DNA Origami Breadboard: A Platform for Cell Activation and Cell Membrane Functionalization

THESIS

Presented in Partial Fulfillment of the Requirements for the Degree Master of Science in the Graduate School of The Ohio State University

By

Molly Y. Mollica, B.S.

Graduate Program in Mechanical Engineering

The Ohio State University

2016

Master’s Examination Committee:

Dr. Carlos Castro, Adviser

Dr. Jonathan Song
Abstract

Structural DNA Nanotechnology (“DNA origami”) techniques have enabled the design and synthesis of complex 3D nanostructures with dynamically controllable features that exploit molecular self-assembly principles. Any component that can be conjugated to an oligonucleotide (oligo) can be attached to a DNA nanostructure at a specific location and quantity with nanometer resolution. This includes some fluorescent dyes, quenchers, peptides, RNA, steroids, vitamins, and, by extension, all molecules capable of biotinylation. A DNA origami “breadboard” with 34 strategically located attachment points can therefore be functionalized with a wide variety of components and used for a multitude of purposes. In this thesis, the design, fabrication, purification, characterization, and application of a 68 x 25 x 6 nanometer honeycomb lattice DNA nanostructure will be presented for use in two distinct functions. In the first, a biotinylated antibody was added to the platform and used to better mimic a cell-to-cell receptor-ligand interaction with tunable antibody quantity, location, and flexibility (i.e. range of motion). This led to determination that ligand flexibility, which can be controlled using DNA origami, influences strength of cell activation. In the second, cholesterol-modified oligonucleotides were added to cells and used to anchor the nanostructures onto the cell surface. The ability to integrate DNA origami nanostructures into a cell membrane can

ii
enable a wide variety of applications such as a intracellular force sensing, programmed cell-cell adhesion, or triggered recruiting of biomolecules from solution.
Acknowledgments

I am honored and grateful to have many individuals to acknowledge. First, I would like to thank Dr. Carlos Castro and Dr. Chris Lucas for all of their support, guidance, and patience. When I committed to a summer-long research experience in the Nanoengineering Biodesign Lab (NBL), I had no idea that I would stay in the lab for the remaining part of my undergraduate education, earn a Master of Science degree, and adjust my plans from medical school to graduate school. None of this would have been possible without Dr. Castro giving me the opportunity to be a part of the lab, Dr. Lucas training me on all I need to know, and both of them guiding me for the last three years and through the PhD application process. I would also like to thank Dr. Jonathan Song for his support and expertise throughout the NBL-MMM collaboration, for being a part of my committee, and for his guidance in my post-graduation plans.

I am extremely thankful to be a member of Team LPP with Dr. Chris Lucas and Randy Patton and Team CellCro with Ehsan Akbari. The intellectual contributions and data collection of Chris, Randy, and Ehsan are included throughout this thesis. I would also like to acknowledge Emily Briggs for her nanostructure design.
I would like to thank my NBL officemates: Chris Lucas, Jenny Le, and Patrick Halley, as well as the rest of my lab mates: Randy Patton, Alex Marras, Josh Johnson, Mike Hudoba, and Lifung Zhou for keeping things interesting and for all that they have taught me in and out of the lab. I would also like to thank Dr. Song’s Microsystems for Mechanobiology and Medicine Laboratory.

I would like to acknowledge the Campus Microscopy & Imaging Facility in which the TEM and confocal imaging presented here was conducted. I would also like to thank the National Science Foundation, the Department of Mechanical and Aerospace Engineering, the Institute for Materials Research, the College of Engineering, and the University Honors & Scholars Center for their funding and support.

Beyond my research in NBL, I would like to thank Dr. Keith Gooch for putting up with me in three of his courses, teaching me to read journal articles, and making me think more critically than I have before. I would like to thank Dr. Shiyong Wu and Dr. Rob Siston for allowing me to get started with research and for helping me vastly improve my communication skills. I would like to thank Dr. Sandra Metzler for her mentorship over the last three years. I am incredibly grateful to Liz Riter for allowing me to be a part of some of the greatest things happening in the College of Engineering, hiring me (at least) six times, and ultimately being a mentor and friend for the last six years. I would also like to thank my Green Engineering Scholars students for keeping me on my toes and
teaching me all the new slang. Finally, I would like to thank my friends and family for their encouragement and support every step of the way.
Vita

June 2010 ................................................. Athens High School
August 2014 .................................................. B.S. Biomedical Engineering, The Ohio State University
2014-present ................................................. Graduate Administrative Associate, The Ohio State University

Publications

Molly Y. Mollica, Ehsan Akbari, Chris Lucas, Jonathan Song, and Carlos Castro. “Cell Membrane Functionalization with DNA Origami Nanostructures.” In Preparation, authorship to be determined.


Fields of Study

Graduate Major Field: Mechanical Engineering
Undergraduate Major Field: Biomedical Engineering
# Table of Contents

Abstract......................................................................................................................................................... ii

Acknowledgments ................................................................................................................................................ iv

Vita ................................................................................................................................................................... vii

Table of Contents ................................................................................................................................................ viii

List of Tables ....................................................................................................................................................... x

List of Figures .................................................................................................................................................... xi

Chapter 1: Introduction ........................................................................................................................................ 1

DNA and Structural DNA Nanotechnology ........................................................................................................ 1

Ligand Presentation Platform ................................................................................................................................ 7

CellCro: Plasma Membrane DNA Origami Attachment .................................................................................... 15

Chapter 2: Materials and Methods ..................................................................................................................... 17

Structure, Cell, Imaging, and Analysis Methods Common to LPP and CellCro.................................................. 17

Design, Fabrication, and Confirmation of Well-Folded Structures .................................................................... 17

Transmission Electron Microscopy ................................................................................................................... 18

Atomic Force Microscopy ................................................................................................................................... 19
List of Tables

Table 1: Breadboard Oligo Plate 1 ................................................................. 60
Table 2: Breadboard Oligo Plate 2 ................................................................. 63
Table 3: Sequences Used to Bind to the Plasma Membrane ....................... 66
Table 4: Sequences Used to Fluorescently Label the Nanostructure or Membrane ...... 66
Table 5: LPP 5T, 15T, and 30T Sequences ..................................................... 67
List of Figures

Figure 1: DNA Double Helix: The chemical structure of DNA, including its 5’ to 3’ directionality, its DNA base pairing, and its natural negative charge are important parameters utilized to fold DNA Origami [2]................................................................. 2

Figure 2: DNA Dimensions: DNA Origami exploits the understanding of the structure of DNA including the diameter of a double helix and the distance and number of bases that make up one turn within the double helix [3]................................................................. 3

Figure 3: DNA Origami Folding: The single stranded DNA scaffold (~7500 bases long) is folded together with single stranded DNA staples (~40 bases long) that are designed and synthesized to bind to specific locations on the scaffold. Using this method, it is possible to form 2D and 3D DNA origami structures [6] [10] [12] [13]......................... 6

Figure 4: LPP Functionalization Introduction: The Ligand Presentation Platform is folded by adding staples, including one biotinylated staple, to scaffold. The LPP is then functionalized using streptavidin and a biotin-labeled antibody to bind an antibody to a specific location on the platform................................................................. 9

Figure 5: LPPs Better Mimic Cell-Cell Interactions: The cellular response caused by ligands are generally studied by presenting ligands to cells in solution. The LPP better
mimics a cell-to-cell interaction by immobilizing the ligand and presenting it with a specific quantity and location relative to other ligands.

Figure 6: LPP Functional Parameters: The LPP makes it possible to study the influence of ligands in solution vs. ligands immobilized on the platform (A), ligand quantity (B), ligand location (C), and ligand flexibility (D.)

Figure 7: Previous Studies Using DNA Origami To Present Ligands [19, 20]: Previous work has immobilized ligands on DNA structures (top) and has studied the influence of quantity and location on cell activation (bottom.)

Figure 8: Previous Nanostructure Attachment to Plasma Membranes: Previous work has bound nanostructures to plasma membranes for the study of nanopores (A), membrane diffusion (B), isotropic-nematic transition (C), and control of DNA nanostructure actuation (D). However, all of this work was completed on artificial plasma membranes and not on living cells.

Figure 9: Nanostructure Attachment to Membrane Protocol: In order to attach DNA nanostructures to a living plasma membrane, nanostructures were removed from the folding reaction and PEG purified. Fluorescently labeled oligos were added and the excess was removed with PEG purification. Meanwhile, cholesterol-labeled oligos were added to cells such that they could imbed into the membrane. These were incubated and washed and a ssDNA bridge was added to bind to the cholesterol-modified oligo. Additional incubation and washing was completed and the purified fluorescently-labeled nanostructure was added to the cells.
Figure 10: DNA Breadboard Design in caDNAno: The scaffold is shown in blue, the 22 bottom overhangs are shown in red and pink, and the 12 top overhangs are shown in green.

Figure 11: Breadboard Binding Locations: As indicated by the blue dots, there are 12 binding locations on top and 22 on bottom of the DNA Breadboard.

Figure 12: MgCl$_2$ Folding Screen: Agarose gel electrophoresis was utilized to determine the optimum amount of magnesium to be added to the DNA folding reaction when folding the DNA Breadboard. The optimal amount was determined to be 20mM.

Figure 13: DNA Breadboard Visualization: TEM and AFM: Two forms of electron microscopy were utilized to visualize the DNA Breadboard. In panel A, a TEM image is shown in which the scale bar is 100nm. In panel B, an AFM image is shown in which the height sensor is from 0.0 to 1.0 um.

Figure 14: Stability of Nanostructures in RPMI Cell Culture Media: In order to confirm DNA Breadboard stability in cell culture conditions, agarose gel electrophoresis with EtBr visualizing total DNA and typhoon visualizing the fluorescent label, was completed with varying levels of FBS. It was determined that the DNA Breadboard is stable in RPMI with 2% FBS but not in RPMI with 10% FBS.

Figure 15: LPP Functionalization Visualization: TEM was used to confirm functionalization of the DNA nanostructure with streptavidin (A), isotype control antibody (B), and anti-CD40 (C). The scale bar in these images is 50nm.

Figure 16: Confirmation of Increased Ligand Range of Motion with Increased Linker Length: Panel A shows the corner reference point used to determine the ligand flexibility.
of the varying length (between 5 bases (5T), 15 bases (15T), and 30 bases (30T)) at the
central point. Panel B shows the distribution as it increases with increasing ligand length.
Panel C displays examples of the streptavidin with increasing distance from the platform
with increasing ligand length. The scale bars in this figure are 100nm.

Figure 17: LPP Binding to B cells: With increasing concentration of fluorescently-labeled
DNA nanostructures, increased binding is observed. This confirms that functionalized
LPP are able to bind to cells. In this image, the scale bar is 10um.

Figure 18: LPP Binding Analysis: A home-built MATLAB code was utilized to quantify
and plot binding around the periphery/circumference of the cell. The colored arrows
pointing to the cell and the graph correspond with each other where the red arrow is the
start/end point, the black arrow indicates the highest intensity, and the gray arrow
indicates the lowest intensity. The dotted line indicates the average.

Figure 19: Linker Length Alterations Does Not Change Nanostructure Binding: The
MATLAB code was used to confirm that equal binding exists for varying levels of linker
length. Each point on the graph is the average intensity around one cell. As determined
by a two-way, student t-test, there is no significant difference in binding.

Figure 20: Linker Length Influences Cell Activation: The cell activation, as measured by
GFP-NFKB reporter activity, is higher for anti-CD40 than the isotype control. In
addition, this preliminary data shows decreased binding with increased ligand flexibility.

Figure 21: Oligo Attachment to Cell Membrane Via Cholesterol-Conjugated Oligo:
Cholesterol-conjugated oligonucleotides were added to HUVECs and visualized with
complementary fluorescently-labeled oligos to confirm that cholesterol imbedded into the plasma membrane. This was assessed and quantified using the MATLAB code. The scale bar in these images is 5um. ............................................................. 47

Figure 22: CellCro Binding is Specific and Controllable on B cells: Nanostructures were bound to CH12.LX cells and showed increased binding relative to three controls: no nanostructures, nanostructures but with no cholesterol added, nanostructures and cholesterol but with a blocking DNA sequence added to lose binding function. The scale bar in these images is 10um. ............................................................. 48

Figure 23: Nanostructure Binding Quantification: The MATLAB code was used to confirm the binding that was qualitatively displayed in Figure 22. Intensity was measured around the circumference of the cell, this was plotted, and the average intensity of each (dotted line) was plotted as one point on the graph on the right. As determined by a two-way, student t-test, there is a significant difference between the positive stimulus (nanostructures with cholesterol) and the three controls (no nanostructures, no cholesterol, and ssDNA blocked binding.) ............................................................. 49

Figure 24: Confocal Imaging of Nanostructures on Cell Membrane: Confocal Microscopy of CellCro on CH12.LX cells with (left) and without (right) cholesterol. The top image of each sample is a 3D rendering while the bottom shows x-, y-, and z-cuts. ......................... 50

Figure 25: CellCro Attachment to Endothelial Cells With and Without Enzyme Treatment: Preliminary data suggests that nanostructure binding is increased with (vs. without) cholesterol on HUVEC with no additional treatments (left.) However, with enzyme treatment to remove the glycocalyx, increased binding occurs (right.) ............... 51
Figure 26: LPP caDNAno Image ................................................................. 101

Figure 27: LPP caDNAno Cross Section ....................................................... 102
Chapter 1: Introduction

DNA and Structural DNA Nanotechnology

DNA (deoxyribose nucleic acid) is composed of nitrogenous bases, five-carbon sugars, and phosphate groups. While DNA is well known for encoding biological information and forensic applications, its intrinsic properties have allowed for the foundation of an entirely new field of bionanotechnology, structural DNA nanotechnology, where the assembly properties of DNA are exploited to construct objects with complex designed geometries.

The molecular structure of DNA, as shown below in Figure 1 and Figure 2, is a twisted ladder shape known as a double helix where the rungs of the ladder are formed by nucleotide bases, adenine (A), thymine (T), cytosine (C), and guanine (G), and the legs of the ladder are formed by the sugar-phosphate backbone. Each of these nucleotide bases has a binding partner: A with T and C with G, which are Watson-Crick base-pairs [1]. The assembly of the double helix is facilitated by hydrogen bonding in a manner such that when the nucleotides find their binding partner, the backbones of each strand run
anti-parallel. This directionality is typically denoted as running 5’ (pronounced “five prime”) to 3’ where this number corresponds to the number of carbons in the sugar.

Figure 1: DNA Double Helix: The chemical structure of DNA, including its 5’ to 3’ directionality, its DNA base pairing, and its natural negative charge are important parameters utilized to fold DNA Origami [2]
Figure 2: DNA Dimensions: DNA Origami exploits the understanding of the structure of DNA including the diameter of a double helix and the distance and number of bases that make up one turn within the double helix [3]
The design and fabrication of DNA nanostructures exploits the well-understood assembly properties of nucleic acids to design two and three-dimensional nanoscale structures. The field of DNA nanotechnology was founded with Ned Seeman’s work on nucleic acid junctions and lattices in the early 1980’s [4] and his subsequent synthesis of a DNA molecule with cube connectivity in the early 1990’s [5]. The term “DNA origami” was coined by Paul Rothemund in 2006 when he reported the folding of several 2D DNA shapes including a star and smiley face (shown in Figure 3) [6]. Since then, cumulative DNA nanotechnology related citations have been published at an exponential rate, and Rothemund’s original paper [6] has gained nearly 3000 citations in just 10 years. A variety of shapes and functions have been proposed for a wide range of potential applications, as reviewed in “The enabled state of DNA nanotechnology [7]” and “Structural DNA Nanotechnology: State of the Art and Future Perspectives [8].”

DNA origami is typically designed in an open source software, caDNAno (cadnano.org) and is created by adding many 30-50 base single stranded DNA (ssDNA) oligomer “staples” to a long (~7000-8000 bases) circular ssDNA “scaffold” with a known sequence, such that the oligomers bind to the scaffold in a piece-wise manner to create a compact, high precision geometry as shown in Figure 3 [6]. The scaffold is typically derived from the single-stranded genome of the M13MP18 bacteriophage. The sample is initially heated to melt all base-pairing interactions followed by slow cooling over several days to room temperature to facilitate annealing in a buffered solution with MgCl₂ to screen charge repulsions of the negatively charged backbone phosphate groups [9]. The
staples bind to the scaffold via complementary base pairing in a piece-wise manner to create a seemingly infinite number of possible 2D or 3D [10] structures and a wide variety of functionalities that are commercially available.

In this work, we will exploit the geometric precision of DNA origami as well as the ability to add chemical functionality [11] in order to precisely control the number and spatial arrangement of ligands specific for cell surface receptors and cholesterol-modified oligos [6].
Figure 3: DNA Origami Folding: The single stranded DNA scaffold (~7500 bases long) is folded together with single stranded DNA staples (~40 bases long) that are designed and synthesized to bind to specific locations on the scaffold. Using this method, it is possible to form 2D and 3D DNA origami structures [6] [10] [12] [13]
Cells transfer information through receptor-ligand interactions. A ligand molecule on one cell binds to a protein receptor on another cell, often causing a conformational change that propagates information across the membrane leading to a cascade of chemical reactions, and, ultimately, signal transduction. Strength of signal induction is essential to determine downstream cellular function and responses mediated by cell surface receptors including B cell antigen receptors [14]. Understanding receptor-ligand and, more broadly, cell-cell interactions is important therapeutically because it will allow for precise intervention and control of cell fate. However, studying these receptor-ligand interactions in a biologically relevant manner is currently highly challenging. Frequently, ligands are placed in solution to study the cell’s response to them. However, this disregards a variety of factors that are well accepted to influence cellular response including ligand clustering and mechanosensitive responses [15, 16].

In order to better mimic a natural receptor-ligand interaction during a cell-cell interaction, it is possible to employ a novel nanotechnological device recently developed in Dr. Carlos Castro’s laboratory in the Department Mechanical and Aerospace Engineering. This device is known as a Ligand Presentation Platform (LPP). An LPP is a 68 nm x 25 nm x 6 nm DNA origami structure [6, 9] that is hypothesized to provide proper engagement configuration and physical support while introducing a ligand to cell surface receptors. This approach makes it possible to examine the influence biophysical properties of force [15], ligand quantity, ligand spatial arrangements [17], and precise
combinations of ligands because the 3 x 4 grid of potential ligand-anchoring positions have a distinct DNA sequence, making it possible to attach any combination of up to 12 ligands at any length from the platform with a variety of spacing for smaller numbers of ligands. In this thesis, only one ligand is attached to the breadboard and a biotin-streptavidin interaction is used. The biotinylated ligands can attach to the biotinylated nucleotide staple due to the addition of streptavidin and the high affinity of streptavidin for biotin. However, it is notable that because streptavidin has four biotin binding sites, another method may be necessary when attaching more than one ligand to prevent attachment of unintended ligands to one streptavidin. As depicted in Figure 4, streptavidin attaches to the platform via a connection to the biotin staple and once streptavidin is attached to the platform, it still presents biotin-binding sites to solution. This makes it possible to stably attach biotin-labeled ligands (biotinylated antibodies) [18] to the rigid DNA structure in any number and location.
Figure 4: LPP Functionalization Introduction: The Ligand Presentation Platform is folded by adding staples, including one biotinylated staple, to scaffold. The LPP is then functionalized using streptavidin and a biotin-labeled antibody to bind an antibody to a specific location on the platform.

With this design, as shown in Figure 5 and 6, LPPs better mimic cell-cell interactions by immobilizing ligands (Figure 5 and 6A) and presenting them with control over quantity (Figure 6B), spatial arrangement (Figure 6C), and range of motion (Figure 6D.)
Figure 5: LPPs Better Mimic Cell-Cell Interactions: The cellular response caused by ligands are generally studied by presenting ligands to cells in solution. The LPP better mimics a cell-to-cell interaction by immobilizing the ligand and presenting it with a specific quantity and location relative to other ligands.
Figure 6: LPP Functional Parameters: The LPP makes it possible to study the influence of ligands in solution vs. ligands immobilized on the platform (A), ligand quantity (B), ligand location (C), and ligand flexibility (D.)
Previous work has demonstrated that DNA Origami has unprecedented control over number and arrangement of ligands [19] (Figure 7, top) and that quantity and arrangement influence strength of signal transduction. As shown in the bottom of Figure 7, it was found that, generally, higher quantities of ligands and ligands spaced further apart led to increased signal strength [20]. Interestingly, Shaw et al. [19] observed the opposite trend. This suggests that the physical details of activation may vary dependent on the receptor and ligand pair.
Previous work has immobilized ligands on DNA structures (top) and has studied the influence of quantity and location on cell activation (bottom.)

However, DNA Origami has not yet been utilized to examine how linker length (Figure 6D) influences signal transduction. In this thesis, the design, purification, functionalization, and activation of cells using a DNA Origami Ligand Presentation
Platform with varying linker lengths is described. The objective of this study is to develop techniques and facilitate those techniques to examine the influence of ligand flexibility on cellular activation.
CellCro: Plasma Membrane DNA Origami Attachment

In order to enhance our understanding of interactions and phenomena that govern membrane function and behavior, multiple attempts have been made to anchor DNA nanostructures to artificially synthesized vesicles. As shown in Figure 8, successful attachment of DNA nanostructures to artificial plasma membranes has allowed for the study of nanopore synthesis and function (Figure 8A) [21], lipid bilayer diffusion (Figure 8B) [22], isotropic-nematic transition (Figure 8C) [23], and DNA nanostructure actuation (Figure 8D) [24].

Figure 8: Previous Nanostructure Attachment to Plasma Membranes: Previous work has bound nanostructures to plasma membranes for the study of nanopores (A), membrane diffusion (B), isotropic-nematic transition (C), and control of DNA nanostructure
actuation (D). However, all of this work was completed on artificial plasma membranes and not on living cells.

While all of this work demonstrates attachment to artificial plasma membranes, no group has yet reported successful DNA nanostructure attachment to living cell plasma membranes. This may be due to obstacles such as membrane-bound protein interference, cellular endocytosis, and DNA nanostructure instability in cell culture conditions. Despite these impediments, this attachment would allow for the examination of all of the aforementioned phenomena as well as for intercellular force-related interactions. This thesis describes the design, purification and labeling, cell membrane attachment optimization, and attachment quantification of the CellCro DNA nanostructure.
Chapter 2: Materials and Methods

Structure, Cell, Imaging, and Analysis Methods Common to LPP and CellCro

*Design, Fabrication, and Confirmation of Well-Folded Structures*

The DNA breadboard nanostructure was designed using the computer-aided design software caDNAno [25] utilizing the honeycomb lattice to form a 68nm x 25nm x 6nm platform with 34 strategically-located oligonucleotide overhangs (12 on the top and 22 on the bottom.) Staple sequences were outputted from caDNAno and ordered from a commercial vendor (Eurofins Genomics, Louisville, KY). Structures were constructed via molecular self-assembly by protocols developed in Castro et al. with the p7560 clone of the M13mp18 genome with 164 single-stranded staples. Briefly, purified p7560 scaffold (20 nM) was combined with staples at 10-fold molar excess (each staple at 200nM) in a 1X FOB solution (5mM Tris, 5mM NaCl 1mM EDTA) with 20mM MgCl₂. Note that all aforementioned concentrations indicated the final concentration and not the concentration at the time of mixing. The self-assembly reactions were conducted in a thermal cycler (Bio-Rad, Hercules, CA) by rapidly heating the folding reaction solution to 65°C followed by slow cooling to 4°C over 2.5 days. Specifically, the folding reaction was brought to 65°C and stepped from 65°C to 24°C in 1°C steps at varying time increments.
From 65°C to 62°C, the temperature was held for one hour; from 61°C to 59°C, the temperature was held for two hours; from 58°C to 46°C, the temperature was held for three hours; from 45°C to 40°C, the temperature was held for one hour; from 39°C to 24°C, the temperature was held for thirty minutes. After being held at 24°C for 1 hour, the temperature was lowered to 4°C. To confirm and purify well-folded structures, folding reaction products subjected to agarose gel electrophoresis in a 2% agarose (Life Technologies) gel (0.5X TBE) in the presence of 11mM MgCl$_2$ and 1uM ethidium bromide (EtBr) followed by band excision and centrifugation in “Freeze ‘N Squeeze” purification tubes (BioRAD, Hurcules, CA). Alternatively, folding reactions were purified by adding equivolume folding reaction products and 15% PEG 8000 (Sigma-Aldrich, St Louis, MO) with 500mM NaCl were mixed and then centrifuged for 30 minutes at 16000 rcg to remove excess staple strands (protocol slightly modified from [26].) Purified structures were resuspended in 1 X FOB with 20mM MgCl$_2$ and fluorescent oligonucleotides (Alexa647, Alexa488, Cy3, Cy5) were added at 5X excess with respect to the number of designated binding sites and incubated for at least 30 minutes. To fully remove the excess fluorescent oligonucleotides, the PEG purification procedure was repeated two times.

*Transmission Electron Microscopy*

TEM grids were prepared as described in Castro et al. [9]. Briefly, 4uL of ~1nM nanostructure was deposited onto a copper TEM grid coated with carbon and formvar (Electron Microscopy Sciences, Hartfield, PA) and incubated for 3 minutes. Filter paper
was used to wick away the solution and 10uL of 2% uranyl formate negative stain was applied, immediately wicked away as a washing step, and 20uL of 2% uranyl formate was immediately added and incubated for 40 seconds. The grid was allowed to dry for at least 30 minutes before being visualized on a Tecnai G2 BioTWIN transmission electron microscope (FEI, Hillsboro, OR) at an electron acceleration voltage of 80kV. This imaging was conducted in the Campus Microscopy & Imaging Facility.

Atomic Force Microscopy

A Bruker AXS Dimension Icon atomic force microscope in ScanAsyst mode (Bruker Corporation, Billerica, MA) was used to collect AFM images. A 5uL drop of DNA nanostructure was placed on the surface of freshly cleaved mica and incubated for 2 minutes. Samples were then washed with 1mL ddH₂O and dried with compressed nitrogen. Collection of images was completed using a ScanAsyst-Air silicon nitride cantilever with a measured spring constant of 0.79 N/m and nominal tip radius of 2nm (Bruker Corporation, Billerica, MA). This imaging was conducted in the NanoSystems Lab in the Ohio State Department of Physics.

Nanostructure Purification

Equivolume folding reaction products and 15% PEG 8000 (Sigma-Aldrich, St Louis, MO) with 500mM NaCl were mixed and then centrifuged for 30 minutes at 16000 rcg to remove excess staple strands. Purified structures were resuspended in 1 X FOB with 18 or 20mM MgCl₂ and fluorescent oligonucleotides (Alexa647, Alexa488, Cy3) were added at 5X excess with respect to the number of designated binding sites and incubated
for at least 30 minutes. To remove the excess fluorescent oligonucleotides, the PEG purification procedure was repeated two additional times after incubation with the fluorescence oligonucleotides.

**Confirmation of Stability in Cell Culture Conditions**

In order to confirm the structural integrity and stable fluorescent labeling of the nanostructures under cell culture conditions, structures were PEG purified, resuspended in cell culture media supplemented with various levels of FBS and MgCl₂, and the structures were incubated for up to 10 days at 37°C. Agarose gel electrophoresis was conducted and the gel was imaged on a Typhoon FLA 9500 (General Electric, Fairfield, CT) to visualize fluorescence from fluorophore-labeled oligonucleotides as well as being imaged on a UV table with EtBr to visualize DNA. In addition to visualization on the gel, bands were excised and imaged on the TEM as described above.

**CH12.LX B Cell Culture**

CH12.LX is a murine B cell lymphoma provided by Dr. Gail Bishop and described previously [27]. CH12.LX cells were maintained in supplemented RPMI in 37°C incubator at 5% CO₂. In order to culture the CH12.LX B cells, the RPMI 1640 without L-glutamine (Corning) was supplemented with 10% heat-inactivated Fetal Bovine Serum (FBS) (Atlas Biologicals), 1% Penicillin-Streptomycin-Glutamine (100X) (Thermo-Fisher), 1% Sodium Pyruvate (100mM) (Life Technologies), 1% HEPES (1M) (Life-Technologies), 1% MEM Non-Essential Amino Acids (100X) (Life Technologies), and 0.1% Beta ME. For experimental purposes, CH12.LX cells were washed twice with 1X
PBS without MgCl₂/CaCl₂ (Corning, Cat# 21-031-CV), followed by one wash in the experimental medium with the corresponding FBS supplementation level and MgCl₂. Finally, the washed CH12-LX cells were resuspended in the experimental medium at 500 cells/uL for incubation with cholesterol-conjugated oligos.

**Imaging on Total Internal Reflection (TIRF) Microscope**

Live cell fluorescent imaging was performed on a Nikon TiE (Belmont, CA) using a 100X oil immersion objective. An ultra-thin 8-well imaging plate (LAB-TEK) was used to image the samples in order to enable TIRF imaging near the surface of the glass. The cells were maintained at 37°C and 5% CO₂ during the imaging using a stage-top incubator (Okolab, Pozzuoli, Italy). For general wide field microscopy, the same setup was used by realigning the excitation angle. Fluorescent excitation was provided by 488 nm (100 mW source), 561 nm (100 mW source), or 640 nm (50 mW source) lasers. Depending on the type of fluorescent dye the CellCro nanostructures were labeled with, the excitation wavelength was adjusted accordingly. Images were acquired using an ANDOR EMCCD camera at 1024*1024 resolution.

**Cell Periphery Fluorescent Analysis**

A home-built MATLAB code was developed to analyze the fluorescent signal observed under epifluorescent imaging. The code provides a user interface to manually select points around the periphery of a cell of interest. The code then measures the intensity of each pixel and reports the average intensity around the circumference of each cell. This code is provided in Appendix B.
LPP Methods

*LPP Functionalization with Streptavidin and Biotinylated Antibodies*

Platform nanostructures were designed to contain up to 12 potential biotinylated-oligos with varying extension lengths (5, 15, 30 thymine nucleotides) on a 4 x 3 grid allowing for streptavidin binding and subsequent biotinylated antibody attachment at precisely defined locations. After one PEG purification as described above, structures were mixed with Alexa Flour 555-labeled streptavidin (Life Technologies) in ~40-fold molar excess for 1 hour at room temperature followed by agarose gel electrophoresis as described above except that due to overlap in excitation and emission fields of Alexa 555 and EtBr, SYBR Gold was used to post-stain the gel for DNA visualization. This gel post-staining included diluting SYBR Gold to 1X with TAE buffer and completely submerging the agarose gel in this solution for 10-40 minutes, depending on the concentration of DNA in the gel. Leading bands were excised and purified as described above and visualized with TEM to confirm streptavidin attachment. In addition, single molecule photobleaching analysis was performed on Alexa Fluor 555-labeled streptavidin:LPP structures using total internal reflection (TIRF) microscopy to confirm the existence of an individual streptavidin attachment per nano platform. Using structures that were PEG purified but not gel purified, two additional PEG purifications were completed and biotinylated anti-CD40 (1C10) or species-/matched-isotype control Abs were added at a ~3-fold molar excess and incubated for at least 1 hour at 37°C. Two PEG purifications were completed as previously described except that the 15% PEG 8000 contained 50mM NaCl instead of...
500mM NaCl. After the final PEG precipitation, structures were resuspended in imaging media (clear RPMI supplemented with 1% P/S/G, 5mM MgCl$_2$, and .5% FBS.) Samples were subjected to gel electrophoresis as described above where leading bands were excised and gel purified. Antibody attachment was confirmed by TEM and Typhoon imaging by conjugating an Alexa Fluor 647 fluorophore to biotinylated Abs (Life Technologies) according to manufacturer’s instructions. Structure bands were visible under 630 nm excitation.

*Activation of B cells with LPP*

To evaluate intracellular signaling induced by nano-platforms functionalized with anti-CD40 Abs at the single cell level, 1 x 10$^6$ CH12.LX B cells were transfected via nucleofection (Amaxa, program 0-003) with 2.5-5 μg of the Cignal GFP-NF-κB reporter construct (SA Biosciences, Qiagen) and incubated overnight in complete medium at 37°C containing 5% CO$_2$. 250-500 x 10$^3$ CH12.LX cells were resuspended in assay medium (clear RPMI 1640 (Gibco) 0.5% FBS, + 100 U/ml penicillin (Gibco), 100 μg/ml streptomycin (Gibco) and 2 mM L-glutamine (Gibco), cultured in live cell imaging wells (Thermo Scientific) followed by the addition of Ab functionalized LPPs (600 pM/1 nM), soluble Abs (600 pM/1 nM), or varying concentrations of LPS and incubated in a temperature, humidity, and CO$_2$ controlled stage top incubator at physiologic conditions over a 20 hour time course in the presence of 7-AAD (1:100). The level of GFP-NF-κB activity and presence of 7-AAD was monitored at 10-minute intervals for 12 hours through epifluorescence imaging at 488 and 561 nm respectively by fluorescence
microscopy (Nikon, Eclipse Ti2). In some experiments, varying concentrations of Bay-11-7082 (Santa Cruz Biotechnology) the IKKα inhibitor, was added to the CH12.LX cells for 30 minutes prior to LPP addition to block NF-κB activation. To visualize LPP DNA nanostructures bound to CH12.LX B cells, nanostructures were either functionalized with Alexa 647 labeled oligos at a 2-fold molar excess to structures or by fluorescent streptavidin (Alexa 488, Alexa 555, or Alexa 647) as described in the functionalization protocol above.

**CellCro Methods**

**HUVEC Culture**

Human Umbilical Vein Endothelial Cells (HUVECs) were maintained in endothelial growth medium (EGM, Lonza) in 37°C incubator at 5% CO₂. The HUVECs with passage numbers between 10-15 were washed with 1X PBS without MgCl₂/CaCl₂ (Corning), followed by tripsonization using 0.05% Trypsin-EDTA (1X) (Corning) in order to detach the ECs from the cell culture flask. The tripsonized HUVECs were then washed once with the experimental medium with the corresponding FBS (Atlas Biologicals, Ft. Collins, CO) supplementation levels from .5% to 2% and MgCl₂ concentration from 0mM to 2mM. Finally, the HUVECs were resuspended in the experimental medium at 500 cells/uL for incubation with cholesterol-conjugated oligos.
**HUVECs Treatment with Heparinase III**

Heparinase III from Flavobacterium heparinum was purchased from Sigma (Sigma-Aldrich) in order to selectively degrade heparin sulfate, which is the major component of the glycocalyx layer on the HUVECs. The enzyme was reconstituted at 1 Units/mL inside 20mM Tris – HCl, pH 7.5, containing 0.1mg/mL BSA and 4mM CaCl$_2$. Before nanostructure addition, the HUVECs were washed once with 1X DPBS without MgCl$_2$/CaCl$_2$ and incubated with 20mU/mL Heparinase III diluted in EGM for 2 hours. The incubated cells were then washed twice with EGM followed by one wash with 1X DPBS and tripsonized in order to be prepared for the CellCro binding experiment.

**CellCro DNA Nanostructure Cell Surface Incorporation**

The experimental medium used for the incorporation of the CellCro nanostructures on the HUVEC membrane consisted of Endothelial Base Medium (Lonza) supplemented with 0.5% heat inactivated FBS (Atlas Biologicals) and 1% Penicillin-Streptomycin-Glutamine (P/S/G) (100X) (Thermo-Fisher). CH12.LX B cells were cultured with clear RPMI 1640 without phenol red (Corning) supplemented with 0.5% heat inactivated FBS (Atlas Biologicals) and 1% Penicillin-Streptomycin-Glutamine (100X) (Thermo-Fisher). Cells were resuspended in the experimental medium with additional 2 mM MgCl$_2$ at 500 cells/µL. The addition of MgCl$_2$ was to ensure the stability of the DNA based CellCro nanostructures throughout the preparation process. The cells were then incubated with 10 uM 3’ cholesterol-conjugated oligos (sequence in Appendix A) for 5 minutes at 37°C. The cells were then washed once in the experimental medium to remove the unbound cholesterol-conjugated oligos. In order to inspect the incorporation of the cholesterol-
conjugated oligos on the cell membrane, the washed HUVECs were then resuspended in the experimental medium with additional 2mM MgCl₂ and incubated with 1uM Cy5 conjugated oligos (sequence in Appendix A) with the reverse complement sequence for the cholesterol-conjugated oligos already anchored in the membranes. In order to attach the 60 base pair bridge oligos to the cholesterol-conjugated oligos that are tethered to the cell membrane, the cells were resuspended in the experimental medium with additional 2mM MgCl₂ at 500 cells/uL and incubated with 1uM 60 nucleotide bridge oligos (sequence in Appendix A) for 5 minutes at 37°C. The incubation with the bridge oligos was followed by the addition of the 20 base pair bridge fortifier oligos (sequence in Appendix A) at 1uM and further incubation for 5 minutes at 37°C. In order to remove the excess bridge and fortifier oligos, the incubated cells were washed once with the experimental medium with additional 2mM MgCl₂ and resuspended at 500 cells/uL using the experimental medium without any additional MgCl₂. Finally, the HUVECs were incubated with CellCro nanostructures at 5nM for 5 minutes at 37°C. The high concentration stock of CellCro nanostructures was at 50nM, as determined by a Nanodrop (Thermo Scientific) inside 1X FoB with 20mM MgCl₂. The CellCro nanostructures were added to the HUVECs at a 1:10 volumetric ration where the final MgCl₂ concentration was 2mM. In order to remove the extra nanostructures and eliminate the background signal during imaging, the HUVECs were washed twice using the experimental medium with additional 2mM Mg and resuspended in experimental medium at 250 cells/uL. The final reconstitute of the cells was then transferred to an 8-well imaging dish. This method is summarized in Figure 9.
Figure 9: Nanostructure Attachment to Membrane Protocol: In order to attach DNA nanostructures to a living plasma membrane, nanostructures were removed from the folding reaction and PEG purified. Fluorescently labeled oligos were added and the excess was removed with PEG purification. Meanwhile, cholesterol-labeled oligos were added to cells such that they could imbed into the membrane. These were incubated and washed and a ssDNA bridge was added to bind to the cholesterol-modified oligo. Additional incubation and washing was completed and the purified fluorescently-labeled nanostructure was added to the cells.

**Imaging on Confocal Microscope**

Confocal images were recorded using a laser scanning confocal microscopy (Nikon A1R) with a 100X oil immersion objective. The samples were maintained at 37°C and 5% CO₂
using a stage-top incubator. Ultra-thin 8 well imaging plates were used to enhance optical resolution. A laser type source of light was used to excite the fluorescent dyes attached to the CellCro nanostructures at the corresponding wavelength. Z-stack images were acquired using a 175 nm optical thickness. Image acquisition was performed using a EMCCD Hamamatsu camera at 1024*1024 resolution. In order to diminish the background noise, 2x averaging was used.
Chapter 3: Results

DNA Breadboard Design, Purification, and Visualization

This 68nm x 25nm x 6nm DNA Nanostructure was designed with a honeycomb lattice in caDNAno as shown in Figure 10. In this figure, the green lines on the top represent the overhangs that make up the 12 top binding spots and the 22 pink and red lines on the bottom represent the overhangs that make up the 22 potential binding spots on the bottom. These binding locations are also marked with blue dots on Figure 11. Due to each binding location having a unique sequence, any binding spot can be added or left out with each folding reaction. It is notable that on the top of the structure, the potential binding locations are a well-organized grid but that on the bottom, the spots are irregular. This is a result of maximizing the number of binding sites without reducing structural integrity by employing short oligonucleotides to make additional binding spots possible.
Figure 10: DNA Breadboard Design in caDNAno: The scaffold is shown in blue, the 22 bottom overhangs are shown in red and pink, and the 12 top overhangs are shown in green.
Magnesium chloride is ionic and the positively charged magnesium facilitates the negatively charged DNA binding to other negatively charged DNA during folding. A magnesium folding screen was completed to determine the optimal amount of magnesium to facilitate DNA breadboard folding. After a 2.5 day folding thermal ramp, as described in the materials and methods, agarose gel electrophoresis was completed to determine which amount of magnesium led to optimal folding. As shown in Figure 12, this works by placing the negatively charged DNA in the wells at the top of the gel and applying a current such that the negatively charged DNA migrates to the positive terminal. Because smaller pieces of DNA are able to migrate more quickly through the gel, the excess staple strands run the furthest while the compact, well-folded DNA forms a tight band and the misfolded structures form a smear that runs slower than the well-folded band. It was determined that 20 mM MgCl₂ is the optimal folding condition due to the high yield within the tight band.
Figure 12: MgCl₂ Folding Screen: Agarose gel electrophoresis was utilized to determine the optimum amount of magnesium to be added to the DNA folding reaction when folding the DNA Breadboard. The optimal amount was determined to be 20mM.

Transmission electron microscopy (TEM) and atomic force microscopy (AFM) was completed to visualize folded structures (Figure 13A and 13B.) In the TEM image in Figure 13A, the scale bar is 100nm while in the AFM image in Figure 13B, the height sensor extends from 0 to 1 μm.
Two forms of electron microscopy were utilized to visualize the DNA Breadboard. In panel A, a TEM image is shown in which the scale bar is 100nm. In panel B, an AFM image is shown in which the height sensor is from 0.0 to 1.0 um.

In order to confirm that the DNA breadboard is in cell culture conditions, gel electrophoresis was used to confirm stability, as shown in Figure 14. In this image, the top gel is DNA stained with EtBr and the bottom gel is imaging Alexa 647 oligos bound to the structure. This gel indicates the structures remain stable up through 48 hours with 2mM MgCl₂ and 2% FBS but some degradation occurs within 24 hours with 10% FBS including loss of the fluorescent overhangs.
Figure 14: Stability of Nanostructures in RPMI Cell Culture Media: In order to confirm DNA Breadboard stability in cell culture conditions, agarose gel electrophoresis with EtBr visualizing total DNA and typhoon visualizing the fluorescent label, was completed with varying levels of FBS. It was determined that the DNA Breadboard is stable in RPMI with 2% FBS but not in RPMI with 10% FBS.

**Ligand Presentation Platform**

The first application of the DNA Breadboard includes examining the hypothesis that local ligand flexibility impacts the level of cell activation. To do this, a biotinylated oligo and a biotin-streptavidin interaction was utilized to functionalize the DNA Breadboard with biotinylated CD40 Ligand, making the Breadboard the Ligand Presentation Platform.
(LPP.) After functionalization, single molecule fluorescence experiments were performed to monitor the response of live cells presented with CD40 ligand at varying ligand flexibilities.

The LPP was functionalized with streptavidin and anti-CD40. As shown in Figure 15, TEM was used to confirm effective functionalization of structures with streptavidin (Figure 15A) as well as structures with isotype control antibody (Figure 15B) and anti-CD40 (Figure 15C.) In these images, the scale bar is 50nm.

![Figure 15: LPP Functionalization Visualization](image)

To confirm that linker length altered range of motion of the ligand, TEM was also conducted. As shown in Figure 16A, measurements were made between a constant length 5 nucleotide linker in the corner and a variable length 5, 15, or 30 nucleotide linker in the
center. As expected, when the linker length increased, the streptavidin molecule had a larger range of motion, as shown in Figure 16B and C.
Figure 16: Confirmation of Increased Ligand Range of Motion with Increased Linker Length: Panel A shows the corner reference point used to determine the ligand flexibility of the varying length (between 5 bases (5T), 15 bases (15T), and 30 bases (30T)) at the central point. Panel B shows the distribution as it increases with increasing ligand length. Panel C displays examples of the streptavidin with increasing distance from the platform with increasing ligand length. The scale bars in this figure are 100nm.
After confirmation of structure functionalization and ability to alter linker length and range of motion, functionalized and fluorescently labeled LPPs were added in increasing concentrations to B cells. As shown in Figure 17, functionalized LPPs bound to the B cell surface, supporting the notion that LPP-bound biotinylated anti-CD40 can still engage CD40 on the B cell.

Figure 17: LPP Binding to B cells: With increasing concentration of fluorescently-labeled DNA nanostructures, increased binding is observed. This confirms that functionalized LPP are able to bind to cells. In this image, the scale bar is 10 um.

Binding was analyzed using a home-build MATLAB code. As shown in Figure 18, the image was imported, the background was subtracted, the user selected points around the cell, and the program measured all pixels along the path. Pixel intensity was plotted around the circumference of the cell. As indicated by the colored arrows in Figure 18, the quantitative plot of intensity around the circumference of the cell correlates with what is seen qualitatively. In this image, the red arrow indicates the start (and end) point, the
black arrow indicates the most intense part of the cell, and the gray arrow indicates the least intense part of the cell. The average of all of these pixel intensities was taken, as indicated by the dotted line in Figure 18.
Figure 18: LPP Binding Analysis: A home-built MATLAB code was utilized to quantify and plot binding around the periphery/circumference of the cell. The colored arrows pointing to the cell and the graph correspond with each other where the red arrow is the start/end point, the black arrow indicates the highest intensity, and the gray arrow indicates the lowest intensity. The dotted line indicates the average.
To confirm that altering ligand length does not alter its binding, this binding analysis was completed on LPP 5T, LPP 15T, and LPP 30T. This analysis is displayed in Figure 19 in which each point is the average intensity taken around a cell (as explained above). As determined by a two-tailed student t-test, no significant difference in binding exists with altered linker length.
Figure 19: Linker Length Alterations Does Not Change Nanostructure Binding: The MATLAB code was used to confirm that equal binding exists for varying levels of linker length. Each point on the graph is the average intensity around one cell. As determined by a two-way, student t-test, there is no significant difference in binding.
Although comparable binding exists for LPP 5T, 15T, and 30T, preliminary data suggests that cell activation, as measured by GFP-NF-κB reporter activity, is influenced by ligand flexibility. In all cases, as expected, the cell activation induced by anti-CD40 is greater than the isotype control antibody. It is notable that the cell activation decreases with increased ligand flexibility. This is displayed in Figure 20.
Figure 20: Linker Length Influences Cell Activation: The cell activation, as measured by GFP-NFκB reporter activity, is higher for anti-CD40 than the isotype control. In addition, this preliminary data shows decreased binding with increased ligand flexibility.

While DNA nanostructures have been previously utilized to present ligands and study cell-to-cell interaction [19, 20], this data is the first to report the use of DNA nanostructures to examine the influence of ligand flexibility on cell activation. This thesis
describes the nanostructure design, functionalization, and quantification of cellular response with varying ligand flexibility.

Given the ability to functionalize the DNA Breadboard with a variety of ligands in various quantities, locations, and flexibilities, this allows for the ability to study factors important in cell-to-cell communication in ways that are unmatched with standard methods. Ultimately, this can be used to deepen understanding and control cell fate experimentally or therapeutically.
CellCro: Plasma Membrane DNA Origami Attachment

The second application of the DNA Breadboard includes examining the hypothesis that it is possible to tune DNA nanostructure design and other methods in order to bind DNA nanostructures to the plasma membrane of a living cell. To do this, overhangs are bound to cholesterol-modified oligonucleotides that, due to their non-polar nature, imbed into the plasma membrane.

In order to confirm that cholesterol-modified oligos would imbed into the plasma membrane of a living cell, these cholesterol-modified oligos were added to HUVEC endothelial cells. Complementary fluorescently labeled oligos such that they would bind to the cholesterol-modified oligos. These were incubated, washed to remove excess, and visualized on a fluorescent microscope. As shown in Figure 21, binding occurred, confirming cholesterol imbedding into the plasma membrane. Because the MATLAB analysis code allows for selection of any shape, the non-round, adhered endothelial cells can be easily analyzed. Figure 21 shows selection of points around an adhered cell as well as a 3D plot of the x-y location of points and the pixel intensity. In this figure, the scale bar is 5um.
Cholesterol-conjugated oligonucleotides were added to HUVECs and visualized with complementary fluorescently-labeled oligos to confirm that cholesterol imbedded into the plasma membrane. This was assessed and quantified using the MATLAB code. The scale bar in these images is 5um.

In addition to the cholesterol-oligo embedding into the cell membrane, DNA nanostructures can be bound to that oligo. As shown in Figure 22, when no nanostructures are added, no fluorescence is visualized. When CellCro is added in the absence of cholesterol, nearly no binding occurs. However, when CellCro is added in the presence of cholesterol, fluorescent circles are visible around the cells, indicating binding to the plasma membrane. Finally, when a competing oligo that binds to the nanostructure overhang is added in order to block the nanostructure from binding to the cholesterol oligo, decreased binding is demonstrated. This exhibits controlled binding. That is, binding that occurs when expected but does not occur when cholesterol is absent or when an oligo is added to block binding. In this image, the scale bar is 10um.
Figure 22: CellCro Binding is Specific and Controllable on B cells: Nanostructures were bound to CH12.LX cells and showed increased binding relative to three controls: no nanostructures, nanostructures but with no cholesterol added, nanostructures and cholesterol but with a blocking DNA sequence added to lose binding function. The scale bar in these images is 10um.

The qualitative images in Figure 22 were quantified using the MATLAB periphery analysis. As shown in Figure 23, fluorescence around the periphery of the cell increases significantly with the positive stimulus, as compared to with all controls when examined using an unpaired, two-sided student t-test (p<.00001.)
Figure 23: Nanostructure Binding Quantification: The MATLAB code was used to confirm the binding that was qualitatively displayed in Figure 22. Intensity was measured around the circumference of the cell, this was plotted, and the average intensity of each (dotted line) was plotted as one point on the graph on the right. As determined by a two-way, student t-test, there is a significant difference between the positive stimulus (nanostructures with cholesterol) and the three controls (no nanostructures, no cholesterol, and ssDNA blocked binding.)

In addition to epifluorescence imaging of CH12.LX cells, confocal microscopy was also conducted to gain a 3D view and ensure that nanostructures were on the periphery of the cell. In comparing samples with cholesterol (positive stimulus) vs. without cholesterol (control), differences were qualitatively visible. As shown in Figure 24, the sample with cholesterol showed fluorescence around the periphery of the cell while the sample
without cholesterol showed less fluorescence and no pattern of fluorescence. This is visible in the 3D image as well as the x-, y-, and z- slices.

Figure 24: Confocal Imaging of Nanostructures on Cell Membrane: Confocal Microscopy of CellCro on CH12.LX cells with (left) and without (right) cholesterol. The top image of each sample is a 3D rendering while the bottom shows x-, y-, and z-cuts.
Endothelial cells proved more difficult to functionalize the membrane with nanostructures. It was hypothesized that this may be due to the glycocalyx layer that endothelial cells produce due to the nanostructure being ~70nm long while the distance between substructures of the glycocalyx bush is only 20nm [28]. To increase binding, endothelial cells were treated with a glycocalyx-removing enzyme, Heparinase III. As displayed in Figure 25, preliminary data suggests that attachment increases with enzyme treatment as compared to without the enzyme treatment.

![Without Enzyme Treatment](image1)

![With Enzyme Treatment](image2)

Figure 25: CellCro Attachment to Endothelial Cells With and Without Enzyme Treatment: Preliminary data suggests that nanostructure binding is increased with (vs. without) cholesterol on HUVECs with no additional treatments (left.) However, with enzyme treatment to remove the glycocalyx, increased binding occurs (right.)
While DNA nanostructures have been previously bound to artificial plasma membranes [21-24], this data is the first report of DNA nanostructures bound to living plasma cell membranes. This thesis describes the nanostructure design, methods development, qualitative binding as visualized by epifluorescence microscopy and confocal microscopy, and quantitative binding as computed by MATLAB.

Given the ability to functionalize both sides of the DNA Breadboard, this transforms the cellular plasma membrane into a surface that can be functionalized for a wide variety of purposes. This includes tuning strength of attachment to other cells or substrates, studying biological phenomena related to the plasma membrane, and examining the mechanics of cell invasion on the nano scale during metastasis or angiogenic sprouting.
Chapter 4: Conclusions and Future Work

In this thesis, the design, fabrication, purification, and application of a 68 x 25 x 6 nanometer hexagonal lattice DNA Breadboard was presented for use in two distinct functions. In the first, a biotinylated antibody was added to the platform and used to better mimic a cell-to-cell receptor-ligand interaction with tunable antibody quantity, location, and range of motion. In the second, cholesterol-modified oligonucleotides were added to the platform and used to cover a cell surface with DNA nanostructures.

Due to the wide variety of binding locations and oligo conjugation options, the DNA Breadboard can be utilized in a variety of applications. This is ideal for labs in which DNA origami is not a focus because with the investment of the oligos listed in Appendix A (~$800), p7560 scaffold (available from Tilibit, Garching, Germany for $125), and a temperature controlling instrument such as a Thermocycler or PCR, a Breadboard can be folded and used to attach a variety of components with precise control over location and quantity. Even in labs in which the focus is DNA Origami, designing nanostructures for use in a variety of applications is a financially efficient way to host a number of projects or to investigate a new idea and collect preliminary data without a large financial investment.
Future LPP work includes fully characterizing the influence of ligand length on strength of cellular response with all relevant controls. For example, one necessary control includes measuring cell response with the LPP attached to the isotype control antibody in all ligand lengths. In addition to ligand length, factors such as quantity, location, and the combination of these factors can be examined. LPPs could also be presented with an optical trap rather than in solution. Because the DNA Breadboard is able to be polymerized, the LPP can become larger so that a wider variety of quantity and location can be investigated. TEM, AFM, and/or single molecule experiments can be used to measure antibody functionalization efficiency and ensure that it is equal for all linker lengths. Ultimately, the refinement of this tool will allow it to be used to study poorly understood ligand-interactions. If cellular response changes with these aforementioned factors, the LPP could even be used therapeutically to deliver cell signals in a way in which the cells will respond more strongly.

Future CellCro work includes developing robust methods to attach DNA nanostructures to more cell types such other types of endothelial and immune cells, epithelial cells, neurons, etc. In addition, it is essential to utilize the ability to attach nanostructures to the membrane. This could be exploited to create an intercellular force sensor to measure cell-to-cell forces during angiogenesis and cancer metastasis or to adhere cells to various substrates such as the extracellular matrix.
Finally, the CellCro and LPP projects could be combined (CeILPP) to embed the ligand-attached DNA Breadboard into a cell’s membrane to present it to activate another cell. This would provide control over ligand quantity, location, and linker length while mimicking a cell-cell interaction even more accurately.

Ultimately, given the ability to functionalize both sides of the DNA Breadboard, attach it to cell plasma membranes, and present ligands to cells with it, this nanostructure is versatile and presents a wide variety of applications to study biological phenomena.
Bibliography


Appendix A: Oligonucleotide Library
The oligos necessary to fold the breadboard are listed in Table 1 and Table 2 below. The “Well” indicates their location within their respective plates EB_P040 and EB_P041. Bolded oligos are potential overhangs. All oligos are listed 5’ to 3’.

Table 1: Breadboard Oligo Plate 1

<table>
<thead>
<tr>
<th>Well</th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>CTATAAGGTGTTCATCGATAGCAGTCAGACGAGCCCCCGTGCGGA</td>
</tr>
<tr>
<td>A2</td>
<td>CTCCCGATTACCAAGCTTAAAGCACAGCG</td>
</tr>
<tr>
<td>A3</td>
<td>CTAAATCAAAGGCAAGAGCGTGCTTTATGTTACCAGAGGA</td>
</tr>
<tr>
<td>A4</td>
<td>TAAGCAGCCAATAATAATCGACGCCCTTTA</td>
</tr>
<tr>
<td>A5</td>
<td>ATGGATTTGAACAAAAAAGGCTACAACCACGAGCCCACGGGCTGTAATAGAT</td>
</tr>
<tr>
<td>A6</td>
<td>CAAAATCCGGAATTTCACCAGTCACGC</td>
</tr>
<tr>
<td>A7</td>
<td>AAGGAAAGCAACGAAAAGGGCCGACATAATAAAATCCGACCGGCCCCAGA</td>
</tr>
<tr>
<td>A8</td>
<td>ATTCGCGAATAGCAGAAGGGCCAAACAGGAGTCGCTG</td>
</tr>
<tr>
<td>A9</td>
<td>ATCGCAAATAAAAAATCATTTTTTACTGATGCTTATA</td>
</tr>
<tr>
<td>A10</td>
<td>CCAGCTTTGAGGGAGCAGATGTAAATCAGTATAAGAGAG</td>
</tr>
<tr>
<td>A11</td>
<td>AAGCAAAGGAAAGACAATAAAAACATTTATCATTTGGTACCT</td>
</tr>
<tr>
<td>A12</td>
<td>AGTAATAATTTGAATATACAAACCAGATACCAGTCCGTT</td>
</tr>
<tr>
<td>B1</td>
<td>AACCCTCTAAATGACCAGACCAGGAGCGAGTCACAATTCATAA</td>
</tr>
<tr>
<td>B2</td>
<td>GACCATAGAACAAGGCCGCTTTATTTAAT</td>
</tr>
<tr>
<td>B3</td>
<td>CGGGATCGCACTAGTATGATCTTGGAGGAGACAAAGGCCG</td>
</tr>
<tr>
<td>B4</td>
<td>TTTAACGGCTGAGATGATACCAGCTACGCCAGGGAATATTATATGAG</td>
</tr>
<tr>
<td>B5</td>
<td>AACGAGTAATTCTTGCTGAGAGAAG</td>
</tr>
<tr>
<td>B6</td>
<td>TAAATAAAAAACATCTTGCTTTTCAAGGAGTTCCAGGAGATA</td>
</tr>
<tr>
<td>B7</td>
<td>TTTTCAGTTAAGACCTACCGCTCTGTTTCTA</td>
</tr>
<tr>
<td>B8</td>
<td>CTCAGGACAGGAACTCAAACCCCTAC</td>
</tr>
<tr>
<td>B9</td>
<td>TGGCGCGTTACAGGGGTTAAAAGGAAAAG</td>
</tr>
<tr>
<td>B10</td>
<td>TAATCATAATTACAGTACTTACATGAGGAGGAAATG</td>
</tr>
<tr>
<td>B11</td>
<td>CGAAAGATAAACAGAAGCATTTAATTTGGAAG</td>
</tr>
<tr>
<td>B12</td>
<td>CCAGGGTAAATAACACGTGAGTCACCCG</td>
</tr>
<tr>
<td>C1</td>
<td>CCAAGAATCCTTTAGTAACATCCTTTGGCCTCTCCAGGATGCTGCTG</td>
</tr>
<tr>
<td>C2</td>
<td>ATCAGCGACCCTGAAATATTG</td>
</tr>
</tbody>
</table>

Continued on next page
Continued from previous page

C3  GCACCATTACCATTGGGAGGGGGGTCGAAGGGCGC
C4  GGGCGCTGGTGGCCGAAGTCCACTATTAAA
C5  GGATGGCAAAGCGATTTAGACTGGATAGAGCAACAAAAAGGAAAATCTA
C6  CCCMTTACAGAGAAGCGGTGTCTTCACTCAAGGCTCTTTCAGGGCT
C7  CGAAATCCTGGAAAGTAGTATGATATAGCTA
C8  AGGAGTGAGATTGGGTCTCGACGAGTGTAGTTGCTG
C9  TGAGGAATTAATTTTTAAGGGAAGACCTTGGCC
C10 CGCGACCTTFCATCGACATCGCTAAACTCGTCTTT
C11 GCCTAACCAACTACACCCCTTAAGAG
C12 TTCCAGTGCCGCAATCGAAGCTGTTTGTCCCCAGTCTTTTTGCATAT
D1  CAACGCCCCAAGGCCTCATCTAGTGAAGTGACAGTGTGACCTC
D2  GAATAACATGAAAATAGCAGCCTAAAGGATAGAAGATGAAAAATA
D3  CGCTAAGCGGTCTCGGTAAAATAGTGTGACAGCAG
D4  AGGGGAGTGGATTAGGGCTCGCCGTCACGACGTTGTACTTGCTG
D5  TGAGGAATTAATTTTTAAGGGAAGACCTTGGCC
D6  GCAAAATTATTTTTGAGAGATCTAGTCTAGTGAAGACGCAAC
D7  CAGTAGACAGGGAAAAAGCGCGGAAAAACTCAGAAAAAGACAGC
D8  ATTATACGCTGATAAACTCAAAGTACCTCAACATGTAAGCCCG
D9  TGGGCGCAGCTTTGAGAGTTG
D10 TTAGTTGACCTTAAAAAGATTAAGAGTTTTCGCCTGATTT
D11 GACTGATTGAATGGAGGTGGTTGAACTGCGCTACAAAAGATTTTCTT
D12 CACTAGTCTTTTTTTATCTGTTGCTCCCGTCCCTCATTACATAATCAAAT
E1  TTTGCCCATATCAGCGGCAACATTATATTAC
E2  GAACGTGTTTTTTGAGAGACAGCGAAAGACAG
E3  TAGCGTCCCAAGCTATGCTCTTTGTA
E4  AGTTTGAACTTCTTGATCTTCTTGTGTA
E5  GGCTGCGCGGTCCGTCCTTTGAGAGATCTAGACGAGCAG
E6  GAGCTTTCTAGAGCTCTCCATAATTGAGGAATCTCAGAAGAGAT
E7  AAAATCTAGAAACGAGACGGGGGTATGGATAAG
E8  ATTTACAGATTAGTTGATGTTATATTGCTA
E9  ATGGAAAGGCATGAGTGCCTAGCTTAAATGGTTGAAGCAATA
E10 TCCGCACAAGCTGTGGGATGTTGAAGGAATAAATC
E11 AAAATCATCCCTAGAATCTAGCTACTGTGAACGTAATTCTGTGGCCAA
E12 GCTTTTGAAGGAACAGACGTGTAGTTGCAGCAAGCGCAACAGGAGCCC
F1  GACCAGGAAGTTGGGAAGAAATTACGAAAGGTTTCTATTACATTT
F2  TGCAACACCAGACTTTCCACCTGGAGATGATAAACTGTTGTCGTGCC

Continued on next page
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>F3</td>
<td>AGAAGATGTAATCTGAAAAACATAGCGA</td>
</tr>
<tr>
<td>F4</td>
<td>CGGTCAAGAAATACAGTCACACCGTTGTTGAATTC</td>
</tr>
<tr>
<td>F5</td>
<td>GTTAAACATTTTTTACAAACGCTTCGTTG</td>
</tr>
<tr>
<td>F6</td>
<td>ATGCCTAGTTAGAAATACGCTATGAAAGCTTAA</td>
</tr>
<tr>
<td>F7</td>
<td>TCGTTACTGCCCAACCTTTATCCGCTGTTG</td>
</tr>
<tr>
<td>F8</td>
<td>TAGAGCCTTTTCATCGAGTTCTCCCTCAATCCCTGAGGTTTCTCA</td>
</tr>
<tr>
<td>F9</td>
<td>ATAAAGGGGTGGTTACATGGAACGAGTA</td>
</tr>
<tr>
<td>F10</td>
<td>GATGAACGGGTAATCAAAAGTATGGACCTTCCTGGCTC</td>
</tr>
<tr>
<td>F11</td>
<td>GATAATACATTAAATCCGGCCGAGTTGAAATATAATCAGG</td>
</tr>
<tr>
<td>F12</td>
<td>CTGAAATCCCTTGCGGGTATTACAAATGA</td>
</tr>
<tr>
<td>G1</td>
<td>CGACAAAAACAGTATATAAGGAAAAAGCCTCTAAACACAAACAT</td>
</tr>
<tr>
<td>G2</td>
<td>AGTAACAGAAACATGCATCTCTGATGCA</td>
</tr>
<tr>
<td>G3</td>
<td>GAGATGGTGATGTACGTACAGTAATAATT</td>
</tr>
<tr>
<td>G4</td>
<td>TTCAATTACGGGAACCTCTGTCAGCAGTCTTTAGATAGGA</td>
</tr>
<tr>
<td>G5</td>
<td>AGCTGCATTAAATGAGTCTAGACGCAACCACATGAC</td>
</tr>
<tr>
<td>G6</td>
<td>TCAATTTATCCGATCGGAAACCCAGGCTGAGCGCCCATGTTGCTGTAAG</td>
</tr>
<tr>
<td>G7</td>
<td>TGATGGGTGTAGGATTAGTATTTAGATGCTCTGTCACAAGTACCTTA</td>
</tr>
<tr>
<td>G8</td>
<td>AAAACACTTACGCAAAACCTCTCATTACCAATCAACTT</td>
</tr>
<tr>
<td>G9</td>
<td>CGCTCATGACAGCTAAATTTGCGGCTTGCAAGGAGTTAAAACGATGTCAA</td>
</tr>
<tr>
<td>G10</td>
<td>TTTAATTTGGAACCCGCTACTTTAAGG</td>
</tr>
<tr>
<td>G11</td>
<td>GTGGATATATCAAAAAGATAGGGGTGATTTAAGGATTGCGCCAGCCAG</td>
</tr>
<tr>
<td>G12</td>
<td>TGAGGCCAGGCGATATGTTTGTAGGCAATCAAAGGATCCTCA</td>
</tr>
<tr>
<td>H1</td>
<td>ACCGCTAAGGGAATCTCCTGACATAGAAGAAAGAAAGATAAG</td>
</tr>
<tr>
<td>H2</td>
<td>CAGAGGAATACAGCCAGCAGCAGTTACCCACCAAAACAGA</td>
</tr>
<tr>
<td>H3</td>
<td>ACCCTCAACAGAGATATAGGAAAA</td>
</tr>
<tr>
<td>H4</td>
<td>ATTCATTGAATCCCCTTCTGATTAAATTTAAGAAATCAAGAAAACAG</td>
</tr>
<tr>
<td>H5</td>
<td>TAAACAAAAAAATTGCAAAATAGCAAGTTAAACATACTATGCAGAAATCCA</td>
</tr>
<tr>
<td>H6</td>
<td>GGGGAGAGAGCTCGAGGCGCAAGCGTCATACATGCTGCTAGT</td>
</tr>
<tr>
<td>H7</td>
<td>CCTTTATTCAGACATAAAGCAGAAATTACGATAGATAATAT</td>
</tr>
<tr>
<td>H8</td>
<td>CGACCCTACCTCGAGGCGCATAGAAGCGAGACCAACAAAAGGCGCTTTTG</td>
</tr>
<tr>
<td>H9</td>
<td>CAAAGGTCAATCTCTACTAATAATATAGCTACAAGGCTCCTTGA</td>
</tr>
<tr>
<td>H10</td>
<td>CCAAATACAAACACCCGCTGAACATTA</td>
</tr>
<tr>
<td>H11</td>
<td>GAGGCATATTGAGACATATCATATAATACATTAAAACCAGTACACAATAGT</td>
</tr>
<tr>
<td>H12</td>
<td>ACCTTTTGAAGAGTTAAATCA</td>
</tr>
<tr>
<td>Well</td>
<td>Sequence</td>
</tr>
<tr>
<td>------</td>
<td>----------</td>
</tr>
<tr>
<td>A1</td>
<td>TCCCCGCCAAGACGGAGGGAAGGCCACCTTATGAGAGGG</td>
</tr>
<tr>
<td>A2</td>
<td>TCACAATGCTTTCCAGTCGGGAACCTGAAATTGTCAAGGCCAATAAGT</td>
</tr>
<tr>
<td>A3</td>
<td>ATGAACGCTGATAAAGTAATGTCACCCCTCAGCGG</td>
</tr>
<tr>
<td>A4</td>
<td>GGCAGAACGATTAGCAGCCCCGAATAGG</td>
</tr>
<tr>
<td>A5</td>
<td>CTAAAGTAGGGAAGGGCAACCCTTTTCCACTGTTTGGGAACAG</td>
</tr>
<tr>
<td>A6</td>
<td>AACGCAAGCGCTAAAAATTAACGACCAG</td>
</tr>
<tr>
<td>A7</td>
<td>CCGGAATAAGAATTATCACTCTTTGCTGAA</td>
</tr>
<tr>
<td>A8</td>
<td>CCTAAAATATAAATAAAAGACTTCAATATAATGCTGTACGGT</td>
</tr>
<tr>
<td>A9</td>
<td>AGAGGGGGTAATAGGTTTACCCCTAATGCACTAAGGACCAACT</td>
</tr>
<tr>
<td>A10</td>
<td>CTGACATATAAGTCCGAGAACCCTGAATTTCTAAACAGCTTAGTACC</td>
</tr>
<tr>
<td>A11</td>
<td>ACCTCAAAGGGGCGATGAAACCATATTCAATTAGAAA</td>
</tr>
<tr>
<td>A12</td>
<td>TTGATCTGCGACGATGGCAGGACCCTAAAAATCCT</td>
</tr>
<tr>
<td>B1</td>
<td>GGGTACCGGCGGTTCACCAGTGAGACGGGTCACGTTGCTT</td>
</tr>
<tr>
<td>B2</td>
<td>GAGCCGGATTGCGTTGCGCTCACTGCCCTCCACACCACCCTCTGGCTT</td>
</tr>
<tr>
<td>B3</td>
<td>TAAATCGTTTTGCGTTTGAGTCAGTTGAGCAACAGAGCCCTA</td>
</tr>
<tr>
<td>B4</td>
<td>GAATCATTGTGAAATAACCCGGATATTTGGACCCCCAGCGATATGTAATCT</td>
</tr>
<tr>
<td>B5</td>
<td>TTTGACATAGTAAGCGTCCAATTAATTC</td>
</tr>
<tr>
<td>B6</td>
<td>GTTACAATATAATTGATAATAACTCCTATTAGATATAATGAA</td>
</tr>
<tr>
<td>B7</td>
<td>CCACCAGTCAGATGTGTCACGTTAGGCA</td>
</tr>
<tr>
<td>B8</td>
<td>AGGGACATCTGTCCTCAGAGCGTAAC</td>
</tr>
<tr>
<td>B9</td>
<td>CAGATATAAGACGCTGAGATCACTAATGAAAGAA</td>
</tr>
<tr>
<td>B10</td>
<td>ATCAACGGGAGCATTACATTAATGACACCAAAAG</td>
</tr>
<tr>
<td>B11</td>
<td>TTATATATAAAAGCCATGAGAACCACCCAG</td>
</tr>
<tr>
<td>B12</td>
<td>GGTGGACGTGAGAAGAGAAATCTCTATATGCGCCATCAGGTAC</td>
</tr>
<tr>
<td>C1</td>
<td>CAATCAATATTCTATATATTTTACAGGCAAAGGCA</td>
</tr>
<tr>
<td>C2</td>
<td>CCAAAAGACAAAGGGAATCTTTTGGCCAG</td>
</tr>
<tr>
<td>C3</td>
<td>TAGAAAAACGACAGGCGAGCTAAACAGGACAGGAGTGAAATTGA</td>
</tr>
<tr>
<td>C4</td>
<td>CAAAAGACGCTGAGAATAATAGTCAAAAATAAAAAGGTTTAGGGTTT</td>
</tr>
<tr>
<td>C5</td>
<td>GGGCTTATTTGCGAGAGTAAAACCGCCTGTATTTTCTATCCCG</td>
</tr>
</tbody>
</table>

Continued on next page
<table>
<thead>
<tr>
<th>Column</th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>C6</td>
<td>GGTTGGGTAGCTTAATAACCTTGTCTTGTAGAAGAAATAA</td>
</tr>
<tr>
<td>C7</td>
<td>GTGGTTTCAGGCGATAGCTAGCGACAGATTGTACACGCCTCCC</td>
</tr>
<tr>
<td>C8</td>
<td>AAAACGAAGATACATGATAGCTCTTCTTTT</td>
</tr>
<tr>
<td>C9</td>
<td>AAAGTACGTAATACGTTGTACCCTTTTTGCAAGTAAGGTACAGGTACGGCTTA</td>
</tr>
<tr>
<td>C10</td>
<td>ATATATGACAAAAATAAATATTATATCATATAATC</td>
</tr>
<tr>
<td>C11</td>
<td>GGAACAATCAGAGGAGCCCTCTCAAGGATAGATAGATGTTAACCA</td>
</tr>
<tr>
<td>C12</td>
<td>CTCTGAATTACGTCGACAAATGGGAGAG</td>
</tr>
<tr>
<td>D1</td>
<td>ATTATTAAATACCAAGTAGATGCTCTTTTTGATAAAAACTCC</td>
</tr>
<tr>
<td>D2</td>
<td>GAATATAAGAATTTTGAACACTCTATGAGCAGACTGCAGATAGAGA</td>
</tr>
<tr>
<td>D3</td>
<td>GTGTAAGACAGATGCGAGCTGAAAAAGTCTGGAGGAATAC</td>
</tr>
<tr>
<td>D4</td>
<td>AAAAGATCTTGGGCAAGAATGCTGGAGAGCACTGTTTCTT</td>
</tr>
<tr>
<td>D5</td>
<td>TICAAACCAAGCTCATACGTTGACCCTAAGGAGAGCGTCTTAAAT</td>
</tr>
<tr>
<td>D6</td>
<td>ATTCACCAGAACGATAATTGTATAGTTGCGCCGAC</td>
</tr>
<tr>
<td>D7</td>
<td>GTGTCGAAATGCGACGATTCAAAAGACGAC</td>
</tr>
<tr>
<td>D8</td>
<td>CGAATCTCCGGCCTGAAAAGTACAGAGCCACACCCCTTTGTC</td>
</tr>
<tr>
<td>D9</td>
<td>TGCTCCATGTITACTAGGGGAACCTGATAACGGAAT</td>
</tr>
<tr>
<td>D10</td>
<td>GCTAACACTACATTAAGCATAAACCCGCTTATAAC</td>
</tr>
<tr>
<td>D11</td>
<td>AAACGACTCTTAATCCGTTGTGAATCATGATCGGCTGCCGCTT</td>
</tr>
<tr>
<td>D12</td>
<td>GTAATGGAATCTCTGCTATATAATTAAATACCT</td>
</tr>
<tr>
<td>E1</td>
<td>GCGAAAGTGGGAGAGCCCAATAATCAGAC</td>
</tr>
<tr>
<td>E2</td>
<td>TAAGCTATTTACATTAGCGTGACGAC</td>
</tr>
<tr>
<td>E3</td>
<td>TAAAGCACATCAACCGATTGAGCAAGGGCCGAAAGCTCAGCTACCAAA</td>
</tr>
<tr>
<td>E4</td>
<td>TGGCATTAAAGAATAAACCAGTTAGTATAAG</td>
</tr>
<tr>
<td>E5</td>
<td>CCCCGCTGGCCAGTGCAAAAGCAGAAAGCGCATTATTATT</td>
</tr>
<tr>
<td>E6</td>
<td>TTCCTGTTGAACGCAGAGAATCTGTACCTGACCCGAGTTT</td>
</tr>
<tr>
<td>E7</td>
<td>TTTGACGCTGAGGAGACGTTCCTGTTCATATAATCAGCT</td>
</tr>
<tr>
<td>E8</td>
<td>AGTAAATCGCATATAACATCCAAACGAGAAACCCTTATTAT</td>
</tr>
<tr>
<td>G1</td>
<td>GGGTCAGAGAACCGCTTATTACGCTTCTGGTAAAACGTGTCTGGC</td>
</tr>
<tr>
<td>G2</td>
<td>GATTTTTAGTTTTTATAAGAGCCTTTTATCCTCTGTTAAAAGCTGCTG</td>
</tr>
<tr>
<td>G3</td>
<td>AGGCAGATCCATCAACGATGACTCTACTGTTAAAAAGCTGCTGGC</td>
</tr>
<tr>
<td>G4</td>
<td>TTAATACCCGTAAAAATAAGTATAGCCCTACGTTTAAACGAAAGGTCTGGC</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>G5</td>
<td>AGAAACGCAATCAATAAAGGTGGAATTAGACCCAGCTCTCTGGTTAACGTGTCTGGGC</td>
</tr>
<tr>
<td>G6</td>
<td>TATTTTCGCTACAGTTAAGGACGTCTCTCTGGTTAACGTGTCTGGGC</td>
</tr>
<tr>
<td>G7</td>
<td>GATTTGGGAACCTACATCGCCAAGAGACTCTCTGGTTAACGTGTCTGGGC</td>
</tr>
<tr>
<td>G8</td>
<td>AATGTGAAACAACTCTTCCCTTAGAAACCTCTGGTTAACGTGTCTGGGC</td>
</tr>
<tr>
<td>H1</td>
<td>CCTCGATAAAAAATAAACCAGGCCACCAGTA</td>
</tr>
<tr>
<td>H2</td>
<td>GAAAGGAGTCCTTTATTTAAGACGGAATAAC</td>
</tr>
<tr>
<td>H3</td>
<td>GATTTCAATGGTTTTACGACTATTAGAGGCC</td>
</tr>
<tr>
<td>H4</td>
<td>GTGTAATGACTCTACTCTGCACCCCTACAGAACG</td>
</tr>
<tr>
<td>H5</td>
<td>TGGCAAGGCTTTGATAACACAAAAAAGTACCAGCCCACCAB</td>
</tr>
<tr>
<td>H6</td>
<td>CAAGAAATGAGTGAGGATTAAGACGCTGATAAACCTC</td>
</tr>
<tr>
<td>H7</td>
<td>CTGAATATCGCGTTTTAATCGGCGGATCGTTTATAATGG</td>
</tr>
<tr>
<td>H8</td>
<td>AGCCAGCGTCAATAAAAAACAATCATCCTGCACTGCAAA</td>
</tr>
<tr>
<td>H9</td>
<td>CTGAGAGAGTTGGGCGGTAAAAATATACAGATAG</td>
</tr>
<tr>
<td>H10</td>
<td>ATAGGGAATACCCCCGCCGGCAGCAATTTGTATCATCGC</td>
</tr>
<tr>
<td>H11</td>
<td>CTTTAATGCACTCATGCTTTAAACAGGTATTTTCA</td>
</tr>
<tr>
<td>H12</td>
<td>GGTGAAGGTAAAATACGCGGTATCCATCCTAATTTAC</td>
</tr>
</tbody>
</table>
The sequences used to bind the nanostructure to the plasma membrane are listed in Table 3 below.

Table 3: Sequences Used to Bind to the Plasma Membrane

<table>
<thead>
<tr>
<th>CellCro Cell Binding Sequences</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3’ Cholesterol-Conjugated Oligo</td>
<td>GATGAATGGTGAGAGG/CholTEG/</td>
</tr>
<tr>
<td>5’ Cholesterol-Conjugated Oligo</td>
<td>/Chol-TEG/GATGAATGGTGAGAGG</td>
</tr>
<tr>
<td>Prev</td>
<td>CTCTGGTTAACGTGTCTGGGC</td>
</tr>
<tr>
<td>Chol</td>
<td>CCTCTCACCCACCATTTCATC</td>
</tr>
<tr>
<td>60 nucleotide bridge</td>
<td>CCTCTCACCCACCATTTCATCCTGCCCCAGACACGTTAACCA</td>
</tr>
<tr>
<td>40 nucleotide bridge</td>
<td>CCTCTCACCCACCATTTCATCCTGCCCCAGACACGTTAAC</td>
</tr>
<tr>
<td>Bridge Fortifier</td>
<td>AAA AAA AAA AAA AAA AAA</td>
</tr>
</tbody>
</table>

The sequences used to fluorescently label the nanostructure or membrane are listed in Table 4.

Table 4: Sequences Used to Fluorescently Label the Nanostructure or Membrane

<table>
<thead>
<tr>
<th>Fluorescent Sequences</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Complement to 3’ Cholesterol</td>
<td>/Cy5/ CCTCTCACCCACCATTTCATC</td>
</tr>
<tr>
<td>Complement to Chol</td>
<td>/Cy5/ GATGAATGGTGAGAGG</td>
</tr>
<tr>
<td>Complement to Prev</td>
<td>/Alexa647/ GCCCAGACACGTTAACCA</td>
</tr>
<tr>
<td>Complement to Prev</td>
<td>/Cy3/ GCCCAGACACGTTAACCA</td>
</tr>
<tr>
<td>Complement to F2</td>
<td>/Alexa488/ CTTGGTCTATTCGCTGATTG</td>
</tr>
<tr>
<td>Complement to F2</td>
<td>/Cy3/ CTTGGTCTATTCGCTGATTG</td>
</tr>
</tbody>
</table>
The sequences used functionalize the LPP with biotin are listed in Table 5.

Table 5: LPP 5T, 15T, and 30T Sequences

<table>
<thead>
<tr>
<th>LPP 5T - 2B</th>
<th>LPP 15T - 2B</th>
<th>LPP 30T - 2B</th>
</tr>
</thead>
<tbody>
<tr>
<td>/Biotin/TTTTTGTTGAGGTAAAAATACGGGTATCCATCCTAATTTAC</td>
<td>/Biotin/TTTTTTTTTTTTTTGTTGAGGTAAAAATACGGGTATCCATCCTAATTTAC</td>
<td>/Biotin/TTTTTTTTTTTTTTTTTTTTTTTTTTTTGTTGAGGTAAAAATACGGGTATCCATCCTAATTTAC</td>
</tr>
</tbody>
</table>

LPP Biotin Overhangs
Appendix B: MATLAB Code To Analyze Nanostructure Binding
This is the MATLAB code used to analyze and export the fluorescent intensity on the periphery of the cell.

```matlab
%% Cell Periphery - Image Data Processing
% Written by MYM with portions from CEC and RAP
% Last updated by MYM 12/28/2015

%% Initialize
clear all
close all
clc

%% Instructions
dateanalyzed = date;
n = 1; % count (# of cells analyzed minus 1)
image = 1;
progress = 0;
Instructions = questdlg('Select an .nd2 file','Instructions','Got it!','Test');
% datedatacollected =

while progress < 1
  if image == 1;
    %% SELECT .ND2 FILE FOR PROCESSING
    % IMPORT .ND2 FILE
    data = bfopen();
    [XY,T,C] = nd2param(data);
    %Query image height and width
    omeMeta = data{1, 4};
    width = omeMeta.getPixelsSizeX(0).getValue(); % image width, pixels
    height = omeMeta.getPixelsSizeY(0).getValue(); % image height, pixels
    voxelSize = omeMeta.getPixelsPhysicalSizeX(0).getValue(); % in µm

    %% Select Image Channels: DIC and Fluorescent
    clc
close all

    Instructions = questdlg('You will now be presented with a figure and asked which fields you are interested. Follow the instructions in the Command Window.','Instructions','Got it!','Test');

    % Display figures for all channels
    figure(1)
    for i = 1:C
      xy = 1;
      t = 1;
      current_XY = data{1, 1};
      RAW image = double(current_XY{((C*(t-1))+i), 1});
      subplot(1,C,i)
```
imshow(RAW_image,[]);
title(sprintf('%d',i));
end
display('Type DIC Channel number and press Enter')
DIC_Channel = input('');
display('Type Fluorescent Channel number and press Enter')
Fluoro_Channel = input('');
close all
clc

%% Background Substraction
for xy = 1:XY
    current_XY = data{xy, 1};
    %current_XY contains images for all illumination channels and all
    %the timepoints **if available** for that XY position
    for c = [Fluoro_Channel]
        for t = 1:T
            Raw_Image = double(current_XY{((C*(t-1))+c), 1});
            DIC_Image = double(current_XY{((C*(t-1))+DIC_Channel), 1});
            %{C*(t-1))+c is how the images are indexed in current_XY
            %
            %Subtract a fitted background from the Fluorescent Image
            fit_length = 25;%number of points used in background
            subtraction
            figure(1)
            contour(DIC_Image,'Fill','on')
            set(gcf,'Position',[25 25 400 400])
            xlabel(sprintf('select %d points for background',fit_length),
            'FontSize',20,'Fontweight','bold')

            x_fit = zeros(fit_length,1);
            y_fit = zeros(fit_length,1);
            z_fit = zeros(fit_length,1);

            % pick points from image
            for i=1:fit_length
                [x_pt, y_pt] = ginput(1);
                x_fit(i) = round(x_pt);
                y_fit(i) = round(y_pt);
                z_fit(i) = Raw_Image(y_fit(i),x_fit(i));
                hold on
            end
            plot(x_fit(i),y_fit(i),'kx','Linewidth',2,'MarkerSize',10)
        end
    end
end
poly3 = polyfitn([x_fit y_fit],z_fit,3);
[r, s] = size(Raw_Image);
\[
[x \_grid, y \_grid] = \text{meshgrid}(1:s,1:r);
\]
\[
x \_bg1 = \text{reshape}(x \_grid,r*s,1);
y \_bg1 = \text{reshape}(y \_grid,r*s,1);
z \_bg1 = \text{polyvaln}(\text{poly3},[x \_bg1 y \_bg1]);
\]
\[
x \_bg = \text{reshape}(x \_bg1,r,s);
y \_bg = \text{reshape}(y \_bg1,r,s);
z \_bg = \text{reshape}(z \_bg1,r,s);
\]

```matlab
figure(3)
surf(Raw \_Image,'\text{EdgeColor}', 'none', '\text{FaceLighting}','gouraud', '\text{EdgeLightin g}', 'gouraud')
view([0 1 0.5])
set(gcf,'\text{Position}',[450 25 400 400])
x \_lim = get(gca,'Xlim');
y \_lim = get(gca,'Ylim');
z \_lim = get(gca,'Zlim');
xlabel('Pre background subtration', 'FontSize',20,'Fontweight','bold')
set(gca,'\text{Clim}',[min(min(Raw \_Image)) max(max(Raw \_Image))])
```

```matlab
figure(4)
surf(z \_bg,'\text{EdgeColor}', 'none', '\text{FaceLighting}', 'gouraud', '\text{EdgeLighting}', 'gouraud')
view([0 1 0.5])
set(gcf,'\text{Position}',[875 25 400 400])
set(gca,'Xlim',x \_lim,'Ylim',y \_lim,'Zlim',z \_lim)
xlabel('Fitted background', 'FontSize',20,'Fontweight','bold')
set(gca,'\text{Clim}',[min(min(Raw \_Image)) max(max(Raw \_Image))])
```

```matlab
BG \_Subt \_Image = Raw \_Image - z \_bg;
```

```matlab
figure(5)
surf(BG \_Subt \_Image,'\text{EdgeColor}', 'none', '\text{FaceLighting}', 'gouraud', '\text{EdgeLighting}', 'gouraud')
view([0 1 0.5])
set(gcf,'\text{Position}',[1300 25 400 400])
xlabel('Post background subtration', 'FontSize',20,'Fontweight','bold')
```

```matlab
figure(6)
imshow(BG \_Subt \_Image,[])
set(gcf,'\text{Position}',[25 425 400 400])
xlabel('Post background subtration', 'FontSize',20,'Fontweight','bold')
```
figure(7)
bar3(Raw_Image)
colorbar

clc
close all

% T
end
% channel
end
% XY Location

%% View Pre and Post Background Subtraction
figure(8)
subplot(1,2,1)
imshow(Raw_Image,[])
colormap hot
title('Before Background Subtraction','FontSize',12)

subplot(1,2,2)
imshow(BG_Subt_Image,[])
colormap hot
title('After Background Subtraction','FontSize',12)

set(gcf,'Color',[1 1 1])

%% Analyze cell
Instructions=questdlg('You will now be presented with your background-subtracted image. Select points around the cell you wish to analyze. Hit enter when you are done with one cell.','Instructions','Got it!','Test');
image = image+1;
end

if image == 2
% Export .tif file
close all
figure(9)
set(gcf,'Position',[25 425 400 400])
a = imshow(BG_Subt_Image,[]);
title('Set points around cell','FontSize',15,'Fontweight','bold')
[r c] = ginput;
hold on
plot(r,c,'r')
roipoly(a,r,c);
A(:,1) = r;
A(:,2) = c;

% Fix length such that 1000 points exist around each cell
A;
f=1;
pointsbetween = ceil(1000/length(A));
for i=1:length(A)
    if i<length(A)
        Axlarge(f:pointsbetween-1+f)=linspace(A(i,1),A(i+1,1),pointsbetween);
        Aylarge(f:pointsbetween-1+f)=linspace(A(i,2),A(i+1,2),pointsbetween);
        f=f+pointsbetween;
    else
        Axlarge(f:pointsbetween-1+f)=linspace(A(i,1),A(1,1),pointsbetween);
        Aylarge(f:pointsbetween-1+f)=linspace(A(i,2),A(1,2),pointsbetween);
        f=f+100;
    end
end
Axlargev=Axlarge';
Aylargev=Aylarge';
clear i
for i=1:length(Axlarge)
    a=impixel(BG_Subt_Image,Axlargev(i),Aylargev(i))
    % ab=impixel(BG_Subt_Image,Axlargev(i)-200,Aylargev(i))
    pixelintensity(i)=a(1)
    % backgroundintensity(i)=ab(1)
end
pixelintensitySAVE(:,n) = pixelintensity(1:1000)'
% backgroundintensitySAVE(n) =backgroundintensity(1:1000)

image = image + 1;
end
if image == 3;
    another = questdlg('What to do next?','Next step','Same image, different cell','Different image','Done','Test')
    if strcmp(another,'Same image, different cell')
        image=2;
        n = n+1;
        clear r c A
    elseif strcmp(another,'Different image')
        image=1;
        n = n+1;
        clear r c A
    else
        progress=1;
    end
end
end
end
%% Export to Excel
% Export .csv file
Save=questdlg('Would you like to export this file?','Export','Yes','No','Test');
if strcmp(Save,'Yes')
    filename1 = inputdlg({''Enter file name for Excel data export:'},'Name',1,{'.'csv'});
    filename = filename1{1};
    T = table(pixelintensitySAVE);
    writetable(T,filename,'WriteRowNames',true);
end

%% Plot

% Average Value
clear N n
s = size(pixelintensitySAVE);
n = s(2);
for N = 1:n
    av(N) = mean(pixelintensitySAVE(:,N))
end
figure(10)
plot(1:n,av,.--',MarkerSize',25)
title('Average Pixel Intensity','FontSize',20,'Fontweight','bold')
xlabel('Cell #','FontSize',15)
ylabel('Average Intensity Around Cell','FontSize',15)
set(gca,'Color',[1 1 1])
axis([0 n+1 min(av)-.2*max(av) max(av)+.2*max(av)])

% Intensity Over Cell
figure(11)
hold on
for N = 1:n
    plot(1:1000,pixelintensitySAVE(:,N),'color',rand(1,3))
title('Pixel Intensity Around Each Cell','FontSize',20,'Fontweight','bold')
xlabel('Pixels Around Cell','FontSize',15)
ylabel('Intensity','FontSize',15)
set(gca,'Color',[1 1 1])
end
hold off
This is the “polyvaln” function used above.

```matlab
function ypred = polyvaln(polymodel,indepvar)
% polyvaln: evaluates a polynomial model as a function of its variables
% usage: ypred = polyvaln(polymodel,indepvar)
% % arguments: (input)
% % indepvar - (n x p) array of independent variables as columns
%  n is the number of data points to evaluate
%  p is the dimension of the independent variable space
%  IF n == 1, then I will assume there is only a
%  single independent variable.
% % polymodel - A structure containing a regression model from polyfitn
%  polymodel.ModelTerms = list of terms in the model
%  polymodel.Coefficients = regression coefficients
%  Note: A polymodel can be evaluated for any set of
%  values with the function polyvaln. However, if you
%  wish to manipulate the result symbolically using my
%  own sympoly tools, this structure should be converted
%  to a sympoly using the function polyn2sympoly.
% % Arguments: (output)
% % ypred - n x 1 vector of predictions through the model.
% % %
% % See also: polyfitn, polyfit, polyval, polyn2sympoly, sympoly
% % % Author: John D'Errico
% % Release: 1.0
% % Release date: 2/19/06
% %
% get the size of indepvar
[n,p] = size(indepvar);
if (n == 1) && (size(polymodel.ModelTerms,2)==1)
  indepvar = indepvar';
  [n,p] = size(indepvar);
elseif (size(polymodel.ModelTerms,2)~=p)
  error 'Size of indepvar array and this model are inconsistent.'
end

% Evaluate the model
nt = size(polymodel.ModelTerms,1);
ypred = zeros(n,1);
for i = 1:nt
  t = ones(n,1);
  for j = 1:p
    t = t.*indepvar(:,j).^polymodel.ModelTerms(i,j);
  end
  ypred = ypred + t*polymodel.Coefficients(i);
end
```
This is the “polymodel” function used above.

```matlab
function polymodel = polyfitn(indepvar,depvar,modelterms)
% polyfitn: fits a general polynomial regression model in n dimensions
% usage: polymodel = polyfitn(indepvar,depvar,modelterms)
% Polyfitn fits a polynomial regression model of one or more
% independent variables, of the general form:
%   z = f(x,y,...) + error
% arguments: (input)
% indepvar - (n x p) array of independent variables as columns
%   n is the number of data points
%   p is the dimension of the independent variable space
%   IF n == 1, then I will assume there is only a
%   single independent variable.
% depvar - (n x 1 or 1 x n) vector - dependent variable
% length(depvar) must be n.
% Only 1 dependent variable is allowed, since I also
% return statistics on the model.
% modelterms - defines the terms used in the model itself
%   IF modelterms is a scalar integer, then it designates
%   the overall order of the model. All possible terms
%   up to that order will be employed. Thus, if order
%   is 2 and p == 2 (i.e., there are two variables) then
%   the terms selected will be:
%       {constant, x, x^2, y, x*y, y^2}
%   Beware the consequences of high order polynomial
%   models.
%   IF modelterms is a (k x p) numeric array, then each
%   row of this array designates the exponents of one
%   term in the model. Thus to designate a model with
%   the above list of terms, we would define modelterms as
%   modelterms = [0 0;1 0;2 0;0 1;1 1;0 2]
%   If modelterms is a character string, then it will be
%   parsed as a list of terms in the regression model.
%   The terms will be assume to be separated by a comma
% or by blanks. The variable names used must be legal
% matlab variable names. Exponents in the model may
% may be any real number, positive or negative.
```
For example, 'constant, x, y, x*y, x^2, x*y*y'
will be parsed as a model specification as if you
had supplied:
modelterms = [0 0;1 0;0 1;1 2 0;1 2]

The word 'constant' is a keyword, and will denote a
countn terms in the model. Variable names will be
sorted in alphabetical order as defined by sort.
This order will assign them to columns of the
independent array. Note that 'xy' will be parsed as
a single variable name, not as the product of x and y.

If modelterms is a cell array, then it will be taken
to be a list of character terms. Similarly,

{'constant', 'x', 'y', 'x*y', 'x^2', 'x*y^-1'}

will be parsed as a model specification as if you
had supplied:

modelterms = [0 0;1 0;0 1;1 2 0;1 -1]

Arguments: (output)
polymodel - A structure containing the regression model
   polymodel.ModelTerms = list of terms in the model
   polymodel.Coefficients = regression coefficients
   polymodel.ParameterVar = variances of model coefficients
   polymodel.ParameterStd = standard deviation of model
   coefficients
   polymodel.R2 = R^2 for the regression model
   polymodel.AdjustedR2 = Adjusted R^2 for the regression model
   polymodel.RMSE = Root mean squared error
   polymodel.VarNames = Cell array of variable names
   as parsed from a char based model specification.

Note 1: Because the terms in a general polynomial
model can be arbitrarily chosen by the user, I must
package the terms and coefficients together into a
structure. This also forces use of a special evaluation
tool: polyvaln.

Note 2: A polymodel can be evaluated for any set
of values with the function polyvaln. However, if
you wish to manipulate the result symbolically using
my own sympoly tools, this structure can be converted
to a sympoly using the function polyn2sympoly. There
is also a polyn2sym tool, for those who prefer the
symbolic TB.

Note 3: When no constant term is included in the model,
the traditional R^2 can be negative. This case is
identified, and then a more appropriate computation
for R^2 is then used.

77
Note 4: Adjusted R^2 accounts for changing degrees of freedom in the model. It CAN be negative, and will always be less than the traditional R^2 values.

% Find my syopoly toolbox here:
% http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=9577&objectType=FILE
% See also: polyvain, polyfit, polyval, polyn2sympoly, sympoly
% Author: John D'Errico
% Release: 2.0
% Release date: 2/19/06

if nargin<1
    help polyfitn
    return
end

% get sizes, test for consistency
[n,p] = size(indepvar);
if n == 1
    indepvar = indepvar';
    [n,p] = size(indepvar);
end
[m,q] = size(depvar);
if m == 1
    depvar = depvar';
    [m,q] = size(depvar);
end
% only 1 dependent variable allowed at a time
if q~=1
    error 'Only 1 dependent variable allowed at a time.'
end
if n~=m
    error 'indepvar and depvar are of inconsistent sizes.'
end

% check for and remove nans in data
nandata = isnan(depvar) | any(isnan(indepvar),2);
if any(nandata)
    depvar(nandata,:) = [];
    indepvar(nandata,:) = [];
    n = size(indepvar,1);
end

% Automatically scale the independent variables to unit variance
stdind = sqrt(diag(cov(indepvar)));
if any(stdind==0)
    warning 'Constant terms in the model must be entered using modelterms'
end

78
\[
\text{stdind} = \begin{cases}
1 & \text{if } \text{stdind} == 0 \\
\text{end}
\end{cases}
\]

% scaled variables
\[
\text{indepvar}_s = \text{indepvar} \times \text{diag}(1./\text{stdind});
\]

% do we need to parse a supplied model?
\[
\text{if} \quad \text{iscell}(\text{modelterms}) \quad \text{||} \quad \text{ischar}(\text{modelterms})
\]
\[
[\text{modelterms}, \text{varlist}] = \text{parsemodel(}\text{modelterms}, \text{p});
\]
\[
\text{if} \quad \text{size}(\text{modelterms}, 2) < \text{p}
\]
\[
\text{modelterms} = [\text{modelterms}, \text{zeros(size(}\text{modelterms}, 1), \text{p} - \text{size(}\text{modelterms}, 2))];
\]
\[
\text{end}
\]
\[
\text{elseif} \quad \text{length(}\text{modelterms}\text{)} == 1
\]
\[
\text{do we need to generate a set of modelterms?}
\]
\[
[\text{modelterms}, \text{varlist}] = \text{buildcompletemodel(}\text{modelterms}, \text{p});
\]
\[
\text{elseif} \quad \text{size(}\text{modelterms}, 2) \neq \text{p}
\]
\[
\text{error 'ModelTerms must be a scalar or have the same # of columns as indepvar'}
\]
\[
\text{else}
\]
\[
\text{varlist} = \text{repmat(}''\text{,}1, \text{p});
\]
\[
\text{end}
\]
\[
\text{nt} = \text{size(}\text{modelterms}, 1);
\]

% check for replicate terms
\[
\text{if} \quad \text{nt}>1
\]
\[
\text{mtu} = \text{unique(}\text{modelterms,} \text{'rows'}\text{)};
\]
\[
\text{if} \quad \text{size(}\text{mtu}, 1)<\text{nt}
\]
\[
\text{warning 'Replicate terms identified in the model.'}
\]
\[
\text{end}
\]
\[
\text{end}
\]

% build the design matrix
\[
\text{M} = \text{ones(}n, \text{nt});
\]
\[
\text{scalefact} = \text{ones(}1, \text{nt});
\]
\[
\text{for} \quad i = 1:\text{nt}
\]
\[
\text{for} \quad j = 1:\text{p}
\]
\[
\text{M}(:, i) = \text{M}(:, i) \times \text{indepvar}_s(:, j) \times \text{modelterms}(i, j);
\]
\[
\text{scalefact}(i) = \text{scalefact}(i)/(\text{stdind}(j)^{\text{modelterms}(i, j)});
\]
\[
\text{end}
\]
\[
\text{end}
\]

% estimate the model using QR. do it this way to provide a % covariance matrix when all done. Use a pivoted QR for % maximum stability.
\[
[\text{Q, R, E}] = \text{qr(}\text{M}, 0);\]

\[
\text{polymodel.ModelTerms} = \text{modelterms};
\]
\[
\text{polymodel.Coefficients(E)} = \text{R}\text{'(}\text{depvar});
\]
\[
\text{yhat} = \text{M}\times\text{polymodel.Coefficients}(:)\]

% recover the scaling
\[
\text{polymodel.Coefficients} = \text{polymodel.Coefficients} \times \text{scalefact};
\]
% variance of the regression parameters
s = norm(depvar - yhat);
if n > nt
    Rinv = R\eye(nt);
    Var(E) = s^2*sum(Rinv.^2,2)/(n-nt);
polymodel.ParameterVar = Var.*(scalefact.^2);
polymodel.ParameterStd = sqrt(polymodel.ParameterVar);
else
    % we cannot form variance or standard error estimates
    % unless there are at least as many data points as
    % parameters to estimate.
polymodel.ParameterVar = inf(1,nt);
polymodel.ParameterStd = inf(1,nt);
end

% degrees of freedom
polymodel.DoF = n - nt;

% coefficient/sd ratio for a p-value
t = polymodel.Coefficients./polymodel.ParameterStd;

% twice the upper tail probability from the t distribution,
% as a transformation from an incomplete beta. This provides
% a two-sided test for the corresponding coefficient.
% I could have used tcdf, if I wanted to presume the
% stats toolbox was present. Of course, then regstats is
% an option. In that case, the comparable result would be
% found in:    STATS.tstat.pval
polymodel.p = betainc(polymodel.DoF./(t.^2 + polymodel.DoF),polymodel.DoF/2,1/2);

% R^2
% is there a constant term in the model? If not, then
% we cannot use the standard R^2 computation, as it
% frequently yields negative values for R^2.
if any((M(1,:) ~= 0) & all(diff(M,1,1) == 0,1))
    % we have a constant term in the model, so the
    % traditional R^2 form is acceptable.
polymodel.R2 = max(0,1 - (s/norm(depvar-mean(depvar)))^2);
    % compute adjusted R^2, taking into account the number of
    % degrees of freedom
    polymodel.AdjustedR2 = 1 - (1 - polymodel.R2).*((n - 1)/(n - nt));
else
    % no constant term was found in the model
    polymodel.R2 = max(0,1 - (s/norm(depvar))^2);
    % compute adjusted R^2, taking into account the number of
    % degrees of freedom
    polymodel.AdjustedR2 = 1 - (1 - polymodel.R2).*((n./n - nt));
end

% RMSE
polymodel.RMSE = sqrt(mean((depvar - yhat).^2));

% if a character 'model' was supplied, return the list
% of variables as parsed out
polymodel.VarNames = varlist;

% ==============================================================
% =============== begin subfunctions ===============
% ==============================================================
function [modelterms,varlist] = buildcompletemodel(order,p)

% arguments: (input)
% order - scalar integer, defines the total (maximum) order
% p - scalar integer - defines the dimension of the
%     independent variable space

% arguments: (output)
% modelterms - exponent array for the model
% varlist - cell array of character variable names

% build the exponent array recursively
if p == 0
  % terminal case
  modelterms = [];
elseif (order == 0)
  % terminal case
  modelterms = zeros(1,p);
elseif (p==1)
  % terminal case
  modelterms = (order:-1:0)';
else
  % general recursive case
  modelterms = zeros(0,p);
  for k = order:-1:0
    t = buildcompletemodel(order-k,p-1);
    nt = size(t,1);
    modelterms = [modelterms;[repmat(k,nt,1),t]];
  end
end

% create a list of variable names for the variables on the fly
varlist = cell(1,p);
for i = 1:p
  varlist{i} = ['X',num2str(i)];
end

% ==============================================================
function [modelterms,varlist] = parsemodel(model,p);

% 81
% arguments: (input)
% model - character string or cell array of strings
% p - number of independent variables in the model
% arguments: (output)
% modelterms - exponent array for the model

modelterms = zeros(0,p);
if ischar(model)
    model = deblank(model);
end

varlist = {};
while ~isempty(model)
    if iscellstr(model)
        term = model{1};
        model(1) = [];
    else
        [term,model] = strtok(model,
            ' ,');
    end

    % We've stripped off a model term. Now parse it.

    % Is it the reserved keyword 'constant'? 
    if strcmpi(term,'constant')
        modelterms(end+1,:) = 0;
    else
        % pick this term apart 
        expon = zeros(1,p);
        while ~isempty(term)
            vn = strtok(term,'*/^. ,');
            k = find(strncmp(vn,varlist,length(vn)));
            if isempty(k)
                % its a variable name we have not yet seen

                % is it a legal name? 
                nv = length(varlist);
                if ismember(vn(1),'1234567890 ')
                    error(["Variable is not a valid name: ",vn," "])
                elseif nv>p
                    error 'More variables a
term = term((i+length(vn)):end);

% is there an exponent?
eflag = false;
if strncmp('^',term,1)
    term(1) = [];
    eflag = true;
elseif strncmp('.',term,2)
    term(1:2) = [];
    eflag = true;
end

% If there was one, get it
ev = 1;
if eflag
    ev = sscanf(term,'%f');
    if isempty(ev)
        error 'Problem with an exponent in parsing the model'
    end
end
expon(k) = expon(k) + ev;

% next monomial subterm?
k1 = strfind(term,'*');
if isempty(k1)
    term = ''; 
else
    term(k1(1)) = ' '; 
end

modelterms(end+1,:) = expon;

end
end

% Once we have compiled the list of variables and 
exponents, we need to sort them in alphabetical order
[varlist,tags] = sort(varlist);
modelterms = modelterms(:,tags);
This is the “nd2param” function used above.

% clear all, close all, clc (troubleshooting)
function [XY,T,C] = nd2param(data)
%NanoEngineering and Biodesign Lab
%Randy Patton 02-12-2014
% Function reads in information contained in the image data file
% obtained with the bfopen function and determines imaging dataset parameters
% Number of XY positions, XY
% Number of timepoints, T
% Number of Illumination sources used, C (**Colors)
% Select and initialize a file (troubleshooting)
% filename = uigetfile('.nd2','Select File');
% data = bfopen(filename);

XY = size(data, 1); % Number of XY locations

% Determine number of Timepoints and Colors
tempseries = data{1,1}; % First XY location - contains all T,C for this XY
infostr = tempseries{1,2}; % a string of the form:
% filename;filename (series n); plane x/ntotal; C=y/Ctot; T=1/Ttot
%n = XY

% Determine overall length of string
strlen = length(infostr); % Total number of images
num_Im = size(tempseries,1);

% Look for color/time info within string
is_T = strfind(infostr,'T=1/');
is_C = strfind(infostr,'C=1/');
if ~isempty(is_T)
    T = str2double(infostr(is_T+4):strlen));
else
    T = 1;
end

if ~isempty(is_C)
    C = str2double(infostr(is_C+4)); % Assumes no more than 9 channels...
else
    C = 1;
end
This is the “bfopen” function used above.

```matlab
function [result] = bfopen(id, varargin)
% Open microscopy images using Bio-Formats.
% SYNOPSIS r = bfopen(id)
% r = bfopen(id, x, y, w, h)

% Input
% r - the reader object (e.g. the output bfGetReader)
% x - (Optional) A scalar giving the x-origin of the tile.
% Default: 1
% y - (Optional) A scalar giving the y-origin of the tile.
% Default: 1
% w - (Optional) A scalar giving the width of the tile.
% Set to the width of the plane by default.
% h - (Optional) A scalar giving the height of the tile.
% Set to the height of the plane by default.

% Output
% result - a cell array of cell arrays of (matrix, label) pairs,
% with each matrix representing a single image plane, and each inner
% list of matrices representing an image series.

% Portions of this code were adapted from:
% http://www.mathworks.com/support/solutions/en/data/1-2WPAYR/
% This method is ~1.5x-2.5x slower than Bio-Formats's command line
% showinf tool (MATLAB 7.0.4.365 R14 SP2 vs. java 1.6.0_20),
% due to overhead from copying arrays.
% Thanks to all who offered suggestions and improvements:
% * Ville Rantanen
% * Brett Shoelson
% * Martin Offterdinger
% * Tony Collins
% * Cris Luengo
% * Arnon Lieber
% * Jimmy Fong

% NB: Internet Explorer sometimes erroneously renames the Bio-Formats
% library
to loci_tools.zip. If this happens, rename it back to
loci_tools.jar.

% For many examples of how to use the bfopen function, please see:
% http://trac.openmicroscopy.org.uk/ome/wiki/BioFormats-Matlab
```

86
OME Bio-Formats package for reading and converting biological file formats.

Copyright (C) 2007 - 2013 Open Microscopy Environment:
- Board of Regents of the University of Wisconsin-Madison
- Glencoe Software, Inc.
- University of Dundee

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as
published by the Free Software Foundation, either version 2 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along
with this program; if not, write to the Free Software Foundation, Inc.,
51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.

-- Configuration - customize this section to your liking --

Toggle the autoloadBioFormats flag to control automatic loading
of the Bio-Formats library using the javaaddpath command.

For static loading, you can add the library to MATLAB's class path:
1. Type "edit classpath.txt" at the MATLAB prompt.
2. Go to the end of the file, and add the path to your JAR file
   (e.g., C:/Program Files/MATLAB/work/loci_tools.jar).
3. Save the file and restart MATLAB.

There are advantages to using the static approach over javaaddpath:
1. If you use bfopen within a loop, it saves on overhead
   to avoid calling the javaaddpath command repeatedly.
2. Calling 'javaaddpath' may erase certain global parameters.

autoloadBioFormats = 1;

Toggle the stitchFiles flag to control grouping of similarly
named files into a single dataset based on file numbering.
stitchFiles = 0;

To work with compressed Evotec Flex, fill in your LuraWave license
code.
lurawaveLicense = 'xxxxxx-xxxxxxx';

-- Main function - no need to edit anything past this point --

87
% load the Bio-Formats library into the MATLAB environment
status = bfCheckJavaPath(autoloadBioFormats);
assert(status, ['Missing Bio-Formats library. Either add loci_tools.jar '
...'
to the static Java path or add it to the Matlab path.']);

% Prompt for a file if not input
if nargin == 0 || exist(id, 'file') == 0
    [file, path] = uigetfile(bfGetFileExtensions, 'Choose a file to open');
    id = [path file];
    if isequal(path, 0) || isequal(file, 0), return; end
end

% initialize logging
loci.common.DebugTools.enableLogging('INFO');

% Get the channel filler
r = bfGetReader(id, stitchFiles);

% Test plane size
if nargin >=4
    planeSize = loci.formats.FormatTools.getPlaneSize(r, varargin{3}, 
varargin{4});
else
    planeSize = loci.formats.FormatTools.getPlaneSize(r);
end

if planeSize/(1024)^3 >= 2,
    error(['Image plane too large. Only 2GB of data can be extracted '
...'
at one time. You can workaround the problem by opening ...
...'
the plane in tiles.']);
end

numSeries = r.getSeriesCount();
result = cell(numSeries, 2);
for s = 1:numSeries
    fprintf('Reading series #%d', s);
    r.setSeries(s - 1);
    pixelType = r.getPixelType();
    bpp = loci.formats.FormatTools.getBytesPerPixel(pixelType);
    bppMax = power(2, bpp * 8);
    numImages = r.getImageCount();
    imageList = cell(numImages, 2);
    colorMaps = cell(numImages);
    for i = 1:numImages
        if mod(i, 72) == 1
            fprintf('
    ');
        end
    fprintf('.');
    arr = bfGetPlane(r, i, varargin{:});

88
% retrieve color map data
if bpp == 1
    colorMaps{s, i} = r.get8BitLookupTable();
else
    colorMaps{s, i} = r.get16BitLookupTable();
end

warning off
if ~isempty(colorMaps{s, i})
    newMap = single(colorMaps{s, i});
    newMap(newMap < 0) = newMap(newMap < 0) + bppMax;
    colorMaps{s, i} = newMap / (bppMax - 1);
end
warning on

% build an informative title for our figure
label = id;
if numSeries > 1
    seriesName = char(r.getMetadataStore().getImageName(s - 1));
    if ~isempty(seriesName)
        label = [label, '; ', seriesName];
    else
        qs = int2str(s);
        label = [label, '; series ', qs, ' /', int2str(numSeries)];
    end
end
if numImages > 1
    qi = int2str(i);
    label = [label, '; plane ', qi, ' /', int2str(numImages)];
    if r.isOrderCertain()
        lz = 'Z';
        lc = 'C';
        lt = 'T';
    else
        lz = 'Z?';
        lc = 'C?';
        lt = 'T?';
    end
    zct = r.getZCTCoords(i - 1);
    sizeZ = r.getSizeZ();
    if sizeZ > 1
        qz = int2str(zct(1) + 1);
        label = [label, '; ', lz, '=' qz, ' /', int2str(sizeZ)];
    end
    sizeC = r.getSizeC();
    if sizeC > 1
        qc = int2str(zct(2) + 1);
        label = [label, '; ', lc, '=' qz, ' /', int2str(sizeC)];
    end
end
int2str(sizeC)];
end
sizeT = r.getSizeT();
if sizeT > 1
    qt = int2str(zct(3) + 1);
    label = [label, '; ', lt, '=' ', qt, ' / ',
int2str(sizeT)];
end
end

% save image plane and label into the list
imageList{i, 1} = arr;
imageList{i, 2} = label;
end

% save images and metadata into our master series list
result{s, 1} = imageList;

% extract metadata table for this series
result{s, 2} = r.getSeriesMetadata();
result{s, 3} = colorMaps;
result{s, 4} = r.getMetadataStore();
fprintf('\n');
end
r.close();
This is the “bfGetReader” function used above.

```matlab
function r = bfGetReader(varargin)
% BFGETREADER return a reader for a microscopy image using Bio-Formats
% SYNOPSIS     r = bfGetReader()
%               r = bfGetReader(path)
% Input
%   id - (Optional - string) A valid path to the microscopy image
%   stitchFiles (Optional - scalar). Toggle the grouping of similarly
%               named files into a single dataset based on file numbering.
%               Default: false;
% Output
%   r - A reader object of class extending loci.formats.ReaderWrapper
% Adapted from bfopen.m

% OME Bio-Formats package for reading and converting biological file formats.
% Copyright (C) 2012 - 2013 Open Microscopy Environment:
  - Board of Regents of the University of Wisconsin-Madison
  - Glencoe Software, Inc.
  - University of Dundee
% This program is free software: you can redistribute it and/or modify
% it under the terms of the GNU General Public License as
% published by the Free Software Foundation, either version 2 of the
% License, or (at your option) any later version.
% This program is distributed in the hope that it will be useful,
% but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
% GNU General Public License for more details.
% You should have received a copy of the GNU General Public License
% along
% with this program; if not, write to the Free Software Foundation,
% Inc.,
% 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.

% Input check
ip = inputParser;
ip.addOptional('id', '', @ischar);
ip.addOptional('stitchFiles', false, @isscalar);
ip.parse(varargin{:});
id = ip.Results.id;
```
% load the Bio-Formats library into the MATLAB environment
status = bfCheckJavaPath();
assert(status, ['Missing Bio-Formats library. Either add loci_tools.jar ...
    'to the static Java path or add it to the Matlab path.'']);

% Prompt for a file if not input
isFile = exist(id, 'file') == 2;
isFake = ischar(id) && strcmp(id(end-3:end), 'fake');
if nargin == 0 || (~isFile && ~isFake)
    [file, path] = uigetfile(bfGetFileExtensions, 'Choose a file to open');
    id = [path file];
    if isequal(path, 0) || isequal(file, 0), return; end
elseif.isFile && ~verLessThan('matlab', '7.9')
    [~, f] = fileattrib(id);
    id = f.Name;
end

% set LuraWave license code, if available
if exist('lurawaveLicense')
    path = fullfile(fileparts(mfilename('fullpath')),
        'lwf_jsdk2.6.jar');
    javaaddpath(path);
    java.lang.System.setProperty('lurawave.license', lurawaveLicense);
end

r = loci.formats.ChannelFiller();
r = loci.formats.ChannelSeparator(r);
if ip.Results.stitchFiles
    r = loci.formats.FileStitcher(r);
end

r.setMetadataStore(loci.formats.MetadataTools.createOMEXMLMetadata());
r.setId(id);
This is the “bfCheckJavaPath” function used above.

```matlab
function [status, version] = bfCheckJavaPath(varargin)
% bfCheckJavaPath check Bio-Formats is included in the Java class path
%
% SYNOPSIS  bfCheckJavaPath()
%           status = bfCheckJavaPath(autoloadBioFormats)
%           [status, version] = bfCheckJavaPath()
%
% Input
%
%    autoloadBioFormats - Optional. A boolean specifying the action to take
%    if loci_tools is not in the Java class path. If true, add
%    loci_tools
%    to the dynamic Java path. Default - true
%
% Output
%
%    status - Boolean. True if loci_tools.jar is in the Java class path.
%             Empty string otherwise.
%
%    version - String specifying the current version of Bio-Formats if
%    loci_tools.jar is in the Java class path. Empty string else.
%
% OME Bio-Formats package for reading and converting biological file formats.
%
% Copyright (C) 2012 - 2013 Open Microscopy Environment:
% - Board of Regents of the University of Wisconsin-Madison
% - Glencoe Software, Inc.
% - University of Dundee
%
% This program is free software: you can redistribute it and/or modify
% it under the terms of the GNU General Public License as
% published by the Free Software Foundation, either version 2 of the
% License, or (at your option) any later version.
%
% This program is distributed in the hope that it will be useful,
% but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
% GNU General Public License for more details.
%
% You should have received a copy of the GNU General Public License
% along
% with this program; if not, write to the Free Software Foundation,
% Inc.,
% 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.
%
% Input check
ip = inputParser;
ip.addOptional('autoloadBioFormats', true, @isscalar);
ip.parse(varargin{:});
```
% Check if loci_tools is in the Java class path (static or dynamic)
jPath = javaclasspath('-all');
isLociTools = cellfun(@(x) ~isempty(regexp(x, '.*loci_tools.jar$', 'once')), jPath);
status = any(isLociTools);

if ~status & ip.Results.autoloadBioFormats,
    % Assume the jar is in Matlab path or under the same folder as this file
    path = which('loci_tools.jar');
    if isempty(path)
        path = fullfile(fileparts(mfilename('fullpath')), 'loci_tools.jar');
    end
    assert(exist(path, 'file') == 2, 'Cannot automatically locate loci_tools.jar');

    % Add loci_tools to dynamic Java class path
    javaaddpath(path);
    status = true;
end

if status
    % Read Bio-Formats version
    version = char(loci.formats.FormatTools.VERSION);
else
    version = '';
end
This is the “bfGetFileExtensions” function used above.

```matlab
function fileExt = bfGetFileExtensions
% bfGetFileExtensions list all extensions supported by Bio-Formats
%
% Synopsis: fileExt = bfGetExtensions()
%
% Input
% none
%
% Output
% fileExt: a cell array of dimensions n x2 where the first column
% gives the extension and the second the name of the corresponding
% format.
% This cell array is formatted to be used with uigetfile function.
%
% OME Bio-Formats package for reading and converting biological file
% formats.
%
% Copyright (C) 2012 - 2013 Open Microscopy Environment:
% - Board of Regents of the University of Wisconsin-Madison
% - Glencoe Software, Inc.
% - University of Dundee
%
% This program is free software: you can redistribute it and/or modify
% it under the terms of the GNU General Public License as
% published by the Free Software Foundation, either version 2 of the
% License, or (at your option) any later version.
%
% This program is distributed in the hope that it will be useful,
% but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
% GNU General Public License for more details.
%
% You should have received a copy of the GNU General Public License
% along
% with this program; if not, write to the Free Software Foundation,
% Inc.,
% 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.

% Get all readers and create cell array with suffixes and names
imageReader = loci.formats.ImageReader();
readers = imageReader.getReaders();
fileExt = cell(numel(readers), 2);
for i = 1:numel(readers)
    suffixes = readers(i).getSuffixes();
    fileExt{i, 1} = arrayFun(@char, suffixes, 'Unif', false);
    fileExt{i, 2} = char(readers(i).getFormat().toString);
end
```
% Concatenate all unique formats
allExt = unique(vertcat(fileExt(:, 1)));
allExt = allExt(~cellfun(@isempty, allExt));
fileExt = vertcat({allExt, 'All formats'}, fileExt);

% Format file extensions
for i = 1:size(fileExt, 1)
    fileExt{i, 1} = sprintf("*.%s;", fileExt{i, 1}{:});
    fileExt{i, 1}(end) = [];
end
This is the “bfGetPlane” function used above.

```matlab
function I = bfGetPlane(r, iPlane, varargin)
% BFGETPLANE Retrieve the plane data from a reader using Bio-Formats
%   I = bfGetPlane(r, iPlane) returns a specified plane from the input
%   format reader. The index specifying the plane to retrieve should be
%   contained between 1 and the number of planes for the series.
%   I = bfGetPlane(r, iPlane, x, y, width, height) only returns the
%   tile
%   which origin is specified by (x, y) and dimensions are specified by
%   (width, height).
% Examples
%   I = bfGetPlane(r, 1)  % First plane of the series
%   I = bfGetPlane(r, r.getImageCount())  % Last plane of the series
%   I = bfGetPlane(r, 1, 1, 1, 20, 20)  % 20x20 tile originated at (0, 0)
% See also: BFGETREADER

% OME Bio-Formats package for reading and converting biological file
% formats.
% Copyright (C) 2012 - 2013 Open Microscopy Environment:
% - Board of Regents of the University of Wisconsin-Madison
% - Glencoe Software, Inc.
% - University of Dundee
% This program is free software: you can redistribute it and/or modify
% it under the terms of the GNU General Public License as
% published by the Free Software Foundation, either version 2 of the
% License, or (at your option) any later version.
% This program is distributed in the hope that it will be useful,
% but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
% GNU General Public License for more details.
% You should have received a copy of the GNU General Public License
% along
% with this program; if not, write to the Free Software Foundation,
% Inc.,
% 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.

% Input check
ip = inputParser;
ip.addRequired('r', @(x) isa(x, 'loci.formats.IFormatReader') && ...
               ~isempty(x.getCurrentFile()));
ip.parse(r);
```
% Plane check
isValidPlane = @(x) isscalar(x) && ismember(x, 1 : r.getImageCount());
ip.addRequired('iPlane', isValidPlane);

% Optional tile arguments check
isValidX = @(x) isscalar(x) && ismember(x, 1 : r.getSizeX());
isValidY = @(x) isscalar(x) && ismember(x, 1 : r.getSizeX());
ip.addOptional('x', 1, isValidX);
ip.addOptional('y', 1, isValidY);
ip.addOptional('width', r.getSizeX(), isValidX);
ip.addOptional('height', r.getSizeY(), isValidY);
ip.parse(r, iPlane, varargin{:});

% Additional check for tile size
assert(ip.Results.x - 1 + ip.Results.width <= r.getSizeX(),
'MATLAB:InputParser:ArgumentFailedValidation',
'Invalid tile size');
assert(ip.Results.y - 1 + ip.Results.height <= r.getSizeY(),
'MATLAB:InputParser:ArgumentFailedValidation',
'Invalid tile size');

% Get pixel type
pixelType = r.getPixelType();
bpp = loci.formats.FormatTools.getBytesPerPixel(pixelType);
floating = loci.formats.FormatTools.isFloatingPoint(pixelType);
signed = loci.formats.FormatTools.isSigned(pixelType);
little = r.isLittleEndian();

plane = r.openBytes(iPlane - 1, ip.Results.x - 1, ip.Results.y - 1,
... ip.Results.width, ip.Results.height);

% convert byte array to MATLAB image
if signed
  % can get the data directly to a matrix
  I = loci.common.DataTools.makeDataArray2D(plane, ...
    bpp, fp, little, ip.Results.height);
else
  % get the data as a vector, either because makeDataArray2D
  % is not available, or we need a vector for typecast
  I = loci.common.DataTools.makeDataArray(plane, ...
    bpp, fp, little);
end

% Java does not have explicitly unsigned data types;
% hence, we must inform MATLAB when the data is unsigned
if ~signed
  % NB: arr will always be a vector here
  switch class(I)
    case 'int8'
      I = typecast(I, 'uint8');
    case 'int16'
      I = typecast(I, 'uint16');
  end

98
    case 'int32'
        I = typecast(I, 'uint32');
    case 'int64'
        I = typecast(I, 'uint64');
    end
end

if isvector(I)
    % convert results from vector to matrix
    shape = [ip.Results.width ip.Results.height];
    I = reshape(I, shape)';
end
Appendix C: caDNAno Images
Figure 25 displays the LPP caDNAno image without the 14 overhangs added for additional CellCro capabilities.

![Figure 25: LPP caDNAno Image](image)

The cross section of this structure is shown in Figure 26. Note that the overhangs on helices 37, 35, 33, 36, 32, 40, 34, and 48 are overhangs while all other helices are the main component of the structure.
Figure 27: LPP caDNAno Cross Section