. Cardiomyocyte sensor responsive to changes in physical and chemical environments. *Journal of Biomechanics*, 47:400–409, 2014.

[103] Haiyang Yu, Sijing Xiong, Chor Yong Tay, Wen Shing Leong, and Lay Poh Tan. A novel and simple microcontact printing technique for tacky, soft substrates and/or complex surfaces in soft tissue engineering. *Acta Biomaterialia*, 8(3):1267–1272, 2012.

[104] Assaf Zemel, Rumi De, and Samuel A. Safran. Mechanical consequences of cellular force generation. *Current Opinion in Solid State and Materials Science*, 15(5):169–176, 2011.

Appendix A

APPENDIX

A.1 Platelet f-actin morphology classification

As part of ongoing research for platelets using the black dots approach (Chapter 3), many images of platelets on surfaces coated with either VWF or fibrinogen were collected. Platelets on these different ECMs showed different organizations of their internal F-actin structure which was characterized into three main classes: hollow, where the F-actin is primarily circumferentially organized with a hollow center region (Fig. A.1A); nodules, where F-actin forms several small dense puncta in the platelet (Fig. A.1B); and solid, where the F-actin is more central or homogeneously spread throughout the platelet (Fig. A.1C). The prevalence of these classes may change due to environmental factors such as the ECM, so we used machine learning to build a classifier that could effectively classify each platelet without user bias.

A.1.1 Building the model

To build the platelet image classifier, we used deep learning in the form of a convolutional neural network (ConvNet). Rather than develop a network from scratch, we instead applied transfer learning to the existing ConvNet *InceptionV3*, which is a model that has previously been shown to achieve good performance at general image classification using ImageNet data with 1000 classes [89]. We modified the model by replacing the last 1000-class layer with a 3-class layer representing our three platelet classes: hollow, solid, and nodules. The idea behind transfer learning is that *InceptionV3* has already learned what image features are important for differentiating between classes in a general sense, and these image features can be used to differentiate between our platelet image classes after some extra training and fine-

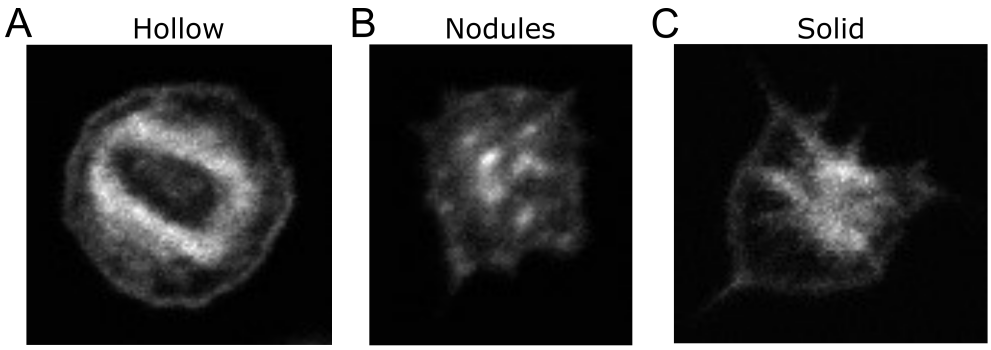


Figure A.1: Examples of platelets with F-actin morphology exhibiting (A) hollow structure, (B) nodule structure, and (C) solid structure.

tuning of the model parameters. The model was built in Python 3.8.10 using TensorFlow 2.7.0. The code for training this model is included in Appendix A.4.

A.1.2 Training the model

A data set of 1,598 images of platelet F-actin fluorescence was split into a training set (80% of images) and a validation set (20% of images). For the 1,280 images in the training set, 340 were labeled as hollow, 244 were labeled as nodules, and 696 were labeled as solid. For the 318 images in the validation set, 85 were hollow, 173 were solid, and 60 were nodules. This uneven class distribution was accounted for by applying a class weighting to the model; for example, each nodules image can affect the model parameters more strongly than each solid image because there are fewer nodules images to learn from. The raw images were cropped around the platelet with some buffer region between the platelet edge and the image edge. As part of preprocessing steps, the images were first reshaped to a standard 40-by-40 pixel image, compressing the image data and allowing the model to be used for images of different resolution as long as they are bigger than 40-by-40 pixels. The images were then normalized and contrast adjusted such that the lowest and highest brightness pixels were under- or over-saturated and all other pixels were given a value between 0-255, with the lowest 5% pixels getting a value of 0 and the highest 5% pixels getting a value of 255. These images were then

reshaped to a 299-by-299-by-3 array which is the required input shape for *InceptionV3* (the 1-channel grayscale image is stretched/shrunk and converted into a 3-channel RGB image).

The model was trained in a series of epochs where in one epoch, each image in the training set is run through the model, its class is predicted by the model, and the model's weights are updated according to how "wrong" the model prediction is using backpropagation. The model weights are updated towards an optimal state following a gradient descent approach (SGD optimizer in TensorFlow), where the learning rate dictates how much the model weights can change in each step. To artificially increase the data set size, the training images are augmented for each epoch by applying a random assortment of: rotation (up to 45°), zoom (up to 10%), translation (up to 5%), and vertical and/or horizontal mirroring; in this way, the model "sees" a bigger set of images than we actually have.

To start, only the final classification layer's model weights were allowed to change with the remaining layers of the *InceptionV3* model frozen. This classification layer was trained for 200 epochs with a learning rate that exponentially decreased from 10^{-2} to 10^{-4} over these 200 epochs. This learning rate approach allows the gradient descent algorithm to take larger steps in the first epochs when the model is poorly optimized, and smaller steps as the model reaches its convergent state. After the classification layer was trained, the entire *InceptionV3* model was fine-tuned by unfreezing everything and training all the layers for another 200 epochs at a learning rate of 10^{-4}. The model was saved after each epoch if the validation loss (i.e.,, how far apart the model's predictions are from the true validation classes) decreased from the last saved model. This training was performed on University of Washington's high-performance computing cluster *Hyak* on a single graphics processor (GPU); GPUs are highly optimized for this training due to the high number of convolutions and matrix algebra.

A.1.3 Final model performance

During training, the model performance is evaluated by predicting the classes of the validation images. Unlike the training images, the validation images are not augmented and are meant to be similar to real images that the model will work on. The accuracy of the

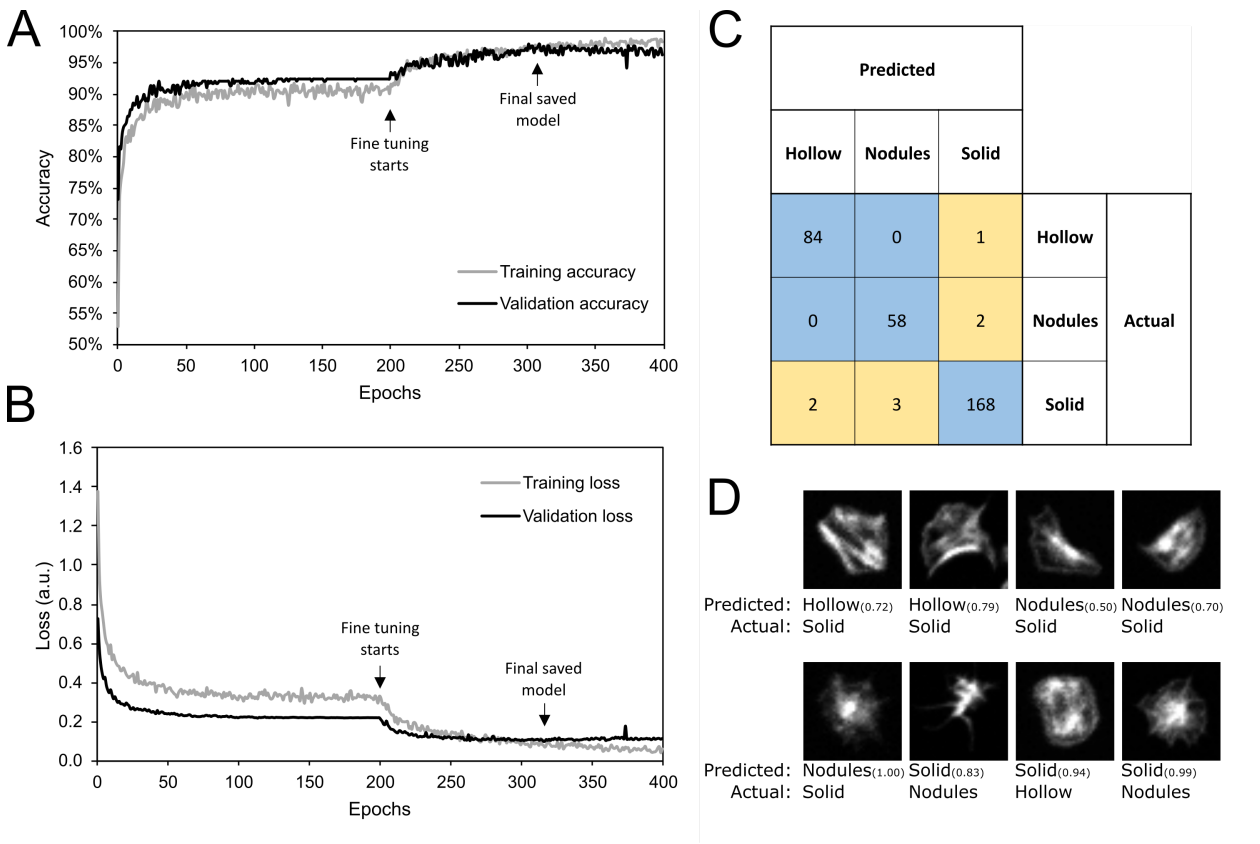


Figure A.2: Performance of the image classifier for classifying platelet F-actin morphology. (A) The model accuracy and (B) the model loss on the training and validation data over the 400 training epochs. At epoch 200, the model is unfrozen fine-tuned. The model is saved whenever the model loss decreases; the lowest model loss and final saved moel is indicated by arrow on both plots. (C) Confusion matrix for the final model, indicating how the model performed for the different classes on the validation data. Blue shaded areas indicate where the model made the correct predictions (or lack of misclassified images), while yellow shaded areas indicate where the model misclassified some images (i.e., predicted a different class than actual). (D) All of the misclassified images from the validation data for the final model.

model at correctly predicting the validation image classes is the primary metric of model performance; as the model is trained, the accuracy should increase and the loss (the accumulated "difference" between the predicted classes and actual classes) should decrease as shown in Figure A.2A-B. Under the training scheme outlined here, the validation accuracy converged to 92.3% in the first 200 epochs, when only the final classification layer was allowed to change. After unfreezing all of the layers and training for another 200 epochs, to final model yielded a 97.6% validation accuracy. A confusion matrix showing how the model predicted classes for the validation data is shown in Figure A.2C; the correct predictions are located on the diagonal of this matrix, while the incorrect predictions are off-diagonal. It can be helpful to look at the misclassified images to see if there is any consistent reason for the incorrect predictions. Looking at the validation images that were misclassified in Figure A.2D, there does not appear to be any systemic issue causing the misclassifications – rather, these cells appear by eye to contain features from multiple classes or could even be considered mislabeled in the original data set. This is reassuring as the morphology of platelet F-actin likely exists on a continuum and thus, in-between states are probably not uncommon and may be difficult to classify. Despite this, the model has a very high accuracy on the validation data and is able to distinguish between hollow, nodules, and solid classes from this relatively small data set.

A.1.4 Applying the model to new images

The model is saved as a single .h5 file, about 100 MB, that can be imported into most installations of TensorFlow in Python. MATLAB contains several functions for importing TensorFlow networks, but they are currently incompatible with models created in recent versions of TensorFlow (>2.6.0); TensorFlow 2.7.0 was used to create this model. Therefore, inferring classes on new images is best performed on Python. For new users, it is suggested to start by first downloading a basic installation of miniconda and then installing the required TensorFlow packages. The code for this project contains a Python script for classifying new images; the script loads and preprocesses the new images into an "ImageDataGenerator",

loads the model, uses the model to predict which class each image belongs to, and then outputs the preprocessed images with their classification into two folders, "classifications" csv file.

A.2 Other Projects

Throughout my years at the University of Washington, I have had the privilege of collaborating on many different and exciting projects. In this section, I summarize several of these projects and outline my contributions to them.

A.2.1 Whole-cell and EHT calcium analysis

Early on in my PhD, I worked on a better way to analyze calcium transients from videos of fluorescent calcium in cardiomyocytes (e.g., GCaMP-3, Fluo-4, or Fura-2). Historically, calcium transients were typically measured by drawing a line through the cell, and measuring the mean or total fluorescence intensity on that line over time. This was done because many of the original measurement techniques used a quick-scanning laser that only measured fluorescence along a single line. However, modern high-speed cameras are capable of capturing videos of cell fluorescence that is no longer bound by a single line. I developed an analysis code that could segment single-cell or multi-cell area based on basic image thresholding and user-drawn lines complete with a somewhat intuitive graphical user interface (GUI). After getting the cell boundary, the algorithm measures the mean fluorescence within that area for each video frame, then automatically finds the start, peak, and 50% and 90% relaxation times for each beat. Due to the larger area of measurement, rather than a single line, the resulting transient was often very smooth and relatively easy to obtain these calcium measurements without extra processing. The algorithm and process has been used in several publications with one in preparation [6, 64, 82]. The code was also expanded to engineered heart tissues (EHTs) by automatically changing the EHT boundary for each frame, because the EHT deformation is large and the boundary does not remain constant throughout the video [53].

A.2.2 Excitation-contraction coupling in cardiomyocytes

One of my larger projects early on was to investigate excitation-contraction coupling in stem cell-derived cardiomyocytes. I collected calcium fluorescence videos for cardiomyocytes on microposts, thereby measuring the excitation (calcium) and contraction (force). By measuring the local "activation" in both the calcium video and the force video, we could in theory measure the coupling between excitation and contraction, and eventually study this coupling as a function of cell maturity or environmental stimulus. In retrospect this project was somewhat aimless, and more observational than hypothesis-driven. Still, one of the roadblocks we encountered was a failure to electrically stimulate individual cardiomyocytes on microposts which ultimately led to the invention of the black dots approach shown here. I was also fortunate to present this calcium-micropost data at the 2016 Keystone conference in front of the entire conference, which was a hugely humbling and rewarding experience.

In collaboration with Dr. Ashley Emery, we developed a finite element model based on literature to simulate local sarcomere shortening due to calcium concentration. This model allowed for spatial propagation of calcium waves due to diffusion, that would then locally cause contraction and force generation. This work was ultimately presented at the 2016 Summer Biomechanics, Bioengineering, and Biotransport Conference.

A.2.3 FRET tension sensors in iPSC-CMs

Stem cell-derived cardiomyocytes are immature and weak as compared to their adult counterparts. Uncovering how these immature cells generate forces is a huge challenge and was a major motivation in developing the black dots approach. However, force sensing assays like the black dots platform are limited to measuring external forces, and are ultimately blinded to the internal force generation in cells. FRET tension sensors have been developed to illuminate these internal forces; briefly, proteins involved in cell adhesion (e.g., vinculin) can be modified to incorporate a FRET pair such that the FRET efficiency decreases when the protein is pulled apart. These probes had been used to investigate forces in fixed cells,

but not in stem cell-derived cardiomyocytes. I performed some initial preliminary work by using an lentivirus to express the FRET tension sensor in hiPSC-CMs and recording videos of these cells beating on glass and PDMS surfaces. I also troubleshooted different imaging possibilities as we were inexperienced with this type of imaging. While we never got a clear, genuine FRET signal from focal adhesions expressing this probe, this work served as preliminary data and led Abby Nagle to create an endogenously-expressing FRET-vinculin iPSC line. For future directions, it would be exciting to use the FRET probe in combination with the black dots platform to help calibrate the internal tension sensors by measuring the external forces, or possibly to use machine learning approaches to predict external forces from the internal sensors or vice versa.

A.2.4 Rainbow reporter for stem cells and stem cell-derived cardiomyocytes

To help understand how cells proliferate and give rise to progeny, Danny El-Nachef developed a rainbow reporter of stem cells that could express three different fluorescent markers in differing quantities when treated with Cre recombinase (Cre) [28]. The cell line has four copies of the gene that creates these fluorescent markers, resulting in 18 different hue possibilities. I developed an algorithm to automatically extract the hues present in images of these cells, ultimately to show that higher Cre resulted in a more diverse color spectrum (indicating that more cells had created multiple different colored probes). I also created visualizations of cell migration to better show how cells proliferated and give rise to daughter cells.

Later, in collaboration with both Danny El-Nachef and Darrian Bugg, we investigated stem cell-derived cardiomyocyte proliferation *in vivo* in grafted rat hearts [27]. Because cell progeny express the same color as the parent cells, these rainbow reporting cells can be effectively used to track the lineage of cells *in vivo*. Using 10X Genomics Visium spatial gene expression protocols, we attempted to measure the gene profiles of local cell clusters to determine if gene expression was tied to cell proliferation. I was responsible for running the collected gene expression data through 10X Genomics Space Ranger pipeline and, with the help of University of Washington's high performance computing cluster Hyak, to analyze

and aggregate the data for viewing. Ultimately, the Visium system was not mature enough at the time, and the lack of official protocols for using Visium on cardiomyocytes resulted in transcription read counts that were too low for any conclusions to be made. We were, however, able to analyze some publicly available single-cell RNA-seq data for differently aged cardiomyocytes using 10X Genomics Cell Ranger pipeline which did not get included in the final publication but was helpful in confirming our internal hypotheses.

A.2.5 Conical heart tissue finite element modeling

In an effort to better mimic native heart tissue, Nisa Williams developed an engineered 3D conical heart tissue composed of a thick, hollow inner fibrin cone wrapped with two sheets of aligned hiPSC-CMs. With this type of engineered heart tissue, the helical architecture of adult hearts could be mimicked and investigated by altering the orientation of these sheets and measuring the tissue force output. The individual sheet layers could have one of four different orientations: circumferential, longitudinal, angled (45°), or isometric (random) alignment. Curiously, tissues made with circumferentially aligned cells tended to reorient to a longitudinal alignment after just four days of culture; tissues that started with isometric or longitudinal alignment did not show any reorientation. To help understand this phenomenon, I created an axisymmetric finite element model to simulate the heart tissue architecture. The two tissue layers were allowed to contract by applying an anisotropic thermal expansion coefficient that would cause the layers to contract in any chosen direction when subjected to a negative temperature. The thermal expansion coefficient was tuned using experimental data to get a realistic contraction magnitude. Using this computational model, we discovered that the circumferential and longitudinal alignments caused high shear stress in the longitudinal direction at the cell-fibrin interface, while the isometric and angled orientation had low shear stress. We ultimately concluded that this shear stress perpendicular to the circumferentially aligned cells caused them to reorient in the direction of the shear; the longitudinal aligned cells were already aligned in that direction and did not need to reorient, and the isometric and angled orientations did not generate enough shear to cause reorientation. This work has

been included in a manuscript that has been submitted to Nature Materials.

A.2.6 Melusin in hiPSC-CMs

Mechanosensitive proteins are able to respond to mechanical stimulus and give rise to a cellular response. Several mechanosensitive proteins have been identified in cardiomyocytes, but one in particular, melusin or ITGB1BP2, has not been well studied. Melusin is a CHORD-containing protein that resides in cellular adhesions where it is able to sense force and cause downstream effects. It had been previously shown by Mara Brancaccio that melusin colocalizes with sarcomeric z-discs in neonatal rat cardiomyocytes, presumably in the costameres where the myofibrils link to the extracellular environment [10]. In collaboration with Gabby Nattenberg, Ruby Padgett, Alessandro Bertero, and Mara Brancaccio's team, we sought to investigate melusin in immature stem cell-derived cardiomyoctes in the context of heart disease and heart development. One step in this process was to visualize the localization of melusin in hiPSC-CMs to both confirm our melusin antibody specificity as well as to investigate where melusin localizes in human cardiomyocytes. I was provided with the two original melusin primary antibodies developed by Mara Brancaccio, 5E1 and C3, for use in immunofluorescent studies on hiPSC-CMs. Our expectations were that melusin would form striated bands corresponding tot he sarcomeric z-discs, but instead we observed diffuse melusin signal throughout the cells. We suspected that the cells were simply not mature enough, and that external maturation by modulating substrate stiffness, cell alignment, or electrical stimulus could help develop well-defined myofibrils, resulting in stronger costameres containing more melusin. However, none of these maturation methods consistently yielded striated melusin staining, nor did using a different embryonic stem cell line to generate cardiomyocytes (RUES2). To ensure the problem was not in the antibody itself, I searched commercial vendors for all available ITGB1BP2 antibodies and chose five that localized to different epitopes of the melusin protein. We tested these commercial antibodies as well as the original 5E1 and C3 antibodies on stem cells that exhibited normal, overexpressing, or knocked out melusin. Of these antibodies, only 5E1 and C3 showed specificity to melusin,

with all other antibodies having no correlation to the melusin expression. For future work, repeating Mara's original results in mouse heart tissue to confirm the antibody viability as well as staining EHTs to try and observe the elusive melusin striations are the primary focus.

A.3 Black dots analysis code

This section contains the core analysis scripts and functions for analyzing images or videos of black dots. The full code contains other minor functions and visualizations and is available upon request.

A.3.1 Main analysis function

```
%% black dots analysis
clearvars; clc; close all;

uM.force_user_vals = false; % set this to 'true' to if you want to use the below values instead of the ND2 values
uM.analyzeVideo = true; % are you analyzing a video or an image? set true for video
uM.verbose = 2; % 0 = none, 1 = display metadata in command window, 2 = display metadata in editable popup window

% these following settings are used only if they cannot be pulled from the video data
uM.CameraPixelSize = 6.5; % physical pixel size [um], from camera manufacturer V3=6.5, Flash2.8=3.63, Andor=6.45
uM.Objective = 60; % e.g. 60x objective
uM.Binning = 2; % e.g. 1x1 binning
uM.FrameRate = 20; % [Hz]
uM.CouplerRatio = 0.7; % e.g. 0.7x coupler
uM.MagMultiplier = 1; % e.g. 1.5x magnification toggle

% video settings (ignored if analyzeVideo is false)
% uM.Frames = 1:50; % video frames to analyze. comment this out to use all frames
uM.uFrame = 55; % which video frame to use for relaxed frame -- this is relative to uM.Frames

% image settings (ignored if analyzeVideo is true)
uM.BDchannel = 1; % channel to use for black dots
% uM.CBchannel = 1; % optional channel to use for cell boundary. comment this out if there is only one channel
```

```
23
24 % DotSize/DotSpacing
25 % N1: 1.2/2.0 or 1.0/1.95 or 0.8/1.95
26 % 2E: 3.0/6.0
27 % 2A: 3.5/8.0
28
29 % PDMS properties:
30 % 0% = 5 kPa
31 % 5% = 13.5 kPa
32 % 10% = 47.8 kPa
33
34 uM.DotSize = 3; % [um] microns
35 uM.DotSpacing = 6; % [um] microns
36 uM.YoungsModulus = 7700; % [N*m^-2] or equivalently [pN*um^-2]
37 uM.Poisson = 0.5; % [] unitless
38
39 % image boundary and cropping
40 uM.Crop = false; % true or give crop boundary[xmin ymin, width height]
41 % uM.crop_around_cells = true;
42 uM.manual_boundary = true;
43
44 % pattern tracking and force calculation settings
45 uM.bright_dots = false; % this code tracks dark dots. Set "true" here to invert image before
       analyzing
46 uM.track_method = 1; % method 1 = more manual method, method 2 = more automated method
47 uM.uPoints = 2; % number of outer undeformed points in each direction to use for grid
      estimation. comment out to use all points
48 uM.useLcurve = false; % calculate regularization parameter from L curve (takes a while)
49 uM.regParam = 5e-8; % regularization parameter guess
50
51 %% Choose file to analyze
52 file_script = mfilename('fullpath');
53 [path_script,~,~] = fileparts(file_script);
54 addpath(path_script)
55 addpath([path_script filesep 'bftools'])
56 addpath([path_script filesep 'bfmatlab'])
57 addpath([path_script filesep 'track'])
58 addpath([path_script filesep 'blackdots_functions'])
59
60 [file_BD, path_BD] = uigetfile({'*.nd2','NIS Elements';'*.cxd','HCImage Live';'*.avi','(AVI)
       Audio Video Interleave';'*.tif;*.tiff','TIFF Image Stacks'},'Select file to analyze.
      NOTE: .nd2 files require bioformats plugin.','MultiSelect','off');
```

```
61 cd(path_BD)
62
63 if uM.analyzeVideo
64     [img_BD,meta_BD] = read_video([path_BD, file_BD],uM);
65 else
66     if isfield(uM,'CBchannel')
67         [img_BD,meta_BD,img_CB] = read_image([path_BD, file_BD],uM);
68     else
69         [img_BD,meta_BD] = read_image([path_BD, file_BD],uM);
70     end
71 end
72 if uM.bright_dots
73     img_BD = imcomplement(img_BD); % use this if dots are bright instead of dark
74 end
75 img_BD_raw = img_BD;
76 meta_BD_raw = meta_BD;
77
78 [~,filename,~] = fileparts(file_BD);
79
80 % create save folder(s)
81 cur_time = char(datetime('now','format','yyyyMMdd_HHmmss'));
82
83 path_save = [path_BD filename '_blackdots_analysis' filesep];
84 if ~exist(path_save,'dir')
85     mkdir(path_save)
86 end
87
88 path_save = [path_save cur_time filesep];
89 if ~exist(path_save,'dir')
90     mkdir(path_save)
91 end
92
93 file_save = ['blackdots_data_' cur_time '.mat'];
94
95 %% Find cell boundary
96 [file_CB_data, path_CB_data] = uigetfile({'*.mat'},'Select mat file containing cell boundary
       coordinates (a previous blackdots_analysis file).');
97 if file_CB_data ~= 0
98     load([path_CB_data file_CB_data],'CB_uncrop','img_CB');
99
100     if meta_BD.Crop ~= false
101         CB = cellfun(@plus,CB_uncrop,{-meta_BD.Crop([1 2])},'UniformOutput',false);
```

```
    else
        CB = CB_uncrop;
    end

     save([path_save file_save],'CB_uncrop','img_CB')
else
    if meta_BD.analyzeVideo || ~isfield(meta_BD,'CBchannel')  % look for another image file
        to get the cell boundary
        [file_CB, path_CB] = uigetfile({'*.nd2','NIS Elements';'*.cxd','HCImage Live';'*.avi
            ','(AVI) Audio Video Interleave';'*.tif;*.tiff','TIFF Image Stacks'},'Select
            image file to use for tracing cell boundary.','MultiSelect','off');
        if file_CB ~= 0
            [img_CB,meta_CB] = read_image([path_CB file_CB]);
            if meta_BD.manual_boundary
                CB = get_cell_boundary(img_CB); % manual clicking boundary
            else
                CB = get_cell_boundary2(img_CB,meta_BD); % semi-automatic segmentation
            end
        else % if no other image is selected, default crop is to just use the whole image
            CB{1} = [1,1; meta_BD.N,1; meta_BD.N,meta_BD.M; 1,meta_BD.M];
            img_CB = img_BD_raw(:,:,1);
            img_CB(:,:) = 0;
        end
        CBraw = CB;
        img_CB_raw = img_CB;
        if meta_CB.Binning ~= meta_BD.Binning % CB and BD images have different binning/
            sizes, do some correction
            for ic = 1:length(CB)
                CB{ic} = CB{ic}*(meta_CB.Binning/meta_BD.Binning);
            end
            img_CB = imresize(img_CB,meta_CB.Binning/meta_BD.Binning);
        end
    else % we already got the cell boundary image from current video/image in CBchannel
        if meta_BD.manual_boundary
            CB = get_cell_boundary(img_CB); % manual clicking boundary
        else
            CB = get_cell_boundary2(img_CB,meta_BD); % semi-automatic segmentation
    %         stats = get_cell_boundary_zizhen(cbFrame,meta_BD); % doesnt work on kb2
        computer
        end
    end

```

```
    if meta_BD.Crop ~= false
        CB_uncrop = cellfun(@plus,CB,{meta_BD.Crop([1 2])},'UniformOutput',false);
    else
        CB_uncrop = CB;
    end

    save([path_save file_save],'CB_uncrop','img_CB')
end

%% Bandpass filter image or all video frames
fprintf('Filtering black dots image(s)...')

bpass_noise = 1; % bandpass characteristic noise length (leave at 1 usually)
dotsize_px = 2*round((meta_BD.DotSize/meta_BD.Calibration + 1)/2) - 1; % dot size in pixels
img_BD_filt = zeros(size(img_BD));
for k = 1:meta_BD.nFrames
    img_BD_filt(:,:,k) = mat2gray(imcomplement(bpass_kb2(imcomplement(img_BD(:,:,k)),
        bpass_noise,dotsize_px)));
end

fprintf('DONE\n')

%% start analysis for each cell
nCells = length(CB);
fprintf('\t%i objects (cells) detected\n',nCells)
skipped = false(nCells,1);
celldata(nCells) = struct();
for ic = 1:nCells
    %% crop to selected cell
    fprintf('Cropping black dots image(s) around cell...')

    % expands crop region to include some undeformed dots outside cell
    celldata(ic).crop_tight = [min(CB{ic}), max(CB{ic})-min(CB{ic})];
    celldata(ic).crop = celldata(ic).crop_tight;
    last_crop = celldata(ic).crop;

    expand = meta_BD.DotSize/meta_BD.Calibration*[-1 -1 2 2]; % crop boundary expands by
        this much every time "expand" is clicked

    fig_cellselect = figure('Units','Normalized','Position',[0.2 0.1 0.6 0.6]);
    ax1 = subplot(1,2,1);
    im1 = imagesc(img_BD_filt(:,:,meta_BD.uFrame));
```

```
179     axis image
180     hold on
181     plot(CB{ic}(:,1),CB{ic}(:,2),'-r')
182     hold off
183     colormap(ax1,gray*[1 0 0; 0 0.75 0; 0 0 0])
184
185     ax2 = subplot(1,2,2);
186     imagesc(img_CB);
187     axis image
188     hold on
189     plot(CB{ic}(:,1),CB{ic}(:,2),'-r')
190     hold off
191     colormap(ax2,gray*[0 0 0; 0 0.75 0; 0 0 1])
192     linkaxes([ax1 ax2])
193     axis(celldata(ic).crop_tight([1 1 2 2]) + [0, celldata(ic).crop_tight(3), 0, celldata(ic
            ).crop_tight(4)] - 0.5)
194
195     but_expand = uicontrol(fig_cellselect,'Units','Normalized','Position',[0 0 0.2 0.1],'
            Style','PushButton','String','expand',...
196         'Callback',['celldata(ic).crop = celldata(ic).crop + expand;', ...
197                     'if celldata(ic).crop(1) < 1, celldata(ic).crop(1) = 1; end;', ...
198                     'if celldata(ic).crop(2) < 1, celldata(ic).crop(2) = 1; end;', ...
199                     'if sum(celldata(ic).crop([2 4])) > meta_BD.M, celldata(ic).crop(4) =
                            meta_BD.M - celldata(ic).crop(2) + 1; end;', ...
200                     'if sum(celldata(ic).crop([1 3])) > meta_BD.N, celldata(ic).crop(3) =
                            meta_BD.N - celldata(ic).crop(1) + 1; end;', ...
201                     'axis(celldata(ic).crop([1 1 2 2]) + [0, celldata(ic).crop(3), 0,
                            celldata(ic).crop(4)] - 0.5);', ...
202                     'last_crop = [last_crop; celldata(ic).crop];']);
203
204     but_undo = uicontrol(fig_cellselect,'Units','Normalized','Position',[0.2 0 0.2 0.1],'
            Style','PushButton','String','undo',...
205         'Callback',['if size(last_crop,1) > 1, last_crop(end,:) = []; end;', ...
206                     'celldata(ic).crop = last_crop(end,:);', ...
207                     'axis(celldata(ic).crop([1 1 2 2]) + [0, celldata(ic).crop(3), 0,
                            celldata(ic).crop(4)] - 0.5);']);
208
209     but_ok = uicontrol(fig_cellselect,'Units','Normalized','Position',[0 0.1 0.2 0.1],'Style
            ','PushButton','String','OK','Callback','uiresume;');
210
211     but_skip = uicontrol(fig_cellselect,'Units','Normalized','Position',[0.2 0.1 0.2 0.1],'
            Style','PushButton','String','skip',...
```

```
212         'Callback','uiresume; skipped(ic) = true;');
213
214     uiwait
215
216     close(fig_cellselect)
217
218     if skipped(ic) % if this cell is to be skipped, immediately move to next cell
219         continue
220     end
221
222     % note: using imcrop is slow compared to this
223     % xrange and yrange are just the range of rows/columns to keep
224     % (identical to what you get out of imcrop)
225     % celldata(ic).crop is float, so we need to round to nearest pixel
226     xrange = celldata(ic).crop([2,4]); xrange(2) = round(xrange(2) + xrange(1) - 1); xrange
            (1) = round(xrange(1));
227     yrange = celldata(ic).crop([1,3]); yrange(2) = round(yrange(2) + yrange(1) - 1); yrange
            (1) = round(yrange(1));
228
229     img_BD_crop = img_BD(xrange(1):xrange(2),yrange(1):yrange(2),:);
230     img_BD_filt_crop = img_BD_filt(xrange(1):xrange(2),yrange(1):yrange(2),:);
231
232     img_REFBD = img_BD(:,:,meta_BD.uFrame);
233     img_REFBD_crop = img_BD_crop(:,:,meta_BD.uFrame);
234     img_REFBD_filt_crop = img_BD_filt_crop(:,:,meta_BD.uFrame);
235
236     celldata(ic).M = size(img_BD_filt_crop,1);
237     celldata(ic).N = size(img_BD_filt_crop,2);
238     celldata(ic).CB = CB{ic} - celldata(ic).crop([1 2]);
239
240     fprintf('DONE\n')
241
242     %% Start analysis
243     if meta_BD.track_method == 1 % more manual method of tracking dots
244         %% Measure rotation angle from image
245         fprintf('Measuring rotation angle...')
246
247         DotSpacing_px = meta_BD.DotSpacing/meta_BD.Calibration;
248
249         celldata(ic).rot_angle = get_rot_from_img(img_REFBD_filt_crop,DotSpacing_px);
250
251         fprintf('DONE\n')
```

```
252
253         %% Choose the sampling dots
254         % allows user to estimate dot grid positions
255         fprintf('Choosing dots...')
256
257         [px_sample,py_sample] = choose_dots(img_REFBD_filt_crop,celldata(ic),meta_BD);
258
259         fprintf('DONE\n')
260
261         if isempty(px_sample)
262             skipped(ic) =  true;
263             continue
264         end
265
266         %% Update rotation angle
267         fprintf('Updating rotation angle...')
268
269         celldata(ic).rot_angle = get_rot_from_gridpts(cat(3,px_sample,py_sample));
270
271         fprintf('DONE\n')
272
273         %% Extend the sampling dots
274         % fits a grid to the sampling dots, and extends the grid of dots
275         % to fill the full image
276         fprintf('Extending dots to full image...')
277
278         celldata.real_points = []; % for debugging
279         [px,py,real_points,img_REFBD_bw] = extend_dots(px_sample,py_sample,
                img_REFBD_filt_crop,celldata(ic),meta_BD);
280
281         celldata(ic).real_points = real_points;
282
283         fprintf('DONE\n')
284
285         %% Characterize dots
286         fprintf('Characterizing dots...')
287
288         img = img_REFBD_filt_crop;
289         img_filt = imgaussfilt(img,1);
290         img_filt_bw = imfill(imcomplement(imbinarize(img_filt)),'holes');
291         img_REFBD_bw = bwselect(img_filt_bw,px(real_points),py(real_points));
292
```

```
293         BD = calc_dot_size_spacing(px,py,img_REFBD_bw,celldata(ic),meta_BD);
294
295         celldata(ic).BD = BD;
296
297         fprintf('DONE\n')
298
299         %% Find undeformed dot positions
300         fprintf('Finding undeformed dot centroids...')
301
302         [px,py,px0,py0] = find_undeformed(px,py,celldata(ic),meta_BD);
303
304         fprintf('DONE\n')
305
306         %% Track dots across all frames
307         % just basically repeats find_centroids for each frame
308         fprintf('Finding dot centroids in each image frame...')
309
310         % this should really call find_centroids for each frame
311         if meta_BD.analyzeVideo
312             [px_k,py_k,real_points,px0,py0] = track_dots_across_frames(img_BD_filt_crop,px,
                    py,px0,py0,celldata(ic),meta_BD);
313
314             celldata(ic).real_points = real_points; %just in case some dots are lost when
                    analyzing
315             px = px_k(:,:,meta_BD.uFrame);
316             py = py_k(:,:,meta_BD.uFrame);
317         else
318             px_k = px(:);
319             py_k = py(:);
320         end
321
322         fprintf('DONE\n')
323
324     elseif meta_BD.track_method == 2
325         % this method uses the "track" package from http://site.physics.georgetown.edu/
                matlab/
326         if ~exist('track','file')
327             error('''track'' not found, visit http://site.physics.georgetown.edu/matlab/')
328         end
329
330         %% Find object centers
331         fprintf('Finding dot centers in each image frame...')
```

```
332
333         dotsize = 2*round((meta_BD.DotSize/meta_BD.Calibration + 1)/2) - 1; % pixels,
                nearest odd integer
334         dotspacing = round(meta_BD.DotSpacing/meta_BD.Calibration);
335
336         bpass_noise = 1; % bandpass noise: 1 pixel is typical      % 1*meta_BD.Calibration
337         pkfnd_threshold = 0.15; % object peak threshold: find by trial and error, 0.15 is
                good
338
339
340         centers = []; % centers has 3 columns: [x, y, frame#]. we don't know how many
                objects there will be
341         pct = 0;
342         fprintf('[%-20s] %3.0f%%\n',repmat('|',1,round(pct/100*20)),pct)
343         for k = 1:meta_BD.nFrames
344             pct = k/meta_BD.nFrames;
345             fprintf([repmat('\b',1,28) '[%-20s] %3.0f%%\n'],repmat('|',1,round(pct*20)),pct
                    *100)
346             I1 = img_BD_crop(:,:,k);
347             I2 = bpass_kb2(imcomplement(I1),bpass_noise,dotsize); % spatial bandpass filter
348             pk = pkfnd_kb2(mat2gray(I2),pkfnd_threshold,dotsize); % find objects whose
                    intensity is above threshold
349             dsz = ceil(dotsize*1.2); if ~mod(dsz,2), dsz = dsz + 1; end % KB2 edit 9/3/2020
350             cnt = cntrd_kb2(I2,pk,dsz); % KB2 edit 9/3/2020
351
352             centers = [centers; cnt(:,1:2), ones(size(cnt,1),2)*k];
353         end
354         fprintf(repmat('\b',1,28))
355
356 %         figure
357 %         imagesc(img_REFBD_crop)
358 %         hold on
359 %         plot(centers(:,1),centers(:,2),'.r','markersize',10)
360 %         hold off
361
362         fprintf('DONE\n')
363
364         %% Track objects across video frames
365         fprintf('Tracking centers...')
366
367         if meta_BD.analyzeVideo
368
```

```
369             param.mem = 5; % mem = # of "dropped" frames before calling it a new object
370             param.good = 0; % good = ?
371             param.dim = 2; % dim = dimensions, (x, y)
372             param.quiet = 0; % quiet = no command line feedback
373 
374             res = track(centers,floor(dotspacing/4),param);
375             res = sortrows(res,5);% res has 5 columns: [x, y, frame#, frame#, object#]
376         else
377             res = [centers(:,1:2), ones(size(centers,1),2), (1:size(centers,1))'];
378         end
379 
380         fprintf('DONE\n')
381 
382         %% Fill in gaps in time for tracked points
383         fprintf('Filling in temporal gaps...')
384 
385         if meta_BD.analyzeVideo
386             [px,py] = fill_in_temporal_gaps(res,celldata(ic),meta_BD);
387         else
388             px = res(:,1);
389             py = res(:,2);
390         end
391 
392         fprintf('DONE\n')
393 
394         %% Manual point removal
395         fig_1 = figure('units','normalized','position',[0.1 0.1 0.8 0.8]);
396         ax_1 = axes;
397         im_1 = imagesc(imadjust(mat2gray(img_BD_crop(:,:,meta_BD.uFrame))));
398         axis image
399     %     colormap(gray)
400         hold on
401         p_dot = plot(px(:,meta_BD.uFrame),py(:,meta_BD.uFrame),'.r','markersize',10);
402         p2_dot = plot(px(1,meta_BD.uFrame),py(1,meta_BD.uFrame),'or');
403         p_dot_remove = plot(0,0,'.k','markersize',10);
404         hold off
405 
406         set(im_1,'hittest','off')
407         set(p_dot,'hittest','off')
408         set(p2_dot,'hittest','off')
409 
```

```
410         fcn1_1 = @(a,b) set(fig_1,'UserData',find(sqrt((px(:,meta_BD.uFrame) - ax_1.
                CurrentPoint(1,1)).^2 + (py(:,meta_BD.uFrame) - ax_1.CurrentPoint(1,2)).^2) ==
                min(sqrt((px(:,meta_BD.uFrame) - ax_1.CurrentPoint(1,1)).^2 + (py(:,meta_BD.
                uFrame) - ax_1.CurrentPoint(1,2)).^2))));
411         fcn1_2 = @(a,b) set(p2_dot,'xdata',px(fig_1.UserData,meta_BD.uFrame),'ydata',py(
                fig_1.UserData,meta_BD.uFrame));
412         fcn1 = @(a,b) cellfun(@feval,{fcn1_1 fcn1_2});
413
414         fcn2_1 = @(a,b) set(fig_1,'UserData',[fig_1.UserData; find(sqrt((px(:,meta_BD.uFrame
                ) - ax_1.CurrentPoint(1,1)).^2 + (py(:,meta_BD.uFrame) - ax_1.CurrentPoint(1,2))
                .^2) == min(sqrt((px(:,meta_BD.uFrame) - ax_1.CurrentPoint(1,1)).^2 + (py(:,
                meta_BD.uFrame) - ax_1.CurrentPoint(1,2)).^2)))]);
415         fcn2_2 = @(a,b) set(p2_dot,'xdata',px(fig_1.UserData,meta_BD.uFrame),'ydata',py(
                fig_1.UserData,meta_BD.uFrame));
416         fcn2 = @(a,b) cellfun(@feval,{fcn2_1 fcn2_2});
417
418         fcn3 = @(a,b) set(ax_1,'buttondownfcn',fcn2);
419         fcn4 = @(a,b) set(ax_1,'buttondownfcn',fcn1);
420
421         set(ax_1,'buttondownfcn',fcn1)
422         set(fig_1,'KeyPressFcn',fcn3,'KeyReleaseFcn',fcn4)
423
424         p_remove = false(size(px));
425         but_remove = uicontrol('parent',fig_1,'units','normalized','Style','pushbutton',...
426             'Position',[0.8 0 0.2 0.05],'HorizontalAlignment','Center','FontUnits','
                    normalized','FontSize',0.8,...
427             'String','REMOVE','Callback','p_remove(fig_1.UserData) = true; set(p_dot_remove,
                    ''xdata'',px(p_remove,meta_BD.uFrame),''ydata'',py(p_remove,meta_BD.uFrame))
                    ;');
428
429         but_done = uicontrol('parent',fig_1,'units','normalized','Style','pushbutton',...
430             'Position',[0.8 0.95 0.2 0.05],'HorizontalAlignment','Center','FontUnits','
                    normalized','FontSize',0.8,...
431             'String','DONE','Callback','uiresume');
432
433         uiwait
434
435         px(p_remove,:) = [];
436         py(p_remove,:) = [];
437
438         close(fig_1)
439
```

```
440         %% get rotation angle based on dot locations
441         fprintf('Finding rotation angle...')
442
443         DotSpacing_px = meta_BD.DotSpacing/meta_BD.Calibration;
444
445         if meta_BD.analyzeVideo
446             celldata(ic).rot_angle = get_rot_from_pts([px(:,meta_BD.uFrame),py(:,meta_BD.
                    uFrame)],DotSpacing_px);
447         else
448             celldata(ic).rot_angle = get_rot_from_pts([px,py],DotSpacing_px);
449         end
450         fprintf('DONE\n')
451
452         %% find undeformed state of grid
453         fprintf('Finding undeformed state of grid...')
454
455         [px_k,py_k,px0,py0,real_points] = find_undeformed_grid(px,py,celldata(ic),meta_BD);
                % IF YOU HAVE A LOT OF DISPLACEMENT, CHANGE INF TO 2 to find fewer points
456         px = px_k(:,:,meta_BD.uFrame);
457         py = py_k(:,:,meta_BD.uFrame);
458         celldata(ic).real_points = real_points;
459
460         fprintf('DONE\n')
461
462         %% calculate dot size and spacing from image
463         fprintf('Characterizing dots...')
464
465         BD = calc_dot_size_spacing(px,py,img_REFBD_bw,celldata(ic),meta_BD);
466
467         celldata(ic).BD = BD;
468
469         fprintf('DONE\n')
470
471     end
472
473     %% Filter locations
474     fprintf('Filtering dot positions...')
475
476     Xloc_k = reshape(px_k,[],meta_BD.nFrames);
477     Xvector = reshape(px0,[],1);
478
479     Yloc_k = reshape(py_k,[],meta_BD.nFrames);
```

```
480     Yvector = reshape(py0,[],1);
481
482     % filter
483     uHz = 4; % lowpass cutoff
484     % lHz = 0.5;
485     if meta_BD.nFrames >= 2
486     %     [B,A] = butter(3,[lHz/(meta_BD.FrameRate/2), uHz/(meta_BD.FrameRate/2)],'bandpass
            ');
487         [B,A] = butter(3,uHz/(meta_BD.FrameRate/2),'low');
488
489         Xloc_k_filt = NaN(size(Xloc_k));
490         Yloc_k_filt = Xloc_k_filt;
491         for pt = 1:size(Xloc_k,1)
492             Xloc_k_temp = [Xloc_k(pt,end:-1:1), Xloc_k(pt,:), Xloc_k(pt,end:-1:1)];
493             Yloc_k_temp = [Yloc_k(pt,end:-1:1), Yloc_k(pt,:), Yloc_k(pt,end:-1:1)];
494             Xloc_k_temp_filt = filtfilt(B,A,Xloc_k_temp);
495             Yloc_k_temp_filt = filtfilt(B,A,Yloc_k_temp);
496
497             Xloc_k_filt(pt,:) = Xloc_k_temp_filt(size(Xloc_k,2)+1:2*size(Xloc_k,2));
498             Yloc_k_filt(pt,:) = Yloc_k_temp_filt(size(Xloc_k,2)+1:2*size(Xloc_k,2));
499         end
500     else
501         Xloc_k_filt = Xloc_k;
502         Yloc_k_filt = Yloc_k;
503     end
504     fprintf('DONE\n')
505
506     %% Calculate displacements
507     fprintf('Calculating displacements...')
508
509     Xdisp_k = Xloc_k - Xvector; % pixels
510     Xdisp_k_filt = Xloc_k_filt - Xvector;
511     Ydisp_k = Yloc_k - Yvector;
512     Ydisp_k_filt = Yloc_k_filt - Yvector;
513
514     Xgrid = px0;
515     Ygrid = py0;
516
517     % XYdisp = [Xdisp_k_filt(real_points(:),1), Ydisp_k_filt(real_points(:),1)]';
518     % for f = 1:size(XYdisp,2)
519     %     totaldisp(f) = norm(XYdisp(:,f))*meta_BD.Calibration;
520     % end
```

```
521
522     fprintf('DONE\n')
523
524     %% find cell points
525
526     % CB = CBraw;
527     % nCells = length(CBraw);
528     % fprintf('\t%i cells detected\n',nCells)
529     % edge_tooclose = 2*mean(BD.DotSpacings)/meta_BD.Calibration;
530     dot_consider = 0.75*mean(celldata(ic).BD.DotSpacings)/meta_BD.Calibration;
531
532     tb = celldata(ic).CB; % temporary cell boundary data
533     %     fprintf('\tCell %i...',nc2)
534
535     %     %check that cell is not too close to outside
536     %     if any([(min(tb(:,1)) - 1), (min(tb(:,2)) - 1), (meta_BD.N - max(tb(:,1))), (
            meta_BD.M - max(tb(:,2)))] < edge_tooclose)
537     %         fprintf('TOO CLOSE TO EDGE\n')
538     %         CB(ic+1) = [];
539     %         continue
540     %     else
541     %          fprintf('OK\n')
542     %          ic = ic + 1;
543     %     end
544
545     dot_dist_from_CB = zeros(size(px));
546     for iy = 1:size(px,1)
547         for jx = 1:size(px,2)
548             dot_dist_from_CB(iy,jx) = min(sqrt((px(iy,jx) - tb(:,1)).^2 + (py(iy,jx) - tb
                    (:,2)).^2));
549         end
550     end
551
552     celldots = inpolygon(px,py,tb(:,1),tb(:,2)) | (dot_dist_from_CB < dot_consider);
553
554     %% threshold displacements
555     % (if displacement < threshold, set = 0)
556     fprintf('Removing points below noise...')
557
558     Xdisp_all = Xdisp_k_filt;
559     Ydisp_all = Ydisp_k_filt;
560     % Xdisp_out = Xdisp_k_filt(real_points(:) & ~celldots(:));
```

```
561     % Ydisp_out = Ydisp_k_filt(real_points(:) & ~celldots(:));
562
563     disp_all = sqrt(Xdisp_all.^2 + Ydisp_all.^2)*meta_BD.Calibration;
564     % disp_out = sqrt(Xdisp_out.^2 + Ydisp_out.^2)*meta_BD.Calibration;
565
566     % dist_out_boot = bootstrp(2000,@median,disp_out);
567     % noise_limit = quantile(disp_out,0.95); % this is the level that includes the lower 95%
            of displacements outside the cell;
568     %
569     % noise_limit = 0.16867; % [um] for 2E substrate 0% stiffness, this is peak difference
570     % noise_limit = 0.22054; % [um] for 2E substrate 0% stiffness, this is 95% quantile
571     % noise_limit = 0.21856; % [um] for 2E substrate 5% stiffness, this is 95% quantile
572     % noise_limit = 0.22054; % [um] for 2E substrate 0% stiffness, this is 95% quantile
573     noise_limit = 0;
574     noise_points = reshape(all(disp_all < noise_limit,2),size(real_points));
575
576     Xdisp_k_dn = Xdisp_k_filt;
577     Ydisp_k_dn = Ydisp_k_filt;
578     Xdisp_k_dn(noise_points(:),:) = 0;
579     Ydisp_k_dn(noise_points(:),:) = 0;
580
581     fprintf('DONE (%i points of %i removed)\n',nnz(noise_points),nnz(celldata(ic).
            real_points))
582
583     %% calculate traction forces
584     fprintf('Calculating tractions...')
585
586     [n_row,n_col] = size(Xgrid);
587
588     E = meta_BD.YoungsModulus;
589     nu = meta_BD.Poisson;
590     d = mean(celldata(ic).BD.DotSpacings);
591
592     Xtrac_k = zeros(size(Xdisp_k_filt));
593     Ytrac_k = zeros(size(Xdisp_k_filt));
594     Xforce_k = zeros(size(Xdisp_k_filt));
595     Yforce_k = zeros(size(Xdisp_k_filt));
596
597     ind_k = [(meta_BD.uFrame:-1:1) (meta_BD.uFrame+1:meta_BD.nFrames)];
598
599     % FTTC with Regularization
600     % for k = 1:meta_BD.nFrames
```

```
601     for kk = 1:length(ind_k)
602         k = ind_k(kk);
603         u_x = reshape(Xdisp_k_dn(:,k),n_row,n_col)*meta_BD.Calibration;
604         u_y = reshape(Ydisp_k_dn(:,k),n_row,n_col)*meta_BD.Calibration;
605         u = cat(3,u_x,u_y);
606         if kk == 1
607             if meta_BD.useLcurve
608                 L = logspace(-3,-8,100);
609     %             [~,resid_norm(k,:),soln_norm(k,:)] = calcforce_regFTTC(u,E,nu,d,L,
        real_points);
610     %             [reg_optimal(k),i_reg_optimal(k)] = find_L(resid_norm(k,:),soln_norm(k,:),
        L,meta_BD.regParam,'optimal');
611     %             [reg_corner(k),i_reg_corner(k)] = find_L(resid_norm(k,:),soln_norm(k,:),L,
        meta_BD.regParam,'corner');
612                 [~,resid_norm,soln_norm] = calcforce_regFTTC(u,E,nu,d,L,celldata(ic));
613                 [reg_optimal,i_reg_optimal] = find_L(resid_norm,soln_norm,L,meta_BD.regParam
                        ,'optimal');
614                 [reg_corner,i_reg_corner] = find_L(resid_norm,soln_norm,L,meta_BD.regParam,'
                        corner');
615     %             L = reg_optimal(k);
616                 L = meta_BD.regParam;
617
618     %             loglog(resid_norm,soln_norm,'-k',resid_norm(i_reg_optimal),soln_norm(
        i_reg_optimal),'or',resid_norm(i_reg_corner),soln_norm(i_reg_corner),'ob')
619             else
620                 L = meta_BD.regParam;
621                 reg_optimal = NaN;
622                 reg_corner = NaN;
623             end
624         end
625
626         trac = calcforce_regFTTC(u,E,nu,d,L,celldata(ic));
627         % trac has same units as E, young's modulus
628         % [N*m^-2] is equivalent to [pN*um^-2]
629
630         Xtrac_k(:,k) = reshape(trac(:,:,1),[],1); % [pN*um^-2] = [uN*mm^-2] = [N*m^-2]
631         Ytrac_k(:,k) = reshape(trac(:,:,2),[],1); % [pN*um^-2]
632         Xforce_k(:,k) = Xtrac_k(:,k)*d^2; % pN
633         Yforce_k(:,k) = Ytrac_k(:,k)*d^2; % pN
634     end
635
636     fprintf('DONE\n')
```

```
637
638     %% calculate data for each cell
639     fprintf('Calculating data for each cell...')
640
641     celldata(ic).px0 = px0;
642     celldata(ic).py0 = py0;
643     celldata(ic).px = px;
644     celldata(ic).py = py;
645     celldata(ic).px_k = px_k;
646     celldata(ic).py_k = py_k;
647
648     celldata(ic).Xloc_k = Xloc_k;
649     celldata(ic).Yloc_k = Yloc_k;
650     celldata(ic).Xloc_k_filt = Xloc_k_filt;
651     celldata(ic).Yloc_k_filt = Yloc_k_filt;
652     celldata(ic).Xvector = Xvector;
653     celldata(ic).Yvector = Yvector;
654     celldata(ic).Xgrid = Xgrid;
655     celldata(ic).Ygrid = Ygrid;
656
657     celldata(ic).Xdisp_k = Xdisp_k;
658     celldata(ic).Ydisp_k = Ydisp_k;
659     celldata(ic).Xdisp_k_filt = Xdisp_k_filt;
660     celldata(ic).Ydisp_k_filt = Ydisp_k_filt;
661
662     celldata(ic).Xdisp_k_dn = Xdisp_k_dn;
663     celldata(ic).Ydisp_k_dn = Ydisp_k_dn;
664
665     celldata(ic).noise_limit = noise_limit;
666     celldata(ic).noise_points = noise_points;
667     celldata(ic).real_points = real_points;
668
669     celldata(ic).Xtrac_k = Xtrac_k;% pN*um^-2
670     celldata(ic).Ytrac_k = Ytrac_k;% pN*um^-2
671     celldata(ic).Xforce_k = Xforce_k; % pN
672     celldata(ic).Yforce_k = Yforce_k; % pN
673
674     celldata(ic).reg_optimal = reg_optimal;
675     celldata(ic).reg_corner = reg_corner;
676     if meta_BD.useLcurve
677         celldata(ic).resid_norm = resid_norm;
678         celldata(ic).soln_norm = soln_norm;
```

```
679     end
680
681     celldata(ic).celldots = celldots;
682
683     celldata(ic).total_trac = sum(sqrt(Xtrac_k(celldots(:),:).^2 + Ytrac_k(celldots(:),:)
            .^2)); % pN*um^-2
684
685     celldata(ic).total_force = sum(sqrt(Xforce_k(celldots(:),:).^2 + Yforce_k(celldots(:),:)
            .^2)); % pN
686     celldata(ic).net_force = sqrt(sum(Xforce_k(celldots(:),:)).^2 + sum(Yforce_k(celldots(:)
            ,:)).^2); % pN
687
688     celldata(ic).total_disp = sum(sqrt(Xdisp_k_dn(celldots(:),:).^2 + Ydisp_k_dn(celldots(:)
            ,:).^2)); % um
689     celldata(ic).net_disp = sqrt(sum(Xdisp_k_dn(celldots(:),:)).^2 + sum(Ydisp_k_dn(celldots
            (:),:)).^2); % um
690
691     celldata(ic).nDots = nnz(celldots);
692     celldata(ic).area = polyarea(tb(:,1),tb(:,2))*meta_BD.Calibration.^2; % [um^2]
693
694     celldata(ic).img_ref = img_REFBD_crop;
695     celldata(ic).img_ref_filt = img_REFBD_filt_crop;
696
697     fprintf('DONE\n')
698
699 end % do next cell
700
701 %% save the data
702 fprintf('Saving...')
703
704 save([path_save file_save],...
705     'file_BD','path_BD','uM','meta_BD',...
706     'img_REFBD*',...
707     'celldata',...
708     '-append')
709
710 fprintf('DONE\n')
711
712 cd(path_save)
713
714 %% Plotting results
715 arrowscale = 0.0005; % 0.002 might need to play around with this one
```

```
716 scalebar_length = 20; % microns
717 scalebar_force = 50; % nN
718 cell_to_plot = 1; % only applies to video
719
720 if metaBD.analyzeVideo
721     plot_forces_video
722 else
723     plot_forces_image
724 end
```

A.3.2 Finding dot centroids

```
1  function [px,py,real_points,img_bw,removed_rows,removed_cols] = find_centroids_new(px,py,img
       ,celldata,meta)
2  % px,py are a grid of points where we think a dot should exist
3  % this code takes those initial guesses and finds the actual centroid of
4  % each dot
5  % real_points tells us if those points are actual dots or imaginary ones
6
7  dotspacing_px = meta.DotSpacing/meta.Calibration;
8  dotsize_px = 2*round((meta.DotSize/meta.Calibration + 1)/2) - 1;
9
10 [num_Y,num_X] = size(px);
11
12 if ~isfield(celldata,'real_points') || isempty(celldata.real_points)
13     real_points = true(num_Y,num_X);
14 else
15     real_points = celldata.real_points;
16 end
17
18 x_ind = [1:floor(num_X/2), num_X:-1:ceil((num_X)/2+0.25)];
19 y_ind = [1:floor(num_Y/2), num_Y:-1:ceil((num_Y)/2+0.25)];
20
21 pxorig = px;
22 pyorig = py;
23
24 img_filt = imgaussfilt(img,1);
25 img_filt_bw = imfill(imcomplement(imbinarize(img_filt)),'holes');
26 D = bwdist(~img_filt_bw);
27 D2 = -bwdist(img_filt_bw);
28 hills = mat2gray(D+D2);
```

```
29
30 % loop over all the points
31 for j = 1:num_X
32     jx = x_ind(j);
33     for i = 1:num_Y
34         iy = y_ind(i);
35         if real_points(iy,jx)
36
37             % figure out how the previous dot "left of" this one was moved
38             disp_y = [];
39             if (y_ind(i) ~= 1) && (y_ind(i) ~= num_Y)
40                 disp_y = [(px(y_ind(i-1),jx) - pxorig(y_ind(i-1),jx)), (py(y_ind(i-1),jx) -
                       pyorig(y_ind(i-1),jx))];
41             end
42
43             % figure out how the previous dot "above" this one was moved
44             disp_x = [];
45             if (x_ind(j) ~= 1) && (x_ind(j) ~= num_X)
46                 disp_x = [(px(iy,x_ind(j-1)) - pxorig(iy,x_ind(j-1))), (py(iy,x_ind(j-1)) -
                       pyorig(iy,x_ind(j-1)))];
47             end
48
49             if isempty(disp_x) && isempty(disp_y)
50                 disp = [];
51             elseif isempty(disp_x)
52                 disp = disp_y;
53             elseif isempty(disp_y)
54                 disp = disp_x;
55             else
56                 disp = mean([disp_y; disp_x],1);
57             end
58
59             % move the current dot according to how the previous left/above ones moved
60             if ~isempty(disp)
61                 px(iy,jx) = px(iy,jx) + disp(1);
62                 py(iy,jx) = py(iy,jx) + disp(2);
63             end
64
65             % do the object tracking
66             pxr = round(px(iy,jx));
67             pyr = round(py(iy,jx));
```

```
68              if pxr < 0.5*dotspacing_px || pxr > (celldata.N - 0.5*dotspacing_px) || pyr <
                    0.5*dotspacing_px || pyr > (celldata.M - 0.5*dotspacing_px)
69                  real_points(iy,jx) = false;
70                  continue
71              end
72              % neighborhood is a 3x3 area of pixels
73              neighborhood = hills(pyr-1:pyr+1,pxr-1:pxr+1);
74              while neighborhood(2,2) ~= max(neighborhood(:))
75                  if pxr < 0.5*dotspacing_px || pxr > (celldata.N - 0.5*dotspacing_px) || pyr
                        < 0.5*dotspacing_px || pyr > (celldata.M - 0.5*dotspacing_px)
76                      real_points(iy,jx) = false;
77                      break
78                  else
79      %                     wx = [1 0 -1;2 0 -2;1 0 -1]; % sobel gradient weights
80      %                     wy = [1 2 1;0 0 0;-1 -2 -1];
81      %                     Gx = sum(neighborhood(:).*wx(:));
82      %                     Gy = sum(neighborhood(:).*wy(:));
83      %                     Gmag = sqrt(Gx^2+Gy^2);
84      %                     Gdir = -atan2(Gy,-Gx)*180/pi;
85      % %                     [Gmag2,Gdir2] = imgradient(neighborhood);
86      % %                     if Gdir2(2,2) ~= Gdir
87      % %                         1;
88      % %                     end
89      %
90      %                     pxr = pxr + cosd(Gdir)/abs(cosd(Gdir))*(abs(cosd(Gdir)) > sqrt(2)/2);
91      %                     pyr = pyr + sind(Gdir)/abs(sind(Gdir))*(abs(sind(Gdir)) > sqrt(2)/2);
92
93                      [~,ind_max] = max(neighborhood,[],'all','linear');
94                      [y_max,x_max] = ind2sub(size(neighborhood),ind_max);
95                      pxr = pxr - 2 + x_max;
96                      pyr = pyr - 2 + y_max;
97                      neighborhood = hills(pyr-1:pyr+1,pxr-1:pxr+1);
98                  end
99              end
100             px(iy,jx) = pxr;
101             py(iy,jx) = pyr;
102         end
103     end
104 end
105 % fprintf('Done')
106
107 % remove rows/columns if there aren't at least 2*uPoints points
```

```
check2 = true;
removed_rows = [];
removed_cols = [];
while check2
    check2 = false;
    Lremove = [];
    for jx = 1:num_X
        if nnz(real_points(:,jx)) < 2
            Lremove = [Lremove; jx];
            check2 = true;
            continue
        end

        if isinf(meta.uPoints)
            % do nothing
        elseif nnz(real_points(:,jx)) < 2*meta.uPoints
            Lremove = [Lremove; jx];
            check2 = true;
        end
    end
    px(:,Lremove) = [];
    py(:,Lremove) = [];
    real_points(:,Lremove) = [];
    removed_cols = [removed_cols; Lremove];
    [num_Y,num_X] = size(px);

    Lremove = [];
    for iy = 1:num_Y
        if nnz(real_points(iy,:)) < 2
            Lremove = [Lremove; iy];
            check2 = true;
            continue
        end

        if isinf(meta.uPoints)
            % do nothing
        elseif nnz(real_points(iy,:)) < 2*meta.uPoints
            Lremove = [Lremove; iy];
            check2 = true;
        end
    end
    px(Lremove,:) = [];
```

```
150     py(Lremove,:) = [];
151     real_points(Lremove,:) = [];
152     removed_rows = [removed_rows; Lremove];
153     [num_Y,num_X] = size(px);
154 end
155 
156 % now update the points to the actual centroid (instead of the rounded
157 % pixel value)
158 img_bw = bwselect(img_filt_bw,px(real_points),py(real_points));
159 CC = bwconncomp(img_bw);
160 
161 % get centroid of each object
162 CC.Centroids = [];
163 for i = 1:CC.NumObjects
164     [y,x] = ind2sub(size(img_bw),CC.PixelIdxList{i});
165     CC.Centroids(i,:) = [mean(x),mean(y)];
166 end
167 
168 % find closest centroid to each px,py, set px,py = centroid
169 [x_ind_real,y_ind_real] = find(real_points);
170 ind_real = sub2ind(size(real_points),x_ind_real,y_ind_real);
171 for np = ind_real'
172     dist = sqrt(sum((CC.Centroids - [px(np),py(np)]).^2,2));
173     [~,idx_min] = min(dist);
174     px(np) = CC.Centroids(idx_min,1);
175     py(np) = CC.Centroids(idx_min,2);
176 end
```

A.3.3 Estimate reference positions of dots

```
1  function [px,py,px0,py0] = find_undeformed(px,py,celldata,meta)
2  real_points = celldata.real_points;
3  
4  [num_Y,num_X] = size(px);
5  
6  %% find slope of each line of dots
7  xlines = zeros(num_Y,2);
8  for iy = 1:num_Y
9      num_points = sum(real_points(iy,:));
10     if num_points >= 2
11         if isempty(meta.uPoints)
```

```
12              jx = find(real_points(iy,:));
13          elseif num_points >= 2*meta.uPoints
14              jx = find(real_points(iy,:));
15              jx = jx([1:meta.uPoints, (end-meta.uPoints+1):end]);
16          else
17              jx = find(real_points(iy,:));
18          end
19          pca_coeff = pca([px(iy,jx)', py(iy,jx)']);
20          slope = pca_coeff(2,1)/pca_coeff(1,1);
21          intercept = -slope*mean(px(iy,jx)) + mean(py(iy,jx));
22          xlines(iy,:) = [slope, intercept];
23      end
24  end
25
26  ylines = zeros(num_X,2);
27  for jx = 1:num_X
28      num_points = sum(real_points(:,jx));
29      if num_points >= 2
30          if isempty(meta.uPoints)
31              iy = find(real_points(:,jx))';
32          elseif num_points >= 2*meta.uPoints
33              iy = find(real_points(:,jx))';
34              iy = iy([1:meta.uPoints, (end-meta.uPoints+1):end]);
35          else
36              iy = find(real_points(:,jx))';
37          end
38          pca_coeff = pca([px(iy,jx), py(iy,jx)]);
39          slope = pca_coeff(2,1)/pca_coeff(1,1);
40          intercept = -slope*mean(px(iy,jx)) + mean(py(iy,jx));
41          ylines(jx,:) = [slope, intercept];
42      end
43  end
44
45  %% correct them with median slope
46  for iy = 1:num_Y
47      num_points = sum(real_points(iy,:));
48      if num_points >= 2
49          if isempty(meta.uPoints)
50              jx = find(real_points(iy,:));
51          elseif num_points >= 2*meta.uPoints
52              jx = find(real_points(iy,:));
53              jx = jx([1:meta.uPoints, (end-meta.uPoints+1):end]);
```

```
54          else
55              jx = find(real_points(iy,:));
56          end
57  %         pca_coeff = pca([px(iy,jx)', py(iy,jx)']);
58          slope = median(xlines(:,1));
59          intercept = -slope*mean(px(iy,jx)) + mean(py(iy,jx));
60          xlines(iy,:) = [slope, intercept];
61      end
62  end
63
64  for jx = 1:num_X
65      num_points = sum(real_points(:,jx));
66      if num_points >= 2
67          if isempty(meta.uPoints)
68              iy = find(real_points(:,jx))';
69          elseif num_points >= 2*meta.uPoints
70              iy = find(real_points(:,jx))';
71              iy = iy([1:meta.uPoints, (end-meta.uPoints+1):end]);
72          else
73              iy = find(real_points(:,jx))';
74          end
75  %         pca_coeff = pca([px(iy,jx), py(iy,jx)]);
76          slope = median(ylines(:,1));
77          intercept = -slope*mean(px(iy,jx)) + mean(py(iy,jx));
78          ylines(jx,:) = [slope, intercept];
79      end
80  end
81
82  %% find undeformed by intersection of 2 lines
83
84  px0 = zeros(size(px));
85  py0 = zeros(size(py));
86  for jx = 1:num_X
87      for iy = 1:num_Y
88          px0(iy,jx) = (xlines(iy,2) - ylines(jx,2))/...
89              (ylines(jx,1) - xlines(iy,1));
90          py0(iy,jx) = xlines(iy,1)*px0(iy,jx) + xlines(iy,2);
91          if ~real_points(iy,jx)
92              px(iy,jx) = px0(iy,jx);
93              py(iy,jx) = py0(iy,jx);
94          end
95      end
```

```
end
```

A.3.4 Transient analysis

```
%% analyze transient waveform and output results to command window

% controls for how a beat "start" and base is defined
start_pct = 0.03;
base_pct = 0.1;

ic = 1; % which cell to analyze

% code is borrowed from calcium analysis, so that is why everything is
% named 'ca_'
ca_rel = celldata(ic).total_force'*10^-6;

caI_peaks_init = peakfinder(ca_rel); % see end of script for peakfinder
if caI_peaks_init(end) == meta_BD.nFrames
    caI_peaks_init = caI_peaks_init(1:end-1);
end

if length(caI_peaks_init) == 1
    [~,caI_troughs(1)] = min(ca_rel(1:caI_peaks_init(1)));
    [~,caI_troughs(2)] = min(ca_rel(caI_peaks_init(1):end));

    caI_troughs(2) = caI_troughs(2) + caI_peaks_init(1) - 1;

    caI_peaks = caI_peaks_init;
elseif length(caI_peaks_init) == 2
    [~,caI_troughs(1)] = min(ca_rel(1:caI_peaks_init(1)));
    [~,caI_troughs(2)] = min(ca_rel(caI_peaks_init(1):caI_peaks_init(2)));
    [~,caI_troughs(3)] = min(ca_rel(caI_peaks_init(2):end));

    caI_troughs(2) = caI_troughs(2) + caI_peaks_init(1) - 1;
    caI_troughs(3) = caI_troughs(3) + caI_peaks_init(2) - 1;

    caI_peaks = caI_peaks_init;
else
    caI_troughs = zeros(length(caI_peaks_init),1);
    for np = 1:length(caI_peaks_init)-1
        [~,caI_troughs(np,1)] = min(ca_rel(caI_peaks_init(np):caI_peaks_init(np + 1)));
```

```
38          caI_troughs(np,1) = caI_troughs(np) + caI_peaks_init(np) - 1;
39      end
40      np = np + 1;
41      [~,caI_troughs(np,1)] = min(ca_rel(caI_peaks_init(np):end));
42      caI_troughs(np,1) = caI_troughs(np) + caI_peaks_init(np) - 1;
43
44      if (caI_peaks_init(1) >= 0.9*mean(caI_peaks_init(2:end-1) - caI_troughs(1:end-2)))
45          [~,first_trough] = min(ca_rel(1:caI_peaks_init(1)));
46
47          if first_trough == 1 %  get rid of first beat if trough is first point
48              caI_peaks = caI_peaks_init(2:end);
49          else
50              caI_peaks = caI_peaks_init(1:end);
51              caI_troughs = [first_trough; caI_troughs];
52          end
53      else % remove first beat if it is too close to starting time
54          caI_peaks = caI_peaks_init(2:end);
55      end
56
57      if ((meta_BD.nFrames - caI_peaks(end)) < 0.9*mean(caI_troughs(2:end-1) - caI_peaks(1:end
            -1)))
58          % remove last beat if it is too close to ending time
59          caI_peaks = caI_peaks(1:end-1);
60          caI_troughs = caI_troughs(1:end-1);
61      end
62  end
63
64  for np = 1:length(caI_peaks)
65      i_L = caI_troughs(np);
66      i_R = caI_troughs(np + 1);
67      temp_ca_rel = ca_rel(i_L:caI_peaks(np));
68      caV_baselines_pre(np,1) = median(temp_ca_rel(temp_ca_rel < (base_pct*(max(temp_ca_rel) -
             min(temp_ca_rel)) + min(temp_ca_rel))));
69      temp_ca_rel = ca_rel(caI_peaks(np):i_R);
70      caV_baselines_post(np,1) = median(temp_ca_rel(temp_ca_rel < (base_pct*(max(temp_ca_rel)
            - min(temp_ca_rel)) + min(temp_ca_rel))));
71
72      caI_start(np,1) = find(ca_rel(i_L:caI_peaks(np)) < (start_pct*(ca_rel(caI_peaks(np)) -
            caV_baselines_pre(np)) + caV_baselines_pre(np)),1,'Last');
73      caI_start(np,1) = caI_start(np) + i_L - 1;
74
75      if (caI_start(np) - i_L) < 0.2*mean(diff(caI_peaks))
```

```
76          caV_baselines_pre(np,1) = ca_rel(i_L);
77          caI_start(np,1) = find(ca_rel(i_L:caI_peaks(np)) < (start_pct*(ca_rel(caI_peaks(np))
                - caV_baselines_pre(np)) + caV_baselines_pre(np)),1,'Last');
78          caI_start(np,1) = caI_start(np) + i_L - 1;
79          if (np > 1)
80              caV_baselines_post(np-1,1) = ca_rel(i_L);
81          end
82      end
83
84      if np == length(caI_peaks)
85          if all(caV_baselines_post(1:length(caI_peaks)-1) == ca_rel(caI_troughs(2:length(
                caI_peaks-1)))) % if all previous beat starts were on a trough
86              caV_baselines_post(np,1) = ca_rel(i_R);
87          end
88      end
89  % %     while ca_rel(caI_start(np) - 1) < ca_rel(caI_start(np))
90  % %         caI_start(np) = caI_start(np) - 1;
91  % %     end
92
93  % %     caI_start(np,1) = find((ca_rel(i_L:caI_peaks(np)) - ca_rel(i_L)) < start_pct*(ca_rel
        (caI_peaks(np)) - ca_rel(i_L)),1,'last');
94  %     caI_start(np,1) = find((ca_rel(i_L:caI_peaks(np)) - linspace(ca_rel(i_L),(ca_rel(i_R)
        - ca_rel(i_L))*(caI_peaks(np) - i_L)/(i_R - i_L) + ca_rel(i_L),(caI_peaks(np) - i_L + 1)
        )') < start_pct*(ca_rel(caI_peaks(np)) - ca_rel(i_L)),1,'last');
95
96      caI_50R(np,1) = find((ca_rel(caI_peaks(np):i_R) - ca_rel(i_R)) < 0.5*(ca_rel(caI_peaks(
            np)) - ca_rel(i_R)),1,'first');
97      caI_50R(np,1) = caI_50R(np) + caI_peaks(np) - 1;
98      caT_50R_i(np,1) = interp1(ca_rel([caI_50R(np)-1, caI_50R(np)]) - ca_rel(i_R),meta_BD.
            Time([caI_50R(np)-1, caI_50R(np)]),0.5*(ca_rel(caI_peaks(np)) - ca_rel(i_R)));
99
100     caI_90R(np,1) = find((ca_rel(caI_peaks(np):i_R) - ca_rel(i_R)) < 0.1*(ca_rel(caI_peaks(
            np)) - ca_rel(i_R)),1,'first');
101     caI_90R(np,1) = caI_90R(np) + caI_peaks(np) - 1;
102     caT_90R_i(np,1) = interp1(ca_rel([caI_90R(np)-1, caI_90R(np)]) - ca_rel(i_R),meta_BD.
            Time([caI_90R(np)-1, caI_90R(np)]),0.1*(ca_rel(caI_peaks(np)) - ca_rel(i_R)));
103
104     caI_10R(np,1) = find((ca_rel(caI_peaks(np):i_R) - ca_rel(i_R)) < 0.9*(ca_rel(caI_peaks(
            np)) - ca_rel(i_R)),1,'first');
105     caI_10R(np,1) = caI_10R(np) + caI_peaks(np) - 1;
106     caT_10R_i(np,1) = interp1(ca_rel([caI_10R(np)-1, caI_10R(np)]) - ca_rel(i_R),meta_BD.
            Time([caI_10R(np)-1, caI_10R(np)]),0.9*(ca_rel(caI_peaks(np)) - ca_rel(i_R)));
```

```
107
108 %     decay_c = lsqcurvefit(@(a,x) a(1)+a(2)*exp(-a(3)*x),[100 100 10],meta_BD.Time(caI_50R(
        np):caI_90R(np))-meta_BD.Time(caI_50R(np)),ca_rel(caI_50R(np):caI_90R(np)),[],[],...
109 %         optimoptions('lsqcurvefit','Display','off','MaxFunctionEvaluations',1000,'
        OptimalityTolerance',1e-6));
110 %     decay_fits(np,:) = decay_c;
111 %     tau(np,1) = 1/decay_c(3);
112
113     % accurate peak fitting
114     caI_80A(np,1) = find((ca_rel(i_L:caI_peaks(np)) - ca_rel(i_L)) < 0.8*(ca_rel(caI_peaks(
            np)) - ca_rel(i_L)),1,'last');
115     caI_80A(np,1) = caI_80A(np) + i_L - 1;
116     caI_20R(np,1) = find((ca_rel(caI_peaks(np):i_R) - ca_rel(i_R)) < 0.8*(ca_rel(caI_peaks(
            np)) - ca_rel(i_R)),1,'first');
117     caI_20R(np,1) = caI_20R(np) + caI_peaks(np) - 1;
118
119     peak_c = polyfit(meta_BD.Time(caI_80A(np):caI_20R(np)),ca_rel(caI_80A(np):caI_20R(np))
            ,3);
120     peak_fits(np,:) = peak_c;
121     roots_peaks = roots(polyder(peak_c));
122     caT_peaks_i(np,1) = roots_peaks(polyval(polyder(polyder(peak_c)),roots_peaks) < 0);
123     caV_peaks_i(np,1) = polyval(peak_c,caT_peaks_i(np));
124
125     % max up and downstroke
126     [~,caI_RMaxUp(np,1)] = max(diff(ca_rel(caI_start(np):caI_peaks(np))));
127     caI_RMaxUp(np,1) = caI_RMaxUp(np) + caI_start(np) - 1;
128     RMaxUp(np,1) = (ca_rel(caI_RMaxUp(np) + 1) - ca_rel(caI_RMaxUp(np)))/(meta_BD.Time(
            caI_RMaxUp(np) + 1) - meta_BD.Time(caI_RMaxUp(np)));
129
130     [~,caI_RMaxDn(np,1)] = min(diff(ca_rel(caI_peaks(np):caI_90R(np))));
131     caI_RMaxDn(np,1) = caI_RMaxDn(np) + caI_peaks(np) - 1;
132     RMaxDn(np,1) = (ca_rel(caI_RMaxDn(np) + 1) - ca_rel(caI_RMaxDn(np)))/(meta_BD.Time(
            caI_RMaxDn(np) + 1) - meta_BD.Time(caI_RMaxDn(np)));
133 end
134
135 caT_start = meta_BD.Time(caI_start);
136 caT_peaks = meta_BD.Time(caI_peaks);
137
138 caV_start = ca_rel(caI_start);
139 caV_peaks = ca_rel(caI_peaks);
140 caV_50R_i = (0.5*(ca_rel(caI_peaks) - ca_rel(caI_troughs(2:end))) + ca_rel(caI_troughs(2:end
        )));
```

```
141 caV_90R_i = (0.1*(ca_rel(caI_peaks) - ca_rel(caI_troughs(2:end))) + ca_rel(caI_troughs(2:end
        )));
142 caV_10R_i = (0.9*(ca_rel(caI_peaks) - ca_rel(caI_troughs(2:end))) + ca_rel(caI_troughs(2:end
        )));
143
144 caR_peaks = (caV_peaks - caV_start)./(caT_peaks - caT_start);
145 caR_50R_i = (caV_peaks - caV_50R_i)./(caT_50R_i - caT_peaks);
146 caR_90R_i = (caV_peaks - caV_90R_i)./(caT_90R_i - caT_peaks);
147 caR_10R_i = (caV_peaks - caV_10R_i)./(caT_10R_i - caT_peaks);
148
149 Tpeak_mean = mean(caT_peaks - caT_start);
150 T50R_mean = mean(caT_50R_i - caT_peaks);
151 T90R_mean = mean(caT_90R_i - caT_peaks);
152 T10R_mean = mean(caT_10R_i - caT_peaks);
153
154 Rpeak_mean = mean(caR_peaks);
155 R50R_mean = mean(caR_50R_i);
156 R90R_mean = mean(caR_90R_i);
157 R10R_mean = mean(caR_10R_i);
158
159 freq_mean = (length(caT_peaks) - 1)./(caT_peaks(end)-caT_peaks(1));
160
161 % tau_mean = mean(tau);
162
163 RMaxUp_mean = mean(RMaxUp);
164 RMaxDn_mean = mean(RMaxDn);
165
166 fprintf('Success: Waveform analysis completed.\n')
167 fprintf('\tAverage beat frequency = %0.4f Hz.\n',freq_mean)
168
169 % figure
170 % hold on
171 % plot(meta_BD.Time,ca_rel,'-k','linewidth',1);
172 % plot(caT_start,caV_start,'>','markeredgecolor','none','markerfacecolor',[0 0.75 0])
173 % plot(caT_peaks,caV_peaks,'^','markeredgecolor','none','markerfacecolor','r')
174 % plot(caT_50R_i,caV_50R_i,'v','markeredgecolor','none','markerfacecolor',[0 0 1])
175 % plot(caT_90R_i,caV_90R_i,'v','markeredgecolor','none','markerfacecolor',[0.6 0 1])
176 % hold off
177
178 figure('Position',[700,100,600,200])
179 hold on
180 plot(meta_BD.Time,ca_rel,'-k','linewidth',2);
```

```
plot(caT_start,caV_start,'>','markeredgecolor','none','markerfacecolor',[0 0.75 0])
plot(caT_peaks,caV_peaks,'^','markeredgecolor','none','markerfacecolor','r')
plot(caT_50R_i,caV_50R_i,'v','markeredgecolor','none','markerfacecolor',[0 0 1])
plot(caT_90R_i,caV_90R_i,'v','markeredgecolor','none','markerfacecolor',[0.6 0 1])
% plot(meta_BD.Time,10^-6*celldata(ic).total_force,'-k','linewidth',2)
hold off
xlabel('Time [s]')
ylabel('Total Force [uN]')
box off
set(gca,'linewidth',1.5,'tickdir','out','XColor','k','YColor','k')

print(sprintf('plot_force_transient_labeled_cell%i', ic),'-dpng')
print(sprintf('plot_force_transient_labeled_cell%i', ic),'-dsvg','-vector')

fprintf(['Results\n',...
    '%s\n',...
    '%s\n',...
    '%12s\t%12s\t',...
    '%12s\t%12s\t%12s\t',...
    '%12s\t%12s\t%12s\t',...
    '%12s\t%12s\t%12s\n',...
    '%12.2f\t%12.2f\t',...
    '%12.2f\t%12.2f\t%12.2f\t',...
    '%12.3f\t%12.3f\t%12.3f\t',...
    '%12.3f\t%12.3f\t%12.3f\n'],...
    'File',...
    [meta_BD.PathName filesep meta_BD.FileName], ...
    'Area(um2)','Freq(Hz)',...
    'F0(uN)','Fmax(uN)','Fmax/F0',...
    'Tpeak(s)','T50R(s)','T90R(s)',...
    'Rpeak(uN/s)','R50R(uN/s)','R90R(uN/s)',...
    celldata(ic).area, freq_mean,...
    mean(caV_baselines_pre), mean(caV_peaks), mean(caV_peaks./caV_baselines_pre),...
    Tpeak_mean, T50R_mean, T90R_mean,...
    Rpeak_mean, R50R_mean, R90R_mean)

%% code from the internet
function varargout = peakfinder(x0, sel, thresh, extrema)
    %PEAKFINDER Noise tolerant fast peak finding algorithm
    %   INPUTS:
    %       x0 - A real vector from the maxima will be found (required)
    %       sel - The amount above surrounding data for a peak to be
```

```
223     %           identified (default = (max(x0)-min(x0))/4). Larger values mean
224     %           the algorithm is more selective in finding peaks.
225     %       thresh - A threshold value which peaks must be larger than to be
226     %           maxima or smaller than to be minima.
227     %       extrema - 1 if maxima are desired, -1 if minima are desired
228     %           (default = maxima, 1)
229     %   OUTPUTS:
230     %       peakLoc - The indicies of the identified peaks in x0
231     %       peakMag - The magnitude of the identified peaks
232     %
233     %   [peakLoc] = peakfinder(x0) returns the indicies of local maxima that
234     %       are at least 1/4 the range of the data above surrounding data.
235     %
236     %   [peakLoc] = peakfinder(x0,sel) returns the indicies of local maxima
237     %       that are at least sel above surrounding data.
238     %
239     %   [peakLoc] = peakfinder(x0,sel,thresh) returns the indicies of local
240     %       maxima that are at least sel above surrounding data and larger
241     %       (smaller) than thresh if you are finding maxima (minima).
242     %
243     %   [peakLoc] = peakfinder(x0,sel,thresh,extrema) returns the maxima of the
244     %       data if extrema > 0 and the minima of the data if extrema < 0
245     %
246     %   [peakLoc, peakMag] = peakfinder(x0,...) returns the indicies of the
247     %       local maxima as well as the magnitudes of those maxima
248     %
249     %   If called with no output the identified maxima will be plotted along
250     %       with the input data.
251     %
252     %   Note: If repeated values are found the first is identified as the peak
253     %
254     % Ex:
255     % t = 0:.0001:10;
256     % x = 12*sin(10*2*pi*t)-3*sin(.1*2*pi*t)+randn(1,numel(t));
257     % x(1250:1255) = max(x);
258     % peakfinder(x)
259     %
260     % Copyright Nathanael C. Yoder 2011 (nyoder@gmail.com)
261
262     % Perform error checking and set defaults if not passed in
263     narginchk(1,4);
264     nargoutchk(0,2);
```

```
265
266     s = size(x0);
267     flipData =  s(1) < s(2);
268     len0 = numel(x0);
269     if len0 ~= s(1) && len0 ~= s(2)
270         error('PEAKFINDER:Input','The input data must be a vector')
271     elseif isempty(x0)
272         varargout = {[],[]};
273         return;
274     end
275     if ~isreal(x0)
276         warning('PEAKFINDER:NotReal','Absolute value of data will be used')
277         x0 = abs(x0);
278     end
279
280     if nargin < 2 || isempty(sel)
281         sel = (max(x0)-min(x0))/4;
282     elseif ~isnumeric(sel) || ~isreal(sel)
283         sel = (max(x0)-min(x0))/4;
284         warning('PEAKFINDER:InvalidSel',...
285             'The selectivity must be a real scalar.  A selectivity of %.4g will be used',sel
                )
286     elseif numel(sel) > 1
287         warning('PEAKFINDER:InvalidSel',...
288             'The selectivity must be a scalar.  The first selectivity value in the vector
                will be used.')
289         sel = sel(1);
290     end
291
292     if nargin < 3 || isempty(thresh)
293         thresh = [];
294     elseif ~isnumeric(thresh) || ~isreal(thresh)
295         thresh = [];
296         warning('PEAKFINDER:InvalidThreshold',...
297             'The threshold must be a real scalar. No threshold will be used.')
298     elseif numel(thresh) > 1
299         thresh = thresh(1);
300         warning('PEAKFINDER:InvalidThreshold',...
301             'The threshold must be a scalar.  The first threshold value in the vector will
                be used.')
302     end
303
```

```
304     if nargin < 4 || isempty(extrema)
305         extrema = 1;
306     else
307         extrema = sign(extrema(1)); % Should only be 1 or -1 but make sure
308         if extrema == 0
309             error('PEAKFINDER:ZeroMaxima','Either 1 (for maxima) or -1 (for minima) must be
                    input for extrema');
310         end
311     end
312
313     x0 = extrema*x0(:); % Make it so we are finding maxima regardless
314     thresh = thresh*extrema; % Adjust threshold according to extrema.
315     dx0 = diff(x0); % Find derivative
316     dx0(dx0 == 0) = -eps; % This is so we find the first of repeated values
317     ind = find(dx0(1:end-1).*dx0(2:end) < 0)+1; % Find where the derivative changes sign
318
319     % Include endpoints in potential peaks and valleys
320     x = [x0(1);x0(ind);x0(end)];
321     ind = [1;ind;len0];
322
323     % x only has the peaks, valleys, and endpoints
324     len = numel(x);
325     minMag = min(x);
326
327
328     if len > 2 % Function with peaks and valleys
329
330         % Set initial parameters for loop
331         tempMag = minMag;
332         foundPeak = false;
333         leftMin = minMag;
334
335         % Deal with first point a little differently since tacked it on
336         % Calculate the sign of the derivative since we taked the first point
337         %  on it does not neccessarily alternate like the rest.
338         signDx = sign(diff(x(1:3)));
339         if signDx(1) <= 0 % The first point is larger or equal to the second
340             ii = 0;
341             if signDx(1) == signDx(2) % Want alternating signs
342                 x(2) = [];
343                 ind(2) = [];
344                 len = len-1;
```

```
345             end
346         else % First point is smaller than the second
347             ii = 1;
348             if signDx(1) == signDx(2) % Want alternating signs
349                 x(1) = [];
350                 ind(1) = [];
351                 len = len-1;
352             end
353         end
354 
355         % Preallocate max number of maxima
356         maxPeaks = ceil(len/2);
357         peakLoc = zeros(maxPeaks,1);
358         peakMag = zeros(maxPeaks,1);
359         cInd = 1;
360         % Loop through extrema which should be peaks and then valleys
361         while ii < len
362             ii = ii+1; % This is a peak
363             % Reset peak finding if we had a peak and the next peak is bigger
364             %   than the last or the left min was small enough to reset.
365             if foundPeak
366                 tempMag = minMag;
367                 foundPeak = false;
368             end
369 
370             % Make sure we don't iterate past the length of our vector
371             if ii == len
372                 break; % We assign the last point differently out of the loop
373             end
374 
375             % Found new peak that was lager than temp mag and selectivity larger
376             %   than the minimum to its left.
377             if x(ii) > tempMag && x(ii) > leftMin + sel
378                 tempLoc = ii;
379                 tempMag = x(ii);
380             end
381 
382             ii = ii+1; % Move onto the valley
383             % Come down at least sel from peak
384             if ~foundPeak && tempMag > sel + x(ii)
385                 foundPeak = true; % We have found a peak
386                 leftMin = x(ii);
```

```
387                 peakLoc(cInd) = tempLoc; % Add peak to index
388                 peakMag(cInd) = tempMag;
389                 cInd = cInd+1;
390             elseif x(ii) < leftMin % New left minima
391                 leftMin = x(ii);
392             end
393         end
394
395         % Check end point
396         if x(end) > tempMag && x(end) > leftMin + sel
397             peakLoc(cInd) = len;
398             peakMag(cInd) = x(end);
399             cInd = cInd + 1;
400         elseif ~foundPeak && tempMag > minMag % Check if we still need to add the last point
401             peakLoc(cInd) = tempLoc;
402             peakMag(cInd) = tempMag;
403             cInd = cInd + 1;
404         end
405
406         % Create output
407         peakInds = ind(peakLoc(1:cInd-1));
408         peakMags = peakMag(1:cInd-1);
409     else % This is a monotone function where an endpoint is the only peak
410         [peakMags,xInd] = max(x);
411         if peakMags > minMag + sel
412             peakInds = ind(xInd);
413         else
414             peakMags = [];
415             peakInds = [];
416         end
417     end
418
419     % Apply threshold value.  Since always finding maxima it will always be
420     %   larger than the thresh.
421     if ~isempty(thresh)
422         m = peakMags>thresh;
423         peakInds = peakInds(m);
424         peakMags = peakMags(m);
425     end
426
427
428
```

```
    % Rotate data if needed
    if flipData
        peakMags = peakMags.';
        peakInds = peakInds.';
    end



    % Change sign of data if was finding minima
    if extrema < 0
        peakMags = -peakMags;
        x0 = -x0;
    end
    % Plot if no output desired
    if nargout == 0
        if isempty(peakInds)
            disp('No significant peaks found')
        else
            figure;
            plot(1:len0,x0,'.-',peakInds,peakMags,'ro','linewidth',2);
        end
    else
        varargout = {peakInds,peakMags};
    end
end
```

A.4 Platelet classification code

This is the primary code used for transfer learning and fine tuning of the *InceptionV3* model, as well as the code for classifying new images using the trained model.

A.4.1 Training and testing

```
import os
import cv2
import tensorflow as tf
import numpy as np
# import pandas as pd
from tensorflow.keras.utils import to_categorical  # , plot_model
from tensorflow.keras.layers import Dense, Dropout, GlobalAveragePooling2D
```

```
8  from tensorflow.keras.layers import Convolution2D, MaxPooling2D, Dense, BatchNormalization,
       Dropout, GlobalAveragePooling2D
9  from tensorflow.keras.models import Sequential, Model, load_model
10 from tensorflow.keras.callbacks import CSVLogger, Callback
11 # from tensorflow.keras.callbacks import ModelCheckpoint, ReduceLROnPlateau,
       LearningRateScheduler
12 from tensorflow.keras.preprocessing.image import ImageDataGenerator, apply_affine_transform
13 from tensorflow.keras.optimizers import SGD
14 # from tensorflow.keras.optimizers.schedules import ExponentialDecay
15 from tensorflow.keras.applications import InceptionV3
16 from tensorflow import math as tfmath
17 # from sklearn.model_selection import train_test_split
18
19 np.random.seed(1313)
20 tf.random.set_seed(1313)
21
22 def main():
23     global min_val_loss
24
25     # path_data = "C:/Users/kevin/git-workspace/tf-platelets/Data/"
26     # path_output = "C:/Users/kevin/git-workspace/tf-platelets/outputtest/"
27     path_data = "/mmfs1/home/beussk/platelet_factin_tf/Data/"
28     path_output = "/mmfs1/home/beussk/platelet_factin_tf/output2022-08-07/"
29
30     epochs = 200  # total epochs to run up to
31     batch_size = 96  # number of images per batch
32
33     tol = 1 # % of pixels to saturate (i.e. 5 means keep pixels within 5-95%)
34
35     initial_learning_rate = 0.01
36     final_learning_rate = 0.0001
37
38     load_checkpoint = True
39
40     custom_model = False
41     fine_tune = True
42     fine_tune_start = 0
43
44     if custom_model:
45         SIZE_ROWS = 96 # 299 for inceptionv3
46         SIZE_COLS = 96 # 96 for custom model
47         SIZE_CHAN = 1 # use 1 for custom model, 3 for pretrained model
```

```
48      else:
49          SIZE_ROWS = 299 # 299 for inceptionv3
50          SIZE_COLS = 299 # 96 for custom model
51          SIZE_CHAN = 3 # use 1 for custom model, 3 for pretrained model
52
53      ######################
54      decay_rate = tfmath.log(final_learning_rate /
55                              initial_learning_rate)/(epochs-1)
56
57      def load_data(import_dir):
58          train_dir = os.path.join(import_dir,'Training')
59          val_dir = os.path.join(import_dir,'Validation')
60          data_labels = os.listdir(train_dir)
61          data_labels.sort()
62
63          num_train_samples = []
64          num_val_samples = []
65          for label in data_labels:
66              num_train_samples.append(len([1 for x in list(os.scandir(os.path.join(train_dir,
                    label))) if x.is_file() and os.path.splitext(x)[1] == ".png"]))
67              num_val_samples.append(len([1 for x in list(os.scandir(os.path.join(val_dir,
                    label))) if x.is_file() and os.path.splitext(x)[1] == ".png"]))
68          tot_train_samples = sum(num_train_samples)
69          tot_val_samples = sum(num_val_samples)
70
71          X_train_data = np.ndarray((tot_train_samples, SIZE_ROWS, SIZE_COLS, SIZE_CHAN),
                dtype=np.uint8)
72          Y_train_data = np.ndarray((tot_train_samples,), dtype=np.uint8)
73          X_val_data = np.ndarray((tot_val_samples, SIZE_ROWS, SIZE_COLS, SIZE_CHAN), dtype=np
                .uint8)
74          Y_val_data = np.ndarray((tot_val_samples,), dtype=np.uint8)
75
76          for j, label in enumerate(data_labels):
77              pct = 0
78              train_image_names = [x for x in list(os.scandir(os.path.join(train_dir, label)))
                    if x.is_file() and os.path.splitext(x)[1] == ".png"]
79              val_image_names = [x for x in list(os.scandir(os.path.join(val_dir, label))) if
                    x.is_file() and os.path.splitext(x)[1] == ".png"]
80
81              print(f'Loading {label}...', end='')
82              for i, image_name in enumerate(train_image_names):
83
```

```
84                # img = cv2.imread(os.path.join(import_dir, label, image_name), cv2.
                      IMREAD_COLOR)
85                img = cv2.imread(os.path.join(import_dir, label, image_name), cv2.
                      IMREAD_GRAYSCALE)
86
87                img = cv2.resize(img, (40, 40))
88
89                # image corrections (contrast 5-95 percentile)
90                hist = cv2.calcHist([img], [0], None, [256], [0, 256])
91                cumhist = np.cumsum(hist)
92                total = img.size
93                low_bound = total * tol / 100  # low number of pixels to remove
94                upp_bound = total * (100-tol) / 100 # upp number of pixels to remove
95                lowb = None
96                uppb = None
97                for k, h in enumerate(cumhist):
98                    if h > low_bound and not lowb:
99                        lowb = k
100                   if h > upp_bound and not uppb:
101                       uppb = k-1
102               img = np.clip(img, lowb, uppb)
103               cv2.normalize(img, img, 0, 255, cv2.NORM_MINMAX)
104
105               img = cv2.resize(img, (SIZE_ROWS, SIZE_COLS))
106               if SIZE_CHAN == 3:
107                   img = cv2.cvtColor(img, cv2.COLOR_GRAY2RGB)
108               elif SIZE_CHAN == 1:
109                   img = img.reshape(SIZE_ROWS, SIZE_COLS, 1)
110
111               idx = i + sum(num_train_samples[:j])
112               X_train_data[idx] = np.array(img)
113               Y_train_data[idx] = j
114
115           for i, image_name in enumerate(val_image_names):
116
117               # img = cv2.imread(os.path.join(import_dir, label, image_name), cv2.
                      IMREAD_COLOR)
118               img = cv2.imread(os.path.join(import_dir, label, image_name), cv2.
                      IMREAD_GRAYSCALE)
119
120               img = cv2.resize(img, (40, 40))
121
```

```
122                 # image corrections (contrast 5-95 percentile)
123                 hist = cv2.calcHist([img], [0], None, [256], [0, 256])
124                 cumhist = np.cumsum(hist)
125                 total = img.size
126                 low_bound = total * tol / 100  # low number of pixels to remove
127                 upp_bound = total * (100-tol) / 100 # upp number of pixels to remove
128                 lowb = None
129                 uppb = None
130                 for k, h in enumerate(cumhist):
131                     if h > low_bound and not lowb:
132                         lowb = k
133                     if h > upp_bound and not uppb:
134                         uppb = k-1
135                 img = np.clip(img, lowb, uppb)
136                 cv2.normalize(img, img, 0, 255, cv2.NORM_MINMAX)
137
138                 img = cv2.resize(img, (SIZE_ROWS, SIZE_COLS))
139                 if SIZE_CHAN == 3:
140                     img = cv2.cvtColor(img, cv2.COLOR_GRAY2RGB)
141                 elif SIZE_CHAN == 1:
142                     img = img.reshape(SIZE_ROWS, SIZE_COLS, 1)
143
144                 idx = i + sum(num_val_samples[:j])
145                 X_val_data[idx] = np.array(img)
146                 Y_val_data[idx] = j
147             print(f'Done ({num_train_samples[j]} training images, {num_val_samples[j]}
                    validation images)')
148         Y_train_data = to_categorical(Y_train_data, len(num_train_samples))
149         Y_val_data = to_categorical(Y_val_data, len(num_val_samples))
150
151         data_weights = {}
152         for j, label in enumerate(data_labels):
153             data_weights[j] = (1 / num_train_samples[j])*(tot_train_samples / 2.0)
154
155         return X_train_data, Y_train_data, X_val_data, Y_val_data, data_labels, data_weights
156
157     def augment_data(X_train, Y_train, X_val, Y_val):
158         # def right_angle_rotate(input_image):
159         #     angle = np.random.choice([0, 90, 180, 270])
160         #     if angle != 0:
161         #         input_image = apply_affine_transform(
162         #             input_image, theta=angle, fill_mode='nearest')
```

```
163         #     return input_image
164
165         # train_datagen = ImageDataGenerator(
166         #     fill_mode = 'nearest',
167         #     preprocessing_function = right_angle_rotate,
168         #     rescale = 1./255, # i.e. get all pixels betwen 0-1, works for 8-bit image
                uint8 = 255
169         #     horizontal_flip = True, # mirror left/right
170         #     vertical_flip = True # mirror up/down
171         #     )
172
173         # train_datagen = ImageDataGenerator(
174         #     rescale=1./255, # i.e. get all pixels betwen 0-1, works for 8-bit image uint8
                = 255
175         #     rotation_range=45, # degrees to rotate image (from -45 to +45 degrees)
176         #     width_shift_range=0.1, # shift left/right by up to 10% image width
177         #     height_shift_range=0.1, # shift up/down by up to 10% image width
178         #     shear_range=0.1, # applies some image shearing transformations
179         #     zoom_range=0.5, # applies random zoom (e.g. if set to 0.5, yields zoom 0.5-1.5
                x)
180         #     horizontal_flip=True, # mirror left/right
181         #     vertical_flip=True, # mirror up/down
182         #     # brightness_range=[0.8,1.0], # modifies brightness
183         #     fill_mode='nearest')
184
185         train_datagen = ImageDataGenerator(
186             fill_mode='reflect',
187             rescale=1./255, # i.e. get all pixels between 0-1, works for 8-bit image uint8 =
                    255
188             rotation_range=45, # degrees to rotate image (from -45 to +45 degrees)
189             width_shift_range=0.05, # translate left/right
190             height_shift_range=0.05, # translate up/down
191             zoom_range=0.1, # zoom
192             horizontal_flip=True,  # mirror left/right
193             vertical_flip=True  # mirror up/down
194         )
195
196         # don't modify validation images
197         val_datagen = ImageDataGenerator(rescale=1./255)
198
199         train_datagen.fit(X_train, augment=True)
200
```

```
201         val_datagen.fit(X_val, augment=True)
202 
203         train_generator = train_datagen.flow(
204             X_train, Y_train,
205             batch_size=batch_size)
206 
207         val_generator = val_datagen.flow(
208             X_val, Y_val,
209             batch_size=batch_size,
210             shuffle=False)
211 
212         return train_generator, val_generator
213 
214     def define_custom_model():
215         model = Sequential()
216         model.add(BatchNormalization(input_shape=(SIZE_ROWS, SIZE_COLS, SIZE_CHAN)))
217         model.add(Convolution2D(64, (5, 5), padding='same', activation='relu'))
218         model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
219         model.add(Dropout(0.25))
220 
221         model.add(BatchNormalization())
222         model.add(Convolution2D(128, (5, 5), padding='same', activation='relu'))
223         model.add(MaxPooling2D(pool_size=(2, 2)))
224         model.add(Dropout(0.25))
225 
226         model.add(BatchNormalization())
227         model.add(Convolution2D(256, (5, 5), padding='same', activation='relu'))
228         model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
229         model.add(Dropout(0.25))
230 
231         model.add(GlobalAveragePooling2D())
232         model.add(Dense(256, activation='relu'))
233         # model.add(Activation('relu'))
234         model.add(Dropout(0.5))
235         model.add(Dense(len(class_labels), activation='softmax'))
236         # model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['
                accuracy'])
237         # lr_schedule = ExponentialDecay(learning_rate, decay_steps, decay_rate)
238 
239         # learning_rate_decay_factor = (final_learning_rate / initial_learning_rate)**(1/
                epochs)
240         # steps_per_epoch = int(train_generator.n/batch_size)
```

```
241         # lr_schedule = ExponentialDecay(
242         #                 initial_learning_rate=initial_learning_rate,
243         #                 decay_steps=steps_per_epoch,
244         #                 decay_rate=learning_rate_decay_factor,
245         #                 staircase=True)
246
247         # model.compile(optimizer=SGD(learning_rate=lr_schedule), loss='
                categorical_crossentropy', metrics=['accuracy'])
248
249         model.compile(optimizer=SGD(), loss='categorical_crossentropy', metrics=['accuracy'
                ])
250
251         return model
252
253     def define_pretrained_model():
254         # InceptionV3
255         base_model = InceptionV3(weights='imagenet', include_top=False, input_shape=(
                SIZE_ROWS, SIZE_COLS, SIZE_CHAN))
256         # base_model = Model(inputs=base_model.input, outputs=base_model.get_layer('
                conv5_block3_3_bn').output)
257
258         for layer in base_model.layers:
259             layer.trainable = False
260
261         x = base_model.output
262         x = GlobalAveragePooling2D()(x)
263         x = Dense(2048, activation='relu')(x)
264         x = Dropout(0.5)(x)
265
266         predictions = Dense(len(class_labels), activation='softmax', name='predictions')(x)
267
268         model = Model(inputs=base_model.input, outputs=predictions)
269
270         # learning_rate_decay_factor = (final_learning_rate / initial_learning_rate)**(1/
                epochs)
271         # steps_per_epoch = int(train_generator.n/batch_size)
272         # lr_schedule = ExponentialDecay(
273         #                 initial_learning_rate=initial_learning_rate,
274         #                 decay_steps=steps_per_epoch,
275         #                 decay_rate=learning_rate_decay_factor,
276         #                 staircase=True)
277
```

```
278         # model.compile(optimizer=SGD(learning_rate=lr_schedule), loss='
                categorical_crossentropy', metrics=['accuracy'])
279
280         model.compile(optimizer=SGD(), loss='categorical_crossentropy', metrics=['accuracy'
                ])
281
282         return model
283
284     X_train, Y_train, X_val, Y_val, class_labels, class_weight = load_data(path_data)
285
286     train_generator, val_generator = augment_data(X_train, Y_train, X_val, Y_val)
287
288     if load_checkpoint and os.path.exists(os.path.join(path_output, 'checkpoint.h5')):
289         print("Loading from existing checkpoint...")
290         model = load_model(os.path.join(path_output, 'checkpoint.h5'))
291
292         with open(os.path.join(path_output, 'history.csv'), "r") as f:
293             for last_line in f:
294                 pass
295             last_line = last_line[:-2].split(',')
296
297         initial_epoch = int(last_line[0]) + 1
298         min_val_loss = float(last_line[4])
299
300         append = True
301     else:
302         if not os.path.exists(path_output):
303             os.mkdir(path_output)
304
305         if custom_model:
306             model = define_custom_model()
307         else:
308             model = define_pretrained_model()
309
310         initial_epoch = 0
311         min_val_loss = float('inf')
312
313         model.summary()
314
315         append = False
316
317         # plot_model(model, to_file='test.png', show_shapes=True, show_layer_names=False)
```

```
318
319     class CustomCallback(Callback):
320         def on_epoch_begin(self, epoch, logs=None):
321             if epoch < epochs:
322                 new_lr = initial_learning_rate * tfmath.exp(decay_rate*epoch)
323             else:
324                 new_lr = final_learning_rate # this is used for fine-tuning
325             tf.keras.backend.set_value(model.optimizer.lr, new_lr)
326
327         def on_epoch_end(self, epoch, logs=None):
328             global min_val_loss
329
330             if logs['val_loss'] < min_val_loss:
331                 model.save(os.path.join(path_output, 'checkpoint.h5'))
332                 min_val_loss = logs['val_loss']
333                 logs['min_val_loss'] = logs['val_loss']
334                 logs['saved'] = '*'
335             else:
336                 logs['min_val_loss'] = min_val_loss
337                 logs['saved'] = ''
338
339             current_lr = model.optimizer.lr.numpy()
340             logs['learning_rate'] = current_lr
341
342     csv_callback = CSVLogger(os.path.join(path_output, 'history.csv'),
343         separator=',',
344         append=append)
345
346     if initial_epoch < epochs:
347         history = model.fit(
348             train_generator,
349             steps_per_epoch=max(1, train_generator.n // batch_size),
350             epochs=epochs,
351             initial_epoch=initial_epoch,
352             validation_data=val_generator,
353             validation_steps=max(1, val_generator.n // batch_size),
354             callbacks=[CustomCallback(), csv_callback],
355             class_weight=class_weight,
356             verbose=2
357         )
358         initial_epoch = epochs
359
```

```
360     if not custom_model and fine_tune:
361         for layer in model.layers[fine_tune_start:]:
362             if not isinstance(layer, tf.keras.layers.BatchNormalization):
363             # if layer.__module__ is not 'keras.layers.normalization.batch_normalization':
364             # might need to skip the batch normalization layers
365                 layer.trainable = True
366 
367         model.compile(optimizer=SGD(), loss='categorical_crossentropy', metrics=['accuracy'
                ])
368 
369         csv_callback.append = True
370 
371         history_fine = model.fit(
372             train_generator,
373             steps_per_epoch=max(1, train_generator.n // batch_size),
374             epochs=2*epochs,
375             initial_epoch=initial_epoch,
376             validation_data=val_generator,
377             validation_steps=max(1, val_generator.n // batch_size),
378             callbacks=[CustomCallback(), csv_callback],
379             class_weight=class_weight,
380             verbose=2
381         )
382 
383     predictions = model.predict(val_generator)
384     predicted_classes = np.argmax(predictions, axis=1)
385     actuals = val_generator.y
386     actual_classes = np.argmax(actuals, axis=1)
387     confusion = tfmath.confusion_matrix(actual_classes, predicted_classes)
388     print('Final trained validation predictions:')
389     print('(predictions on top, actual on right)')
390     print(class_labels)
391     print(confusion.numpy()) # predictions on top, actual on right
392 
393 if __name__ == "__main__":
394     main()
```

A.4.2 Classifying new images

```
1 import os
2 import cv2
```

```
3 from tensorflow.keras.models import load_model
4 from tensorflow.keras.preprocessing.image import ImageDataGenerator
5 import numpy as np
6 import csv
7 from pathlib import Path
8
9 def main():
10
11     path_data = "G:/.shortcut-targets-by-id/1SYh5QCYA01sjD05rNMxSCwqUTrxombOD/Platelet
           higher resolution analysis - collab with Adithan Kandasamy, Kevin Beussman, Molly
           Mollica/New data - May 2022/FactinMorphology_analysis"
12     # path_output = path_data
13     path_output = "C:/Users/kevin/OneDrive/Desktop/test/"
14
15     # path_model = "C:/Users/kevin/git-workspace/tf-platelets/"
16     # path_model = "C:/Users/kevin/OneDrive/Desktop/output2022-06-08_2"
17     path_model = "G:/My Drive/Machine learning platelets/output2022-08-07"
18
19     model = load_model(os.path.join(path_model, 'platelet_classifier.h5'))
20
21
22     class_labels = {0:"Hollow", 1:"Nodules", 2:"Solid"}
23
24     SIZE_ROWS = 299 # 299 for inceptionv3
25     SIZE_COLS = 299 # 96 for custom model
26     SIZE_CHAN = 3 # use 1 for custom model, 3 for pretrained model
27
28     tol = 1 # % of pixels to saturate (i.e. 5 means keep pixels within 5-95%)
29
30     image_names = [x for x in list(os.scandir(path_data)) if x.is_file() and os.path.
           splitext(x)[1] == ".png"]
31     image_names.sort(key=lambda x: x.name)
32     num_images = len(image_names)
33     X_data = np.ndarray((num_images, SIZE_ROWS, SIZE_COLS, SIZE_CHAN), dtype=np.uint8)
34
35     for i, image_name in enumerate(image_names):
36         img = cv2.imread(os.path.join(path_data, image_name), cv2.IMREAD_GRAYSCALE)
37
38         img = cv2.resize(img, (40, 40))
39
40         # image corrections (contrast 5-95 percentile)
41         hist = cv2.calcHist([img], [0], None, [256], [0, 256])
```

```
        cumhist = np.cumsum(hist)
        total = img.size
        low_bound = total * tol / 100  # low number of pixels to remove
        upp_bound = total * (100-tol) / 100 # upp number of pixels to remove
        lowb = None
        uppb = None
        for k, h in enumerate(cumhist):
            if h > low_bound and not lowb:
                lowb = k
            if h > upp_bound and not uppb:
                uppb = k-1
        img = np.clip(img, lowb, uppb)
        cv2.normalize(img, img, 0, 255, cv2.NORM_MINMAX)

        img = cv2.resize(img, (SIZE_ROWS, SIZE_COLS))
        if SIZE_CHAN == 3:
            img = cv2.cvtColor(img, cv2.COLOR_GRAY2RGB)
        elif SIZE_CHAN == 1:
            img = img.reshape(SIZE_ROWS, SIZE_COLS, 1)

        X_data[i] = np.array(img)

    X_datagen = ImageDataGenerator(rescale=1./255)
    X_datagen.fit(X_data, augment=True)
    X_generator = X_datagen.flow(
        X_data,
        batch_size=32,
        shuffle=False)

    predictions = model.predict(X_generator, verbose=1)
    predicted_classes = np.argmax(predictions, axis=1)
    counts_classes = np.bincount(predicted_classes)

    with open(os.path.join(path_output, 'classifications.csv'), 'w', newline='') as f1:
        writer1 = csv.writer(f1)
        writer1.writerow(['Total', class_labels[0], class_labels[1], class_labels[2]])
        writer1.writerow(['', counts_classes[0], counts_classes[1], counts_classes[2]])
        writer1.writerow(['filename', 'classification', 'max class', class_labels[0],
            class_labels[1], class_labels[2]])
        for i in range(len(predictions)):
            cv2.imwrite(os.path.join(path_output, 'p' + class_labels[predicted_classes[i]] +
                 f'{predictions[i][predicted_classes[i]]:.2f}' + '_' + image_names[i].name),
```

```
                X_data[i,:,:,0])
82            writer1.writerow([image_names[i].name, class_labels[predicted_classes[i]],
                  predictions[i][predicted_classes[i]]] + list(predictions[i]))
83
84    # for i in range(len(predictions)):
85    #     img = X_data[i]
86    #     cv2.imshow('image', img)
87    #     cv2.setWindowTitle('image', f'{class_labels[predicted_classes[i]]} {predictions[i
          ][predicted_classes[i]]:.3f}')
88    #     cv2.waitKey(0)
89
90 if __name__ == "__main__":
91     main()
```

A.5 KLayout geometry macros

These are a set of KLayout geometry macros that will auto-generate layout files for use with cleanroom mask writers. The code for generating black dots allows for changing the size and spacing of the dots, and automatically interspaces numerical identifiers throughout the black dots pattern. The code for generating microposts creates smaller arrays of posts that are separated by large blocks which help support the posts during the stamping process.

A.5.1 Black dots

```
1  import pya
2  import math
3  import time
4
5  pya.MainWindow.instance().create_layout(1)
6  view = pya.LayoutView.current()
7  cv = pya.CellView.active()
8  layout = cv.layout()
9
10 now = time.strftime('%Y%m%d_%H%M%S', time.localtime(time.time()))
11 savename = "C:\\Users\\kevin\\OneDrive\\Desktop\\kb2_blackdots_" + now + ".gds"
12
13 wafer = layout.create_cell("Wafer")
14 while False:
```

```
15    # create the wafer representation (outer circle = wafer edge, inner circle = working area
          limit)
16    l1 = layout.layer("Silicon")
17    r = 50000 / layout.dbu # 50 mm radius wafer
18    da = math.pi * 2 / 64
19    pts = []
20    for i in range(64):
21      pts.append(pya.Point.from_dpoint(pya.DPoint(r*math.cos(i*da), r*math.sin(i*da))))
22    pts.append(pya.Point.from_dpoint(pya.DPoint(r, 0)))
23
24    r = 40000 / layout.dbu # 40 mm radius working area
25    da = math.pi * 2 / 64
26    for i in range(64):
27      pts.append(pya.Point.from_dpoint(pya.DPoint(r*math.cos(i*da), r*math.sin(i*da))))
28    pts.append(pya.Point.from_dpoint(pya.DPoint(r, 0)))
29    wafer.shapes(l1).insert(pya.Polygon(pts))
30
31  def create_circle_array_with_IDs(cell,n,d,dx,dy,totalw,totalh,nxid,nyid):
32    # create one circle
33    circ = layout.create_cell("Circle,D=" + str(d))
34    l1 = layout.layer("SU8")
35    r = d / 2 / layout.dbu
36    da = math.pi * 2 / n
37    pts = []
38    for i in range(n):
39      pts.append(pya.Point.from_dpoint(pya.DPoint(r*math.cos(i*da), r*math.sin(i*da))))
40    circ.shapes(l1).insert(pya.Polygon(pts))
41
42    # creat circle arrays
43    nx = (totalw)//dx # total number dots x-direction
44    ny = (totalh)//dy # total number dots y-direction
45    idw = totalw//200//dx # number of dots wide for ID spot
46    dx = dx / layout.dbu
47    dy = dy / layout.dbu
48    dxid = dx*(nx//nxid)
49    dyid = dy*(ny//nyid)
50    #idw = 10 # number of dots wide for ID spot
51    idwx = idw
52    idwy = idwx*7//11 + (idwx % (7/11) &gt; 0) # 7/11 is the height/width aspect ratio of 2-
          character wide text
53
54    smallx = idwx
```

```
55  smally = ny//nxid-idwy
56  bigx = nx//nxid-idwx
57  bigy = ny - ny%(smally + idwy)
58
59  # create arrays for each large area
60  for k in range(nxid):
61    tx = k*dxid + idw*dx
62    trans = pya.Trans(pya.Point(tx,0))
63    ax = pya.Point(dx,0)
64    ay = pya.Point(0,dy)
65    cell.insert(pya.CellInstArray(circ.cell_index(), trans, ax, ay, bigx, bigy))
66
67  # fill in the gaps
68  for j in range(nyid):
69    for k in range(nxid):
70      tx = k*dxid
71      ty = j*dyid + idwy*dy
72      trans = pya.Trans(pya.Point(tx,ty))
73      ax = pya.Point(dx,0)
74      ay = pya.Point(0,dy)
75      cell.insert(pya.CellInstArray(circ.cell_index(), trans, ax, ay, smallx, smally))
76
77  # create text IDs
78  txt_layer = pya.LayerInfo("SU8")
79  layout.layer(txt_layer)
80
81  txtmag = 6*idw/2*dx*layout.dbu/4
82  for j in range(nyid):
83    for k in range(nxid):
84      param = { "layer": txt_layer, "text": str(k) + str(j), "mag": txtmag }
85      txtcell = layout.create_cell("TEXT", "Basic", param)
86      w = txtcell.bbox().width()
87      h = txtcell.bbox().height()
88      cx = dx*(idwx-1)/2 - w/2
89      cy = dy*(idwy-1)/2 - h/2
90      trans = pya.Trans(pya.Point(k*dxid + cx,j*dyid + cy))
91      post.insert(pya.CellInstArray.new(txtcell.cell_index(), trans))
92
93  print("actual w = " + str((bigx - 1 + smallx - 1 + 2)*dx*nxid*layout.dbu - 1*dx*layout.dbu
        + d) + " microns")
94  print("actual h = " + str((bigy - 1)*dy*layout.dbu + d) + " microns")
95
```

```
96  # parameters
97  n = 16 # points in circle
98  totalw = 25000 # total pattern width (micron)
99  totalh = 25000 # total pattern height (micron)
100 nxid = 10 # number of ID spots x-direction
101 nyid = 10 # number of ID spots y-direction
102
103 d = 1 # diameter (micron)
104 dx = 2 # spacing x direction (micron)
105 dy = 2 # spacing y direction (micron)
106 post = layout.create_cell("Post,D" + str(d) + "_SP" + str(dx) + "x" + str(dy))
107 create_circle_array_with_IDs(post,n,d,dx,dy,totalw,totalh,nxid,nyid)
108 # trans = pya.Trans(pya.Point(-(3000 + totalw) / layout.dbu,3000 / layout.dbu)) # translate
        up/left
109 trans = pya.Trans(90, True, pya.Point(-3000 / layout.dbu,3000 / layout.dbu)) # translate up/
        left
110 wafer.insert(pya.CellInstArray.new(post.cell_index(), trans))
111
112 d = 2 # diameter (micron)
113 dx = 4 # spacing x direction (micron)
114 dy = 4 # spacing y direction (micron)
115 post = layout.create_cell("Post,D" + str(d) + "_SP" + str(dx) + "x" + str(dy))
116 create_circle_array_with_IDs(post,n,d,dx,dy,totalw,totalh,nxid,nyid)
117 trans = pya.Trans(90, True, pya.Point((3000 + totalw) / layout.dbu,3000 / layout.dbu)) #
        translate up/right
118 wafer.insert(pya.CellInstArray.new(post.cell_index(), trans))
119
120 d = 3 # diameter (micron)
121 dx = 6 # spacing x direction (micron)
122 dy = 6 # spacing y direction (micron)
123 post = layout.create_cell("Post,D" + str(d) + "_SP" + str(dx) + "x" + str(dy))
124 create_circle_array_with_IDs(post,n,d,dx,dy,totalw,totalh,nxid,nyid)
125 trans = pya.Trans(90, True, pya.Point(-3000 / layout.dbu, -(3000 + totalh) / layout.dbu)) #
        translate down/left
126 wafer.insert(pya.CellInstArray.new(post.cell_index(), trans))
127
128 d = 10 # diameter (micron)
129 dx = 20 # spacing x direction (micron)
130 dy = 20 # spacing y direction (micron)
131 post = layout.create_cell("Post,D" + str(d) + "_SP" + str(dx) + "x" + str(dy))
132 create_circle_array_with_IDs(post,n,d,dx,dy,totalw,totalh,nxid,nyid)
```

```
trans = pya.Trans(90, True, pya.Point((3000 + totalw) / layout.dbu, -(3000 + totalh) / layout.dbu)) # translate down/left
wafer.insert(pya.CellInstArray.new(post.cell_index(), trans))

# cleanup
view.add_missing_layers()
cv.cell = wafer
view.zoom_fit()

opt = pya.SaveLayoutOptions()
opt.write_context_info = False
layout.write(savename,opt)
print(savename + " Done")</text>
</klayout-macro>
```

A.5.2 Microposts

```
import pya
import math
import time

pya.MainWindow.instance().create_layout(1)
view = pya.LayoutView.current()
cv = pya.CellView.active()
layout = cv.layout()

now = time.strftime('%Y%m%d_%H%M%S', time.localtime(time.time()))
savename = "C:\\Users\\kevin\\OneDrive\\Desktop\\kb2_posts_" + now + ".gds"

wafer = layout.create_cell("Wafer")
while False:
  # create the wafer representation (outer circle = wafer edge, inner circle = working area limit)
  l1 = layout.layer("Silicon")
  r = 50000 / layout.dbu # 50 mm radius wafer
  da = math.pi * 2 / 64
  pts = []
  for i in range(64):
    pts.append(pya.Point.from_dpoint(pya.DPoint(r*math.cos(i*da), r*math.sin(i*da))))
  pts.append(pya.Point.from_dpoint(pya.DPoint(r, 0)))

```

```
24    r = 40000 / layout.dbu # 40 mm radius working area
25    da = math.pi * 2 / 64
26    for i in range(64):
27      pts.append(pya.Point.from_dpoint(pya.DPoint(r*math.cos(i*da), r*math.sin(i*da))))
28    pts.append(pya.Point.from_dpoint(pya.DPoint(r, 0)))
29    wafer.shapes(l1).insert(pya.Polygon(pts))
30
31  def create_circle_array(cell,d,dx,dy,bignx,bigny,wid,blockw,gap):
32    # create one circle
33    circ = layout.create_cell("Circle,D=" + str(d))
34    l1 = layout.layer("SU8")
35    r = d / 2 / layout.dbu
36    da = math.pi * 2 / n
37    pts = []
38    for i in range(n):
39      pts.append(pya.Point.from_dpoint(pya.DPoint(r*math.cos(i*da), r*math.sin(i*da))))
40    circ.shapes(l1).insert(pya.Polygon(pts))
41
42    # creat circle arrays
43    nx = wid//dx # total number dots x-direction
44    ny = wid//dy # total number dots y-direction
45    dx = dx / layout.dbu
46    dy = dy / layout.dbu
47
48    blockw = blockw / layout.dbu
49    gap = gap / layout.dbu
50
51    lx = dx*nx+2*r
52    ly = dy*ny+2*r
53    # create arrays for each large area
54    for j in range(bigny):
55      for k in range(bignx):
56        tx = (blockw+gap)*(k+1) + gap*k + lx*k + r
57        ty = (blockw+gap)*(j+1) + gap*j + ly*j + r
58        trans = pya.Trans(pya.Point(tx,ty))
59        ax = pya.Point(dx,0)
60        ay = pya.Point(0,dy)
61        cell.insert(pya.CellInstArray(circ.cell_index(), trans, ax, ay, nx, ny))
62
63        # add block below
64        pts = [pya.Point(tx-r,ty-r-gap-blockw),pya.Point(tx+lx-r,ty-r-gap-blockw),pya.Point(tx
              +lx-r,ty-r-gap),pya.Point(tx-r,ty-r-gap)]
```

```
65        cell.shapes(l1).insert(pya.Polygon(pts))
66
67        if j == bigny-1:
68          topmost = ty+ly-r+gap+blockw
69
70          # add block to left of column, using final value of ty
71          pts = [pya.Point(tx-r-gap-blockw,0),pya.Point(tx-r-gap,0),pya.Point(tx-r-gap,topmost
                ),pya.Point(tx-r-gap-blockw,topmost)]
72          cell.shapes(l1).insert(pya.Polygon(pts))
73
74          # add blocks above top row, using final value of ty
75          pts = [pya.Point(tx-r,topmost-blockw),pya.Point(tx+lx-r,topmost-blockw),pya.Point(tx
                +lx-r,topmost),pya.Point(tx-r,topmost)]
76          cell.shapes(l1).insert(pya.Polygon(pts))
77
78          # if on the last column, add a block to the right as well
79          if k == bignx-1:
80            rightmost = tx+lx-r+gap+blockw
81            pts = [pya.Point(rightmost-blockw,0),pya.Point(rightmost,0),pya.Point(rightmost,
                  topmost),pya.Point(rightmost-blockw,topmost)]
82            cell.shapes(l1).insert(pya.Polygon(pts))
83
84
85  # parameters for all
86  n = 32 # points in circle
87  bignx = 10
88  bigny = 10
89  blockw = 275
90  gap = 30
91  wid = 2000
92
93  # parameter for array 1
94  d = 3 # diameter (micron)
95  dx = 6 # spacing x direction (micron)
96  dy = 6 # spacing y direction (micron)
97  actualwid = wid//dx*dx
98  totalw = (blockw+gap+actualwid)*bignx + gap + blockw
99  totalh = (blockw+gap+actualwid)*bigny + gap + blockw
100 post = layout.create_cell("Post,D" + str(d) + "_SP" + str(dx) + "x" + str(dy))
101 create_circle_array(post,d,dx,dy,bignx,bigny,wid,blockw,gap)
102 trans = pya.Trans(pya.Point(-(3000+totalw)/layout.dbu,3000/layout.dbu)) # translate up/left
103 wafer.insert(pya.CellInstArray.new(post.cell_index(), trans))
```

```
104
105 trans = pya.Trans(pya.Point(3000/layout.dbu,3000/layout.dbu)) # translate up/right
106 wafer.insert(pya.CellInstArray.new(post.cell_index(), trans))
107
108 d = 2 # diameter (micron)
109 dx = 6 # spacing x direction (micron)
110 dy = 6 # spacing y direction (micron)
111 actualwid = wid//dx*dx
112 totalw = (blockw+gap+actualwid)*bignx + gap + blockw
113 totalh = (blockw+gap+actualwid)*bigny + gap + blockw
114 post = layout.create_cell("Post,D" + str(d) + "_SP" + str(dx) + "x" + str(dy))
115 create_circle_array(post,d,dx,dy,bignx,bigny,wid,blockw,gap)
116 trans = pya.Trans(pya.Point(-(3000+totalw)/layout.dbu,-(3000+totalh)/layout.dbu)) #
      translate down/left
117 wafer.insert(pya.CellInstArray.new(post.cell_index(), trans))
118 trans = pya.Trans(pya.Point(3000/layout.dbu,-(3000+totalh)/layout.dbu)) # translate down/
      right
119 wafer.insert(pya.CellInstArray.new(post.cell_index(), trans))
120
121 # cleanup
122 view.add_missing_layers()
123 cv.cell = wafer
124 view.zoom_fit()
125
126 opt = pya.SaveLayoutOptions()
127 opt.write_context_info = False
128 layout.write(savename,opt)
129 print(savename + " Done")</text>
130 </klayout-macro>
```