NORTHWESTERN UNIVERSITY

Experiment-Driven Modeling of Plasmonic Nanostructures

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

DOCTOR OF PHILOSOPHY

Field of Materials Science and Engineering

By

Alexander John Hryn

EVANSTON, ILLINOIS

December 2016

© Copyright Alexander Hryn 2016

All Rights Reserved

ABSTRACT

Experiment-Driven Modeling of Plasmonic Nanostructures

Alexander John Hryn

Plasmonic nanostructures can confine light at their surface in the form of surface plasmon polaritons (SPPs) or localized surface plasmons (LSPs) depending on their geometry. SPPs are excited on nano- and micropatterned surfaces, where the typical feature size is on the order of the wavelength of light. LSPs, on the other hand, can be excited on nanoparticles much smaller than the diffraction limit. In both cases, far-field optical measurements are used to infer the excited plasmonic modes, and theoretical models are used to verify those results. Typically, these theoretical models are tailored to match the experimental nanostructures in order to explain observed phenomena. In this thesis, I explore incorporating components of experimental procedures into the models to increase the accuracy of the simulated result, and to inform the design of future experiments. First, I examine SPPs on nanostructured metal films in the form of low-symmetry moiré plasmonic crystals. I created a general Bragg model to understand and predict the excited SPP modes in moiré plasmonic crystals based on the nanolithography masks used in their fabrication. This model makes use of experimental parameters such as periodicity, azimuthal rotation, and number of sequential exposures to predict the energies of excited SPP modes and the opening of plasmonic band gaps. The model is further expanded to apply to multiscale gratings, which have patterns that contain hierarchical periodicities: a sub-micron primary periodicity, and microscale superperiodicity. A new set of rules was established to determine how superlattice SPPs are excited, and informed development of a new fabrication technique to create superlattices with multiple primary periodicities that absorb light over a wider spectral range than other plasmonic structures. The second half of the thesis is based on development of finite-difference time-domain (FDTD) simulations of plasmonic nanoparticles. I created a new technique to model pyramidal bowtie nanoparticle dimers based on the experimental fabrication procedure. This model was used to sweep various experimental parameters to identify their effect on the LSP resonance of the bowties. Analyzing the near-field distribution around these particles revealed the origin of a miscategorized LSP mode to be an out-of-plane dipole. Finally, I developed a finite-difference time-domain model that simulates the images generated by differential interference contrast (DIC) microscopy of gold nanorods. I discovered that the image contrast of gold nanorods is dependent on the wavelength of incident light relative to the LSP resonance wavelength. Incorporating experimental parameters into the DIC model allowed me to find a correlation between the electric near-field and far-field image contrast, uncovering the origin of this wavelength dependence. Additionally, the simulated DIC image patterns aid in breaking the angular degeneracy associated with the rotation of symmetric nanorods and can be used as training data for future machine learning algorithms to predict the size, shape, and orientation of nanoparticles from far-field images alone.

ACKNOWLEDGEMENTS

I would first like to thank my family starting with my parents, John and Roberta, for always supporting me through many, many years of school, and being a source of confidence and encouragement as I progressed through my time at Northwestern. My brother, David, has been a great friend over the last 23 years, and has always been interested and supportive of my studies and research. I thank my aunts, uncles, cousins, and grandparents for always encouraging my work towards a PhD; it is a wonderful reminder of this accomplishment in the face of many challenges.

My advisor, Teri Odom, has always pushed me to work harder, write better, and make more beautiful figures for any project I have worked on. She has been a wonderful role model, and always supportive of my work. It has been a pleasure to work in her lab, and I take many lessons into my personal and professional future. I also thank my committee members, George Schatz, Bruce Wessels, Lincoln Lauhon, and Robert Chang, for providing helpful input during and after my graduate exams.

I am grateful for Northwestern University, the Materials Science and Chemistry departments, Argonne National Laboratory, and all the centers that I have been fortunate to have used during my graduate work. The incredible facilities available to me, especially EPIC, NUFAB, CNM, and their amazing staff helped make my research possible. I also acknowledge my various sources of funding over the last six years including the NDSEG fellowship for providing the freedom to pursue new and exciting research.

Finally, I give great thanks to all my friends, new and old, that have helped me make this journey though grad school. My girlfriend Kayla has been with me since we met our first year, and we have grown together both personally and professionally throughout our time at

Northwestern. There is no other person with whom I would have rather spent the last five years of my life. My lab mates in The Odom Group have been a pleasure to work with, and several of us have become good friends over the years. Specifically, Steve Lubin, Mark Huntington, Cliff Engel, Angela Chang, Michael Knudson, and Kavita Chandra have been great friends both in and out of lab, and I value the relationships we have built.

I have been incredibly fortunate to have so many people who supported me during my time at Northwestern, and the work in this thesis was possible because of them.

LIST OF ABBREVIATIONS

AuNR	Gold nanorod
DF	Darkfield
DIC	Differential interference contrast
EBL	Electron-beam lithography
FDTD	Finite-difference time-domain
FIB	Focused ion-beam milling
FT	Fourier transform
LELE	Litho-etch-litho-etch
LSP	Localized surface plasmon
MNL	Moiré nanolithography
NIR	Near-infrared
NP	Nanoparticle
PC	Plasmonic crystal
PDMS	Polydimethylsiloxane
PML	Perfectly matched layers
PR	Photoresist
PSP	Phase-shifting photolithography
RIE	Reactive ion etcher
SPP	Surface plasmon polariton
UV	Ultraviolet

TABLE OF CONTENTS

ABSTRACT **3**

ACKNOWLEDGEMENTS **5**

LIST OF ABBREVIATIONS **7**

TABLE OF CONTENTS **8**

LIST OF FIGURES **12**

CHAPTER 1 : INTRODUCTION TO PLASMONIC NANOSTRUCTURES **21**

1.1 Introduction to Plasmonics 22

1.1.1 Surface plasmon polaritons 23

1.1.2 Localized surface plasmons 24

1.2 Fabrication of nanostructures 25

1.2.1 Phase-shifting photolithography 26

1.2.2 Large-area plasmonic substrates 26

1.2.3 Masked deposition of metal nanoparticles 27

1.3 Measurements and models of surface plasmons 28

1.3.1 Microscopy and spectroscopy 28

1.3.2 Finite-difference time-domain simulations 29

1.4 Scope of this thesis 29

CHAPTER 2 : LOW-SYMMETRY MOIRÉ PLASMONIC CRYSTALS **32**

2.1 Background 33

2.2 Results and Discussion 34

2.2.1 Fabrication and modeling of quasiperiodic patterns 34

2.2.2 SPP excitations in high-symmetry 10-fold quasiperiodic PCs 41

2.2.3 Reducing SPP degeneracy with low-symmetry quasiperiodic lattices 45

2.2.4 Multi-periodic quasicrystals and a general Bragg model 47

2.2.5 Plasmonic band gaps in low-symmetry PCs 50

2.3 Experimental methods 52

2.3.1 Lithography procedure for MNL 52

2.3.2 FDTD simulations of phase-shifting lithography 52

2.3.3 Pattern transfer to quasiperiodic PCs 53

2.3.4 Angle-resolved reflectance spectroscopy 53

2.4 Summary 53

CHAPTER 3 : SURFACE PLASMON POLARITONS ON MULTISCALE GRATINGS .55

3.1 Background 56

3.2 Results and Discussion 57

3.2.1 Fabrication of multiscale arrays 57

3.2.2 SPP excitations on 1D plasmonic superlattices 61

3.2.3 Effects of alternate superperiodicity on SPPs 64

3.2.4 Incommensurate multiscale arrays with varying azimuthal angle. 67

3.2.5 Double periodicity superlattices 71

3.3 Experimental methods 76

3.3.1 Lithography procedure for multiscale arrays 76

3.3.2 Pattern transfer to plasmonic superlattice crystals 76

3.3.3 Angle-resolved reflectance spectroscopy 77

3.4 Summary 77

CHAPTER 4 : PROCESS-BASED MODELS OF PLASMONIC BOWTIE ANTENNAS .78

4.1 Background 79

4.2 Results and Discussion 80

4.2.1 Fabrication parameters in experiment and FDTD model 80

4.2.2 Linear properties of bowties with various geometries 84

4.2.3 Out-of-plane bowtie LSP mode 90

4.3 Experimental methods 95

4.3.1 Bowtie fabrication 95

4.3.2 FDTD simulations 96

4.4 Summary 96

CHAPTER 5 : CONTRAST INVERSION IN DIC OF GOLD NANORODS.......................97

5.1 Background 98

5.2 Results and Discussion 101

5.2.1 DIC microscopy simulation scheme 101

5.2.2 DIC FDTD simulations of single AuNR and dimers 104

5.2.3 Wavelength-dependent image contrast of AuNR 108

5.3 Experimental methods 118

5.3.1 FDTD simulations of DIC microscopy 118

5.3.2 AuNR sample preparation 122

5.3.3 DF imaging 122

5.3.4 DIC imaging 122

5.3.5 SEM imaging 123

5.4 Summary 123

REFERENCES 124

APPENDIX A: MATLAB CODE FOR MOIRÉ STRUCTURE GENERATION 135

APPENDIX B: MATLAB CODE FOR SPP MODE CALCULATIONS 143

APPENDIX C: LUMERICAL SCRIPT TO CREATE BOWTIE DIMERS 187

APPENDIX D: LUMERICAL SCRIPT TO PROCESS FDTD DIC IMAGES 194

CURRICULUM VITAE 198

LIST OF FIGURES

Figure 2.1: Fabrication scheme for moiré nanolithography. PDMS masks with different periodicities a_i and azimuthal angles φ_i are put into conformal contact with a PR-coated Si wafer and exposed to UV light. This process is repeated for the desired number of exposures. The resulting pattern is developed, then transferred to a Ag plasmonic crystal. 35

Figure 2.2: Quasiperiodic moiré patterns. (a) High symmetry quasicrystals are fabricated with by 5 exposures with PDMS masks having equal periodicities a_0 = 480 nm equiangular azimuthal rotations ($\Delta\varphi = \pi/5 = 36°$) (b) Rotationally asymmetric patterns with asymmetric exposure angles fabricated by 4 exposures with equal periodicities a_0 = 480 nm but all angles are within a single quadrant ($\varphi_4 - \varphi_1 = \pi/2 = 90°$). (c) Multiperiodic patterns fabricated by 3 exposures with different periodicities a_1 = 480 nm, a_2 = 645 nm, a_3 = 730 nm, and equiangular rotations ($\Delta\varphi = \pi/3 = 60°$). (a-c) SEM images (top) of the Si template or PR pattern agree with simulated structures (middle). FT of the SEM images reveal the reciprocal lattice of the patterns (bottom)..............................37

Figure 2.3: FDTD Simulation of phase-shifting photolithography. (a) Simulation scheme with electric field data from the 2D monitor overlaid onto the PR slab. 1D cross-section monitor is located in the center of the PR (white dashed line). (b) Cross sectional intensiy from the 1D monitor in (a) compared to a sinusoidal profile. 39

Figure 2.4: Graphical user interface (GUI) for the MATLAB computational program to predict the resultant patterns from MNL. Parameters panel (left) indicate where values are input into the program and the structure is displayed in the axis (right). The displayed pattern is from the multiperiodic quasicrystal pattern...40

Figure 2.5: Reflection spectra from high-symmetry quasiperiodic PC. (a) Calculated FT with basis vectors $\mathbf{k}_i$ shown. (b) Angle-resolved reflection spectra for the 10-fold quasiperiodic pattern taken from $\theta = 10°$ to $60°$ converted to dispersion diagram. (c) Predicted SPP mode dispersion calculated from the Bragg coupling equation for a Ag/air interface with some first order modes labeled. In (a,c) blue objects represent first-order modes and orange objects represent second-order modes. 42

Figure 2.6: Measurement scheme for plasmonic crystals. (left) Bragg coupling equation and incident light path for reflection spectra. (right) Relationship between FT and SPP dispersion for a 10-fold high-symmetry quasicrystal. First- (blue) and second- (orange) order mode families are indexed. Modes with energies beyond the measurement range are faded. 44

Figure 2.7: GUI for MATLAB program to process reflection spectra and calculate SPP mode dispersion. Import data panel is where the raw reflection spectra are input to the program. SPP modes panel is used to provide parameters to the Bragg coupling equation for SPP mode calculation. Axes (bottom, center and right) display spectra and modes in E–**k** or λ–θ format. Fourier transform axes (top right) displays the reciprocal lattice of the given structure. First- and second-order modes are colored blue and orange respectively. 46

Figure 2.8: Reflection spectra from quasiperiodic PC with asymmetric rotation angles. (a) Calculated FT with basis vectors $\mathbf{k}_i$ shown. (b) Angle-resolved reflection spectra taken from $\theta = 10°$ to $80°$ converted to dispersion diagram. (c) Predicted SPP mode dispersion calculated from the Bragg coupling equation for a Ag/air interface with some first order modes labeled. In (a,c) blue objects represent first-order modes and orange objects represent second-order modes. 48

Figure 2.9: Reflection spectra from a multiperiodic patterned PC. (a) Calculated FT with basis vectors $\mathbf{k}_i$ shown. (b) Angle-resolved reflection spectra taken from $\theta = 10°$ to 80° converted to dispersion diagram. (c) Predicted SPP mode dispersion calculated from the Bragg coupling equation for a Ag/air interface with some first order modes labeled. In (a,c) blue objects represent first-order modes and orange objects represent second-order modes. 49

Figure 2.10: Plasmonic band gaps in multiperiodic patterned PC. (center) Zoom of dispersion diagram with overlaid relevant SPP modes. Numbered circles highlight plasmonic band gaps. (left, right) FT of the pattern with modes and Bragg lines highlighted for specific band gaps. Arrows represent direction and magnitude of $\boldsymbol{k}_{||}$ and $\boldsymbol{k}_{\mathrm{SPP}}$ for the given E–$\mathbf{k}$ coordinates. 51

Figure 3.1: Fabrication scheme for multiscale superlattices. Multiscale superlattices use two consecutive exposures of phase shifting lithography and traditional contact lithography for the primary and superperiodicities respectively. After the first round of etching, the process can be repeated to fabricate doubly patterned superlattices. 58

Figure 3.2: 1D Superlattice. A SEM (a) of a 1D plasmonic superlattice and (b) its Fourier transform. (c) The Bragg peaks in the Fourier transform can be indexed as combinations of the primary periodicity (a_0) and superperiodicity (A_0) 60

Figure 3.3: Reflection spectra of superlattices with indexed SPP modes. Reflection spectrum ($\theta = 20°$) for a plasmonic superlattice with $a_0 = 400$ nm and $A_0 = 10$ μm and for an infinite 400-nm array (inset) (b) Dispersion diagram with the -1_0 SPP mode overlaid. (c) Zoom of the dispersion diagram in (b, dotted box) with SPP modes shown and labeled. The dashed white line indicates the light line (*i.e.* $\theta = 90°$)..62

Figure 3.4: Multiscale gratings with varying superperiodicity. SEM images of Si templates with primary periodicity $a_0 = 400$ nm and $A_0 = 10$ µm, 20 µm, 30 µm, and 50 µm (top to bottom). ...65

Figure 3.5: Reflection spectra of multiscale gratings with varying superperiodicity. Reflection spectra ($\theta = 65°$) of multiscale gratings with $a_0 = 400$ nm and $A_0 = 10$, 20, 30, and 50 µm (bottom to top). SPP modes are labeled according to their index. 66

Figure 3.6: Dispersion diagrams of multiscale gratings with varying superperiodicity. Multiscale gratings have primary periodicity $a_0 = 400$ nm and $A_0 = 10$, 20, 30, and 50 µm as indicated. 68

Figure 3.7: Multiscale gratings with varying azimuthal angles (φ). (a,c,e) SEM (top) and FT (bottom) for superlattices with $a_0 = 475$ nm and $A_0 = 10$ µm, and $\varphi = 0°$, 45° and 90° respectively. (b,d,f) Dispersion diagrams of the above structures showing the decrease in ΔE for the satellite peaks with increasing φ...69

Figure 3.8: Comparison between high-order and satellite SPP modes. Dispersion diagram of a multiscale grating with $a_0 = 475$ nm, $A_0 = 10$ µm, and $\varphi = 45°$. Overlay (red lines) shows the expected dispersion of high order SPP modes from the superperiodicity alone...........................72

Figure 3.9: Multiscale gratings with two primary periodicities. (a) SEM of the multiscale array with $a_0 = 400$ nm, $b_0 = 475$ nm, $A_0 = 10$ µm. (b) FT and (c) indexing scheme of the structure showing each primary periodicity with its own set of satellite peaks. (d) Dispersion diagram of the array with SPP modes from the primary periodicities only as overlay...73

Figure 3.10: Wide view FT for double patterned superlattices. The groupings for the two primary periodicities $a_0 = 400$ nm and $b_0 = 475$ nm are identified. The slight misalignment between the two periodicities is evident especially at high orders. The satellite modes are all aligned however, since there is only one superperiodicity $A_0 = 10$ µm..75

Figure 4.1: Fabrication scheme. Steps of the bowtie fabrication procedure from lithography until bowtie deposition. The final 3D image shows the particle deposition direction through the nanohole mask (red arrow) characterized by the deposition angle Ψ, azimuthal angle φ, and thickness t..81

Figure 4.2: Flexibility of FDTD model. (a) Scheme for the nanohole projection in FDTD to generate the nanoparticle object. (b) FDTD layout images showing nanoparticle objects created with various hole shapes and projection planes...83

Figure 4.3: Effect of RIE etch on Si pyramidal pit shape. (a) Undercut distances between pyramid edge and hole edge for 10 different RIE etch times followed by anisotropic wet etching. Each point represents an average of 5–10 measurements on different holes in the same sample. (b) SEM images of samples with 4 different etch times...85

Figure 4.4: Sweeps of Ψ and t for Au corner-centered bowties. (a,c) Schemes of the simulation parameters to generate the nanoparticle objects, with images representing the high and low extremes for Ψ (a) or t (c). (b,d) Simulated transmission spectra for bowties in the visible-NIR wavelength range. 87

Figure 4.5: FDTD and experimental comparison. (a,c) FDTD objects and SEM images of bowtie nanostructures using the same parameters. Simulation objects were made by using experimental conditions as inputs. (b,d) Experimental and simulated transmission spectra for each dimer. 88

Figure 4.6: Variations in experimental bowtie nanoparticle shapes. (a) Large-area SEM image of a bowtie dimer array. (b) Highlighted particles from (a) showing slight variations in geometry that could affect the bowtie LSP resonance. 91

Figure 4.7: LSP dependence on azimuthal deposition angle *φ*. (a) Transmission spectra for 10 different azimuthal angles *φ* from 0° to 45° in 5° increments show an increase in LSP mode intensity at 662 nm with increasing angle. The LSP mode at 796 nm remains unchanged. (b) Cross-sectional near-field intensity maps between the nanoparticles in the plane perpendicular to the dimer axis (dashed black lines). Dashed white lines show the boundaries of the pyramid. Near-field intensity is plotted on a log scale. 92

Figure 4.8: Geometric analysis of LSP modes for corner-centered dimers. Schemes of the overall dipolar charge distribution for the in-plane and out-of-plane LSP modes (left). The electric field vectors and charge distribution (red shows positive charge, blue shows negative charge) for each LSP mode on resonance (right) 94

Figure 5.1: FDTD scheme for DIC microscopy. (a) Side-view of the optical path for a DIC microscope. There are three components to the FDTD-DIC simulation environment: Setup, FDTD simulation, Processing. Each section produces optical polarization states corresponding to specific of the DIC microscope. (b) Top-down view of polarization state at different locations in the microscope. The phase shift in the intermediate beams induced by the AuNR causes the bright and dark contrast depending on particle orientation. 102

Figure 5.2: Simulated DIC images with different boundary conditions. (a) DIC images with height and width double the periodicity of the simulation. As the periodicity decreases, the background level between particles becomes smoother. (b) The same series of simulated DIC images as in (a) with only the center 4 μm × 4 μm square visible. 103

Figure 5.3: Phase difference for intermediate and combined beams. Images displaying the phase induced by a dark-aligned (a) or bright-aligned (b) AuNR compared to the background for the two intermediate beams. The beams are recombined at the Nomarski prism and the final phase difference is shown. 105

Figure 5.4: Rotation study of AuNRs. DIC images were measured and simulated for a single AuNR (a) and a V-shaped dimer (b) with rotation angles from $\theta = 0°$ to $\theta = 180°$ in 10° increments. The DIC images were correlated with SEM images and the SEM shown represents the particle orientation for $\theta = 0°$. 107

Figure 5.5: Distribution of LSP resonances for AuNR sample. (a) Darkfield image (true color) at 100× magnification showing the scattering of several AuNR. (b) Scattering spectra from select labeled AuNR with λ_{LSP} ranging from ca. 550 nm to 850 nm. 109

Figure 5.6: Large-area experimental DIC images of AuNR. A single set of AuNR with $\lambda_{\mathrm{DIC}} = 640$ nm (a) and $\lambda_{\mathrm{DIC}} = 750$ nm (b). Several particles show an inversion in contrast from bright to dark or vice versa between the two images. 111

Figure 5.7: Wavelength dependence of DIC images for a single AuNR. (a) SEM and scattering spectra for a single AuNR. Experimental (b) and simulated (c) DIC images at two DIC wavelengths: λ_{DIC} = 640 nm (top) and λ_{DIC} = 750 nm (bottom). (d) Contrast difference calculated from the DIC images showing the contrast inversion between the two wavelengths. Solid lines with markers are from experimental data; dotted lines are calculated from simulated DIC images. 112

Figure 5.8: FDTD near-field analysis of DIC contrast inversion. (a) Scattering simulation for the longitudinal mode of AuNR (25 × 75 nm in oil) with a scheme of the 1D and 2D monitor locations (inset). (b) E-field amplitude (1D monitor) as a function of wavelength for the AuNR. (c) 2D cross-sections of electric field amplitude at two wavelengths on the blue (square, λ_{DIC} = 808 nm) and red (triangle, λ_{DIC} = 856 nm) sides of the LSP wavelength. 114

Figure 5.9: FDTD near-field analysis of DIC contrast inversion. (a) Scattering simulation for the transverse orientation of AuNR (25 × 75 nm in oil) with a scheme of the 1D and 2D monitor locations (inset). (b) E-field amplitude (1D monitor) as a function of wavelength for the AuNR. (c) 2D cross-sections of electric field amplitude at two wavelengths (square, λ_{DIC} = 808 nm) and red (triangle, λ_{DIC} = 856 nm)...116

Figure 5.10: DIC response of AuNR with three different resonance locations. (a) The measured scattering spectra (normalized) of three individual AuNR. (b) SEM of the AuNRs aligned to $\theta = 0°$ and DIC image sets for λ_{DIC} = 640 nm (blue border) and λ_{DIC} = 750 nm (red border) for $\theta = 0°$ to 150°. 117

Figure 5.11: Rotational correlation study for AuNR **I**. (a) DIC images (2.6-µm width) at 5 DIC wavelengths from 600 nm to 750 nm and 18 rotational angles from 0° to 180°. (b) contrast difference calculated from the DIC images in (a)..119

Figure 5.12: Rotational correlation study for AuNR **II**. (a) DIC images (2.6-µm width) at 5 DIC wavelengths from 600 nm to 750 nm and 18 rotational angles from 0° to 180°. (b) contrast difference calculated from the DIC images in (a)..120

Figure 5.13: Rotational correlation study for AuNR **III**. (a) DIC images (2.6-µm width) at 5 DIC wavelengths from 600 nm to 750 nm and 18 rotational angles from 0° to 180°. (b) contrast difference calculated from the DIC images in (a)..121

CHAPTER 1:

INTRODUCTION TO PLASMONIC NANOSTRUCTURES

1.1 Introduction to Plasmonics

The field of plasmonics studies the confinement and manipulation of electromagnetic waves at length scales smaller than the wavelength of free-space light. For plasmons excited in the visible spectrum, the confinement exists on the nanoscale, having dimensions on the order of hundreds of nanometers or smaller. Advancements in plasmonics have grown over the last few decades with parallel improvements in nanofabrication techniques. These techniques have driven the use of plasmonic nanostructures in a wide range of applications across multiple disciplines. In the areas of chemistry and physics, plasmons have enhanced photovoltaics,[1-2] chemical[3] and biosensing,[4-6] subwavelength focusing[7-8] and imaging,[9-10] and nanoscale lasing.[11-12] Plasmonic properties of nanoparticles have extended to the rapidly expanding field of nanobiology, where they have seen use in the identification[13] and treatment of diseases through direct heating[14-15] and light triggered drug release.[16]

The response of a metal nanostructure when exposed to incident light, is described as a surface plasmon: a collective oscillation of free electrons at the interface between a metal and dielectric material. The nature of the surface plasmon resonance is dependent on the metal used, the refractive index of the dielectric environment, and the nanoscale geometry of the interface. Two types of surface plasmons can be excited on a metal surface: surface plasmon polaritons (SPP), which propagate along continuous films, and localized surface plasmons (LSP), which surround isolated nanoparticles.

This introduction will provide a brief overview of the fundamentals behind both SPPs and LSPs. We will discuss nanofabrication techniques that enable features that support surface plasmons; specifically, we focus on top-down nanofabrication techniques for the formation of nanostructured metal films and particle arrays. Finally, we will explore different measurement techniques that are used to identify and characterize plasmonic structures.

1.1.1 Surface plasmon polaritons

The existence of surface plasmon polaritons can be shown by solving Maxwell's equations at the metal-dielectric interface.[17] The solution produces SPP modes that propagate along the interface, and display evanescent decay into both the metal substrate and the dielectric superstrate. Their dispersion is given by the following equation:[17]

$$k_{\mathrm{SPP}} = k_0\sqrt{\frac{\varepsilon_{\mathrm{d}}\varepsilon_{\mathrm{m}}}{\varepsilon_{\mathrm{d}} + \varepsilon_{\mathrm{m}}}} \tag{1.1}$$

where k_{SPP} is the wavevector of the SPP, $k_0 = \omega / c$ is the wavevector of free space light, and ε_{m} and ε_{d} are the complex permittivity of the metal and dielectric, respectively, which are functions of the optical frequency ω. The dielectric material is assumed to have a negligible absorption and therefore its permittivity has no complex component and can be described by $\varepsilon_{\mathrm{d}} = n^2$, where n is the refractive index. Boundary conditions at the metal–dielectric interface place restrictions on the permittivity of the metal, specifically that $\mathrm{Re}(\varepsilon_{\mathrm{m}})$ is negative. Based on equation (1.1), the wavevector of the SPP will exceed that of free-space light, resulting in a momentum mismatch that puts the SPPs outside of the light cone and must be overcome.

Multiple ways exist to increase the momentum of incident light to overcome the momentum mismatch,[18] but in the context of this thesis, we will discuss the use of grating coupling. A periodic structure on the metal surface creates Bragg boundary conditions along the interface. Similar to the influence of an atomic lattice on electrons, this periodicity allows the SPP modes to be reflected, or folded, into the light cone. The Bragg coupling condition describes this effect:

$$\mathbf{k}_{\text{SPP}} = \mathbf{k}_{||} - \mathbf{G} \tag{1.2}$$

where $\boldsymbol{k}_{\text{SPP}}$ is the wave vector of the SPP mode, $\mathbf{G}$ is the grating vector, and $\boldsymbol{k}_{\|}$ is the in-plane wavevector of light, described by

$$\mathbf{k}_{||} = \mathbf{k} \sin \theta \tag{1.3}$$

With the addition of the grating momentum, the contribution to $\boldsymbol{k}_{\text{SPP}}$ from incident light with wavevector $\boldsymbol{k}$ is dictated by the incident angle θ. The grating vector is defined by the reciprocal lattice of the grating,

$$\mathbf{G} = c_1 \frac{2\pi}{a_0} \hat{\mathbf{k}}_x \tag{1.4}$$

where a_0 is the periodicity of the grating, and c_1 is an integer. For SPP excitations in the visible–NIR spectrum, as discussed in this thesis, strong modes are generated by gold and silver metals when coupled to light through periodic structures with characteristic $a_0 \leq 1$ µm.

1.1.2 Localized surface plasmons

Contrary to the propagating waves of SPPs, localized surface plasmons are characterized by confined evanescent fields around the surface of a nanoparticle. This confinement can produce local electric field intensities orders of magnitude larger than the incident fields.[17, 19-20] Since the LSP is dominated by confinement effects, tuning the resonance wavelength is achievable not only by changing the material[21-22] or dielectric environment,[23] but also by tuning the size[24-26] and shape[24, 27] of the nanoparticles. Specifically, many different nanoparticle shapes such as spheres,[28] rods,[29-30] prisms,[31] pyramids,[32] and stars[16, 33-34] have been studied.

The plasmonic response of a metal nanoparticle is observed as the scattering and absorption of incident light. For spherical nanoparticles that are much smaller than the wavelength of light, the incident light can be treated as uniform oscillating electric field that induces a dipole moment in the nanoparticle. The nanoparticle therefore emits an oscillating, dipolar electromagnetic field in response to the plasmonic excitation. As with SPPs, application of Maxwell's equations give the solutions for the LSP in the form of scattering and absorption cross sections:[17]

$$\sigma_{\mathrm{scatt}} = \frac{8\pi}{3} k^4 a^6 \left| \frac{\varepsilon_{\mathrm{m}} - \varepsilon_{\mathrm{d}}}{\varepsilon_{\mathrm{m}} + 2\varepsilon_{\mathrm{d}}} \right|^2 \tag{1.5}$$

$$\sigma_{\mathrm{abs}} = 4\pi k a^3 \,\mathrm{Im}\left[\frac{\varepsilon_{\mathrm{m}} - \varepsilon_{\mathrm{d}}}{\varepsilon_{\mathrm{m}} + 2\varepsilon_{\mathrm{d}}} \right] \tag{1.6}$$

where *a* is the radius of the nanoparticle. The resonance maximum exists as $|\varepsilon_{\mathrm{m}} + 2\varepsilon_{\mathrm{d}}|$ approaches 0, where ε_{m} is the complex, frequency dependent permittivity of the metal. As nanoparticles become larger, *e.g.* ~100 nm in size for interactions with visible and near-infrared (NIR) light, the simple dipole approximation no longer applies, and Mie's theory in electrodynamics must be applied.[17] Dipole absorption and scattering still provide a good approximation on resonance, and will be used for the plasmonic systems discussed in this thesis.

1.2 Fabrication of nanostructures

As described in the previous section, nanostructures with dimensions near or below the wavelength of light are required for excitation of surface plasmons. In the visible and NIR region of the spectrum, these dimensions produce a challenge in the fabrication of such structures. Top-down fabrication is a series of processes where features are defined first in photoresist (PR) using lithographic techniques, followed by transfer to silicon, metal, or dielectric materials. Large-scale fabrication of micron-scale features typically has been achieved by photolithography, the dominant technique in the semiconductor industry. The increased demand to reduce feature sizes in accordance with Moore's Law[35] has driven major advancements in photolithography technology, and similarly, the increase in associated cost and accessibility. Serial patterning techniques such as electron-beam lithography (EBL)[36] or focused ion-beam milling (FIB),[37] have been developed as alternatives for producing subwavelength structures, but their total patterned areas are limited to the microscale.

1.2.1 Phase-shifting photolithography

Standard photolithography relies on differences in intensity of incident light to pattern photoresist. For example, a quartz plate is patterned with a thin layer of chromium metal and placed into contact with a PR-coated silicon wafer so that the PR is exposed only through transparent regions of the mask. Phase-shifting photolithography (PSP) is a variation of photolithography where a fully transparent mask manipulates the phase of an incident light wave to create interference patterns that dictate the patterns formed in PR.[38-40] This technique uses soft elastomeric photomasks based on polydimethylsiloxane (PDMS) patterned with periodic features that can be created by multiple methods including traditional photolithography, interference lithography,[41] or EBL. Soft PDMS alone is unable to replicate submicron features, so composite masks made of a patterned hard-PDMS layer (~100 μm thick) bonded to a slab of soft PDMS are required.[42-43] With these composite masks, PSP is capable of producing feature sizes ≤ 100 nm, ideal for visible/NIR excitation of surface plasmons.

1.2.2 Large-area plasmonic substrates

Once features in PR can be patterned on the appropriate length scale, these features can be transferred into periodic nanostructured films to excite SPPs,[44-45] or into periodic arrays of nanoparticles that support LSPs.[46-47] To create patterned metal films with nanoscale periodicity, referred to as plasmonic crystals (PCs), several additional fabrication steps are required. First, the photoresist pattern is transferred into the silicon substrate. This transfer relies on the deposition of an etch mask, typically a thin (10 nm) metal such as Cr, followed by a chemical or physical etch of the underlying Si. Anisotropic chemical etching is achieved with a mixture of KOH and isopropanol that produces smooth, square pyramidal pits in a Si[100] wafer.[48] A physical/chemical etch can be achieved in a reactive ion etcher (RIE) or deep RIE (DRIE) with a plasma of fluorinated gasses, usually SF_6, CHF_3, and C_4F_8. The profile of the etched pits can be tuned from a rounded bowl-like shape to pits with steep sidewalls depending on the parameters of the etch.[49] After the Si substrates are fabricated, SPP-supporting films such as nanohole arrays or gratings can be generated by deposition of a plasmonic metal.

1.2.3 Masked deposition of metal nanoparticles

The size and shape of nanoparticles greatly affects their plasmon resonance, thus the ability to create nanoparticles with well-controlled geometry is desired. The simplest means to create nanoparticles with arbitrary shape on a flat substrate is by using EBL to create the desired pattern in resist followed by metal deposition and lift-off. The limitations of available patterned area, as with PC fabrication, has enabled advanced fabrication techniques by masked deposition through large-area substrates. This process relies on fabrication of large area patterns elevated on a higher plane than the target surface. Self-assembly of polystyrene nanospheres into ordered hexagonal arrays provides a nanohole array suspended above a surface that can be used as a deposition mask for large-area arrays of nanoparticles.[26] The same processes used to fabricate PCs by phase-shifting photolithography can also be used to create similar, elevated hole arrays. Nanopyramids[32] and pyramidal particle assemblies[50] can be fabricated by using the Si etch mask as a deposition mask as well. Additionally, the plasmonic hole arrays generated by Si templates can also be repurposed as deposition masks to create arrays of nanodisks on other substrates.[51]

1.3 Measurements and models of surface plasmons

1.3.1 Microscopy and spectroscopy

Refractive and diffractive spectroscopy in the visible spectrum have existed for hundreds of years to analyze the absorption and emission profiles of various materials. Some of the first evidence of surface plasmons was observed and described by Wood[52] and Lord Raleigh[53-54] at the beginning of the 20th century by studying the diffraction of light from metallic gratings. Since plasmons are excited in nanostructures over specific wavelength ranges, examining the reflection, scattering, and transmission spectra of such structures is required to deduce the plasmonic effects. Minima in reflection spectra of metal films indicate the trapping of light at the surface of the material by the excitation of SPPs. Transmission through sub-wavelength hole arrays in thin metal films at specific wavelengths, however, can be increased due to the excitation and coupling of SPPs.[55] As described above, the excitation of SPPs on metal films is dispersive in nature, and the wavevector $\mathbf{k}_{\mathrm{SPP}}$ is dependent on the parallel component of incident light (equation (1.3)). To identify the full dispersive properties of SPPs, angle-resolved spectroscopy to high angle θ is required. A simple geometric analysis reveals that increasing the incident angle requires increased sample size in the direction of the incident plane. A custom rotational stage spectrometer system, combined with large-area nanofabrication, enables characterization of SPP modes.[5, 45, 56-57]

On the other end of the size spectrum, LSPs can be identified by the scattering spectrum from a single nanoparticle. Darkfield microscopy is a technique where a hollow cone of light with high numerical aperture ($\mathrm{NA} = n \sin \theta$) is focused onto a sample, and light is collected with an objective lens having a *smaller* NA. The difference in NA results in the incident light not entering the objective lens, and only light scattered by the sample is collected. For plasmonic nanoparticles, the dipolar emission of light is collected, and a spectrometer is used to identify its wavelength distribution. Darkfield microscopy has been used to identify of various types of nanoparticles by their size and shape,[58] orientation,[59] and material composition,[59] or to track their movement in biological environments.[60]

1.3.2 Finite-difference time-domain simulations

The previous section discussed ways to measure the far-field response of light interacting with a plasmonic system. In some cases, the far-field distribution alone is insufficient to determine the nature of the plasmon mode, and the near-field must also be studied. Examining the near-field distribution around nanoparticles is difficult to achieve experimentally compared to far-field properties, but experimental techniques to do so are available.[61] An alternative approach is to generate numerical simulations of plasmonic systems, where the nearfield distribution can be calculated. A common approach for computations of plasmonic materials is the finite-difference time-domain (FDTD) method. Briefly, FDTD involves discretizing materials into cells then solving time-dependent Maxwell's equations for the system.[62] In practice for plasmonic materials, a broadband light pulse is incident on a nanostructure, and the natural evolution of the plasmonic response is simulated. Near-field visualization can provide information on the origin of LSP modes, or identify coupling between neighboring nanoparticles.[24, 63]

1.4 Scope of this thesis

This dissertation discusses the excitation of surface plasmons in both continuous and isolated nanostructures. We focus on how the plasmonic response of these nanostructures can be better understood and contextualized through calculations and modeling derived by experimental conditions. Incorporation of the experimental processing into the model as opposed to only examining the final structures can improve the overall result and build intuition about the origin of plasmonic modes.

The first chapter introduces surface plasmons and their excitation on both nanostructured surfaces as SPPs and isolated nanoparticles as LSPs. We introduce nanofabrication processes to create large area plasmonic substrates; continuous metal films that support SPPs, or arrays of individual nanoparticles with tunable LSPs. We finally discuss the tools available to measure and characterize the plasmonic response of these materials.

The next two chapters cover large-area, micro- and nanostructured metal films. In the second chapter, we examine the fabrication and SPP excitation on quasiperiodic PCs. We identify the mechanism behind moiré nanolithography (MNL) and fabricate both high-symmetry and low-symmetry quasiperiodic geometries. We develop a universal model for the Bragg coupling equation directly influenced by the steps in MNL to index and characterize SPP modes. The development of plasmonic band gaps at the intersection of different SPP modes can also be understood by the new model.

The third chapter examines the fabrication of multiscale plasmonic gratings, or plasmonic superlattices. We combine traditional photolithography with PSP to fabricate PCs with periodicities on multiple length scales. The SPP modes excited on these multiscale gratings show different dispersion than expected, and we determine the origin of this difference to be directly related to the nanofabrication conditions.

The following two chapters discuss the modeling of individual plasmonic nanoparticles. Chapter four describes a FDTD model of plasmonic bowtie antennas. The model is not based on the final shape of the nanoparticle, but instead only relies on the metal deposition parameters used in the fabrication process. This new model predicts the far-field spectra of bowtie arrays, and is used to determine the origin of measured plasmonic resonances based on nearfield analysis.

The final chapter is based on differential interference contrast (DIC) microscopy of gold nanorods (AuNR). We develop a FDTD model of the DIC microscope that predicts the far-field orientation-dependent images of the AuNR. An inversion of DIC image contrast as a function of wavelength was discovered, and nearfield analysis through the FDTD model revealed the origin to be fundamental to the localized plasmon resonance.

CHAPTER 2:

LOW-SYMMETRY MOIRÉ PLASMONIC CRYSTALS

2.1 Background

Light can be concentrated below the diffraction limit as surface plasmon polaritons (SPPs),[64] when incident light and conduction band electrons of a metal couple at a metal-dielectric interface.[8, 18] SPPs cannot be excited by free-space light incident on a flat metal film due to the difference of in-plane momentum between the two. One method to overcome this momentum mismatch is to pattern the metal surface with periodic, subwavelength 1D and 2D arrays, referred to as plasmonic crystals (PCs).[8, 18, 56, 65] The SPP resonance modes of PCs are determined by the geometry of the patterns and by the dielectric function of the metal and dielectric.[17-18] The tunability of these resonances has enabled use of PCs in a broad range of applications, such as biosensing,[4, 66] far-field focusing of light,[55, 67-68] and plasmonic lasing.[69] Since the number and range of SPP modes are mostly limited by the periodicity of the subwavelength pattern, however, other potential applications have not been realized, such as plasmon-enhanced photovoltaics, where broadband light trapping and waveguide effects are desired.[1] One approach to increase the spectral range of SPP excitations is to change the symmetry of the PC to lift degeneracies among SPP modes or to increase the overall number of available modes.[57, 70]

Periodic arrays in PCs are restricted to 6-fold symmetry due to the crystallographic restriction theorem,[71] so further increasing the symmetry requires quasicrystalline patterns. High-symmetry patterns such as the 10-fold Penrose tiling[72] or 12-fold Socolar tiling[73] have demonstrated enhanced optical transmission due to SPPs when transferred to metal nanohole arrays.[74-76] These structures were usually fabricated using focused ion beam (FIB) milling or e-beam lithography, limiting their total area (< 1 mm^2) and available measurement techniques. Recently, we developed a new nanofabrication technique to create high-symmetry nanoscale patterns over large areas: moiré nanolithography (MNL).[77]

MNL has been used to fabricate high-symmetry lattices from 4-fold periodic arrays to 36-fold quasicrystals. Using MNL, patterns with 8-fold symmetry have been transferred into large-area quasiperiodic PCs, which contain over double the number of modes as a 4-fold, square lattice.[57] The analysis of the SPP modes required the derivation of an quasiperiodic Bragg coupling equation with a modified grating vector with four components. MNL and other quasiperiodic PCs, however, are not limited to high-symmetry lattices. Additionally, a general form of the Bragg coupling equation to describe any symmetry has not been established.

Here we show the mechanism behind MNL and create a computational tool to predict and identify any arbitrary quasiperiodic pattern that can be fabricated. We develop a general form of the Bragg coupling equation with an indexing scheme that can predict the dispersion of SPP modes in PCs with any symmetry. These two models are tested on the fabrication and SPP excitation in a 10-fold PC. Finally, we fabricate low-symmetry quasiperiodic PCs with asymmetric rotational symmetry and multiple independent symmetries, index their SPP modes and identify the formation of plasmonic band gaps at first and second order Bragg lines of the lattice.

2.2 Results and Discussion

2.2.1 Fabrication and modeling of quasiperiodic patterns

Fabrication of quasiperiodic nanopatterns was achieved by moiré nanolithography (**Figure 2.1**). First, a Si wafer coated with photoresist (PR) was placed in conformal contact with a polydimethylsiloxane (PDMS) phase-shifting lithography[38-39] mask and exposed to ultraviolet (UV) light. For any desired number of exposures *n*, this process was repeated. Each iteration used

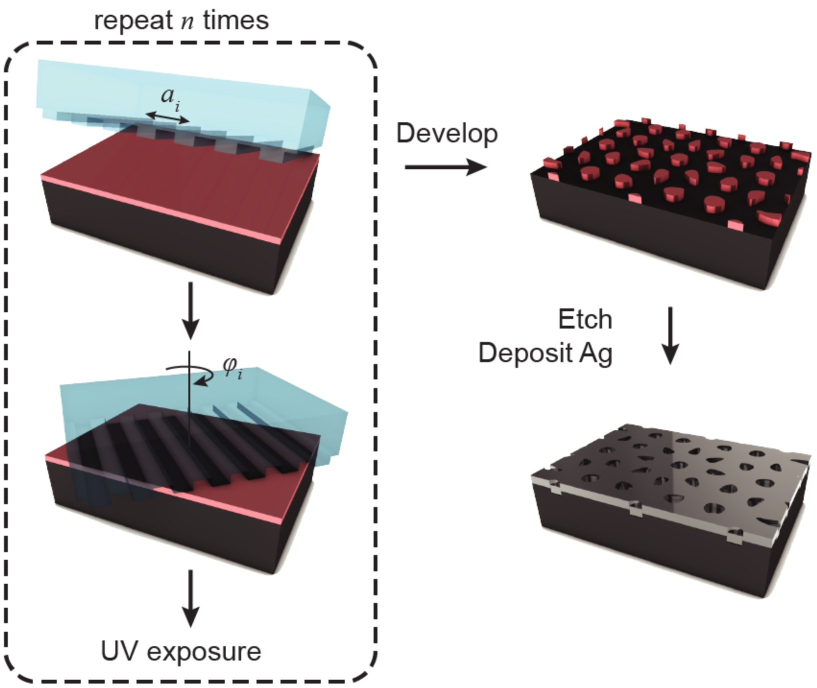


Figure 2.1: Fabrication scheme for moiré nanolithography. PDMS masks with different periodicities a_i and azimuthal angles φ_i are put into conformal contact with a PR-coated Si wafer and exposed to UV light. This process is repeated for the desired number of exposures. The resulting pattern is developed, then transferred to a Ag plasmonic crystal.

a PDMS mask with periodicity a_i and a relative azimuthal angle φ_i. Each UV exposure was approximately $1/n$ × the standard exposure time, though the optimal timing was determined empirically. After the multiple exposures, the pattern was developed and a thin (~10 nm) Cr layer was deposited followed by PR lift-off. Using the Cr as an etch mask, 50-nm pits were etched into the Si, and the Cr was removed to reveal a bare Si template. A silver film (160 nm) was deposited on the template to form the PC. The relatively thick layer of Ag was chosen to ensure it was optically opaque, and SPPs would only be excited on the top (Ag/air) interface, and not on the backside (Ag/Si interface).

Three types of quasiperiodic patterns were fabricated by MNL: high-symmetry quasicrystal (single periodicity with 10-fold rotational symmetry, **Figure 2.2a**), low-symmetry pattern with non-uniform rotational symmetry (single periodicity, **Figure 2.2b**), and low-symmetry pattern with multiple periodicities (6-fold rotational symmetry, **Figure 2.2c**). The top row of **Figure 2.2** shows scanning electron microscopy (SEM) images of the fabricated structures. All patterns are quasiperiodic in nature; that is, they do not possess translational symmetry, but show rotational symmetry. While the high-symmetry quasicrystal (**Figure 2.2a**) has 10-fold rotational symmetry, the other two patterns have at most 2-fold rotational symmetry on any given symmetry axis (though they contain multiple symmetry axes). Simulated real-space images are shown below the SEM images and agree well with the experimental structures. The bottom row of images are reciprocal space representations of the quasicrystals calculated by performing a Fourier transform (FT) on the real-space SEM images, and clearly reveal the rotational symmetry of each lattice.

The simulated real-space images of the different patterns were obtained by modeling MNL. First, the intensity profile of each partial exposure was identified using finite-difference time-

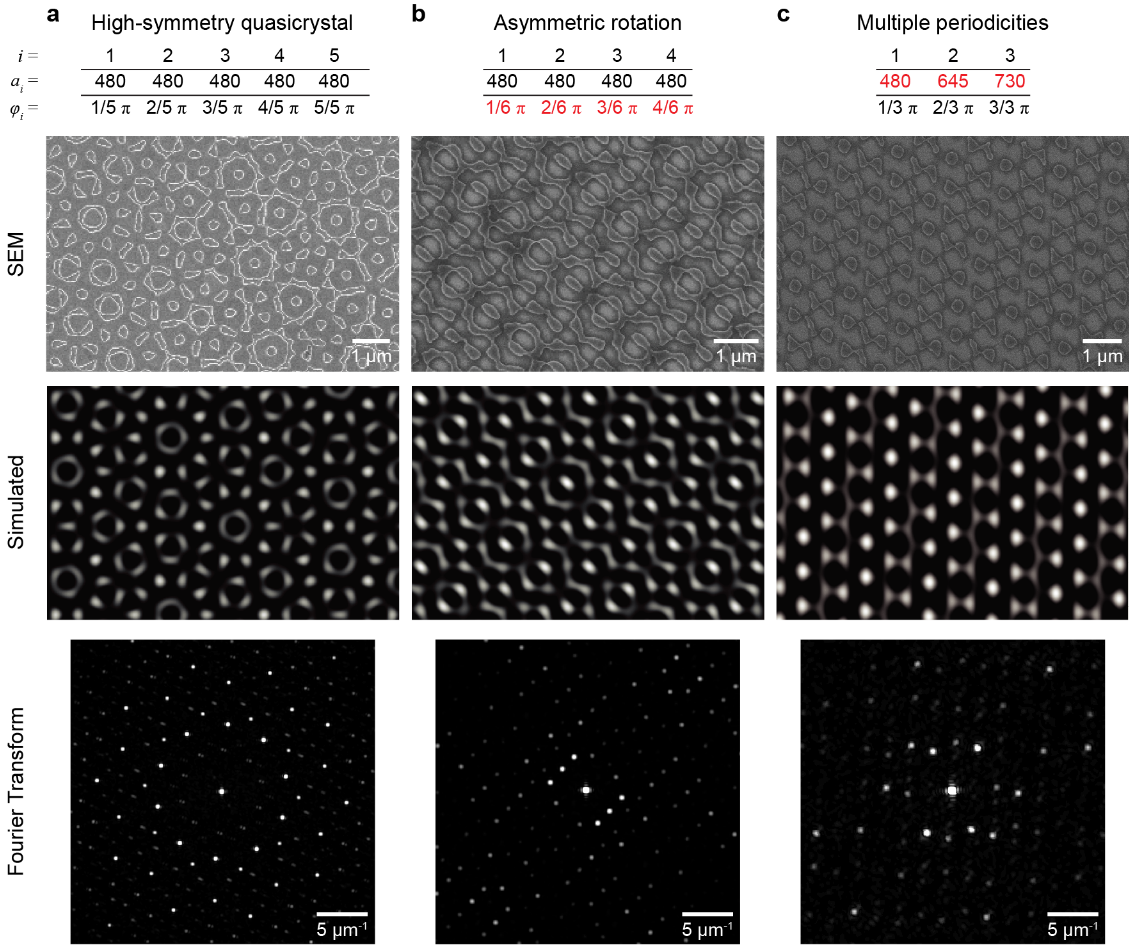


Figure 2.2: Quasiperiodic moiré patterns. (a) High symmetry quasicrystals are fabricated with by 5 exposures with PDMS masks having equal periodicities a_0 = 480 nm equiangular azimuthal rotations ($\Delta\varphi = \pi/5 = 36°$) (b) Rotationally asymmetric patterns with asymmetric exposure angles fabricated by 4 exposures with equal periodicities a_0 = 480 nm but all angles are within a single quadrant ($\varphi_4 - \varphi_1 = \pi/2 = 90°$). (c) Multiperiodic patterns fabricated by 3 exposures with different periodicities a_1 = 480 nm, a_2 = 645 nm, a_3 = 730 nm, and equiangular rotations ($\Delta\varphi = \pi/3 = 60°$). (a-c) SEM images (top) of the Si template or PR pattern agree with simulated structures (middle). FT of the SEM images reveal the reciprocal lattice of the patterns (bottom).

domain (FDTD) simulations (**Figure 2.3a**). A 2D simulation environment was used with periodic boundaries in the *x*-direction and perfectly matched layers (PML) in the *y*-direction to simulate an infinite array of lines. The simulated PDMS mask had 400-nm periodicity with 50% duty cycle (*i.e.* 200-nm features) and a feature height of 200 nm. These parameters were chosen to replicate closely the dimensions of experimental masks. The simulated mask was placed above regions representing a 120-nm layer of photoresist and a Si substrate. A plane-wave light source at a single wavelength $\lambda = 405$ nm illuminated the material stack from above. Finally, a 2D monitor recorded the electric field intensity within the PR layer, as overlaid on the PR region in **Figure 2.3a**. These phase-shifting masks are known to create periodic intensity profiles,[78] and the intensity cross-section through the center of the photoresist (dotted white line) shows a near-sinusoidal line shape (**Figure 2.3b**). The simulated intensity profile shows a slightly narrower region of high intensity compared to the perfectly symmetric cosine wave for this specific geometry. The mean-squared error between the two curves is 0.016.

Based on the exposure intensity profile for phase-shifting photolithography for a line-array mask, we developed a computational tool in MATLAB to simulate the exposures in photoresist for MNL (**Figure 2.4** — full code available in **Appendix XX**). This tool was designed to test how various MNL parameters would affect final geometric structures, and to verify the geometry after fabrication. The available parameters for each exposure are: the geometry of the PDMS stamp (line, square, or hexagonal), its periodicity a_0, and the azimuthal rotation angle φ. Each line exposure is assumed to have a sinusoidal profile, which closely approximates experimental conditions. The square and hexagonal options input two lines offset by a 90° rotation, or three lines offset by 60° rotations, respectively. When a moiré structure is generated, the final tuning

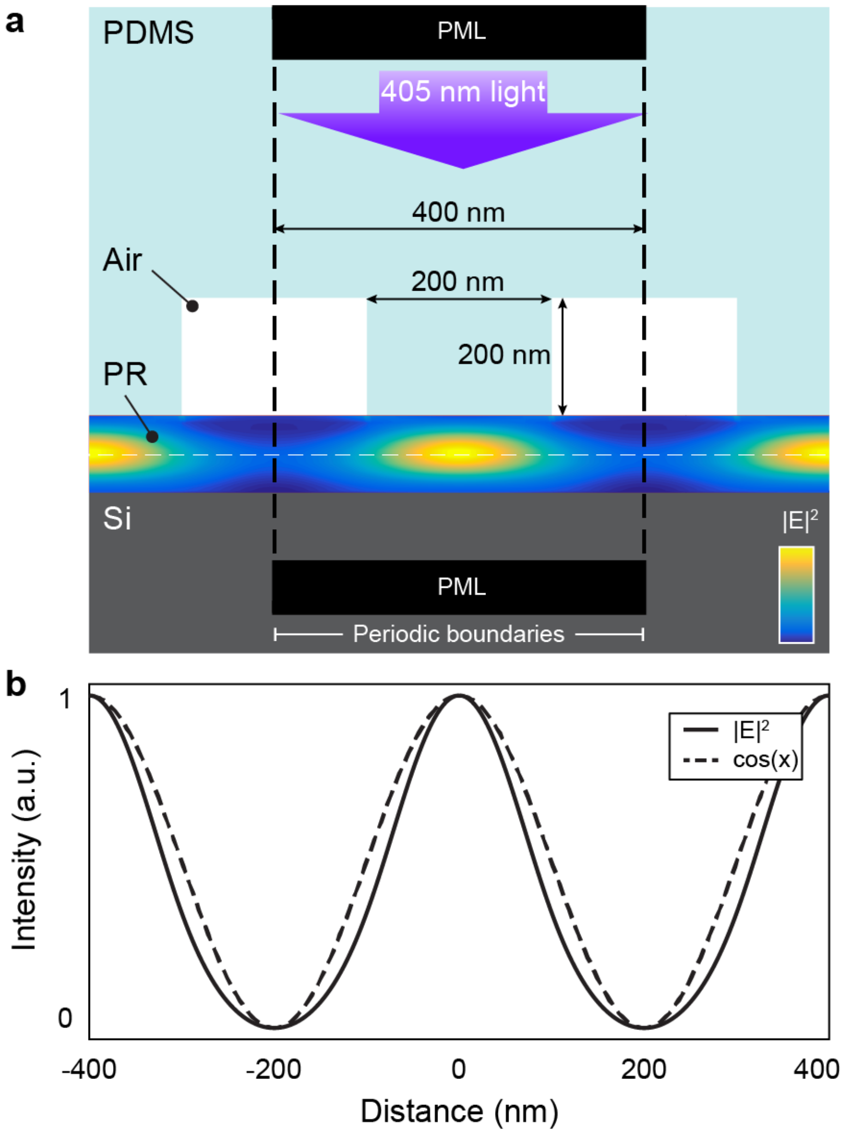


Figure 2.3: FDTD Simulation of phase-shifting photolithography. (a) Simulation scheme with electric field data from the 2D monitor overlaid onto the PR slab. 1D cross-section monitor is located in the center of the PR (white dashed line). (b) Cross sectional intensity from the 1D monitor in (a) compared to a sinusoidal profile.

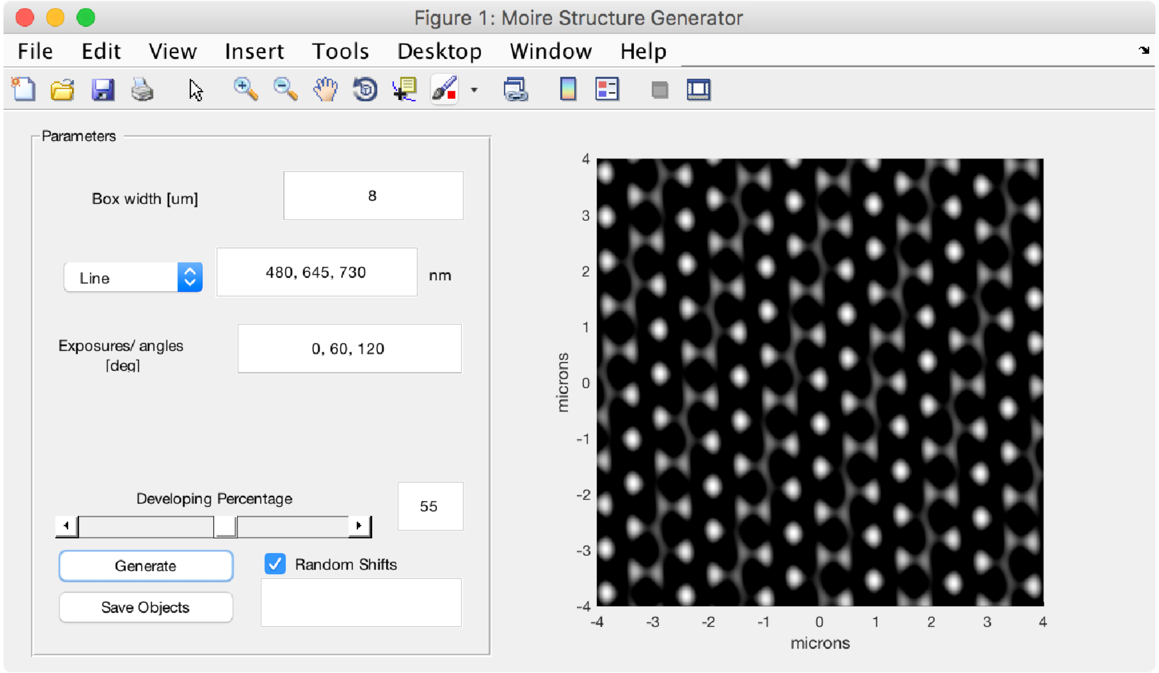


Figure 1: Moire Structure Generator

Box width [um] 8

Line 480, 645, 730 nm

Exposures/ angles [deg] 0, 60, 120

Developing Percentage 55

☑ Random Shifts

Figure 2.4: Graphical user interface (GUI) for the MATLAB computational program to predict the resultant patterns from MNL. Parameters panel (left) indicate where values are input into the program and the structure is displayed in the axis (right). The displayed pattern is from the multiperiodic quasicrystal pattern.

parameter is the developing percentage. Since the optical intensity in the PR layer varies continuously instead of with a binary on/off profile as in traditional photolithography, the developing time can be tuned to adjust the overall density of patterned features. The simulated SEM images in **Figure 2.2** were created using this tool, and the displayed parameters in **Figure 2.4** were used to generate the simulated structure for the varying periodicity quasicrystal (**Figure 2.2c**).

2.2.2 SPP excitations in high-symmetry 10-fold quasiperiodic PCs

We described quasiperiodic PCs by their reciprocal lattices generated by the FT of their SEM images. To determine the dispersive behavior of excited SPP modes, we used the Bragg coupling condition:

$$\mathbf{k}_{\mathrm{SPP}} = \mathbf{k}_0 \sin\theta - \mathbf{G} \tag{2.1}$$

where $\mathbf{k}_{\mathrm{SPP}}$ is the momentum of the SPP mode, $\mathbf{k}_0$ is the momentum of free space light incident on the surface at angle θ, and **G** is the grating vector. **G** is related to the FT of a PC and therefore its overall geometry by

$$\mathbf{G} = \frac{2\pi}{a_0}\mathbf{k} \tag{2.2}$$

For 2D periodic patterns, $\boldsymbol{k} = c_1\boldsymbol{k}_x + c_2\boldsymbol{k}_y$, a linear combination of the 2D basis vectors in reciprocal space: $\boldsymbol{k}_x$ and $\boldsymbol{k}_y$. The Fourier transform of the 10-fold quasiperiodic lattice, however, revealed 5 basis vectors (**Figure 2.5a**). We redefined $\boldsymbol{k} = c_1\boldsymbol{k}_1 + c_2\boldsymbol{k}_2 + c_3\boldsymbol{k}_3 + c_4\boldsymbol{k}_4 + c_5\boldsymbol{k}_5$, where c_1, c_2, c_3, c_4, and c_5 are integers. By defining five basis vectors, the scattering order m of each mode, indexed as $[c_1\ c_2\ c_3\ c_4\ c_5]$, can be determined from the sum of the absolute values of each constant (*i.e.* $m = |c_1| + |c_2| + |c_3| + |c_4| + |c_5|$). Notably, the reciprocal lattice vectors of quasiperiodic structures exhibit characteristics different from periodic arrays. For example, the vector magnitudes of

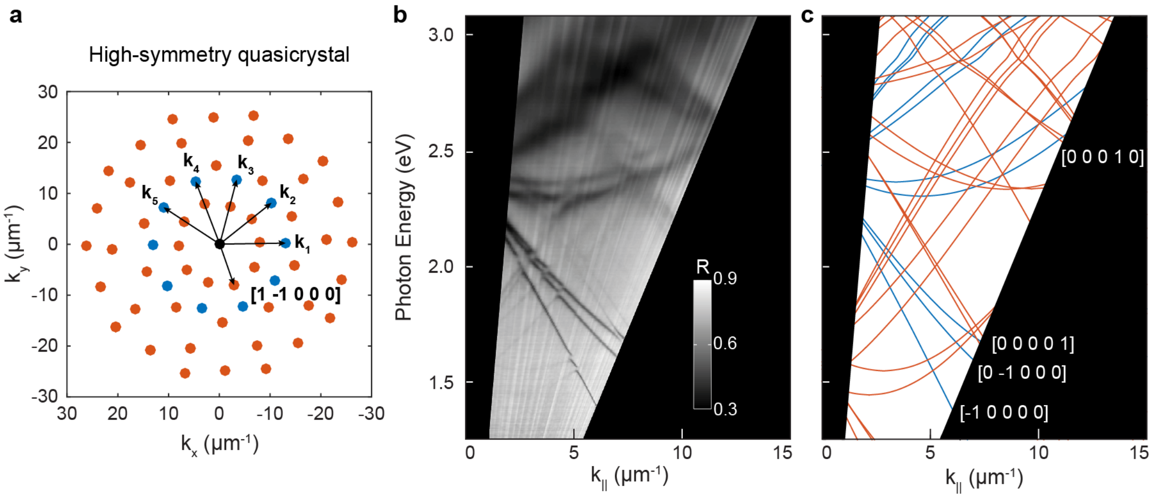


Figure 2.5: Reflection spectra from high-symmetry quasiperiodic PC. (a) Calculated FT with basis vectors $\mathbf{k}_i$ shown. (b) Angle-resolved reflection spectra for the 10-fold quasiperiodic pattern taken from $\theta = 10°$ to $60°$ converted to dispersion diagram. (c) Predicted SPP mode dispersion calculated from the Bragg coupling equation for a Ag/air interface with some first order modes labeled. In (a,c) blue objects represent first-order modes and orange objects represent second-order modes.

higher-order reciprocal vectors can be *less* than those of lower-order vectors.[79] In **Figure 2.5a**, [1 -1 0 0 0] is a second-order reciprocal vector; however, its magnitude is less than that of the first-order [1 0 0 0 0] family. This property of quasiperiodic lattices leads to a denser concentration of reciprocal vectors than in periodic lattices.

Angle-resolved reflectance spectroscopy revealed the propagation of SPP modes associated with the 10-fold lattices (**Figure 2.5b**). Dispersion diagrams were constructed by stitching together a series of angle-resolved spectra and then converting them to energy and in-plane momentum to depict the band structures of the PCs. **Figure 2.5c** shows SPP modes on the 10-fold PC at the Ag/air interface, where minima in intensity of the reflected light indicate the excitation of plasmon modes. The strongest modes visible in the dispersion diagram correspond to first order (m = 1) modes, and several faint second order modes are also present. In several cases, most noticeable at a photon energy of 2.0 eV, plasmonic band gaps form at the intersections between first and second order SPP modes.

While the dispersion diagram appears complex, the dispersion of SPP modes was understood by considering the FT of a given PC of any symmetry. A scheme of the angle-resolved spectroscopy measurement shows incident light with p-polarization and momentum k incident on a quasiperiodic PC (**Figure 2.6a**). The parallel component of the momentum of the photon $k_{||}$ is displayed along the x-axis of the dispersion diagram, while the photon energy $E \propto |k|$ is displayed on the y-axis. At $k_{||} = 0$, when incident light is normal to the surface, several families of degenerate SPP modes were identified (**Figure 2.6b**). The first order modes are all described by the [1 0 0 0 0] family, and five families of second order modes exist for the 10-fold PC. Each family was grouped by the angular difference of its constituent first-order vectors. For example, the ring of modes with

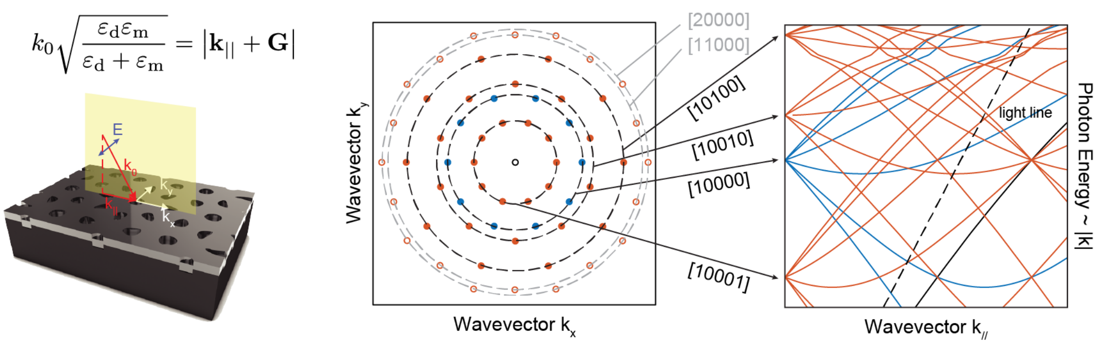


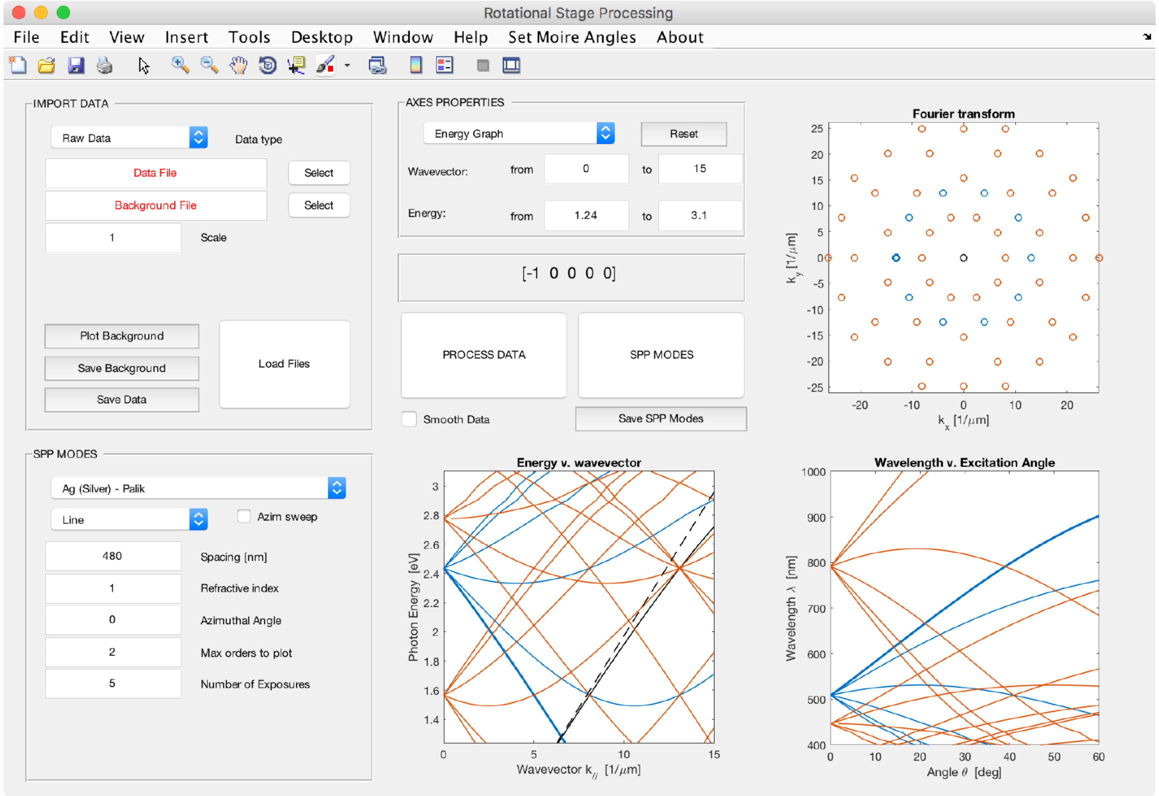
Figure 2.6: Measurement scheme for plasmonic crystals. (left) Bragg coupling equation and incident light path for reflection spectra. (right) Relationship between FT and SPP dispersion for a 10-fold high-symmetry quasicrystal. First- (blue) and second- (orange) order mode families are indexed. Modes with energies beyond the measurement range are faded.

smallest radius (lowest energy on the dispersion diagram) contains second order modes where the two components are separated by 4π/5 radians, such as [1 0 0 0 1] or [-1 1 0 0 0].

Since quasiperiodic PCs with high rotational symmetry have several times the number of SPP modes compared to their periodic counterparts, indexing them and teasing out the origin of plasmonic band gaps provides a challenge. To improve this process, a second computational tool was developed, also built in MATLAB (**Figure 2.7 – Full code in Appendix A:**). This tool was used to (1) process angle-resolved spectra and create dispersion diagrams, (2) display theoretical SPP modes calculated by the Bragg coupling condition, and (3) identify intersections of indexed SPP modes to analyze the formation of plasmonic band gaps. The top left panel is used to import experimental data, collected as a function of wavelength and input angle, and convert them to functions of photon energy and wave vector. The dispersion diagram and raw data are displayed in the bottom center and bottom right axes, respectively. The bottom left panel accepts input variables to the Bragg coupling equation, then overlays the expected SPP modes onto the dispersion diagram and wavelength–angle axes. SPP modes at the Ag/air (n = 1) interface for a 10-fold symmetry lattice with periodicity a_0 = 480 nm are currently displayed. Additionally, the expected FT is displayed in the top right axis. Selecting any SPP mode would display its index [c_1 … c_n] in the center of the window so that intersections between different modes can be identified. For example, the [-1 0 0 0 0] mode of the 10-fold PC is currently selected in **Figure 2.7** and displayed on bold on the dispersion diagram and simulated FT.

2.2.3 Reducing SPP degeneracy with low-symmetry quasiperiodic lattices

One method to reduce the symmetry in quasiperiodic lattices is by decreasing the angular degeneracy. We analyzed the angle-resolved reflection spectra of the quasiperiodic pattern with



Raw Data — Data type

1 — Scale

Energy Graph

Wavevector: from 0 to 15

Energy: from 1.24 to 3.1

[-1 0 0 0 0]

Ag (Silver) - Palik

Line

480 — Spacing [nm]

1 — Refractive index

0 — Azimuthal Angle

2 — Max orders to plot

5 — Number of Exposures

Figure 2.7: GUI for MATLAB program to process reflection spectra and calculate SPP mode dispersion. Import data panel is where the raw reflection spectra are input to the program. SPP modes panel is used to provide parameters to the Bragg coupling equation for SPP mode calculation. Axes (bottom, center and right) display spectra and modes in E–**k** or *λ*–*θ* format. Fourier transform axes (top right) displays the reciprocal lattice of the given structure. First- and second-order modes are colored blue and orange respectively.

asymmetric angles (**Figure 2.8**). The simulated FT reflected the angle preference of the lattice towards 45°, though the first order SPP modes are still distributed equally across the entire spectrum, similar to the high-symmetry quasicrystal (**Figure 2.8b**). The direction of $\boldsymbol{k}_{||}$ in equation (2.1) is aligned to the x-axis and therefore mirror symmetry across this axis is required when calculating the excited SPP modes. The FT of the lattice reveals that for positive $\boldsymbol{k}_1$, $\boldsymbol{k}_2$, and $\boldsymbol{k}_3$, $\boldsymbol{k}_x$ is positive, while for negative $\boldsymbol{k}_1$, $\boldsymbol{k}_2$, and $\boldsymbol{k}_3$, $\boldsymbol{k}_x$ is negative (**Figure 2.8a**). Therefore, the rotational degeneracy of the SPP modes was removed with the exception of $\pm\, \boldsymbol{k}_4$, where $\boldsymbol{k}_x = 0$.

2.2.4 Multi-periodic quasicrystals and a general Bragg model

Removing the angular symmetry from the SPP dispersion preserved the broad spectral range of first-order SPP excitations at large wavevector, but near normal incidence ($\boldsymbol{k}_{||} = 0$) the SPP modes remained degenerate as illustrated in **Figure 2.6b** due to the single periodicity of the PDMS mask used to fabricate the pattern. To remove this final degeneracy, we fabricated quasiperiodic patterns with three (symmetric) exposure angles, but with PDMS masks containing different periodicities a_i (**Figure 2.2c**). The FT reflects the varying periodicities as different magnitudes of primary vectors $\boldsymbol{k}_1$, $\boldsymbol{k}_2$, and $\boldsymbol{k}_3$ (**Figure 2.9a**). With the removal of periodicity and azimuthal symmetry in these quasicrystal patterns, we developed a general form of grating vector **G** for the Bragg coupling condition.

$$\mathbf{G} = \sum_{i=1}^{n} c_i \mathbf{k}_i \tag{2.3}$$

$$\mathbf{k}_i = \frac{2\pi}{a_i}\left(\cos\varphi_i \hat{\mathbf{k}}_x + \sin\varphi_i \hat{\mathbf{k}}_y\right) \tag{2.4}$$

In the general model, each basis vector $\boldsymbol{k}_i$ includes unique periodicity a_i and azimuthal angle φ_i terms. Importantly, we established a direct link between **G** and the MNL process, where each set

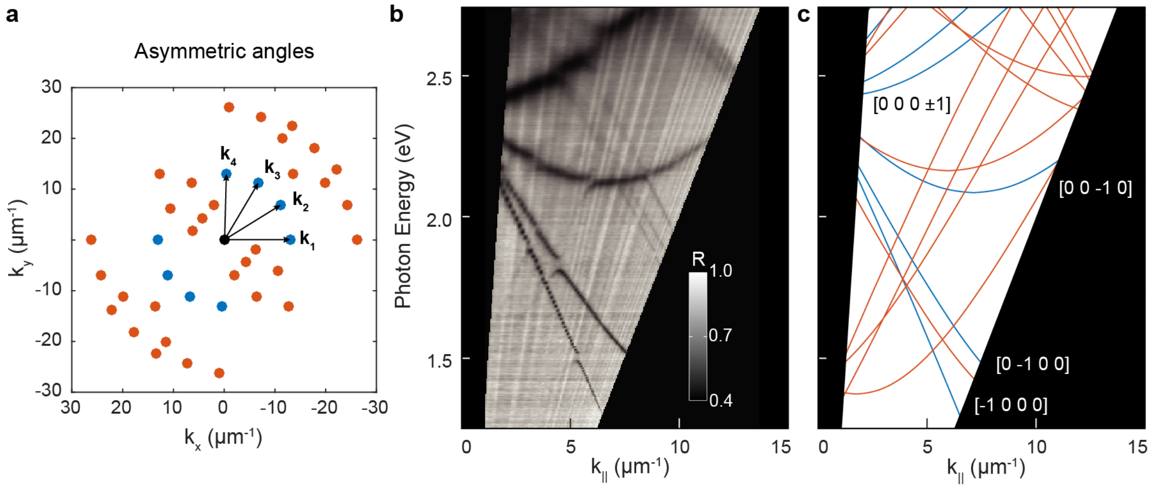


Figure 2.8: Reflection spectra from quasiperiodic PC with asymmetric rotation angles. (a) Calculated FT with basis vectors $\mathbf{k}_i$ shown. (b) Angle-resolved reflection spectra taken from θ = 10° to 80° converted to dispersion diagram. (c) Predicted SPP mode dispersion calculated from the Bragg coupling equation for a Ag/air interface with some first order modes labeled. In (a,c) blue objects represent first-order modes and orange objects represent second-order modes.

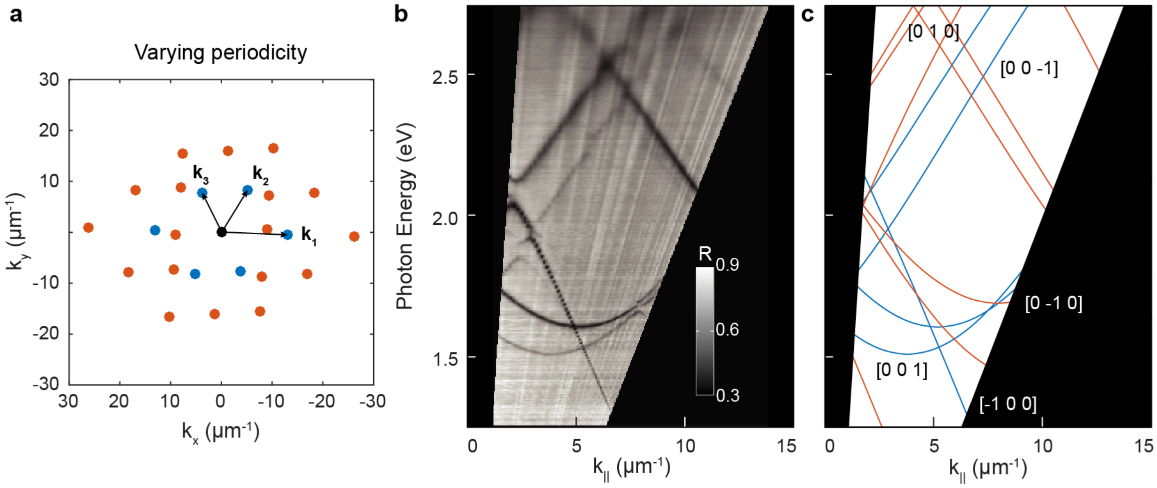


Figure 2.9: Reflection spectra from a multiperiodic patterned PC. (a) Calculated FT with basis vectors $\mathbf{k}_i$ shown. (b) Angle-resolved reflection spectra taken from $\theta = 10°$ to $80°$ converted to dispersion diagram. (c) Predicted SPP mode dispersion calculated from the Bragg coupling equation for a Ag/air interface with some first order modes labeled. In (a,c) blue objects represent first-order modes and orange objects represent second-order modes.

of (a_i, φ_i) directly corresponds to one MNL mask exposure. First-order SPP modes were observed over a wide range of energies, and intersecting at nonzero wavevector (**Figure 2.9b**). A similar indexing scheme was used to describe the SPP modes, and the predicted dispersion using equation (2.3) agrees with the experimentally spectra (**Figure 2.9c**). The [-1 0 0] mode based on the 480-nm periodicity shows the same dispersion as in the previous quasiperiodic lattices. The [0 ±1 0] and [0 0 ±1] modes, however, are at lower energies due to the increase in magnitudes of $\boldsymbol{k}_2$, and $\boldsymbol{k}_3$ respectively. While, in general, the second order (m = 2) modes were weaker than in previous structures, several band gaps were observed at intersections between them and first-order modes.

2.2.5 Plasmonic band gaps in low-symmetry PCs

A closer examination of the dispersion diagram of the multiperiodic quasicrystalline PC revealed the origin of several band gaps (**Figure 2.10**). We demonstrated that the origin of plasmonic band gaps at first and second-order Bragg lines is not only true for high-symmetry quasicrystals,[57] but also applies to other, low-symmetry patterns. Band gaps formed at **I** and **II** are second-order crossings between two first-order modes. In these cases, the strong [-1 0 0] mode from the 480-nm periodicity formed band gaps with the first-order modes from each of the other two periodicities. These band gaps opened at the (-1 1 0) and (-1 0 -1) Bragg lines respectively. Band gap **III** occurred at the first order Bragg line (-1 0 0) between the [0 0 -1] and [-1 0 -1] modes. This gap was the largest, likely due to the strength of [-1 0 0] mode, where the Bragg reflection takes place. The large gap for **I**, even though it occurred at a second order Bragg line, was likely enhanced due to the intersection between the [0 1 -1] and [0 1 0] modes (not shown) at the same E-k position. Finally, a second order band gap between the first-order [0 0 1] and unobservable third-order [0 -1 2] modes took place at **IV**. Again, the intersection of several SPP modes at similar

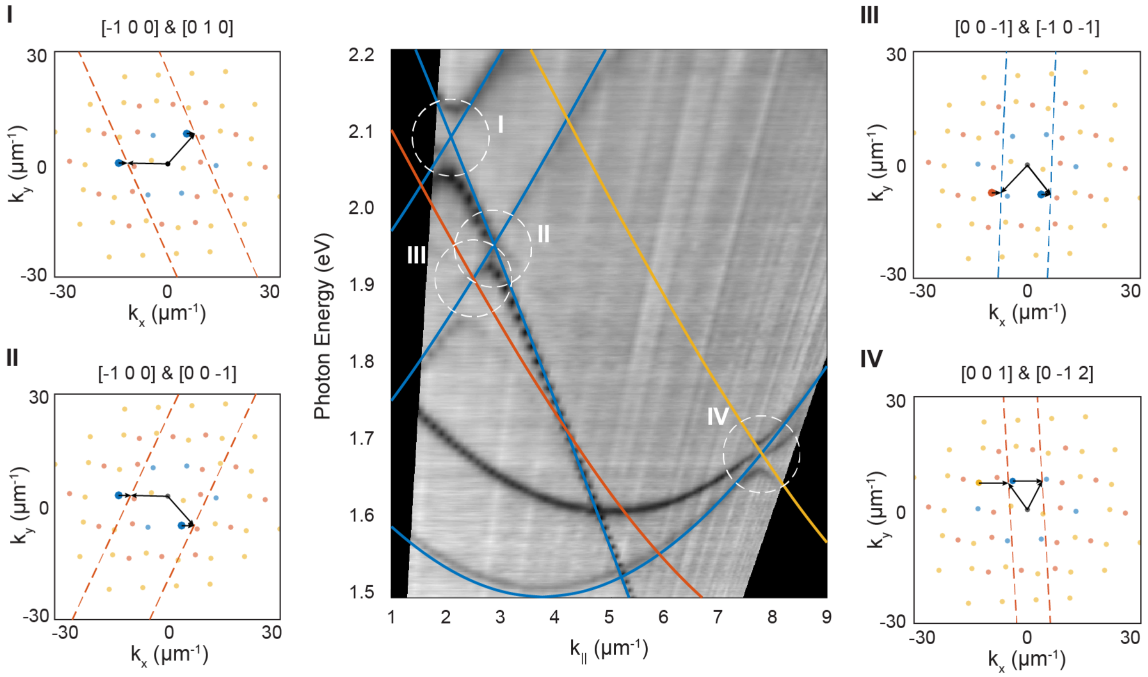


Figure 2.10: Plasmonic band gaps in multiperiodic patterned PC. (center) Zoom of dispersion diagram with overlaid relevant SPP modes. Numbered circles highlight plasmonic band gaps. (left, right) FT of the pattern with modes and Bragg lines highlighted for specific band gaps. Arrows represent direction and magnitude of $\boldsymbol{k}_{||}$ and $\boldsymbol{k}_{\text{SPP}}$ for the given E–**k** coordinates.

E-k coordinates likely enhanced the size of band gap, since the third-order mode was relatively weak.

2.3 Experimental methods

2.3.1 Lithography procedure for MNL

To create quasicrystalline moiré patterns, a PDMS photomask with 1D lines was placed into conformal contact with positive-tone, g-line photoresist (Shipley S1805) diluted with poly(ethylene glycol) methyl ethyl acrylate in a 1:2 ratio by volume (~120 nm thick) on a Si [100] wafer and exposed to UV light in a home-built narrow-band LED light source.[80] After the exposure through the PDMS photomask, additional exposures were done with the same or alternate photomasks at various azimuthal rotation angles. The alignment of the PDMS masks was typically within 2° of the target angle. The exposed PR was developed (1:5 dilution of Microposit 351 developer) and resulted in arrays of PR posts on a Si substrate.

2.3.2 FDTD simulations of phase-shifting lithography

FDTD simulations were performed with commercial software (FDTD Solutions, Lumerical Inc.). A 2D simulation environment was used with periodic boundaries in the x-direction and PML boundaries in the y-direction. The PDMS material was represented as a uniform dielectric constant of $n = 1.4$. The refractive index of the PR was modeled as a Cauchy material with coefficients provided by the manufacturer. Neither the absorption of PR nor a change in refractive index with exposure to UV light were considered. A mesh accuracy level of 5 was used for the simulation. A 2D monitor recorded the electric field in the entire PR region, and a 1D monitor recorded the electric field only through the center of the PR. A plane wave source with free-space wavelength $\lambda = 405$ nm and polarization in the x-direction illuminated the sample from within the PDMS layer.

2.3.3 Pattern transfer to quasiperiodic PCs

To clear away residual photoresist, the samples were exposed to an O_2 plasma in an RIE (50 sccm, 300 mtorr, 30 watts) for ~15 s. A 10-nm Cr sacrificial layer was deposited onto the substrate through line-of-sight physical vapor electron beam deposition with a Kurt J. Lesker PVD 75. The PR posts were lifted off with Microposit Remover 1165 leaving holes of bare Si in the Cr layer. Trenches with depths of ~50 nm were etched anisotropically using a C_4F_8/SF_6 co-flow recipe with a STS LpX Pegasus Deep Reactive Ion Etcher. The Cr layer was removed with Cr etchant (Transene), and 160 nm of Ag were deposited in the PVD-75 onto the Si grating to create the plasmonic crystals.

2.3.4 Angle-resolved reflectance spectroscopy

Zero-order reflectance spectra were collected from $\theta = 10°$ to 60 or 80° in 1° or 0.5° increments using an automated, self-designed LightField software add-in (Princeton Instruments). Collimated, unpolarized white light from a halogen lamp (100 W) illuminated the sample with a spot size of 2 mm. The reflected light was coupled into a bundled optical fiber connected to a Princeton Instruments Acton SP2500 spectrometer with a PIXIS:400 CCD detector. A linear interpolation algorithm in Matlab converted the measured optical data — wavelength (λ) and excitation angle (θ) — into dispersion diagrams — photon energy (E_{photon}) and in-plane wavevector of light ($\boldsymbol{k}_{||}$) — using: $E_{photon} = hc/\lambda$ and $\boldsymbol{k}_{||} = (2\pi/\lambda) \sin \theta$.

2.4 Summary

We have shown the fabrication of high- and low-symmetry moiré plasmonic crystals and analyzed the excitation of SPP modes on their surface. We modeled the nanofabrication technique MNL, and developed a computational tool to visualize the resultant quasiperiodic patterns. We developed a general form of the Bragg coupling equation that incorporates experimental inputs from MNL and can describe the SPP modes on quasiperiodic PCs. Finally, we developed a general indexing scheme for the SPP modes and identified the origin of plasmonic band gaps at first- and second-order Bragg lines in low symmetry quasicrystalline patterns.

CHAPTER 3:

SURFACE PLASMON POLARITONS ON MULTISCALE GRATINGS

3.1 Background

Surface plasmons can be excited on the surface of a metal in the form of surface plasmon polaritons (SPPs), where incident light is trapped at the metal-dielectric interface through the assistance of submicron periodic patterns.[8, 18] A momentum mismatch between free-space light and surface plasmons can be overcome by introducing this submicron periodic pattern to the metal surface. The SPP resonance wavelength is then dictated by the periodicity of the surface pattern.[55] SPPs typically show narrow resonances due to the strong confinement of localized electric fields. This localization can be beneficial for applications in photovoltaics,[1, 81] refractive-index-based biosensing,[4, 6] and plasmonic lasing.[11, 69] One challenge is to maintain narrow resonances, while covering a large portion of the visible light spectrum. Since periodic nanopatterning is required for SPP excitation, one approach to expand the available wavelength range is to change the symmetry of the pattern. Low-symmetry lattices,[70] quasicrystal geometries,[57, 82] and superlattices[83-84] have been shown to increase the number of available SPP modes while maintaining individual narrow resonances.

Photonic and plasmonic superlattices contain hierarchical periodicities where the primary periodicity is near the wavelength of light (*e.g.* 400 nm), and the superperiodicity is larger by at least an order of magnitude (*e.g.* 10 μm). Nanowire superlattices have been studied to determine the effect of the hierarchical periodicity on the excited optical modes.[85] These superlattices, fabricated by e-beam lithography (EBL), were commensurate in nature, *i.e.* each super period contained the same number and arrangement of sub-units, resulting in the conclusion that the primary periodicity selected certain high-order modes of the superperiodicity to be excited. Due to the limitations of EBL, these patterns covered small areas (100 × 100 μm^2), restricting the optical measurements to be done (*e.g.* off-angle reflection). A second variety of superlattices fabricated by multiscale patterning[44] are incommensurate in nature, yet appeared to show similar optical properties.[83, 86] The SPP modes excited in these multiscale arrays showed a strong resonance related to the primary periodicity and so-called satellite modes influenced by the superperiodicity. While the satellite modes are based on SPPs folded into the Brillouin zones of the superperiodicity, the explicit relationship between the primary and superperiodicities and its effect on SPPs is not fully understood.

Here we show fabrication of multiscale gratings that support SPPs over a broad range in the visible spectrum. We develop and indexing model to relate the primary and superperiodicity SPP modes based on the reciprocal-space representation of the arrays. To validate the model, we systematically show the influence of primary and superperiodicity on the excited SPP modes. We identify that the origin of excited SPPs in multiscale arrays in fundamentally different from those in commensurate superlattices. Finally, we expand multiscale patterning with a modified litho-etch-litho-etch (LELE) technique to create multiscale grating with two independent primary periodicities and analyze the SPP modes on these patterns.

3.2 Results and Discussion

3.2.1 Fabrication of multiscale arrays

Multiscale gratings were fabricated over large areas (2 cm × 2 cm) with a double-exposure lithography + etch (LE) technique (**Figure 3.1**). First, multiscale arrays of photoresist (PR) lines were fabricated on a Si wafer by photolithography with two exposures and development followed by Si etch and Ag deposition. The first exposure was through a PDMS phase-shifting mask[38, 87] with primary periodicity a_0, and the second exposure was through a standard contact mask with

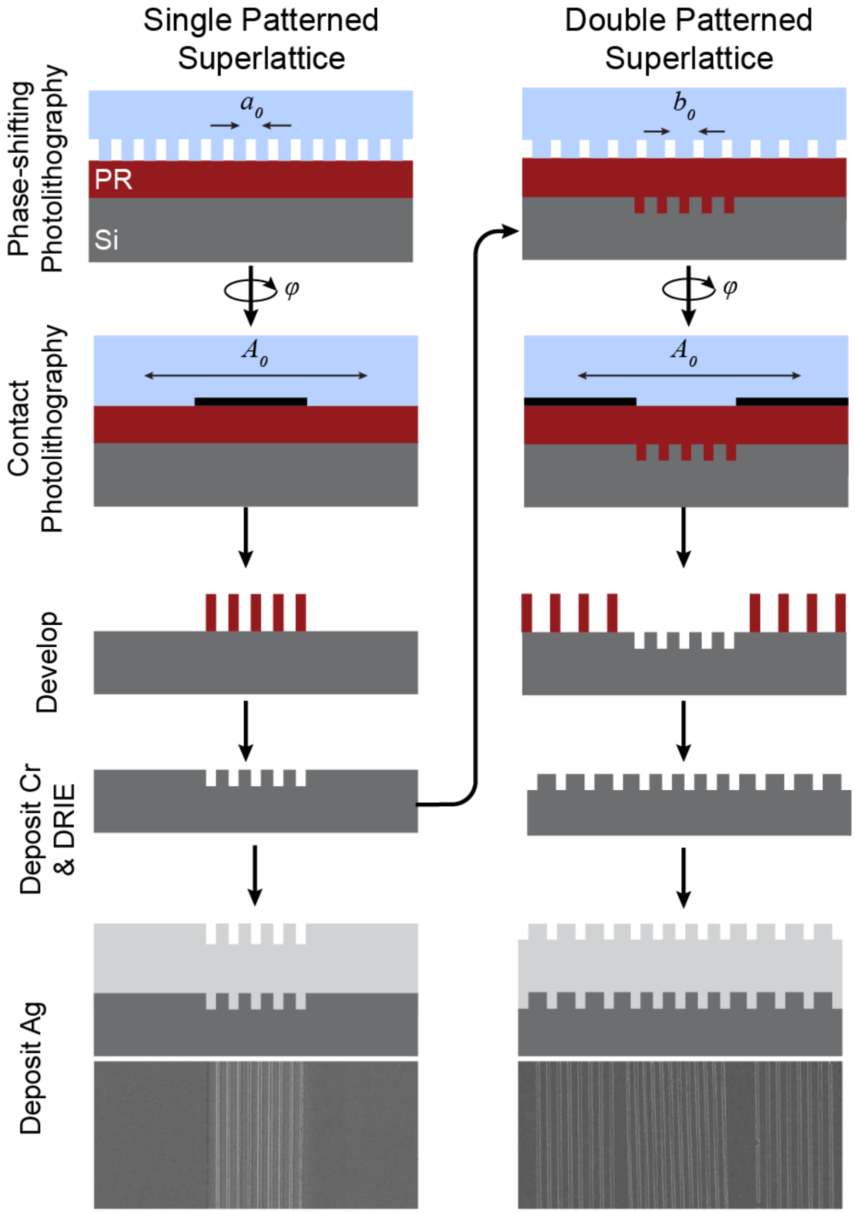

Single Patterned Superlattice
Double Patterned Superlattice
Phase-shifting Photolithography
Contact Photolithography
Develop
Deposit Cr & DRIE
Deposit Ag
a_0
b_0
PR
Si
φ
A_0

Figure 3.1: Fabrication scheme for multiscale superlattices. Multiscale superlattices use two consecutive exposures of phase shifting lithography and traditional contact lithography for the primary and superperiodicities respectively. After the first round of etching, the process can be repeated to fabricate doubly patterned superlattices.

superperiodicity A_0 and an azimuthal rotation of φ relative to the first mask. After development, a thin layer of Cr was deposited as an etch mask, and the pattern was transferred to 50-nm deep grooves in Si. Removal of the Cr layer left a bare Si template. A second iteration of LE can be done on this existing template to create double-patterned superlattices with two different primary periodicities, which will be discussed in detail later in the chapter. Finally, 160 nm of Ag was deposited on the Si template to create the plasmonic crystal. The relatively thick layer of Ag was chosen to ensure it was optically opaque, and excited SPPs would only exist on the top (Ag/air) interface, not on the backside (Ag/Si interface). Importantly, since the two exposures are independent of each other, the resulting superlattice is incommensurate except in the case where A_0 is an integer multiple of a_0 and perfect alignment yields $\varphi = 0°$. Each of the two periodicities, however, still has long range order as dictated by the initial lithography masks.

This process was used to create a multiscale grating with primary periodicity $a_0 = 400$ nm and superperiodicity $A_0 = 10$ μm (**Figure 3.2a**). The duty cycle of the superperiodicity lithography-mask was 50% resulting in patterned regions with the primary periodicity and unpatterned regions having equal areas. A Fourier transform (FT) was performed on this scanning electron microscopy (SEM) image to generate the reciprocal space representation of the structure (**Figure 3.2b**). The FT shows two three equally spaced groupings of peaks, where the peaks within each group also have a regular, smaller spacing. Reciprocal lattices are used to identify plasmonic crystals such that each peak in the FT can be represented by a grating vector **G**:

$$\mathbf{G} = \sum_{i=1}^{n} c_i \mathbf{k}_i \tag{3.1}$$

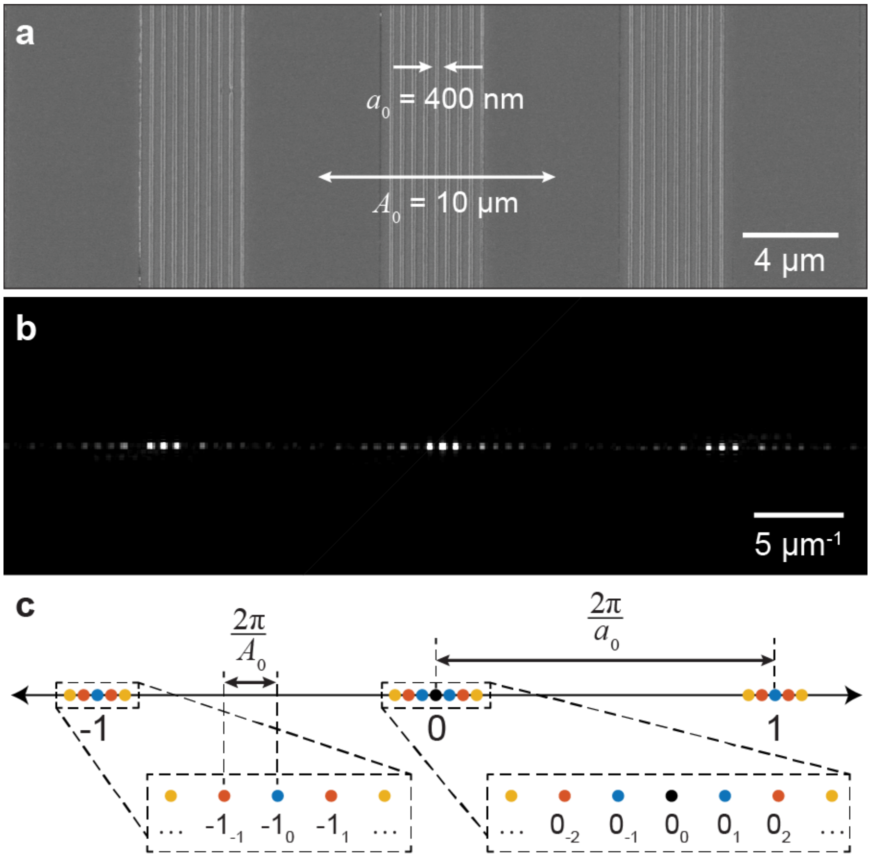


Figure 3.2: 1D Superlattice. A SEM (a) of a 1D plasmonic superlattice and (b) its Fourier transform. (c) The Bragg peaks in the Fourier transform can be indexed as combinations of the primary periodicity (a_0) and superperiodicity (A_0)

where $|\mathbf{k}| = 2\pi / a_0$ describes the periodicity of the lattice. In the case of plasmonic superlattices, we define **G** as

$$\mathbf{G} = i\,\mathbf{k} + J\,\mathbf{K} \tag{3.2}$$

where i and J are integers, $|\mathbf{k}| = 2\pi / a_0$ and $|\mathbf{K}| = 2\pi / A_0$. For the discussion of multiscale gratings, we use the notation that lower-case letters represent a property of the primary periodicity, while upper-case letters are related to the superperiodicity. Based on the equation for **G**, we were able to identify and index each of the peaks in the reciprocal lattice. We indexed each of the peaks with the notation i_J (*e.g.* -1_1) and defined the order of each mode $m=|i|+|J|$ — *e.g.* -1_1 is 2nd order (m=2). **Figure 3.2c** shows how the peaks for the multiscale grating are identified and indexed according to the above rules. The peaks corresponding to the primary periodicity (i_0) are surrounded by families of satellite peaks.

3.2.2 SPP excitations on 1D plasmonic superlattices

The excitation of SPPs on a nanostructured metal surface is given by the Bragg coupling condition[88]

$$k_0\sqrt{\frac{\varepsilon_\mathrm{d}\varepsilon_\mathrm{m}}{\varepsilon_\mathrm{d}+\varepsilon_\mathrm{m}}} = \mathbf{k}_\mathrm{SPP} = |\mathbf{k}_0 \sin\theta + \mathbf{G}| \tag{3.3}$$

where $\mathbf{k}_\mathrm{SPP}$ is the momentum of the SPP mode, $\mathbf{k}_0$ is the momentum of free space light incident on the surface at angle θ, **G** is the grating vector, and ε_m and ε_d are the complex permittivity of the metal and dielectric, respectively. Zero-order reflection spectra of a 400-nm/10-μm pattern with an angle of incidence $\theta = 20°$ showed several SPP modes, indicated as reflection minima (**Figure 3.3a**). Applying the grating vector from equation (3.2) into equation (3.3), we identified the strongest modes as -1_0 and -1_1. The -1_0 mode, that is, the mode that only has a contribution from the primary periodicity, is consistent with the SPP mode of an infinite array with the same

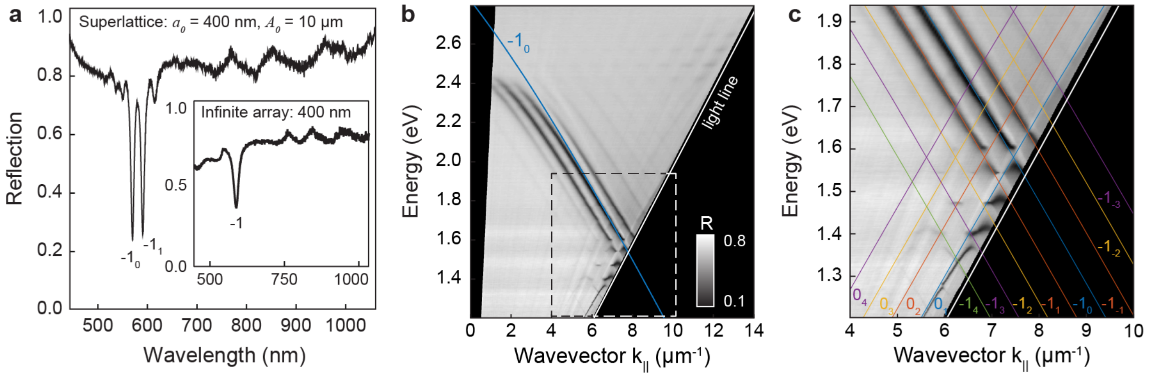


Figure 3.3: Reflection spectra of superlattices with indexed SPP modes. Reflection spectrum ($\theta = 20°$) for a plasmonic superlattice with $a_0 = 400$ nm and $A_0 = 10$ µm and for an infinite 400-nm array (inset) (b) Dispersion diagram with the -1_0 SPP mode overlaid. (c) Zoom of the dispersion diagram in (b, dotted box) with SPP modes shown and labeled. The dashed white line indicates the light line (*i.e.* $\theta = 90°$)

periodicity, a_0 = 400 nm (**Figure 3.3a inset**). Angle-resolved reflection spectra over a large angle range (5–80°) were stitched together into dispersion diagrams according to

$$k_{||} = \frac{2\pi}{\lambda}\sin\theta \tag{3.4}$$

$$E = \frac{hc}{\lambda} \tag{3.5}$$

to analyze the dispersive properties of the SPP modes (**Figure 3.3b**). The expected -1_0 SPP mode as calculated from equation (3.3) is indicated on the dispersion diagram. The calculated and experimental modes overlap well at high wavevector (large θ), but near $\mathbf{k}_{||} = 0$, band banding can be observed.[89] Due to the symmetry of the 1D superlattice, the energy spacing between the satellite modes remains constant at $\Delta E = 0.11$, consistent with the magnitude $|\mathbf{K}|$.

The most prominent SPP modes for this plasmonic superlattice were evident at low energy and high wavevector (**Figure 3.3c**). For the -1_J family of modes, several high order modes were visible up to -1_6, a seventh order ($m = 7$) mode. Additionally, while the -1_J modes progressed from upper left to bottom right, there are sharp SPP resonances near the light line progressing from bottom left, to upper right. These SPP modes are part of the 0_J family, *i.e.* their grating vector only has contributions from the superperiodicity. Several of the intersections between the -1_J and 0_J modes resulted in plasmonic band gaps, including those of high orders (*e.g.* 0_1 and -1_5). Previous studies on formation of plasmonic band gaps on high symmetry plasmonic crystals concluded that band gaps would only be observed if the two intersecting modes were within two orders of each other (*i.e.* $\Delta m < 2$).[57] The SPPs in these plasmonic superlattices clearly go beyond this restriction, showing band gaps up to $\Delta m = 5$. These band gaps, including the observation of the 0_J modes, were only possible due to the large-scale fabrication techniques that enable large measurement areas at high angles of incidence ($\theta > 60°$).

3.2.3 Effects of alternate superperiodicity on SPPs

The flexibility of multiscale patterning allowed for the modification of various fabrication parameters to create different superlattice structures and better understand the relationship between the primary periodicity and superperiodicity in the excitation of SPP modes. Using contact lithography masks with different periodicities A_0, three additional superlattices were fabricated with the same primary periodicity a_0 = 400 nm, but with varying superperiodicity A_0 = 20 µm, 30 µm, and 50 µm (**Figure 3.4**). All structures maintained the 50% duty cycle for the superperiodicity, and therefore the patterned regions increased in size with increasing A_0. Slight misalignment in the azimuthal angle φ can be identified at the edges of the patterned regions, where some of the individual lines are cut-off.

High-angle ($\theta = 65°$) reflection spectra for all four of the plasmonic superlattices are shown in **Figure 3.5**. Since each superlattice had the same primary periodicity, the location of the -1_0 mode in each spectrum was at the same wavelength. Meanwhile, as the superperiodicity increased, the corresponding value of **K** decreased, thereby decreasing **G** according to equation (3.2) and decreasing the difference in wavelength between the resonance peaks. As with the spectra for the A_0 = 10 µm superlattice, several high order modes were observable at these high angles of incidence. The A_0 = 50 µm superlattice demonstrated the highest order SPP mode (-1_9, $m = 10$) ever reported for nanostructured metal films. The highest order mode observed before this work was $m = 4$ in a Au/Si system.[90-91]

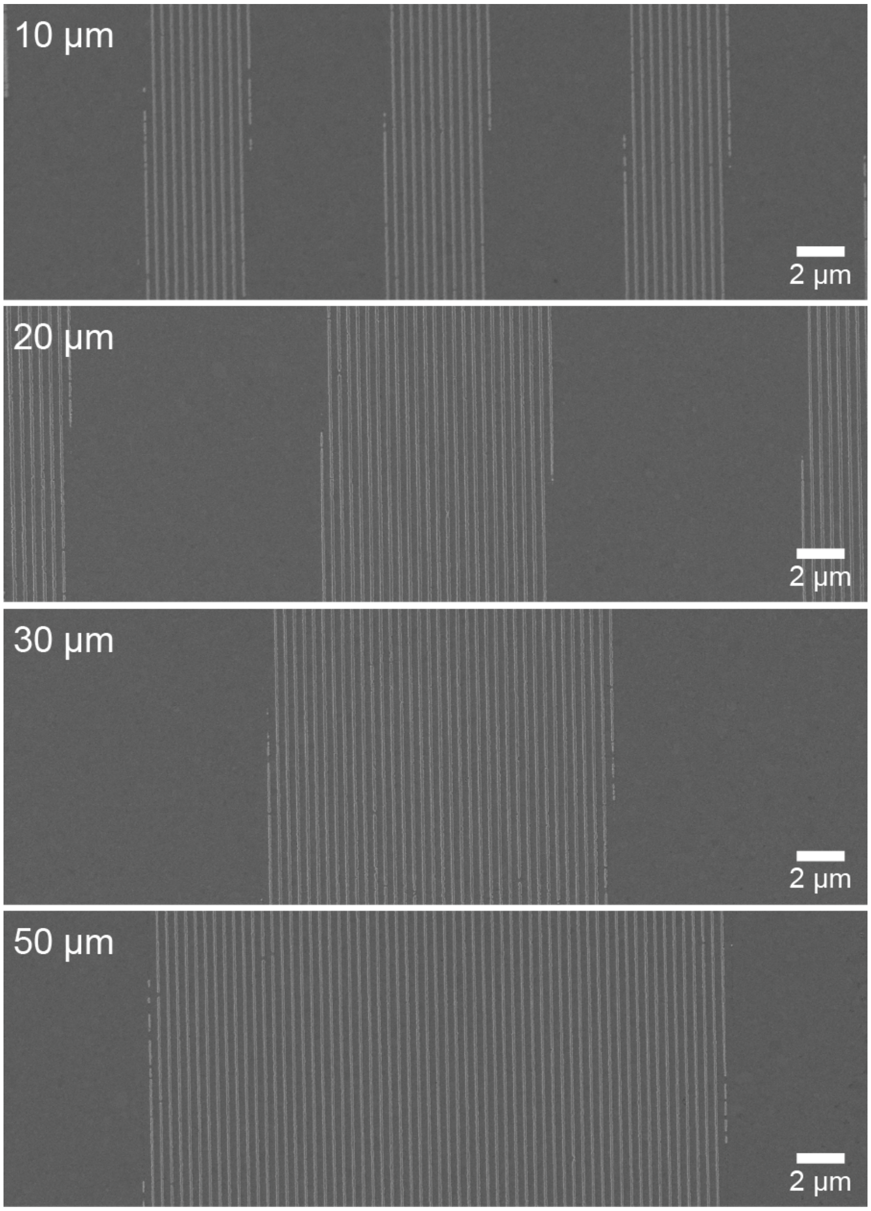


Figure 3.4: Multiscale gratings with varying superperiodicity. SEM images of Si templates with primary periodicity a_0 = 400 nm and A_0 = 10 μm, 20 μm, 30 μm, and 50 μm (top to bottom).

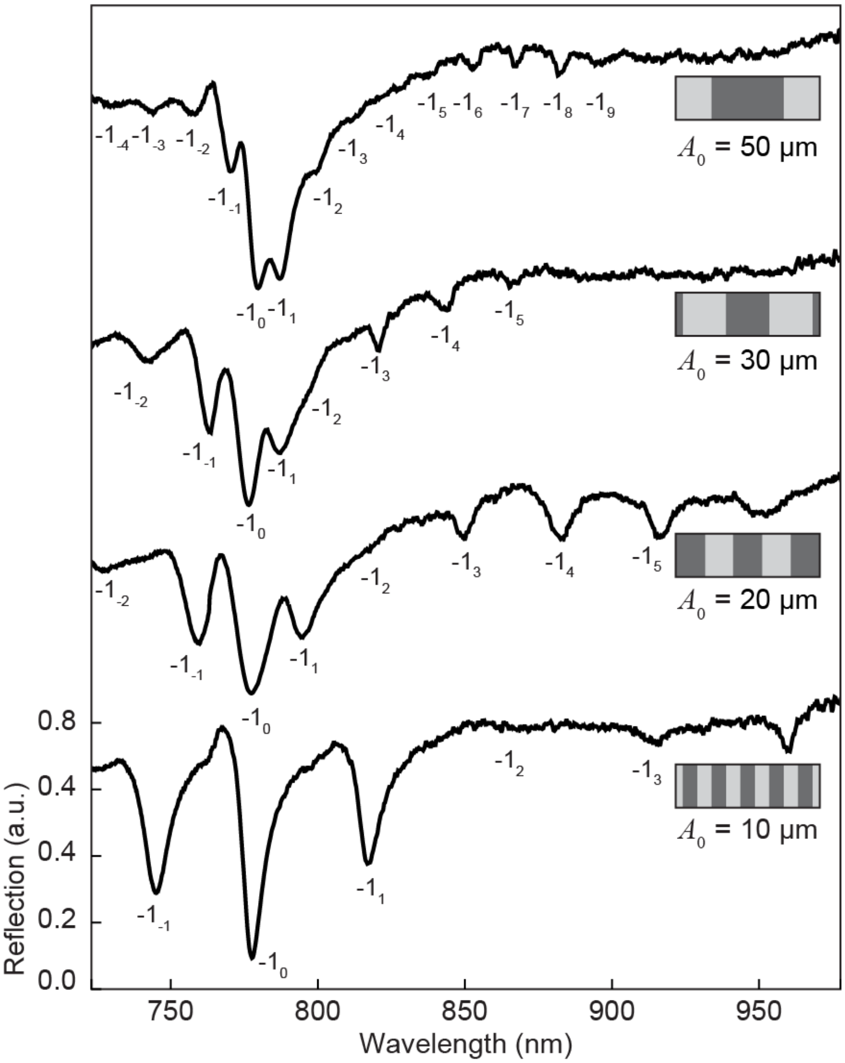


Figure 3.5: Reflection spectra of multiscale gratings with varying superperiodicity. Reflection spectra ($\theta = 65°$) of multiscale gratings with $a_0 = 400$ nm and $A_0 = 10$, 20, 30, and 50 µm (bottom to top). SPP modes are labeled according to their index.

The full dispersion diagrams clearly show a decrease in ΔE with increasing periodicity (**Figure 3.6**), and share similar features with the spectra of the 10-μm superlattice in **Figure 3.3**. For each superperiodicity, ΔE between the individual satellite modes remains constant. The SPPs for each substrate also showed similar degrees of band bending at $\mathbf{k}_{||} = 0$ indicating that the band banding is not influenced by the microscale periodicity, but is likely a result of the nanoscale geometry. Finally, each dispersion diagram showed several high order satellite modes at high angles, as shown in **Figure 3.5**, with clear bad gap formation at high wavevector and low energies where the -1_J and 0_J modes intersect. Remarkably, these high order modes and band gaps were prevalent even at the largest superperiodicity ($A_0 = 50$ μm). The existence of strong SPPs influenced by this large periodicity can be better appreciated in the context of small area fabrication by EBL. In both experiments and simulations of plasmonic crystals, unit cells ≤ 50 μm are often used to mimic isolated structures. The strong coupling of SPP modes over 50-μm superperiodicity, however, indicated that SPPs can still interact across these unit cell dimensions, and care should be taken to exclude any superperiodicity effects in future work.

3.2.4 Incommensurate multiscale arrays with varying azimuthal angle.

We fabricated multiscale gratings with larger primary periodicity ($a_0 = 475$ nm) such that the 10-μm superperiodicity is explicitly not an integer multiple. A SEM image of the structure and its FT are shown in **Figure 3.7a**. As with the 400-nm primary periodicity superlattice, the FT shows groupings of peaks, where the spacing between satellite peaks is the based on the 10-μm periodicity as those in **Figure 3.2**, but the larger spacing between primary peaks was decreased due to the decrease (increase) in $|\mathbf{k}|$ (a_0). This decrease in $\mathbf{k}$ was also observed in the primary -1_0 SPP mode excitation occurring at lower energies in **Figure 3.7b** compared to in **Figure 3.3b** (*e.g.*

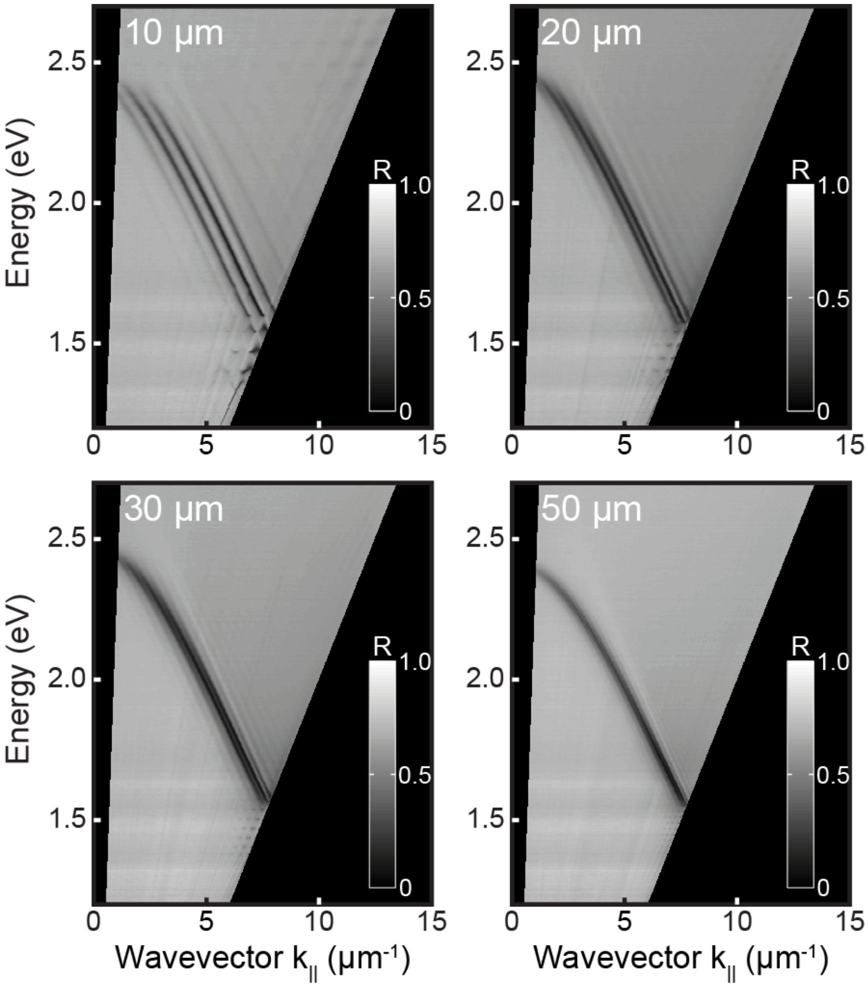


Figure 3.6: Dispersion diagrams of multiscale gratings with varying superperiodicity. Multiscale gratings have primary periodicity a_0 = 400 nm and A_0 = 10, 20, 30, and 50 μm as indicated.

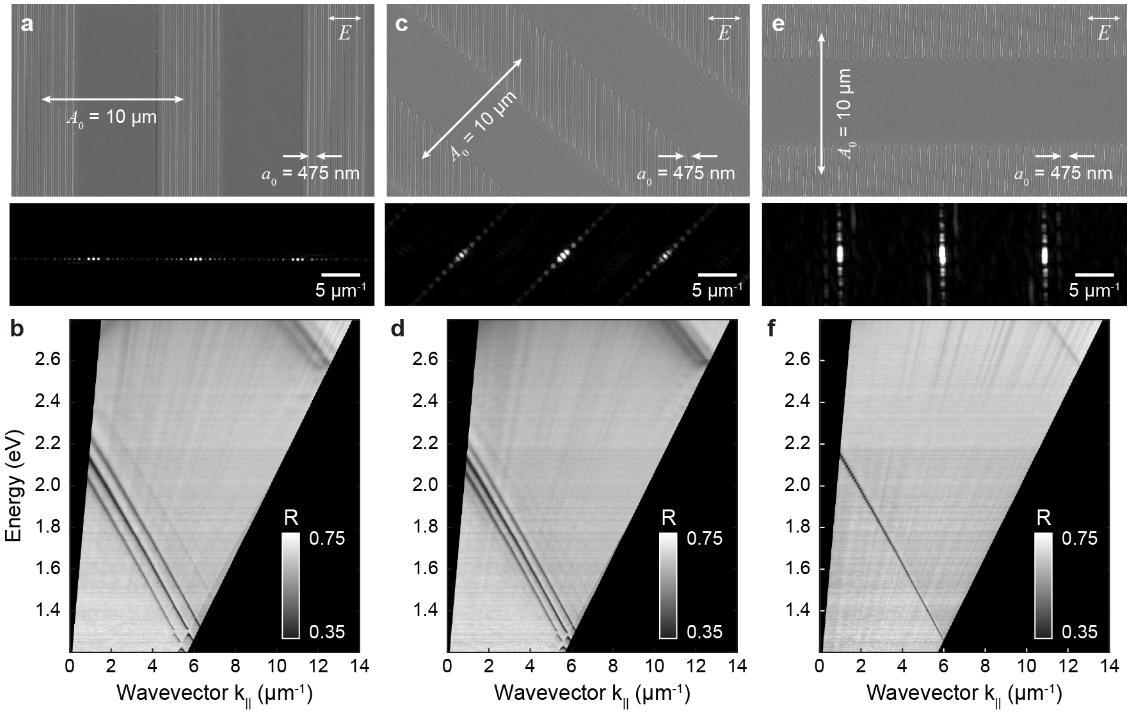


Figure 3.7: Multiscale gratings with varying azimuthal angles (φ). (a,c,e) SEM (top) and FT (bottom) for superlattices with a_0 = 475 nm and A_0 = 10 μm, and φ = 0°, 45° and 90° respectively. (b,d,f) Dispersion diagrams of the above structures showing the decrease in ΔE for the satellite peaks with increasing φ.

for the -1_0 mode at $k_{||} = 4\ \mu m^{-1}$, $E^{400\text{ nm}} = 2.17$ eV while $E^{475\text{ nm}} = 1.77$ eV). The overall reduction in excitation photon energy also allowed a second-order primary periodicity mode and its satellite modes (-2_J) to be visible at high wavevectors and high energy. While the 0_1 mode was not as prominent in this sample as in the 400-nm superlattice, the opening of band gaps between low-order 0_J modes and -1_J modes was still apparent at low energies and high wavevector. Band gaps also appeared at high energies between 0_J modes and -2_J modes.

While the two periodicities are incommensurate even when aligned at $\varphi = 0°$, increasing the relative azimuthal angle φ between the two lithography masks still affected the SPP mode excitation. **Figure 3.7c** shows the SEM and FT of a multiscale array with the same periodicities as **Figure 3.7a**, but with an azimuthal offset of $\varphi = 45°$. The orientation was such that the primary periodicity maintained the same alignment, while the superperiodicity was rotated. This rotation is reflected in the FT where the satellite peaks maintained the same spacing (*i.e.* $|\mathbf{K}|$ did not change), but the orientation of the peaks had rotated. The alignment and spacing of the primary periodicity spacing $|\mathbf{k}|$ was unchanged. The dispersion diagram of the 45° superlattice (**Figure 3.7d**) shows identical dispersive properties for the -1_0 mode, but the spacing between the satellite modes was reduced. This decrease existed because the direction of $\mathbf{k}_{||}$ is defined as the $+\boldsymbol{k_x}$ direction. Even though $|\mathbf{K}|$ remained unchanged, the projection along k_x is reduced according to $\cos(\varphi)$. For $\varphi = 45°$, therefore, the spacing between the satellite modes was reduced by a factor of $\sim\cos(45°) = 0.707$, while the dispersion remained parallel to the -1_0 mode. This effect of φ on the SPP modes was direct evidence that the behavior of SPPs in these incommensurate multiscale arrays is different from the optical modes of EBL-fabricated nanowire superlattices. If the primary periodicity selected which higher order superperiodicity modes were excited, then the dispersion diagram would look very different. **Figure 3.8** shows the calculated high-order modes of the superperiodicity alone at φ = 45° (red lines), which display different E–**k** dispersion than the observed satellite modes. Finally, multiscale gratings with φ = 90° were fabricated, as shown in **Figure 3.7e**. The fabrication of this superlattice had a slight misalignment, which is clearly visible in the FT as weak modes off-axis from the direction of the satellite peaks. These artifacts are due to pixilation in the SEM image being misaligned to the orientation of the primary periodicity. The dispersion diagram of this structure shows only one strong first-order resonance, and a second-order resonance at higher energy (**Figure 3.7f**). Since the spacing between satellite peaks was nearly reduced to $\Delta E = 0$, all the -1_J modes were overlapped on the dispersion diagram and appear to be a single mode. Additionally, the 0_J modes were also no longer visible at high wavevector due to their overlap with the fundamental 0_0 SPP mode, which exists beyond the light line.

3.2.5 Double periodicity superlattices

We took advantage of the 50% duty cycle of the superperiodicity to fabricate multiscale arrays with two independent superperiodicities. As stated earlier in the chapter, the Si template of a single multiscale array can be used in another phase of lithography + etch (**Figure 3.1**, right). This technique is similar to the standard LELE process in the semiconductor industry, but with modified double-exposure conditions during the litho steps. After reapplication of PR, exposure through a second PDMS mask with different periodicity (b_0) was done. The second exposure used the same mask as in the first LE phase, and the alignment was inverted such that the unpatterned areas were blocked to preserve the second periodicity in these regions. After an additional round of Cr deposition, lift-off, and Si etch, the doubly patterned superlattice template was completed. A SEM image of this template is shown in **Figure 3.9a** where the two primary periodicities were

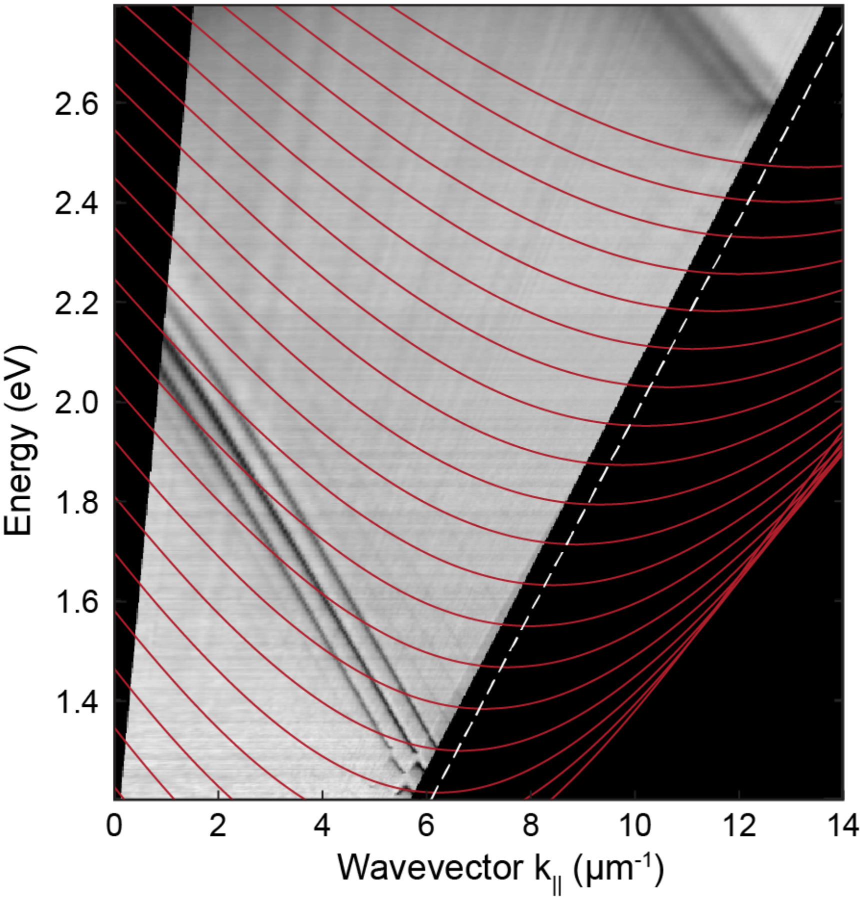


Figure 3.8: Comparison between high-order and satellite SPP modes. Dispersion diagram of a multiscale grating with a_0 = 475 nm, A_0 = 10 μm, and φ = 45°. Overlay (red lines) shows the expected dispersion of high order SPP modes from the superperiodicity alone

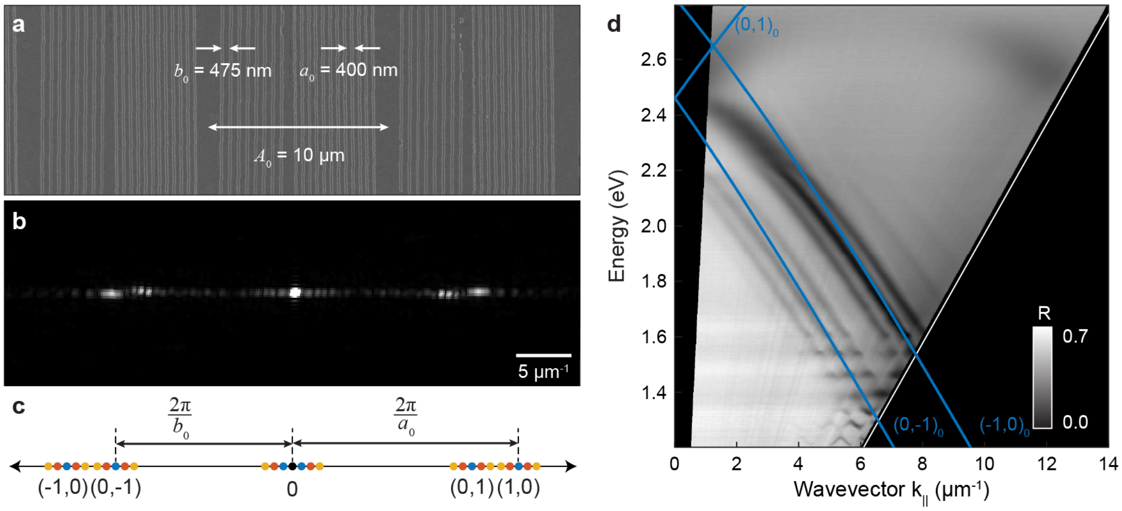


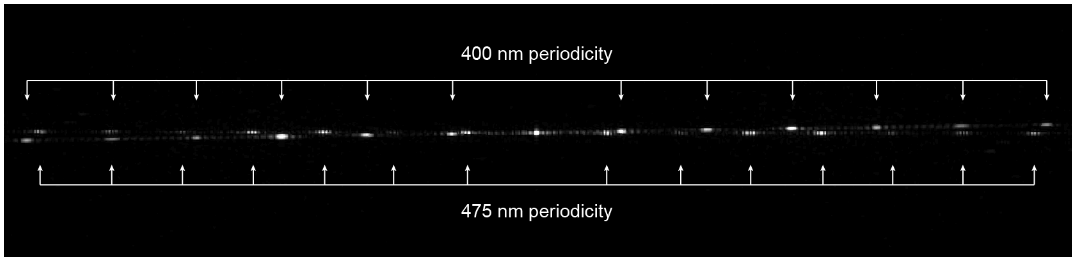
Figure 3.9: Multiscale gratings with two primary periodicities. (a) SEM of the multiscale array with a_0 = 400 nm, b_0 = 475 nm, A_0 = 10 μm. (b) FT and (c) indexing scheme of the structure showing each primary periodicity with its own set of satellite peaks. (d) Dispersion diagram of the array with SPP modes from the primary periodicities only as overlay

$a_0 = 400$ nm, and $b_0 = 475$ nm with a superperiodicity of $A_0 = 10$ μm. Good alignment of the contact lithography mask resulted in no overlap between the two regions. Over-exposure of the superperiodicity mask, however, resulted in a duty cycle < 50% and therefore some narrow areas remained unpatterned. The FT shows two families of primary periodicity peaks with identical satellite modes (**Figure 3.9b**). The spacing of the primary periodicities is described by $|\mathbf{k_a}| = 2\pi / a_0$ and $|\mathbf{k_b}| = 2\pi / b_0$ with $|\mathbf{k_a}| > |\mathbf{k_b}|$. A zoom-out view of the FT is displayed in **Figure 3.10** where the different spacings of $|\mathbf{k_a}|$ and $|\mathbf{k_b}|$ are apparent. A slight misalignment between the two PDMS masks was evident by an angular offset between the $\mathbf{k_a}$ and $\mathbf{k_b}$ groupings of peaks. Since the patterns shared the same superperiodicity, the satellite peaks were aligned even for high orders. The inclusion of the second primary periodicity modifies the equation of **G**:

$$\mathbf{G} = (h\,\mathbf{k}_a + i\,\mathbf{k}_b) + J\,\mathbf{K} \tag{3.6}$$

and provides a modified indexing scheme for the modes: $(h, i)_J$ (**Figure 3.9c**).

The dispersion diagram shows excitation of both first order modes of the primary periodicity $(-1, 0)_0$ and $(0, -1)_0$ and their associated satellite modes (**Figure 3.9d**). The solid lines indicate the expected dispersion according to equation (3.6), which agrees well at high wavevector and deviates from the experimental spectra at low wavevector due to band bending. The energy spacing ΔE of the satellite modes around each $(h,i)_0$ mode was constant and consistent with that of single periodicity pattern (**Figure 3.3b**) since it is based on the same value of $|\mathbf{K}|$. One key difference between the $(0, -1)_J$ and $(-1, 0)_J$ modes is a decreased intensity for the SPP modes from the 475-nm periodicity. One cause of this difference could be the fewer number of units within a given superperiodicity compared to the 400-nm area[85]. According to **Figure 3.9a**, there are 12 units of 400-nm periodicity patterns while, only 8 units of the 475-nm periodicity pattern.



s

Figure 3.10: Wide view FT for double patterned superlattices. The groupings for the two primary periodicities $a_0 = 400$ nm and $b_0 = 475$ nm are identified. The slight misalignment between the two periodicities is evident especially at high orders. The satellite modes are all aligned however, since there is only one superperiodicity $A_0 = 10$ µm.

3.3 Experimental methods

3.3.1 Lithography procedure for multiscale arrays

To create periodic 1D photoresist patterns, a PDMS photomask with a linear feature-to-feature lattice spacing of 400 or 475 nm was placed into conformal contact with positive-tone, g-line photoresist (Shipley S1805) diluted with poly(ethylene glycol) methyl ethyl acrylate in a 1:2 ratio by volume (~120 nm thick) on a Si [100] wafer and exposed to a broadband Hg-vapor light source (SUSS MicroTec MA6) for 2.5 s at a power density of 11 mW/cm^2. After the exposure with a PDMS photomask, a quartz photomask was placed into low-vacuum contact with the PR-coated wafer and exposed for 2.5 s. This hard mask consisted of four different periodicities (10 µm, 20 µm, 30 µm, 50 µm) with 50% duty cycle of transparent and opaque linear regions. For these superlattices, the direction of the submicron pattern was manually aligned to be the desired azimuthal angle φ. The alignment accuracy was generally within 1°.

3.3.2 Pattern transfer to plasmonic superlattice crystals

The exposed PR was developed (1:5 dilution of Microposit 351 developer in water) and resulted in arrays of PR lines on a Si substrate. In order to clear away excess photoresist, the samples were exposed to an O_2 plasma in an Samco reactive ion etcher (50 sccm, 300 mtorr, 30 watts) for ~15 s. A 10-nm Cr sacrificial layer was deposited onto the substrate through line-of-sight physical vapor electron beam deposition with a Kurt J. Lesker PVD 75. The PR was lifted off with Microposit Remover 1165 leaving lines of bare Si exposed in the Cr layer. Trenches with depths of ~50 nm were etched anisotropically using a C_4F_8/SF_6 gas co-flow recipe in a STS LpX Pegasus Deep Reactive Ion Etcher. The Cr layer was removed, and 160 nm of Ag were deposited in the PVD 75 onto the Si template.

3.3.3 Angle-resolved reflectance spectroscopy

Zero-order reflectance spectra were collected from $\theta = 5°$ to $80°$ in $0.5°$ increments using an automated, self-designed LightField software add-in (Princeton Instruments). Collimated, unpolarized white light from a halogen lamp (100 W) illuminated the sample with a spot size of 2 mm. The reflected light was coupled into a bundled optical fiber connected to a Princeton Instruments Acton SP2500 spectrometer with a PIXIS:400 CCD detector. A linear interpolation algorithm in MATLAB converted the measured optical data – wavelength (λ) and excitation angle (θ) – into dispersion diagrams as photon energy (E_{photon}) and in-plane wavevector of light ($\mathbf{k}_{||}$).

3.4 Summary

We have shown the fabrication of plasmonic multiscale gratings and investigated their optical properties. The addition of a superperiodicity to a plasmonic crystal generated satellite SPP modes that were explained by increasing the components of the grating vector **G** of the Bragg coupling equation. These terms are independent unlike in photonic superlattices, and varying the primary periodicity, superperiodicity or relative orientation between them altered the dispersion of SPP modes. Finally, we combined multiscale patterning with the LELE technique to fabricate superlattices with two primary periodicities. The inclusion of a superperiodicity and further a second primary periodicity to a plasmonic crystal greatly increased the spectral bandwidth available for exciting SPP modes in applications requiring broadband light trapping from photovoltaics to plasmonic lasing.

CHAPTER 4:

PROCESS-BASED MODELS OF PLASMONIC BOWTIE ANTENNAS

4.1 Background

Plasmonic nanostructures demonstrate large enhancement of optical fields in confined volumes.[19] Assemblies of particles such as dimers can further increase the local field intensity by orders of magnitude due to near-field interactions between the nanoparticles at their localized surface plasmon (LSP) resonance wavelength. These high fields can show non-linear behavior of the nanoparticles,[92-93] and have applications in sensing,[94] surface enhanced Raman spectroscopy,[95] and plasmonic lasing.[96] Understanding the origins of LSP modes and high field enhancement requires knowledge of the near-field distribution for a given structure. While the near-field can be measured directly, finite-difference time-domain (FDTD) simulations of the structures are a more appealing approach to visualize local fields due to their speed and ease of use.

FDTD simulations of plasmonic nanoparticles determine the full electromagnetic fields in the simulation environment by solving time-dependent Maxwell's equations.[62] Results from FDTD simulations therefore include both near-field properties such as plasmonic field enhancement as well as far-field properties such as transmission or reflection spectra. The exact far-field spectra of specific nanoparticles can be replicated by making detailed measurements of the nanoparticles and using those measurements to create accurate simulation objects.[97] Advancements in object design in FDTD simulation software has enabled modeling of particles with various shapes. Features on 2D shapes, or extruded 3D volumes can be modeled with a high degree of accuracy, including tapered edges and rounded corners. A downside to these advancements is that large degrees of freedom in the design of nanoparticles can produce large variations in both near-field and far-field properties.[97]

Here we design FDTD simulations of plasmonic systems that incorporate experimental processing parameters into construction of the object models. We model our simulation after the shapes of 3D bowtie dimers and show how varying the fabrication procedure alters the optical properties of the final structure. For specific nanofabrication parameters, FDTD simulations match trends in experimental far-field spectra, though non-uniformity in the real structures prevents high accuracy in modeling. Finally, we reveal the origin of an out-of-plane LSP mode that appears under specific geometric conditions.

4.2 Results and Discussion

4.2.1 Fabrication parameters in experiment and FDTD model

Arrays of nanoparticle bowtie dimers are fabricated using a technique based on PEEL[44] that has been reported previously.[96, 98] The procedure is outlined in **Figure 4.1**. Briefly, square arrays of round photoresist (PR) posts were formed on the surface of a Si(100) wafer by phase-shifting photolithography with a soft, elastomeric mask followed by development. A thin, sacrificial layer of metal (typically Cr) to be used as an etch- and deposition-mask was deposited on the post array followed by lift-off to create a nanohole array. The exposed Si was anisotropically etched in a KOH-based solution to form a self-aligned array of pyramidal pits beneath the nanoholes. Two off-normal depositions of Au or Ag with deposition angle Ψ, azimuthal angles φ and $\varphi + 180°$, and thickness t created the templated bowtie dimer. For optical measurements, the particles were template-stripped onto a transparent substrate.

Several aspects of the fabrication process affected the final optical response of the dimers. Since each nanoparticle was created by masked deposition through the hole array, the diameter of the initial PR posts (as determined by both the phase-shifting mask and exposure/development

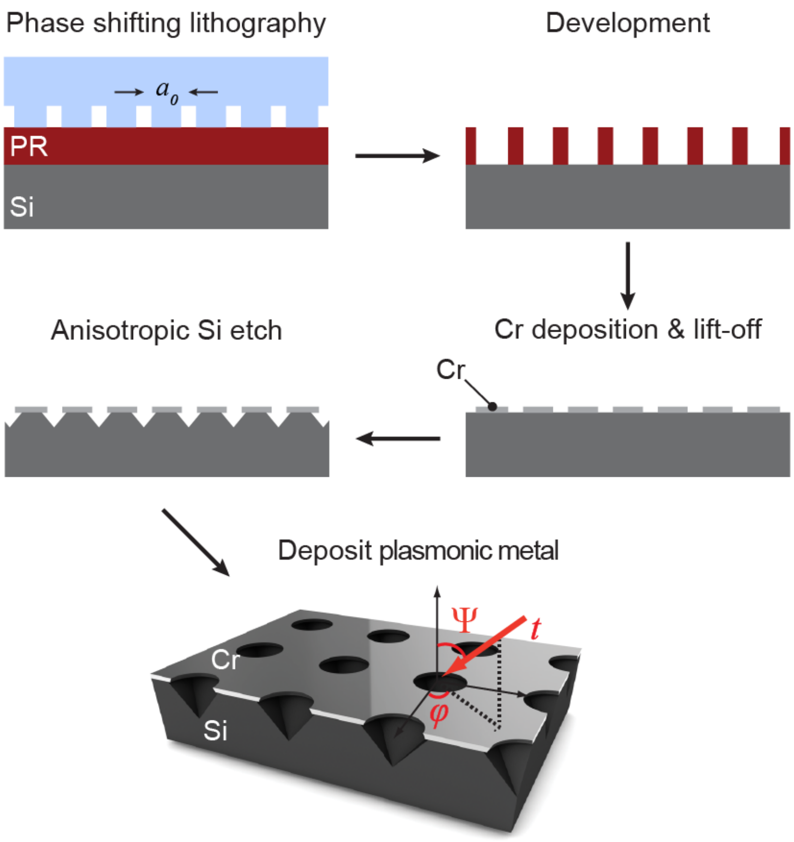


Figure 4.1: Fabrication scheme. Steps of the bowtie fabrication procedure from lithography until bowtie deposition. The final 3D image shows the particle deposition direction through the nanohole mask (red arrow) characterized by the deposition angle Ψ, azimuthal angle *φ*, and thickness *t*.

conditions) directly influenced the size of the final particles. The thickness of the sacrificial metal layer affected the width of the nanoparticle in the plane of the deposition direction since a thicker layer would shadow a larger cross-sectional area of the hole. Finally, the three deposition parameters (Ψ, φ, t) were the dominating factors in the final particle shape.

The fabrication parameters were used to create an FDTD model for the bowtie dimer based on geometric affine transformations. The basis of this technique is the orthogonal projection of a hole shape onto an arbitrary plane. First, the elliptical shape was determined based on the size of the hole, film thickness, and deposition angle. As the deposition occurred, the top film also increased in thickness according to $t\sin(\Psi)$, which shrinks the final hole shape. The hole was projected in 3D onto the desired surface plane by the following matrix equation

$$J_p \mathbf{Y}_p = \left(\mathbf{I}_3 - \frac{D N_p^T}{D \cdot N_p} \right) J_e \mathbf{Y}_e \tag{4.1}$$

where J is the basis vectors, $\mathbf{Y}$ is a matrix of vectors describing the coordinates of the hole, $\mathbf{I}_3$ is the 3×3 identity matrix, D is the projection direction vector, and N is the surface normal of the desired plane. The subscripts p and e represent the projected and starting bases respectively, and ^{T} denotes the transpose operator. A schematic diagram of this projection is shown in **Figure 4.2a**. The projection vector D was defined by the experimental deposition angle Ψ and azimuthal angle φ. The nanoparticle was created by stacking a series of slabs approximately 1 nm in thickness linearly changing size and shape from the starting hole shape to the final hole shape. Since this technique is generalized for any shape of hole and any target surface, it can be used for several applications (**Figure 4.2b**). For bowtie dimers, the hole was projected onto each of the 4 surfaces of the pyramidal pit (only two or three of the surfaces would contain real particles). Deposition

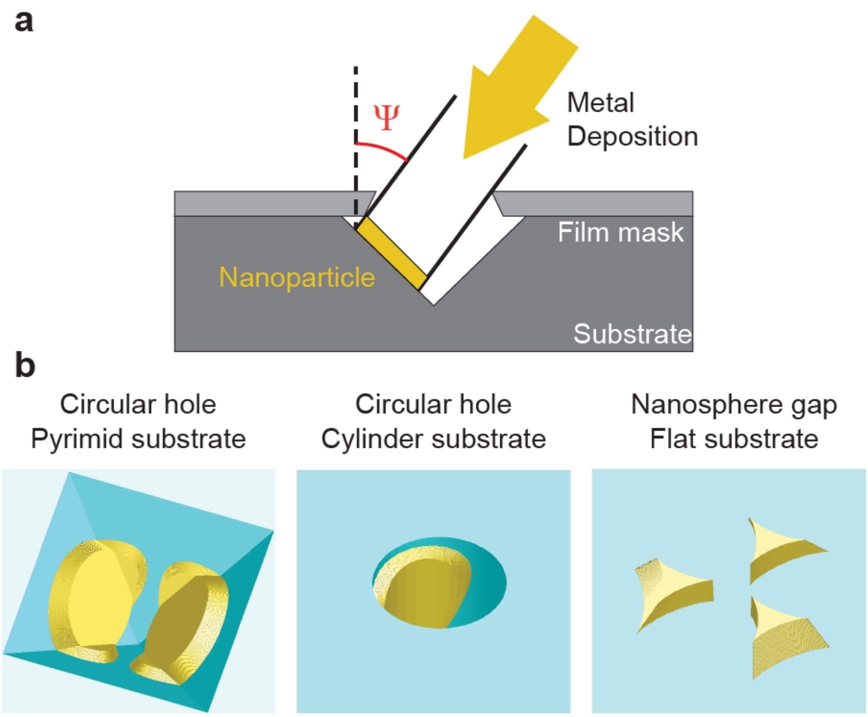


Figure 4.2: Flexibility of FDTD model. (a) Scheme for the nanohole projection in FDTD to generate the nanoparticle object. (b) FDTD layout images showing nanoparticle objects created with various hole shapes and projection planes.

from a hole onto the edge of a cylinder is more complex due to the curved walls. For this case, the cylinder is approximated as an extruded n-gon where $n \approx 100$. Finally, this technique is also suited to simpler projections between parallel planes such as nanoparticle structures fabricated by nanosphere lithography.[26] The particles shown in **Figure 4.2b** (right) are from three angled depositions through a hole at the intersection of three spheres.

The size of each particle of the bowtie dimer is set by the circular hole in the Cr mask, and these particles can be large compared to the size of pyramidal pit that contains them, limiting the available scope of angles Ψ and φ. The size of the Si pyramid created during the fabrication process was dictated by the diameter of the nanohole due to the anisotropic nature of the Si wet etch.[48] Specifically, longer etch times do not increase the size of the pyramid after the final shape has been formed. To increase the width of the pyramidal pit and allow for more flexibility in nanoparticle deposition, an *isotropic* etch must first be done. By treating the exposed Si wafer through the nanohole mask in a reactive ion etcher (RIE), the size of the Si pyramid after KOH etching was increased (**Figure 4.3**). The relationship between RIE time and increased pyramid size was found to be linear with a rate of 5.1 nm of increased width per second of RIE treatment. Initially, there was no change in width for RIE times below 10 s, since any etched volume would fall inside the boundary of the initial pyramidal pit. Scanning electron microscopy (SEM) images of several hole arrays on Si after both RIE and wet etching show the increase in width of the Si pyramid for similarly shaped holes (**Figure 4.3b**). Additionally, the quality of the final pyramidal pit was not damaged by the RIE treatment.

4.2.2 Linear properties of bowties with various geometries

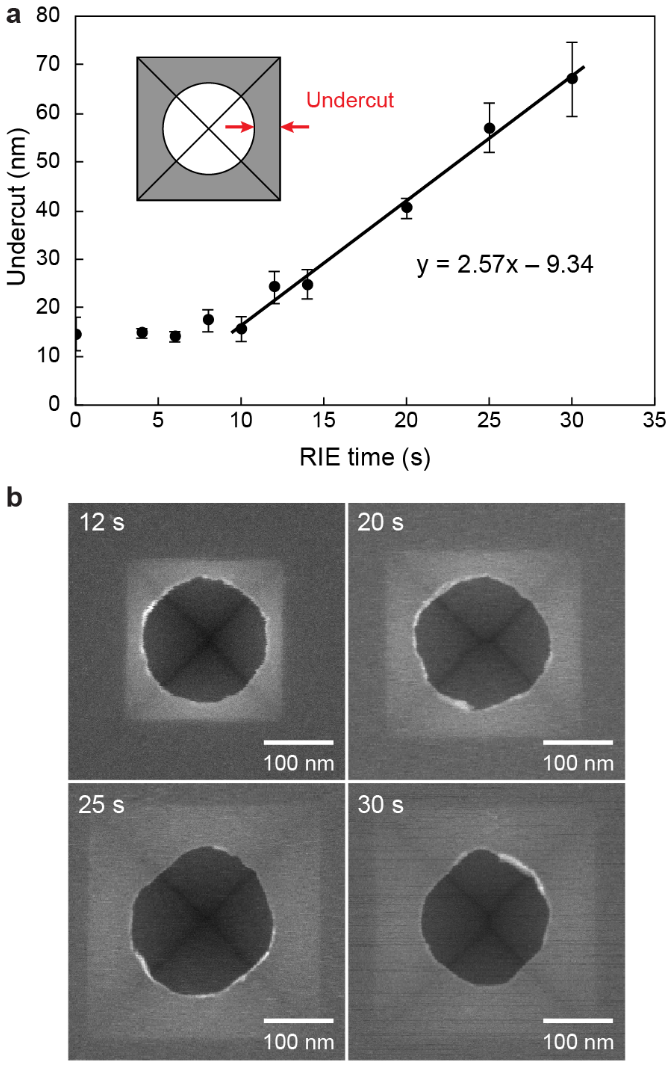


Figure 4.3: Effect of RIE etch on Si pyramidal pit shape. (a) Undercut distances between pyramid edge and hole edge for 10 different RIE etch times followed by anisotropic wet etching. Each point represents an average of 5–10 measurements on different holes in the same sample. (b) SEM images of samples with 4 different etch times.

A larger Si pyramid enables a wider range of angles and thicknesses for deposited nanoparticles. To discover the range of available LSP resonance wavelengths, a series of angles (Ψ) and thickness (t) were tested with typical dimensions for nanoholes and pyramidal pits (**Figure 4.4**). We simulated bowties centered in the corners of the pyramids (*i.e.* $\varphi_1 = 45°$) over a 10° range of Ψ from 30° to 40° (**Figure 4.4a**). These limits were chosen because for $\Psi < 30°$, the two bowtie particles would be in contact in the center of the pyramid, and for $\Psi > 40°$, the top of the bowtie would make contact with the upper film, forming a continuous structure. The transmission spectra showed two characteristic LSP peaks, though for decreasing Ψ, the second peak is redshifted beyond the wavelength range of the simulation, and likewise beyond the measureable range for a visible-wavelength spectrometer (**Figure 4.4b**). We observed that the redder of the two peaks shifted more significantly than the bluer peak, which remained relatively unchanged for $36° < \Psi < 40°$. At $\Psi = 30°$, the resonance near 800 nm showed a splitting into several lower intensity peaks. This change in line shape is likely due to simulation errors at the ~5-nm gap between two particles in the center of the pyramid — a precision that typically is not achievable in experiment.

We then fixed the deposition direction, and varied the deposition thickness from 20 nm to 50 nm in 5-nm intervals (**Figure 4.4c**). For this specific experiment, there were two competing factors to determine the spectrum of a bowtie dimer. Increasing the thickness of an isolated nanoparticle causes a *blueshift* of the LSP wavelength,[99] however decreasing the spacing between two particles in a dimer produces a *redshift*.[100] Increasing the thickness of the particles in the bowtie results in both effects as a consequence of the angled plane of the pyramid. The transmission spectra for this sweep of thicknesses (20 nm < t < 50 nm) shows that both LSP

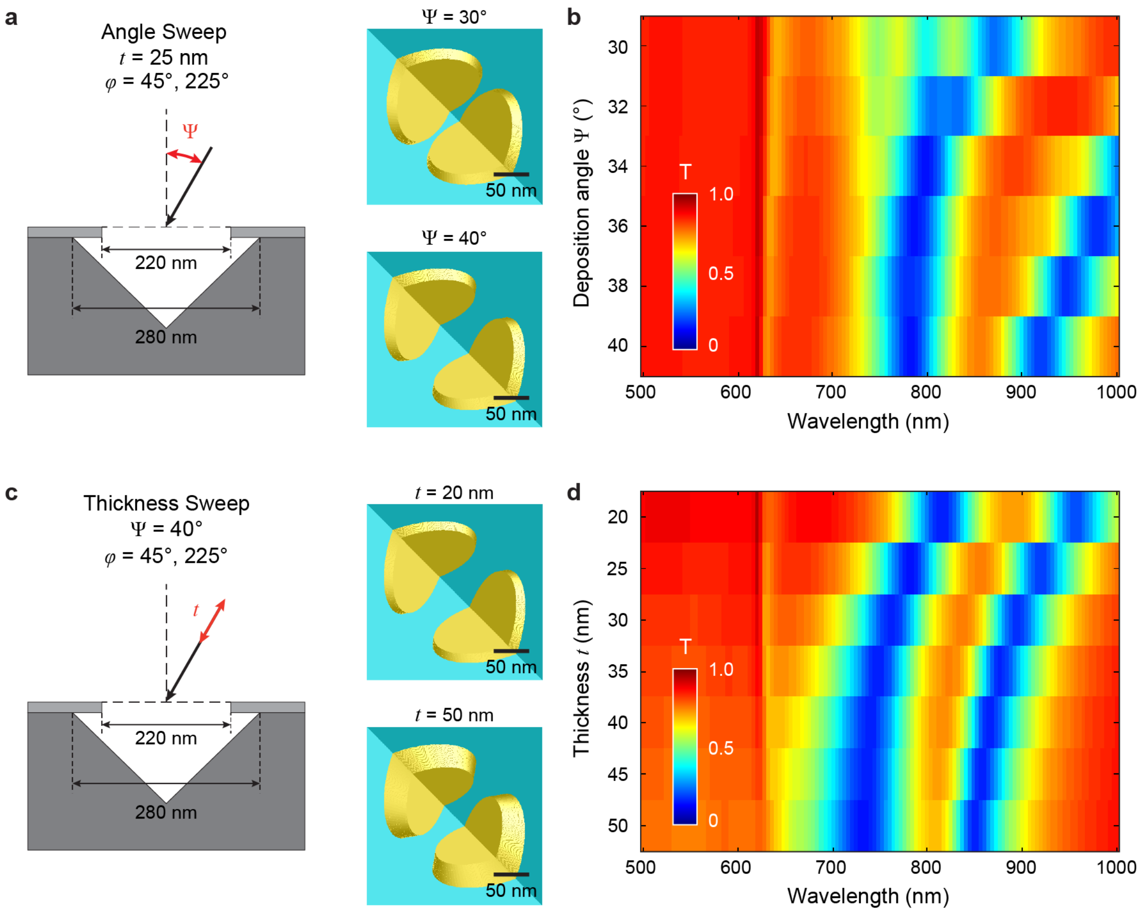


Figure 4.4: Sweeps of Ψ and t for Au corner-centered bowties. (a,c) Schemes of the simulation parameters to generate the nanoparticle objects, with images representing the high and low extremes for Ψ (a) or t (c). (b,d) Simulated transmission spectra for bowties in the visible-NIR wavelength range.

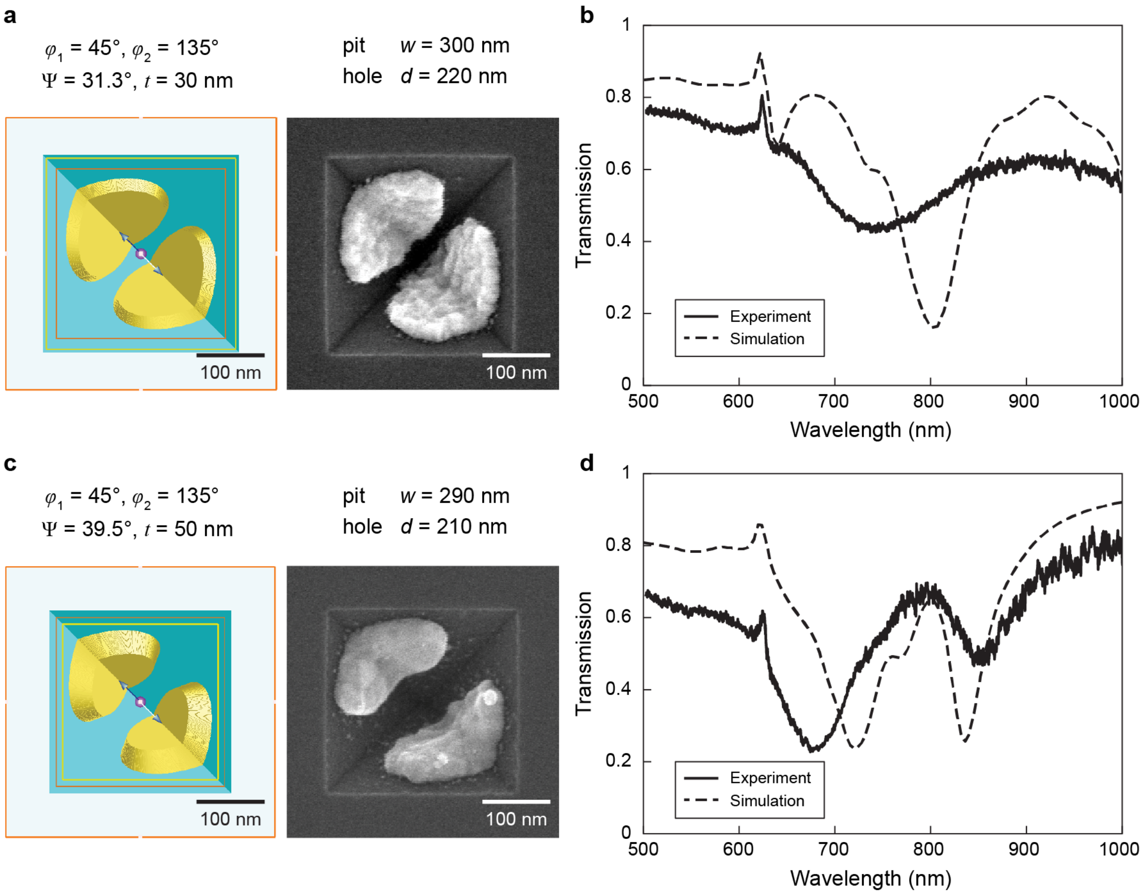


Figure 4.5: FDTD and experimental comparison. (a,c) FDTD objects and SEM images of bowtie nanostructures using the same parameters. Simulation objects were made by using experimental conditions as inputs. (b,d) Experimental and simulated transmission spectra for each dimer.

resonances *blueshift* with increasing thickness, even though the gap between particles also decreases. Unlike with increasing the spacing between the particles with deposition angle (Ψ), both peaks in the bowtie spectrum showed equal wavelength shifts with changing (*t*).

To verify the predictive ability of the FDTD simulations, we fabricated arrays of bowtie dimers with parameters near the extremes of the tested values of Ψ and *t*. The first bowtie was fabricated near the smallest values of each parameter, namely at Ψ = 31.3° and *t* = 30 nm. Top-down views of both the FDTD model and experimental structure by SEM show good agreement in the geometric shape (**Figure 4.5a**). To create the model, SEM images of the nanohole array were used to extract the diameter of the holes and the width of the pyramidal pits, found to be 220 nm and 300 nm respectively. The experimental and simulated spectra are shown in **Figure 4.5b**. While an offset of ~50 nm existed between the observed LSP peaks, the two spectra showed similar trends. They both displayed a broad LSP mode in the 700–800-nm range, and the tail of a NIR-peak beyond 1000 nm. Additionally, the asymmetry on the blue side of the resonance peak was visible in both spectra, though more pronounced in the simulation results.

Near the opposite end of the range of fabrication parameters, Ψ = 39.5° and *t* = 50 nm were used to fabricate the second array of bowties (**Figure 4.5c**). Top down images of the model and experimental structure again show good geometric agreement. The spectra between the simulation and experiment have similar features, but still lack a quantitative match (**Figure 4.5d**). Similar to the previous bowties, the experimental wavelength for the blue LSP was ~50 nm blueshifted compared to the simulated spectrum. For the longer wavelength peak, however, the experimental peak was redshifted compared to simulation. This relationship could not be observed for the previous bowties due to a resonance wavelength beyond 1000 nm. The two spectra showed similar features such as the shoulder on the red side of the blue peak, near 750 nm. Also, both simulations displayed a narrower linewidth for the red peak near 850 nm compared to the blue peak near 700 nm.

While we expected the simulated spectra to match better with experiment, the variation in bowtie geometry across the substrate could have had a role in the differences. The fabrication technique generated large-area (~1 cm^2) samples, and variations in hole diameter and shape existed across the sample (**Figure 4.6a**). Local variations showed a distribution in geometry for bowties that were in close proximity to each other (**Figure 4.6b**). The top particle in **I** showed an asymmetry towards the left, terminating in a sharp peak that decreased the gap size between the two particles. The bottom edge of the particles in **II** had a defect that was inserted into the central gap between particles. In bowtie **III**, both the size of the pyramidal pit and the size of the nanoparticles was less than those of the majority of other particles. As demonstrated in **Figure 4.4**, relatively small changes in the geometry produced dramatic shifts in the optical properties of the bowties.

4.2.3 Out-of-plane bowtie LSP mode

With the ability to freely tune the deposition parameters, we revealed the origin of an out-of-plane plasmonic mode in the bowtie dimers. Previously, the LSP mode only visible for bowties deposited in the corners of pyramidal pits, and not for those deposited on the faces, was identified as the antibonding mode of the dimer.[98] We reproduced this LSP mode in FDTD simulations of Ag bowties by varying the starting azimuthal angle φ_1 of deposition from 0° (face-centered bowties) to 45° (corner-centered bowties) (**Figure 4.7a**). Two LSP resonance peaks were visible in the spectrum: The first was a NIR resonance near 796 nm that was largely unchanged in position or

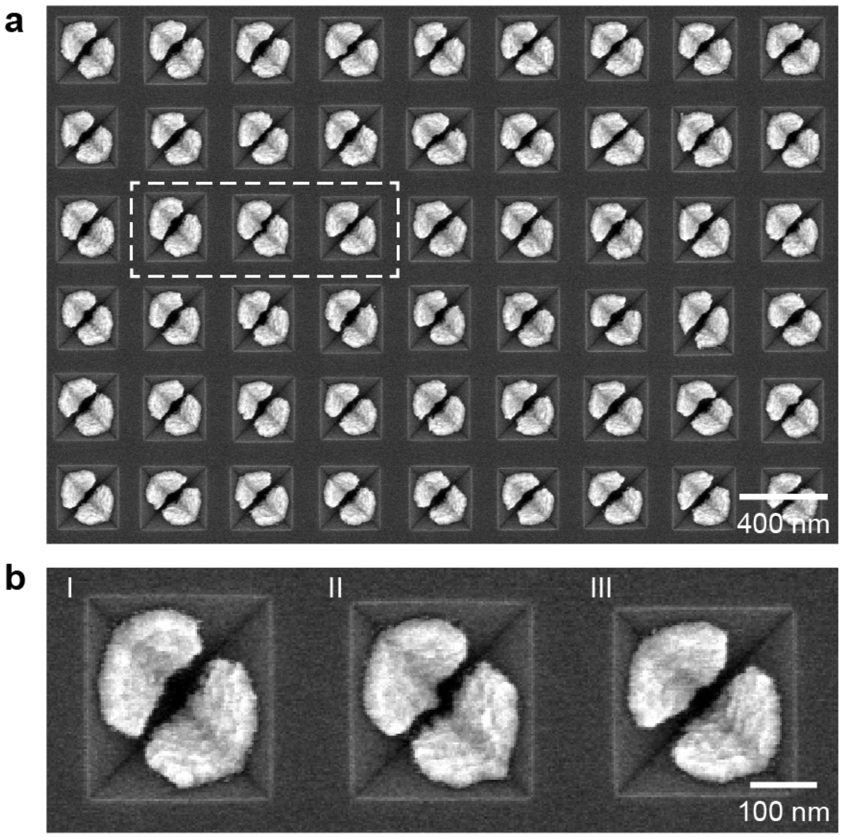


Figure 4.6: Variations in experimental bowtie nanoparticle shapes. (a) Large-area SEM image of a bowtie dimer array. (b) Highlighted particles from (a) showing slight variations in geometry that could affect the bowtie LSP resonance.

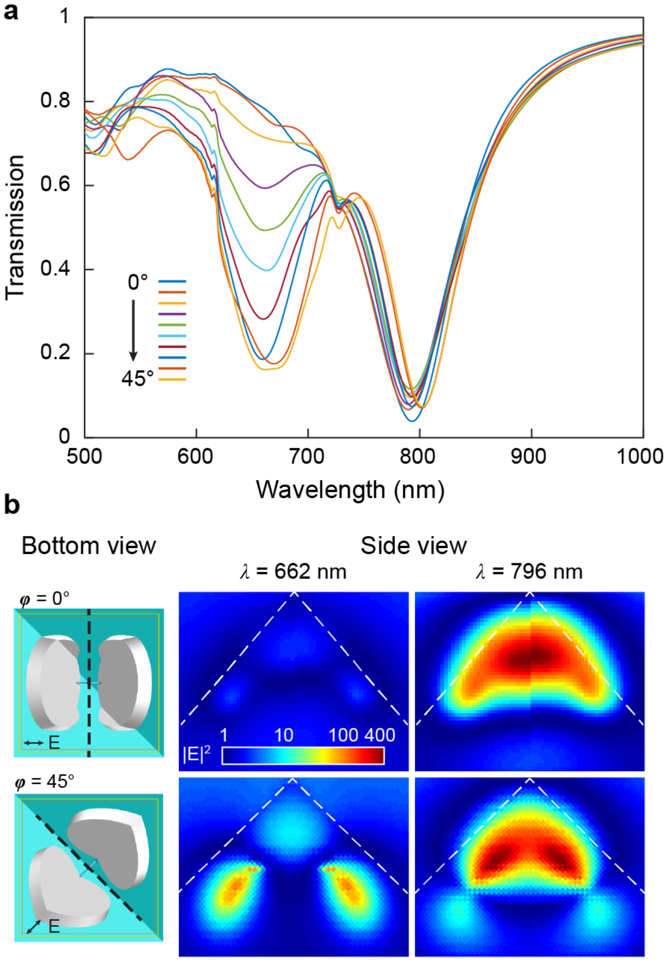


Figure 4.7: LSP dependence on azimuthal deposition angle φ. (a) Transmission spectra for 10 different azimuthal angles φ from 0° to 45° in 5° increments show an increase in LSP mode intensity at 662 nm with increasing angle. The LSP mode at 796 nm remains unchanged. (b) Cross-sectional near-field intensity maps between the nanoparticles in the plane perpendicular to the dimer axis (dashed black lines). Dashed white lines show the boundaries of the pyramid. Near-field intensity is plotted on a log scale.

intensity with changing φ. The other resonance, previously identified as the anti-bonding mode, was in the visible regime at 662 nm and gradually increased in intensity as φ approached 45°. The polarization state of incident light for each measurement was rotated to match the deposition angle φ, *i.e.* the polarization remained parallel to the dimer axis.

To identify the nature of each LSP mode, we visualized the near-field intensity in the gap with a cross section perpendicular to the axis of the dimer (**Figure 4.7b**). For the face-centered bowties ($\varphi = 0°$), only a single LSP mode was visible in the transmission spectrum, at 796 nm. The cross-sectional image shows a strong field enhancement in the gap between the two particles, centered at the tip of the pyramid. There was no significant evidence of an LSP at 662 nm. Conversely, there was strong field enhancement and a peak in the transmission spectrum for both modes in the transmission spectrum of corner-centered bowties ($\varphi = 45°$). The near-field intensity at the 796-nm peak had the same attributes as in the face-centered bowties, though the strongest regions of field enhancement were offset from the midpoint of the structure to match the local geometry of the particles. For the LSP at 662-nm, there was a strong field enhancement focused towards the bottom of the structure, in the gap between the edges of the particles, and much less intensity in the gap at the pyramid tip.

The complex, 3D shape of the bowties required additional, 3D near-field visualizations of the electric near-field intensity to better understand the nature of the two modes. A 3D model of the electric near-field for the corner-centered bowties was used to classify the two LSP modes as in-plane and out-of-plane (**Figure 4.8**). The near-field distribution for the LSP at 796 nm showed an in-plane bonding LSP mode between the two particles. The surface charge density of the nanoparticles was also calculated based on the gradient of the electric field vectors, displayed as

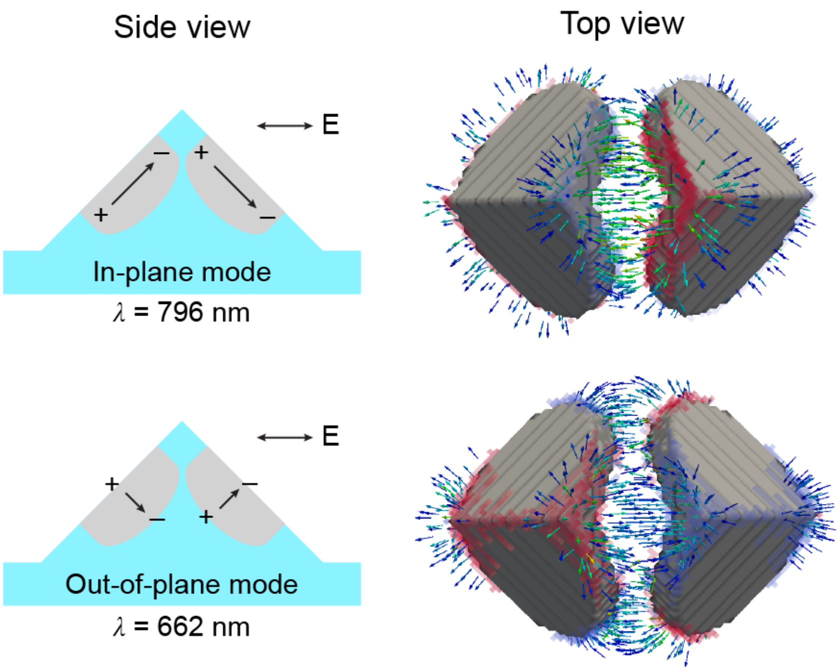


Figure 4.8: Geometric analysis of LSP modes for corner-centered dimers. Schemes of the overall dipolar charge distribution for the in-plane and out-of-plane LSP modes (left). The electric field vectors and charge distribution (red shows positive charge, blue shows negative charge) for each LSP mode on resonance (right).

red (+) and blue (–) shading. The charge and near-field distributions for the LSP at 662 nm revealed its origin to be an out-of-plane mode, caused by the folding of the nanoparticle in the corner of the pyramidal pit. The unique geometry of these pyramidal bowties allowed a single incident polarization to excite both the in-plane and out-of-plane modes simultaneously. The dependence of the out-of-plane mode on the folding of the nanoparticle also agreed with the transmission spectra (**Figure 4.7a**) where the intensity of the mode increased as the azimuthal angle (φ) increased and an increasing portion of the nanoparticle was folded by the corner of the pyramid.

4.3 Experimental methods

4.3.1 Bowtie fabrication

To create 2D square arrays of photoresist posts, a PDMS photomask with feature-to-feature spacing of 400 nm was placed into contact with positive-tone, g-line photoresist (Shipley S1805) diluted with poly(ethylene glycol) methyl ethyl acrylate in a 1:2 ratio by volume (~120 nm thick) on a Si [100] wafer and exposed to UV light in a home-built narrow-band LED light source.[80] After development in a 1:5 dilution of Microposit 351 developer, a 10-nm layer of Cr was deposited through line-of-sight physical vapor electron beam deposition with a Kurt J. Lesker PVD 75, followed by lift-off with Microposit Remover 1165 leaving holes of bare Si in the Cr layer. The substrates were exposed to a CF_4/O_2 (25:3 sccm ratio) plasma in a Samco RIE (13.3 Pa, 100 W) for variable times. The samples were then placed in a bath of 4:1 solution of 30 wt% KOH to isopropanol for 90 s at 72 °C to complete the etch of the pyramidal pit. After Au/Ag depositions in the PVD 75, the top film was removed by etching the Cr layer in Cr etchant (Transene). A UV-curable polymer (NOA 61, Norland Products Inc.) was used to template strip the bowties onto a glass slide for transmission measurement.

4.3.2 FDTD simulations

FDTD simulations were carried out using commercial software (FDTD Solutions, Lumerical Inc.) FDTD regions had spans of 400 nm in x and y with periodic boundary conditions, and 2 μm in z with PML boundaries. A broadband plane-wave source illuminated the samples from the tip of the pyramid. The simulation region had an index of $n = 1.525$ below the bowtie and an index of $n = 1$ above the bowtie. The dielectric functions of the metal nanoparticles were given by Johnson and Christy.[101] A mesh override region around the particle forced a 2-nm mesh in all dimensions. A 2D monitor recorded the full electric and magnetic fields 50 nm above the sample. A 3D monitor to cover the pyramidal pit recorded the nearfield distribution.

4.4 Summary

We have shown development of an FDTD simulation scheme of plasmonic nanoparticles following experimental process constraints. We verified the technique with simulations of pyramidal bowtie nanoparticle dimers, and expanded their nanofabrication process to increase the available range deposition parameters. Variations in the experimental structure placed limits on the agreement between simulated and measured spectral properties. Finally, we identified the origin of a geometry-dependent LSP mode for the pyramidal bowtie system with FDTD near-field analysis.

CHAPTER 5:

CONTRAST INVERSION IN DIC OF GOLD NANORODS

5.1 Background

Nanoscale imaging probes in microscopy such as nanoparticles (NPs) and quantum dots (QDs) are an important tool to probe local properties and mechanisms in soft materials.[102] NPs of noble metals, such as Au, are favorable probes due to strong optical cross-sections derived from their localized surface plasmon (LSP) resonance.[29, 103] These plasmonic particles are non-emitting, unlike QDs or organic fluorophores, which suffer from difficulties in measurement due to QD blinking[104-105] or fluorophore bleaching. AuNPs are therefore beneficial imaging probes, especially for biological applications,[106-107] and various microscopy techniques have been developed to visualize them in biological systems such as thermal imaging,[108-109] darkfield (DF) microscopy,[58, 60] phase contrast microscopy,[110-111] and differential interference contrast (DIC) microscopy.[112-113] Additionally, AuNPs can be used in wide-field, low-light imaging which is beneficial for large-area, long-term studies. AuNPs are also attractive multi-purpose probes that can be used for simultaneous targeting and imaging.[15, 114-116]

Imaging with DIC microscopy has been recently identified as a strong candidate for AuNPs in biological systems due to it being a wide-field, low light, and high resolution technique.[117-119] Image contrast can be optimized based on the target specimen and use of high-numerical-aperture (NA) objective lenses makes optical sectioning practical. While darkfield microcopy has been used extensively to image plasmonic NPs in biological environments, it suffers from problems such as interference from strong scattering from cell membranes,[120] and limited optical sectioning due to a required low-NA objective.

DIC is a phase-contrast imaging technique, though unlike phase-contrast microscopy, which visualizes refractive index differences against a reference, DIC represents *gradients* in the optical path length (*i.e.* refractive index) as intensity contrast. The mechanism of DIC microscopy stems from dual Nomarski prisms that coherently separate elliptically polarized light into two orthogonally polarized and spatially separated beams (the ordinary and extraordinary beams), direct them through a sample, and then recombine the beams into a new elliptical polarization state. The separation distance between the two intermediate beams is referred to as the shear distance, and is typically on the order of 100 nm or less. A linear polarizer before the camera with orientation parallel to the minor axis of the polarization ellipse maps the variable polarization states to an intensity distribution.

Anisotropic nanoparticles, such as gold nanorods (AuNR), with dimensions near or below the shear distance show orientation-dependent properties when visualized under DIC.[121-122] As the AuNR rotates, the intensity contrast alternates between "bright" and "dark" modes. This changing contrast can be used to track the movement, 3D orientation angle, and rotational speed of the particle.[119] Though DIC image contrast is strongest near the LSP resonance wavelength (λ_{LSP}) of the AuNR,[121] varying the LSP by changing size or aspect ratio of the AuNR has not been well studied. Typically, band-pass filters with narrow bandwidths (~10 nm) near the plasmon resonance are used to maximize the imaging of the NP. Previous wavelength-dependent studies on plasmonic nanospheres showed a reduction in overall intensity for DIC wavelengths far from λ_{LSP}.[112] For AuNR, different DIC image contrast was observed at wavelengths corresponding to the longitudinal and transverse plasmon modes.[121] Understanding the effect of LSP wavelength on DIC image contrast is necessary due to the inherent heterogeneity of solution-based NR synthesis techniques.[123-125]

Simulated DIC images can expand on the information gained from DIC microscopy by predicting full image patterns instead of relying on intensity levels alone. Experimental images are inherently noisy and limited in resolution, thus simulations can reveal the underlying intensity distribution for improved pattern recognition.[126] The finite-difference time-domain (FDTD) method[62] is commonly used to simulate plasmonic nanoparticles, but simulations that include macroscale optical systems together with plasmonic particles has been limited. Recently, FDTD models have been developed for both darkfield[127] and phase-contrast microscopy,[110-111] but no simulations exist for plasmonic particles in DIC. Analytical models based on the point-spread function of a DIC microscope are not ideal since assumptions are required for the nanoparticle response.[128] Analytical solutions to determine the optical scattering of isolated plasmonic nanoparticles, such as Mie's theory, can be used for spheres and other simple shapes, but complex shapes or particle assemblies require FDTD simulations.[17]

FDTD simulations can offer additional benefits to DIC microscopy of AuNR. Various particle sizes and aspect ratios can be systematically tested in simulations to build a DIC image library for automated identification and classifications of large-area distributions of nanoparticles. Second, since FDTD can visualize the electric near-field distribution as well as far-field images, correlations between the two can be established so that nanoscopic information such as size, shape, and LSP resonance wavelength can be deduced from DIC images alone without the need for additional measurements with other microscopy techniques (*e.g.* DF, scanning electron microscopy) Finally, the image patterns generated from DIC represent an ideal system, and specific images patters such as relative orientation between bright and dark intensity regions could be used in future studies as training data for advanced machine learning algorithms.

Here we show the development of an FDTD model for DIC microscopy. We verify the validity of the simulation with studies of single AuNR and expand the scope to AuNR dimers. We discover the phenomenon of DIC contrast inversion for plasmonic nanoparticles and determine its origin to be based on the inversion of the electric-near field distribution around the particles on either side of λ_{LSP}. Finally, we show that the DIC contrast inversion is common to all anisotropic AuNR, independent of aspect ratio or LSP wavelength.

5.2 Results and Discussion

5.2.1 DIC microscopy simulation scheme

We developed a 3-stage FDTD simulation scheme for DIC microscopy of plasmonic nanoparticles (**Figure 5.1a**). The optics between the light source and the sample area are replicated in a pre-processing script that sets the initial conditions of the simulation. The initial polarization state of incident light is determined by the experimental settings of the first polarizer and quarter wave plate. The effect of the Nomarski prisms is set by preparing two simulation environments with opposite polarization and their contents shifted by the experimental shear distance (120 nm). The FDTD region exists between the condenser and objective, and simulates the interactions of light with the NP. The simulations are periodic, but large enough to avoid interactions between adjacent particles. **Figure 5.2** shows simulated DIC images with different distances between boundaries. Distances between 2 and 12 μm were tested. Examining the entire simulation space (**Figure 5.2a**), showed that a stable background level exists for 12-μm separation. A 4-μm-wide window around the particles showed minimal influence of neighboring particles at periodicities greater than 6 μm. All simulations in this chapter used a distance of 12 μm. A 2D monitor records transmitted and scattered light leaving the simulation that is processed in a second script, which

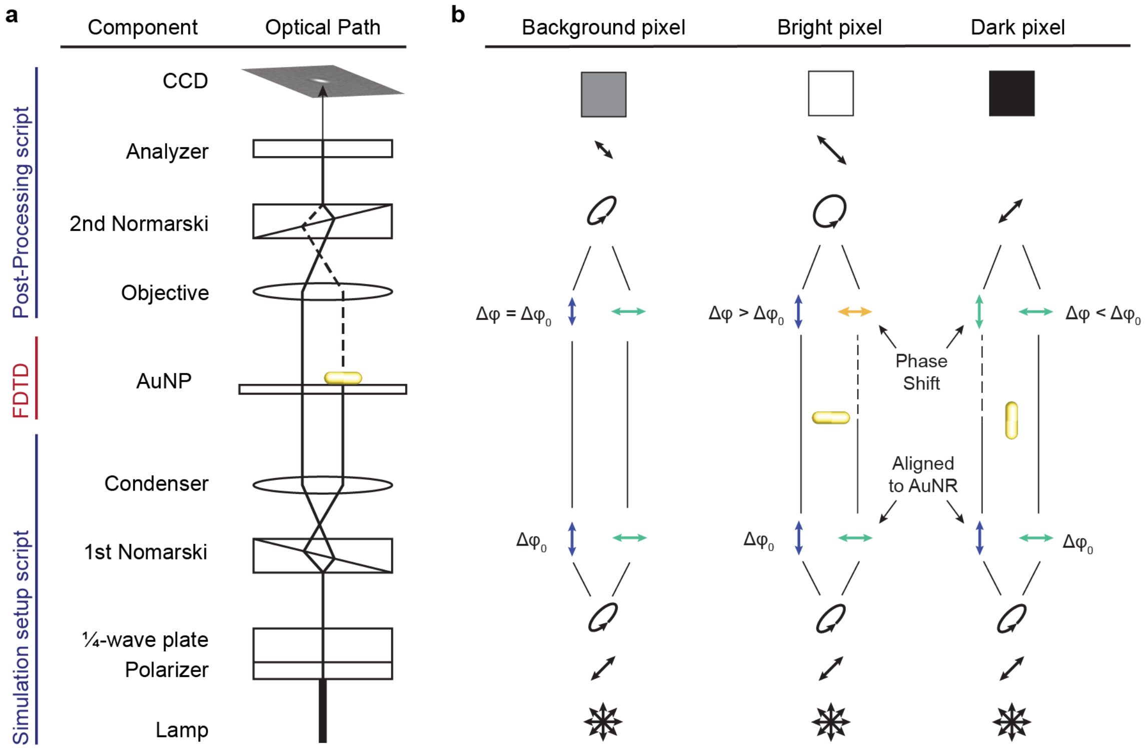


Figure 5.1: FDTD scheme for DIC microscopy. (a) Side-view of the optical path for a DIC microscope. There are three components to the FDTD-DIC simulation environment: Setup, FDTD simulation, Processing. Each section produces optical polarization states corresponding to specific of the DIC microscope. (b) Top-down view of polarization state at different locations in the microscope. The phase shift in the intermediate beams induced by the AuNR causes the bright and dark contrast depending on particle orientation.

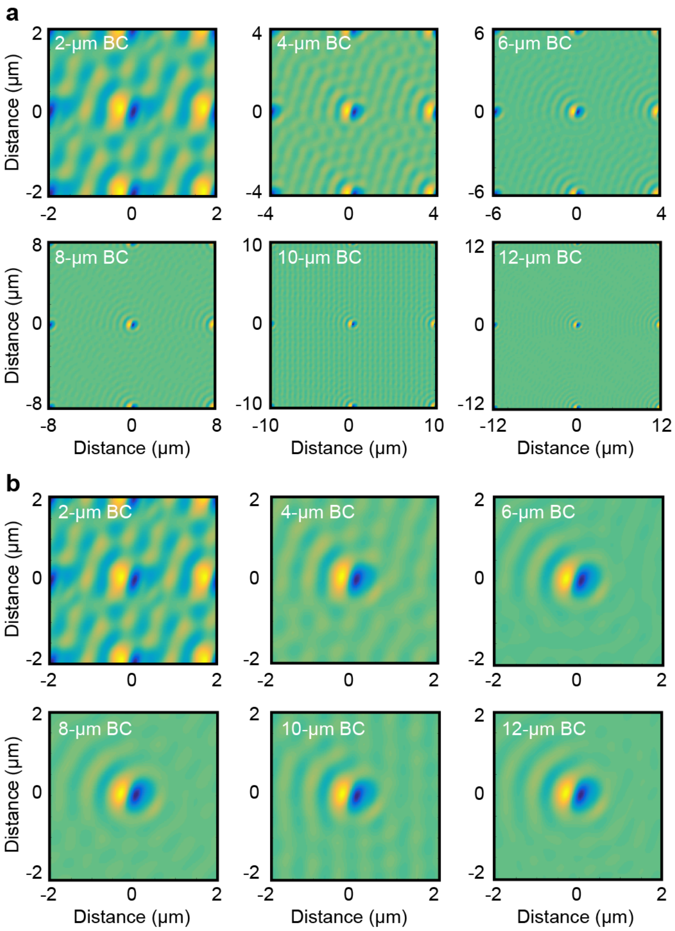


Figure 5.2: Simulated DIC images with different boundary conditions. (a) DIC images with height and width double the periodicity of the simulation. As the periodicity decreases, the background level between particles becomes smoother. (b) The same series of simulated DIC images as in (a) with only the center 4 μm × 4 μm square visible.

represents the optics between the sample and the detector. The light is projected to the farfield to isolate propagating waves, and then the image plane is reconstructed by a Fourier transform.

5.2.2 DIC FDTD simulations of single AuNR and dimers

The expected mechanism for a AuNR to generate bright and dark contrast as a function of orientation is illustrated in **Figure 5.1b**. The background intensity was defined by the initial elliptical polarization state, determined by the angle of the first polarizer with respect to the quarter wave plate. Bright and dark intensities of AuNR were achieved when the longitudinal axis of the particle was aligned to the polarization direction of one of the two beams. In both cases the nanoparticle induced a phase shift in transmitted light. The bright intensity resulted from the nanoparticle increasing the phase difference between the two light beams, causing the final elliptical polarization to be more circular. Similarly, the dark intensity occurred when the phase difference between the two beams was reduced (in the ideal case $\Delta\varphi = 0$), and the final polarization state was more linear.

With the two intermediate beams implemented as separate simulations in FDTD, we examined the interaction of a AuNR with each beam separately. Specifically, by measuring the phase shift induced by the nanoparticle, we verified the proposed mechanism for orientation-dependent contrast (**Figure 5.3**). For the particle alignment that gives a dark-contrast image, the axis of the particle only caused a phase shift in the ordinary beam, resulting in a negative phase difference when the beams were recombined (**Figure 5.3a**). Conversely, when the particle was rotated by 90°, there was only a phase shift in the beam with perpendicular polarization, causing an opposite phase difference when the beams were recombined. For this type of AuNR with dimensions 40 × 90 nm, the maximum final phase difference was approximately $0.06\ \pi$.

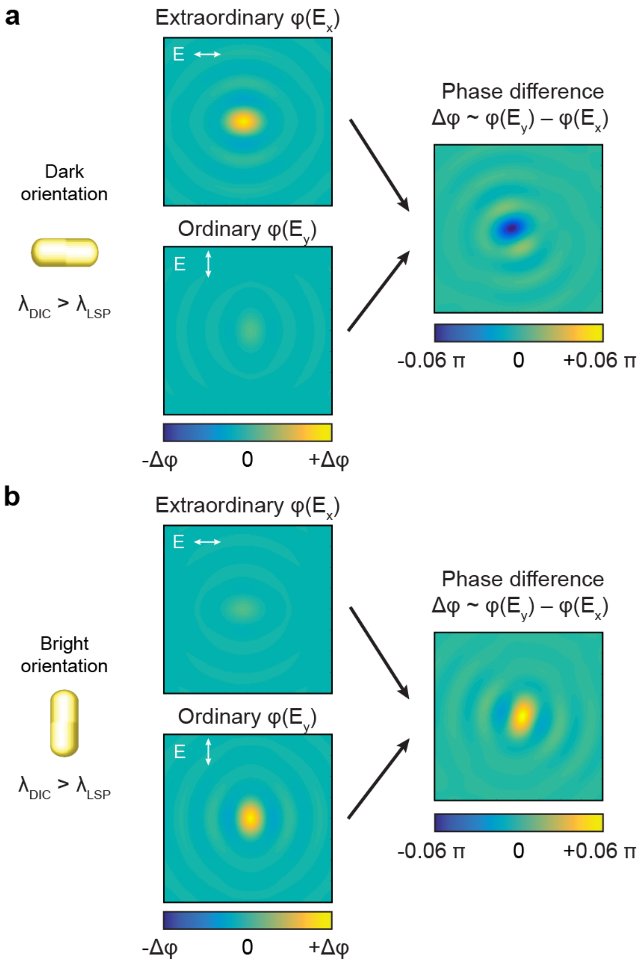


Figure 5.3: Phase difference for intermediate and combined beams. Images displaying the phase induced by a dark-aligned (a) or bright-aligned (b) AuNR compared to the background for the two intermediate beams. The beams are recombined at the Nomarski prism and the final phase difference is shown.

We validated the FDTD simulation results with an in-plane rotational correlation study of AuNRs, combining scanning electron microscopy (SEM), DF, and DIC. DIC images of a single AuNR were taken over 180° of rotation in 5° increments (**Figure 5.4a**). SEM provided the dimensions of the nanoparticle, which were imported to the FDTD simulation. The same rotational angles were simulated and the resulting DIC images are displayed. The orientation dependent contrast of the AuNR was observed in both sets of images, with bright and dark maxima showing the same orientation — aligned to the shear axis of the Nomarski prisms. Additionally, the image patterns near 45° and 135° also show similar trends, with the 45°-images being dimmer, and the 135°-images showing a more even 50/50 bright/dark pattern. Since FDTD does not have restrictions on particle geometry, more complex systems were studied.

A self-assembled AuNR dimer was imaged by DIC and simulated in FDTD (**Figure 5.4b**). The orientation of the dimer was so that one of the particles (left) had the same orientation as the single nanoparticle — aligned to the shear axis of the microscope. The DIC images revealed a difference in the rotation angles corresponding to the maximum bright and dark images, however, compared to the single AuNR. Specifically, the maximum bright and dark intensities, were offset by approximately 40°. The FDTD-simulated images also show this trend, where the maximum intensities are offset by ~40°. The FDTD images also showed higher intensity scattering rings around the dimer than the single particle likely due to the simulations adding the combined scattering effect of both particles. The offset of the peak bright and dark intensities of the dimer was due to the dimer unit effectively acting as a single nanoparticle. In that scheme, the dimer would have a single longitudinal axis, offset by 40° from the axes of the two individual nanoparticles.

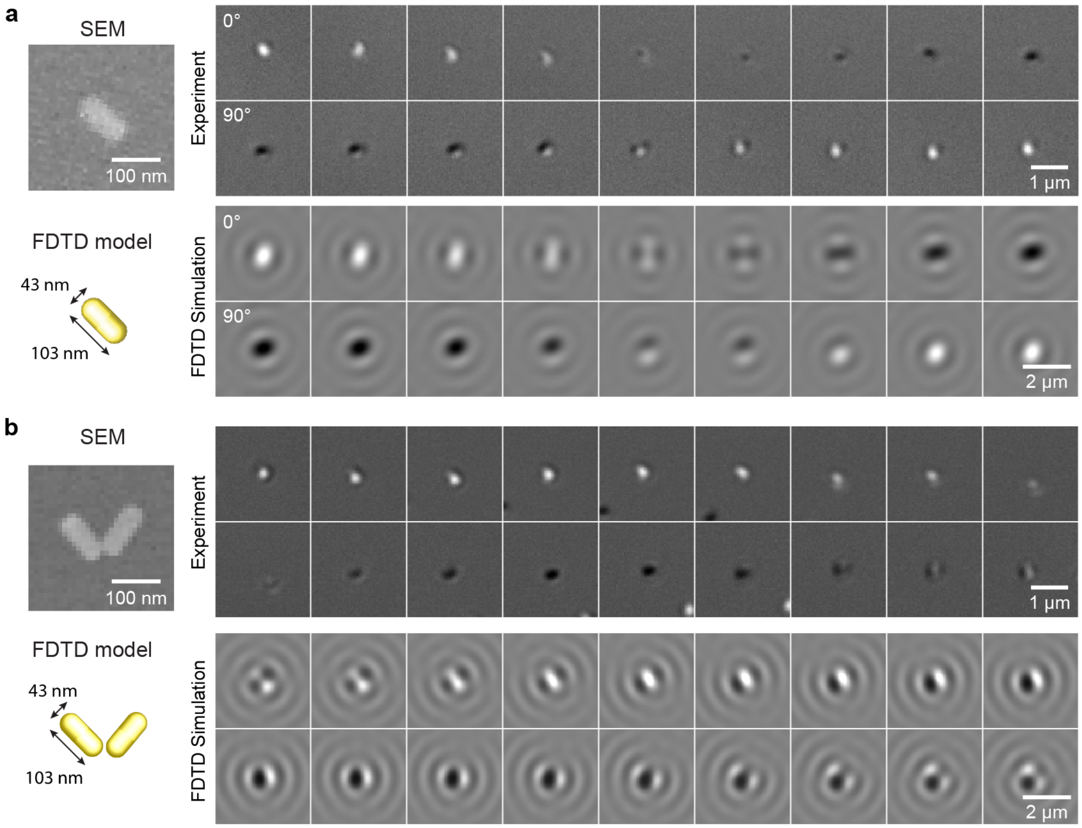


Figure 5.4: Rotation study of AuNRs. DIC images were measured and simulated for a single AuNR (a) and a V-shaped dimer (b) with rotation angles from $\theta = 0°$ to $\theta = 180°$ in 10° increments. The DIC images were correlated with SEM images and the SEM shown represents the particle orientation for $\theta = 0°$.

Our simulations make no assumptions about the interaction between incident light and the AuNR. While previous analytical simulations based on a DIC point-spread-function assume a $\pi/2$ phase shift between the two intermediate beams (*i.e.* circular polarization) and $\pi/6$ phase shift from the AuNR,[128] we relied on experimental parameters to determine the phase differences. The phase offset between the two intermediate beams in this simulation was $\sim\pi/18$, empirically determined to provide high DIC contrast, and in line with the calculated phase shift induced by the nanoparticle. The AuNR-induced phase shift is a consequence of the dielectric function of gold[101] and the dimensions measured from SEM images.

A primary advantage of FDTD simulations is that a broad parameter space can be explored with minimal effort. Sweeping different lengths and widths of AuNR resulted in particles with different LSP resonance wavelengths. This ability is necessary given that commercially available AuNR show a wide range of λ_{LSP}. **Figure 5.5a** shows a large-area, true-color DF microscopy image where each spot indicates a single nanoparticle (verified by correlated SEM). The color of each spot is representative of the scattering wavelength of an AuNR. Several particles were indexed with alphabetic labels, and a selection have their darkfield scattering spectra shown in **Figure 5.5b**. The wavelength range of λ_{LSP} for these particles spread approximately 300 nm. Additionally, the relative scattering intensity varied dramatically between nanoparticle. The intensity values are shown to-scale since identical acquisition settings were used for each NP spectrum. The ability to compare an experimental DIC image to a library of simulated AuNR leads to the use of DIC to determine AuNR geometry, even in diverse samples.

5.2.3 Wavelength-dependent image contrast of AuNR

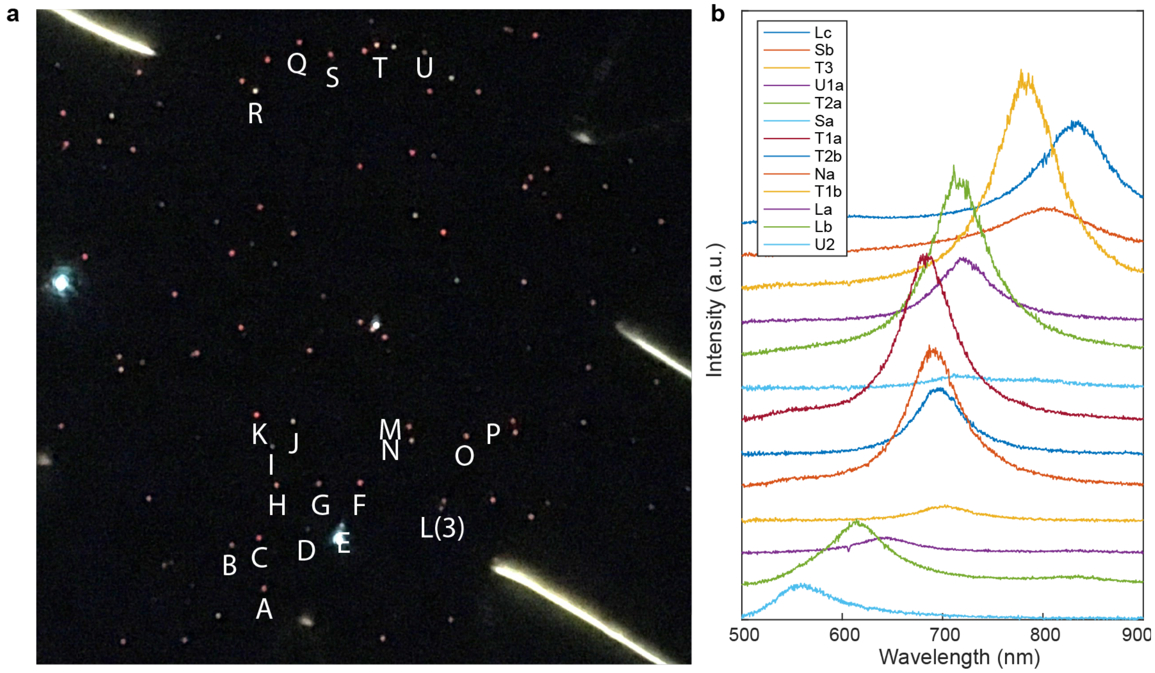


Figure 5.5: Distribution of LSP resonances for AuNR sample. (a) Darkfield image (true color) at 100× magnification showing the scattering of several AuNR. (b) Scattering spectra from select labeled AuNR with λ_{LSP} ranging from ca. 550 nm to 850 nm.

Large-area processing of samples with nanoparticles having different LSP resonances required imaging at multiple DIC wavelengths (λ_{DIC}). Two wide-view images at different DIC wavelengths ($\lambda_{DIC} = 640$ nm and $\lambda_{DIC} = 750$ nm) are shown in **Figure 5.6**. With over 100 individual AuNR in a single image, the importance of computations tools to automate particle identification and tracking is obvious. Between these two DIC images, we discovered that the majority of AuNR switched between bright and dark contrast *without* changing rotation angle. To quantify this result, we examined the rotational dependence of DIC contrast for one of the AuNR (**Figure 5.7**). DF spectroscopy revealed that the LSP resonance wavelength was 678 nm, at the maximum scattering intensity (**Figure 5.7a**). SEM images of the AuNR determined its dimensions to be 43 × 78 nm, which were used to calculated the scattering cross section by FDTD. The two spectra matched well, with both showing a λ_{LSP} between the two DIC wavelengths (640 nm and 750 nm), indicating the FDTD model was an accurate representation of the AuNR.

DIC images of the AuNR were taken for rotation angles between 0° and 180° at 10° increments for both values of λ_{DIC} (**Figure 5.7b**). At 640 nm (*i.e.* $\lambda_{DIC} < \lambda_{LSP}$), alignment of the particle to 0° resulted in a fully bright contrast image, while 90° alignment produced dark contrast. For 750 nm, the contrast at these angles was inverted, but the change in contrast did not correspond to a particle rotation. Angles near 45° or 135° for $\lambda_{DIC} = 640$ nm show regions of both bright and dark contrast in the DIC image. In these images, the dark contrast was in the lower left portion of the image compared to the bright contrast region. In the series of images at $\lambda_{DIC} = 750$ nm, at similar angles, the contrast in these regions is reversed. These images support that the change in contrast is due a flip between bright and dark, rather than a rotation about 90°. The simulated images (**Figure 5.7c**)

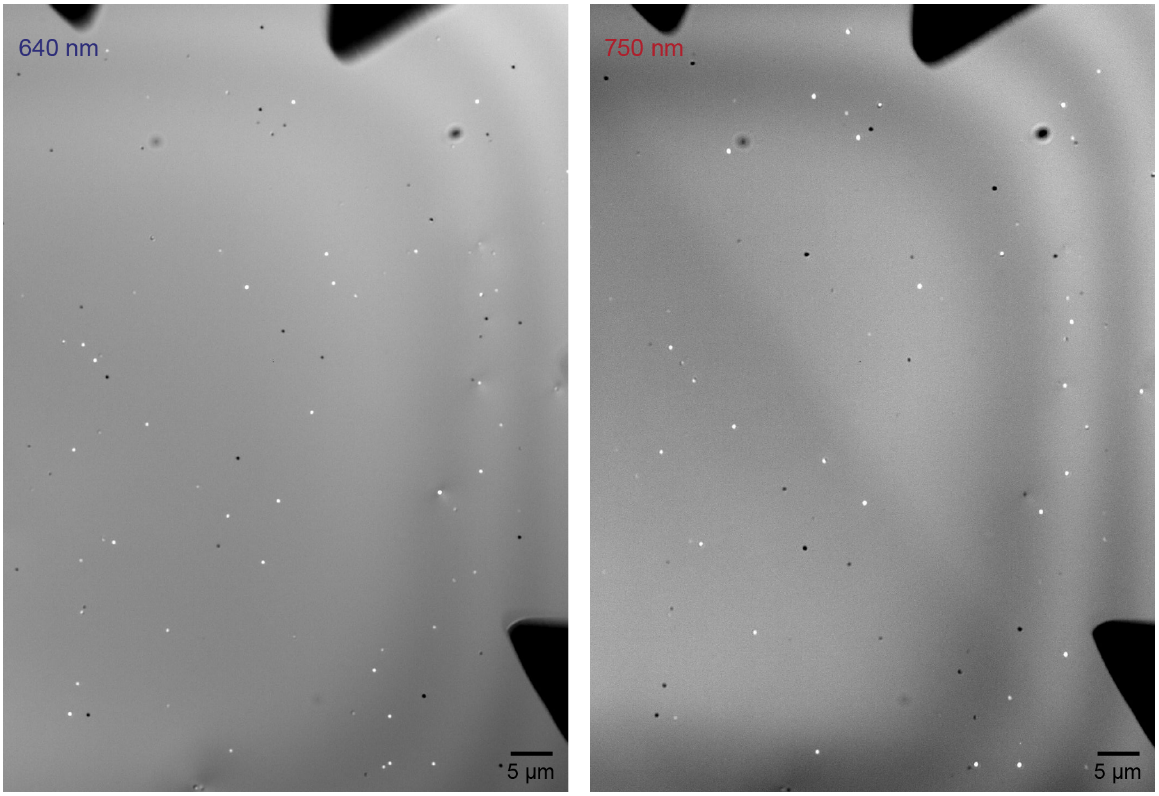


Figure 5.6: Large-area experimental DIC images of AuNR. A single set of AuNR with λ_{DIC} = 640 nm (a) and λ_{DIC} = 750 nm (b). Several particles show an inversion in contrast from bright to dark or vice versa between the two images.

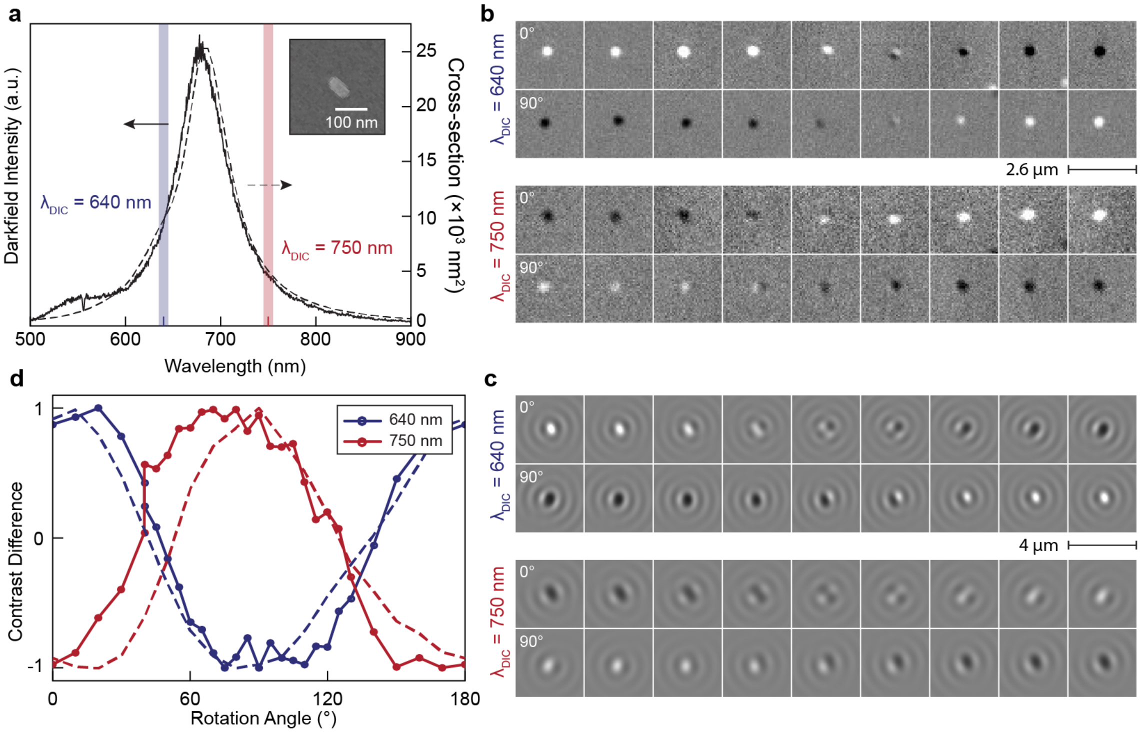


Figure 5.7: Wavelength dependence of DIC images for a single AuNR. (a) SEM and scattering spectra for a single AuNR. Experimental (b) and simulated (c) DIC images at two DIC wavelengths: λ_{DIC} = 640 nm (top) and λ_{DIC} = 750 nm (bottom). (d) Contrast difference calculated from the DIC images showing the contrast inversion between the two wavelengths. Solid lines with markers are from experimental data; dotted lines are calculated from simulated DIC images.

showed similar image patterns, and also replicated the observed contrast inversion between $\lambda_{DIC} < \lambda_{LSP}$ and $\lambda_{DIC} > \lambda_{LSP}$.

To quantify the image contrast, we calculated the contrast difference of images in each rotational series (**Figure 5.7d**). First, we identified bright (dark) regions with intensity above (below) a threshold level relative to the local background intensity. The threshold was tunable, and typically 1–2 standard deviations away from the background mean. The average intensity level in these regions, after subtracting the average background level, defined the bright and dark signals. Then, the signals were divided by the background, giving the bright and dark contrast for each image, where the contrast is always a positive value. Bright and dark contrast were independently normalized to the range [0,1], and the contrast difference (CD) is defined as the difference between them. Therefore, an image with the largest bright intensity level has CD ≈ 1, the largest dark intensity has CD ≈ –1, and background-level bright and dark has CD ≈ 0. This metric is preferred to polarization anisotropy,[118] which does not take into account the intensity of the local background. Variable background intensity across the field-of-view is inherent to DIC and the contrast difference metric corrects for these variations.

We examined the near-field data recorded by the FDTD simulation to identify the origin of the contrast inversion at different wavelengths. For this analysis, we used a test particle with dimensions 25 × 75 nm in an oil ($n = 1.525$) environment, typical of a DIC microscopy sample. **Figure 5.8a** shows the scattering cross section for the longitudinal mode of the AuNR with its peak at 830 nm. The near field of the nanoparticle was recorded by both a 1D cross-section along the x-axis, and the 2D plane corresponding to $z = 0$. The particle was illuminated with a broadband plane wave propagating in the z-direction, polarized along the x-axis. We examined the amplitude

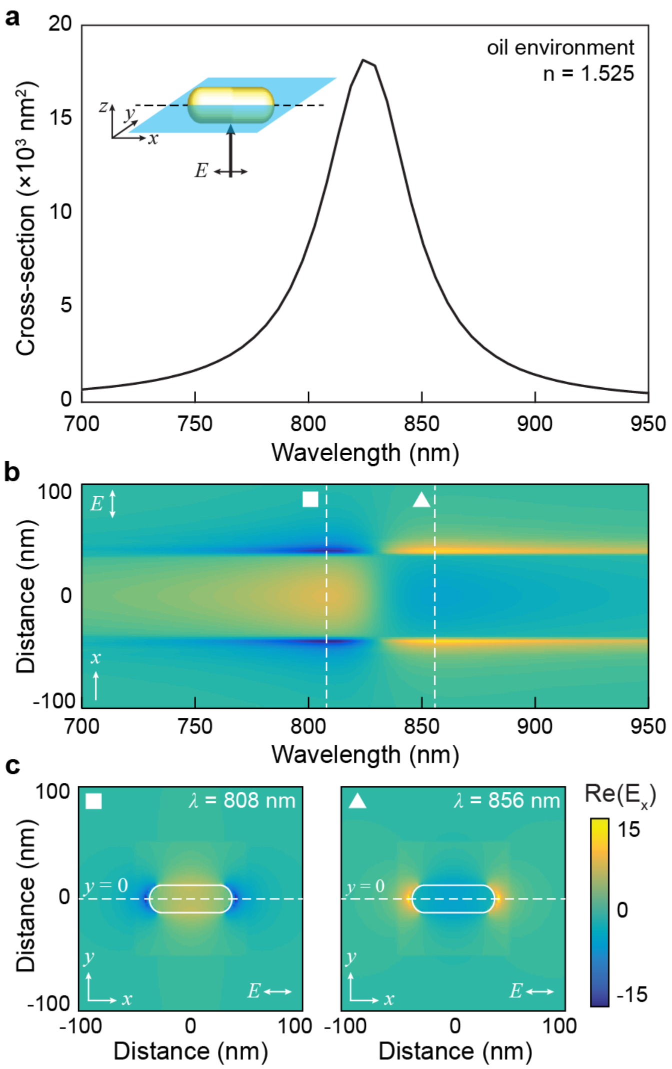


Figure 5.8: FDTD near-field analysis of DIC contrast inversion. (a) Scattering simulation for the longitudinal mode of AuNR (25 × 75 nm in oil) with a scheme of the 1D and 2D monitor locations (inset). (b) E-field amplitude (1D monitor) as a function of wavelength for the AuNR. (c) 2D cross-sections of electric field amplitude at two wavelengths on the blue (square, λ_{DIC} = 808 nm) and red (triangle, λ_{DIC} = 856 nm) sides of the LSP wavelength.

of Re(E_x) as recorded by the 1D monitor as a function of wavelength (**Figure 5.8b**). Only the real-part of the electric field was visualized since DIC depends on propagating waves into the farfield to generate an image of the NP. **Figure 5.8b** shows an inversion of the amplitude of Re(E_x) as a function of wavelength across the LSP resonance peak. For $\lambda = 808$ nm $< \lambda_{LSP}$, (**Figure 5.8c, left**) Re(E_x) was negative outside the particle, indicating a dipolar charge distribution with positive charge on the left surface of the particle and negative charge on the right. At $\lambda = 856$ nm $> \lambda_{LSP}$, however, the sign of Re(E_x) flipped, corresponding to a reversal of the charge distribution. For polarization in the perpendicular direction (along the *y*-axis), the scattering intensity was 3 orders of magnitude weaker, and Re(E_y) did not show an inversion in amplitude (**Figure 5.9**). The sign of Re(E_y) remained constant since the transverse LSP wavelength of the AuNR is below 600 nm and far from our test wavelengths. These near-field studies provided evidence that the DIC image contrast inversion is due to the change in sign of the local near-field and surface charge distribution on either side of the LSP wavelength.

We examined the DIC contrast at two wavelengths ($\lambda^1_{DIC} = 640$ nm; $\lambda^2_{DIC} = 750$ nm) for AuNR with different sizes and therefore different LSP resonance wavelengths: $\lambda^I_{LSP} = 609$ nm, $\lambda^{II}_{LSP} = 694$ nm, and $\lambda^{III}_{LSP} = 798$ nm (**Figure 5.10a**). These three particles were chosen since they represent the three possible relationships between λ_{LSP} and λ_{DIC}. Particle **I** was always blue of the DIC wavelengths ($\boldsymbol{\lambda^{I}_{LSP}} < \lambda^1_{DIC} < \lambda^2_{DIC}$), **II** was between them ($\lambda^1_{DIC} < \boldsymbol{\lambda^{II}_{LSP}} < \lambda^2_{DIC}$), and **III** was always on the red side ($\lambda^1_{DIC} < \lambda^2_{DIC} < \boldsymbol{\lambda^{III}_{LSP}}$). DIC images for the three particles show the same relationship between image contrast and LSP wavelength. For $\lambda_{LSP} < \lambda_{DIC}$, DIC images showed fully dark contrast when the nanoparticle rotation was aligned to the shear axis of the microscope (0°), while bright contrast was observed at this orientation when $\lambda_{LSP} > \lambda_{DIC}$ (**Figure 5.10b**). We

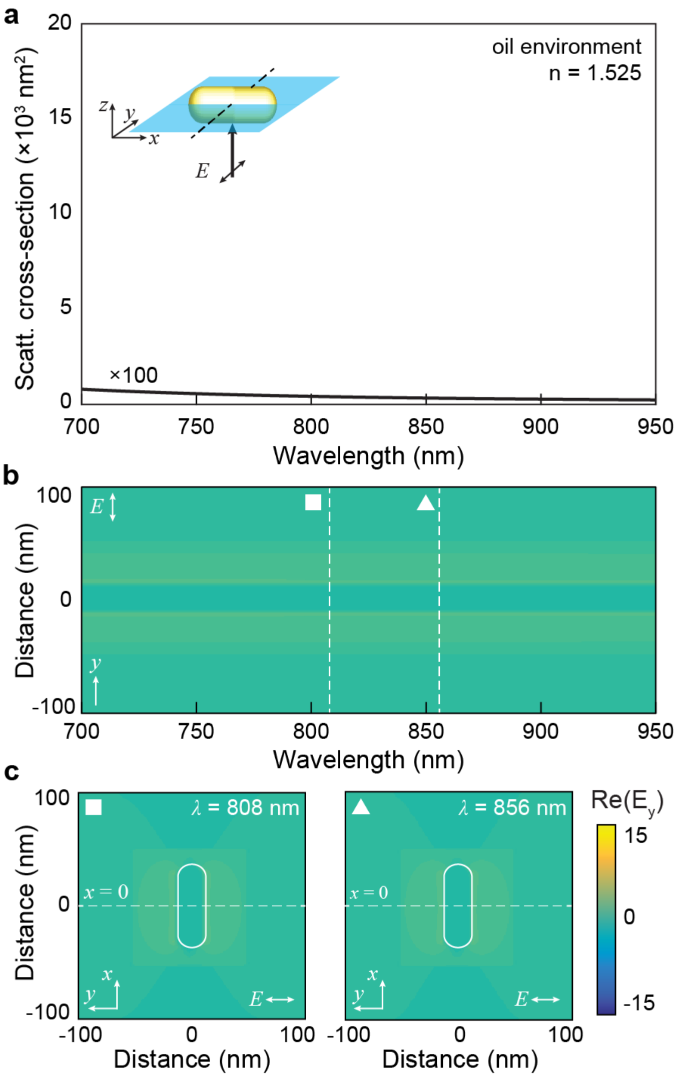


Figure 5.9: FDTD near-field analysis of DIC contrast inversion. (a) Scattering simulation for the transverse orientation of AuNR (25 × 75 nm in oil) with a scheme of the 1D and 2D monitor locations (inset). (b) E-field amplitude (1D monitor) as a function of wavelength for the AuNR. (c) 2D cross-sections of electric field amplitude at two wavelengths (square, λ_{DIC} = 808 nm) and red (triangle, λ_{DIC} = 856 nm).

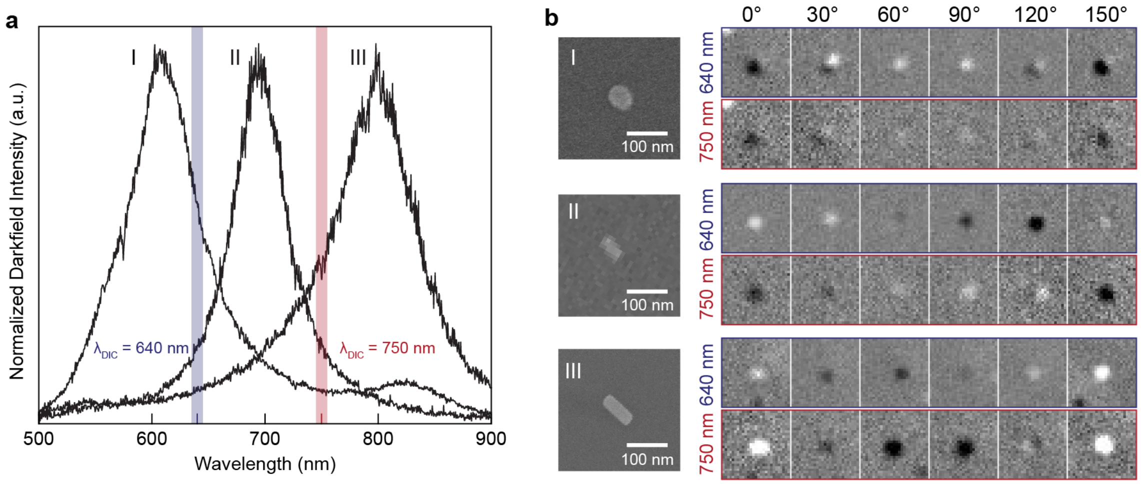


Figure 5.10: DIC response of AuNR with three different resonance locations. (a) The measured scattering spectra (normalized) of three individual AuNR. (b) SEM of the AuNRs aligned to $\theta = 0°$ and DIC image sets for λ_{DIC} = 640 nm (blue border) and λ_{DIC} = 750 nm (red border) for $\theta = 0°$ to 150°.

also observed a decrease in overall signal when λ_{LSP} was far from λ_{DIC}, such as for **I** at 750 nm and **III** at 640 nm.

We expanded the rotation study of these three AuNR to include three additional DIC wavelengths 600 nm, 680 nm, and 700 nm. The full rotation series (0°–180° by 10° increments) of **I** is shown in **Figure 5.11**. Contrast inversion was visible between the 600-nm and 640-nm series, consistent with the 609-nm LSP resonance wavelength. The remaining angles all showed the same contrast difference, and similar image patterns. The rotation series of DIC images for **II** did not show a sudden shift in the image contrast as seen for **I** (**Figure 5.12**). Comparisons between image contrast for λ_{DIC} = 600 nm, 640 nm and 680 nm did not reveal significant differences in image patterns or contrast difference. The contrast was inverted at λ_{DIC} = 750 nm as seen in **Figure 5.10**, but the images and contrast difference at λ_{DIC} = 700 nm showed only a partial inversion. For example, at 70° (160°), the image contrast progressed from fully bright (dark) at λ_{DIC} = 680 nm, to partial/dim contrast at λ_{DIC} = 700 nm, then to fully dark (bright) at λ_{DIC} = 750 nm. The origin for this partial shift is currently unknown, but has only been observed when $\lambda_{DIC} \approx \lambda_{LSP}$. Finally, the DIC images and contrast difference for **III** are displayed in **Figure 5.13**. For this particle, $\lambda_{LSP} > \lambda_{DIC}$ for all λ_{DIC}, and no inversion was observed. The only effect of changing DIC wavelength was the decrease in overall signal to noise ratio for blue wavelengths, far from the LSP wavelength.

5.3 Experimental methods

5.3.1 FDTD simulations of DIC microscopy

Simulations were performed with commercially available software (FDTD Solutions, Lumerical Inc.). FDTD regions has spans of 12 μm in *x* and *y* with periodic boundary conditions, and 2 μm in *z* with PML boundaries. A plane wave source (broadband or single frequency)

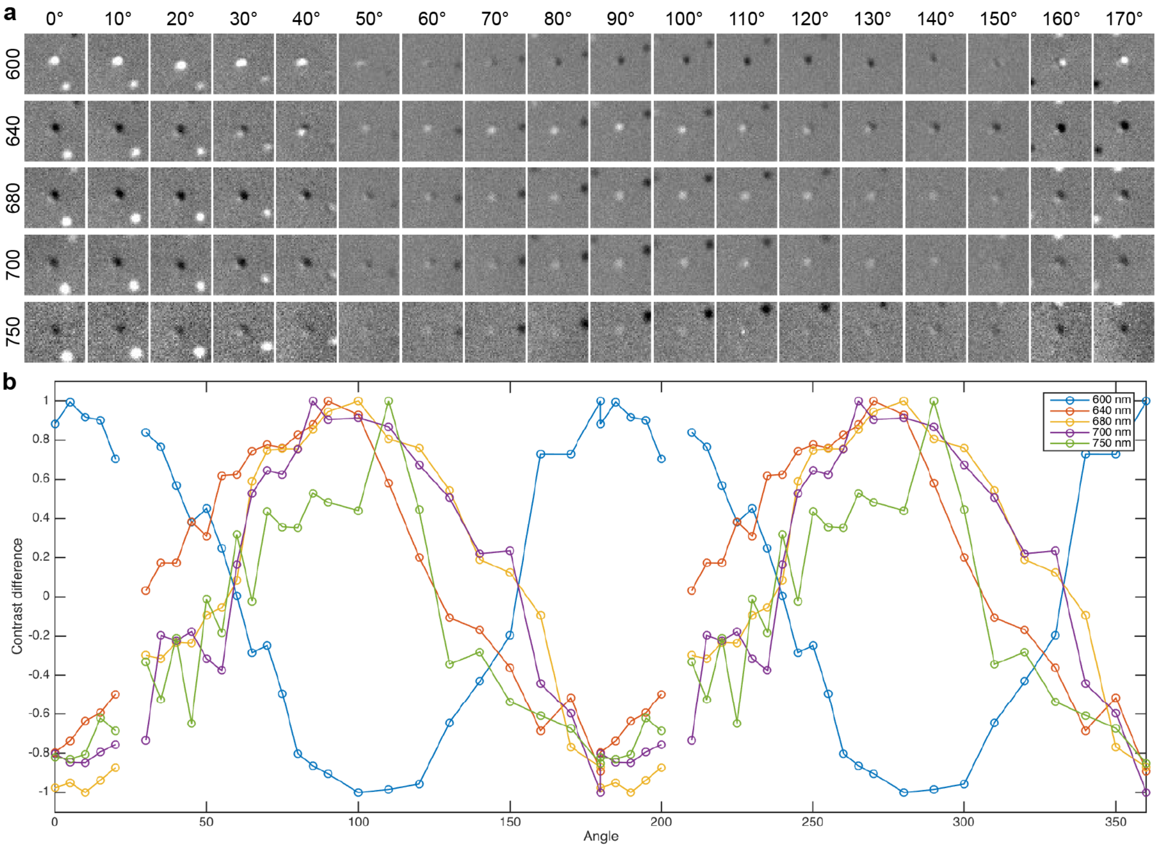


Figure 5.11: Rotational correlation study for AuNR **I**. (a) DIC images (2.6-µm width) at 5 DIC wavelengths from 600 nm to 750 nm and 18 rotational angles from 0° to 180°. (b) contrast difference calculated from the DIC images in (a).

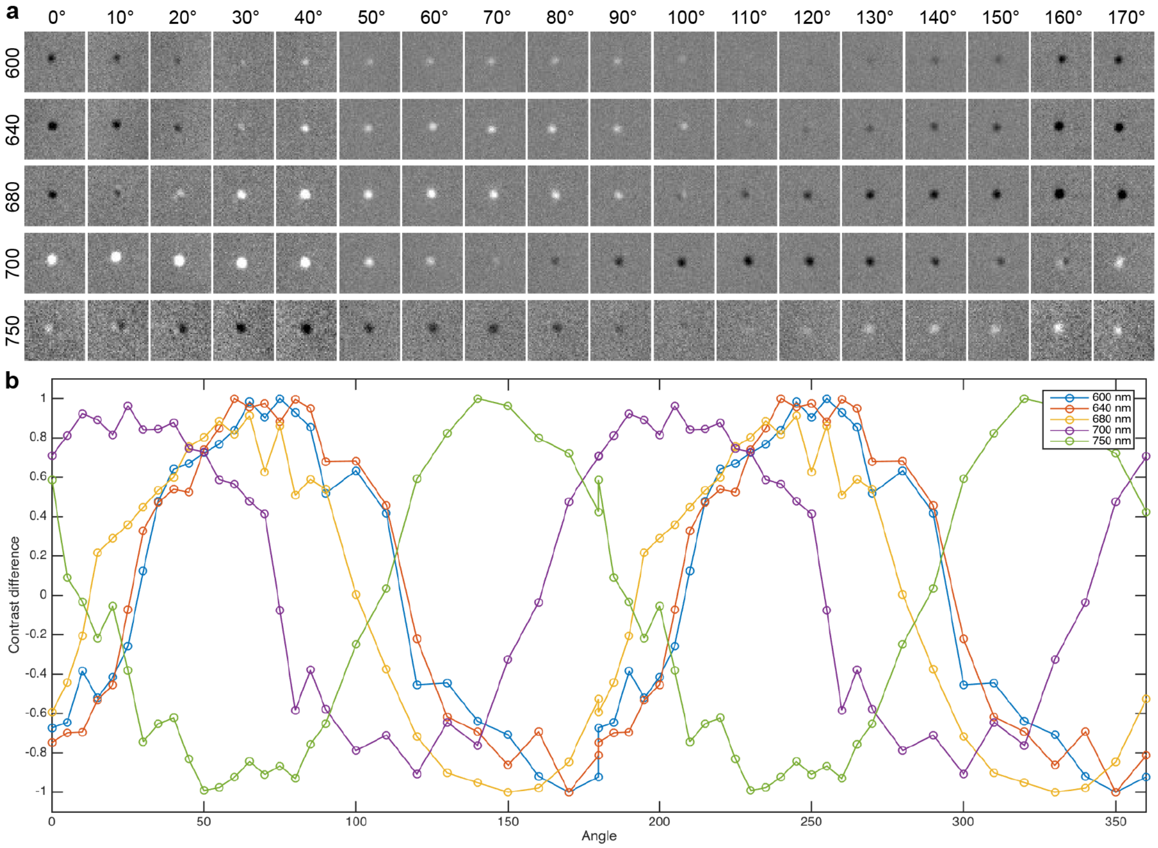


Figure 5.12: Rotational correlation study for AuNR **II**. (a) DIC images (2.6-µm width) at 5 DIC wavelengths from 600 nm to 750 nm and 18 rotational angles from 0° to 180°. (b) contrast difference calculated from the DIC images in (a).

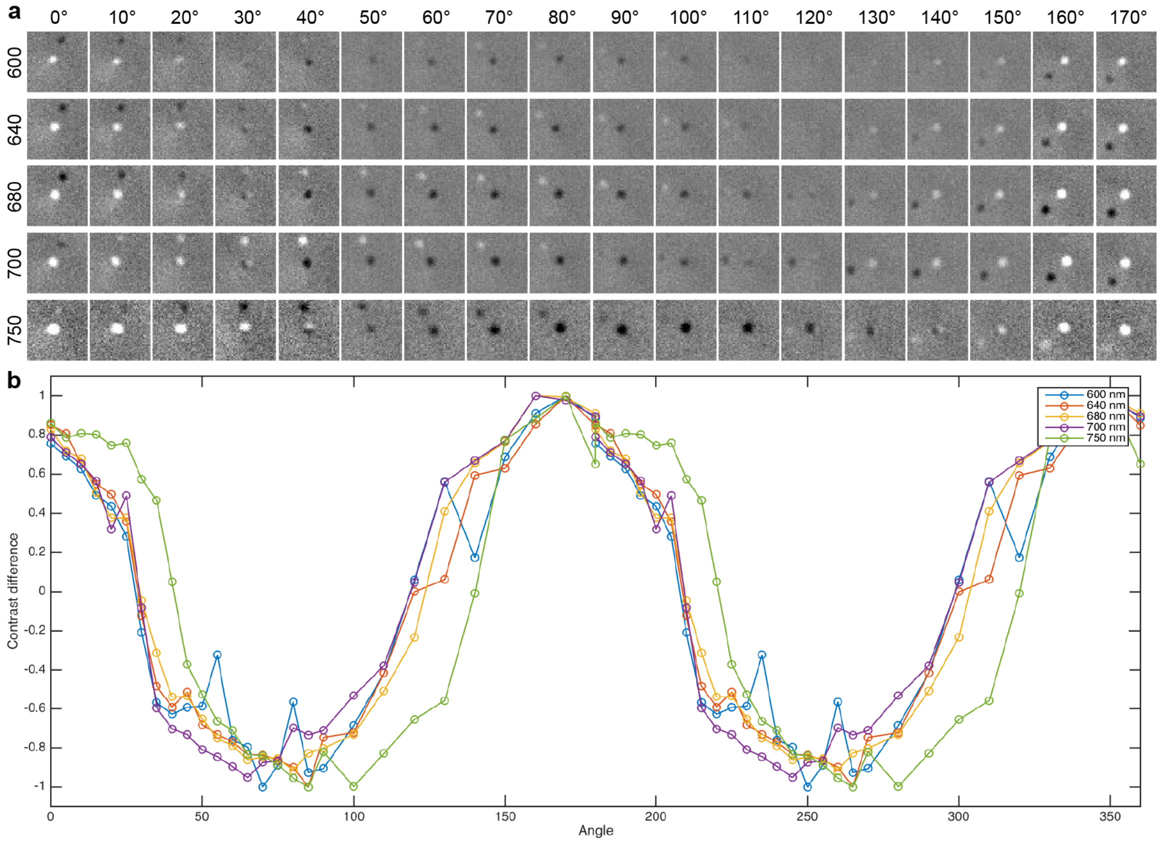


Figure 5.13: Rotational correlation study for AuNR **III**. (a) DIC images (2.6-µm width) at 5 DIC wavelengths from 600 nm to 750 nm and 18 rotational angles from 0° to 180°. (b) contrast difference calculated from the DIC images in (a).

illuminated the AuNR from below. The AuNR was designed as the union of two spheres and a cylinder having dimensions taken from SEM images and material properties defined by a built-in model based on data from Johnson and Christy.[101] A mesh override region around the particle forced a 2-nm mesh in all dimensions. A 2D monitor recorded the full electric and magnetic fields 50 nm above the sample. A sweep object was used to generate the two simulations by changing the particle position and rotating the source polarization. A second sweep rotated the AuNR for the rotational studies. Results from the 2D monitor were processed in a Lumerical script, found in **Appendix D**:

5.3.2 *AuNR sample preparation*

Reference markers are created on #1.5 coverslips by shadow deposition of 5-nm Cr through an indexed TEM reference grid (Ted Pella). A 10x diluted solution of bare AuNR (40 × 92, 25 × 75, Nanopartz Inc.) are drop cast on the coverslips for one minute and then rinsed with deionized water and dried with nitrogen gas. The coverslip is taped to a glass slide either with an air gap or a drop of $n = 1.52$ immersion oil in the gap.

5.3.3 *DF imaging*

DF imaging was done on an inverted Nikon TE2000-U with an oil immersion 100× objective (variable NA = 0.7–1.3) and either an air darkfield condenser (NA = 0.8–0.95) or an oil darkfield condenser (NA = 1.2–1.5). The light source was an unfiltered, unpolarized 100 W tungsten-halogen lamp. Transmitted light was sent to a spectrometer (Acton SP2300, Princeton Instruments) with a liquid nitrogen cooled CCD (Princeton Instruments).

5.3.4 *DIC imaging*

DIC was done on an inverted Nikon TE2000-E with an oil immersion, NA = 1.4 condenser, and an oil immersion NA = 1.49, APO, TIRF, 100× objective. The light source was a 100 W unfiltered, unpolarized tungsten-halogen lamp. The light passed through a band pass filter (hard-coated OD4, center wavelength = 600, 640, 680, 700, 750 nm with 10-nm bandwidth, Edmund Optics), a dé Senarmont compensator, one Nomarski prism, the condenser, sample, objective, second Nomarski prism, analyzer, then hit the detector (Andor Zyla 4.2 sCMOS). Metamorph software was used to collect the image files, and they were processed with ImageJ or custom MATLAB scripts.

5.3.5 SEM imaging

SEM images were obtained on untreated AuNR-on-glass samples with a FEI Quanta ESEM. Low-vacuum mode (1.2 torr) was used to enable imaging of the insulating samples.

5.4 Summary

We have shown FDTD simulations of DIC microscopy of AuNR. These simulations used parameters for the optics and nanoparticles directly from experimental conditions and produced DIC images that agreed with experimental results. We verified the expected mechanism for generating bright and dark image contrast by using FDTD simulations to determine the phase of the intermediate beams. We discovered DIC image contrast inversion of AuNR as a function of wavelength and analyzed the electric near-field distribution to identify its origin. Finally, we showed that the contrast inversion phenomenon is independent of resonance wavelength by measuring and simulating DIC images from several different sizes of AuNR.

REFERENCES

1. Atwater, H. A.; Polman, A., Plasmonics for improved photovoltaic devices. *Nature Materials* **2010,** *9* (3), 205-213.

2. Ferry, V. E.; Munday, J. N.; Atwater, H. A., Design considerations for plasmonic photovoltaics. *Advanced Materials* **2010,** *22* (43), 4794-4808.

3. Haynes, C. L.; Van Duyne, R. P., *Plasmon-sampled surface-enhanced raman excitation spectroscopy†*. American Chemical Society: 2003; Vol. 107, p 7426-7433.

4. Stewart, M. E.; Mack, N. H.; Malyarchuk, V.; Soares, J. A. N. T.; Lee, T. W.; Gray, S. K.; Nuzzo, R. G.; Rogers, J. A., Quantitative multispectral biosensing and 1D imaging using quasi-3D plasmonic crystals. *Proceedings of the National Academy of Sciences of the United States of America* **2006,** *103* (46), 17143-17148.

5. Gao, H.; Yang, J.-C.; Lin, J. Y.; Stuparu, A. D.; Lee, M. H.; Mrksich, M.; Odom, T. W., Using the angle-dependent resonances of molded plasmonic crystals to improve the sensitivities of biosensors. *Nano Letters* **2010,** *10* (7), 2549-2554.

6. Lin, J. Y.; Stuparu, A. D.; Huntington, M. D.; Mrksich, M.; Odom, T. W., Nanopatterned substrates increase surface sensitivity for real-time biosensing. *Journal of Physical Chemistry C* **2013,** *117* (10), 5286-5292.

7. Kawata, S.; Inouye, Y.; Verma, P., Plasmonics for near-field nano-imaging and superlensing. *Nature Photonics* **2009,** *3* (7), 388-394.

8. Barnes, W. L.; Dereux, A.; Ebbesen, T. W., Surface plasmon subwavelength optics. *Nature* **2003,** *424* (6950), 824-830.

9. Liu, Z.; Lee, H.; Xiong, Y.; Sun, C.; Zhang, X., Far-field optical hyperlens magnifying sub-diffraction-limited objects. *Science* **2007,** *315* (5819), 1686.

10. Fang, N.; Lee, H.; Sun, C.; Zhang, X., Sub–diffraction-limited optical imaging with a silver superlens. *Science* **2005,** *308* (5721), 534.

11. Oulton, R. F.; Sorger, V. J.; Zentgraf, T.; Ma, R.-M.; Gladden, C.; Dai, L.; Bartal, G.; Zhang, X., Plasmon lasers at deep subwavelength scale. *Nature* **2009,** *461* (7264), 629.

12. Zhou, W.; Dridi, M.; Suh, J. Y.; Kim, C. H.; Co, D. T.; Wasielewski, M. R.; Schatz, G. C.; Odom, T. W., Lasing action in strongly coupled plasmonic nanocavity arrays. *Nature Nanotechnology* **2013,** *8* (7), 506-511.

13. Rotz, M. W.; Culver, K. S. B.; Parigi, G.; MacRenaris, K. W.; Luchinat, C.; Odom, T. W.; Meade, T. J., High relaxivity gd(iii)–DNA gold nanostars: Investigation of shape effects on proton relaxation. *ACS Nano* **2015,** *9* (3), 3385-3396.

14. Dickerson, E. B.; Dreaden, E. C.; Huang, X.; El-Sayed, I. H.; Chu, H.; Pushpanketh, S.; McDonald, J. F.; El-Sayed, M. A., Gold nanorod assisted near-infrared plasmonic photothermal therapy (pptt) of squamous cell carcinoma in mice. *Cancer Letters 269* (1), 57-66.

15. Huang, X.; El-Sayed, I. H.; Qian, W.; El-Sayed, M. A., Cancer cell imaging and photothermal therapy in the near-infrared region by using gold nanorods. *J. Am. Chem. Soc* **2006,** *128* (6), 2115-2120.

16. Dam, D. H. M.; Lee, R. C.; Odom, T. W., Improved in vitro efficacy of gold nanoconstructs by increased loading of g-quadruplex aptamer. *Nano Letters* **2014,** *14* (5), 2843-2848.

17. Maier, S., Plasmonics: Fundamentals and applications. *Springer Science* **2007**.

18. Raether, H., *Surface-plasmons on smooth and rough surfaces and on gratings*. Springer-Verlag Berlin: 1988.

19. Kelly, K. L.; Coronado, E.; Zhao, L. L.; Schatz, G. C., The optical properties of metal nanoparticles: The influence of size, shape, and dielectric environment. *Journal of Physical Chemistry B* **2003,** *107* (3), 668-677.

20. Link, S.; El-Sayed, M. A., Shape and size dependence of radiative, non-radiative and photothermal properties of gold nanocrystals. *International Reviews in Physical Chemistry* **2000,** *19* (3), 409-453.

21. Jain, P. K.; El-Sayed, M. A., Plasmonic coupling in noble metal nanostructures. *Chemical Physics Letters* **2010,** *487* (4-6), 153-164.

22. Lee, K. S.; El-Sayed, M. A., Gold and silver nanoparticles in sensing and imaging: Sensitivity of plasmon response to size, shape, and metal composition. *The Journal of Physical Chemistry B* **2006**.

23. Sun, Y.; Xia, Y., Increased sensitivity of surface plasmon resonance of gold nanoshells compared to that of gold solid colloids in response to environmental changes. *Analytical Chemistry* **2002,** *74* (20), 5297-5305.

24. Hao, E.; Schatz, G. C., Electromagnetic fields around silver nanoparticles and dimers. *The Journal of Chemical Physics* **2004,** *120* (1), 357-366.

25. Crozier, K. B.; Sundaramurthy, A.; Kino, G. S.; Quate, C. F., Optical antennas: Resonators for local field enhancement. *Journal of Applied Physics* **2003,** *94* (7), 4632.

26. Hulteen, J. C.; Treichel, D. A.; Smith, M. T.; Duval, M. L.; Jensen, T. R.; Van Duyne, R. P., Nanosphere lithography: Size-tunable silver nanoparticle and surface cluster arrays. *Journal of Physical Chemistry B* **1999,** *103* (19), 3854-3863.

27. Orendorff, C. J.; Sau, T. K.; Murphy, C. J., Shape-dependent plasmon-resonant gold nanoparticles. *Small* **2006,** *2* (5), 636-639.

28. Brown, K. R.; Walter, D. G.; Natan, M. J., Seeding of colloidal au nanoparticle solutions. 2. Improved control of particle size and shape. *Chemical Physics Letters* **2000,** *12* (2), 306-313.

29. Link, S.; El-Sayed, M. A., Spectral properties and relaxation dynamics of surface plasmon electronic oscillations in gold and silver nanodots and nanorods. *Journal of Physical Chemistry B* **1999,** *103* (40), 8410-8426.

30. Gao, J.; Bender, C. M.; Murphy, C. J., Dependence of the gold nanorod aspect ratio on the nature of the directing surfactant in aqueous solution. *Langmuir* **2003,** *19* (21), 9065-9070.

31. Nikoobakht, B.; El-Sayed, M. A., Preparation and growth mechanism of gold nanorods (nrs) using seed-mediated growth method. *Chemistry of Materials* **2003,** *15* (10), 1957-1962.

32. Henzie, J.; Kwak, E.-S.; Odom, T. W., Mesoscale metallic pyramids with nanoscale tips. *Nano Letters* **2005,** *5* (7), 1199-1202.

33. Webb, J. A.; Erwin, W. R.; Zarick, H. F.; Aufrecht, J.; Manning, H. W.; Lang, M. J.; Pint, C. L.; Bardhan, R., Geometry-dependent plasmonic tunability and photothermal characteristics of multibranched gold nanoantennas. *Journal of Physical Chemistry C* **2014,** *118* (7), 3696-3707.

34. Xie, J.; Lee, J. Y.; Wang, D. I., Seedless, surfactantless, high-yield synthesis of branched gold nanocrystals in hepes buffer solution. *Chemistry of Materials* **2007,** *19* (11), 2823-2830.

35. Moore, G. E., Cramming more components onto integrated circuits, reprinted from electronics, volume 38, number 8, april 19, 1965, pp.114 ff. *IEEE Solid-State Circuits Society Newsletter* **2006,** *11* (5), 33-35.

36. Vieu, C.; Carcenac, F.; Pepin, A.; Chen, Y.; Mejias, M.; Lebib, A.; Manin-Ferlazzo, L.; Couraud, L.; Launois, H., Electron beam lithography: Resolution limits and applications. *Applied Surface Science* **2000,** *164*, 111-117.

37. Volkert, C. A.; Minor, A. M., Focused ion beam microscopy and micromachining. *MRS Bulletin* **2007,** *32* (5), 389-395.

38. Rogers, J. A.; Paul, K. E.; Jackman, R. J.; Whitesides, G. M., Using an elastomeric phase mask for sub-100 nm photolithography in the optical near field. *Applied Physics Letters* **1997,** *70* (20), 2658.

39. Rogers, J. A.; Paul, K. E.; Jackman, R. J.; Whitesides, G. M., Generating~ 90 nanometer features using near-field contact-mode photolithography with an elastomeric phase mask. *Journal of Vacuum Science and Technology B* **1998,** *16*, 59-68.

40. Aizenberg, J.; Rogers, J. A.; Paul, K. E.; Whitesides, G. M., Imaging profiles of light intensity in the near field: Applications to phase-shift photolithography. *Applied Optics* **1998,** *37* (11), 2145-2152.

41. Wolf, A. J.; Hauser, H.; Kübler, V.; Walk, C.; Höhn, O.; Bläsi, B., Origination of nano- and microstructures on large areas by interference lithography. *Microelectronic Engineering* **2012,** *98*, 293-296.

42. Odom, T. W.; Love, J.; Wolfe, D.; Paul, K.; Whitesides, G., Improved pattern transfer in soft lithography using composite stamps. *Langmuir* **2002,** *18* (13), 5314-5320.

43. Odom, T. W.; Thalladi, V. R.; Love, J. C.; Whitesides, G. M., Generation of 30−50 nm structures using easily fabricated, composite pdms masks. *Journal of the American Chemical Society* **2002,** *124* (41), 12112-12113.

44. Henzie, J.; Lee, M. H.; Odom, T. W., Multiscale patterning of plasmonic metamaterials. *Nature Nanotechnology* **2007,** *2* (9), 549.

45. Gao, H.; Henzie, J.; Lee, M. H.; Odom, T. W., Screening plasmonic materials using pyramidal gratings. *Proceedings of the National Academy of Sciences of the United States of America* **2008,** *105* (51), 20146-20151.

46. Zhou, W.; Odom, T. W., Tunable subradiant lattice plasmons by out-of-plane dipolar interactions. *Nature Nanotechnology* **2011,** *6* (7), 423-427.

47. Zhou, W.; Hua, Y.; Huntington, M. D.; Odom, T. W., Delocalized lattice plasmon resonances show dispersive quality factors. *Journal Of Physical Chemistry Letters* **2012,** *3* (10), 1381-1385.

48. Yang, C.-R.; Chen, P.-Y.; Chiou, Y.-C.; Lee, R.-T., Effects of mechanical agitation and surfactant additive on silicon anisotropic etching in alkaline koh solution. *Sensors and Actuators A: Physical* **2005,** *119* (1), 263-270.

49. Henry, M. D.; Walavalkar, S.; Homyk, A.; Scherer, A., Alumina etch masks for fabrication of high-aspect-ratio silicon micropillars and nanopillars. *Nanotechnology* **2009,** *20* (25), 255305.

50. Lin, J. Y.; Hasan, W.; Yang, J.-C.; Odom, T. W., Optical properties of nested pyramidal nanoshells. *The Journal of Physical Chemistry C* **2010,** *114* (16), 7432-7435.

51. Yang, A.; Hoang, T. B.; Dridi, M.; Deeb, C.; Mikkelsen, M. H.; Schatz, G. C.; Odom, T. W., Real-time tunable lasing from plasmonic nanocavity arrays. *Nature Communications* **2015,** *6*, 6939.

52. Wood, R. W., On a remarkable case of uneven distribution of light in a diffraction grating spectrum. *Philosophical Magazine* **1902,** *4* (19-24), 396-402.

53. Rayleigh, On the dynamical theory of gratings. *Proceedings of the Royal Society of London Series a-Containing Papers of a Mathematical and Physical Character* **1907,** *79* (532), 399-416.

54. Rayleigh, L., Note on the remarkable case of diffraction spectra described by prof. Wood. *Philosophical Magazine* **1907,** *14* (79-84), 60-65.

55. Ebbesen, T. W.; Lezec, H. J.; Ghaemi, H. F.; Thio, T.; Wolff, P. A., Extraordinary optical transmission through sub-wavelength hole arrays. *Nature* **1998,** *391* (6668), 667-669.

56. Gao, H.; Zhou, W.; Odom, T. W., Plasmonic crystals: A platform to catalog resonances from ultraviolet to near-infrared wavelengths in a plasmonic library. *Advanced Functional Materials* **2010,** *20* (4), 529-539.

57. Lubin, S. M.; Hryn, A. J.; Huntington, M. D.; Engel, C. J.; Odom, T. W., Quasiperiodic moiré plasmonic crystals. *ACS Nano* **2013,** *7* (12), 11035-11042.

58. Chaudhari, K.; Pradeep, T., Spatiotemporal mapping of three dimensional rotational dynamics of single ultrasmall gold nanorods. *Scientific Reports* **2014,** *4*.

59. Sweeney, C. M.; Nehl, C. L.; Hasan, W.; Liang, T.; Eckermann, A. L.; Meade, T. J.; Odom, T. W., Three-channel spectrometer for wide-field imaging of anisotropic plasmonic nanoparticles. *Journal of Physical Chemistry C* **2011,** *115* (32), 15933-15937.

60. Nan, X.; Sims, P. A.; Xie, X. S., Organelle tracking in a living cell with microsecond time resolution and nanometer spatial precision. *Chemphyschem : a European journal of chemical physics and physical chemistry* **2008,** *9* (5), 707-712.

61. Dürig, U.; Pohl, D. W.; Rohner, F., Near-field optical-scanning microscopy. *Journal of Applied Physics* **1986,** *59* (10), 3318.

62. Taflove, A.; Hagness, S. C., *Computational electrodynamics: The finite-difference time-domain method*. Artech House: 2005.

63. Zou, S. L.; Janel, N.; Schatz, G. C., Silver nanoparticle array structures that produce remarkably narrow plasmon lineshapes. *Journal of Chemical Physics* **2004,** *120* (23), 10871-10875.

64. Genet, C.; Ebbesen, T. W., Light in tiny holes. *Nature* **2007,** *445* (7123), 39-46.

65. Barnes, W. L.; Preist, T. W.; Kitson, S. C.; Sambles, J. R.; Cotter, N. P. K.; Nash, D. J., Photonic gaps in the dispersion of surface-plasmons on gratings. *Physical Review B* **1995,** *51* (16), 11164-11167.

66. Chen, Y. C.; Chang, Y. T.; Chen, H. H.; Chuang, F. T.; Cheng, C. H.; Lee, S. C., Enhanced transmission of higher order plasmon modes with random au nanoparticles in periodic hole arrays. *IEEE Photonics Technology Letters* **2013,** *25* (1), 47-50.

67. Gao, H.; Hyun, J. K.; Lee, M. H.; Yang, J.-C.; Lauhon, L. J.; Odom, T. W., Broadband plasmonic microlenses based on patches of nanoholes. *Nano Letters* **2010,** *10* (10), 4111-4116.

68. Huntington, M. D.; Lauhon, L. J.; Odom, T. W., Subwavelength lattice optics by evolutionary design. *Nano Letters* **2014,** *14* (12), 7195-7200.

69. Yang, A.; Li, Z.; Knudson, M. P.; Hryn, A. J.; Wang, W.; Aydin, K.; Odom, T. W., Unidirectional lasing from template-stripped two-dimensional plasmonic crystals. *ACS Nano* **2015,** *9* (12), 11582-11588.

70. Zhou, W.; Gao, H.; Odom, T. W., Toward broadband plasmonics: Tuning dispersion in rhombic plasmonic crystals. *ACS Nano* **2010,** *4* (2), 1241-1247.

71. Hiller, H., The crystallographic restriction in higher dimensions. *Acta Crystallogr. A* **1985,** *41* (6), 541-544.

72. Penrose, R., Pentaplexity a class of non-periodic tilings of the plane. *The Mathematical Intelligencer* **1979,** *2* (1), 32-37.

73. Socolar, J. E. S., Simple octagonal and dodecagonal quasicrystals. *Physical Review B* **1989,** *39* (15), 10519-10551.

74. Przybilla, F.; Genet, C.; Ebbesen, T. W., Enhanced transmission through penrose subwavelength hole arrays. *Applied Physics Letters* **2006,** *89* (12).

75. Agrawal, A.; Matsui, T.; Vardeny, Z. V.; Nahata, A., Extraordinary optical transmission through metallic films perforated with aperture arrays having short-range order. *Optics Express* **2008,** *16* (9), 6267-6273.

76. Pacifici, D.; Lezec, H. J.; Sweatlock, L. A.; Walters, R. J.; Atwater, H. A., Universal optical transmission features in periodic and quasiperiodic hole arrays. *Optics Express* **2008,** *16* (12), 9222-9238.

77. Lubin, S. M.; Zhou, W.; Hryn, A. J.; Huntington, M. D.; Odom, T. W., High-rotational symmetry lattices fabricated by moiré nanolithography. *Nano Letters* **2012,** *12* (9), 4948-4952.

78. Toh, K. K., Proceedings of spie. In *10th Annual Symp. on Microlithography*, SPIE: 1991; Vol. 1496, pp 27-53.

79. Levine, D.; Steinhardt, P. J., Quasicrystals - a new class of ordered structures. *Physical Review Letters* **1984,** *53* (26), 2477-2480.

80. Huntington, M. D.; Odom, T. W., A portable, benchtop photolithography system based on a solid-state light source. *Small* **2011,** *7* (22), 3144-3147.

81. Munday, J. N.; Atwater, H. A., Large integrated absorption enhancement in plasmonic solar cells by combining metallic gratings and antireflection coatings. *Nano Letters* **2011,** *11* (6), 2195-2201.

82. Bauer, C.; Kobiela, G.; Giessen, H., Optical properties of two-dimensional quasicrystalline plasmonic arrays. *Physical Review B* **2011,** *84* (19), 193104.

83. Odom, T. W.; Gao, H. W.; McMahon, J. M.; Henzie, J.; Schatz, G. C., Plasmonic superlattices: Hierarchical subwavelength hole arrays. *Chemical Physics Letters* **2009,** *483* (4-6), 187-192.

84. Guo, J.; Leong, H., Mode splitting of surface plasmon resonance in super-period metal nanohole array gratings. *Applied Physics Letters* **2012,** *101* (24), 241115.

85. Zentgraf, T.; Christ, A.; Kuhl, J.; Gippius, N. A.; Tikhodeev, S. G.; Nau, D.; Giessen, H., Metallodielectric photonic crystal superlattices: Influence of periodic defects on transmission properties. *Physical Review B* **2006,** *73* (11), 115103.

86. Kichin, G.; Weiss, T.; Gao, H.; Henzie, J.; Odom, T. W.; Tikhodeev, S. G.; Giessen, H., Metal–dielectric photonic crystal superlattice: 1D and 2D models and empty lattice approximation. *Physica B-Condensed Matter* **2012,** *407* (20), 4037-4042.

87. Xia, Y. N.; Whitesides, G. M., Soft lithography. *Annual Review of Materials Science* **1998,** *28*, 153-184.

88. Barnes, W. L.; Murray, W. A.; Dintinger, J.; Devaux, E.; Ebbesen, T. W., Surface plasmon polaritons and their role in the enhanced transmission of light through periodic arrays of subwavelength holes in a metal film. *Physical Review Letters* **2004,** *92* (10), 107401.

89. Abbas, M. N.; Chang, Y.-C.; Shih, M. H., Plasmon-polariton band structures of asymmetric t-shaped plasmonic gratings. *Optics Express* **2010,** *18* (3), 2509-2514.

90. Chen, Y.-B.; Ho, I.-C.; Chiu, F.-C.; Chang, C.-S., In-plane scattering patterns from a complex dielectric grating at the normal and oblique incidence. *JOSA A* **2014,** *31* (4), 879-885.

91. Chen, Y.-C.; Hsiao, H.-H.; Lu, C.-T.; Chang, Y.-T.; Chen, H.-H.; Chuang, F.-T.; Huang, S.-Y.; Yu, C.-W.; Chang, H.-C.; Lee, S.-C., Effect of paired apertures in a periodic hole array on higher order plasmon modes. *IEEE Photonics Technology Letters* **2012,** *24* (22), 2052-2055.

92. Uchida, K.; Kaneko, S.; Omi, S.; Hata, C.; Tanji, H.; Asahara, Y.; Ikushima, A. J.; Tokizaki, T.; Nakamura, A., Optical nonlinearities of a high concentration of small metal particles dispersed in glass: Copper and silver particles. *Journal of the Optical Society of America B* **1994,** *11* (7), 1236-1243.

93. Haraguchi, M.; Okamoto, T.; Inoue, T.; Nakagaki, M.; Koizumi, H.; Yamaguchi, K.; Lai, C.; Fukui, M.; Kamano, M.; Fujii, M., Linear and nonlinear optical phenomena of metallic nanoparticles. *IEEE Journal of Selected Topics in Quantum Electronics* **2008,** *14* (6), 1540-1551.

94. Yang, A.; Huntington, M. D.; Cardinal, M. F.; Masango, S. S.; Van Duyne, R. P.; Odom, T. W., Hetero-oligomer nanoparticle arrays for plasmon-enhanced hydrogen sensing. *ACS Nano* **2014,** *8* (8), 7639-7647.

95. Tabatabaei, M.; Najiminaini, M.; Davieau, K.; Kaminska, B.; Singh, M. R.; Carson, J. J. L.; Lagugné-Labarthet, F., Tunable 3D plasmonic cavity nanosensors for surface-enhanced raman spectroscopy with sub-femtomolar limit of detection. *ACS Photonics* **2015,** *2* (6), 150513111019000-759.

96. Suh, J. Y.; Kim, C. H.; Zhou, W.; Huntington, M. D.; Co, D. T.; Wasielewski, M. R.; Odom, T. W., Plasmonic bowtie nanolaser arrays. *Nano Letters* **2012,** *12* (11), 5769-5774.

97. McMahon, J. M.; Wang, Y.; Sherry, L. J.; Van Duyne, R. P.; Marks, L. D.; Gray, S. K.; Schatz, G. C., Correlating the structure, optical spectra, and electrodynamics of single silver nanocubes. *The Journal of Physical Chemistry C* **2009,** *113* (7), 2731-2735.

98. Suh, J. Y.; Huntington, M. D.; Kim, C. H.; Zhou, W.; Wasielewski, M. R.; Odom, T. W., Extraordinary nonlinear absorption in 3D bowtie nanoantennas. *Nano Letters* **2012,** *12* (1), 269-274.

99. Henson, J.; DiMaria, J.; Paiella, R., Influence of nanoparticle height on plasmonic resonance wavelength and electromagnetic field enhancement in two-dimensional arrays. *Journal of Applied Physics* **2009,** *106* (9), 093111.

100. Muskens, O. L.; Giannini, V.; Sánchez-Gil, J. A.; Gómez Rivas, J., Optical scattering resonances of single and coupled dimer plasmonic nanoantennas. *Optics Express* **2007,** *15* (26), 17736.

101. Johnson, P. B.; Christy, R. W., Optical constants of the noble metals. *Physical Review B* **1972,** *6* (12), 4370-4379.

102. Ross, B. M.; Wu, L. Y.; Lee, L. P., Omnidirectional 3D nanoplasmonic optical antenna array via soft-matter transformation. *Nano Letters* **2011,** *11* (7), 2590-2595.

103. Jain, P. K.; Lee, K. S.; El-Sayed, I. H.; El-Sayed, M. A., Calculated absorption and scattering properties of gold nanoparticles of different size, shape, and composition: Applications

in biological imaging and biomedicine. *Journal of Physical Chemistry B* **2006,** *110* (14), 7238-7248.

104. Peterson, J. J.; Nesbitt, D. J., Modified power law behavior in quantum dot blinking: A novel role for biexcitons and auger ionization. *Nano Letters* **2009,** *9* (1), 338-345.

105. Nirmal, M.; Dabbousi, B. O.; Bawendi, M. G.; Macklin, J. J.; Trautman, J. K.; Harris, T. D.; Brus, L. E., Fluorescence intermittency in single cadmium selenide nanocrystals. *, Published online: 31 October 1996; | doi:10.1038/383802a0* **1996,** *383* (6603), 802-804.

106. De Brabander, M.; Nuydens, R.; Geuens, G.; Moeremans, M.; De Mey, J., The use of submicroscopic gold particles combined with video contrast enhancement as a simple molecular probe for the living cell. *Cell Motility and the Cytoskeleton* **1986,** *6* (2), 105-113.

107. Sönnichsen, C.; Alivisatos, A. P., Gold nanorods as novel nonbleaching plasmon-based orientation sensors for polarized single-particle microscopy. *Nano Letters* **2005,** *5* (2), 301-304.

108. Leduc, C.; Si, S.; Gautier, J.; Soto-Ribeiro, M.; Wehrle-Haller, B.; Gautreau, A.; Giannone, G.; Cognet, L.; Lounis, B., A highly specific gold nanoprobe for live-cell single-molecule imaging. *Nano Letters* **2013**, 130306160416009-6.

109. Lasne, D.; Blab, G. A.; Berciaud, S.; Heine, M.; Groc, L.; Odom, T. W., Single nanoparticle photothermal tracking (snapt) of 5-nm gold beads in live cells. *Biophysical Journal* **2006,** *91* (12), 4598-4604.

110. Tanev, S.; Pond, J.; Paddon, P.; Tuchin, V. V.; Tanev, S.; Pond, J.; Paddon, P., A new 3D simulation method for the construction of optical phase contrast images of gold nanoparticle clusters in biological cells. *Advances in Optical Technologies* **2008,** *2008* (7), 1-9.

111. Tanev, S.; Tuchin, V.; Cheben, P.; Bock, P.; Schmid, J.; Janz, S.; Xu, D.; Lapointe, J.; Densmore, A.; Pond, J., Advances in the fdtd design and modeling of nano- and bio-photonics applications. *Photonics and Nanostructures - Fundamentals and Applications* **2011,** *9* (4), 315-327.

112. Sun, W.; Wang, G.; Fang, N.; Yeung, E. S., Wavelength-dependent differential interference contrast microscopy: Selectively imaging nanoparticle probes in live cells. *Analytical Chemistry* **2009,** *81* (22), 9203-9208.

113. Luo, Y.; Sun, W.; Gu, Y.; Wang, G.; Fang, N., Wavelength-dependent differential interference contrast microscopy: Multiplexing detection using nonfluorescent nanoparticles. *Analytical Chemistry* **2010,** *82* (15), 6675-6679.

114. Lee, H.; Dam, D. H. M.; Ha, J. W.; Yue, J.; Odom, T. W., Enhanced human epidermal growth factor receptor 2 degradation in breast cancer cells by lysosome-targeting gold nanoconstructs. *ACS Nano* **2015,** *9* (10), 9859-9867.

115. Jensen, S. A.; Day, E. S.; Ko, C. H.; Hurley, L. A.; Luciano, J. P.; Kouri, F. M.; Merkel, T. J.; Luthi, A. J.; Patel, P. C.; Cutler, J. I.; Daniel, W. L.; Scott, A. W.; Rotz, M. W.; Meade, T. J.; Giljohann, D. A.; Mirkin, C. A.; Stegh, A. H., Spherical nucleic acid nanoparticle conjugates as an rnai-based therapy for glioblastoma. *Science Translational Medicine* **2013,** *5* (209), 209ra152-209ra152.

116. Mieszawska, A. J.; Mulder, W. J. M.; Fayad, Z. A.; Cormode, D. P., Multifunctional gold nanoparticles for diagnosis and therapy of disease. *Molecular Pharmaceutics* **2013,** *10* (3), 831-847.

117. Wang, G.; Stender, A. S.; Sun, W.; Fang, N., Optical imaging of non-fluorescent nanoparticle probes in live cells. *Analyst* **2010,** *135* (2), 215-221.

118. Ha, J. W.; Sun, W.; Wang, G.; Fang, N., Differential interference contrast polarization anisotropy for tracking rotational dynamics of gold nanorods. *Chemical Communications* **2011,** *47* (27), 7743-7745.

119. Gu, Y.; Di, X.; Sun, W.; Wang, G.; Fang, N., Three-dimensional super-localization and tracking of single gold nanoparticles in cells. *Analytical Chemistry* **2012,** *84* (9), 4111-4117.

120. Tsunoda, M.; Isailovic, D.; Yeung, E. S., Real-time three-dimensional imaging of cell division by differential interference contrast microscopy. *J Microsc* **2008,** *232* (2), 207-11.

121. Wang, G.; Sun, W.; Luo, Y.; Fang, N., Resolving rotational motions of nano-objects in engineered environments and live cells with gold nanorods and differential interference contrast microscopy. *J. Am. Chem. Soc* **2010,** *132* (46), 16417-16422.

122. Gu, Y.; Sun, W.; Wang, G.; Fang, N., Single particle orientation and rotation tracking discloses distinctive rotational dynamics of drug delivery vectors on live cell membranes. *Journal of the American Chemical ...* **2011**.

123. Akbulut, O.; Mace, C. R.; Martinez, R. V.; Kumar, A. A.; Nie, Z.; patton, M. R.; whitesides, G. M., Separation of nanoparticles in aqueous multiphase systems through centrifugation. *Nano Letters* **2012,** *12* (8), 4060-4064.

124. Sharma, V.; Park, K.; Srinivasarao, M., Shape separation of gold nanorods using centrifugation. *Proceedings of the National Academy of Sciences of the United States of America* **2009,** *106* (13), 4981-4985.

125. Scarabelli, L.; Sánchez-Iglesias, A.; Pérez Juste, J.; Liz-Marzán, L. M., A “tips and tricks” practical guide to the synthesis of gold nanorods. *The Journal of Physical Chemistry Letters* **2015,** *6* (21), 4270-4279.

126. King, S. V.; Libertun, A.; Piestun, R.; Cogswell, C. J.; Preza, C., Quantitative phase microscopy through differential interference imaging. *Journal of Biomedical Optics* **2008,** *13* (2), 024020-024020-10.

127. Jiang, L.; Yin, T.; Dong, Z.; Liao, M.; Tan, S. J.; Goh, X. M.; Allioux, D.; Hu, H.; Li, X.; Yang, J. K. W.; Shen, Z., Accurate modeling of dark-field scattering spectra of plasmonic nanostructures. *ACS Nano* **2015,** *9* (10), 10039-10046.

128. Stender, A. S.; Wang, G.; Sun, W.; Fang, N., Influence of gold nanorod geometry on optical response. *ACS Nano* **2010,** *4* (12), 7667-7675.

APPENDIX A: MATLAB CODE FOR MOIRÉ STRUCTURE GENERATION

```matlab
function Generate_Moire
clc
%Prevent more than one occurance of Program to run
doesExist = findobj('name','Moire Structure Generator');
if ~isempty(doesExist)
    figure(doesExist);
    return
end
%% Initialize and Set Default Values
width = 2; %um
pixels = 2000+1; %1 nm resolution
% pixels = 1000;
maskChoices = {'Line','Square','Hex'};
typeOfMask = 'square';
periodicity = [400 400];
moireAngles = [0 90];
numSineFunctions = 1;
developPercent = 55;
randomShift = true;
shift = [0 0];
cmap = [0 0 0];
data = 0;
drawData = 0;
BWimage = false;
moireImage = [];
%% Create figure window.
mainFigure = figure('position',[200 200 800 500],...
    'name','Moire Structure Generator');
mainAxes = axes('outerposition',[.45 0 .5 1],...
    'dataAspectRatio',[1,1,1]);
%FTAxes = axes('outerposition',[.65 0 .35 1],...
%    'dataAspectRatio',[1,1,1]);
contextMenu = uicontextmenu;
subMenu1 = uimenu(contextMenu,...
    'label','Toggle B/W image',...
    'callback',@toggleBW);
subMenu2 = uimenu(contextMenu,...
    'label','Pop out figure',...,
    'callback',@popOutFigure);
inputPanel = uipanel(mainFigure,...
    'title','Parameters',...
    'pos',[.025 .025 .4 .95]);
uicontrol(inputPanel,...
    'style','text',...
    'units','norm',...
    'pos',[.05 .85 .4 .07],...
    'string','Box width [um]');
widthBox = uicontrol(inputPanel,...
    'style','edit',...
    'units','norm',...
    'pos',[.55 .85 .4 .1],...
    'callback',@updateData);
```

```
typeOfMaskPop = uicontrol(inputPanel,...
    'style','pop',...
    'units','norm',...
    'pos',[.05 .7 .35 .07],...
    'string','Default',...
    'callback',@updateData);
periodicityBox =  uicontrol(inputPanel,...
    'style','edit',...
    'units','norm',...
    'pos',[.4 .7 .45 .1],...
    'string','400',...
    'callback',@updateData);
uicontrol(inputPanel,...
    'style','text',...
    'units','norm',...
    'pos',[.85 .7 .1 .07],...
    'string','nm');
uicontrol(inputPanel,...
    'style','text',...
    'units','norm',...
    'pos',[.05 .55 .3 .08],...
    'string','Exposures/ angles [deg]');
exposureBox = uicontrol(inputPanel,...
    'style','edit',...
    'units','norm',...
    'pos',[.45 .55 .5 .1],...
    'string','1',...
    'callback',@updateData);
uicontrol(inputPanel,...
    'style','text',...
    'units','norm',...
    'pos',[.05 .4 .4 .08],...
    'string','Number of FS terms per line');
numSineFunctionsBox = uicontrol(inputPanel,...
    'style','edit',...
    'units','norm',...
    'pos',[.55 .4 .4 .1],...
    'callback',@updateData);
uicontrol(inputPanel,...
    'style','text',...
    'units','norm',...
    'pos',[.05 .27 .7 .06],...
    'string','Developing Percentage');
developPercentSlider = uicontrol(inputPanel,...
    'style','slider',...
    'units','norm',...
    'pos',[.05 .25 .7 .02],...
    'min',0,'max',100,...
    'callback',@updateData);
developPercentBox = uicontrol(inputPanel,...
    'style','edit',...
    'units','norm',...
    'pos',[.8 .24 .15 .1],...
    'callback',@updateData);
generateStructuresButton = uicontrol(inputPanel,...
    'units','norm',...
    'pos',[.05 .13 .4 .08],...
    'string','Generate',...
    'callback',@generateStructures);
outputObjectsButton = uicontrol(inputPanel,...
    'units','norm',...
    'pos',[.05 .05 .4 .08],...
    'string','Save Objects',...
    'callback',{@outputObjects,moireImage,pixels,width,mainAxes});
randomShiftCheck = uicontrol(inputPanel,...
    'style','check',...
    'units','norm',...
    'pos',[.5 .15 .45 .05],...
    'string','Random Shifts',...
    'callback',@updateData);
randomShiftBox = uicontrol(inputPanel,...
    'style','edit',...
    'units','norm',...
    'pos',[.5 .05 .45 .1],...
    'callback',@updateData);
drawnow;
updateFigure;
%% Nested functions for callbacks
    function updateData(obj,~)
        switch obj
            case widthBox
                width = str2double(get(obj,'string'));
%                 pixels = width*1000+1;
            case typeOfMaskPop
                typeOfMask = maskChoices{get(obj,'value')};
                updateData(exposureBox);
            case periodicityBox
                periodicity = eval(['[' get(obj,'string') ']'] );
                if length(periodicity) == 1
                    periodicity = repmat(periodicity,1,length(moireAngles));
                else
                    multiples =
strfind({'line','square','hex'},lower(typeOfMask));
                    multiples =
find(arrayfun(@(X)~isempty(multiples{X}),1:length(multiples)));
                    periodicity = repmat(periodicity,1,multiples);
                end
            case exposureBox
                exposures = eval(['[' get(obj,'string') ']']);
                if length(exposures) == 1
                    multiples =
strfind({'line','square','hex'},lower(typeOfMask));
                    multiples =
find(arrayfun(@(X)~isempty(multiples{X}),1:length(multiples)));
                    angles = linspace(0,180/multiples,exposures+1);
                    angles(end) = [];
                    moireAngles = angles;
                    for ii = 1:multiples-1
                        moireAngles = [moireAngles angles+ii*180/multiples];
                    end
        else
          multiples =
strfind({'line','square','hex'},lower(typeOfMask));
          multiples =
find(arrayfun(@(X)~isempty(multiples{X}),1:length(multiples)));
          for ii = 1:multiples-1
            exposures = [exposures exposures+ii*180/multiples];
          end
          moireAngles = exposures;
        end
        updateData(periodicityBox);
      case numSineFunctionsBox
        numSineFunctions = str2double(get(obj,'string'));
      case developPercentBox
        developPercent = str2double(get(obj,'string'));
      case developPercentSlider
        developPercent = get(obj,'value');
      case randomShiftCheck
        randomShift = get(obj,'value');
      case randomShiftBox
        input = get(obj,'string');
        shift = [0 0 eval(input)];
    end
    updateFigure
  end
  function updateFigure
    set(widthBox,'string',num2str(width));
    setTypeOfMask;
    function setTypeOfMask
      set(typeOfMaskPop,'string',maskChoices);
      drawnow;
      switch lower(typeOfMask)
        case 'line'
          set(typeOfMaskPop,'value',1);
        case 'square'
          set(typeOfMaskPop,'value',2);
        case 'hex'
          set(typeOfMaskPop,'value',3);
      end
    end
    set(numSineFunctionsBox,'string',num2str(numSineFunctions));
    setDevelopPercent;
    function setDevelopPercent
      set(developPercentBox,'string',sprintf('%.0f',developPercent));
      set(developPercentSlider,'value',developPercent);
      developPercent = round(developPercent);
      if BWimage
        colorVector = [zeros(1,developPercent-1) ones(1,100-
developPercent)];
      else
        colorVector = [zeros(1,developPercent-1) linspace(0,1,100-
developPercent)];
      end
      cmap = repmat(colorVector',1,3);
      colormap(mainAxes,cmap);
```

```
            moireImage = data>=developPercent/100;

set(outputObjectsButton,'callback',{@outputObjects,moireImage,pixels,width,ma
inAxes});
        end
        set(randomShiftCheck,'value',randomShift);
        if strcmpi(typeOfMask,'line') && ~randomShift
            set(randomShiftCheck,'string','Set Shifts [um]');
            set(randomShiftBox,'string',['[ ' sprintf('%.2f ',shift(3:end))
']']);
        else
            set(randomShiftCheck,'string','Random Shifts');
        end
    end
    function generateStructures(~,~)
        %identify the mask used
        switch lower(typeOfMask)
            case 'line'
                if randomShift
                    shift = rand(1,length(moireAngles))...
                        .*(periodicity/1000); %1-d
                    shift(1:2) = 0;
                else
                    shift = zeros(1,length(moireAngles));
                end
            case 'square'
                if randomShift
                    shift = rand(2,length(moireAngles)/2); %2-d
                    shift = [shift(1,:) shift(2,:)]...
                        .*(periodicity/1000);
                else
                    shift = zeros(1,length(moireAngles)*2);
                end
            case 'hex'
                if randomShift
                    shift = rand(2,length(moireAngles)/3); %2-d w/ hex
symmetry

                    shift = [shift(1,:) ...
                        shift(1,:)*cosd(60)+shift(2,:)*sind(60) ...
                        shift(1,:)*cosd(120)+shift(2,:)*sind(120)]...
                        .*(periodicity/1000);
                else
                    shift = zeros(1,length(moireAngles)*3);
                end
        end
        theta = moireAngles;
        numAngles = length(theta);
        [drawX,drawY,drawData] = makedata(pixels);
        save('image','drawData');
%         [X,Y,data] = makedata(pixels);
        h = surf(mainAxes,drawX,drawY,drawData,'LineStyle','none');
        axes(mainAxes);
        view(0,90), colormap(cmap),caxis([-
1,1]),xlabel('microns'),ylabel('microns')
        set(mainAxes,'dataaspectratio',[1 1 2/width]);
        set([mainAxes,h],'uicontextmenu',contextMenu);
        moireImage = data>=developPercent/100;

set(outputObjectsButton,'callback',{@outputObjects,moireImage,pixels,width,ma
inAxes});
        function [X,Y,data] = makedata(pixels)
            x = linspace(-width/2,width/2,pixels);
            y = linspace(-width/2,width/2,pixels);
            [X,Y] = meshgrid(x,y);
            for jj = 1:numAngles
                XX(:,:,jj) = X*cosd(theta(jj)) + Y*sind(theta(jj));
            end
            data = 0;
            for ii = 1:numSineFunctions
                for kk = 1:numAngles
                    func(:,:,kk) =
(1/(numAngles*ii)*cos(2*pi/(periodicity(kk)/1000)*ii*(XX(:,:,kk)-
shift(kk))));
                end
                data = data+sum(func,3);
            end
            %data = data-min(data(:));
            %data = data/max(data(:));
        end
    end
    function toggleBW(~,~)
        BWimage = ~BWimage;
        updateFigure;
    end
    function popOutFigure(~,~)
        newfig = figure;
        newaxes = copyobj(mainAxes,newfig);
        set(newaxes,'units','normalized','position',[.13 .11 .775 .815],...
            'dataAspectRatio',[1,1,2/width])
        colormap(newaxes,cmap);
    end
end
function outputObjects(~,~,moireImage,pixels,width,mainAxes)
if isempty(moireImage);
    return
end
[B,L,~,A] = bwboundaries(moireImage);
[objectList, objectNames] = sortObjects(A);
removedObjects = [];
% objectList = (max(objectList)-objectList+1); %to prepare for mesh order
pixelFactor = width/(pixels-1); %fence post
pixelShift = (pixels-1)/2 + 1;
colors = ['b' 'y' 'r' 'c' 'm' 'g'];
lineHandles = zeros(1,length(B));
hold(mainAxes,'all')
% figure; newAxes = axes;
% hold(newAxes,'all')
[~,~,~] = rmdir('ObjectFolder','s'); %outputs are to prevent warning that
directory exists
mkdir('ObjectFolder');
```

```
for jj = 1:length(B)
    if ~isempty(find(B{jj}(:)==1,1))||~isempty(find(B{jj}(:)==pixels,1))
        removedObjects(end+1) = jj;
        continue
    end
    clearvars temp temp2
    convertedObject = fliplr((B{jj}-pixelShift).*pixelFactor);
    temp(1,:) = convertedObject(1,:);
    counter = 1;
    for ii = 2:length(convertedObject)
        if temp(counter,1)==convertedObject(ii,1) || ...
                temp(counter,2)==convertedObject(ii,2)
            continue
        else
            counter = counter+1;
            temp(counter,1:2) = convertedObject(ii,:);
        end
    end
    temp(1,:) = temp(end,:);
    temp2(1,:) = convertedObject(end,:);
    counter = 1;
    for ii = length(convertedObject)-1:-1:1
        if temp2(counter,1)==convertedObject(ii,1) || ...
                temp2(counter,2)==convertedObject(ii,2)
            continue
        else
            counter = counter+1;
            temp2(counter,1:2) = convertedObject(ii,:);
        end
    end
    temp(1,:) = []; temp2(1,:) = [];
    averageTemp = (temp+flipud(temp2))/2;
    % smooth data
    averageTemp = [averageTemp(end-2:end,:); averageTemp;
averageTemp(1:3,:)];
    smoothData = [smooth(averageTemp(:,1)), smooth(averageTemp(:,2))];
    smoothData = smoothData(4:end-3,:);
    data = smoothData;
    %     eval(sprintf('%s = averageTemp;',objectNames(ii,:))); %save as
variables

eval(sprintf('dlmwrite(''ObjectFolder/%1$s.txt'',averageTemp);',objectNames(j
j,:))); %save as text files
    lineHandles(jj) = plot3(mainAxes,data(:,1),data(:,2),...
        ones(1,length(data)),colors(objectList(jj)));
    %
plot(newAxes,data(:,1),data(:,2),temp(:,1),temp(:,2),temp2(:,1),temp2(:,2));
    %     plot(newAxes,convertedObject(:,1),convertedObject(:,2),
temp(:,1),temp(:,2));
end
objectList(removedObjects) = [];
dlmwrite('ObjectFolder/objectList.txt',objectList);
dlmwrite('ObjectFolder/width.txt',width);
set(lineHandles,'uicontextmenu',get(mainAxes,'uicontextmenu'));
hold(mainAxes,'off')
```

```
% save('objects.mat','object*');
end
function [objectList,objectNames] = sortObjects(A)
objectList = zeros(length(A),1);
for object = 1:length(A)
    level = 1;
    findLevel(object);
    objectList(object) = level;
    objectNames(object,:) = sprintf('object%03d',object);
end
    function findLevel(object)
        nextObject = find(A(object,:));
        if nextObject
            level = level + 1;
            findLevel(nextObject);
        end
    end
end
```

APPENDIX B: MATLAB CODE FOR SPP MODE CALCULATIONS

```
function Rot_Stage_Proc_GUI
%% Prevent more than one occurrenece of Program to run
h = findobj('name','Rotational Stage Processing');
if ~isempty(h)
    UserData = get(h,'userdata');
    H = UserData{1};
    D = UserData{2};
    S = UserData{3};
    figure(H.figure.main)
    return
end
%% initialize data and set some defaults
D.info.version = '4.3.0';
D.info.lastUpdated = 'June 7, 2013';
try
    D.materials = load('matdata.mat');
catch exception
    if strfind('couldNotReadFile',exception.identifier)
        D.materials.matdate = '';
        D.materials.matdata = {};
    else
        rethrow(exception);
    end
end
D.graphType = 1; % 1: angle resolved spectra, 2: azimuthal stage, 3: single
spectrumres
D.currentGraph = 1; % 1: energy, 2: wavelength
%Default axis for E-k, L-t
D.axes.figureText{1} = { {'Wavevector:' 'Energy:'} {'Exc. Angle:'
'Wavelength:'}};
D.axes.titles{1} = { 'Wavevector k_{//}  [1/\mum]' 'Photon Energy  [eV]'...
    'Angle \theta  [deg]' 'Wavelength \lambda  [nm]'...
    'Energy v. wavevector' 'Wavelength v. Excitation Angle'};
D.axes.limits{1} = {[0 18 1.24 3.1] [0 60 400 1000]};
% Default Axis for E-phi, L-phi (azim)
D.axes.figureText{2} = {{'Azim. Angle:' 'Energy:'} {'Azim. Angle:'
'Wavelength:'}};
D.axes.titles{2} = {'Azimuthal Angle \phi  [deg]' 'Photon Energy  [eV]' ...
    'Azimuthal Angle \phi  [deg]' 'Wavelength \lambda  [nm]' ...
    'Energy v. Azimuthal Angle' 'Wavelength v. Azimuthal Angle'};
D.axes.limits{2} = {[0 180 1.24 3.1] [0 180 400 1000]};
% Default Axis for I-E, I-L (azim)
D.axes.figureText{3} = {{'Energy:' 'Intensity:'} {'Wavelength:'
'Intensity:'}};
D.axes.titles{3} = {'Photon Energy [eV]' 'Intensity [arb. units]' ...
    'Wavelength \lambda  [nm]' 'Intensity [arb. units]'...
    'Intensity v. Energy' 'Intensity v. Wavelength'};
D.axes.limits{3} = {[1.24 3.1 0 1] [400 1000 0 1]};
figure;D.colors = repmat(get(gcf,'DefaultAxesColorOrder'),6,1);close;
%D.colors([2 9],:) = [0 1 0;0 1 0]; %make the green brighter;
D.axes.defaultLimits = D.axes.limits;
D.defaultAngles.square = [0 90];
```

```
D.defaultAngles.hex = [0 60 120];
D.dataType = 1; % 1: raw data, 2: processed data, 3: FDTD data
D.isDataProcessed = false;
D.filename = {0 0 0 0};
D.path = [pwd '/'];
D.isFileLoaded = {false false false false};
D.data = [];
D.backgroundData = [];
D.scale = 1;
D.FDTDwavelengths = [];
D.FDTDangles = [];
D.wavelengths = [];
D.backgroundAngles = [];
D.angles = [];
D.smooth = {'' ''};
D.energyData = [];
D.wavelengthData = [];
D.isPIdata = false;
D.angleInput = {0,0,0,0};
S.energyModes = [];
S.wavelengthModes = [];
S.wavevectorModes = [];
S.brillouinZones = {{} {}};
S.isDataProcessed = false;
S.zonesVisible = [false false];
S.kParallel = 0;
S.handles = struct('energy',[],'wavelengths',[],'wavevector',[],...
    'zones',struct('one',[],'two',[]),'arrows',[],'saved',[],'all',[]);
S.material = '';
S.geometry = 'square';
S.graphType = 1; % 1: excitation, 2: azimuthal
S.spacing = [400 400];
S.refractiveIndex = 1.56;
S.angle = 0;
S.maxOrders = 2;
S.exposures = 1;
S.moireAngles = [0 90];
S.drude = struct('e',1,'hw',1,'hg',1);
%% Make the main figure and its menus
H.figure.main = figure('name','Rotational Stage Processing',...
    'NumberTitle','off',...
    'Position',[20 20 960 640],...
    'Toolbar','Figure');
set(H.figure.main,'units','normalized')
colormap('gray');
H.menu.moire = uimenu(H.figure.main,'label','Set Moire Angles');
H.moireMenu.square = uimenu(H.menu.moire,'label','Square');
H.squareMenu.perfect = uimenu(H.moireMenu.square,...
    'label','90',...
    'callback',@updateData,...
    'checked','on');
H.squareMenu.odom = uimenu(H.moireMenu.square,...
    'label','88.93',...
    'callback',@updateData,...
    'checked','off');
```

```
H.squareMenu.other = uimenu(H.moireMenu.square,...
    'label','Other...',...
    'callback',@updateData,...
    'checked','off');
H.moireMenu.hex = uimenu(H.menu.moire,'label','Hex');
H.hexMenu.perfect = uimenu(H.moireMenu.hex,...
    'label','60,120',...
    'callback',@updateData,...
    'checked','on');
H.hexMenu.other = uimenu(H.moireMenu.hex,...
    'label','Other...',...
    'callback',@updateData,...
    'checked','off');
H.menu.about = uimenu(H.figure.main,'label','About','callback',@aboutMenu);
%% Create the panels
% SPPmodes panel Panel
H.panel.sppModes = uipanel(H.figure.main,...
    'title','SPP MODES',...
    'Units','normalized',...
    'Position',[ .02 .02 .303 .47 ]);
% Status Window Panel
H.panel.status = uipanel(H.figure.main,...
    'Units','normalized',...
    'Position',[0.343 0.69 0.303 0.07 ]);
% Axes Properties Panel
H.panel.axes = uipanel(H.figure.main,...
    'title','AXES PROPERTIES',...
    'Units','normalized',...
    'Position',[0.343 0.78 0.303 0.2 ]);
% Process Data Panel
H.panel.processData = uipanel(H.figure.main,...
    'title','IMPORT DATA',...
    'units','normalized',...
    'position',[.02 .51 .303 .47]);
%% Make the Plot Buttons
H.main.processData = uicontrol(H.figure.main,...
    'callback',@processData,...
    'units','normalized',...
    'position',[.343 .55 .15 .13],...
    'string','PROCESS DATA');
H.main.SPPmodes = uicontrol(H.figure.main,...
    'callback',@processSPPmodes,...
    'units','normalized',...
    'position',[.497 .55 .15 .13],...
    'string','SPP MODES');
H.main.smooth = uicontrol(H.figure.main,...
    'callback',@updateData,...
    'units','normalized',...
    'position',[.343 .51 .15 .035],...
    'style','check',...
    'string','Smooth Data');
H.main.save = uicontrol(H.figure.main,...
    'callback',@saveSPPmodes,...
    'units','normalized',...
    'position',[.497 .51 .15 .035],...
    'string','Save SPP Modes');
%% Create the axes
%FT axes
H.axes.wavevector = axes('outerposition',[.677 .51 .303 .47],'box','on');...
    hold('all'),xlabel('k_x [1/\mum]'),ylabel('k_y [1/\mum]'),title('Fourier
transform');...
    axis([-1 1 -1 1]);...
    set(H.axes.wavevector,'dataaspectratio',[1 1 1]);
%Energy axes
H.axes.energy = axes('outerposition',[.343 .02 .303 .47],'box','on');...
    hold('all'),xlabel(D.axes.titles{1}{1}),ylabel(D.axes.titles{1}{2}),...
    title(D.axes.titles{1}{5}),axis([D.axes.limits{1}{1}]);
%Wavelength axes
H.axes.wavelength = axes('outerposition',[.677 .02 .303 .47],'box','on');...
    hold('all'),xlabel(D.axes.titles{1}{3}),ylabel(D.axes.titles{1}{4}),...
    title(D.axes.titles{1}{6}), axis([D.axes.limits{1}{2}]);
%% Create the context menus
H.menu.context = uicontextmenu;
H.contextMenu.SPPModes = uimenu(H.menu.context,...
    'label','SPP Modes');
H.contextMenu.Graphs = uimenu(H.menu.context,...
    'label','Pop Out Graph');
H.SPPcontext.Remove = uimenu(H.contextMenu.SPPModes,...
    'label','Remove Unsaved Lines',...
    'callback',@clearUnsaved);
H.SPPcontext.Return = uimenu(H.contextMenu.SPPModes,...
    'label','Return Unsaved Lines',...
    'callback',@returnUnsaved);
H.SPPcontext.Delete = uimenu(H.contextMenu.SPPModes,...
    'label','Delete Saved Lines',...
    'callback',@deleteSaved);
H.graphsContext.wavevector = uimenu(H.contextMenu.Graphs,...
    'label','Wavevector',...
    'callback',@popOutFigure);
H.graphsContext.energy = uimenu(H.contextMenu.Graphs,...
    'label','Energy Graph',...
    'callback',@popOutFigure);
H.graphsContext.wavelength = uimenu(H.contextMenu.Graphs,...
    'label','Wavelength Graph',...
    'callback',@popOutFigure);
H.contextMenu.hide = uimenu(H.menu.context,...
    'label','Hide Data',...
    'callback',@toggleShowData,...
    'enable','off');
H.contextMenu.zones = uimenu(H.contextMenu.SPPModes,...
    'label','Show Bragg Lines',...
    'enable','off');
H.zonesContext.one = uimenu(H.contextMenu.zones,...
    'label','1st order',...
    'callback',@toggleShowZones,...
    'checked','off');
H.zonesContext.two = uimenu(H.contextMenu.zones,...
    'label','2nd order',...
    'callback',@toggleShowZones,...
    'checked','off');
```

```
set(struct2array(H.axes),'uicontextmenu',H.menu.context);
%% Populate the Process Data Panel
H.processData.dataType = uicontrol(H.panel.processData,...
    'callback',@updateData,...
    'style','pop',...
    'Units','normalized',...
    'Position',[.05 .9 .5 .05],...
    'String',{'Raw Data','Processed Data','FDTD Data'});
uicontrol(H.panel.processData,...
    'style','text',...
    'units','normalized',...
    'position',[.6 .84 .35 .1],...
    'horizontalalignment','left',...
    'string','Data type');
H.processData.dataFile = uicontrol(H.panel.processData,...
    'style','edit',...
    'enable','inactive',...
    'units','normalized',...
    'position',[.05 .75 .65 .1],...
    'string','Data file');
H.processData.selectData = uicontrol(H.panel.processData,...
    'callback',@updateData,...
    'units','normalized',...
    'position',[.75 .75 .2 .1],...
    'string','Select');
H.processData.backgroundFile = uicontrol(H.panel.processData,...
    'style','edit',...
    'enable','inactive',...
    'units','normalized',...
    'position',[.05 .65 .65 .1],...
    'string','Background file');
H.processData.selectBackground = uicontrol(H.panel.processData,...
    'callback',@updateData,...
    'units','normalized',...
    'position',[.75 .65 .2 .1],...
    'string','Select');
H.processData.scale = uicontrol(H.panel.processData,...
    'callback',@updateData,...
    'Units','normalized',...
    'Position',[ .05 .55 0.4 0.1 ],...
    'String','1',...
    'Style','edit');
H.processData.scaleText = uicontrol(H.panel.processData,...
    'Units','normalized',...
    'Position',[ .5 0.55 0.45 0.08 ],...
    'String','Scale',...
    'HorizontalAlignment','left',...
    'Style','text');
H.processData.wavelengthsFile = uicontrol(H.panel.processData,...
    'style','edit',...
    'enable','inactive',...
    'units','normalized',...
    'position',[.05 .45 .65 .1],...
    'string','Wavelengths file',...
    'visible','off');
```

```
H.processData.selectWavelengths = uicontrol(H.panel.processData,...
    'callback',@updateData,...
    'units','normalized',...
    'position',[.75 .45 .2 .1],...
    'string','Select',...
    'visible','off');
H.processData.anglesFile = uicontrol(H.panel.processData,...
    'style','edit',...
    'enable','inactive',...
    'units','normalized',...
    'position',[.05 .35 .65 .1],...
    'string','Angles file',...
    'visible','off');
H.processData.selectAngles = uicontrol(H.panel.processData,...
    'callback',@updateData,...
    'units','normalized',...
    'position',[.75 .35 .2 .1],...
    'string','Select',...
    'visible','off');
H.processData.plotBackground = uicontrol(H.panel.processData,...
    'callback',@plotBackground,...
    'Units','normalized',...
    'Position',[ .05 0.25 0.45 0.08 ],...
    'String','Plot Background');
H.processData.saveBackground = uicontrol(H.panel.processData,...
    'callback',@saveBackground,...
    'Units','normalized',...
    'Position',[ .05 0.15 0.45 0.08 ],...
    'String','Save Background');
H.processData.saveData = uicontrol(H.panel.processData,...
    'callback',@saveData,...
    'Units','normalized',...
    'Position',[ .05 0.05 0.45 0.08 ],...
    'String','Save Data');
H.processData.loadFiles = uicontrol(H.panel.processData,...
    'callback',@updateData,...
    'units','normalized',...
    'position',[ .55 .05 .4 .3],...
    'string','Load Files');
%% Populate the SPPModes Panel
% Material Selection
if isempty(D.materials.matdata)
    materialString = {'No Materials File Available','Diffraction
Modes','Drude Model'};
else
    materialString = {sprintf('Material list
updated %s',D.materials.matdate),...
        D.materials.matdata{:,2},'Drude Model','Diffraction Modes'};
end
H.SPPmodes.materialList = uicontrol(H.panel.sppModes,...
    'style','pop',...
    'callback',@updateData,...
    'Units','normalized',...
    'Position',[.05 .9 .9 .05],...
    'String',materialString);
```

```
% Geometry Selection
H.SPPmodes.geometryList = uicontrol(H.panel.sppModes,...
    'style','popup',...
    'callback',@updateData,...
    'Units','normalized',...
    'Position',[.05 .8 .5 .05],...
    'String',{'Line','Square','Hex'});
% Azim sweep checkbox
H.SPPmodes.azim = uicontrol(H.panel.sppModes,...
    'callback',@updateData,...
    'Units','normalized',...
    'Position',[ .6 .8 .35 .05 ],...
    'String','Azim sweep',...
    'Style','checkbox');
% Spacing
H.SPPmodes.spacing = uicontrol(H.panel.sppModes,...
    'callback',@updateData,...
    'Units','normalized',...
    'Position',[ .05 .65 0.4 0.1 ],...
    'String','400',...
    'Style','edit');
uicontrol(H.panel.sppModes,...
    'Units','normalized',...
    'Position',[ .5 0.65 0.45 0.08 ],...
    'String','Spacing [nm]',...
    'HorizontalAlignment','left',...
    'Style','text');
% Index
H.SPPmodes.refractiveIndex = uicontrol(H.panel.sppModes,...
    'callback',@updateData,...
    'Units','normalized',...
    'Position',[ .05 .55 0.4 0.1 ],...
    'String','1.56',...
    'Style','edit');
uicontrol(H.panel.sppModes,...
    'Units','normalized',...
    'Position',[ .5 0.55 0.45 0.08 ],...
    'String','Refractive index',...
    'HorizontalAlignment','left',...
    'Style','text');
%Angle
H.SPPmodes.angle = uicontrol(H.panel.sppModes,...
    'callback',@updateData,...
    'Units','normalized',...
    'Position',[ .05 .45 0.4 0.1  ],...
    'String','0',...
    'Style','edit');
H.SPPmodes.angleText = uicontrol(H.panel.sppModes,...
    'Units','normalized',...
    'Position',[ .5 0.45 0.45 0.08 ],...
    'String','Azimuthal angle',...
    'HorizontalAlignment','left',...
    'Style','text');
%Max order to plot
H.SPPmodes.orders = uicontrol(H.panel.sppModes,...
    'callback',@updateData,...
    'style','edit',...
    'units','normalized',...
    'position',[.05 .35 0.4 0.1],...
    'string','2');
uicontrol(H.panel.sppModes,...
    'style','text',...
    'unit','normalized',...
    'position',[ .5 0.35 0.45 0.08],...
    'HorizontalAlignment','left',...
    'string','Max orders to plot');
%Moire Offset Angle
H.SPPmodes.moireAngles = uicontrol(H.panel.sppModes,...
    'callback',@updateData,...
    'Units','normalized',...
    'Position',[ .05 .25 0.4 0.1  ],...
    'String','1',...
    'Style','edit');
H.SPPmodes.moireAnglesText = uicontrol(H.panel.sppModes,...
    'Units','normalized',...
    'Position',[ .5 0.25 0.45 0.08 ],...
    'String','Number of Exposures',...
    'HorizontalAlignment','left',...
    'Style','text');
%Drude Coefficients
H.SPPmodes.drude.e = uicontrol(H.panel.sppModes,...
    'callback',@updateData,...
    'Units','normalized',...
    'Position',[ 0.05 0.05 0.283 0.1 ],...
    'String','1',...
    'Style','edit',...
    'visible','off');
H.SPPmodes.drude.eText = uicontrol(H.panel.sppModes,...
    'Units','normalized',...
    'Position',[ 0.05 0.15 0.283 0.08 ],...
    'String','e',...
    'Style','text',...
    'visible','off');
H.SPPmodes.drude.hw  = uicontrol(H.panel.sppModes,...
    'callback',@updateData,...
    'Units','normalized',...
    'Position',[ 0.358 0.05 0.283 0.1 ],...
    'String','1',...
    'Style','edit',...
    'visible','off');
H.SPPmodes.drude.hwText  = uicontrol(H.panel.sppModes,...
    'Units','normalized',...
    'Position',[ 0.358 0.15 0.283 0.08 ],...
    'String','h/w',...
    'Style','text',...
    'visible','off');
H.SPPmodes.drude.hg  = uicontrol(H.panel.sppModes,...
    'callback',@updateData,...
    'Units','normalized',...
    'Position',[ 0.667 0.05 0.283 0.1 ],...
    'String','1',...
    'Style','edit',...
    'visible','off');
H.SPPmodes.drude.hgText  = uicontrol(H.panel.sppModes,...
    'Units','normalized',...
    'Position',[ 0.667 0.15 0.283 0.08 ],...
    'String','h/g',...
    'Style','text',...
    'visible','off');
%% Populate the Axes Properties Panel
H.axesProperties.graphList = uicontrol(H.panel.axes,...
    'callback',@updateData,...
    'style','popup',...
    'Units','normalized',...
    'Position',[0.05 0.7 0.6 0.2],...
    'String',{'Energy Graph','Wavelength Graph'});
H.axesProperties.reset = uicontrol(H.panel.axes,...
    'callback',@updateData,...
    'units','normalized',...
    'position',[.7 .7 .25 .2],...
    'string','Reset');
H.axesProperties.XaxisText = uicontrol(H.panel.axes,...
    'HorizontalAlignment','left',...
    'Units','normalized',...
    'Position',[ 0.02 0.43 0.28 0.15 ],...
    'String','Wavevector:',...
    'Style','text');
uicontrol(H.panel.axes,...
    'HorizontalAlignment','left',...
    'Units','normalized',...
    'Position',[ 0.32 0.45 0.2 0.15 ],...
    'String','from',...
    'Style','text');
uicontrol(H.panel.axes,...
    'HorizontalAlignment','left',...
    'Units','normalized',...
    'Position',[ 0.7 0.45 0.2 0.15 ],...
    'String','to',...
    'Style','text');
H.axesProperties.YaxisText = uicontrol(H.panel.axes,...
    'HorizontalAlignment','left',...
    'Units','normalized',...
    'Position',[ 0.02 0.10 0.28 0.15 ],...
    'String','Energy:',...
    'Style','text');
uicontrol(H.panel.axes,...
    'HorizontalAlignment','left',...
    'Units','normalized',...
    'Position',[ 0.32 0.08 0.2 0.15 ],...
    'String','from',...
    'Style','text');
uicontrol(H.panel.axes,...
    'HorizontalAlignment','left',...
    'Units','normalized',...
    'Position',[ 0.7 0.08 0.2 0.15 ],...
    'String','to',...
    'Style','text');
H.axesProperties.XaxisMin = uicontrol(H.panel.axes,...
    'callback',@updateData,...
    'Units','normalized',...
    'Position',[ 0.42 0.4 0.25 0.25 ],...
    'String',D.axes.limits{1}{2}(1),...
    'Style','edit');
H.axesProperties.XaxisMax = uicontrol(H.panel.axes,...
    'callback',@updateData,...
    'Units','normalized',...
    'Position',[ 0.75 0.4 0.25 0.25 ],...
    'String',D.axes.limits{1}{2}(2),...
    'Style','edit');
H.axesProperties.YaxisMin = uicontrol(H.panel.axes,...
    'callback',@updateData,...
    'Units','normalized',...
    'Position',[ 0.42 0.03 0.25 0.25 ],...
    'String',D.axes.limits{1}{1}(1),...
    'Style','edit');
H.axesProperties.YaxisMax = uicontrol(H.panel.axes,...
    'callback',@updateData,...
    'Units','normalized',...
    'Position',[ 0.75 0.03 0.25 0.25 ],...
    'string',D.axes.limits{1}{1}(2),...
    'Style','edit');
%% Status Message
H.status = uicontrol(H.panel.status,...
    'Units','normalized',...
    'Position',[ 0.1 0.1 0.8 0.8 ],...
    'String','This is the status!',...
    'Style','text',...
    'fontsize',14);
%% Store the handles and data
set(H.figure.main,'userdata',{H,D,S});
updateFigure;
%% Nested Functions Used in Callbacks
    function updateData(obj,~)
        if ~exist('obj','var')
            set(H.figure.main,'userdata',{H,D,S});
            updateFigure;
            return
        end
        switch obj
            case H.squareMenu.perfect
                D.defaultAngles.square = [0 90];
            case H.squareMenu.odom
                D.defaultAngles.square = [0 88.93];
            case H.squareMenu.other
                D.defaultAngles.square = defineAngles(2);
            case H.hexMenu.perfect
                D.defaultAngles.hex = [0 60 120];
            case H.hexMenu.perfect
                D.defaultAngles.hex = defineAngles(3);
            case H.main.smooth
```

```
                if get(obj,'value')
                    D.smooth = {defineSmoothParameters};
                else
                    D.smooth = {'',''};
                end
            case H.processData.dataType
                D.dataType = get(obj,'value');
                D.isFileLoaded = {false false false false};
            case H.processData.selectData
                [D.filename{1} D.path] = setFilename(D.path,'data');
                D.isFileLoaded{1} = false;
            case H.processData.selectBackground
                [D.filename{2} D.path] = setFilename(D.path,'background');
                D.isFileLoaded{2} = false;
            case H.processData.selectWavelengths
                [D.filename{3} D.path] = setFilename(D.path,'wavelengths');
                D.isFileLoaded{3} = false;
            case H.processData.selectAngles
                [D.filename{4} D.path] = setFilename(D.path,'angles');
                D.isFileLoaded{4} = false;
            case H.processData.scale
                D.scale = str2double(get(obj,'string'));
            case H.processData.loadFiles
                fields = 
{'data','backgroundData','FDTDwavelengths','FDTDangles'};
                for ii = 1:length(D.filename)
                    if D.filename{ii} ~= 0
                        [~,~,fileType] = fileparts(D.filename{ii});
                        switch lower(fileType)
                            case '.txt'
                                D.(fields{ii}) = load([D.path 
D.filename{ii}]);
                                D.isPIdata = false;
                            case '.mat'
                                L = load([D.path D.filename{ii}]);
                                graphType = strfind({'Dispersion 
Data','Azimuthal Sweep','Single Spectrum'},L.graphType);
                                graphType = 
find(arrayfun(@(X)~isempty(graphType{X}),1:length(graphType)));
                                wavelengths = L.wavelengths;
                                angles = L.angles;
                                if ii == 1
                                    data = L.data;
                                    D.data = [graphType-1, angles; 
wavelengths, data];
                                elseif ii==2
                                    backgroundData = L.backgroundData;
                                    D.backgroundData = sortrows([wavelengths 
backgroundData]);
                                    D.backgroundData = [graphType-1, 
backgroundAngles; D.backgroundData];
                                end
                            case '.csv'
                                A = importdata([D.path 
D.filename{ii}],',',1);
```

```
                                if sum(strcmpi('frame',A.colheaders))
                                    angleAns = inputdlg('Please input
[startAngle,endAngle,incriment]',...
                                        'Define Angles',1,{'[0,70,1]'});
                                    D.angleInput{ii} =
cell2mat(textscan(angleAns{1},'[%f,%f,%f]'));
                                end
                                D.(fields{ii}) = A.data;
                                D.isPIdata = true;
                            otherwise
                                setStatus('Unsupported File type');
                        end
                        D.isFileLoaded{ii} = true;
                    else
                        D.(fields{ii}) = [];
                    end
                end
                D.isDataProcessed = false;
            case H.SPPmodes.materialList
                materials = get(obj,'string');
                choice = get(obj,'value');
                if choice == 1
                    S.material = '';
                else
                    S.material = materials{choice};
                end
                S.isDataProcessed = false;
            case H.SPPmodes.geometryList
                geometry = get(obj,'string');
                choice = get(obj,'value');
                S.geometry = geometry{choice};
                updateData(H.SPPmodes.moireAngles);
            case H.SPPmodes.azim
                if get(obj,'value')
                    S.graphType = 2;
                else
                    S.graphType = 1;
                end
            case H.SPPmodes.spacing
                spacings = eval(['[' get(obj,'string') ']'] );
                if length(spacings) == 1
                    S.spacing = repmat(spacings,1,length(S.moireAngles));
                else
                    multiples =
strfind({'line','square','hex'},lower(S.geometry));
                    multiples =
find(arrayfun(@(X)~isempty(multiples{X}),1:length(multiples)));
                    S.spacing = repmat(spacings,1,multiples);
                end
            case H.SPPmodes.refractiveIndex
                S.refractiveIndex = str2double(get(obj,'string'));
                if isnan(S.refractiveIndex)
                    string = get(obj,'string');
                    if strcmpi(string,'noa 61')||strcmpi(string,'noa61')
                        S.refractiveIndex = 'noa 61';
                        setStatus('This is the status!');
                    else
                        setStatus('Invalid entry for Index');
                    end
                end
            case H.SPPmodes.angle
                S.angle = str2double(get(obj,'string'));
            case H.SPPmodes.orders
                S.maxOrders = str2double(get(obj,'string'));
            case H.SPPmodes.moireAngles
                exposures = eval(['[' get(obj,'string') ']']);
                if length(exposures) == 1
                    multiples =
strfind({'line','square','hex'},lower(S.geometry));
                    multiples =
find(arrayfun(@(X)~isempty(multiples{X}),1:length(multiples)));
                    S.exposures = exposures;
                    angles = linspace(0,180/multiples,exposures+1);
                    angles(end) = [];
                    S.moireAngles = angles;
                    for ii = 1:multiples-1
                        S.moireAngles = [S.moireAngles
angles+ii*180/multiples];
                    end
                else
                    multiples =
strfind({'line','square','hex'},lower(S.geometry));
                    multiples =
find(arrayfun(@(X)~isempty(multiples{X}),1:length(multiples)));
                    S.exposures = length(exposures);
                    for ii = 1:multiples-1
                        exposures = [exposures exposures+ii*180/multiples];
                    end
                    S.moireAngles = exposures;
                end
                updateData(H.SPPmodes.spacing);
            case H.SPPmodes.drude.e
                S.drude.e = str2double(get(obj,'string'));
            case H.SPPmodes.drude.hw
                S.drude.hw = str2double(get(obj,'string'));
            case H.SPPmodes.drude.hg
                S.drude.hg = str2double(get(obj,'string'));
            case H.axesProperties.graphList
                D.currentGraph = get(obj,'value');
            case H.axesProperties.reset
                D.axes.limits = D.axes.defaultLimits;
                updateAxes;
            case H.axesProperties.XaxisMin
                D.axes.limits{D.graphType}{D.currentGraph}(1) =
str2double(get(obj,'string'));
                updateAxes;
            case H.axesProperties.XaxisMax
                D.axes.limits{D.graphType}{D.currentGraph}(2) =
str2double(get(obj,'string'));
                updateAxes;
            case H.axesProperties.YaxisMin
                D.axes.limits{D.graphType}{D.currentGraph}(3) =
str2double(get(obj,'string'));
                updateAxes;
            case H.axesProperties.YaxisMax
                D.axes.limits{D.graphType}{D.currentGraph}(4) =
str2double(get(obj,'string'));
                updateAxes;
            otherwise
        end
        set(H.figure.main,'userdata',{H,D,S,'H,D,S'});
        updateFigure;
        function array = defineAngles(number)
            answer = inputdlg(sprintf('Indicate the %i default angles to
use',number));
            if isempty(answer)
                array = 0:180/number:179;
            else
                array = eval(sprintf('[ %s ]',answer{1}));
            end
            if ~(length(answer)==number && isnumeric(answer))
                errordlg('Invalid imput')
                defineAngles(number)
            end
        end
        function [points, method] = defineSmoothParameters()
            answer = inputdlg({'How many points to span? [odd number]' ...
                'What method to use? [lowess, loess, rlowess,
rloesss]'},'Smoothing',1, {'21' 'moving'});
            if isempty(answer)
                points = '';
                method = '';
            else
                points = str2double(answer{1});
                method = answer{2};
            end
        end
        function [filename, newPath] = setFilename(path,label)
            [filename, newPath] =
uigetfile('*.txt;*.mat;*.csv',sprintf('Select the %s file to
load',label),path);
            if newPath == 0
                newPath = path;
            end
        end
    end
    function updateFigure
        updateProcessDataPanel(D.dataType);
        updateFilenames;
        updateSPPmodesPanel(S.material);
        updateAxesPanel;
        switch S.graphType
            case 1 %variable excitation angle
                set(H.SPPmodes.angleText,'string','Azimuthal Angle')
            case 2 %variable azimuthal angle
                set(H.SPPmodes.angleText,'string','Excitation Angle')
        end
        function updateAxesPanel
            set(H.axesProperties.XaxisText,'string',...
                D.axes.figureText{D.graphType}{D.currentGraph}{1})
            set(H.axesProperties.YaxisText,'string',...
                D.axes.figureText{D.graphType}{D.currentGraph}{2})
            set(H.axesProperties.XaxisMin,'string',...
                D.axes.limits{D.graphType}{D.currentGraph}(1))
            set(H.axesProperties.XaxisMax,'string',...
                D.axes.limits{D.graphType}{D.currentGraph}(2))
            set(H.axesProperties.YaxisMin,'string',...
                D.axes.limits{D.graphType}{D.currentGraph}(3))
            set(H.axesProperties.YaxisMax,'string',...
                D.axes.limits{D.graphType}{D.currentGraph}(4))
        end
        function updateProcessDataPanel(type)
            switch type
                case 1 % raw data
                    set(H.processData.dataFile,'visible','on');
                    set(H.processData.selectData,'visible','on');
                    set(H.processData.backgroundFile,'visible','on');
                    set(H.processData.selectBackground,'visible','on');
                    set(H.processData.scale,'visible','on');
                    set(H.processData.scaleText,'visible','on');
                    set(H.processData.wavelengthsFile,'visible','off');
                    set(H.processData.selectWavelengths,'visible','off');
                    set(H.processData.anglesFile,'visible','off');
                    set(H.processData.selectAngles,'visible','off');
                    set(H.processData.plotBackground,'visible','on');
                    set(H.processData.saveBackground,'visible','on');
                    set(H.processData.saveData,'visible','on');
                    D.filename(3:4) = {0 0};
                case 2 % processed data
                    set(H.processData.dataFile,'visible','on');
                    set(H.processData.selectData,'visible','on');
                    set(H.processData.backgroundFile,'visible','off');
                    set(H.processData.selectBackground,'visible','off');
                    set(H.processData.scale,'visible','off');
                    set(H.processData.scaleText,'visible','off');
                    set(H.processData.wavelengthsFile,'visible','off');
                    set(H.processData.selectWavelengths,'visible','off');
                    set(H.processData.anglesFile,'visible','off');
                    set(H.processData.selectAngles,'visible','off');
                    set(H.processData.plotBackground,'visible','off');
                    set(H.processData.saveBackground,'visible','off');
                    set(H.processData.saveData,'visible','off');
                    D.filename(2:4) = {0 0 0};
                case 3 % FDTD data
                    set(H.processData.dataFile,'visible','on');
                    set(H.processData.selectData,'visible','on');
                    set(H.processData.backgroundFile,'visible','off');
                    set(H.processData.selectBackground,'visible','off');
                    set(H.processData.scale,'visible','off');
                    set(H.processData.scaleText,'visible','off');
```

```
                set(H.processData.wavelengthsFile,'visible','on');
                set(H.processData.selectWavelengths,'visible','on');
                set(H.processData.anglesFile,'visible','on');
                set(H.processData.selectAngles,'visible','on');
                set(H.processData.plotBackground,'visible','off');
                set(H.processData.saveBackground,'visible','off');
                set(H.processData.saveData,'visible','on');
                D.filename(2) = {0};
            end
        end
        function updateFilenames
            files = {'Data','Background','Wavelengths','Angles'};
            for ii = 1:length(D.filename)
                if D.filename{ii} == 0
                    set(H.processData.([lower(files{ii})
'File']),'string',[files{ii} ' File']);
                else
                    set(H.processData.([lower(files{ii})
'File']),'string',D.filename{ii});
                end
                if D.isFileLoaded{ii}
                    set(H.processData.([lower(files{ii})
'File']),'foregroundcolor','green')
                else
                    set(H.processData.([lower(files{ii})
'File']),'foregroundcolor','red')
                end
            end
        end
        function updateSPPmodesPanel(material)
            if strcmpi(material,'drude model')
                set(H.SPPmodes.drude.e,'visible','on');
                set(H.SPPmodes.drude.eText,'visible','on');
                set(H.SPPmodes.drude.hw,'visible','on');
                set(H.SPPmodes.drude.hwText,'visible','on');
                set(H.SPPmodes.drude.hg,'visible','on');
                set(H.SPPmodes.drude.hgText,'visible','on');
            else
                set(H.SPPmodes.drude.e,'visible','off');
                set(H.SPPmodes.drude.eText,'visible','off');
                set(H.SPPmodes.drude.hw,'visible','off');
                set(H.SPPmodes.drude.hwText,'visible','off');
                set(H.SPPmodes.drude.hg,'visible','off');
                set(H.SPPmodes.drude.hgText,'visible','off');
            end
            set(H.SPPmodes.azim,'value',S.graphType-1);
            geometry = strfind({'line','square','hex'},lower(S.geometry));
            geometry =
find(arrayfun(@(X)~isempty(geometry{X}),1:length(geometry)));
            set(H.SPPmodes.geometryList,'value',geometry);
        end
    end
    function updateAxes
        axis(H.axes.energy,D.axes.limits{D.graphType}{1})
        axis(H.axes.wavelength,D.axes.limits{D.graphType}{2})
        xlabel(H.axes.energy,D.axes.titles{D.graphType}{1})
        ylabel(H.axes.energy,D.axes.titles{D.graphType}{2})
        xlabel(H.axes.wavelength,D.axes.titles{D.graphType}{3})
        ylabel(H.axes.wavelength,D.axes.titles{D.graphType}{4})
        title(H.axes.energy,D.axes.titles{D.graphType}{5})
        title(H.axes.wavelength,D.axes.titles{D.graphType}{6})
    end
    function resetSPPHandles
        names = fieldnames(S.handles);
        for ii = 1:length(names)
            if strcmpi(names{ii},'zones')
                S.handles.(names{ii}) = struct('one',[],'two',[]);
                continue
            end
            S.handles.(names{ii}) = [];
        end
    end
    function setStatus(string)
        set(H.status,'string',string);
    end
% Toolbar Menus
    function aboutMenu(~,~) %About menu
        msgbox({'Rotational Stage Processing',...
            'This GUI processes spectra taken from the Rotational Stage
and',...
            'calculates the associated optical properites',...
            '','Design and Coding by: Alex Hryn','Contributions from: Mark
Huntington and Steve Lubin',...
            sprintf('You are using version %s',D.info.version),...
            sprintf('Last updated on %s',D.info.lastUpdated)},'Program
Information');
    end
% Process Data Panel
    function plotBackground(~,~) %Plot Background
        if ~D.isDataProcessed
            setStatus('Process Data First');
            return
        end
        figure;
        plot(D.wavelengths,D.backgroundData)
    end
    function saveBackground(~,~) %Save Background
        if ~D.isDataProcessed
            setStatus('Process Data First');
            return
        end
        outName=sprintf('%s_proc_v%s.txt',D.filename{2}(1:end-4),
strrep(D.info.version,'.','_'));
        [savePath,outName,fileType] = saveFile(outName,D.path,'Background');
        setStatus('Writing background file...')
        drawnow
        switch fileType
            case '.txt'
                backgroundOutput = sortrows([D.wavelengths
D.backgroundData]);
```

```
                backgroundOutput = [D.graphType-1, D.backgroundAngles;
backgroundOutput];
                dlmwrite([savePath outName], backgroundOutput, 'precision',
'%.4f', 'delimiter', '\t');
            case '.mat'
                graphTypes = {'Dispersion Data','Azimuthal Sweep','Single
Spectrum'};
                graphType = graphTypes{D.graphType};
                wavelengths = D.wavelengths;
                angles = D.backgroundAngles;
                backgroundData = D.backgroundData;

save(outName,'graphType','wavelengths','angles','backgroundData');
            case ''
                setStatus('');
                drawnow
                return
            otherwise
                setStatus('Only .txt and .mat files currently supported');
                return
        end
        setStatus('Background file saved!')
    end
    function saveData(~,~) %Save Data
        if ~D.isDataProcessed
            setStatus('Process Data First');
            return
        end
        outName=sprintf('%s_proc_v%s.txt',D.filename{1}(1:end-4),
strrep(D.info.version,'.','_'));
        [savePath,outName,fileType] = saveFile(outName,D.path,'Data');
        setStatus('Writing data file...')
        drawnow
        switch fileType
            case '.txt'
                dataOutput = [D.graphType-1, D.angles; D.data];
                dlmwrite([savePath outName], dataOutput, 'precision', '%.4f',
'delimiter', '\t');
            case '.mat'
                graphType = {'Dispersion Data','Azimuthal Sweep','Single
Spectrum'};
                graphType = graphType{D.graphType};
                wavelengths = D.wavelengths;
                angles = D.angles;
                data = D.data(:,2:end);
                save([savePath
outName],'graphType','wavelengths','angles','data');
            case ''
                setStatus('')
                drawnow;
                return
            otherwise
                setStatus('Only .txt and .mat files currently supported');
                return
        end
        setStatus('Data file saved!')
    end
    function saveSPPmodes(~,~) %Save SPP Modes
        if ~S.isDataProcessed
            setStatus('Calculate SPP Modes First');
            return
        end
        outNames = {'SPPModes_EnergyData.txt','SPPModes_WavelengthData.txt'};
        saveLabels  = {'Energy','Wavelength'};
        xVariable = {{'k','angle'},{'angle','angle'}};
        savePath = [pwd '/'];
        %
        for ii = 1:2
            [savePath,outName,fileType] = 
saveFile(outNames{ii},savePath,['SPP ' saveLabels{ii}]);
            setStatus(sprintf('Writing SPP %s file...',saveLabels{ii}))
            drawnow
            switch fileType
                case '.txt'
                    temp = vertcat(S.([lower(saveLabels{ii}) 'Modes']){1}, 
S.([lower(saveLabels{ii}) 'Modes']){2});
                    output = reshape(temp, size(S.([lower(saveLabels{ii}) 
'Modes']){1},1),[]);
                    dlmwrite([savePath outName], output, 'precision', '%.4f', 
'delimiter', '\t');
                case '.mat'
                    eval(sprintf('%s = S.([lower(saveLabels{ii}) 
''Modes''']){1};',xVariable{S.graphType}{ii}))
                    eval(sprintf('%s = S.([lower(saveLabels{ii}) 
''Modes''']){2}(:,1);',saveLabels{S.graphType}{ii}))
                    save([savePath 
outName],xVariable{S.graphType}{ii},saveLabels{S.graphType}{ii});
                case ''
                    setStatus('')
                    drawnow;
                    return
                otherwise
                    setStatus('Only .txt and .mat files currently 
supported');
                    return
            end
        end
        setStatus('SPP Modes saved!')
    end
    function [path,filename,extension] = 
saveFile(defaultFilename,defaultPath,titleString)
        [filename,path] = uiputfile({'*.txt','Text Files 
(.txt)';'*.mat','Matlab Files (.mat)'}...
            ,sprintf('Save %s Data',titleString),[defaultPath 
defaultFilename]);
        if filename == 0
            filename = '';
            path = defaultPath;
            extension = '';
        else
```

```
            [~,~,extension] = fileparts(filename);
        end
    end
% Figure Buttons
    function processData(~,~) %Process data
        if D.isDataProcessed
            return
        end
        switch D.dataType
            case 1
                allFilesLoaded = D.isFileLoaded{1} && D.isFileLoaded{2};
            case 2
                allFilesLoaded = D.isFileLoaded{1};
            case 3
                allFilesLoaded = D.isFileLoaded{1} && D.isFileLoaded{3} &&
D.isFileLoaded{4};
        end
        if ~allFilesLoaded
            setStatus('Select and load files first');
            return
        end
        setStatus('Processing Data...');
        drawnow
        try
            [D.wavelengths, D.backgroundAngles, D.angles, D.backgroundData,
D.data, energyData, wavelengthData, D.graphType] ...
                =
ProcessData(D.data,D.backgroundData,D.scale,D.dataType,D.isPIdata,D.angleInpu
t{1}, D.smooth{:},D.FDTDwavelengths,D.FDTDangles);
        catch exception
            %  exp = exception %for debugging errors
            D.isDataProcessed = false;
            switch exception.identifier
                case 'MATLAB:dimagree' %dimension mismatch
                    setStatus('Data and Background do not match');
                case 'MATLAB:load:couldNotReadFile'
                    if strfind(exception.message,'calibration.mat')
                        setStatus('No Calibration File')
                    else
                        rethrow(exception)
                    end
                otherwise
                    rethrow(exception)
            end
            return
        end
        cla(H.axes.energy)
        cla(H.axes.wavelength)
        cla(H.axes.wavevector)
        energies=1240./D.wavelengths;
        switch D.graphType
            case 1 % normal spectra
                wavevectors=1000.*(2.* pi./D.wavelengths)*sind(D.angles);
                D.energyData = image([min(wavevectors(:)),
max(wavevectors(:))], [min(energies), max(energies)], ...
energyData,'Parent',H.axes.energy,'CDataMapping','scaled');
                D.wavelengthData = image([D.angles(1) 
D.angles(end)],[D.wavelengths(1) D.wavelengths(end)], ...

wavelengthData,'Parent',H.axes.wavelength,'CDataMapping','scaled');
                D.axes.limits{1} = {[floor(min(wavevectors(:))) 
ceil(max(wavevectors(:))) min(energies) max(energies)] ...
                    [min(D.angles) max(D.angles) min(D.wavelengths) 
max(D.wavelengths)]};
            case 2 % azim spectra
                D.energyData = image([D.angles(1) D.angles(end)], 
[min(energies), max(energies)], ...

energyData,'Parent',H.axes.energy,'CDataMapping','scaled');
                D.wavelengthData = image([D.angles(1) 
D.angles(end)],[D.wavelengths(1) D.wavelengths(end)], ...

wavelengthData,'Parent',H.axes.wavelength,'CDataMapping','scaled');
                D.axes.limits{2} = {[min(D.angles) max(D.angles) 
min(energies) max(energies)] ...
                    [min(D.angles) max(D.angles) min(D.wavelengths) 
max(D.wavelengths)]};
            case 3 % single spaectrum
                D.energyData = plot(H.axes.energy,energies,energyData,'k');
                D.wavelengthData = plot(H.axes.wavelength, D.wavelengths, 
wavelengthData,'k');
                D.axes.limits{3} = {[min(energies) max(energies) 0 
max(wavelengthData)] ...
                    [min(D.wavelengths) max(D.wavelengths) 0 
max(wavelengthData)]};
        end
        set([D.energyData; D.wavelengthData],'uicontextmenu',H.menu.context);
        uistack(D.energyData,'bottom')
        uistack(D.wavelengthData,'bottom')
        updateAxes;
        setStatus('Data Processed!');
        resetSPPHandles;
        D.isDataProcessed = true;
        set(H.contextMenu.hide,'enable','on','label','Hide Data');
        updateAxes;
        updateData;
    end
    function processSPPmodes(~,~) %Calcuate SPP Modes
        if D.isDataProcessed && (S.graphType ~= D.graphType)
            button = questdlg('Types of data do not match. Calculating SPP 
Modes will remove the processed data.',...
                'Erase Processed Data?','Erase','Cancel','Erase');
            if strcmp(button,'Erase')
                cla(H.axes.energy);
                cla(H.axes.wavelength);
                updateData(H.processData.loadFiles)
                D.graphType = S.graphType;
            else
                return
            end
        else
            D.graphType = S.graphType;
        end
        if length(S.spacing)~=length(S.moireAngles)
            setStatus('Spacings and exposures do not match');
            return
        end
        setStatus('Calculating SPP modes...');
        drawnow
        [S.energyModes, S.wavelengthModes, S.wavevectorModes, indLab,
modeOrder] = ...
            SPPCalc(S.material, S.spacing, S.angle, S.refractiveIndex,
S.moireAngles, S.graphType, S.drude, H.status, S.maxOrders);
        if strcmp(get(H.status,'userdata'),'error')
            set(H.status,'userdata',[]);
            return
        end
        [S.brillouinZones] =
BZcalc(S.wavevectorModes{1}(:,1:2),S.wavevectorModes{1}(:,3));
        for cc = 1:max(modeOrder)+1
            colorInd = (modeOrder == cc);
            SPPcolorOrd(colorInd,1:3) =
repmat(D.colors(cc,:),sum(colorInd),1);
        end
        delete(S.handles.all);
        delete(S.handles.arrows);
        resetSPPHandles;
        % Plot modes on Energy axes
        for jj = 1:size(S.energyModes{1},2)
            S.handles.energy(jj) =
plot(H.axes.energy,S.energyModes{1}(:,jj),...

S.energyModes{2}(:,jj),'color',SPPcolorOrd(jj,:),'linewidth',1);
            set(S.handles.energy(jj),'UserData',indLab{jj})
        end
        if S.graphType == 1 %Standard SPP calc
            lightline_x = (0:0.01:6)./(1.05459e-34*299792458).*1e-
6.*1.60219e-19;
            lightline_y = 0:0.01:6;
            S.handles.energy(end+1) =
plot(H.axes.energy,lightline_x,lightline_y,'--black','linewidth',1);
            set(S.handles.energy(end),'UserData',{'light line!'});
        end
        % Plot modes on Wavelength axes
        for kk = 1:size(S.wavelengthModes{1},2)
            S.handles.wavelengths(kk) =
plot(H.axes.wavelength,S.wavelengthModes{1}(:,kk),...

S.wavelengthModes{2}(:,kk),'color',SPPcolorOrd(kk,:),'linewidth',1);
            set(S.handles.wavelengths(kk),'UserData',indLab{kk})
        end
        % Plot modes on wavevector axes
        XYcoordinates = S.wavevectorModes{1};
        FTlabel = S.wavevectorModes{2};
        FT_colors = vertcat([0 0 0], D.colors);
        cla(H.axes.wavevector);
        for mm = 1:size(XYcoordinates,1)
            if XYcoordinates(mm,3) <= S.maxOrders
                n = XYcoordinates(mm,3);
            else
                continue
            end
            S.handles.wavevector(mm) =
scatter(H.axes.wavevector,XYcoordinates(mm,1),XYcoordinates(mm,2),[],FT_color
s(n+1,:));
            set(S.handles.wavevector(mm),'userdata',{['['
int2str(FTlabel(mm,:)) ']']});
        end
        limits = [min(XYcoordinates(:)) max(XYcoordinates(:))];
        set(H.axes.wavevector,'Xlim',limits,'Ylim',limits)
        % Plot zones on wavevector axes
        for oo = 1:2
            zoneString = {'one','two'};
            for bb = 1:length(S.brillouinZones{oo})
                S.handles.zones.(zoneString{oo})(bb) = ...

plot(H.axes.wavevector,S.brillouinZones{oo}{bb}(:,1),S.brillouinZones{oo}{bb}
(:,2),...
                    'color',D.colors(oo,:),'linestyle','--');
            end
            uistack(S.handles.zones.(zoneString{oo}),'bottom');

set(S.handles.zones.(zoneString{oo}),'buttondownfcn',@BZButtonDownFcn);
            if S.zonesVisible(oo)
                set(S.handles.zones.(zoneString{oo}),'visible','on')
            else
                set(S.handles.zones.(zoneString{oo}),'visible','off')
            end
        end
        set(H.contextMenu.zones,'enable','on');
        S.handles.all = [S.handles.energy(:)' S.handles.wavelengths(:)'
S.handles.wavevector(:)'];
        set(S.handles.all,'buttondownfcn',@SPPmodesButtonDownFcn);
        S.handles.saved = [];
        setStatus('SPP Modes plotted');
        S.isDataProcessed = true;
        updateData;
        updateAxes;
    end
% Plots
    function popOutFigure(obj,~) % Copy figure submenu
        switch obj %find out which one you want to copy
            case H.graphsContext.wavevector
                copyAxes('wavevector');
            case H.graphsContext.energy
                copyAxes('energy');
            case H.graphsContext.wavelength
                copyAxes('wavelength');
        end
        function copyAxes(type)
            newfig = figure;
            colormap('gray');
            newaxes = copyobj(H.axes.(type),newfig);
            cla;
            newlines = copyobj(get(H.axes.(type),'children'),newaxes);
            if S.isDataProcessed
                newlines(end) = []; %to clear data
            end
            set(newaxes,'units','normalized','position',[.13 .11 .775 .815])
            set(newlines(:),'buttondownfcn',{@newfig_bdfcn,newfig,newlines});
            dcm_obj = datacursormode(newfig);
            set(dcm_obj,'UpdateFcn',@displayUserdata)
            function txt = displayUserdata(~,obj)
                line = obj.target;
                txt = get(line,'UserData');
            end
        end
    end
    function SPPmodesButtonDownFcn(obj,~)
        selectSPPlines(obj);
        switch get(obj,'parent')
            case H.axes.wavevector
            case H.axes.energy
                currentPoint = get(gca,'currentpoint');
                S.kParallel = currentPoint(1,1);
            case H.axes.wavelength
                currentPoint = get(gca,'currentpoint');
                angle = currentPoint(1,1);
                wavelength = currentPoint(1,2);
                S.kParallel = 2*pi/(wavelength*1e-3)*sind(angle);
        end
        showKsppVectors;
        if sum(S.zonesVisible)
            set(S.handles.arrows,'visible','on')
        end
        function selectSPPlines(line) % SPP Line selection
            mode = get(line,'UserData'); %get the indicies from that line's
UserData
            setStatus(sprintf('%s ',mode{:}));
            if strcmpi(mode,'light line!')
                return
            end
            [Ekind,FTind] = findLines(mode);
            switch get(H.figure.main,'selectiontype')
                case 'normal'
                    S.handles.saved = unique([S.handles.saved
S.handles.energy(Ekind) ...
                        S.handles.wavelengths(Ekind)
S.handles.wavevector(FTind)]);
                case 'alt'
                    remove = unique([S.handles.energy(Ekind)
S.handles.wavelengths(Ekind) S.handles.wavevector(FTind)]);
                    set(remove,'linewidth',1);
                    erasedLines = arrayfun(@(X)find(S.handles.saved ==
remove(X)),1:length(remove),'uni',false);
                    erasedLines = [erasedLines{:}];
                    S.handles.saved(erasedLines) = [];
                otherwise
                    return
            end
            set(S.handles.saved,'linewidth',2);
        end
        function showKsppVectors()
            %find out which FT points are saved
            points = arrayfun(@(X)find(S.handles.wavevector ==
S.handles.saved(X)),1:length(S.handles.saved),'uni',false);
            points = [points{:}];
            pointHandles = S.handles.wavevector(points);
            colors = get(pointHandles,'cdata');
            if isnumeric(colors)
                colors = {colors};
            end
            %get the x,y data
            coordinates = S.wavevectorModes{1}(points,1:2);
            %add k// from graph
            coordinates(:,1) = coordinates(:,1)+S.kParallel;
            %convert to figure units for annotation
            [Ox,Oy] = axescoord2figurecoord(0,0,H.axes.wavevector);
            [x,y] =
axescoord2figurecoord(coordinates(:,1),coordinates(:,2),H.axes.wavevector);
            energyCoordinates = [[S.kParallel; S.kParallel],
D.axes.limits{D.graphType}{1}(3:4)'];
            wavelengthCoordinates(:,2) =
linspace(D.axes.limits{D.graphType}{2}(3),D.axes.limits{D.graphType}{2}(4),20
0);
            wavelengthCoordinates(:,1) =
real(asind(S.kParallel/(2*pi)*wavelengthCoordinates(:,2)*1e-3));
            %plot the arrows
            delete(S.handles.arrows)
            S.handles.arrows = zeros(size(coordinates,1)+2,1);
            for ii = 1:size(coordinates,1)
                S.handles.arrows(ii) = annotation('arrow',...
                    [Ox x(ii)],[Oy y(ii)],...
                    'color',colors{ii},'linewidth',1.5,...
                    'headwidth',6,'headlength',6,...
                    'headstyle','plain','visible','off');
            end
            uistack(S.handles.arrows(1:end-2),'bottom');
            S.handles.arrows(end-1) = plot(H.axes.energy,...
                energyCoordinates(:,1),energyCoordinates(:,2),...
                'color',[0 0 0],'linewidth',1.5,'visible','off');
            uistack(S.handles.arrows(end-1),'bottom');
            S.handles.arrows(end) = plot(H.axes.wavelength,...
                wavelengthCoordinates(:,1),wavelengthCoordinates(:,2),...
                'color',[0 0 0],'linewidth',1.5,'visible','off');
            uistack(S.handles.arrows(end),'bottom');
            if D.isDataProcessed
                uistack(D.energyData,'bottom');
                uistack(D.wavelengthData,'bottom');
            end
        end
        function [Ekind, FTind]  = findLines(mode)
            Eklabels = get(S.handles.energy,'userdata'); %cell arrary of
arrays
            FTlabels = get(S.handles.wavevector,'userdata'); %cell array of
cells of arrays
            FTlabels =
arrayfun(@(X)FTlabels{X}{1},1:length(FTlabels),'uni',false)'; %cell array of
arrays
            Ekind = [];
            FTind = [];
            for ii = 1:length(mode)
                FTind(end+1) = find(strcmp(mode{ii},FTlabels));
                for jj = 1:length(Eklabels)
                    if sum(strcmp(mode{ii},Eklabels{jj}));
                        Ekind(end+1) = jj;
                    end
                end
            end
        end
    end
    function BZButtonDownFcn(obj,~)
        switch get(H.figure.main,'selectiontype')
            case 'normal'
                set(obj,'linewidth',2)
            case 'alt'
                set(obj,'linewidth',1)
            otherwise
                return
        end
    end
    function clearUnsaved(~,~) % Clear Unsaved submenu
        if ~S.isDataProcessed
            setStatus('Calculate SPPModes First');
            return
        end
        set([S.handles.energy S.handles.wavevector
S.handles.wavelengths],'visible','off')
        set(S.handles.saved,'visible','on')
    end
    function returnUnsaved(~,~) % Return Unsaved submenu
        if ~S.isDataProcessed
            setStatus('Calculate SPPModes First');
            return
        end
        set([S.handles.energy S.handles.wavevector
S.handles.wavelengths],'visible','on')
    end
    function deleteSaved(~,~) % Delete Saved submenu
        if ~S.isDataProcessed
            setStatus('Calculate SPPModes First');
            return
        end
        set([S.handles.energy S.handles.wavevector
S.handles.wavelengths],'visible','on')
        set(S.handles.all,'linewidth',1)
        S.handles.saved = [];
        delete(S.handles.arrows)
        S.handles.arrows = [];
        setStatus('SPP Modes plotted');
        updateData;
    end
    function toggleShowData(obj,~) % Show/Hide displayed data
        str = get(obj,'label');
        if strcmp(str,'Hide Data')
            set([D.energyData, D.wavelengthData],'visible','off');
            set(obj,'label','Show Data');
        elseif strcmp(str,'Show Data')
            set([D.energyData, D.wavelengthData],'visible','on');
            set(obj,'label','Hide Data');
        end
    end
    function toggleShowZones(obj,~) % Show/Hide brillouin zones
        switch obj
            case H.zonesContext.one
                if strcmp(get(obj,'Checked'),'on')
                    set(S.handles.zones.one,'visible','off');
                    S.zonesVisible(1) = false;
                    set(obj,'checked','off')
                else
                    set(S.handles.zones.one,'visible','on');
                    S.zonesVisible(1) = true;
                    set(obj,'checked','on')
                end
            case H.zonesContext.two
                if strcmp(get(obj,'Checked'),'on')
                    set(S.handles.zones.two,'visible','off');
                    S.zonesVisible(2) = false;
                    set(obj,'checked','off')
                else
                    set(S.handles.zones.two,'visible','on');
                    S.zonesVisible(2) = true;
                    set(obj,'checked','on')
                end
        end
        if sum(S.zonesVisible)
            set(S.handles.arrows,'visible','on')
        else
            set(S.handles.arrows,'visible','off')
        end
    end
end
function [Ek_calc,Lt_calc,FT_calc,indLab,modeOrder] = ...
    SPPCalc(material, spacing, angle, refractiveIndex, moireAngles, type,
drudeFactors,statusHandle,maxOrder)
% [Ek_calc, Lt_calc, indLab, modeOrder]  = SPPCalc(material, numberOfLines,
spacing, azAngle, refractiveIndex, moireAngles,
drudeFactors,statusHandle,maxOrder)
%     material is the material 'Ag' and 'Au' are availible
%     Geometry is the lattice geometry 'Line', 'Hex', and 'Sq' are
%     availible
%     spacing is the spacing between lattice points in nm
%     azAngle is the azimuthal angle in degrees
%     refractiveIndex is the dielectric enviivironment for the spp
%     Created by Mark Huntington May 2011.
%     Contributions from: Steve Lubin and Alex Hryn
if type == 1
    azimuthalAngle = angle;
elseif type == 2
    excitationAngle = angle;
end
nInterp = 10; %degree of interpolation
hBar = 1.05459e-34; % Reduced Planck's constant = 1.05459e-34 m2 kg / s
c = 299792458; %speed of light = 299,792,458 m / s
spacing = spacing*1e-9; %from nm to m
if strcmpi('drude model', material)
    %make energy column
    EmInt(:,1) = (0.6:0.01:4); %in eV
    %make delectric column
    w = EmInt(:,1).*1.51927e15; %0.6eV/hbar to 4eV/hbar
    epsil = drudeFactors.e - ((drudeFactors.hw*1.51927e15)^2./(w.*(w +
1i*(drudeFactors.hg*1.51927e15))));
    EmInt(:,2) = real(epsil);
elseif strcmpi('diffraction modes',material)
    %make energy column
    EmInt(:,1) = (0.6:0.01:4); %in eV
    %make dielectric column
    EmInt(:,2) = inf;
else
    L = load('matdata.mat');
    %find the material
    index =
arrayfun(@(x)strcmpi(L.matdata{x,2},material),1:length(L.matdata),'UniformOut
put', false);
    index = [index{:}];
    if sum(index) ~= 1
        set(statusHandle,'string','Choose material','Userdata','error')
        Ek_calc = 0;
        Lt_calc = 0;
        FT_calc = 0;
        indLab = 0;
        modeOrder = 0;
        return
    end
    Em = L.matdata{index,1};
    if size(Em,2)==3 %imaginary part included
        Em(:,2) = complex(Em(:,2),Em(:,3));
    end
    EmInt(:,1) = interp1(1:length(Em), Em(:,1), 1:1/nInterp:length(Em),
'spline');% energy
    EmInt(:,2) = interp1(1:length(Em), Em(:,2), 1:1/nInterp:length(Em),
'spline');% epsilon
end
Energy = EmInt(:,1);
```

```
lambda = 1240./Energy;
EnergyJoule = EmInt(:,1).*1.60219e-19;
if ischar(refractiveIndex)
    if strcmpi(refractiveIndex,'noa 61')
        refractiveIndex = 1.5375 + 8290.45./(lambda.^2) -
2.11046e8./(lambda.^4);
    end
end
dielectric = refractiveIndex.^2;
numberOfLines = length(moireAngles);
if numberOfLines == 0
    set(statusHandle,'string','Choose Geometry','Userdata','error')
    Ek_calc = 0;
    Lt_calc = 0;
    FT_calc = 0;
    indLab = 0;
    modeOrder = 0;
    return
end
indicies = Modes(numberOfLines, maxOrder);
indicies = double(indicies);
newIndicies = indicies.*2.*pi./repmat(spacing,size(indicies,1),1);
X_coord = sum(newIndicies .* repmat(cosd(moireAngles),size(indicies,1),1),2);
Y_coord = sum(newIndicies .* repmat(sind(moireAngles),size(indicies,1),1),2);
switch type
    case 1
        angle = zeros(length(EmInt),2,size(X_coord,1));
        k = zeros(length(EmInt),2,size(X_coord,1));
        for kk = 1:size(X_coord,1)
            xx = X_coord(kk);
            yy = Y_coord(kk);
            LHS =
(EnergyJoule./(hBar.*c)).^2.*1./(1./dielectric+1./EmInt(:,2)); % LHS=beta in
Maier Ch. 2
            LHS = real(LHS);
            aa = 1;
            bb = 2.*(xx.*cosd(azimuthalAngle) + yy.*sind(azimuthalAngle));
            cc = (xx.^2+yy.^2)-LHS;
            kpos = (-bb+(bb.^2-4.*aa.*cc).^0.5)./(2.*aa);
            kneg = (-bb-(bb.^2-4.*aa.*cc).^0.5)./(2.*aa);
            kpos = real(kpos)./1e6;
            kneg = real(kneg)./1e6;
            k(:,1,kk)= kpos;
            k(:,2,kk)= kneg;
            angle(:,1,kk)=
asin((real(kpos*1e6).*c)./(EnergyJoule./hBar)).*(180./pi);
            angle(:,2,kk)=
asin((real(kneg*1e6).*c)./(EnergyJoule./hBar)).*(180./pi);
        end
        orderTemp = sum(abs(indicies),2);
        rotateMat = [cosd(-azimuthalAngle) -sind(-azimuthalAngle); sind(-
azimuthalAngle) cosd(-azimuthalAngle)];
        proj = [(rotateMat*[X_coord Y_coord].').'./1e6 orderTemp];
        kkind = (abs(real(diff(k))) < 1e-10); %look for ==0, but allow for
rounding errors
```

```
        kkind = cat(1, kkind, true([1 size(kkind,2) size(kkind,3)]));
        %%%% this "reals" thing breaks using line, dont know why right now...
        %%%% the following 3 commented lines are removed to make it work
        %%%%~Alex 11/29/2011
        % reals = find(sum(kkind(:,1,:))~=1);%find the ones that are not all
zero
        k(kkind) = 0;
        angle(kkind) = 0;
        % k = k(:,:,reals); %only take the ones that are not all zero
        % angle = angle(:,:,reals); %only take the ones that are not all zero
        Ek_calc = {zeros(size(k,1),2*kk) zeros(size(k,1),2*kk)};
        Lt_calc = {zeros(size(angle,1),2*kk) zeros(size(angle,1),2*kk)};
        dd = 1; %counter
        for mm = 1:size(k,3)
            %only pick the ones that cross into the first brillouin zones
            if max(real(k(:,1,mm)))>-pi/(min(spacing)*1e6) &&
min(real(k(:,1,mm))) < pi/(min(spacing)*1e6) %check positive solutions
                real_k = real(k(:,1,mm));
                real_angle = real(angle(:,1,mm));
                if dd>1
                    %find any degenerate vectors fom previous set
                    degen = (abs(bsxfun(@minus,Ek_calc{1}(:,1:dd-
1),real_k))<1e-10);
                    degen_ind=sum(~degen)==0;
                    if sum(degen_ind)
                        %if degenerate, append the indicies with new index
                        indLab{degen_ind,1}{end+1} = ['['
int2str(indicies(mm,:)) ']'];
                        modeOrder(degen_ind) =
min(modeOrder(degen_ind),orderTemp(mm));
                    else
                        %add to the list
                        Ek_calc{1}(:,dd) = real_k;
                        Ek_calc{2}(:,dd) = Energy;
                        Lt_calc{1}(:,dd) = real_angle;
                        Lt_calc{2}(:,dd) = lambda;
                        indLab{dd,1} = {['[' int2str(indicies(mm,:)) ']']};
                        modeOrder(dd) = orderTemp(mm);
                        dd = dd+1;
                    end
                else
                    %add the first one to the list
                    Ek_calc{1}(:,dd) = real_k;
                    Ek_calc{2}(:,dd) = Energy;
                    Lt_calc{1}(:,dd) = real_angle;
                    Lt_calc{2}(:,dd) = lambda;
                    indLab{dd,1} = {['[' int2str(indicies(mm,:)) ']']};
                    modeOrder(dd) = orderTemp(mm);
                    dd = dd+1;
                end
            end
            if max(real(k(:,2,mm)))>-pi/(min(spacing)*1e6) &&
min(real(k(:,2,mm))) < pi/(min(spacing)*1e6) %check negative solutions
                real_k = real(k(:,2,mm));
                real_angle = real(angle(:,2,mm));
        if dd>1
          %find any degenerate vectors fom previous set
          degen = (abs(bsxfun(@minus,Ek_calc{1}(:,1:dd-
1),real_k))<1e-10);
          degen_ind=sum(~degen)==0;
          if sum(degen_ind)
            %if degenerate, append the indicies with new index
            indLab{degen_ind,1}{end+1} = ['['
int2str(indicies(mm,:)) ']'];
            modeOrder(degen_ind) =
min(modeOrder(degen_ind),orderTemp(mm));
          else
            %add to the list
            Ek_calc{1}(:,dd) = Ek_calc{1}(:,dd)+real_k;
            Ek_calc{2}(:,dd) = Energy;
            Lt_calc{1}(:,dd) = Lt_calc{1}(:,dd)+real_angle;
            Lt_calc{2}(:,dd) = lambda;
            indLab{dd,1} = {['[' int2str(indicies(mm,:)) ']']};
            modeOrder(dd) = orderTemp(mm);
            dd = dd+1;
          end
        else
          %add the first one to the list
          Ek_calc{1}(:,dd) = Ek_calc{1}(:,dd)+real_k;
          Ek_calc{2}(:,dd) = Energy
          Lt_calc{1}(:,dd) = Lt_calc{1}(:,dd)+real_angle;
          Lt_calc{2}(:,dd) = lambda;
          indLab{dd,1} = {['[' int2str(indicies(mm,:)) ']']};
          modeOrder(dd) = orderTemp(mm);
          dd = dd+1;
        end
      end
    end
    ordering = sortrows([proj indicies], -3);
    proj = ordering(:,1:3);
    indicies = ordering(:,4:end);
    FT_calc = {proj indicies};
    %remove the empty rows (from preallocation) and turn all the zeros to
NaN
    %to show proper plot
    Ek_calc{1}(:,dd:end) = [];
    makeNaN = Ek_calc{1}==0;
    Ek_calc{1}(makeNaN)=NaN;
    Ek_calc{2}(:,dd:end) = [];
    makeNaN = Ek_calc{2}==0;
    Ek_calc{2}(makeNaN)=NaN;
    Lt_calc{1}(:,dd:end) = [];
    makeNaN = Lt_calc{1}==0;
    Lt_calc{1}(makeNaN)=NaN;
    Lt_calc{2}(:,dd:end) = [];
    makeNaN = Lt_calc{2}==0;
    Lt_calc{2}(makeNaN)=NaN;
  case 2
    phiout = zeros(2*length(EmInt),length(X_coord));
    for kk = 1:length(X_coord)
```

```
            ii = X_coord(kk);
            jj = Y_coord(kk);
            kspp = (EnergyJoule./(hBar.*c)).*(1./dielectric+1./EmInt(:,2)).^-
0.5;
            magG = (ii^2+jj^2)^0.5;
            kpar = EnergyJoule./(hBar*c).*sind(excitationAngle);
            LHS = (kspp.^2-magG.^2-kpar.^2)./(2.*kpar.*magG);
            im = imag(LHS)~=0;
            LHS(im) = NaN;
            chi = atan2(jj,ii)*(180/pi);%could be from -180 to 180
            phi = asind(LHS)-chi; %could be from -270 to 270
            phi = real(phi);
            phi2 = (180-asind(LHS))-chi; %the upper half of asind
            phi2 = real(phi2);
            %shift phi and phi2 so that [0,180] is covered
            if min(phi) < -180
                phi = 360+phi;
                phiout(:,kk) = [phi2; phi];
            elseif min(phi) < 0
                phiout(:,kk) = [phi; phi2];
            elseif min(phi) < 180
                phi2 = phi2-360;
                phiout(:,kk) = [phi2; phi];
            else
                phi = phi-360;
                phi2 = phi2-360;
                phiout(:,kk) = [phi; phi2];
            end
        end
        modeOrder = sum(abs(indicies),2);
        proj = [X_coord./1e6 Y_coord./1e6 modeOrder];
        ordering = sortrows([proj indicies], -3);
        proj = ordering(:,1:3);
        indicies = ordering(:,4:end);
        FT_calc = {proj indicies};
        phiind = (abs(diff(phiout)) < 1e-10); %look for ==0, but allow for
rounding errors
        phiind = cat(1,phiind, true([1 size(phiind,2)])) & cat(1, true([1
size(phiind,2)]),phiind);
        phiout(phiind) = NaN;
        Energyout = repmat(Energy,2,size(phiout,2));
        Energyout(phiind) = NaN;
        lambdaout = repmat(lambda,2,size(phiout,2));
        lambdaout(phiind) = NaN;
        Ek_calc = {phiout Energyout};
        Lt_calc = {phiout lambdaout};
        indLab = arrayfun(@(X){['[' int2str(indicies(X,:))
']']},1:size(indicies,1),'uni',false);
end
ordering = sortrows([proj indicies], -3);
proj = ordering(:,1:3);
indicies = ordering(:,4:end);
end
function [wavelengths, backgroundAngles, dataAngles, backgroundData,
saveData, energyData, wavelengthData, type] = ...
```

```
ProcessData(rawData,rawBackground,scale,dataType,isPIdata,angleInput,smoothSp
an,smoothType,FDTDwavelengths,FDTDangle)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%CCD takes 1340 data points%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Load the Background data
switch dataType
    case 2 %Processed Data
        if ~isempty(rawBackground)
            backgroundData = rawBackground;
            if backgroundData(1,1) == 1
                isAzimuthalData = true;
                backgroundData(1,1) = 0;
            else
                isAzimuthalData = false;
            end
            if backgroundData(1,1) == 0
                backgroundAngles = backgroundData(1,2:end);
                backgroundData(1,:) = [];
            else
                reloadans = inputdlg({'What is the range of angles used? (In
matlab syntax)'},'Angle Input',1,{'0'});
                backgroundAngles = eval(reloadans{1});
            end
            wavelengths = backgroundData(:,1);
            backgroundData(:,1) = [];
        else
            backgroundData = 0;
            backgroundAngles = 0;
        end
    case 3 % FDTD Data
        backgroundAngles = 0;
        backgroundData = 0;
        isAzimuthalData = false;
    case 1 % Raw Data
        if isPIdata
            spectraLength = find(rawBackground(:,1)>rawBackground(1),1);
            if isempty(spectraLength)
                numberOfSpectra = length(rawBackground(:,2))/1340;
            else
                numberOfSpectra = (spectraLength-1)/1340;
            end
            backgroundAngles =
rawBackground(1:(1340*numberOfSpectra):end,1)';
            wavelengths = rawBackground(1:(1340*numberOfSpectra),2);
            backgroundData =
reshape(rawBackground(:,3),length(wavelengths),length(backgroundAngles));
            isAzimuthalData = false;
        else

[backgroundAngles,backgroundCenterWavelength,backgroundGrating,backgroundData
,isAzimuthalData] = extractData(rawBackground);
            wavelengths =
pixel2wavelength(backgroundGrating,backgroundCenterWavelength);
```

```
            backgroundData =
calibrateData(wavelengths,backgroundData,backgroundCenterWavelength,backgroun
dGrating);
            isAzimuthalData = false;
        end
        %smooth background
        backgroundData = sortrows([wavelengths (1:size(backgroundData,1))'
backgroundData]);
        numlist = backgroundData(:,2);
        if ~isempty(smoothSpan) && ~isempty(smoothType)
            for ii = 3:size(backgroundData,2)
                backgroundData(:,ii) =
smooth(backgroundData(:,ii),smoothSpan,smoothType);
            end
        end
        backgroundData = sortrows([numlist backgroundData]);
        backgroundData(:,1:3)=[];
end
%% Process Data
switch dataType
    case 2 % Processed Data
        data = rawData;
        switch data(1,1)
            case {0,1,2}
                type = data(1,1)+1;
                dataAngles = data(1,2:end);
                data(1,:) = [];
            otherwise % processed data before v3.0
                reloadans = inputdlg({'What was the range of angles used? (In
matlab syntax)'},...
                    'Angle Input',1,{'10:60'});
                dataAngles = eval(['[' reloadans{1} ']']);
        end
        wavelengths = data(:,1);
        saveData = data;
        data = data(:,2:end);
    case 3 %FDTD Data
        data = rawData';
        dataAngles = FDTDangle';
        wavelengths = FDTDwavelengths;
        %wavelengths = FDTDwavelengths.*1000;
        %make data matrix
        data = [wavelengths data];
        %data = flipud(data);
        %wavelengths = flipud(wavelengths);
        FDTDans = inputdlg({'What index to use for adjusting the angles?',...
            sprintf('What range of angles to show of
[%.1f, %0.1f]?',dataAngles(1),dataAngles(end))},...
            'FDTD Data',1,{'1',sprintf('[%.1f, %.1f]',dataAngles(1),
dataAngles(end))});
        %ask for dielectric index
        if ~isempty(FDTDans)
            FDTDind = str2double(FDTDans{1});
            dataAngles = asind(sind(dataAngles).*FDTDind);
            range = eval(FDTDans{2});
            angindex = (dataAngles>=range(1)) & (dataAngles<=range(2));
            data = data(:,[true angindex]);
            dataAngles = dataAngles(angindex);
        end
        saveData = data;
        data = data(:,2:end);
        type = 1;
    case 1 % Raw Data
        if isPIdata
            spectraLength = find(rawData(:,1)>rawData(1),1);
            if isempty(spectraLength)
                numberOfSpectra = length(rawData(:,2))/1340;
            else
                numberOfSpectra = (spectraLength-1)/1340;
            end
            dataAngles = rawData(1:(1340*numberOfSpectra):end,1)';
%             if ~isempty(angleInput)
%                 dataAngles = angleInput(1):angleInput(3):angleInput(2);
%             end
            % wavelengths = rawData{2}(1:(1340*numberOfSpectra));
            data = 
reshape(rawData(:,3),length(wavelengths),length(dataAngles));
%             data = quickCalibrateData(wavelengths,data);
        else
            [dataAngles,centerWavelength,grating,data,isAzimuthalData] = 
extractData(rawData);
            data = calibrateData(wavelengths,data,centerWavelength,grating);
        end
        if min(size(data))==1
            type = 3;
        elseif isAzimuthalData
            type = 2;
        else
            type = 1;
        end
        %smooth data
        data = sortrows([wavelengths (1:size(data,1))' data]);
        numlist = data(:,2);
        if ~isempty(smoothSpan)
            for ii = 3:size(data,2)
                data(:,ii) = smooth(data(:,ii),smoothSpan,smoothType);
            end
        end
        data = sortrows([numlist data]);
        data(:,1:3)=[];
        %scale data with background
        if size(backgroundData,2) == 1
            data = data./(repmat(backgroundData,1,size(data,2)).*scale);
        else
            data = data./(backgroundData.*scale);
        end
        saveData(:,2:(size(data,2)+1)) = data;
        saveData(:,1) = wavelengths;
        backgroundTemp(:,2:(size(backgroundData,2)+1)) = backgroundData;
        backgroundTemp(:,1) = wavelengths;
        backgroundTemp = sortrows(backgroundTemp);
        backgroundData = backgroundTemp(:,2:end);
        saveData = sortrows(saveData);
        wavelengths = saveData(:,1);
        data = saveData(:,2:end);
end
%% Interpolate Data
%Set image resolution (completely arbitrary)...
Xpixels = (700*2);
Ypixels = (1068*2);
numberOfAngles = length(dataAngles);
numberOfWavelengths = length(wavelengths);
if type == 1
    %Build photoenergy array
    %Energy in eV = 1240/lamda(nm)
    energies = 1240./wavelengths;
    %Build the wavevector array (in 1/um)for the dispersion curve
    wavevectors=1000.*(2.* pi./wavelengths)*sind(dataAngles);
    %Build dispersion curve map
    %vec_max=max(max(vectordata));
    maxWavevector=max(wavevectors(:));
    minWavevector=min(wavevectors(:));
    maxEnergy=max(energies);
    minEnergy=min(energies);
    XpixelWidth = (maxWavevector - minWavevector)/(Xpixels-1);
    YpixelWidth = (maxEnergy - minEnergy)/(Ypixels-1);
    %mapdata = zeros(Ypixels, Xpixels);
    mapdata2 = NaN(Ypixels,Xpixels);
    ncounter = zeros(1,Ypixels);
    %loop for the interpolation between angles (columns)
    for i=1:numberOfWavelengths
        n = floor((energies(i)-minEnergy)/YpixelWidth)+1;
%          if i == 1
%              %this initializes the matrix, since evergies are high->low
%              ncounter(1,n) = 0;
%          end
        %ncounter is how many data are crammed into this point
        if ncounter(n) == 0
            for j=1:numberOfAngles
                m = floor((wavevectors(i,j)-minWavevector)/XpixelWidth)+1;
                %mapdata(n,m)= data(i,j);
                %interpolate along the row
                if j~=1
                    mapdata2(n,m_old:m)= linspace(data(i,j-1),data(i,j),m-
m_old+1);
                end
                m_old=m;
            end
        else
            Nordata2 = mean(data(i-ncounter(n):i,:)); %average data instead
of rewriting
            for j=1:numberOfAngles
                m = floor((wavevectors(i,j)-minWavevector)/XpixelWidth)+1;
                %mapdata(n,m)= Nordata2(j);
                %interpolate along the row
                if j~=1
                    mapdata2(n,m_old:m)= linspace(Nordata2(j-
1),Nordata2(j),m-m_old+1);
                end
                m_old=m;
            end
        end
        ncounter(n) = ncounter(n)+1;
    end
    %interpolate between rows
    filledn = find(ncounter~=0);
    energyData =
interp2(1:Xpixels,filledn',mapdata2(filledn,:),1:Xpixels,(1:Ypixels)');
elseif type == 2
    ind = [false; diff(wavelengths)<=1e-6];
    wavelengths = wavelengths(~ind);
    data = data(~ind,:);
    energies = sortrows(1240./wavelengths);
    [X, Y] = meshgrid(dataAngles, energies);

energyData=interp2(X,Y,flipud(data),linspace(dataAngles(1),dataAngles(end),Xp
ixels),linspace(energies(1),energies(end),Ypixels)');
elseif type == 3
    ind = [false; diff(wavelengths)<=1e-6];
    wavelengths = wavelengths(~ind);
    data = data(~ind);
    energyData = flipud(data);
end
%interpolate angle data
ind = [false; diff(wavelengths)<=1e-6];
wavelengths1 = wavelengths(~ind);
data1 = data(~ind,:);
if ~(backgroundData==0)
    backgroundData = backgroundData(~ind,:);
end
if type == 1 || type == 2
    [X, Y] = meshgrid(dataAngles, wavelengths1);

wavelengthData=interp2(X,Y,data1,linspace(dataAngles(1),dataAngles(end),Xpixe
ls),linspace(wavelengths1(1),wavelengths1(end),Ypixels)');
elseif type == 3
    wavelengthData = data;
end
end
function [BZ] = BZcalc(XYcoordinates,order)
Xlimits = [min(XYcoordinates(:,1)); max(XYcoordinates(:,1))];
tooHigh = (order > 2); %hard limit of 2 orders
XYcoordinates(tooHigh,:) = [];
order(tooHigh) = [];
BZ = {cell(sum(order==1),1), cell(sum(order==2),1)};
counter1 = 1;
counter2 = 1;
for ii = 1:size(XYcoordinates,1)
    % no zone for the zero order point
    if order(ii) == 0
```

```
        continue
    end
    X0 = XYcoordinates(ii,1); Y0 = XYcoordinates(ii,2);
    point = ([X0 Y0]/2);
    slope = -X0/Y0;
    if abs(slope) < 1
        X = Xlimits;
        Y = slope.*(X-point(1))+point(2);
    else
        Y = Xlimits;
        X = 1./slope.*(Y-point(2))+point(1);
    end
    if order(ii) == 1
        BZ{1}{counter1} = [X Y];
        counter1 = counter1 + 1;
    else
        BZ{2}{counter2} = [X Y];
        counter2 = counter2 + 1;
    end
end
end
function newfig_bdfcn(obj,~,fig,lines)
seltype = get(fig,'selectiontype');
switch seltype
    case 'normal'
        width = 2;
    case 'alt'
        width = 1;
    case 'open'
        set(lines(:),'linewidth',1);
        return
    otherwise
        width = get(obj,'linewidth');
end
set(obj,'linewidth',width);
drawnow;
end
function [angles, centerWavelengths, grating, data, isAzimuthalData] =
extractData(rawData)
%columns of angles, stacked gratings...
if mod(size(rawData,1),1342)==2 && (size(rawData,2)==1 ||
rawData(4,1)==rawData(4,2)) %...azim included (check same center wavelength)
    angles = rawData(2,:); %extract azimuthal angles
    rawData(1:2,:) = []; %removes angles headers
    grating = rawData(1:1342:end,1).'; %extract grating numbers
    rawData(1:1342:end,:) = []; %removes grating headers
    centerWavelengths = rawData(1:1341:end,1).'; %extract center wavelengths
    rawData(1:1341:end,:) = []; %removes center headers
    data = rawData;
    isAzimuthalData = true;
elseif mod(size(rawData,1),1342)==1 && (size(rawData,2)==1 ||
rawData(3,1)==rawData(3,2)) %...no azim (check same center wave)
    angles = rawData(1,:); %extract angles
    rawData(1,:) = []; %removes angle header
    grating = rawData(1:1342:end,1).'; %extract grating numbers
    rawData(1:1342:end,:) = []; %removes grating header
    centerWavelengths = rawData(1:1341:end,1).'; %extract center wavelengths
    rawData(1:1341:end,:) = []; %removes center header
    data = rawData;
    isAzimuthalData = false;
    %columns of gratings, stacked angles...
elseif mod(size(rawData,1),1344)==0 && (size(rawData,2)==1 ||
rawData(4,1)~=rawData(4,2)) %...azim included (check diff center wavelength)
    rawData(1:1343:end,:) = []; %removes angle header
    angles = rawData(1:1343:end,1).'; %extract azimuthal angles
    rawData(1:1343:end,:) = []; %removes azimuthal angle header
    grating = rawData(1,:); %extract grating numbers
    rawData(1:1342:end,:) = []; %removes grating header
    centerWavelengths = rawData(1,:); %extract center wavelengths
    rawData(1:1341:end,:) = []; %removes center header
    %reposition data into appropriate columns/rows
    temp = reshape(rawData, 1340, size(rawData,2)*size(angles,2));
    data(1340*size(rawData,2), size(angles,2)) = 0;
    for ii = 1:(size(rawData,2))
        data(((ii-1)*1340+1):(ii*1340),:) = temp(:,((ii-
1)*size(angles,2)+1):(ii*size(angles,2)));
    end
    isAzimuthalData = true;
elseif mod(size(rawData,1),1343)==0 && (size(rawData,2)==1 ||
rawData(3,1)~=rawData(3,2)) %...no azim (check diff center wavelength)
    angles = rawData(1:1343:end,1).'; %extract angles
    rawData(1:1343:end,:) = []; %removes angle header
    grating = rawData(1,:); %extract grating numbers
    rawData(1:1342:end,:) = []; %removes grating header
    centerWavelengths = rawData(1,:); %extract center wavelengths
    rawData(1:1341:end,:) = []; %removes center header
    %reposition data into appropriate columns/rows
    temp = reshape(rawData, 1340, size(rawData,2)*size(angles,2));
    data(1340*size(rawData,2), size(angles,2)) = 0;
    for ii = 1:(size(rawData,2))
        data(((ii-1)*1340+1):(ii*1340),:) = temp(:,((ii-
1)*size(angles,2)+1):(ii*size(angles,2)));
    end
    isAzimuthalData=false;
end
%get rid of data beyond calibration
switch grating(1)
    case {2,3}
        extra = sum(centerWavelengths > 990);
    case 1
        extra = sum(centerWavelengths > 1074);
end
grating(end-extra+1:end) = [];
centerWavelengths(end-extra+1:end) = [];
data(end-(1340*extra)+1:end,:) = [];
end
function data = quickCalibrateData(wavelengths,data)
numberOfParts = size(data,1)/1340;
numberOfAngles = size(data,2);
scaleEff = ones(numberOfParts+1,numberOfAngles);
```

```
testdata = data;
for pp = 1:numberOfParts-1
    part1 = data(1340*(pp-1)+1:1340*pp,:);
    part2 = data(1340*pp+1:1340*(pp+1),:);
    wavelengthsPart1 = wavelengths(1340*(pp-1)+1:1340*pp);
    wavelengthsPart2 = wavelengths(1340*pp+1:1340*(pp+1));
    overlapRegion1 = wavelengthsPart1>=wavelengthsPart2(1);
    overlapRegion2 = wavelengthsPart2<=wavelengthsPart1(end);
    interpNumber = round(min(sum(overlapRegion1),sum(overlapRegion2))./2);
    for aa = 1:numberOfAngles
        interpPart1 =
interp1(wavelengthsPart1(overlapRegion1),part1(overlapRegion1,aa),linspace(mi
n(wavelengthsPart1(overlapRegion1)),max(wavelengthsPart2(overlapRegion2)),int
erpNumber));
        interpPart2 =
interp1(wavelengthsPart2(overlapRegion2),part2(overlapRegion2,aa),linspace(mi
n(wavelengthsPart1(overlapRegion1)),max(wavelengthsPart2(overlapRegion2)),int
erpNumber));
        scaleEff(pp+1,aa) = lsqlin(interpPart2',interpPart1,[],[]);
        data(1340*pp+1:end,aa) = data(1340*pp+1:end,aa).*scaleEff(pp+1,aa);
    end
end
end
function data = calibrateData(wavelengths,data,centerWavelength,grating)
eff = [];
centerWave = [];
dark = [];
load('calibration.mat')
centerWave = centerWave{grating(1)};
eff = eff{grating(1)};
numberOfParts = size(data,1)/1340;
numberOfAngles = size(data,2);
%subtract dark current
data = data - repmat(dark,numberOfParts,numberOfAngles);
%apply spectrum calibration to data
scaleEff = ones(numberOfParts+1,numberOfAngles);
testdata = SpectScale(centerWavelength,data,centerWave,eff,scaleEff);
%apply vertical scaling to parts of spectra
for pp = 1:numberOfParts-1
    part1 = testdata(1340*(pp-1)+1:1340*pp,:);
    part2 = testdata(1340*pp+1:1340*(pp+1),:);
    overlapRegion1 = wavelengths(1340*(pp-
1)+1:1340*pp)>=wavelengths(1340*pp+1);
    overlapRegion2 =
wavelengths(1340*pp+1:1340*(pp+1))<=wavelengths(1340*pp);
    scaleEff(pp+1,:) =
arrayfun(@(A)lsqlin(part2(overlapRegion2,A),part1(overlapRegion1,A),[],[]),1:
numberOfAngles);
    % data(1340*pp+1:end,:) =
data(1340*pp+1:end,:).*repmat(scaleEff(pp+1,:),1340*(numberOfParts-pp),1);
end
data = SpectScale(centerWavelength,data,centerWave,eff,scaleEff);
end
function wavelengths = pixel2wavelength(grating,center)
%calibration from pixel to wavelength: W = m*P + b
```

```
%index is grating number (1 = 1200 groove/mm 500 nm blaze, 2 = 300 groove/mm
%300 nm blaze,3 = 300 groove/mm 500 nm blaze)
%
%Calibration w/ Hg spectrum for gratings 1&3 by Alex Hryn: 28 May 2012
m(1) = .02792;
m(2) = .13046;
m(3) = .13046;
%------X
offset(1) = m(1)*670.5 + 557.53 - 576.655;
offset(2) = m(2)*670.5 + 433.90 - 524.12;
offset(3) = m(3)*670.5 + 433.90 - 524.12;
%------X
b(1,:) = (center + offset(1)) - m(1).*670.5;
b(2,:) = (center + offset(2)) - m(2).*670.5;
b(3,:) = (center + offset(3)) - m(3).*670.5;
waveconv = @(p,g,c) m(g).*p + b(g,c);
%waveconv takes p: pixel , g: grating number, c: center wavelength and
%gives the wavelength in nm
%make wavelength vector
temp =
arrayfun(@(g,c)waveconv((1:1340)',g,c),grating,1:length(center),'UniformOutpu
t',false);
wavelengths = vertcat(temp{:});
end
function data = SpectScale(center,data,centerWave,eff,scaleEff)
%data comes in as angle columns of stacked gratings
for pp = 1:size(center,2)
    ind=find(centerWave >= center(pp),1);
    x = (center(pp)-centerWave(ind-1))/(centerWave(ind)-centerWave(ind-1));
    for ii = 1:size(data,2)

factor=((prod(scaleEff(1:pp,ii)).^0.5)./(prod(scaleEff(pp+1:end,ii)).^0.5)).*
eff;
        data(1340*(pp-1)+1:1340*pp,ii) = data(1340*(pp-
1)+1:1340*pp,ii).*(x.*factor(ind,:)+(1-x).*factor(ind-1,:))';
    end
end
end
function totalSolutions = Modes(nLines, maxorder)
nSolutions = 1;
for ii = 1:maxorder
    nSolutions = nSolutions +
sum(arrayfun(@(X)nchoosek(nLines,X)*2^X*nchoosek(ii-1,(X-
1)),1:ii,'ErrorHandler',@(x,y)0));
    %     nSolutions = nSolutions +
sum(arrayfun(@(X)nchoosek(nLines,X)*2^X*nchoosek(ii-1,(X-
1)),1:min(ii,nLines)));
end
solution = zeros(1,nLines,'int8');
totalSolutions = zeros(nSolutions,nLines,'int8');
nSolutions = 0;
for order = 0:maxorder
    check(nLines, order);
end
    function check(nLines, order)
```

```
        nLinesRemaining = nLines - 1;
        if nLinesRemaining == 0
            if order >=0 && order <=maxorder
                nSolutions = nSolutions+1;
                solution(nLines) = order;
                totalSolutions(nSolutions,:) = solution;
                if order > 0
                    nSolutions = nSolutions+1;
                    solution(nLines) = -order;
                    totalSolutions(nSolutions,:) = solution;
                end
            end
            return;
        end
        for numroll = -maxorder:maxorder
            targetRemaining = order - abs(numroll);
            if targetRemaining >= 0 && targetRemaining <=
maxorder*nLinesRemaining
                solution(nLines) = numroll; %record this value
                check(nLinesRemaining,targetRemaining);
            end
        end
    end
end
function [xfigure, yfigure]=axescoord2figurecoord(varargin)
% AXESCOORD2FIGURECOORD Transform axes coordinates in current
% figure units coordinate to the figure for annotation location
% [xfigure, yfigure]=axescoord2figurecoord(xaxes,yaxes)
% [xfigure, yfigure]=axescoord2figurecoord(xaxes,yaxes,handle_axes)
%
% Ex.
%       % Create some data
%       t = 0:.1:4*pi;
%       s = sin(t);
%
%       % Add an annotation requiring (x,y) coordinate vectors
%       plot(t,s);ylim([-1.2 1.2])
%       set(gcf,'Units','normalized');
%       xa = [1.6 2]*pi;
%       ya = [0 0];
%       [xaf,yaf] = axescoord2figurecoord(xa,ya);
%       annotation('arrow',xaf,yaf)
%
% Acknowledgments are due to Scott Hirsch (shirsch@mathworks.com) for is
% function ds2nfu. Some part of the present function derived from ds2nfu.
%
% Valley Beno”t / Jan 2007
% valley@erdw.ethz.ch
% Process inputs
narginchk(2, 3)
if nargin==2
    xaxes=varargin{1};
    yaxes=varargin{2};
    h_axes = get(gcf,'CurrentAxes');
else
```

```
    xaxes=varargin{1};
    yaxes=varargin{2};
    h_axes = varargin{3};
end
% get axes properties
funit=get(get(h_axes,'Parent'),'Units');
% get axes properties
aunit=get(h_axes,'Units');
darm=get(h_axes,'DataAspectRatioMode');
pbarm=get(h_axes,'PlotBoxAspectRatioMode');
dar=get(h_axes,'DataAspectRatio');
pbar=get(h_axes,'PlotBoxAspectRatio');
xlm=get(h_axes,'XLimMode');
ylm=get(h_axes,'YLimMode');
xd=get(h_axes,'XDir');
yd=get(h_axes,'YDir');
% set the right units for h_axes
set(h_axes,'Units',funit);
axesoffsets = get(h_axes,'Position');
x_axislimits = get(h_axes, 'xlim');     %get axes extremeties.
y_axislimits = get(h_axes, 'ylim');     %get axes extremeties.
x_axislength = x_axislimits(2) - x_axislimits(1); %get axes length
y_axislength = y_axislimits(2) - y_axislimits(1); %get axes length
% mananged the aspect ratio problems
set(h_axes,'units','centimeters');
asc=get(h_axes,'Position');
rasc=asc(4)/asc(3);
rpb=pbar(2)/pbar(1);
if rasc<rpb
    xwb=axesoffsets(3)/rpb*rasc;
    xab=axesoffsets(1)+axesoffsets(3)/2-xwb/2;
    yab=axesoffsets(2);
    ywb=axesoffsets(4);
elseif rasc==rpb
    xab=axesoffsets(1);
    yab=axesoffsets(2);
    xwb=axesoffsets(3);
    ywb=axesoffsets(4);
else
    ywb=axesoffsets(4)*rpb/rasc;
    yab=axesoffsets(2)+axesoffsets(4)/2-ywb/2;
    xab=axesoffsets(1);
    xwb=axesoffsets(3);
end
if strcmp(darm,'auto') & strcmp(pbarm,'auto')
    xab=axesoffsets(1);
    yab=axesoffsets(2);
    xwb=axesoffsets(3);
    ywb=axesoffsets(4);
end
% compute coordinate taking in account for axes directions
if strcmp(xd , 'normal')==1
    xfigure = xab+xwb*(xaxes-x_axislimits(1))/x_axislength;
else
    xfigure = xab+xwb*(x_axislimits(2)-xaxes)/x_axislength;
```

```
end
if strcmp(funit,'normalized');
    xfigure(find(xfigure>1))=1;
    xfigure(find(xfigure<0))=0;
end
if strcmp(yd , 'normal')==1
    yfigure = yab+ywb*(yaxes-y_axislimits(1))/y_axislength;
else
    yfigure = yab+ywb*(y_axislimits(2)-yaxes)/y_axislength;
end
if strcmp(funit,'normalized');
    yfigure(find(yfigure>1))=1;
    yfigure(find(yfigure<0))=0;
end
set(h_axes,'Units',aunit); % put axes units back to original state
end
function a = struct2array(s)
%STRUCT2ARRAY Convert structure with doubles to an array.
%   Author(s): R. Losada
%   Copyright 1988-2013 The MathWorks, Inc.
narginchk(1,1);
% Convert structure to cell
c = struct2cell(s);
% Construct an array
a = [c{:}];
end
```

APPENDIX C: LUMERICAL SCRIPT TO CREATE BOWTIE DIMERS

```
for (tl = 40) {
for (th = 50) {
#selectall;delete;clear;
selectpartial("particle");delete;
select("Si mask");delete;
#eval("counter = 1;");
redrawoff;
######################
# Change things here #
######################
d = .350; #width of the pit - all is in microns
w = 1; # width of the masked region
d1 = .260; #diameter in +x direction of hole
d2 = d1; #diameter in +y direction of hole
nSides = 95; #number of sides of regular polygon
defaultThicknesses = [th/2, th, th/2]*1e-3; #thicknesses of each step of
particle
resolution = 0.001;
depositionAngles = [tl, tl, tl]; #degrees from +z axis towards +x axis
azimAngles = [0, 180, 0]+45; #degrees from +x axis towards +y axis
holeAzim = 0; #degrees from +x axis towards +y axis
startFilmThickness = 0.008; #top film thickness
startParticleHeight = [0, 0, 1]; #particle start for offset
defaultMaterial = "Au (Gold) - Johnson and Christy";
meshOrder = 3;
maskMaterial = "<Object defined dielectric>";
maskIndex = 1;
maskMeshOrder = 2;
numberOfDepositions = 3;
drawParticle = [1, 1, 1];
######################
######################
numpoints = nSides+1; # +1 for linspace
#Define the Si Pit properties
if(getnamednumber("Si mask")==0)
{
makeSiPyramid;
select("Si mask");
adduserprop("pit size",0,d);
}
SiAngles = (0:90:360)*pi/180;
SiAngles = SiAngles(1:length(SiAngles)-1);
# organize the particle heights for offsets
allParticleHeights = matrix(length(startParticleHeight),length(SiAngles));
defaultThickness=0;
normalOffset=0;
glancingOffset = matrix(1,numpoints);
#make the initial hole
r1 = d1/2;
r2 = d2/2;
holeAzim = holeAzim*pi/180;
# make some regular 2D shapes in standard basis centered at origin
```

```
phi = linspace(0,2*pi,numpoints); #overlap the final point to close the shape
x0 = r1 * cos(phi);
y0 = r2 * sin(phi);
hole = [x0,y0];
azimMatrix = [cos(holeAzim),-sin(holeAzim);sin(holeAzim),cos(holeAzim)];
hole = transpose(mult(azimMatrix,transpose(hole)));
startHole = hole;
for (nD = 1:numberOfDepositions) {
counter = nD;
defaultThickness = defaultThicknesses(nD);
tilt = depositionAngles(nD);
azim = azimAngles(nD);
#gap = 0.030;
#matlabput(d,d1,defaultThickness,holeCloseRate,gap);
#matlab("tilt =
fsolve(@(X)(findparticlegap(X,d,d1,defaultThickness,holeCloseRate)-
gap),50);");
#matlabget(tilt);
tilt = (90-tilt)*pi/180;
azim = azim*pi/180;
if(startParticleHeight(nD)==0) {
currentParticleHeight =  matrix(1,length(SiAngles));
} else {
previousHeightIndex = startParticleHeight(nD);
currentParticleHeight =
allParticleHeights(previousHeightIndex,1:length(SiAngles));
}
finalFilmThickness = startFilmThickness + defaultThickness*sin(tilt);
# find total hole closing from deposition parameters
normalClosing =  0.3*defaultThickness*sin(tilt); #nm/nm deposited at normal
incidence
glancingClosing = 0.3*defaultThickness*cos(tilt); #growth on one side of hole
filmEdgeClosingI = startFilmThickness/tan(tilt); #shift due to tilted
deposition
filmEdgeClosingF = finalFilmThickness/tan(tilt); #shift due to tilted
deposition
# counter rotate shape against deposition direction
depAzimMatrix = [cos(-azim),-sin(-azim);sin(-azim),cos(-azim)];
startHole = transpose(mult(depAzimMatrix,transpose(startHole)));
## make the hole at start and end of deposition ##
# get the coordinates
x0=startHole(1:numpoints,1);
y0=startHole(1:numpoints,2);
#plot(x0,y0);
# shrink due to glancing closing
V1=[x0,y0];
V2=[x0+glancingClosing,y0];
V3=polyand(V1,V2); # polygon intersection
s=size(V3);
x3=V3(1:s(1),1);
y3=V3(1:s(1),2);
xG=interp(x3,linspace(1,numpoints,length(x3)),1:numpoints);
yG=interp(y3,linspace(1,numpoints,length(y3)),1:numpoints);
#plot(xG,yG);
# shrink due to normal closing
```

```
newCenter = [mean(xG),mean(yG)];
xx=(xG-newCenter(1));
yy=(yG-newCenter(2));
rI = sqrt(xx^2+yy^2);
phiI = atan2(yy,xx);
rI=rI-normalClosing;
xf = rI*cos(phiI)+newCenter(1);
yf = rI*sin(phiI)+newCenter(2);
#plot(xf,yf);
finalHole = [xf,yf];
if (~drawParticle(nD)) {
depAzimMatrix = [cos(azim),-sin(azim);sin(azim),cos(azim)];
startHole = transpose(mult(depAzimMatrix,transpose(startHole)));
finalHole = transpose(mult(depAzimMatrix,transpose(finalHole)));
startHole = finalHole;
startFilmThickness=finalFilmThickness;
} else {
## make the shape used to project during deposition ##
# do the starting shape
x0=startHole(1:numpoints,1);
y0=startHole(1:numpoints,2);
plot(x0,y0);
V1=[x0,y0];
V2=[x0-filmEdgeClosingI,y0];
V3=polyand(V1,V2);
s=size(V3);
x3=V3(1:s(1),1);
y3=V3(1:s(1),2);
xx=interp(x3,linspace(1,numpoints,length(x3)),1:numpoints);
yy=interp(y3,linspace(1,numpoints,length(y3)),1:numpoints);
xyShape = [xx,yy];
#plot(xx,yy);
# do the ending shape
x0=finalHole(1:numpoints,1);
y0=finalHole(1:numpoints,2);
plot(x0,y0);
V1=[x0,y0];
V2=[x0-filmEdgeClosingF,y0];
V3=polyand(V1,V2);
s=size(V3);
x3=V3(1:s(1),1);
y3=V3(1:s(1),2);
xx=interp(x3,linspace(1,numpoints,length(x3)),1:numpoints);
yy=interp(y3,linspace(1,numpoints,length(y3)),1:numpoints);
xyShapeF = [xx,yy];
#plot(xx,yy);
#plotxy(startHole(1:41,1),startHole(1:41,2),finalHole(1:41,1),finalHole(1:41,
2));
#plotxy(xyShape(1:41,1),xyShape(1:41,2),xyShapeF(1:41,1),xyShapeF(1:41,2));
# rotate everything back to normal from the deposition rotation
depAzimMatrix = [cos(azim),-sin(azim);sin(azim),cos(azim)];
startHole = transpose(mult(depAzimMatrix,transpose(startHole)));
finalHole = transpose(mult(depAzimMatrix,transpose(finalHole)));
#plotxy(startHole(1:numpoints,1),startHole(1:numpoints,2),finalHole(1:numpoin
ts,1),finalHole(1:numpoints,2));
```

```
startHole = finalHole;
xyShape = mult(depAzimMatrix,transpose(xyShape));
xyShapeF = mult(depAzimMatrix,transpose(xyShapeF));
# put the shapes into the same oreintation
x0=xyShape(1,1:numpoints);
y0=xyShape(2,1:numpoints);
newCenter = [mean(x0),mean(y0)];
xx=(x0-newCenter(1));
yy=(y0-newCenter(2));
phiI = atan2(yy,xx); # returns in range -pi to pi
index = find(phiI,min(phiI));
x0=transpose([x0(index:numpoints);x0(1:index-1)]);
y0=transpose([y0(index:numpoints);y0(1:index-1)]);
xyShape=[x0;y0];
# put the shapes into the same oreintation
x0=xyShapeF(1,1:numpoints);
y0=xyShapeF(2,1:numpoints);
newCenter = [mean(x0),mean(y0)];
xx=(x0-newCenter(1));
yy=(y0-newCenter(2));
phiI = atan2(yy,xx); # returns in range -pi to pi
index = find(phiI,min(phiI));
x0=transpose([x0(index:numpoints);x0(1:index-1)]);
y0=transpose([y0(index:numpoints);y0(1:index-1)]);
xyShapeF=[x0;y0];
#plotxy(xyShape(1,1:numpoints),xyShape(2,1:numpoints),xyShapeF(1,1:numpoints)
,xyShapeF(2,1:numpoints));
# Give starting basis (unit) vectors (centered at origin)
# unit vectors for 100 plane
Ue = [1;0;0];
Ve = [0;1;0];
Ne = [0;0;1];
Je = [Ue, Ve];
basis_e = [Ue, Ve, Ne];
# build group for particle
addstructuregroup;
particleName = "particle"+num2str(counter);
set("name",particleName);
set("x",0);
set("y",0);
set("z",0);
adduserprop("material",5,defaultMaterial);
adduserprop("meshOrder",0,meshOrder);
adduserprop("tilt",0,depositionAngles(nD));
adduserprop("thickness",0,defaultThicknesses(nD));
adduserprop("hole d1",0,d1);
adduserprop("hole d2",0,d2);
set("script",
"selectall;
set('material',material);
set('meshOrder',meshOrder);
");
for (ii=1:length(SiAngles)) {
    ##############
    # caclulate basis (unit) vectors projection plane (111)
    # 3 points form a plane
    # These 3 points are for Si pyramid of sidewall length d
    Ap = d/2*[-(cos(SiAngles(ii))+sin(SiAngles(ii))); cos(SiAngles(ii))-
sin(SiAngles(ii)); 0];
    Bp = d/2*[-cos(SiAngles(ii))+sin(SiAngles(ii)); -
(cos(SiAngles(ii))+sin(SiAngles(ii))); 0];
    Cp = [0;0;-d/sqrt(2)];
    #Cp = [0;0;-d/2];
    Np = cross(Cp-Bp,Ap-Bp);
    Np = Np/sqrt(dot(Np,Np)); #normal unit vector (basis vector 3)
    Vp = (Ap-Bp);
    Vp = Vp/sqrt(dot(Vp,Vp)); #basis vector 2
    Up = cross(Vp,Np); #basis vector 1
    Jp = [Up, Vp];
    basis_p = [Up, Vp, Np];
    #Do the projection from old basis (e) to new basis (p) along D
    # Direction of Projection
    D = [cos(tilt)*cos(azim);cos(tilt)*sin(azim);sin(tilt)];
    # solve for new center (old center at origin)
    t = dot(Ap,Np)/dot(D,Np);
center = t*D; #this is the translation vector
# solve for the 2D scale/rotation matrix from start basis to new basis
# in *coordinates of new basis*
I3=[1,0,0;0,1,0;0,0,1];
temp = I3-mult(D,transpose(Np))/dot(D,Np);
A = mult(mult(transpose(Jp),temp),Je);
# new coordinates including only scale/rotation
xyNewShape = mult(A,xyShape);
xyNewShapeF = mult(A,xyShapeF);
newShapeAxes = [A;matrix(1,2)]; # make 3D
newShapeAxes = [newShapeAxes,Ne]; # make 3D
Rotation = mult(basis_p,newShapeAxes); # include transformation to new basis
# rotations for new basis
# euler angles (z,x',z'')
x = atan2(basis_p(3,2), basis_p(3,3));
y = atan2(-basis_p(3,1), sqrt(basis_p(3,2)*basis_p(3,2) +
basis_p(3,3)*basis_p(3,3)));
z = atan2(basis_p(2,1), basis_p(1,1));
eulerAngles = [x,y,z]*180/pi;
##############
# only make particles that are in the pyramid
if (center(3)<0) {
    # This group will control translation
    addstructuregroup;
    set("name","part" + num2str(ii));
    adduserprop("material",5,defaultMaterial);
    adduserprop("meshOrder",0,meshOrder);
    set("script",
    "selectall;
    set('material',material);
    set('meshOrder',meshOrder);
    ");
    set("x",center(1)*1e-6);
    set("y",center(2)*1e-6);
    set("z",center(3)*1e-6);
    # The next group layer controls rotation
    addstructuregroup;
    set("name","rotation group");
    adduserprop("material",5,defaultMaterial);
    adduserprop("meshOrder",0,meshOrder);
    set("script",
    "selectall;
    set('material',material);
    set('mesh order',meshOrder);
    ");
    set("x",0);
    set("y",0);
    set("z",0);
    set("first axis","z");
    set("rotation 1",eulerAngles(1));
    set("second axis","y");
    set("rotation 2",eulerAngles(2));
    set("third axis","z");
    set("rotation 3",eulerAngles(3));
    # Now come the polygons
    # Calculate the centers, vertices, and thicknesses
    D_thick = mult(inv(basis_p),D*defaultThickness); #find actual direction
of deposition
    actualStart = mult(inv(basis_p),D*currentParticleHeight(ii));
    numSlices = round((defaultThickness/resolution))+1;
    slice_X = linspace(0,D_thick(1),numSlices)+actualStart(1);
    slice_Y = linspace(0,D_thick(2),numSlices)+actualStart(2);
    slice_Z = linspace(0,D_thick(3),numSlices)+actualStart(3);
    currentParticleHeight(ii) = currentParticleHeight(ii)+defaultThickness;
    # Make the shapes
    for (jj = 1:(numSlices-1)) {
        addpoly;
        set("x",slice_X(jj)*1e-6);
        set("y",slice_Y(jj)*1e-6);
        set("z min",slice_Z(jj)*1e-6);
        set("z max",slice_Z(jj+1)*1e-6);
        V=(xyNewShape+(jj/numSlices*(xyNewShapeF-xyNewShape)))*1e-6; #convert
from um
        set("vertices",V);
    } # jj+1
    select("polygon");
    set("override mesh order from material database",1);
    set("material",defaultMaterial);
        set("mesh order",meshOrder);
        set("detail",0.2);
        addtogroup("rotation group");
        select("rotation group");
        addtogroup("part" + num2str(ii));
        select("part" + num2str(ii));
        addtogroup(particleName);
    } # if (center(3)<0)
} # ii+1
allParticleHeights(nD,1:length(SiAngles)) = currentParticleHeight;
startFilmThickness=finalFilmThickness;
} #draw particle
} # nD
redrawon;
save("tilt_"+num2str(tl)+"-thick_"+num2str(th)+"perp.fsp"); }} #sweeps
```

APPENDIX D: LUMERICAL SCRIPT TO PROCESS FDTD DIC IMAGES

```
#########################################################################
# based on phase_constrast_analysis by Lumerical Inc.
########################################################################
willSaveToMatlab = true;
willSaveRAWToMatlab = true;
willSaveZToMatlab = false;
matlabFilename = "DIC_periodic_NR_dimer";
#####################################################
# calculate the specimen beam
#####################################################
# Objective NA (in air)
NA = 1;

#clear;

maxI = matrix(1,18);
maxI_mean = matrix(1,18);
minI = matrix(1,18);
minI_mean = matrix(1,18);
bgI = matrix(1,18);
numSimulations = 19;
simulationAngles = linspace(0,180,numSimulations);
load("particle_angle_1DIC_1");
f=getdata("Monitor2","f");
res = 201; # resolutions is res x res
Zres = 1;
ImageMatrix = matrix(res,res,length(f),numSimulations);
RawDataMatrix = matrix(res,res,3,2,length(f),numSimulations);
#ZProjectMatrix = matrix(res,res,Zres,length(f),numSimulations);
for (ii=1:numSimulations-1) { #angle in degrees
    ?"Analyzing simulation "+num2str(ii);
    # choose the filename of the template file
    filename = "particle_angle_"+num2str(ii)+"DIC";
    # the name of the monitor recording the data
    mname = "Monitor2";
    f=getdata(mname,"f");
    lambda = c/f;
    for (ff = 1:length(f)) { #multiple frequency points
        # choose the resolution for far field projections
        farfieldfilter(0);
#if (1) {

        load(filename+"_1");
        Ev = gratingvector(mname,ff);
        Ex_near_1 = pinch(Ev,3,1);
        Ey_near_1 = pinch(Ev,3,2);
        Ez_near_1 = pinch(Ev,3,3);
        load(filename+"_2");
        Ev = gratingvector(mname,ff);
        Ex_near_2 = pinch(Ev,3,1);
        Ey_near_2 = pinch(Ev,3,2);
        Ez_near_2 = pinch(Ev,3,3);
        ux = gratingu1(mname,ff);
        uy = gratingu2(mname,ff);
        Ux = meshgridx(ux,uy);
        Uy = meshgridy(ux,uy);
        Uxy = sqrt(Ux^2+Uy^2)+1e-20; # add 1e-20 to avoid divide by zero
problems
        Uz  = sqrt(1-Uxy^2);
        ## filter for propagating waves through aperture
        filter = real(Uxy) < NA;
        #####################################################
        # calculate field at image plane
        #####################################################
        k=2*pi/lambda(ff);
        ## define image plane (magnify object X2)
        npts=res;
        simulation;
        xmin=getnamed("FDTD","x min");
        xmax=getnamed("FDTD","x max");
        ymin=getnamed("FDTD","y min");
        ymax=getnamed("FDTD","y max");
        x=linspace(xmin*2,xmax*2,npts);
        y=linspace(ymin*2,ymax*2,npts);
        x=linspace(-3e-6,3e-6,npts);
        y=linspace(-3e-6,3e-6,npts);
        # calculate the image, uzing chirped z-transform
        kx = ux*k;
        ky = uy*k;
z_range = -50e-9;
for (zz=1:Zres) {
    z_project = z_range(zz);
    z_phase = 1i*exp(1i*k*Uz*filter*(z_project)-1i*k*1);
    Ex_1 = czt(filter*Ex_near_1*z_phase,kx,ky,x,y);
    Ey_1 = czt(filter*Ey_near_1*z_phase,kx,ky,x,y);
    Ez_1 = czt(filter*Ez_near_1*z_phase,kx,ky,x,y);
    #image(x*1e6,y*1e6,imag(Ex_0));
    RawDataMatrix(1:res,1:res,1,1,ff,ii)=Ex_1;
    RawDataMatrix(1:res,1:res,2,1,ff,ii)=Ey_1;
    RawDataMatrix(1:res,1:res,3,1,ff,ii)=Ez_1;
    Ex_2 = czt(filter*Ex_near_2*z_phase,kx,ky,x,y);
    Ey_2 = czt(filter*Ey_near_2*z_phase,kx,ky,x,y);
    Ez_2 = czt(filter*Ez_near_2*z_phase,kx,ky,x,y);
    #image(x*1e6,y*1e6,imag(Ey_90));
    RawDataMatrix(1:res,1:res,1,2,ff,ii)=Ex_2;
    RawDataMatrix(1:res,1:res,2,2,ff,ii)=Ey_2;
    RawDataMatrix(1:res,1:res,3,2,ff,ii)=Ez_2;
    #} #if
    ## combine the beams (with source phase) and apply the analyzer via jones
matrix
    ## set up the polarization and phase of optics
    # Nomarskis are fixed at 45 degrees.
    # Incident light aligned to 0, exits at 90
    for (testangle=8) {
        # de Senarmont angle for phase shift (0=linear, 45 = RHCP, -45 =
LHCP)
        source_angle = testangle;
        # polarization angle on analyzer
}#z

analyzer_angle = 45;
analyzer_radians = analyzer_angle*pi/180;
source_radians = source_angle*pi/180;
# recombine the beams into some elliptical polarization state
Ex_final = Ex_1+Ex_2*exp(-1i*2*source_radians);
Ey_final = Ey_1+Ey_2*exp(-1i*2*source_radians);
#Ez = Ez_1 + Ez_2*exp(-1i*2*source_radians);
## Apply the polarizer
Ex_image = Ex_final * cos(analyzer_radians)^2
Ey_image = Ex_final * sin(analyzer_radians)*cos(analyzer_radians) + Ey_final
* sin(analyzer_ra
#Ez_image = 0;
## calculate |E|^2 at the image plane
# adding up the intensities for all components
E_image_total = abs(Ex_image)^2;
#ZProjectMatrix(1:res,1:res,zz,ff,ii)=E_image_total;
#E2_ref_image = E2_ref_image + abs(Ex_ref_image)^2 + abs(Ey_ref_image)^2 +
abs(Ez_ref_image)^2
#E2_scaled = E2_scaled + abs(Ex_scaled)^2 + abs(Ey_scaled)^2 +
abs(Ez_scaled)^2;
#  E2_pc_image(1:npts,1:npts,j) = pinch(E2_pc_image(1:npts,1:npts,j))
    #
    #
    #
    #E2_int_image=abs(Ex_total)^2 + abs(Ey_total)^2 + abs(Ez_total)^2;
    ## plot field at image plane with no re-interference, ie bright field
result
    #image(x*1e6,y*1e6,E_image_total,"x (microns)","y
(microns)",num2str(simulationAngles(ii))+"-d
    #setplot("grey scale",1);
    #setplot("colorbar min",0.006);
    #setplot("colorbar max",0.014);
    #exportfigure(num2str(ii)+"deg");
}#test angle
xx = meshgridx(x,y)*1e6;
yy = meshgridy(x,y)*1e6;
distances = sqrt((xx-0.0)^2+yy^2);
distance_mask = distances>1e-6;
#background = sum(E_image_total*distance_mask)/sum(distance_mask);
background = mean(E_image_total);
background_std = std(E_image_total);
bright_filter = E_image_total>(background+2*background_std);
dark_filter = E_image_total<(background-2*background_std);
#?contrast_Michelson = (max(E_image_total)-
min(E_image_total))/(max(E_image_total)+min(E_image_tot
#?contrast_absolute = (max(E_image_total)-min(E_image_total));
#((max(E_image_total)-background)+(background-
min(E_image_total)))/((max(E_image_total)-background
#ii=ii*10;
#maxI(ii/10+1)=max(E_image_total);
#maxI_mean(ii/10+1) = sum(E_image_total*bright_filter)/sum(bright_filter);
#minI(ii/10+1)=min(E_image_total);
#minI_mean(ii/10+1) = sum(E_image_total*dark_filter)/sum(dark_filter);
#bgI(ii/10+1)=background;
  + abs(Ex_image+exp(1i*phase_delay(j))*Ex_ref_image)^2
 + abs(Ey_image+exp(1i*phase_delay(j))*Ey_ref_image)^2
+ abs(Ez_image+exp(1i*phase_delay(j))*Ez_ref_image)^2;
+ Ey_final * cos(analyzer_ra
        ImageMatrix(1:res,1:res,ff,ii) = E_image_total;
    } # ff
} # ii
#image(x*1e6,y*1e6,abs(Ex),"x (microns)","y (microns)","angle x");
#image(x*1e6,y*1e6,abs(Ey),"x (microns)","y (microns)","angle y");
#image(x*1e6,y*1e6,angle(Ex_90),"x (microns)","y (microns)","angle x-90");
#image(x*1e6,y*1e6,angle(Ex_90),"x (microns)","y (microns)","angle y-90");
# reload the template file
#plot(0:10:350,[maxI,maxI],[minI,minI],[bgI,bgI]);
#exportfigure("90deg_oop_sweep");
#zStack = matrixdataset("Z_stack");
#zStack.addparameter("x_um",x*1e6);
#zStack.addparameter("y_um",y*1e6);
#zStack.addparameter("z_um",z_range*1e6);
#zStack.addparameter("f",f,"lambda_nm",lambda*1e9);
#zStack.addparameter("particleAngle",simulationAngles);
#zStack.addattribute("DIC_image",ZProjectMatrix);
#visualize(zStack);
images = matrixdataset("DIC_images");
images.addparameter("x_um",x*1e6);
images.addparameter("y_um",y*1e6);
images.addparameter("f",f,"lambda_nm",lambda*1e9);
images.addparameter("particleAngle",simulationAngles);
images.addattribute("DIC_image",ImageMatrix);
#visualize(images);
matlabsave("../"+matlabFilename,images);
matlabsave("../"+matlabFilename+"RAW",RawDataMatrix,x,y,lambda,simulationAngl
es);
#matlabsave("../"+matlabFilename+"Z",ZProjectMatrix,x,y,z_range,lambda,simula
tionAngles);
```

CURRICULUM VITAE

Department of Materials Science and Engineering
2220 Campus Drive, Cook Hall 2036
Evanston, IL 60208
Office: 847-491-7553
ahryn@u.northwestern.edu

EDUCATION

Northwestern University, Evanston, IL **December 2016**

- PhD in Materials Science and Engineering
- GPA: 3.8/4.0
- Advisor: Teri W. Odom

University of Illinois, Urbana, IL **May 2010**

- BS in Materials Science and Engineering with Highest Honors
- GPA: 3.8/4.0
- Coursework included: concentration in electronic materials, minor in physics

RESEARCH EXPERIENCE

Graduate Research Assistant, *Northwestern University*: Teri W. Odom September 2010 – Present

- Generation of nanopatterned surfaces by photo- and soft- lithography
- Simulation of plasmonic structures in FDTD software
- Fabrication and measurement of Au and Ag plasmonic nanoparticle arrays
- Development of data acquisition and analysis software
- Identification of methods to describe optical properties of Moiré nanopatterns

Production Engineering Intern, *Microlink Devices*: Chris Youtsey June – August 2014

- Development of procedures to achieve PEC etching of GaN
- Achievement of uniform etch rates on cm-scale areas
- Coordination with industry and academia for processing and analysis

Short Term Appointment, *Argonne National Laboratory*: Alex B. F. Martinson May – August 2010

- Creation and testing of dye sensitized solar cells using new materials
- Building of Edwards A306 evaporator and new Savannah S200 ALD systems

NNIN REU Intern, *University of Michigan – Ann Arbor*: A. John Hart June – August 2009

- Growth and analysis of large-area carbon nanotube structures
- Coating of carbon nanotubes with oxide coatings by ALD

Research Intern, *Kingston Process Metallurgy*: Boyd R. Davis May – August 2008

- Research of new high-κ dielectric material based on barium titanate
- Design of experiments to synthesize barium titanate films

SRP REU Intern, *Argonne National Laboratory*: Jeffrey W. Elam June – August 2007

- Coating of nanoparticle networks with various TCO films by ALD for OLEDs
- Characterization of oxide films via spectroscopic ellipsometry

PCRP REU Intern, *Argonne National Laboratory*: Jeffrey W. Elam June – August 2006

- Deposition of TCO thin films by ALD for photovoltaics
- Identification of thin film composition with XPS

TEACHING EXPERIENCE

Graduate Teaching Assistant, *Northwestern University* September – December 2014

- Graduate course: Physics of Nanostructures
- Designed and presented a class lecture
- Led weekly discussion sections

Graduate Teaching Assistant, *Northwestern University* April – June 2012

- Undergraduate 4th-year course: Physics of Materials
- Led weekly discussion and laboratory sessions

Undergraduate Teaching Assistant, *University of Illinois* January – May 2010

- Undergraduate 3rd-year course: Electronic Properties of Materials
- Led weekly discussion sections

PROFICIENCIES AND SKILLS

Micro/Nanofabrication: Clean room operation, photolithography, soft lithography, interference lithography, chemical etching, reactive ion etching, atomic layer deposition, physical vapor deposition
Materials analysis: Optical microscopy, scanning electron microscopy, spectroscopic ellipsometry, UV-vis-NIR spectroscopy
Software: MATLAB, Lumerical FDTD Solutions, LabView, ImageJ, Blender, Adobe Illustrator, Adobe Photoshop, MS Office
Programming: MATLAB (scripting, GUI design, OOP), UNIX/ bash scripting, Lumerical scripting, Visual C#

PUBLICATIONS AND PRESENTATIONS

- **Hryn, A. J.**; Culver, K. S. B.; Odom, T. W., DIC image contrast dependence on gold nanoparticle resonance wavelength. (*In preparation*).
- **Hryn, A. J.**; Odom, T. W., Low-symmetry plasmonic crystals. (*In preparation*).
- **Hryn, A. J.**; Lubin, S. M.; Odom, T. W., Surface Plasmon Polaritons in Multiscale Gratings. (*In preparation*).
- Yang, A.; **Hryn, A. J.**; Bourgeois, M. R.; Lee, W.-K.; Hu, J.; Schatz, G. C.; Odom, T. W. Programmable and Reversible Plasmon Mode Engineering. *Proceedings of the National Academy of Sciences,* Online (2016).
- Yang, A.; Li, Z.; Knudson, M. P.; **Hryn, A. J.**; Wang, W.; Aydin, K.; Odom, T. W. Unidirectional Lasing From Template-Stripped Two-Dimensional Plasmonic Crystals. *ACS Nano* **9**, 11582–11588 (2015).
- Wang, D.; Yang, A.; **Hryn, A. J.**; Schatz, G. C.; Odom, T. W. Superlattice Plasmons in Hierarchical Au Nanoparticle Arrays. *ACS Photonics* **2**, 1789–1794 (2015)

- **Hryn, A. J.**; Lubin, S. M.; Odom, T. W., Plasmonic Crystals with High and Low Symmetry. *Physical Electronics Conference*, Rutgers University, New Brunswick, NJ. June 16 – 19, 2015 (*Poster presentation*)
- Lubin, S. M.; **Hryn, A. J.;** Huntington, M . D.; Engel, C. J.; Odom, T. W. Quasiperiodic Moiré plasmonic crystals. *ACS Nano* **7**, 11035-11042 (2013).
- **Hryn, A. J.**, Plasmonic Quasicrystal Lattices. *American Vacuum Society 60th Symposium*, Long Beach, CA. October 28 – 31, 2013 (*Oral Presentation*)
- Huntington, M. D; **Hryn, A. J.**; Divan, R.; Odom, T. W., Nanomanufacturing for Plasmonic and Photonic Crystals. *Center for Nanoscale Materials Users Meeting*, Argonne National Laboratory, Argonne, IL. May 6 – 8, 2013 (*Poster Presentation)*
- Huntington, M. D.; Engel, C. J.; **Hryn, A. J.**; Odom, T. W. Polymer nanowrinkles with continuously tunable wavelengths. *ACS Applied Materials and Interfaces* **5**, 6438-6442 (2013).
- Lubin, S. M.; Zhou, W.; **Hryn, A. J.;** Huntington, M. D.; Odom, T. W. High-rotational symmetry lattices fabricated by Moiré nanolithography. *Nano Letters* **12**, 4948-4952 (2012).
- Narayan, R. J.; Adiga, S. P.; Pellin, M. J.; Curtiss, L. A.; **Hryn, A. J.**; Stafslien, S.; Chisholm, B.; Shih, C.-C.; Shih, C.-M.; Lin, S.-J.; Su, Y.-Y.; Jin, C.; Zhang, J.; Monteiro-Riviere, N. A.; Elam, J. W., Atomic layer deposition-based functionalization of materials for medical and environmental health applications. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* **368**, 2033-2064 (2010).
- Elam, J. W.; Baker, D. A.; **Hryn, A. J.**; Martinson, A. B. F.; Pellin, M. J.; Hupp, J. T., Atomic layer deposition of tin oxide films using tetrakis(dimethylamino) tin. *J Vac Sci Technol A* **26**, 244-252 (2008).

AWARDS

CNM Users Meeting Best Student Poster 2013
Argonne National Laboratory, Argonne, IL

National Defense Science and Engineering Graduate (NDSEG) Fellowship 2012 – 2015
Northwestern University, Evanston, IL

Hierarchical Materials Cluster Program 2012
Northwestern University, Evanston, IL

NSF Graduate Research Fellowship Honorable Mention 2011
Northwestern University, Evanston, IL

Larry D. and Carol Rakers Scholarship 2009
University of Illinois, Urbana, IL

Lucille and Charles Wert Scholarship 2008
University of Illinois, Urbana, IL

Kaiser-Materials Science Engineering Scholarship 2007
University, of Illinois, Urbana, IL

Ivan Racheff Scholarship 2006
University of Illinois, Urbana, IL