STRUCTURE AND ELECTRONIC PROPERTIES OF LEAD-SELENIDE NANOCRYSTAL SOLIDS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Kevin Whitham

August 2016

STRUCTURE AND ELECTRONIC PROPERTIES OF LEAD-SELENIDE NANOCRYSTAL SOLIDS

Kevin Whitham, Ph.D.

Cornell University 2016

Recent advances in the controlled formation of nanocrystal superlattices have potential for creating materials with properties by design. The ability to tune nanocrystal size, shape and composition as well as symmetry of the superlattice opens routes to new materials. Calculations of such materials predict interesting electronic phenomena including topological states and Dirac cones, however experimental support is lacking. We have investigated electron localization in nanocrystal superlattices using a combination of advanced structural characterization techniques and charge transport measurements.

Recent experimental efforts to improve the electronic properties of nanocrystal solids have focused on increasing inter-dot coupling. However, this approach only leads to electronic bands if the coupling energy can overcome energetic and translational disorder. We have investigated oriented-attachment as a method to create nanocrystal superlattices with increased coupling and translational order. We show that epitaxially connected superlattices form by a coherent phase transformation that is sensitive to structural defects and ligand length. In order to measure intrinsic electronic properties we demonstrate control over electronic defects by tailoring surface chemistry and device architecture.

To probe charge transport in these structures we performed variable temperature field-effect measurements. By integrating structure analysis, surface chemistry, and transport measurements we find that carriers are localized to a few superlattice constants due to disorder. Importantly, our analysis shows that greater delocalization is possible by optimizing dot-to-dot bonding, thus providing a path forward to create quantum dot solids in which theoretically predicted properties can be realized.

BIOGRAPHICAL SKETCH

Kevin Whitham graduated from Rensselaer Polytechnic Institute with a B.S. in Electrical and Computer Engineering in 2006. In 2007 he graduated from Cornell University with an M.Eng. in Electrical Engineering after researching photonic crystals under the guidance of Prof. Michal Lipson. He joined the group of Prof. Tobias Hanrath in 2009.

Dedication

To my mother, for showing me that inside of a book is a great place to be.

To my father, who told me "work with your head, not with your hands."

To my older sister, for always leading the way, especially to Cornell.

To my younger sister, for being the toughest and the funniest.

To Meagan, for getting both of us through grad school.

ACKNOWLEDGEMENTS

I credit Prof. Melissa Hines for inspiring in me a an interest in fundamental science while I was an undergraduate at a very industrially oriented engineering college. Her seminar demonstrating nanotechnology research planted a seed which grew into a master's degree and now a doctorate. I want to thank the several faculty at Cornell who have encouraged and inspired me, starting with Prof. Hines. When I was an M.Eng. student, Prof. Michal Lipson renewed my interest in science by giving me the freedom to explore. I learned to let curiosity be my motivation and to ask questions for which there were no easy answers. Although I had some regrets after declining an offer to pursue a Ph.D. with Prof. Lipson, I eventually found my way back to science (and Cornell) thanks to Prof. Tobias Hanrath. His support over the years has been invaluable. His role has not simply been as an enabler and advisor, but more importantly as a collaborator, fostering a real sense of team; team Helios, and proud we are of all of them.

I owe much to the other members of team Helios, firstly to Prof. Josh Choi for bringing infectious energy and enthusiasm to everything he does. Josh, along with Dr. Will Baumgardner and Dr. Kaifu Bian welcomed me to the team. They made working in the lab a pleasure and group meetings enjoyable and intellectually invigorating. I thank Will in particular for always following his interests, I like to think I've done the same in my work. I thank Kaifu for keeping all of us on our toes by always recognizing something for what it is and calling it so, which is not difficult when you're as smart and wise as Kaifu. I'd like to thank Dr. Yee-Fun Lim for being our sage guide in the ways of fabricating and testing solar cells. I appreciated his friendliness and professionalism as a young grad student. I want to thank Dr. Jacek Jasieniak who, for a short time energized our labs and shared his knowledge with us. He is truly a professional scientist, although not a very fast eater of pumpkin pie. I thank Dr. David Moore for all his efforts to accelerate research in our labs with various gizmos to make our tasks a little easier. I'd like to thank Ben Richards for working at all hours and for an endless stream of Lebowski quotes and other daily comic relief. I thank Dr.

Ben Treml for carrying the nanocrystal research torch to creative places and for reminding me there are opportunities everywhere if you keep an open mind. To the younger members of the group: Victor Lambert, Jessica Akemi Cimada da Silva, Kevin Kimura, Eliad Peretz, Ali Tirmzi, Doug Nevers, Curtis Williamson, and all the undergraduate researchers over the years, thank you for bringing fresh insight, enthusiasm, suggestions and ideas.

I want to thank the staff at the Cornell Center for Materials Research, the Cornell High Energy Synchrotron Source, and the Cornell Nanofabrication Facility. Special thanks goes to John Grazul for always being supportive, when I need microscopy help I pray to John G. I also owe a special thank you to Dr. Detlef Smilgies, beamline scientist at CHESS. All of his work on D1 makes our lives as grad students easy.

I want to thank my collaborators, without whom this work would not have been completed. In particular I'd like to thank Prof. Frank Wise for his level-headed comments and guidance, and Prof. Lena F. Kourkoutis for raising the quality of our work through her openness and constructive comments. I want to thank Ben Savitzky for cutting his teeth on our project and coming through with incredible microscopy images from the NION, a very impressive feat for a young grad student. Special thanks to Jun Yang for his work measuring optical properties, theoretical calculations, and ability to understand almost anything.

Finally I owe thanks to my family and friends for supporting me throughout a very long Ph.D. Thank you to my parents and sisters for never questioning my choice to stay in school well past a reasonable age. Your interest and encouragement kept me going even when I wasn't sure that I could. Most of all I want to thank Meagan Hinze for showing me what hard work and dedication truly look like. Her drive and work ethic in the face of difficulty inspired me to accomplish much of what is contained in this dissertation. Thanks for all the late night snacks, dinners, and recuperating trips to see waterfalls. You know what I need better than I do, and I hope I can do the same for you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

1D one-dimensional.

2D two-dimensional.

3D three-dimensional.

ACN acetonitrile.

ALD atomic layer deposition.

BCC body centered cubic.

$C_3H_8O_2$ 2-methoxyethanol.

DFT density functional theory.

EDA ethylenediamine.

EG ethylene glycol.

FCC face centered cubic.

FET field-effect transistor.

FFT fast Fourier transform.

FWHM full width at half maximum.

GISAXS grazing incidence small angle X-ray scattering.

HCP hexagonal close-packed.

HRTEM high resolution transmission electron microscopy.

LED light emitting diode.

MeOH methanol.

MOSFET metal oxide semiconductor field effect transistor.

NC nanocrystal.

NH_4Cl ammonium chloride.

NN nearest neighbor.

NNH nearest neighbor hopping.

Pb-CH_3COOH^- lead acetate.

PETM pentaerythritol-tetrakis(3-mercaptopropionate).

PMF potential of mean force.

PV photovoltaic.

SAXS small-angle X-ray scattering.

SC simple cubic.

STEM scanning transmission electron microscopy.

TATATO 1,3,5-triallyl-1,3,5-triazine-2,4,6(1H,3H,5H)-trione.

TBAI tetrabutylammonium iodide.

TEM transmission electron microscopy.

TOPSe trioctylphosphine selenide.

UV ultraviolet.

VRH variable range hopping.

CHAPTER 1

INTRODUCTION

1.1 Materials for Energy

The performance of energy conversion devices such as photovoltaics (PVs) and thermoelectrics is limited by material properties. Certain applications demand specific properties be optimized to maximize performance. We often wish to change certain properties of a material, for example the bandgap, to meet the requirements of a particular application without affecting other properties such as electrical conductivity. However, many properties are coupled because of their dependence on composition, crystal structure, or defects. For example, the bandgap of a semiconductor is an important property for PV devices. The bandgap defines the minimum energy photon that can be absorbed to generate an electron-hole pair. Therefore it is a property that must be tuned with respect to the incident spectral power distribution to maximize efficiency. However there is a trade-off between the number of photons absorbed and the voltage output. A larger bandgap produces a larger voltage output at the expense of smaller absorption and therefore smaller current. The ideal material would provide the opportunity to decouple output voltage and current.

These types of trade-offs are common hurdles in engineering. In the case of PV devices, the maximum power conversion efficiency of a device with a single band gap is 30%[1]. A bandgap of 1.1 eV is required to reach this efficiency. Silicon has a bandgap of 1.1 eV. Combined with the scale of the silicon industry, this has led to the dominance of silicon in PV devices. A similar trade-off exists for thermoelectric materials. Electrical conductivity should be maximized and thermal conductivity minimized, yet both increase (or decrease) as crystalinity increases (or decreases).

In the case of PV, few materials posses optimal values for bandgap, absorptivity, stability,

and conductivity. For example silicon has an ideal bandgap, but the absorptivity is small which demands a thicker layer and therefore a low defect density to maintain a sufficient carrier diffusion length. The combination of multiple materials into multijunction cells can overcome these limitations, but challenges such as lattice matching greatly increases cost. Similar challenges are encountered in the search for thermoelectric materials with high electrical conductivity and low thermal conductivity. However these properties are difficult to change independent of each other in bulk crystals.

Nanoscale engineering could overcome the intrinsic limits of bulk crystals to design materials with desired bandgap, electrical and thermal conductivity, or absorptivity. The quantum dot is one example of a nanoscale system with many degrees of freedom available for tuning material properties.

1.2 Semiconductor Nanocrystals

The term quantum dot can refer to any zero-dimensional object and implies that the properties of the material differ significantly from the bulk[2]. Nanocrystals (NCs) in particular are inorganic crystals with anywhere from tens to several thousands of atoms. Semiconductor NCs, as opposed to metal or oxide NCs, are particularly interesting for electronic and optoelectronic applications. Semiconductor NCs can be synthesized by precipitation in glasses or by strain induced phase change in epitaxially grown thin films[3, 4]. Alternatively, NCs synthesized as colloids in solution can be produced with very good quality[5, 6, 7]. Metrics of high quality include small size dispersion ($< 5\%$), high fluorescence quantum yield ($> 80\%$), and long colloidal stability ($>$ months).

Reducing crystal size to the nanoscale introduces quantum effects that can allow tuning of key properties. For example, the quantum confinement effect refers to a change in the

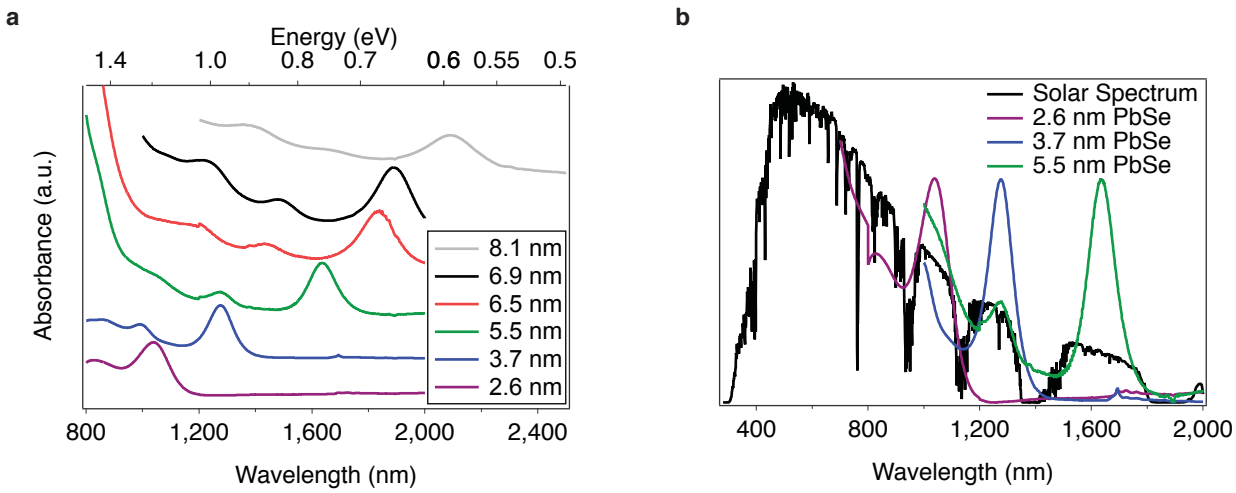

Figure 1.1: Illustration of the quantum size effect on absorption spectra of PbSe NCs. **a**, Absorption spectra of a range of PbSe NC diameters showing the quantum size effect on bandgap energy. Spectra are offset vertically for clarity. **b**, The AM1.5 terrestrial solar energy spectrum with the absorption spectra of three sizes of PbSe NCs.

bandgap of a material as the crystal size becomes less than the Bohr radius. This effect is especially strong in the lead chalcogenides (PbS, PbSe, PbTe) which have large electron and hole Bohr radii of tens of nanometers[8]. The bandgap of lead chalcogenide NCs can be tuned throughout the near infra-red[9, 10] as shown in figure 1.1. This tunability combined with facile and scalable solution processing has led to intense research on lead chalcogenide NC PV devices with power conversion efficiency currently up to 10.7%[11].

In addition to remarkably tunable properties, the colloidal nature of lead chalcogenide and other NCs makes large scale, low cost fabrication a real possibility. Already semiconductor NC products have been developed by major manufacturers for enhanced light management in displays[12]. Scalability is facilitated by moderate synthesis temperatures (100 °C to 400 °C) and industrially available precursors.

Colloidal NCs are modular or building-block like, even gaining the nickname 'artificial atoms'[6, 2, 7, 13]. The extremely wide variety of NC materials, size, shape, dopants, and surface functionalization have led to diverse applications. For PV and light emitting diode

(LED) devices the current challenge is to translate the unique properties of colloidal NCs into thin films.

1.2.1 Quantum Size Effect

The most interesting property of semiconductor NCs is the ability to tune energy levels by changing NC size. This is called the 'quantum size effect' and can be understood with elementary quantum mechanics. When an electron is confined within a volume of material with dimensions on the order of the electron wavelength in the bulk material, the wavelength must decrease and therefore the energy of the electron must increase. This qualitative reasoning finds its quantitative explanation in the quantum mechanical problem of the infinite potential well. An analogous model for the NC in one-dimension is the infinite square well with a potential function given by equation (1.1). The energy levels of a particle with mass m can be found by solving the time-independent Schrödinger equation (1.2).

$$V(x) = \begin{cases} 0 & 0 \leq x \leq a \\ \infty & \text{otherwise} \end{cases} \tag{1.1}$$

$$\begin{aligned} -\frac{\hbar^2}{2m}\frac{d^2\psi}{dx^2} &= E\psi \\ \frac{d^2\psi}{dx^2} &= -k^2\psi \end{aligned} \tag{1.2}$$

The solution for the wave function $\psi(x)$ is the simple harmonic oscillator function $\psi(x) = A\sin(kx) + B\cos(kx)$. Since the wave function must be zero outside of the well and therefore zero at the boundaries, $B = 0$ and k can only take on values which give $\psi(a) = A\sin(ka) = 0$. This leads to discrete values of $k_n = n\pi/a$ with $n = 1, 2, 3, ...$ which depend on a and therefore

the energy eigenvalues $E = \hbar^2 k^2/2m$ depend on a. Equation (1.3) captures the quantum size effect: the energy of a particle confined to a box increases as the box gets smaller.

$$E = \frac{\hbar^2 n^2 \pi^2}{2ma^2} \tag{1.3}$$

The picture for NCs is more complicated. NCs are three-dimensional (3D) with faceted shapes, the electron mass is not the free electron mass, but an effective mass that depends on wave vector, etc. The exact electronic structure of a NC of arbitrary material is therefore not trivial to calculate. However, much effort has been invested to understand the exact relationship between size and electronic structure for several interesting materials, especially the II-VI and IV-VI materials namely CdS, CdSe, PbS, and PbSe.

For PbSe in particular, Kang and Wise calculated the exciton energies and dipole transition strengths to compare with measured absorption spectra[14]. Controversy over certain features of the PbSe NC spectrum remain[15, 16, 17], however experiments have produced a reliable empirical relation between diameter and exciton absorption energy by correlating spectra with transmission electron microscopy (TEM) images[9]. Several such spectra are shown in figure 1.1. As an example of the utility of the quantum size effect, we show PbSe NC absorption spectra with the solar energy spectrum (AM1.5 terrestrial normal incidence at the equator) in figure 1.1. Simply by changing NC size different portions of the solar spectrum can be utilized in NC PV devices.

1.2.2 Synthesis of PbSe Nanocrystals

The synthesis and study of low-dimensional nano-size materials began in earnest during the 1980s. In 1982 Hayashi *et al* reported germanium crystals as small as 8 nm made by thermal evaporation[18]. Ekimov and Efros reported the forerunner to today's NCs when they pre-

cipitated CdS, CdSe, CuCl, and CuBr crystals as small as 3 nm within a silicate matrix[19]. The field of nanomaterials took off when in 1985 Smalley and co-workers discovered C_{60}[20].

Materials with large Bohr radii are most interesting as the confinement effect is most pronounced. The II-VI and IV-VI metal chalcogenides are well studied for this reason. With an exciton Bohr radius of 40 nm, PbSe is especially suited for studying the quantum size effect[8]. An early synthesis produced a distribution of PbSe NCs from 2 nm to 15 nm by precipitation in a glass host[21]. Access to PbSe NC materials increased after the development of a colloidal synthesis[22]. The colloidal route improved size dispersion, yield, and allowed the facile manipulation of NCs for characterization. The colloidal synthesis of PbSe has improved steadily over the past 15 years, with current methods capable of $< 3\%$ size dispersion[23, 24].

Colloidal synthesis of NCs depends on the coordination of the inorganic NC surface with ligand molecules. This interaction is key to achieving colloidal stability, small size dispersion, and shape control. Synthesis involves the preparation of organo-metallic precursors and their reaction in an organic solvent. Injecting one precursor into another at a temperature high enough to facilitate growth, a burst of nucleation occurs generating many NC nuclei in a short time. This ensures small size distribution. The creation of the NC phase in the growth medium raises the free energy due to the surface energy of the particles. Once particles reach a critical radius r_c, the free energy decreases due to the volumetric energy contribution of the NC phase. This method of NC synthesis is referred to as the hot-injection method or the LaMer model[25]. Figure 1.2 illustrates the colloidal synthesis of PbSe NCs by the hot-injection method. Details are given in appendix A.1, but briefly a solution of trioctylphosphine selenide (TOPSe) is injected into hot Pb-oleate. After a burst of nucleation indicated by a rapid color change, the NCs are allowed to grow to a certain size before the reaction is quenched, arresting further growth.

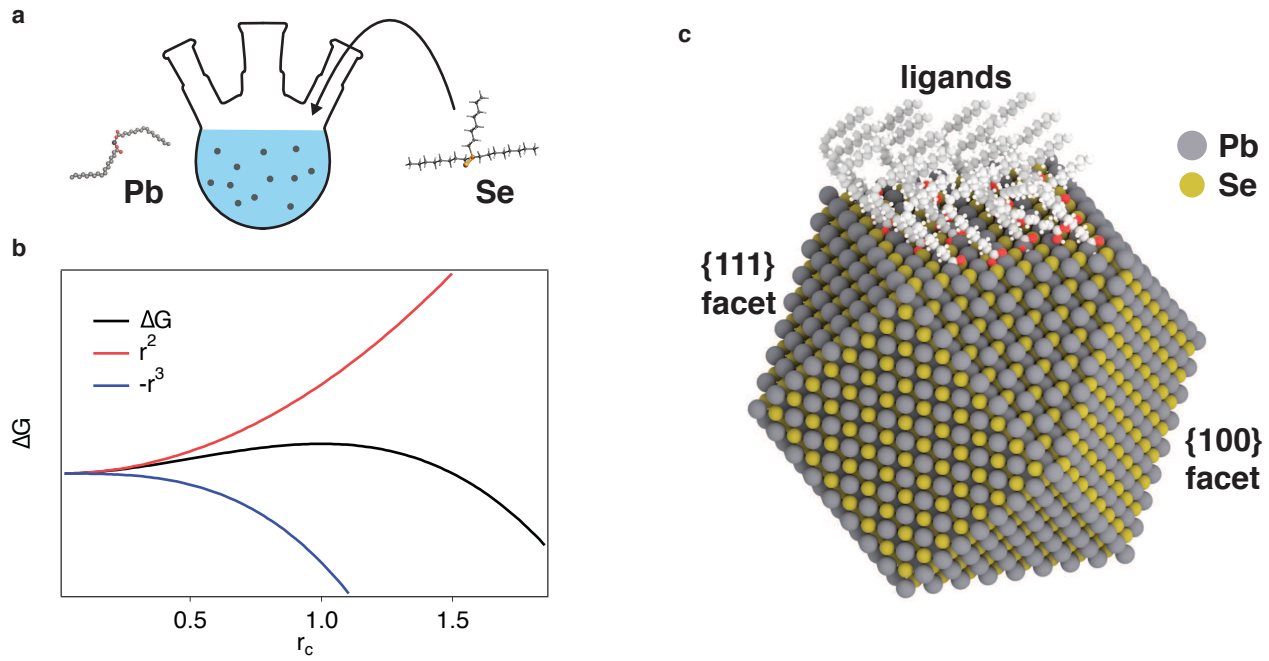

Figure 1.2: Synthesis of PbSe NCs. **a**, Illustration of the hot-injection method. Percursor molecules Pb-oleate and TOPSe are shown. **b**, Driving force for growth is the reduction in free energy for NCs larger than the critical radius r_c. **c**, Illustration of a truncated cubic PbSe NC. Ligands attach to all facets but are drawn only on one facet for clarity.

1.2.3 Nanocrystal Shape Control

The shape of a NC can influence assembly symmetry, electronic, optic, and catalytic properties. Shape is determined by facet surface energies through both thermodynamic and kinetic factors. The equilibrium shape can be predicted if the surface energies of the material are known. The energies of the three lowest index facets of PbSe have been calculated by Fan *et al* and are shown in table 1.1. These are the energies of bare surfaces, without ligands. Based on these values, the shape can be determined using the Wulff construction. The Wulff construction is simply the application of the principle that the surface of the equilibrium shape will be dominated by lower surface energy facets. The Wulff construction is the minimum volume convex hull created by planes parallel to crystal facets displaced from the origin a distance proportional to the the surface energy of said facet[26]. As an example, figure 1.3 shows that the equilibrium shape of a PbSe crystal is a cube.

Facet	γ_{surface} J/m^2
{100}	0.184
{110}	0.318
{111}	0.328

Table 1.1: Surface energies of low-index facets of PbSe.

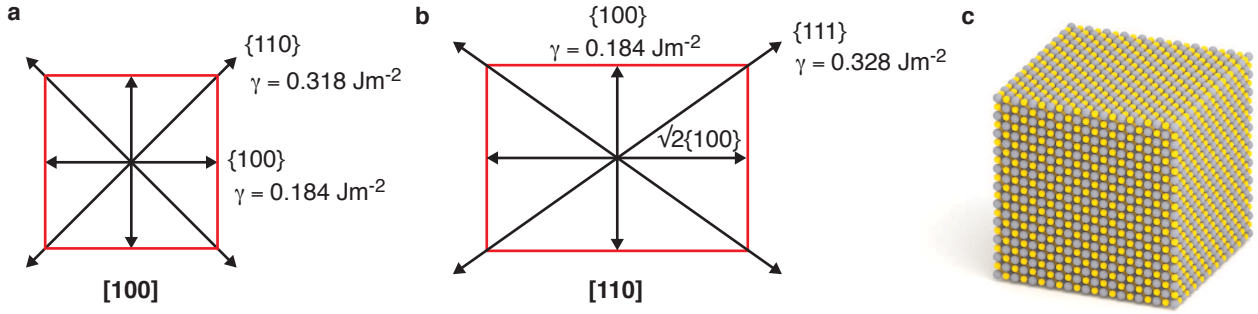

Figure 1.3: Wulff construction for PbSe. **a**, Wulff construction of the $< 100 >$ projection. Because the surface energy of the {110} facet is $> \sqrt{2}$ the {100} facet energy, there are no {110} facets. **b**, Wulff construction of the $< 110 >$ projection. The {111} facet energy is $> \sqrt{3}$ the {100} facet energy, therefore there are no {111} facets. **c**, The equilibrium shape is a cube.

Real NCs however do not have bare facets, but are coordinated with ligands. Ligand binding decreases surface energy, therefore surface energy is a function of ligand coverage. Bealing *et al* calculated equilibrium ligand densities on the {100} and {111} facets of PbSe NCs using density functional theory (DFT)[27]. By changing the free ligand concentration or using multiple ligand species during synthesis, it is possible to change relative ligand coverage on specific facets to control NC shape[28, 29]. Figure 1.4 shows PbSe NCs of several sizes and shapes.

The preceding sections have shown that semiconductor NCs are hightly versatile and tunable building blocks. Control over size, shape, core material, and ligand composition provide means to synthesize nano materials with desirable properties. The complexity and versatility of nano materials can be extended by assembly of these building blocks into superlattices, analogous to creating atomic crystals from combinations of atoms. The following

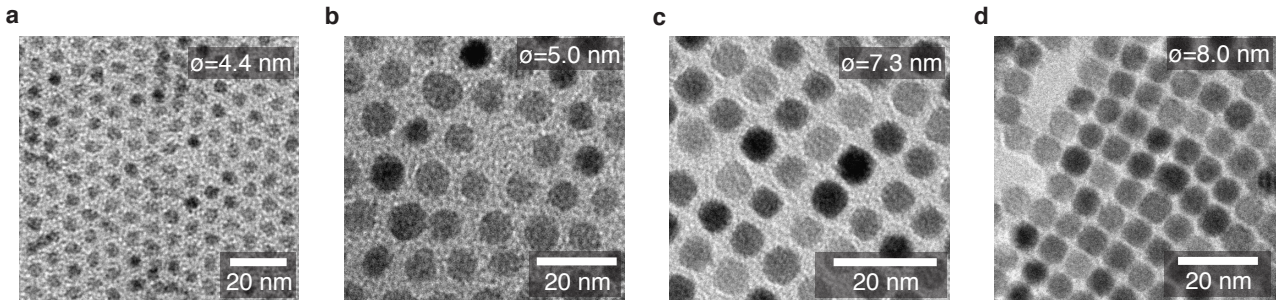

Figure 1.4: Sizes and shapes of PbSe NCs. **a**, 4.4 nm diameter. **b**, 5 nm diameter. **c**, 7.3 nm diameter. **d**, 8 nm diameter.

chapters explore assembly of PbSe NCs into ordered solids and their electronic properties.

1.3 BIBLIOGRAPHY

[1] William Shockley and Hans J Queisser. "Detailed balance limit of efficiency of p-n junction solar cells". *Journal of applied physics* 32.3 (1961), pp. 510–519.

[2] A Paul Alivisatos. "Nanocrystals: building blocks for modern materials design". *Endeavour* 21.2 (1997), pp. 56–60.

[3] R Leon et al. "Spatially resolved visible luminescence of self-assembled semiconductor quantum dots". *Science* 267.5206 (1995), p. 1966.

[4] G Springholz et al. "Self-organized growth of three-dimensional quantum-dot crystals with fcc-like stacking and a tunable lattice constant". *Science* 282.5389 (1998), pp. 734–737.

[5] L Brus. "Quantum crystallites and nonlinear optics". *Applied Physics A* 53.6 (1991), pp. 465–474.

[6] A Paul Alivisatos. "Semiconductor clusters, nanocrystals, and quantum dots". *Science* 271.5251 (1996), p. 933.

[7] C. B. Murray, C. R. Kagan, and M. G. Bawendi. "Synthesis And Characterization Of Monodisperse Nanocrystals And Close-Packed Nanocrystal Assemblies". *Annual Review of Materials Science* 30.1 (2000), pp. 545–610.

[8] Frank W Wise. "Lead salt quantum dots: the limit of strong quantum confinement". *Accounts of Chemical Research* 33.11 (2000), pp. 773–780.

[9] Iwan Moreels et al. "Composition and size-dependent extinction coefficient of colloidal PbSe quantum dots". *Chemistry of Materials* 19.25 (2007), pp. 6101–6106.

[10] Quanqin Dai et al. "Size-Dependent Composition and Molar Extinction Coefficient of PbSe Semiconductor Nanocrystals". *ACS Nano* 3.6 (2009). PMID: 19435305, pp. 1518–1524.

[11] Gi-Hwan Kim et al. "High-Efficiency Colloidal Quantum Dot Photovoltaics via Robust Self-Assembled Monolayers". *Nano letters* 15.11 (2015), pp. 7691–7696.

[12] Seth Coe-Sullivan et al. "Quantum dots for LED downconversion in display applications". *ECS Journal of Solid State Science and Technology* 2.2 (2013), R3026–R3030.

[13] Tobias Hanrath. "Colloidal nanocrystal quantum dot assemblies as artificial solids". *Journal of Vacuum Science & Technology A* 30.3 (2012), p. 030802.

[14] Inuk Kang and Frank W Wise. "Electronic structure and optical properties of PbS and PbSe quantum dots". *JOSA B* 14.7 (1997), pp. 1632–1646.

[15] JM An et al. "The peculiar electronic structure of PbSe quantum dots". *Nano Letters* 6.12 (2006), pp. 2728–2735.

[16] M Tuan Trinh et al. "Nature of the second optical transition in PbSe nanocrystals". *Nano letters* 8.7 (2008), pp. 2112–2117.

[17] Jeffrey J Peterson et al. "Uncovering forbidden optical transitions in PbSe nanocrystals". *nano Letters* 7.12 (2007), pp. 3827–3831.

[18] S. Hayashi, M. Ito, and H. Kanamori. "Raman study of gas-evaporated germanium microcrystals". *Solid State Communications* 44.1 (1982), pp. 75–79.

[19] Alexey I Ekimov, Al L Efros, and Alexei A Onushchenko. "Quantum size effect in semiconductor microcrystals". *Solid State Communications* 56.11 (1985), pp. 921–924.

[20] HW Kroto et al. "C60 Buckminsterfuler". *Nature* 162 (1985), pp. 318–320.

[21] A Lipovskii et al. "Synthesis and characterization of PbSe quantum dots in phosphate glass". *Applied physics letters* 71.23 (1997), pp. 3406–3408.

[22] Christopher B Murray et al. "Colloidal synthesis of nanocrystals and nanocrystal superlattices". *IBM Journal of Research and Development* 45.1 (2001), pp. 47–56.

[23] William W Yu et al. "Preparation and characterization of monodisperse PbSe semiconductor nanocrystals in a noncoordinating solvent". *Chemistry of materials* 16.17 (2004), pp. 3318–3322.

[24] Mark P Hendricks et al. "A tunable library of substituted thiourea precursors to metal sulfide nanocrystals". *Science* 348.6240 (2015), pp. 1226–1230.

[25] Victor K La Mer. "Nucleation in Phase Transitions." *Industrial & Engineering Chemistry* 44.6 (1952), pp. 1270–1277.

[26] G Wulff. "On the question of speed of growth and dissolution of crystal surfaces". *Z. Kristallogr* 34 (1901), pp. 449–530.

[27] Clive R Bealing et al. "Predicting nanocrystal shape through consideration of surface-ligand interactions". *ACS nano* 6.3 (2012), pp. 2118–2127.

[28] Kyung-Sang Cho et al. "Designing PbSe nanowires and nanorings through oriented attachment of nanoparticles". *Journal of the American Chemical Society* 127.19 (2005), pp. 7140–7147.

[29] Arjan J Houtepen et al. "The hidden role of acetate in the PbSe nanocrystal synthesis". *Journal of the American Chemical Society* 128.21 (2006), pp. 6792–6793.

CHAPTER 2

ASSEMBLY OF ORDERED NANOCRYSTAL SOLIDS

Thin films of NCs have been deposited by drop-casting[1], spin-casting[2], doctor-blade casting[3], spray-casting[4], and ink-jet printing[5, 6]. The film morphology is affected by thermodynamic variables including interfacial energies between solvent, substrate, and colloids, and the potential of mean force (PMF) between colloids. Kinetic factors may also influence film morphology including solvent evaporation rate and colloid diffusivity. One method that affords flexibility over the thermodynamic and kinetic variables is the formation of a Langmuir (or Gibbs) layer at a liquid-vapor (or liquid-liquid) interface. This method allows a greater range and ease of tuning interfacial energies between substrate and colloids or substrate and solvent. Dong *et al* demonstrated binary superlattices with grain sizes of hundreds of nanometers over cm^2 area[7]. Because the film forms at an interface rather than in the colloidal solution, film thickness can be tuned from a monolayer to multiple layers by controlling the colloid concentration. Highly ordered monolayer superlattices have been demonstrated using this technique[8, 9].

Assembly at a liquid-vapor interface is dominated by entropy. In the following section we describe this entropic process and the experimental variables that control superlattice order. Although assembly of colloidal NCs is important to create ordered superlattices, the superlattice must be transformed to produce an electronically active material. Later we show that oriented attachment is an interesting method to produce a conductive NC solid from an insulating one. Oriented attachment is an enthalpic process involving creation of covalent bonds between NCs. Defects in the superlattice depend on both the entropic and enthalpic

processes. Therefore both processes must be understood to control nanostructure.

2.1 Entropic Assembly

The narrow size distribution of colloidal PbSe NCs enables self-assembly of structures with long-range order. For example figure 2.1 shows individual PbSe NCs with an average diameter of 7.3 nm. These interchangeable nanometer building blocks can self-assemble into structures micrometers in size. The structure and degree of order depend on several factors including NC shape[10], ligand length[11], ligand-solvent interaction, NC concentration, solvent-substrate interaction, and solvent evaporation rate[12, 13, 14]. Given the wide parameter space, predicting self-assembly behavior in general is not straightforward. The PMF describes the potential energy of a pair of NCs as a function of distance. The PMF is influenced by several forces including Coulomb, van der Waals, and depletion[15]. In general a pair potential may not accurately describe the system if many-body interactions are important[16]. In certain cases however, the PMF can be simplified significantly.

The simplest PMF used to describe the interaction of colloids is the hard sphere model. This model is appropriate when colloids exert only repulsive force at very short distance[10]. In a good solvent, in which the colloid-solvent interfacial energy is less than the colloid-colloid interfacial energy, the hard sphere model can accurately predict structures seen in experiment[17]. Long-chain hydrocarbon ligands are used to provide colloidal stability in non-polar solvents. The entropy of ligand solvation results in a repulsive potential of mean force between NCs[17]. Because there is no attractive component to the PMF, self-assembly of colloidal crystals occurs by maximizing entropy of the colloid-solvent system. If the volume fraction taken up by colloids exceeds 0.545, close packing of the colloids maximizes the volume available to the solvent and therefore maximizes the entropy of the solvent[18]. Therefore the total free energy of the system is minimized by close-packing of the colloids[10]. This method of self-assembly is commonly called evaporative or convective self-assembly where

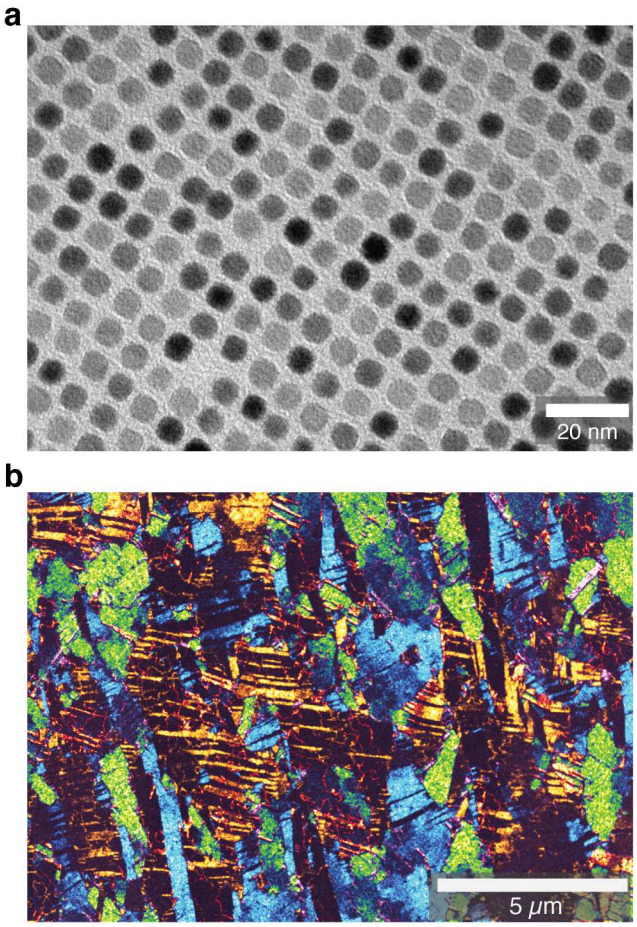

Figure 2.1: Self-assembled structures made using 7.3 nm PbSe NCs. **a**, Individual NCs can be seen in this monolayer TEM sample. **b**, A composite of several false color dark-field TEM images shows superlattice grains differentiated by the atomic orientation of NCs.

the volume fraction of colloids increases as the solvent is allowed to evaporate such as in the case of an evaporating droplet of a NC suspension[19].

During deposition of a thin-film from a colloidal suspension the solvent must evaporate. Because evaporation is a non-equilibrium process, evaporative self-assembly is kinetically controlled. The substrate has a strong effect on film uniformity, with solid substrates and liquid substrates showing opposite behavior. Rabani *et al* investigated the morphology of NC thin films deposited on solid substrates[12]. When the solvent evaporation rate is spatially uniform (outside of the binodal region) and the NC diffusion is much faster than the solvent-vapor phase boundary velocity then the NC film grows by coarsening of NC aggregates.

Aggregates form in a thin liquid layer on the solid substrate. When solvent evaporation is spatially inhomogeneous but NC diffusion still faster than the solvent liquid-vapor phase boundary motion, the resulting film is inhomogeneous with NCs collecting in boundaries between vapor cells.

With a liquid substrate however, uniform and ordered superlattices form when NC diffusion is slower than evaporation. Bigioni *et al* and Narayanan *et al* showed that alkanethiol coated gold NCs monolayers form at the surface of an evaporating colloidal droplet when evaporation is sufficiently fast[14, 13]. Under these conditions NCs are kinetically trapped at the liquid-vapor interface. When evaporation is slowed, superlattice clusters form in the bulk of the solvent as NC density increases. The same authors also showed that a minimum concentration of free alkanethiol was necessary for uniform film formation.

The relation between free ligand concentration (amphiphiles such as alkanethiol and alkyl-carboxylic acids) and two-dimensional (2D) film uniformity remains unclear. In the case of film formation at a liquid-vapor interface it has been argued that a layer of the amphiphilic ligand forms at the interface thus trapping NCs at the interface by inhibiting diffusion into the bulk of the solvent[14]. Lin *et al* however have argued that excess ligand prevents dewetting of the solvent from the solid substrate thus preventing inhomogeneous evaporation[20].

It has been shown that solvent vapor annealing promotes long-range order[21, 22]. During evaporation of a NC suspension, control of solvent vapor concentration is critical to produce highly ordered structures. The effect of solvent vapor concentration on disorder is shown in figure 2.2. To investigate the effect of solvent vapor concentration on long-range superlattice order during evaporative self-assembly, grazing incidence small angle X-ray scattering (GISAXS) data was collected in-situ. During the experiment, 6 µL of a 5 µM concentration of 7.3 nm diameter PbSe NCs in hexane was deposited onto the surface of ethylene glycol within a sealed chamber at 26 °C. In one case the chamber contained a reservoir of hexane while

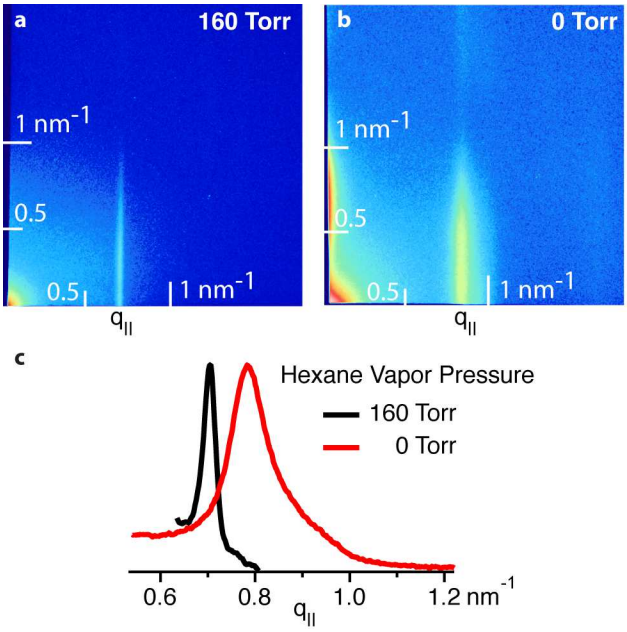

Figure 2.2: Effect of solvent vapor concentration on superlattice order. **a**, GISAXS pattern from a NC superlattice formed at 26 °C in saturated hexane vapor. **b**, GISAXS pattern from a NC superlattice formed without solvent vapor annealing. **c**, Vertically integrated intensities show disorder induced line broadening.

in the other case no hexane was added. The absence of structure in the vertical direction shows that in both cases a monolayer NC film formed at the interface. In the presence of saturated hexane vapor, which has a vapor pressure of 160 mTorr at 26 °C[23], the width of the in-plane reflection is limited only by instrumental broadening. Given our experimental setup, this indicates superlattice grain size greater than 200 nm[24]. Without solvent vapor, the linewidth broadens to a full width at half maximum (FWHM) of 0.138 nm^{-1}. Using the Scherrer equation with a constant $K = 0.9$, gives an average grain size of 41 nm.

Control over evaporation rate, free amphiphile concentration, and solvent vapor concentration can greatly improve superlattice film uniformity. However, films prepared from colloidal suspensions are quasi-crystalline with correlation lengths of a few hundred nanometers at best[25, 26]. It follows from the discussion regarding solvent vapor annealing that solvated ligands result in a purely repulsive PMF similar to hard spheres and therefore assembly is driven by entropy. As the solvent vapor is removed, attractive van der Waals

force between ligands causes distortion[27]. Distortion of the superlattice can be observed by in-situ GISAXS measurement of the interparticle distance and Scherrer broadening.

Scattering by a superlattice during drying is shown in figure 2.3. To slow the drying rate, the superlattice was contained within a sealed chamber that contained a reservoir of hexane. The superlattice was deposited on a substrate of ethylene glycol supported in a 1 cm by 1 cm by 5 mm rectangular well of teflon, mounted on a temperature controlled stage at 26 °C. After adding hexane the chamber was sealed, then 3 µL of a 5 µM concentration of 7.3 nm diameter PbSe NCs in hexane was deposited on the ethylene glycol surface by an air-tight syringe through a rubber septum. GISAXS patterns were collected every 72 s while hexane evaporated from the NC suspension. After most of the solvent had evaporated, a superlattice assembled at the ethylene glycol liquid surface. The superlattice structure was face centered cubic (FCC) with the {111} planes parallel to the liquid-vapor interface. Initially the interparticle distance was 10.48 nm and the average superlattice grain size 378 nm. As the solvent continued to evaporate, the interparticle distance reduced to 9.73 nm and the average grain size reduced to 130 nm. One contribution to broadening is instrumental, which increases with the in-plane scattering vector $q_{||}$. However the change in peak width observed is at least four times greater than the change expected from instrumental broadening[24].

As solvent leaves the superlattice, an attractive van der Waals force between ligands increases causing the interparticle distance to decrease. Concomitantly the superlattice disorder increases. The {100} superlattice peak does not broaden homogeneously rather a broad shoulder develops from grains with smaller interparticle spacing. This suggests that the sample does not dry homogeneously, rather there is a distribution of interparticle spacings. We could assume, for simplicity, that only two types of superlattice grains dominate rather than a continuous distribution. This assumption can be rationalized by considering that a thin layer of solvent condensate is metastable during drying. Some superlattice grains are wetted by the condensate and others have lost all solvent. TEM shows that the symmetry

of the superlattice does not change after complete drying. Therefore the {100} reflection in figure 2.3 could be deconvolved into two peaks during the latter stages of drying. This interpretation leads to structures with a superlattice constant differing by 2 Å to 3 Å. The actual influence of drying on the superlattice structure may be significantly more complicated because not only the interparticle distance can change, but also the grain size and coherence length.

Distortion of the superlattice by solvent loss is a major obstacle to the fabrication of highly ordered superlattices. One possible solution is to functionalize the ligands thereby tuning the PMF in the absence of solvent. However, this poses a challenge to the ultimate goal of strong electronic coupling. An alternative solution could be to arrest the crystalline order of a solvated colloidal crystal by forming covalent bonds between NCs before complete drying. To this end a liquid substrate may be advantageous for introduction of chemical reagents before or during self-assembly to initiate covalent bonding. In the next section we describe such a process, called oriented attachment.

2.2 Enthalpic Assembly

Coupling of PbS or PbSe NCs using short organic ligands is a common approach for fabricating NC solids for PV applications. Cross-linking ligands such as ethaneditihol are used to reduce interparticle distance and improve electronic coupling[28, 29]. Whether bi-functional ligands actually cross-link NCs remains unclear. Choi *et al* have shown that interparticle distance scales with ligand length for a series of dithiols[30]. However Weidman *et al* measured interparticle distances that were not strongly correlated to dithiol ligand length and showed that monothiol and dithiol ligands of similar length gave similar interparticle distances[31]. Commonly the native oleic acid ligands are exchanged for shorter ligands after deposition and drying of a superlattice film on a solid substrate. Due to the volume change

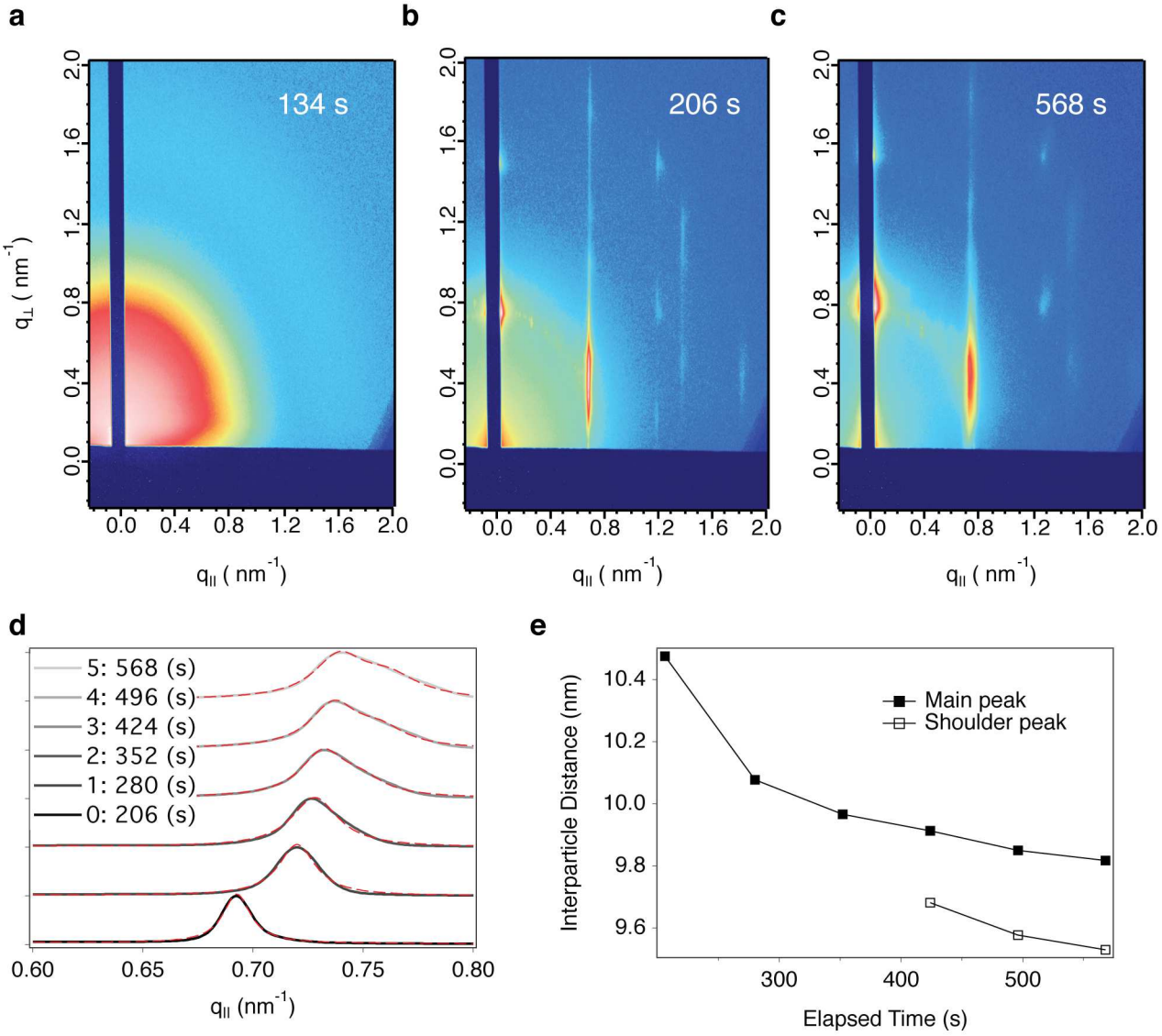

Figure 2.3: Effect of drying on superlattice structure. **a**, GISAXS pattern from a colloidal suspension of 7.3 nm diameter NCs. **b-c**, Scattering patterns recorded just after assembly and after several minutes of drying. **d**, Traces showing the location and width of the {100} peak during drying. Dashed red lines represent a fit to the data by an exponential background and one or two Lorentz peaks. **e**, Evolution of interparticle distance during drying.

during solid-state ligand exchange, microscopic defects are introduced and long-range order is compromised[31]. However by allowing sufficient time during for the exchange, short-range order can be preserved in thick superlattices (15 unit cells) [31, 32]. Similar preservation of order during exchange on monolayer films has been demonstrated using a liquid substrate instead of a solid substrate[33].

Short organic linker molecules reduce interparticle distance but do not enforce translational order. The order of the initial superlattice may be preserved by slow ligand exchange. Alternatively, one can use the atomic structure of the NC to impose translational order on the superlattice through the NC-NC interaction. Oriented attachment is one method to enforce atomic coherence between neighboring NCs.

2.2.1 Oriented Attachment

A possible route to increase order in superlattices is by covalent attachment of NCs in specific crystallographic directions. The atomic structure of the substituent NCs can be used to control NC orientation to achieve a higher degree of order than by entropic close-packing. Oriented attachment is the covalent bonding of two NCs along a single crystallographic axis. The mechanism of oriented attachment has been studied in diverse NC systems including metals, metal oxides and semiconductors[34, 35, 36, 37, 38, 39]. Anisotropic nanostructures such as rods, wires, rings, and sheets have been formed by oriented attachment[40, 38, 41]. Oriented attachment involves displacement of ligands, self-alignment of crystal facets and covalent bonding.

Schapotschnikow *et al* simulated the attachment of ligand-free PbSe NCs by molecular dynamics and found the process occurs within tens of ns[42]. In their simulation, oriented attachment was driven by covalent bond formation. Thermal motion of the NCs resulted in, at first, a single covalent bond between an atom on each NC. This was followed by alignment

of the NCs to each other and finally bonding of all atoms at the facets.

The crystallographic direction of attachment can be controlled by altering the surface energies of different facets. This was demonstrated using co-surfactants in addition to the native ligands. Cho *et al* showed that attachment of the {100} or {111} facets could be selected by introducing alkyl-phosphonic acids or alkyl-amines respectively[40]. Schliehe *et al* observed attachment of {110} facets of PbS NCs in the presence of chlorinated solvents[41]. These examples of oriented attachment were conducted in solution at elevated temperature. Alternative to oriented attachment in solution, Evers *et al* showed that thermally induced oriented attachment of PbSe NCs could be accomplished in a thin film at a liquid-vapor interface[9]. Without co-surfactants to influence the surface energies of different facets, attachment was predominantly through {100} facets. Although there was initial evidence that suggested attachment through {110} facets, high resolution transmission electron microscopy (HRTEM) later showed attachment only of {100} facets[43].

With a single species of ligand bound to the NC surface, attachment occurs at the facet with the least ligand binding energy. DFT calculations have shown that the binding energy of lead oleate at the {100} surface of PbSe is about half that of the {111} surface[44]. Therefore upon heating, ligands are displaced from the {100} surface at a greater rate than the {111}. This is a thermodynamic explanation for preferential attachment of PbSe NCs at the {100} facet.

Displacement of ligands can be induced chemically instead of thermally. Lead-chalcogenide NCs are synthesized using an X-type (anionic) ligand, the conjugate base of the alkyl carboxylic acid oleic acid[45]. One method for chemical displacement of the X-type ligands is to introduce L-type (neutral electron donor) ligands. The L-type ligand coordinates with a surface metal atom (M) bound to two carboxylates (X) to displace L-MX_2 from the surface. Charge neutrality is maintained by an L-type ligand bound to the NC[46]. Anderson *et al* classified the equilibrium constants for the displacement reaction 2L + NC-MX_2 $\leftrightarrows$ L-MX_2

+ NC-L for various L-type ligands. Small alkyl amines and diamines most readily displaced ligands from lead and cadmium chalcogenide NCs while fewer ligands were displaced by larger diamines, alcohols, tri-alkyl amines and tri-alkyl phosphines[46].

Chemically induced oriented attachment was shown by Baumgardner *et al* using dimethylformamide to displace oleate ligands from PbSe NCs[47]. Attachment occurred preferentially at the {100} facet. Sandeep *et al* also observed oriented attachment of PbSe {100} facets using tetramethylethylene-1,2-diamine, hexylamine, butylamine, and ethylenediamine[48]. However these examples of chemically induced oriented attachment created structures that were not colloidally stable, therefore oriented attachment was conducted by deposition of NCs onto a solid substrate followed by exposure to an L-type ligand solution. The resulting structures exhibited only short-range order, up to several NCs.

2.3 Combined Entropic & Enthalpic Assembly

We have outlined how superlattice order can be affected by entropic and enthalpic processes. Entropic assembly of colloids with long alkyl ligands that behave like hard-spheres in the presence of solvent leads to long range order and weak electronic coupling[32]. Enthalpic assembly by oriented attachment leads to short-range order and strong electronic coupling[47]. Superlattices with long-range order can be templated by entropic self-assembly, followed by oriented attachment for strong coupling. We now consider the conversion from entropically driven, weakly coupled assemblies to enthalpically driven, strongly coupled assemblies.

In figure 2.4 we show three distinct morphologies: hexagonally packed unconnected NCs, linear segments of unconnected NCs, and linear segments of connected NCs that are connected at right angles. This suggests that formation of an epitaxially connected superlattice by thermal treatment may occur by nucleation of the connected phase within the uncon-

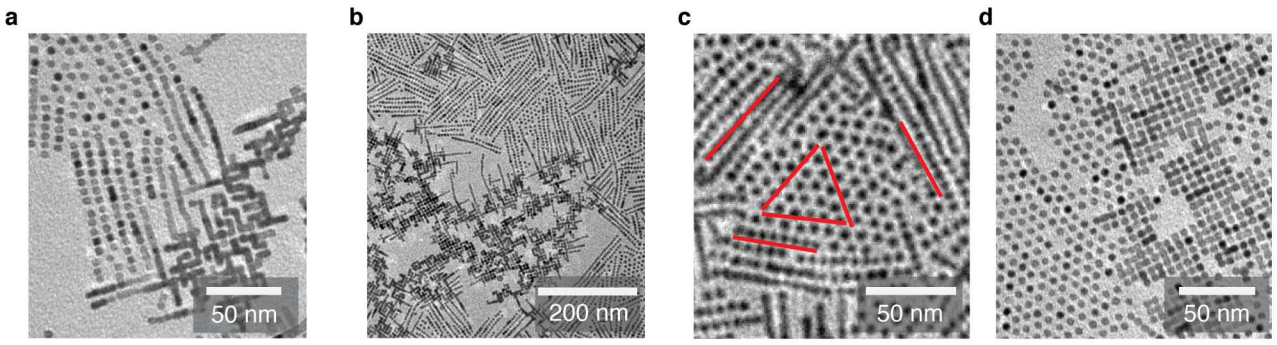

Figure 2.4: Growth of a square NC superlattice from hexagonally close-packed NCs by oriented attachment. **a**,TEM image of PbSe NCs assembled on EG at 70 °C. **b**, Lower magnification shows the dendritic nature of the square superlattice. **c**, Oriented attachment of NCs into linear segments occurs along the three axes of the hexagonal superlattice. **d**, Square and hexagonal superlattices share a common axis of symmetry.

nected phase followed by growth of the connected phase at the expense of the unconnected phase. For thermal ligand displacement at least, figure 2.4 shows that oriented attachment propagates along a primitive lattice vector of the unconnected superlattice. In the case of a hexagonal unconnected superlattice, linear segments adopt three orientations due to the three-fold rotational symmetry of the hexagonal lattice. This suggests that attachment may be directed by a dipole oriented along the linear segment rather than by random attachment of nearest neighbors (NNs)[40].

Nucleation sites for the connected phase may be at defects in the unconnected phase. At defects such as grain boundaries, vacancies, and dislocations NCs have a lower coordination number. Fewer NNs results in fewer van der Waals bonds between ligands and therefore greater probability of ligand displacement. When ligands are displaced from an undercoordinated NC, its neighbors become undercoordinated and ligand displacement propagates outward from the defect site along the primitve lattice vectors connecting NNs. After ligands are displaced, the NC is free to rotate and translate allowing oriented attachment to propagate.

The hypothesis that a connected superlattice nucleates at defects in an unconnected su-

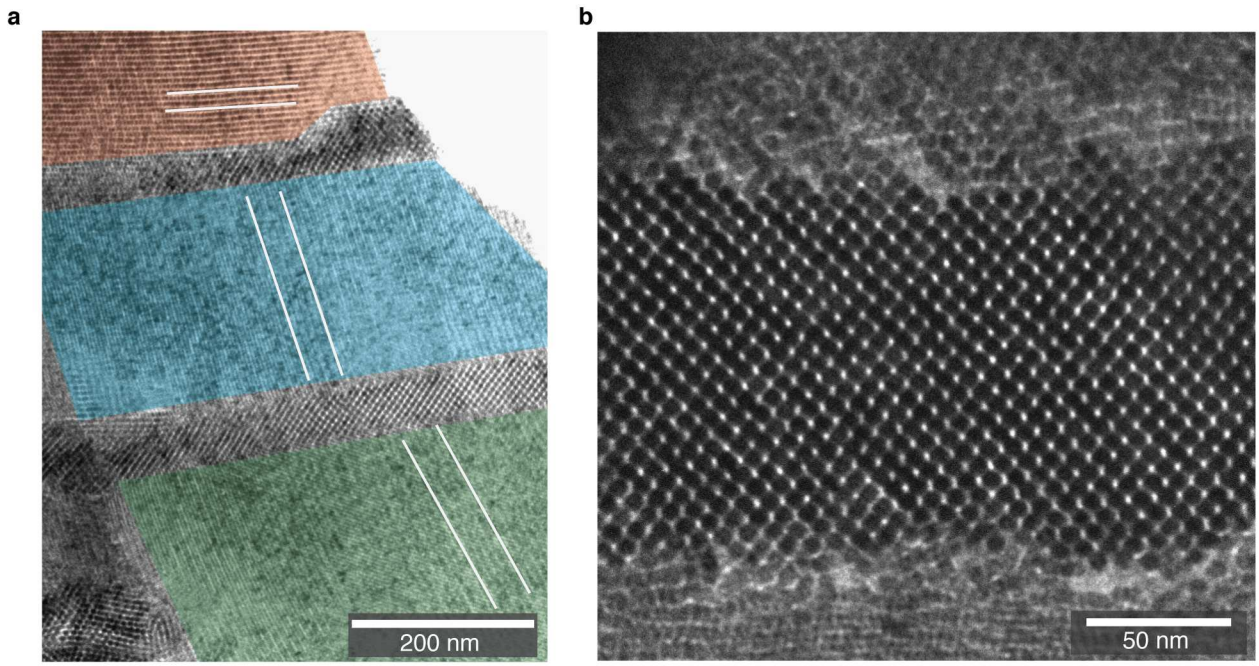

Figure 2.5: Connected superlattices at unconnected superlattice grain boundaries. **a**, Three unconnected superlattice grains are shaded. The angles of the {110} planes are indicated by white lines. **b**, A connected superlattice at the boundary of two unconnected superlattice grains.

perlattice is supported by TEM images of 6.5 nm diameter PbSe NC superlattices during oriented attachment. In figure 2.5 three unconnected superlattice grains are separated by two connected superlattice grains. Each unconnected superlattice grain has a unique orientation, thus creating grain boundaries at their intersections. It is clear that the connected superlattice grains are located at the boundaries where the unconnected grains intersect.

The structure of unconnected superlattices is determined, as explained above, by the PMF and interaction with the substrate. Because fully ligated NCs interact like hard spheres with no enthalpic contribution, assembly is driven by entropy to minimize superlattice volume, thus forming close packed FCC or hexagonal close-packed (HCP) structures. Deviation of NC shape from quasi-spherical (or partial ligand loss) can change the PMF to be non-centrosymmetric resulting in less close packed structures such as body centered cubic (BCC)[49]. Minimization of liquid-vapor or liquid-liquid interfacial energy by maximal pack-

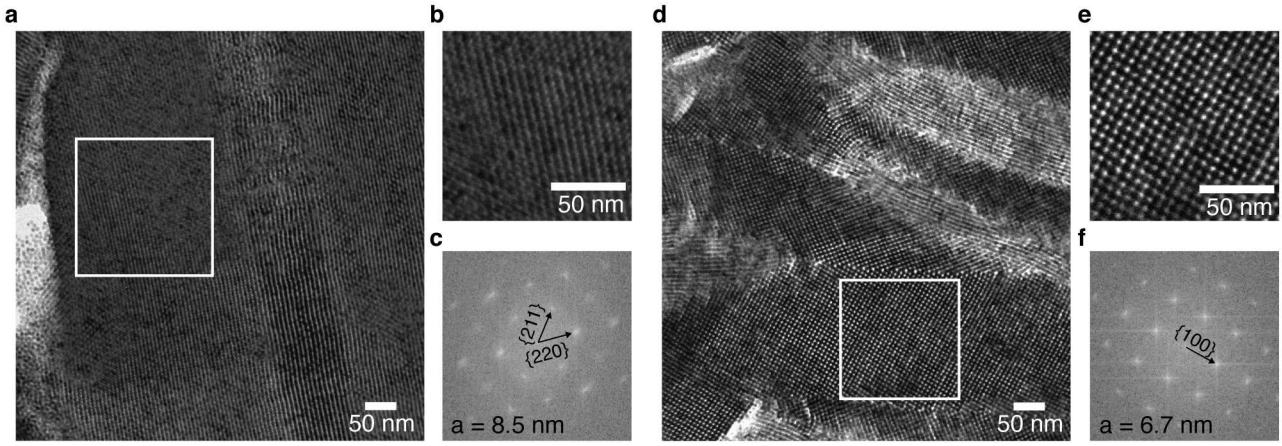

Figure 2.6: Superlattice structures before **(a-c)** and after **(d-f)** ligand displacement by EDA. **a**, TEM image of a superlattice of 6.5 nm diameter PbSe NCs assembled at the interface of EG. **b**, Magnified selected area. **c**, FFT of selected area. **d**, TEM image of superlattice grains after exposure to EDA. **e** Magnified selected area. **f**, FFT of selected area.

ing of NCs at the interface results in the close packed plane parallel to the substrate[50]. For FCC this is the {111} plane, and for BCC the {110}[21, 49, 22]. Connected superlattice structure is determined by the direction of oriented attachment. As explained above, for PbSe NCs attachment occurs in the ⟨100⟩ directions resulting in simple cubic structure.

An unconnected superlattice of 6.5 nm diameter PbSe is shown in figure 2.6. The fast Fourier transform (FFT) shows a pattern consistent with the {110} plane of a BCC superlattice parallel to the substrate. No orientations other than {110} parallel to the substrate were observed except monolayers. The lattice constant was calculated to be 8.5 nm from the {220} plane frequency in the FFT. After exposure to 50 mM ethylenediamine (EDA) in ethylene glycol (EG) for 5 min, a new structure was observed. This structure is simple cubic (SC) with the {100} plane parallel to the substrate and a lattice constant of 6.7 nm. The SC phase is formed by oriented attachment through the {100} NC facets[26]. Grain boundaries exist between SC and SC and between SC and BCC superlattices. Twin boundaries in unconnected NC superlattices are common (see figure 2.1)[51, 52]. However, grain boundaries between connected and unconnected superlattices are unexplored.

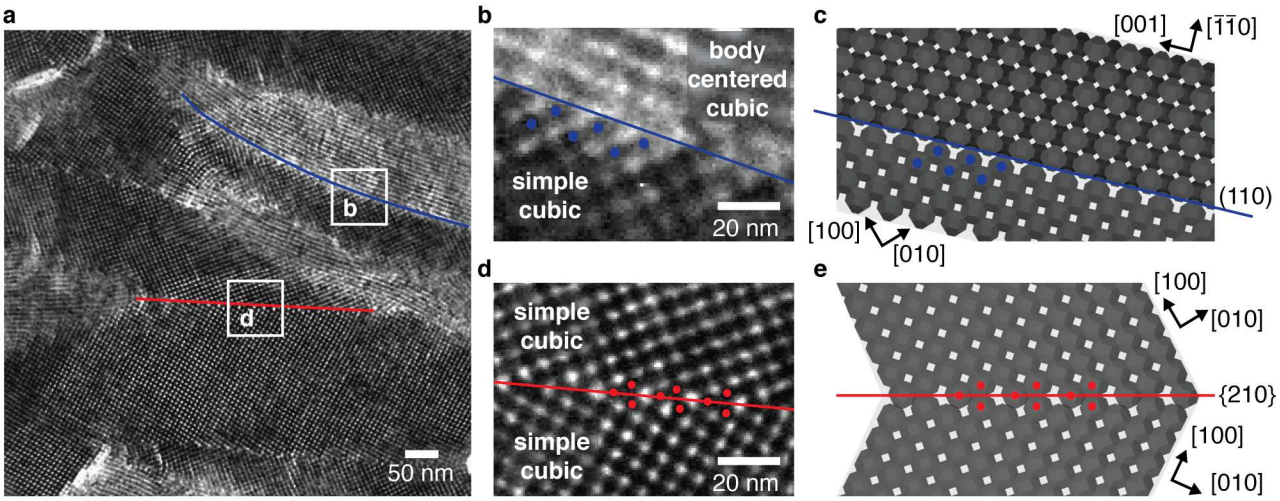

Figure 2.7: Grain boundaries between connected and unconnected superlattices. **a**, Two of several superlattice grain boundaries are indicated. One between two connected superlattices (d), and one between a connected and an unconnected superlattice (b). **b**, Detailed view of the interface between a connected SC superlattice and an unconnected BCC superlattice. **c**, Schematic representation of the SC-BCC grain boundary. **d**, Detailed view of the interface between two connected SC superlattices. **e**, Schematic representation of the SC-SC grain boundary.

Just as in the case of monolayers (figure 2.4), the transformation of an unconnected superlattice to a connected superlattice propagates in a crystallographic direction. The interface between unconnected and connected phases is shown in figures 2.7 and 2.8. The {110} plane is the boundary between the connected SC superlattice and the unconnected BCC superlattice. The lattice constant of the unconnected superlattice is about 2 nm larger than the connected superlattice because of the ligands. Since the NC diameter is 6.5 nm, the two lattice constants differ by roughly $\sqrt{2}$ (6.7 nm vs. 8.5 nm). If the lattice constant of a BCC superlattice is $\sqrt{2}$ times larger than the lattice constant of a SC superlattice, the {110} plane of the BCC superlattice is coherent with the {110} plane of the SC superlattice. As a result of this coherence, the BCC superlattice can transform into the SC superlattice with little strain or lattice defects. During the transformation, NCs in the BCC phase are displaced by $\sqrt{2}a$ toward the interface (in the $\langle 110 \rangle$ direction) where a is the SC superlattice constant. In addition, every other {110} plane of the BCC phase parallel to the substrate must be displaced by $a/\sqrt{2}$ in the $\langle 001 \rangle$ direction.

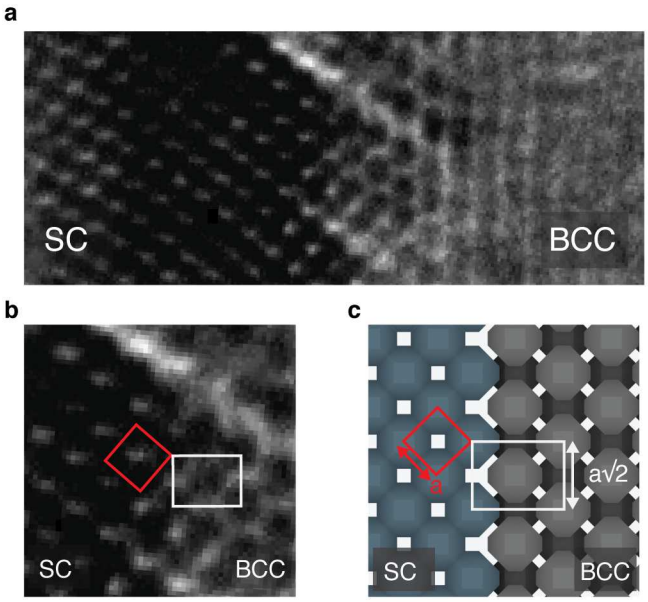

Figure 2.8: The SC-BCC interface. **a**, TEM image of a SC-BCC grain boundary. **b**, Selected area of the boundary with the SC unit cell in the {100} plane (red) and the BCC unit cell in the {110} plane (white). **c**, Schematic representation of the grain boundary. A coherent interface requires the BCC lattice constant be $\sqrt{2}$ larger than the SC.

Coherent transformation of lattice matched unconnected superlattices to connected superlattices minimizes strain induced defects such as dislocations and vacancies. Therefore the degree of order in the unconnected superlattice influences the degree of order in the connected superlattice. As described above, grain size of the unconnected superlattice can be increased by solvent vapor annealing thereby increasing the average grain size of the unconnected superlattice. Through these methods, connected superlattices with grain sizes $> 1\,\mu m$ with coherence lengths of hundreds of nm have been achieved, as shown by microscopy and X-ray data in figure 2.9. In the next section we will quantify disorder in this structure using both TEM and GISAXS.

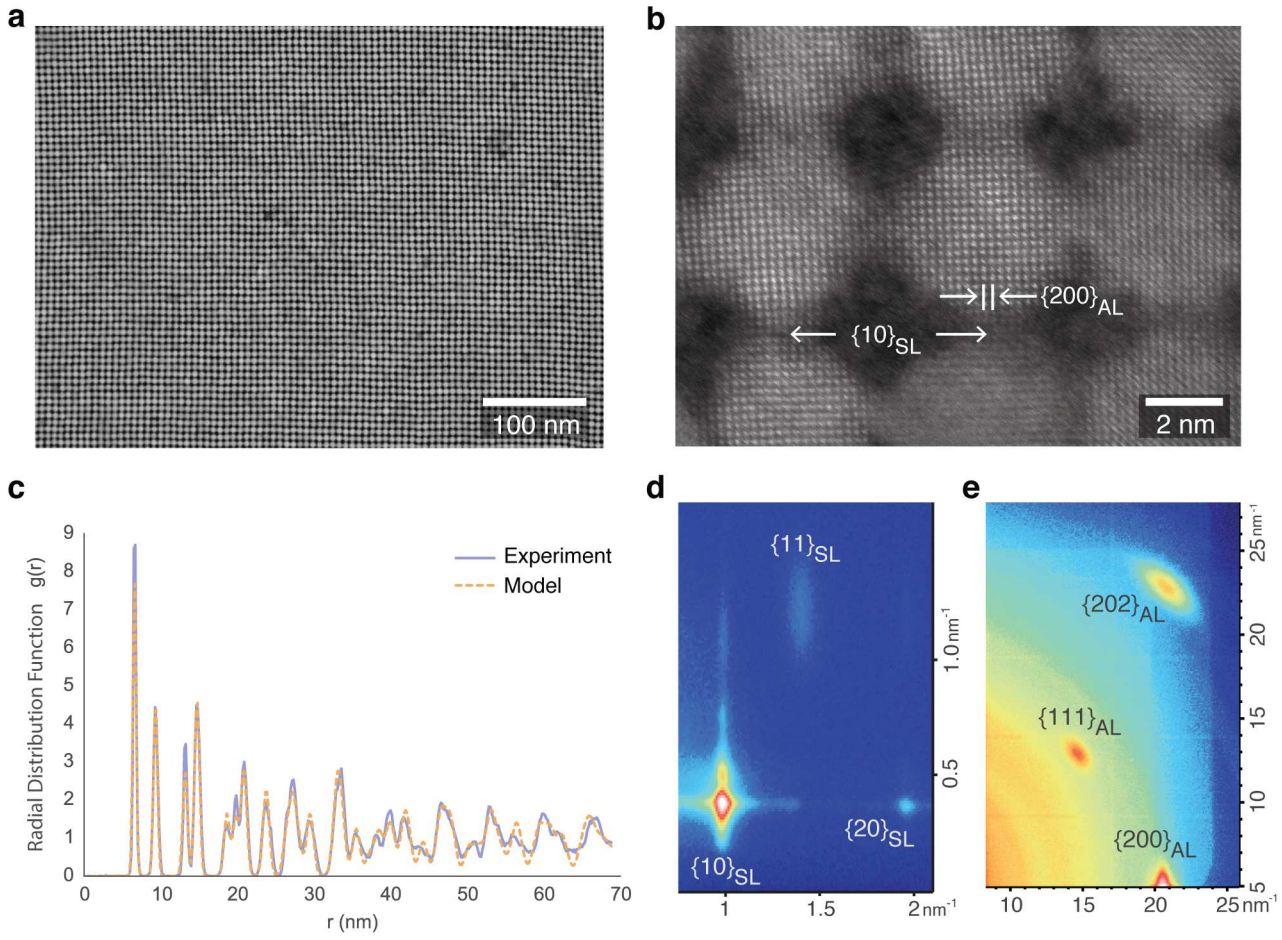

Figure 2.9: Structural characterization of a connected NC superlattice. **a-b**, Annular dark field STEM shows (a), superlattice structure and (b), atomic structure. **c**, Measured radial distribution function and calculated radial distribution function of a paracrystalline square lattice with 3.4% translational disorder. **d**, GISAXS image from a square superlattice. Subscript SL denotes superlattice. **e**, GIWAXS image showing alignment of the atomic lattices of the NCs. Subscript AL denotes atomic lattice.

2.4 Disorder in Connected Superlattices

2.4.1 Quantification of Superlattice Disorder by X-ray Scattering

To quantify disorder of the superlattice we measured GISAXS of a NC film on the surface of a field-effect transistor. The sample was prepared for GISAXS after transport measurements by dissolving the polymer encapsulation layer in chloroform. The sample was mounted on

an open stage (in air) attached to a four-axis goniometer at the D1 station of the Cornell High Energy Synchrotron Source. The sample was positioned for grazing incidence with an incident angle of 0.25 degrees. The incident radiation flux was of the order $1 \times 10^{12}\,\mathrm{s}^{-1}\mathrm{mm}^{-2}$) with a wavelength of 0.1157 nm. Scattered X-rays were collected by a Pilatus 200K detector at a distance of 902 mm with an exposure time of 3 s.

We analyzed a line-profile of the intensity along the Yoneda band. To slightly improve the signal to noise, we averaged the intensity in the vertical direction over 8 pixels or $0.08\,\mathrm{nm}^{-1}$ about the Vineyard peak.

Because the incident angle was sufficiently above the critical angle (the reflected intensity was about 2% of the incident intensity) and because the NC layer is thin, we can use the quasi-kinematic approximation instead of the distorted wave Born approximation. We calculated scattered intensity using both the local monodisperse approximation and the decoupling approximation. We find the decoupling approximation gives a better fit to experimental data. This is consistent with our assumption that there is a random spatial distribution of different size and shape NCs. We calculated the scattered intensity from a 2-D powder of a simple square superlattice, with a uniform distribution of in-plane orientations. The calculated intensity is given by: $I(q) = B_0 + S\left(V(q_\perp^0)\left\{\left(\left\langle|F(q)|^2\right\rangle - |\langle F(q)\rangle|^2\right) + |\langle F(q)\rangle|^2 Z(q)\right\}\right)$ where q is the scattering vector, B_0 is the background intensity, $V(q_\perp^0)$ is the Vineyard factor at the Yoneda band where $q_\perp^0 = 0.41$ nm^{-1}, S is a scaling factor to account for the incident photon flux, $F(q)$ is the form factor, and $Z(q)$ is the structure factor.

The spherical form factor in three dimensions is given by[53] equation (2.1) where R is the sphere radius, $q_{x,y,z}$ are the components of the scattering vector in the sample reference frame and $q_{||} = \sqrt{q_x^2 + q_y^2}$.

$$F_S(q_x, q_y, q_z, R) = \frac{4}{3}\pi R^3 exp(iq_z R)\left(3\frac{sin(q_{||}R) - q_{||}R\ cos(q_{||}R)}{(q_{||}R)^3}\right) \tag{2.1}$$

The form factor was calculated to account for a distribution of NC sizes. The distribution of sizes is given by equation (2.2) where R is the radius or other critical dimension. Equations (2.3) and (2.4) give the form factor averaged over NC size and orientation where θ is the angle between the in-plane component of the scattering vector $q_{||}$ and the x-axis of the sample reference frame.

$$P(R) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{-(R-R_0)^2}{2\sigma^2}\right) \tag{2.2}$$

$$\left\langle \left|F(q_{||})\right|^2 \right\rangle = \frac{1}{2\pi}\int_0^{2\pi}\int P(R)\ \left|F_{\mathrm{S}}(q_{||}cos(\theta), q_{||}sin(\theta), q_z, R)\right|^2 \mathrm{dRd}\theta \tag{2.3}$$

$$\left|\left\langle F(q_{||})\right\rangle\right|^2 = \left|\frac{1}{2\pi}\int_0^{2\pi}\int P(R)\ F_S(q_{||}cos(\theta), q_{||}sin(\theta), q_z, R)\ dRd\theta\right|^2 \tag{2.4}$$

Two models for the structure factor were compared. Disorder of the Debye-Waller kind, also called static disorder, describes structures where long range order is preserved, but the scattering objects are randomly displaced from the lattice points according to some probability distribution. The complete structure factor is given by $Z(q) = Z_0(q)W(q) + (1 - W(q))$. The Bragg peaks are given by $Z_0(q) = \sum_{\mathrm{hk}} m_{\mathrm{hk}}\left(\pi\Gamma\left(1 + \left(\frac{(q-q_{\mathrm{hk}})}{\Gamma}\right)^2\right)\right)^{-1}$ where m_{hk} is the multiplicity of the hk reflection located at q_{hk}, and the half width at half maximum $\Gamma = \frac{\pi K}{L}$ where L is the average superlattice grain size, and the Scherrer constant $K = 0.9$[54]. The Debye-Waller factor is given by $W(q) = exp(-B^2q^2)$ where B is the standard deviation of a normal distribution of NC displacement[53].

Paracrystalline disorder describes structures where long range order is compromised by correlation between the displacement of adjacent scatters[55]. The 2D paracrystal structure factor for a square lattice is given by[53] $Z_P(q_x, q_y) = \prod_{i=x,y} \frac{\left(1-\phi^2\right)}{(1+\phi^2-2\phi cos(q_i D))}$ where $\phi = exp(-\frac{1}{2}q_{||}^2\omega^2 - \frac{D}{L})$ and $q_{||} = \sqrt{q_x^2 + q_y^2}$. Disorder is introduced by the term ω, which is the

standard deviation of the Gaussian distribution describing the nearest neighbor distance. The effect of the finite size of superlattice grains is included by the term $\frac{D}{L}$ where D is the average nearest neighbor distance and L is the average grain size. To account for the 2D powder-like scattering, we average over all grain orientations in the sample plane. The full structure factor is therefore given by $Z_P(q_{||}) = \frac{1}{2\pi}\int_0^{2\pi} Z_P(q_{||}cos(\theta), q_{||}sin(\theta))\ d\theta$ where θ is the angle between $q_{||}$ and the x-axis of the superlattice.

We find better agreement with the paracrystal model than with the Debye-Waller model, as shown in figure 2.10. The Debye-Waller model is unable to account for the increasingly broad higher order peaks nor the diffuse scattering at small $q_{||}$. We find satisfactory agreement with equal scattering contributions from all grain orientations, as expected for random in-plane grain orientations where the average grain size is small (μm^2) compared to the size of the sampled area (mm^2). Parameters that give the best fit curves shown in figure 2.10 are, for the Debye-Waller model: $B = 0.6$ nm, $D = 6.4$ nm and $L = 100$ nm, and for the paracrystal model: $\omega = 0.3$ nm, $D = 6.4$ nm and $L = 100$ nm. For both models the best fit was found with mean NC radius $R_0 = 3.05$ nm and standard deviation $\sigma = 0.15$. The paracrystalline disorder parameter $\omega = 0.3$ nm is comparable to the value $\omega = 0.22$ nm found by fitting the radial distribution function from TEM images (see figure 2.12).

We also characterized the out-of-plane scattering to investigate the superlattice structure normal to the substrate. Being a very thin film, the structure normal to the substrate is necessarily less defined than the in-plane structure. Therefore instead of fitting the vertically scattered intensity, we simulated 2D scattering patterns for a series of thicknesses and compared to the experimental data.

Using parameters determined from analysis of the in-plane structure and form factors, we calculated the full 2D scattering images using the software package BornAgain (version 1.4.0). The model consisted of spherical particles with diameter (6.1 ± 0.3) nm arranged in a square 2D paracrystalline superlattice with superlattice constant (6.4 ± 0.3) nm and average grain

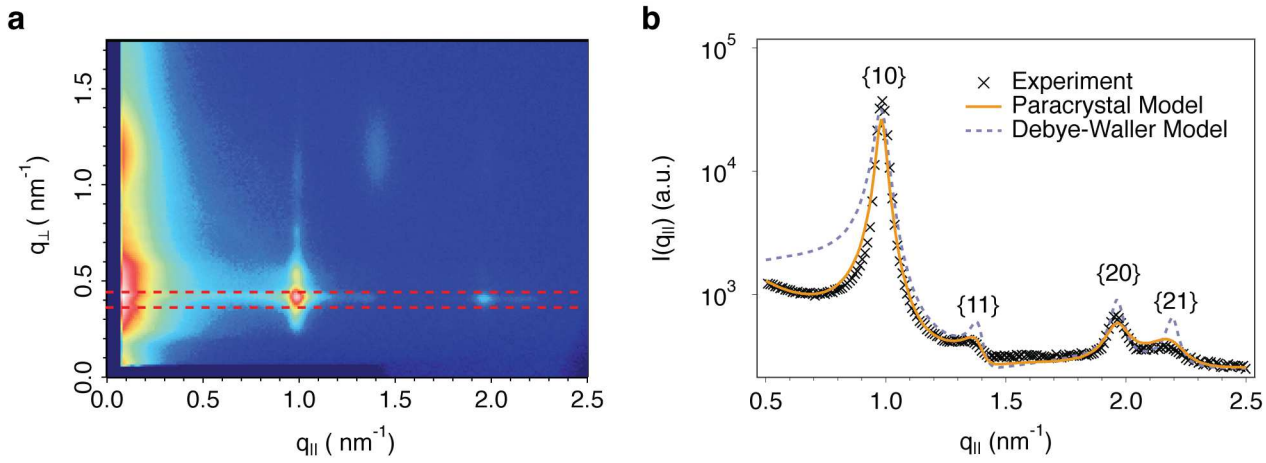

Figure 2.10: Paracrystalline superlattice structure analysis by GISAXS. **a**,Scattering from a NC superlattice on a field effect transistor. The color scale is logarithmic with arbitrary units. The Yoneda band is marked by red dashed lines. **b**, Scattered intensity along the Yoneda band. Solid lines show calculated intensity from a square lattice using a paracystal disorder model or a Debye-Waller disorder model.

size of 100 nm. The superlattice layers were on a substrate of 200 nm SiO_2 on Si (infinite thickness model). The incident angle was 0.25 deg, as in the experimental setup. Angular divergence and energy bandwidth of the beam were neglected as these were small in our experimental setup[24]. Scattering was calculated using the decoupling approximation and the distorted-wave Born approximation. Figure 2.11 shows calculated scattering intensities from monolayer to six-layer models. The layers were positioned (6.4 ± 1.0) nm apart vertically.

We found this model unable to exactly replicate the measured scattering data, however we can estimate the thickness based on features in the scattered intensity normal to the sample. In figure 2.11 we show line profiles of calculated and measured intensity along the {10} Bragg rod at $q_{||} = 0.98\,\mathrm{nm}^{-1}$. In the experimental data we observe a sharp feature near the Vineyard peak and several more peaks of increasing width and rapidly decreasing intensity. Though the calculated intensity does not reproduce the rapidly decreasing intensity, we note that a single sharp peak is only observed for superlattices more than 3 and less than 10 layers thick. As the number of layers increases, interference causes the peak to become sharper. However, as thickness continues to increase, the number of peaks at small $q_\perp$increases, leading to a

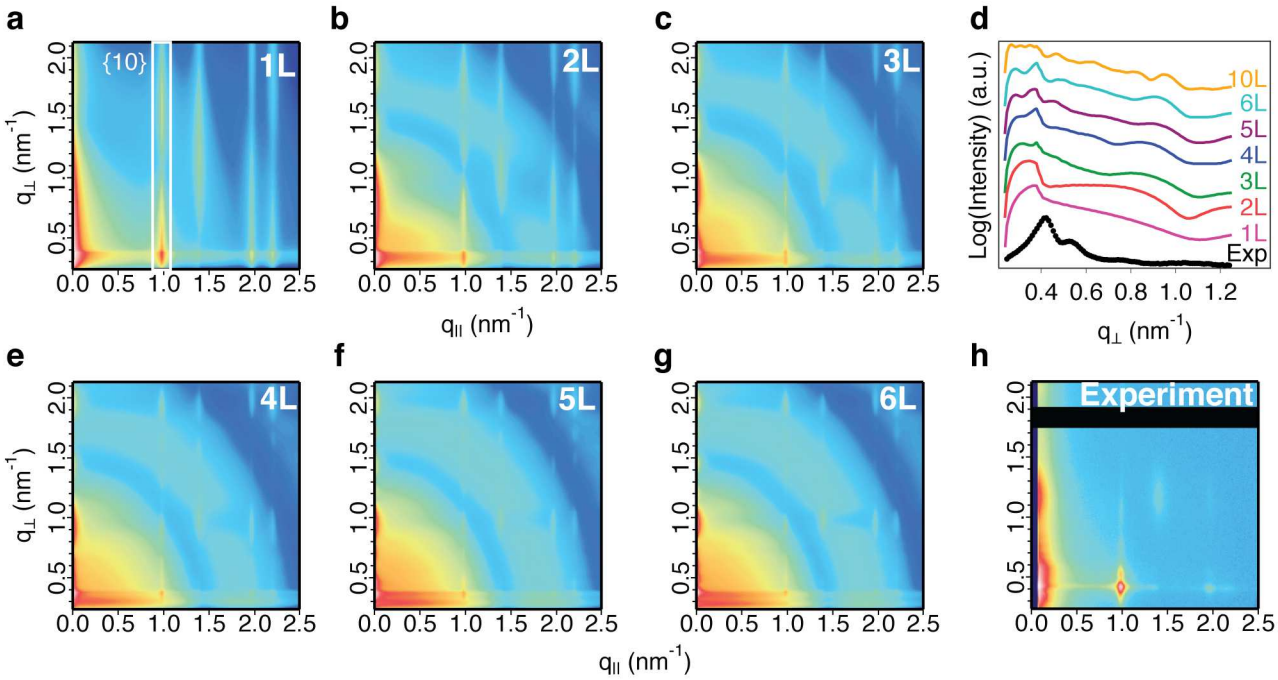

Figure 2.11: Estimation of superlattice thickness by GISAXS. **a-c,e-g**, Calculated scattered intensity from superlattice models one layer (1L) to six layers (6L) thick. Color scale is logarithmic. **d**, Intensity along the 10 Bragg rod for 1-6 layers and 10 layers (2D image not shown) compared to experimental data (Exp.). **h**, Measured scattered intensity. The black region is from a beam-stop. Color scale is logarithmic.

disappearance of the single peak. We therefore estimate the average thickness to be more than 2 but less than 10 NC layers.

2.4.2 Quantification of Superlattice Disorder by Microscopy

We calculated the radial distribution function (RDF) based on NC locations directly extracted from scanning transmission electron microscopy (STEM) images. Using a paracrystalline model, we fit the measured RDF with only one free parameter and find the standard deviation of the nearest neighbor distance to be 0.22 nm or approximately one PbSe bond length (0.306 nm).

We considered two models to fit *g(r)*: with and without correlation. Uncorrelated disorder

is modeled by equation (2.5), where the coherence between particles is independent of the interparticle distance.

$$g\left(r\right) = \frac{L^2}{2\pi r}\sum_{m,n}\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(\frac{-\left(r-L\sqrt{m^2+n^2}\right)^2}{2\sigma^2}\right) \tag{2.5}$$

$$g\left(r\right) = \frac{L^2}{2\pi r}\sum_{m,n}\frac{1}{\sqrt{2\pi(|m|+|n|)\ \sigma^2}}\exp\left(\frac{-\left(r-L\sqrt{m^2+n^2}\right)^2}{2\sigma^2(|m|+|n|)}\right) \tag{2.6}$$

Paracrystalline disorder is described by equation (2.6), in which coherence decreases with interparticle distance[55]. In both equation (2.5) and equation (2.6) L is the superlattice constant, r is the radial distance, σ is the standard deviation of the nearest neighbor distance, and m,n are the indices for the two-dimensional lattice vectors. Because oriented attachment requires the correlated motion of NCs, it is not surprising that the measured *g(r)* is better described by equation (2.6). In figure 2.12 we show fits using equations (2.5) and (2.6) for comparison. The fitting was calculated by setting the superlattice constant L to 6.6 nm, the average nearest neighbor distance between NCs in the image (see appendix A.8). The lattice constant found by GISAXS is 2 Å smaller than that measured from the STEM image, less than a single Pb-Se bond length. We find the standard deviation of the nearest neighbor distance $\sigma = 0.22\,\text{nm}$, or 3.4% relative to the superlattice constant.

2.5 BIBLIOGRAPHY

[1] Michael B Sigman, Aaron E Saunders, and Brian A Korgel. "Metal nanocrystal superlattice nucleation and growth". *Langmuir* 20.3 (2004), pp. 978–983.

[2] Ilan Gur et al. "Air-stable all-inorganic nanocrystal solar cells processed from solution". *Science* 310.5747 (2005), pp. 462–465.

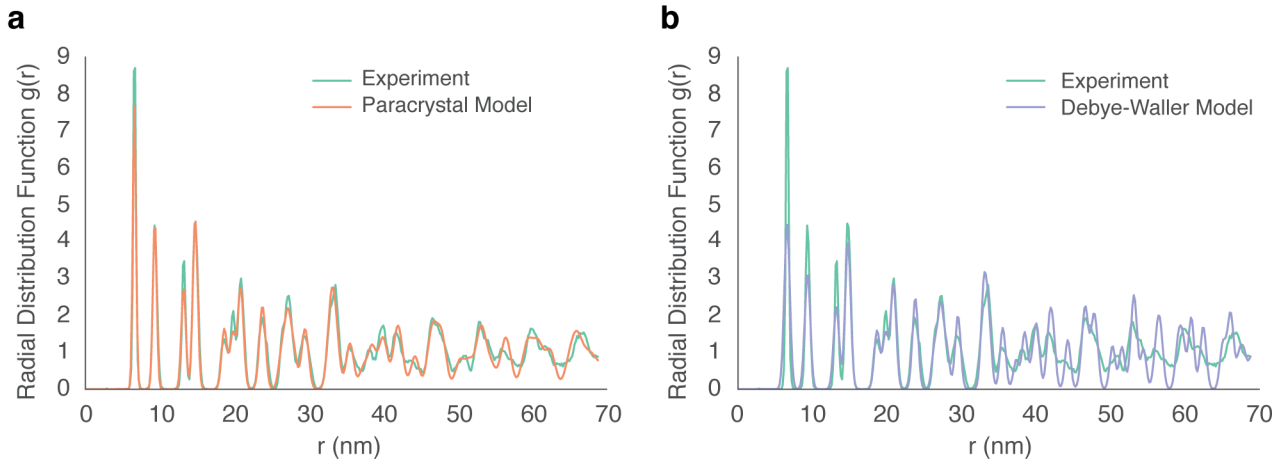

Figure 2.12: Analysis of the superlattice radial distribution function. **a**, Data measured from TEM image analysis overlaid with the calculated radial distribution function of a paracrystalline square lattice with lattice constant of 6.6 nm and standard deviation of the nearest neighbor distance of 0.22 nm. **b**, The same data overlaid with the best fit of a square lattice radial distribution function with static disorder. The lattice constant is 6.6 nm and the standard deviation of the nearest neighbor distance is 0.38 nm.

[3] Maryna I Bodnarchuk et al. "Large-area ordered superlattices from magnetic Wustite/cobalt ferrite core/shell nanocrystals by doctor blade casting". *ACS nano* 4.1 (2009), pp. 423–431.

[4] Vahid A Akhavan et al. "Spray-deposited CuInSe2 nanocrystal photovoltaics". *Energy & Environmental Science* 3.10 (2010), pp. 1600–1606.

[5] Hwa-Young Ko et al. "Rapid self-assembly of monodisperse colloidal spheres in an ink-jet printed droplet". *Chemistry of materials* 16.22 (2004), pp. 4212–4215.

[6] Brent A Ridley, Babak Nivi, and Joseph M Jacobson. "All-inorganic field effect transistors fabricated by printing". *Science* 286.5440 (1999), pp. 746–749.

[7] Angang Dong et al. "Binary nanocrystal superlattice membranes self-assembled at the liquid-air interface". *Nature* 466.7305 (2010), pp. 474–477.

[8] Angang Dong et al. "Two-dimensional binary and ternary nanocrystal superlattices: the case of monolayers and bilayers". *Nano letters* 11.4 (2011), pp. 1804–1809.

[9] Wiel H Evers et al. “Low-dimensional semiconductor superlattices formed by geometric control over nanocrystal attachment”. *Nano letters* 13.6 (2012), pp. 2317–2323.

[10] Pablo F. Damasceno, Michael Engel, and Sharon C. Glotzer. “Predictive Self-Assembly of Polyhedra into Complex Structures”. *Science* 337.6093 (2012), pp. 453–457.

[11] Uzi Landman and WD Luedtke. “Small is different: energetic, structural, thermal, and mechanical properties of passivated nanocluster assemblies”. *Faraday discussions* 125 (2004), pp. 1–22.

[12] Eran Rabani et al. “Drying-mediated self-assembly of nanoparticles”. *Nature* 426.6964 (2003), pp. 271–274.

[13] Terry P Bigioni et al. “Kinetically driven self assembly of highly ordered nanoparticle monolayers”. *Nature materials* 5.4 (2006), pp. 265–270.

[14] Suresh Narayanan”. “Dynamical Self-Assembly of Nanocrystal Superlattices during Colloidal Droplet Evaporation by ¡i¿in situ¡/i¿ Small Angle X-Ray Scattering”. *Physical Review Letters* 93.13 (2004).

[15] Kyle JM Bishop et al. “Nanoscale forces and their uses in self-assembly”. *small* 5.14 (2009), pp. 1600–1630.

[16] Carlos A Silvera Batista, Ronald G Larson, and Nicholas A Kotov. “Nonadditivity of nanoparticle interactions”. *Science* 350.6257 (2015), p. 1242477.

[17] Philipp Schapotschnikow, Rene Pool, and Thijs JH Vlugt. “Molecular simulations of interacting nanocrystals”. *Nano letters* 8.9 (2008), pp. 2930–2934.

[18] Zhengdong Cheng, William B Russel, and PM Chaikin. “Controlled growth of hard-sphere colloidal crystals”. *Nature* 401.6756 (1999), pp. 893–895.

[19] RV Craster, OK Matar, and Khellil Sefiane. “Pinning, retraction, and terracing of evaporating droplets containing nanoparticles”. *Langmuir* 25.6 (2009), pp. 3601–3609.

[20] XM Lin et al. "Formation of long-range-ordered nanocrystal superlattices on silicon nitride substrates". *The Journal of Physical Chemistry B* 105.17 (2001), pp. 3353–3357.

[21] Kaifu Bian et al. "Shape-anisotropy driven symmetry transformations in nanocrystal superlattice polymorphs". *ACS nano* 5.4 (2011), pp. 2815–2823.

[22] Tobias Hanrath, Joshua J Choi, and Detlef-M Smilgies. "Structure/processing relationships of highly ordered lead salt nanocrystal superlattices". *ACS nano* 3.10 (2009), pp. 2975–2988.

[23] K. Ruzicka and Vladimir Majer. "Simultaneous Treatment of Vapor Pressures and Related Thermal Data Between the Triple and Normal Boiling Temperatures for n-Alkanes C5–C20". *Journal of Physical and Chemical Reference Data* 23.1 (1994), pp. 1–39.

[24] D-M Smilgies. "Scherrer grain-size analysis adapted to grazing-incidence scattering with area detectors. Erratum". *Journal of Applied Crystallography* 46.1 (2013), pp. 286–286.

[25] Stefan Pichler et al. "Evaluation of ordering in single-component and binary nanocrystal superlattices by analysis of their autocorrelation functions". *ACS nano* 5.3 (2011), pp. 1703–1712.

[26] Kevin Whitham et al. "Charge transport and localization in atomically coherent quantum dot solids". *Nature Materials* 15.5 (2016), pp. 557–563.

[27] Zhang Jiang et al. "Capturing the crystalline phase of two-dimensional nanocrystal superlattices in action". *Nano letters* 10.3 (2010), pp. 799–803.

[28] Joshua J Choi et al. "PbSe nanocrystal excitonic solar cells". *Nano letters* 9.11 (2009), pp. 3749–3755.

[29] Ryan W Crisp et al. "Metal Halide Solid-State Surface Treatment for High Efficiency PbS and PbSe QD Solar Cells". *Scientific reports* 5 (2015).

[30] Joshua J Choi et al. "Photogenerated exciton dissociation in highly coupled lead salt nanocrystal assemblies". *Nano letters* 10.5 (2010), pp. 1805–1811.

[31] Mark C Weidman, Kevin G Yager, and William A Tisdale. "Interparticle spacing and structural ordering in superlattice PbS nanocrystal solids undergoing ligand exchange". *Chemistry of Materials* 27.2 (2014), pp. 474–482.

[32] Dmitri V. Talapin and Christopher B. Murray. "PbSe Nanocrystal Solids for n- and p-Channel Thin Film Field-Effect Transistors". *Science* 310.5745 (2005), pp. 86–89.

[33] A Dong, Y Jiao, and DJ Milliron. "Electronically coupled nanocrystal superlattice films by in situ ligand exchange at the liquid-air interface." *ACS Nano* 7.12 (2013), pp. 10978–10984.

[34] R Lee Penn and Jillian F Banfield. "Imperfect oriented attachment: dislocation generation in defect-free nanocrystals". *Science* 281.5379 (1998), pp. 969–971.

[35] Muralikrishna Raju, Adri CT Van Duin, and Kristen A Fichthorn. "Mechanisms of oriented attachment of TiO2 nanocrystals in vacuum and humid environments: reactive molecular dynamics". *Nano letters* 14.4 (2014), pp. 1836–1842.

[36] R Lee Penn and Jillian F Banfield. "Oriented attachment and growth, twinning, polytypism, and formation of metastable phases: Insights from nanocrystalline TiO2". *American Mineralogist* 83.9-10 (1998), pp. 1077–1082.

[37] Markus Niederberger and Helmut Cölfen. "Oriented attachment and mesocrystals: non-classical crystallization mechanisms based on nanoparticle assembly". *Physical chemistry chemical physics* 8.28 (2006), pp. 3271–3287.

[38] Weon-kyu Koh et al. "Synthesis of monodisperse PbSe nanorods: a case for oriented attachment". *Journal of the American Chemical Society* 132.11 (2010), pp. 3909–3913.

[39] Aditi Halder and N Ravishankar. "Ultrafine Single-Crystalline Gold Nanowire Arrays by Oriented Attachment". *Advanced Materials* 19.14 (2007), pp. 1854–1858.

[40] Kyung-Sang Cho et al. "Designing PbSe nanowires and nanorings through oriented attachment of nanoparticles". *Journal of the American Chemical Society* 127.19 (2005), pp. 7140–7147.

[41] Constanze Schliehe et al. "Ultrathin PbS sheets by two-dimensional oriented attachment". *Science* 329.5991 (2010), pp. 550–553.

[42] Philipp Schapotschnikow et al. "Morphological transformations and fusion of PbSe nanocrystals studied using atomistic simulations". *Nano letters* 10.10 (2010), pp. 3966–3971.

[43] MP Boneschanscher et al. "Long-range orientation and atomic attachment of nanocrystals in 2D honeycomb superlattices". *Science* 344.6190 (2014), pp. 1377–1380.

[44] Clive R Bealing et al. "Predicting nanocrystal shape through consideration of surface-ligand interactions". *ACS nano* 6.3 (2012), pp. 2118–2127.

[45] William W Yu et al. "Preparation and characterization of monodisperse PbSe semiconductor nanocrystals in a noncoordinating solvent". *Chemistry of materials* 16.17 (2004), pp. 3318–3322.

[46] NC Anderson et al. "Ligand exchange and the stoichiometry of metal chalcogenide nanocrystals: spectroscopic observation of facile metal-carboxylate displacement and binding." *J Am Chem Soc* 135.49 (2013), pp. 18536–18548.

[47] William J Baumgardner, Kevin Whitham, and Tobias Hanrath. "Confined-but-connected quantum solids via controlled ligand displacement". *Nano letters* 13.7 (2013), pp. 3225–3231.

[48] CS Suchand Sandeep et al. "Epitaxially Connected PbSe Quantum-Dot Films: Controlled Neck Formation and Optoelectronic Properties". *ACS nano* 8.11 (2014), pp. 11499–11511.

[49] Joshua J Choi et al. "Controlling nanocrystal superlattice symmetry and shape-anisotropic interactions through variable ligand surface coverage". *Journal of the American Chemical Society* 133.9 (2011), pp. 3131–3138.

[50] Joshua J Choi et al. "Interface-induced nucleation, orientational alignment and symmetry transformations in nanocube superlattices". *Nano letters* 12.9 (2012), pp. 4791–4798.

[51] Zhong Lin Wang et al. "Structural analysis of self-assembling nanocrystal superlattices". *Advanced Materials* 10.1 (1998), pp. 13–30.

[52] Sara M Rupich et al. "Size-dependent multiple twinning in nanocrystal superlattices". *Journal of the American Chemical Society* 132.1 (2009), pp. 289–296.

[53] Rémi Lazzari. "IsGISAXS: a program for grazing-incidence small-angle X-ray scattering analysis of supported islands". *Journal of Applied Crystallography* 35.4 (2002), pp. 406–421.

[54] Detlef-M Smilgies, Andrew T Heitsch, and Brian A Korgel. "Stacking of hexagonal nanocrystal layers during Langmuir–Blodgett deposition". *The Journal of Physical Chemistry B* 116.20 (2012), pp. 6017–6026.

[55] W. Vogel and R. Hosemann. "Evaluation of paracrystalline distortions from line broadening". *Acta Crystallogr A Cryst Phys Diffr Theor Gen Crystallogr* 26.2 (1970), pp. 272–277.

CHAPTER 3

CHARGE TRANSPORT IN NANOCRYSTAL SOLIDS

3.1 Localized Transport: Theory

The theory of electron motion in crystalline materials describes wave functions in a periodic array of atomic potentials. A NC superlattice is a periodic array of potential wells with a lattice constant about a factor of 10 larger than an atomic crystal. The electronic structures of ideal simple cubic[1, 2], tetragonal[1], honeycomb[3], and silicene[3] semiconductor NC superlattices have been calculated. The electronic structure of a binary superlattice has also been calculated and shows interesting electron pairing behavior, similar to Cooper pairing in superconductors[4]. Interesting phenomena such as Dirac cones, flatbands, and electron pairing result from the symmetry of a NC superlattice. As discussed in chapter 2, the symmetry of a NC superlattice can be controlled by NC size, shape, and ligands. In this way NC superlattices offer the possibility to design a material with a specific electronic structure.

Real superlattices however are not ideal, but have energetic and spatial disorder. For this reason the unique electronic properties predicted in ideal superlattices have not been realized. Disorder causes electrons to remain localized and their behavior is not affected by superlattice symmetry. The theory regarding localization was first developed for disordered atomic crystals. For NC superlattices, energetic disorder comes not from bonding disorder or changes in the local environment as in atomic crystals, but is dominated by size disorder because of the size dependent quantum confinement effect.

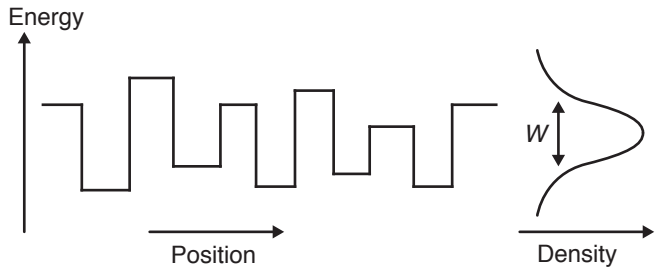

Figure 3.1: Example of disordered potential wells as described by the Anderson model. The distribution of energies is described by W.

Anderson Localization

Anderson's theory describes the relationship between energetic disorder and the coupling of localized wave functions to form delocalized wave functions[5]. In Anderson's model states are ordered in space but randomly distributed in energy, as shown in figure 3.1. An important assumption in the theory is that electron interactions can be ignored. To illustrate the effect of electron interactions, consider the energy of a single electron confined to a potential well. Next consider the change in energy caused by addition of a second electron to the well, raising the energy due to the Coulomb interaction. Anderson's theory is applicable if the energy band over which the single electron states are distributed is small compared to the Coulomb energy.

Anderson defined delocalization by considering the time evolution of an electron wave function initially confined to a single site. If the electron is localized then there will be a non-zero probability of finding the electron at the initial site in the limit as time goes to infinity. This is a strong definition in that delocalization occurs over infinite space. A localized electron wave function may extend over many sites, even macroscopic distances, yet by Anderson's strict definition is still localized. In our discussion of hopping transport we will investigate the scaling of localization radius with disorder, however it is instructive to consider the Anderson model as a fundamental limit.

We are interested to know for what amount of disorder a system transitions between localized states and delocalized states. The key value to consider is the amount of energetic disorder relative to the coupling between neighboring sites. In Anderson's model, energetic disorder is defined by the range W over which energy states are uniformly distributed. The coupling between neighboring sites is the overlap integral I. For large W/I all states are localized. According to Anderson, there must be a critical value of disorder W_c at which delocalization occurs. A mathematical solution for W_c has not been found, but numerical solutions have been found for specific lattices. Delocalization can not occur in one-dimensional (1D) or 2D lattices with any disorder. For the 3D diamond and SC lattices W_c/I is 8 and 15, respectively[6].

Anderson's theory predicts the maximum allowable disorder for infinite delocalization, however it is more practical to know how localization scales with disorder. Percolation theory is a general theory that describes how connectivity in networks scales with a physical parameter of the network. It has been used to model electrical conductivity in disordered systems. Charge transport through a system of localized wave functions occurs by phonon assisted tunneling between localized states, also known as hopping.

Hopping Transport

Hopping transport refers to the transport of charge by energy transfer with phonons to overcome the energetic barrier between localized states. As it is a thermally assisted process, temperature dependence of the conductivity is used to evaluate transport through disordered systems. Hopping transport can be categorized into three types which differ by the temperature dependence of the conductivity. In general all hopping transport can be described by percolation theory, yet the three main types represent limiting cases often encountered experimentally.

Hopping conductance is modeled by the expression $G = G_0 e^{-\left(\frac{T_0}{T}\right)^p}$, where p depends on the type of hopping and T_0 depends on material properties[7]. In all types of hopping transport, conductance is described by $G = G_0' e^{-\xi}$, where $\xi = \frac{2r}{a} + \frac{\varepsilon}{\mathrm{kT}}$, r is the hopping distance, a is the electron or hole localization length, and ε is the hopping energy[8]. In the limit $\frac{2r}{a} \gg \frac{\varepsilon}{\mathrm{kT}}$, the hopping distance is the nearest neighbor distance. This type of transport is called nearest neighbor hopping (NNH). In NNH the hopping distance r is the nearest neighbor distance regardless of temperature, therefore the conductance follows Arrhenius behavior, $G = G_0'' e^{-\frac{\varepsilon_{\mathrm{NN}}}{\mathrm{kT}}}$ where $\varepsilon_{\mathrm{NN}}$is the nearest neighbor hopping energy.

At high temperature, where $\frac{2r}{a} \gg \frac{\varepsilon}{\mathrm{kT}}$ hopping occurs between nearest neighbors. As the temperature decreases, the density of states accessible to a charge carrier within a few kT of the chemical potential becomes increasingly sparse. At a certain temperature, hopping a farther distance becomes more favorable than hopping to NNs if the energy between initial and final states is small enough to decrease ξ. This transition occurs at a higher temperature for more delocalized states: if a is large the magnitude of the energetic term in $2r/a + \varepsilon/kT$ becomes comparable in magnitude to the spatial term at higher temperature.

Our analysis of conductance *vs.* temperature follows that of Efros and Shklovskii[8] It is a generalization of Mott's variable range hopping equation using percolation theory[9]. Here we replace Mott's assumption of a constant density of states with one appropriate for the energetic and spatial distribution of states of a NC superlattice. The analysis proceeds exactly as by Efros and Shklovskii although the solution becomes non-trivial and was solved numerically.

The governing equation of percolation theory is the bonding criterion $\xi_{ij} \leq \xi$. The parameter ξ_{ij} is captures the connectivity of the system. In the case of thermally assisted electron transfer, the connectivity between two localized states depends on distance, electron localization length, energy difference, and temperature. These four variables determine the probability for charge transfer between any two states i, j which scales according to

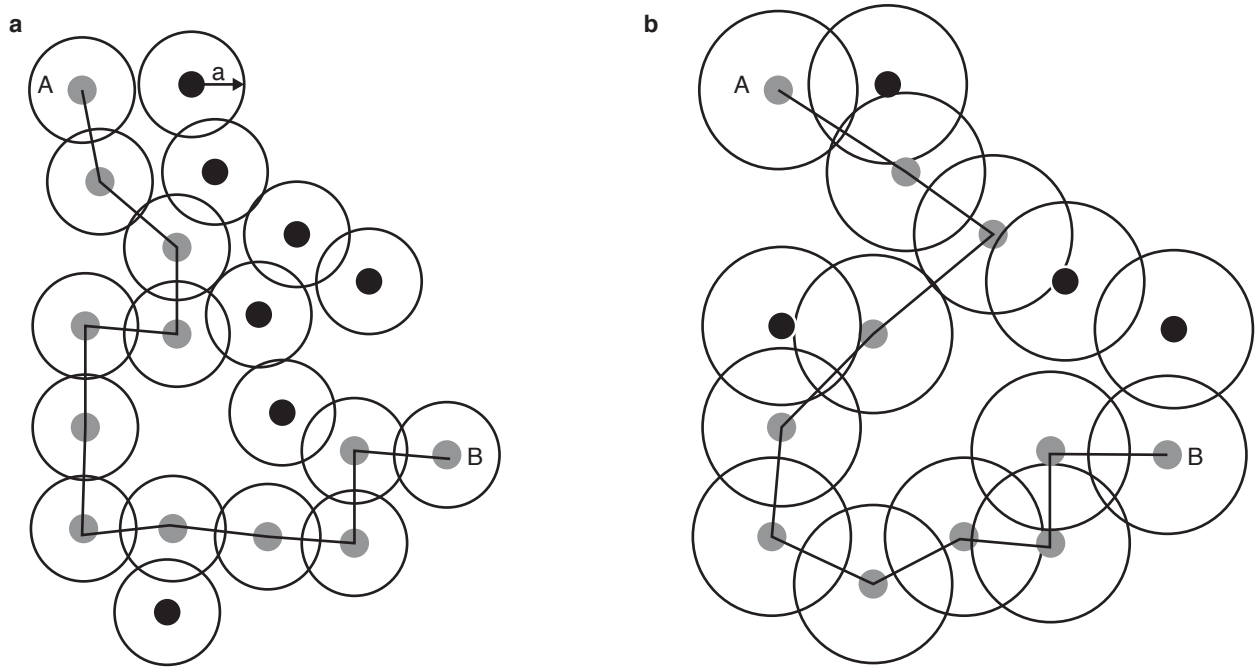

Figure 3.2: Percolation networks of different densities and localization lengths. Grey sites are close in energy while black sites are not. **a**, A percolation path from site **A** to site **B** exists with a small localization length a if the density is high engouh. **b**, At lower site density the percolation threshold is satisfied with a larger localization length.

equation (3.1)[9].

$$e^{-\left(\frac{2r_{\mathrm{ij}}}{a} + \frac{\varepsilon_{\mathrm{ij}}}{\mathrm{kT}}\right)} = e^{-(\xi_{ij})} \tag{3.1}$$

The distance between sites is r_{ij}, the energy difference is ε_{ij}, a is the electron or hole localization length, the thermal energy is kT where k is the Boltzmann constant and T is the temperature. The wave function of the $1\mathrm{S_e}$ or $1\mathrm{S_h}$ state of a NC is approximately spherical[10], therefore we may use a scalar localization length. In the case of a single NC, the localization length is the same as the decay length of the $1\mathrm{S_e}$ ($1\mathrm{S_h}$) wave function for an electron (hole).

Figure 3.2 illustrates the concept of a percolation network through sites with random spatial and energetic distribution. The existence of a percolation path depends not only on

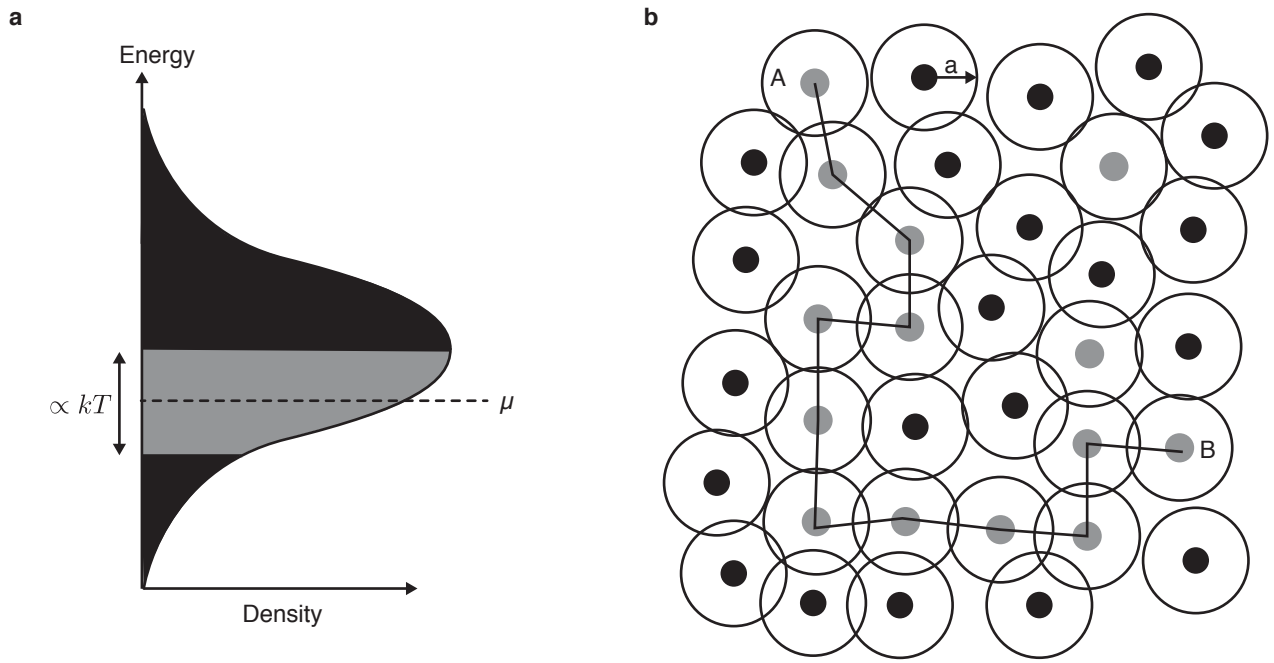

Figure 3.3: Percolation network density depends on temperature and chemical potential. **a**, An example density of states appropriate for a gaussian distribution of energy levels. Electron (or hole) transitions occur between states near the chemical potential, μ. The grey shaded region includes states that satisfy the bonding critereon. **b**, A percolation path through sites that satisfy the bonding critereon. Grey sites have energies in the grey shaded region of the density of states.

the overlap of wave functions in space but also the difference in energy. In figure 3.2 states that are close enough in energy to satisfy the bonding critereon are colored grey. Sites with ε_{ij}/kT too large to satisfy $\xi_{ij} \leq \xi$ are black.

Black and grey sites together contribute to the total density of states of the system, but only the grey sites (which satisfy the bonding critereon) contribute to the percolation network density. The total number of sites is constant, but the number of grey sites *vs* black sites changes with temperature and chemical potential. Figure 3.3 illustrates how the percolation network density (the density of grey sites) depends on the chemical potential and temperature.

The percolation network density depends directly on the localization length. The localization length can therefore be calculated if the network density is known at a specific

temperature and chemical potential. The percolation network density is a dimensionless number n_c defined by equation (3.2). The density of states distribution can be calculated from the superlattice structure and energetic disorder of the constiuent NCs. For a 2D superlattice, the density of states is given by equation (3.3) where N_{2D} is the 2D density of NC sites and σ is the standard deviation of energy levels caused by dispersion of NC diameters.

$$n_c = r_{\max}^2 \int_{-\varepsilon_{\max}}^{\varepsilon_{\max}} g\left(\varepsilon + \mu\right) d\varepsilon \tag{3.2}$$

$$g\left(\varepsilon\right) = \frac{N_{2D}}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-\varepsilon^2}{2\sigma^2}\right) \tag{3.3}$$

We are interested in n_c at temperature T_c, therefore we integrate over an energy range defined by the largest possible energetic transition at temperature T_c, *i.e.*, $\varepsilon_{\max} = kT_c\xi_c$. The characteristic length scale is the largest possible hopping distance, *i.e.*, $r_{\max} = \frac{a\xi_c}{2}$. Integrating the density of states about the chemical potential μ, we find a relationship between n_c, a, ξ_c, and μ given in equation (3.4). For a random network, the critical value of the network density was calculated to be 6.9 by Skal *et al*[11].

$$n_c = \frac{a^2 {\xi_c}^2 N_{2D}}{8} \left(\mathrm{erf}\left(\frac{\mu + kT_c\xi_c}{\sigma\sqrt{2}}\right) - erf\left(\frac{\mu - kT_c\xi_c}{\sigma\sqrt{2}}\right)\right) \tag{3.4}$$

Next we find μ corresponding to each measured gate voltage. We define zero chemical potential as the average $1S_e$ ($1S_h$) site energy. The expectation value of the nearest neighbor hopping energy $\langle\varepsilon_{\mathrm{NN}}\rangle$ depends on the position of the chemical potential. We numerically evaluated the relationship between nearest neighbor hopping energy and chemical potential using equations (3.5) to (3.7). Equation (3.6) defines the probability distribution of the energies ε_i and ε_j, with the same standard deviation as in equation (3.3). Equation (3.7) defines the transition energy $\varepsilon_{\mathrm{ij}}$ between sites *i,j* relative to the chemical potential μ. Equation (6)

was integrated numerically over the range $|\varepsilon| \leq 10\sigma$. The numerical result was approximated as a power law, given by equation (3.8) and shown in figure 3.8. Using measured nearest neighbor hopping energies $\varepsilon_{\mathrm{NN}}$, we thus calculated the position of the chemical potential for electrons and holes at all measured gate voltages.

$$\langle \varepsilon_{\mathrm{NN}} \rangle = \iint w(\varepsilon_i)\, w(\varepsilon_j)\, \varepsilon_{\mathrm{ij}} d\varepsilon_i d\varepsilon_j \tag{3.5}$$

$$w(\varepsilon) = \frac{1}{\sigma\sqrt{2}} \exp\left(\frac{-\varepsilon^2}{2\sigma^2}\right) \tag{3.6}$$

$$\varepsilon_{\mathrm{ij}} = \frac{1}{2}\left(|\varepsilon_i - \varepsilon_j| + |\varepsilon_i - \mu| + |\varepsilon_j - \mu|\right) \tag{3.7}$$

$$\langle \varepsilon_{\mathrm{NN}} \rangle \approx \sigma\left(1.36 + 1.2669\left(\frac{\mu}{\sigma}\right)^{1.3}\right) \tag{3.8}$$

$$\frac{\langle \varepsilon_{\mathrm{NN}} \rangle}{kT_c} = \xi_c - \frac{2r_{\mathrm{NN}}}{a} \tag{3.9}$$

We introduce equation (3.9) by realizing that at T_c, the hopping distance is approximately the nearest neighbor distance, and the hopping energy is approximately the nearest neighbor hopping energy $\langle \varepsilon_{\mathrm{NN}} \rangle$. Rearranging equation (3.8), we substitute for μ in equation (3.4) and solve equations (3.4) and (3.9) numerically to find the localization length a for each measured gate voltage.

We now estimate the uncertainty in the analysis of temperature dependent conductance by the percolation interpretation outlined above. Experimental sources of uncertainty include the transition temperature T_c, the distribution of NC energy levels σ, the nearest neighbor hopping energy $\langle \varepsilon_{\mathrm{NN}} \rangle$, and the nearest neighbor distance r_{NN}. The uncertainty in T_c is the

largest source of error on the calculated localization length. This error, on the order of 10 K, leads to an uncertainty in the localization length a, of up to 1 nm. Other parameters have a smaller significance.

The localization lengths estimated from conductance measurements by the above method are qualitatively consistent with an interpretation based only on the change in conductance with temperature in the variable range hopping (VRH) regime. In the case of Efros-Shklovskii variable range hopping, the range of values for the static dielectric constant that give localization lengths consistent with our calculated range of 5 nm to 10 nm is 30 to 19. This is reasonable given that the static dielectric constant was calculated to be 12 for PbSe NCs in a matrix of alkane ligands whereas the value for bulk PbSe is near 200[12, 13]. Therefore a value moderately higher than 12 is reasonable for an all inorganic PbSe NC solid.

3.2 Localized Transport: Background

3.2.1 Measurement of Transport in Disordered Materials

The previous section has highlighted the importance of the chemical potential in determining the mobility and localization length in disordered materials. Experimental control over the chemical potential can be accomplished by electrostatic or electrochemical potential. Electrochemical doping or gating fills localized electronic states by electron transfer from an electrolytic solution. Guyot-Sionnest and co-workers have used this method to measure the conductivity and mobility of the 1S and 1P states of PbSe and CdSe NC solids[14, 15, 16].

Alternatively, the chemical potential can be controlled by filling NC states via an electrostatic potential. This is employed using a device called a field-effect transistor (FET)[17]. A transistor is a three-terminal electronic device in which the current flow between two of

the terminals (called the source and drain) can be controlled, or gated, by the third terminal (called the gate). One specific physical realization of a transistor is the metal oxide semiconductor field effect transistor (MOSFET). In the MOSFET no current flows between the gate and the source or drain. The gate consists of a metal layer, a dielectric layer (commonly an oxide), and a semiconductor layer. The potential of the metal layer is the gate voltage which controls the potential difference across a capacitor formed by the metal-oxide-semiconductor stack. The semiconductor forms the second electrode of the capacitor. By changing the gate potential, the charge density in the semiconductor at the oxide-semiconductor interface can be controlled. By changing the charge density in the semiconductor the chemical potential in the semiconductor can be controlled.

Both electrochemcial gating and the FET are limited in the maximum chemical potential shift. In the case of electrochemical gating the electrochemical window of the electrolyte solvent limits the maximum potential that can be applied. For the MOSFET the gate potential is limited by the dielectric strength of the oxide. The most commonly used oxide material, SiO_2 has a dielectric strength on the order of $1 \times 10^7\,\mathrm{MV\,cm^{-1}}$[18]. From this perspective, a thicker oxide layer would be advantageous to achieve higher electrostatic potential.

The thickness of the oxide also comes into play in terms of the capacitance and therefore the maximum carrier density achievable in the semiconductor layer. The formula for the capacitance of a parallel plate capacitor is given by $C = \epsilon_0 \epsilon_r A/d$ where ϵ_0 is the permittivity of free space, ϵ_r is the relative permittivity of the dielectric, A is the interfacial area of the plate-dielectric interface, and d is the thickness of the dielectric. Therefore a thin dielectric is required to achieve high carrier density and chemical potential in the semiconductor of a MOSFET device.

The dielectric layer thickness is therefore a compromise between stability and capacitance. The breakdown voltage of SiO_2 is not a constant but decreases with decreasing thickness.

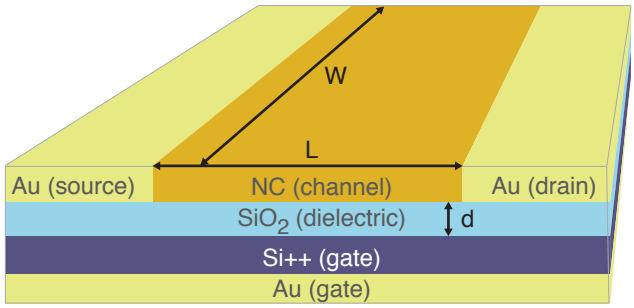

Figure 3.4: Schematic of a MOSFET.

This is due to the breakdown mechanism which involves ionic impurities in the oxide and positive feedback in the breakdown process due to thermal energy released as current flows through the oxide. In practice, oxide layers of a few hundred nm provide enough stability to protect against breakdown from electrostatic discharge while allowing sufficient carrier density to be achieved with gate voltages of a few tens of volts.

The chemical potential of a NC solid can be controlled using the MOSFET device by depositing the NC solid on the oxide as the semiconductor layer. Figure 3.4 shows an a schematic of the particular device architecture commonly used to fabricate a NC FET. This gate is called a back gate because it is formed on the back of the silicon wafer during fabrication. Most of the 500 µm thickness of the wafer is silicon which has been highly doped such that the conductivity is similar to a metal ($< 0.002\,\Omega\,\mathrm{cm}$). This serves as the metal layer in the MOSFET architecture. The back of the silicon wafer is coated with a metal to facilitate connection to a measurement device and to prevent oxidation. The top of the wafer is oxidized to form the SiO_2 dielectric. This method ensures a low-defect interface between the gate electrode and the dielectric and easily controlled dielectric thickness. Because of the high purity of the silicon, the oxide can be of very high quality with few ionic impurities.

The source and drain electrodes are fabricated by lithographic patterning of a metal layer on the surface of the oxide. In this way the dimensions of the NC solid can be easily defined. Since the NC layer is deposited on top of the source and drain electrodes, this configuration

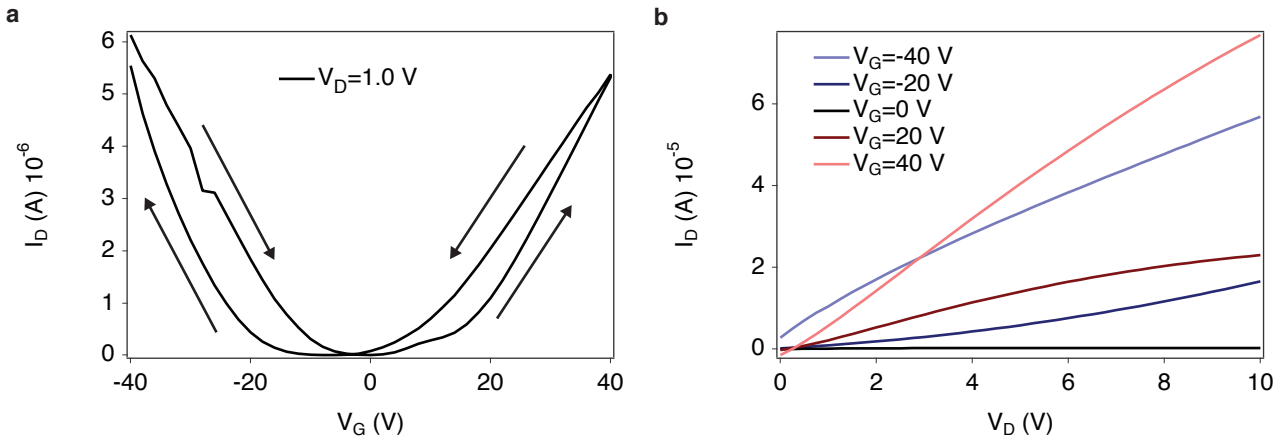

Figure 3.5: Illustration of a FET device. **a**, An example transfer curve, also called a V_G - I_D plot. The current between the source and drain is plot against the gate potential. **b**, An example plot of characteristic curves, also called a V_D - I_D plot. The current between the source and drain is plot *vs* the source-drain voltage at several gate potentials.

is referred to as a bottom contact device. By depositing the NC solid as the last step in the process, exposure of the NC solid to oxygen can be avoided.

In figure 3.5 we show an example measurement of the current flow between source and drain electrodes as a function of the potential of the gate electrode. The source electrode is commonly grounded (0 V) and therefore the source to drain potential V_{SD} and source to drain current I_{SD} are commonly abbreviated as V_D and I_D, respectively. The change in current with gate voltage is called a transfer curve. The transfer curve can be used to determine the mobility of electrons or holes in the semiconductor. As the gate voltage increases, the chemical potential for electrons near the oxide-semiconductor interface decreases, therefore increasing the density of electrons. As the gate voltage decreases, the chemical potential for holes decreases, therefore increasing the density of holes. Therefore if the current increases with increasing gate voltage, the current is carried by electrons. If the current increases as the gate voltage decreases, then the current is carried by holes. If the transfer curve shows both positive and negative slope, meaning current is carried by electrons for some range of gate potential and by holes for some range of gate potential, then the device is called ambipolar.

The previous description of the transfer curve assumes that the mobility of electrons or holes is independent of chemical potential. This assumption is true for non-degenerately doped semiconductors, but not necessarily for disordered materials. In spite of this caveat, the transfer curve is generally used to calculate mobility for NC solids. To derive the relationship between mobility and the transfer curve, we start with the so-called drift-diffusion equation (3.10), which is Fick's law for electrons where n is the electron density, μ_n is the mobility, E_y the electric field between source and drain, D_n the diffusivity, and q is the electron charge. The coordinate system is defined such that the x-axis is normal to the semiconductor-oxide interface, the y-axis is parallel to the direction of current flow between source and drain, and the z-axis is parallel to the transistor channel.

$$J_N = q\mu_n nE + qD_N \nabla n \tag{3.10}$$

Assuming the contact between source-drain electrodes and the semiconductor is ohmic and the carrier density in the semiconductor is large, the diffusion component can be neglected. The current between source and drain is found by integrating the current density through a cross-sectional plane (the x-z plane) and given by equation (3.11) where J_{Ny} is the current density in the y-direction, and W is the width of the transistor channel. Given that the current density is constant in z we can substitute equation (3.10) for J_{Ny}. Now we make an important approximation, that the electron density can be modeled as a 2D sheet at the oxide-semiconductor interface. Therefore the integral of the electron density moving away from the interface is simply the density at the interface Q_N.

$$\begin{aligned} I_D &= -\int\int J_{Ny} dx dz = -W \int J_{Ny} dx \\ &= (-WE)\left(-q \int \mu_n n(x,y) dx\right) \\ &= -W\mu_n Q_N E \end{aligned} \tag{3.11}$$

As mentioned above, the gate electrode, oxide, and semiconductor form a capacitor where the charge density at the oxide-semiconductor interface is Q_N. Therefore the charge density at the interface is given by equation (3.12) where V_G is the gate voltage, V_T is the threshold voltage, and ϕ is the potential in the semiconductor due to the source-drain potential $E = d\phi/dy$. The threshold voltage is the potential required to fill electronic states near the interface that do not contribute to current. In silicon based MOSFETs the threshold voltage is the voltage required to accumulate a density of minority charges equal to the majority carrier density. For disordered materials, the threshold voltage is due to immobile charges (traps) at the oxide surface or in the semiconductor itself.

$$Q_N(y) = -\frac{\epsilon_0 \epsilon_r}{d} \left(V_G - V_T - \phi(y) \right) \tag{3.12}$$

Substituting $d\phi/dy$ for E in equation (3.11), rearranging and integrating we get equation (3.13).

$$\begin{aligned} \int_0^L I_D dy = I_D L &= -W \int_0^{V_D} \mu_n Q_N d\phi \\ I_D &= -\frac{W \mu_n}{L} \int_0^{V_D} Q_N d\phi \end{aligned} \tag{3.13}$$

Substituting equation (3.12) into equation (3.13) and integrating we arrive at the so-called square law equation (3.14) where we have replaced $\epsilon_0 \epsilon_r / d$ with C_o, the oxide capacitance per unit area. To determine the mobility from a transfer curve, we simply take the derivative of I_D with respect to V_G and rearrange to yield equation (3.15) where g_m represents the slope of the transfer curve dI_D/dV_G also called the transconductance.

$$I_D = \frac{W \mu_n C_o}{L} \left(\left(V_G - V_T \right) V_D - \frac{V_D^2}{2} \right) \tag{3.14}$$

$$\frac{dI_D}{dV_G} = \frac{W\mu_n C_o V_D}{L}$$
$$\mu_n = \frac{g_m L}{W C_o V_D} \tag{3.15}$$

Equation (3.15) is valid only in the so-called linear regime, where $V_D < V_D^{sat}$ and the relationship between I_D and V_D is linear. Saturation occurs when the drain voltage is great enough such that the charge $Q_N(y_0) = 0$ at some point y_0. This occurs, according to equation (3.12), when the potential at some point in the channel equals the gate potential $V_G = \phi(y_0)$. This occurs at the drain-semiconductor interface when $V_D = V_G$ and is called pinch-off. For $V_D > V_D^{sat}$, I_D is constant (as long as L is not too small).

Figure 3.5 actually shows two transfer curves, one measured while sweeping V_G from $-40\,\mathrm{V}$ to $40\,\mathrm{V}$ and another while sweeping from $40\,\mathrm{V}$ to $-40\,\mathrm{V}$. The difference between the two transfer curves is called hysteresis and contains information about trapped charge. The area between the two curves is proportional to the immobile charge at the oxide-semiconductor interface. Fixed charge can accumulate in trap states that exist in the semiconductor or at the oxide surface. This charge decreases the effective gate potential by an amount called the threshold voltage V_T. The amount of fixed charge depends on V_G and therefore V_T also depends on V_G. There is a kinetic component to hysteresis as well because the average time for charge to be trapped or released may be much less than, similar to, or much greater than the time to measure a transfer curve.

3.3 Transport in Epitaxially Connected PbSe Superlattices

Using the method described in section 2.3, we fabricated epitaxially connected PbSe NC superlattices with long range order as described in section 2.4. Superlattices were deposited onto pre-patterned FET substrates then encapsulated before measurement (see appendix A).

Transfer curves (current *vs* gate voltage) in figure 3.6 show ambipolar transport. We find that electron and hole transport are thermally activated (hopping) with conductance decreasing exponentially with decreasing temperature, a clear sign of carrier localization. Electronic coupling through epitaxial connections combined with translational order should have a large effect on carrier localization. We therefore sought to estimate the electron and hole localization lengths and compare to theoretical prediction.

To solve for the localization length we use the method detailed in section 3.1. Briefly, we find the temperature at which transport transitions from NNH to VRH, we use this temperature to find the percolation network density and in turn the localization length. A plot of $\ln(G)$ vs $\ln(T^{-1})$ in figure 3.6 shows that a single value of p fails to describe the data over the full temperature range. Instead, the data suggests that there is a transition from NNH to VRH.

Conductance at low temperature deviates from Arrhenius behavior as expected for VRH. The transition temperature between NNH and VRH can be determined by fitting two linear regimes to $\ln(G)$ vs $\ln(T^{-1})$ as in figure 3.6. Figure 3.7 clearly shows the transition temperature T_c and nearest neighbor hopping energy ε_{NN} change with applied gate voltage V_G. Assuming the chemical potential is less than the peak density of states, increasing gate bias moves the chemical potential to a region of greater density of states. The average hopping energy decreases as the density of states at the chemical potential increases (see equation (3.5)). This is the physical reason for the relation between $\varepsilon_{\mathrm{NN}}$ and V_G. Below, we will estimate the localization length as a function of V_G from the transition temperature T_c.

From TEM image analysis and X-ray scattering we measure the NC diameter to be $(6.1 \pm 0.3)\,\mathrm{nm}$. We used the empirical relation by Moreels et. al.[19]: $E_0 = 0.278 + (0.016d^2 + 0.209d + 0.45)^{-1}$ where E_0 is the first exciton energy and d is the NC diameter in nm, to determine the energy levels. This gives energies of 732 meV and 687 meV for 5.8 nm and 6.4 nm diameters respectively, at one standard deviation from the mean diameter. The

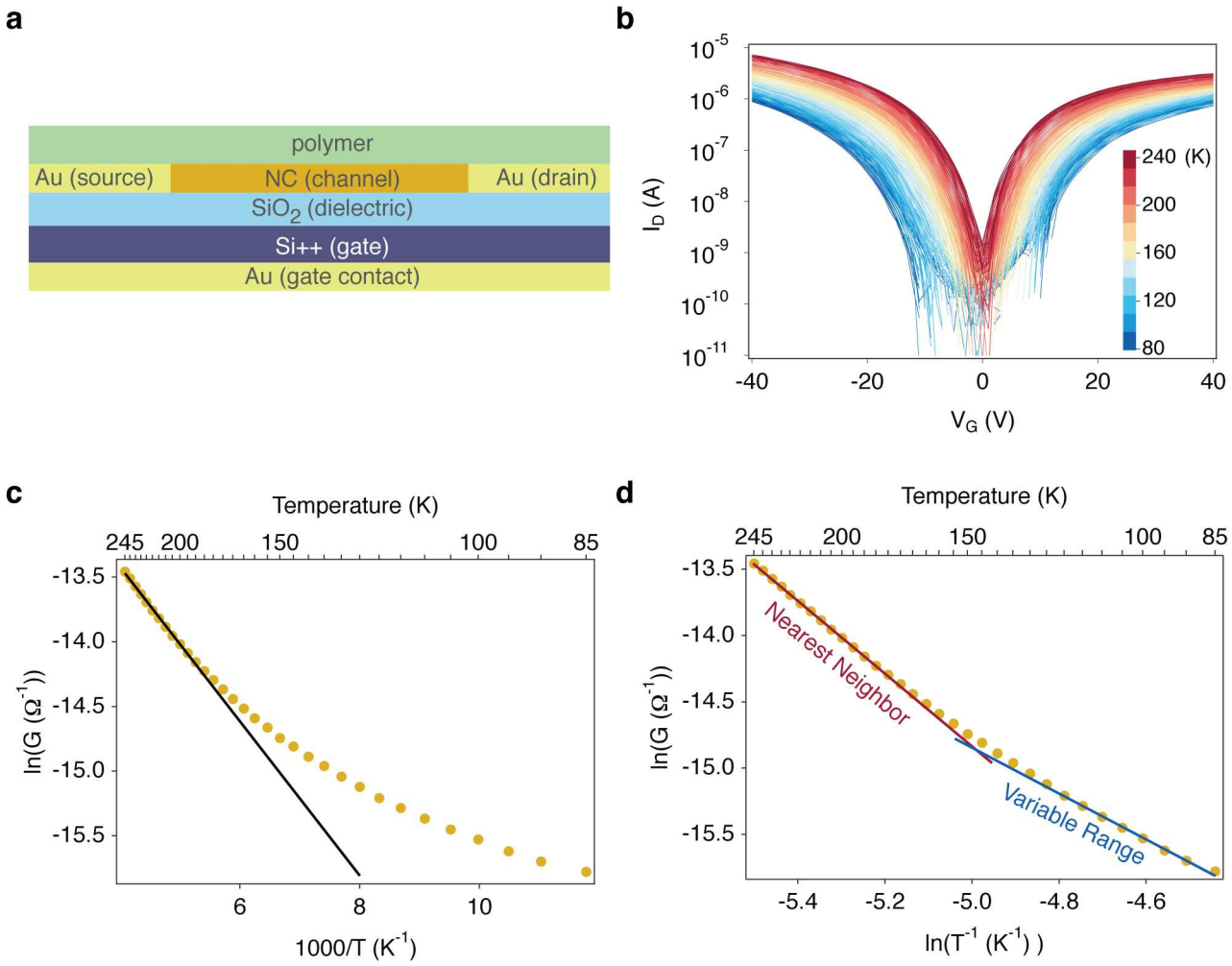

Figure 3.6: Charge transport measurement by FET. **a**, Cross-sectional schematic of the FET. The gate is formed by highly doped silicon (Si++) with a 200 nm dielectric layer of SiO_2. The source and drain electrodes were patterned on the SiO_2 to form a channel 100 mm x 3 mm. A layer of photo-cured polymer served as a barrier between the NC layer and ambient water and oxygen. **b**, Transfer curves show ambipolar transport with electron current at positive gate voltage (VG) and hole current at negative gate voltage. The source-drain bias was 1 V. The conductance was measured every 5 K from 85 K to 245 K. **c**, An Arrhenius plot of electron conductance at a gate bias of 22 V with a source-drain bias of 1 V. The solid line is a linear fit to the higher temperature data. **d**, A log-log plot of the data in panel **c**. Straight lines illustrate that the data is not a single power law over this temperature range.

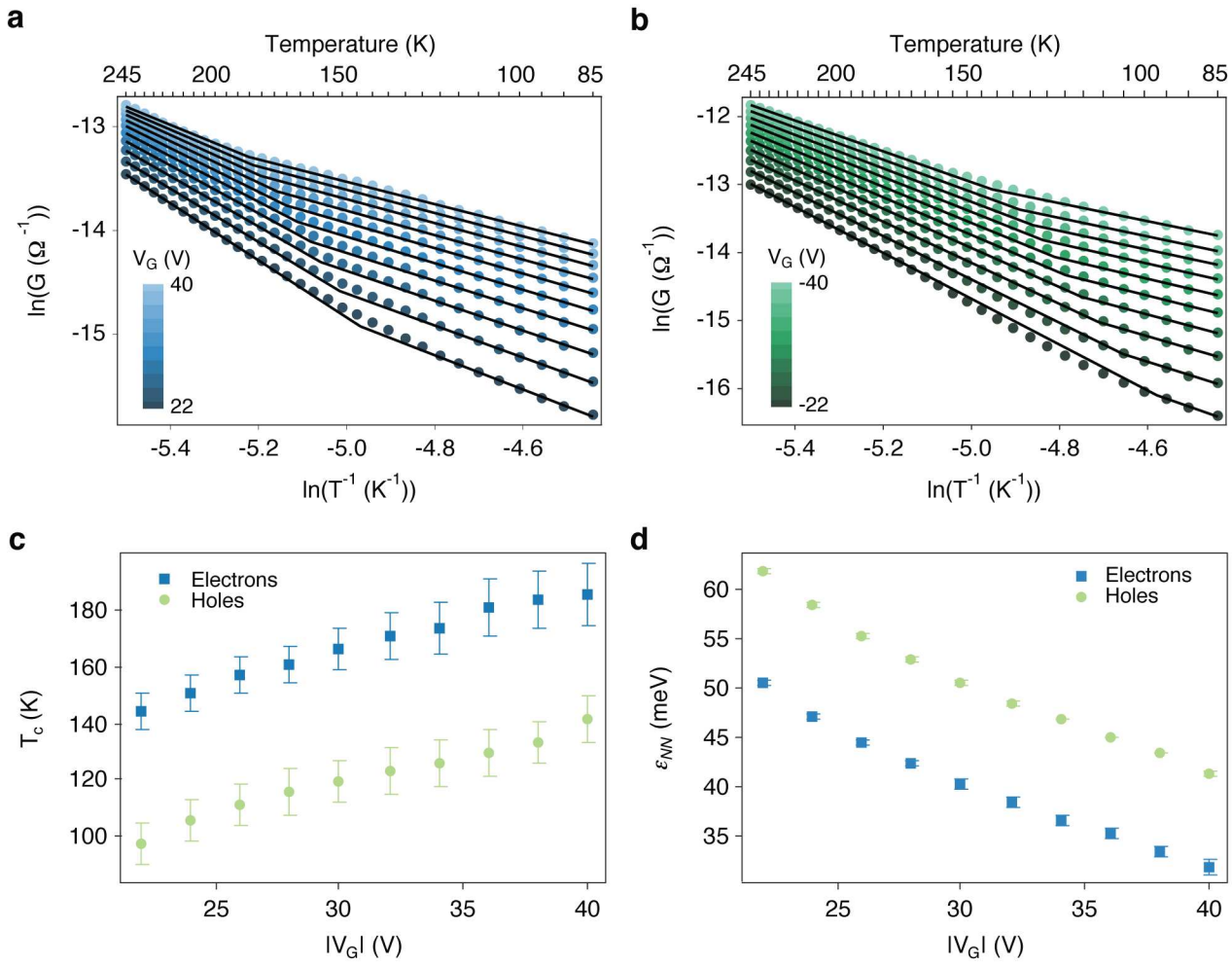

Figure 3.7: Hopping behavior of electrons and holes modulated by gate voltage. **a,b** Conductance vs. inverse temperature on logarithmic scales for **a**, electrons and **b**, holes. Points are measured data, lines are linear fits at high and low temperature. **c**, Transition temperatures calculated from the intersection of linear fits of $ln(G)$ *vs* $ln(T^{-1})$. Error bars were calculated by adding the standard errors of the high and low temperature linear regressions. **d**, Nearest neighbor hopping energies for electrons and holes, calculated from the slope of $ln(G)$ *vs* T^{-1} above T_c. Error bars represent the standard error of the slope from the linear regression.

difference of 45 meV is the difference between a $1S_e$ level one standard deviation above the mean $1S_e$ level and a $1S_h$ level one standard deviation below the mean $1S_h$ level. Therefore the standard deviation of a $1S_e$ or a $1S_h$ energy level is $45/4 = 11$ meV. The empirical relation between diameter and energy given by Allan and Delerue[20] gives a similar value of 13 meV.

The distribution of isolated NC energy levels is convolved with the distribution of coupling energies to give the distribution of energy levels in the superlattice. Kalesaki *et al* showed that the coupling energy of epitaxially connected PbSe NCs depends on the cross-sectional area of the connection. The coupling energy was calculated using an atomistic model by the tight-binding method for a range of connection sizes and NC diameters[2].

For 6.1 nm diameter NCs joined by an epitaxial connection nine PbSe bonds wide (2.8 nm), the coupling energy is 12 meV[2]. The distribution of epitaxial connection widths is gaussian with a standard deviation of three PbSe bonds (0.92 nm). If we linearly extrapolate the coupling energy over this range, the standard deviation is 4 meV. Convolving this distribution with the isolated NC energy level distribution gives a total standard deviation of $\sqrt{11^2 + 4^2} = 11.7\,\text{meV}$.

Figure 3.8 shows the localization lengths calculated using the measured distribution of energy levels. We find that charge carriers are localized to just a few NC diameters at a gate bias of ± 40 V. We measure mobilities of $0.54\,\text{cm}^2/\text{V s}$ for holes and $0.2\,\text{cm}^2/\text{V s}$ for electrons at 245 K. Despite relatively good mobilities compared to ligand exchanged quantum dot solids[21, 22, 23, 24], the localization length reveals that transport is still far from band-like.

We note that Coulomb blockade should not be an important factor in the measured temperature range. Coulomb blockade could play an important role if the charging energy is greater than the thermal energy. We can estimate the charging energy as $E_C = e^2/(C_g + 4C)$[25]. The self-capacitance is given by $C_g = 4\pi\varepsilon_{0r}$, where ε is the dielectric constant of the PbSe superlattice and r is the NC radius. The coupling capacitance between NCs is given by $C = 2\pi\varepsilon_{0r} ln\left((2r + s)/s\right)$ where s is the length of the tunnel barrier. We estimate $s = 0.3$ nm by the difference between the superlattice constant 6.4 nm and the average NC diameter 6.1 nm. The factor of 4 accounts for coupling between the 4 nearest neighbors in a2D square superlattice. We use the value of 12 found by Luther *et al*[12] for ligand passivated NCs as a lower bound for the effective dielectric constant in a PbSe NC solid.

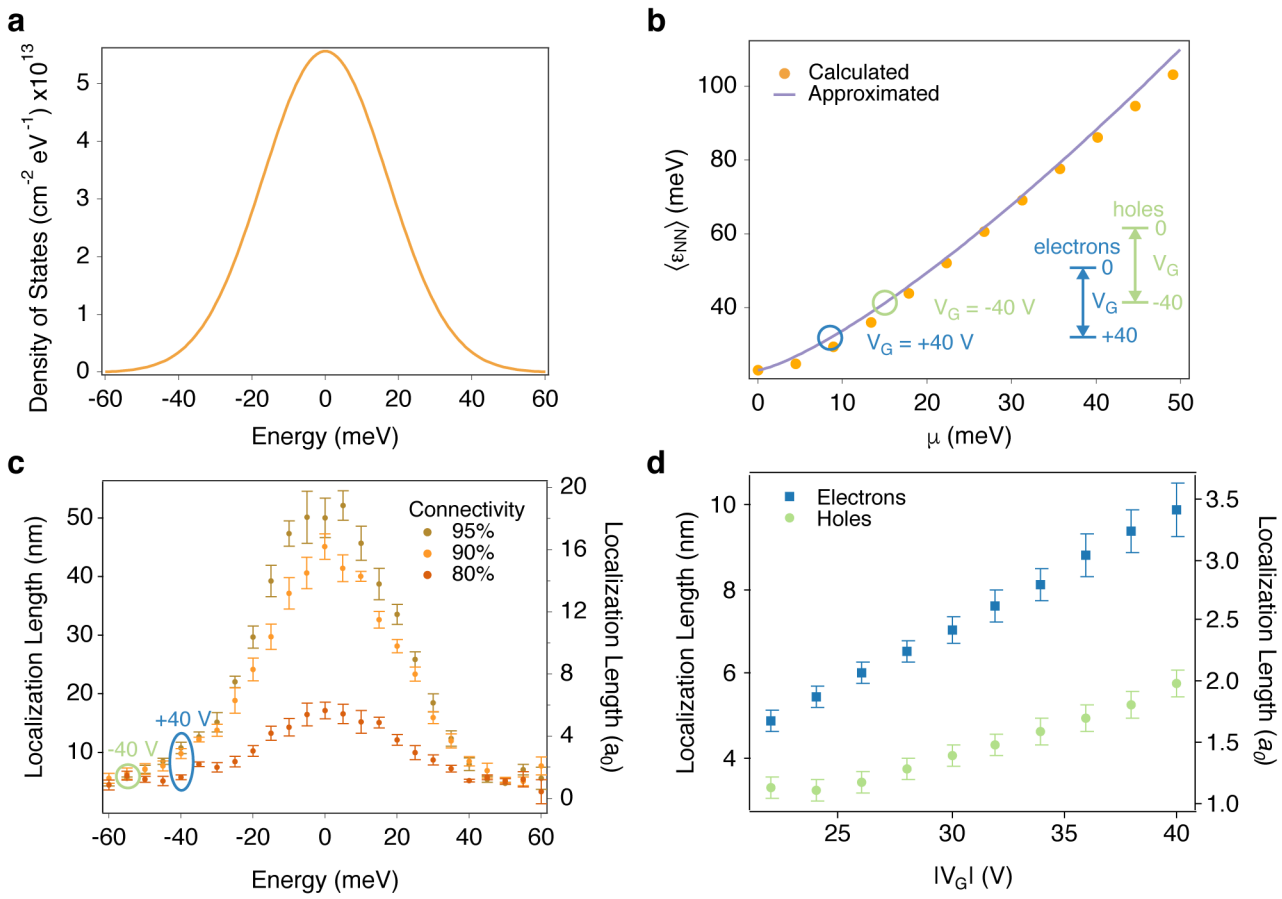

Figure 3.8: Extent of delocalization in a disordered superlattice. **a**,Model density of $1S_e$ or $1S_h$ states based on experimental measurements of the superlattice structure. **b**,Relationship between nearest neighbor hopping energy and chemical potential for the density of states model in panel **a**. Points were calculated numerically, the solid line is a power law approximation. Inset arrows show the ranges of hopping energies measured for electrons and holes at gate voltages between 0 and ±40 V. Open circles mark the measured hopping energies of electrons (holes) for a gate bias of 40 V (−40 V). **c**,Theoretical localization length of electrons (holes) in $1S_e$ ($1S_h$) states for a range of connectivity values. Zero energy refers to the average $1S_e$ or $1S_h$ energy level. The lowest energy states accessible to electrons (holes) at a gate bias of 40 V (−40 V) are indicated. Error bars represent the standard deviation of multiple numerical calculations using the Monte Carlo method. Localization lengths are given in nanometers and in by dimensionless units of a_0, where a_0 represents the localization length of a charge carrier in a completely isolated NC, equal to the NC radius. **d**,The localization lengths of electrons and holes calculated from the measured temperature dependence of the conductance. Error bars reflect the uncertainty of the transition temperature.

This gives $E_C = 5.5\,\text{meV}$, which is smaller than the thermal energy at 85 K, 7.3 meV. If we use a more realistic estimate of 20 for the dielectric constant of an all inorganic PbSe NC array, the charging energy decreases to 3.4 meV. Therefore we do not consider Coulomb blockade to be an important factor in this temperature range.

In this work we use 2D models for charge transport and electronic structure. Although we estimate the average thickness of the superlattice to be less than 10 NC layers (64 nm), we assume that charge transport is confined to the first NC layer. This is due to the electric field created by the gate voltage. Because transport is ambipolar, we assume the Fermi level is near mid-gap and any doping caused by mid-gap states is small. The transistor therefore operates in accumulation mode. We estimate the depth of the charge accumulation layer away from the oxide interface by[26]: $w = L_D \left(2\left[exp(-U) + exp(U) - 2\right]^{1/2} / \left(exp(U) - exp(-U)\right)\right)$. The Debye screening length is given by $L_D = \left[\varepsilon\varepsilon_S \mathrm{kT} / \left(2q^2 n_i\right)\right]^{\frac{1}{2}}$ where ε is the vacuum permittivity, ε_S is the relative dielectric constant of the semiconductor, k is the Boltzmann constant, T is the temperature, q is the charge of the electron, and n_i is the intrinsic doping concentration. The potential at the oxide-semiconductor interface, U is calculated from the gate voltage by: $V_G = kT/q \left(U + \varepsilon_S x_o \left[exp(-U) + exp(U) - 2\right]^{1/2} / \left(\varepsilon_o L_D\right)\right)$ where x_o is the oxide thickness (200 nm) and ε_o is the relative dielectric constant of the oxide (3.9 for SiO_2). Using conservative values for gate voltage (22 V), dielectric constant (20), and the intrinsic carrier concentration ($3.8 \times 10^{12}\,\text{cm}^{-3}$, or one electron per 10^6 NCs), the effective charge layer is 2.5 nm. Only if the relative dielectric constant of the superlattice exceeded 50 would the charge layer exceed the size of one NC layer.

3.4 Doping and Transport in PbSe Nanocrystal Solids

Because of the large surface-to-volume ratio and strong interaction of the wave function with the surface, NC solids are greatly affected by surface dopants. Dopants may be introduced

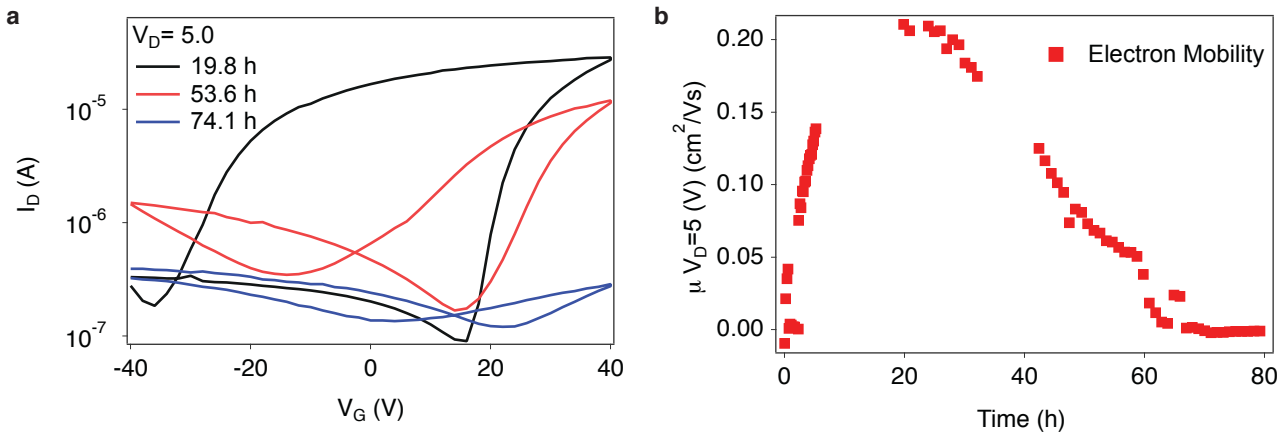

Figure 3.9: Time dependent electron mobility of an epitaxially connected PbSe superlattice. **a**, Three transfer curves show changing hole and electron currents over time. **b**, The electron mobility (determined as the transconductance between 0 V and 40 V) *vs* time.

intentionally to produce n- or p-type solids or unintentionally by oxidation. Both organic and inorganic species have been used to dope or prevent unintentional doping of NC solids, both during synthesis[27, 28] and post-synthesis[29, 30, 31, 32, 33, 34, 35, 36, 37, 38].

3.4.1 Oxidation and Halide Passivation

In the case of epitaxially connected PbSe NCs, we find the electron and hole mobilities change several-fold over the course of hours. An initial increase in electron mobility is followed by a decrease at a slower rate with a concurrent increase in hole mobility. Figure 3.9 shows an example of this response.

It has been shown that oxidation of PbSe occurs rapidly in ambient atmosphere at room temperature and that increased hole doping occurs on exposure to molecular oxygen[39, 40]. Voznyy *et al* proposed that doping levels in PbS NCs can be accounted for by the sum of oxidation states of the constituent atoms, with oxidation of Pb atoms leading to p-type behavior[41]. Figure 3.10 shows the effect of exposure to ambient atmosphere on electron transport in an epitaxially connected PbSe NC FET. After 5 min exposure electron

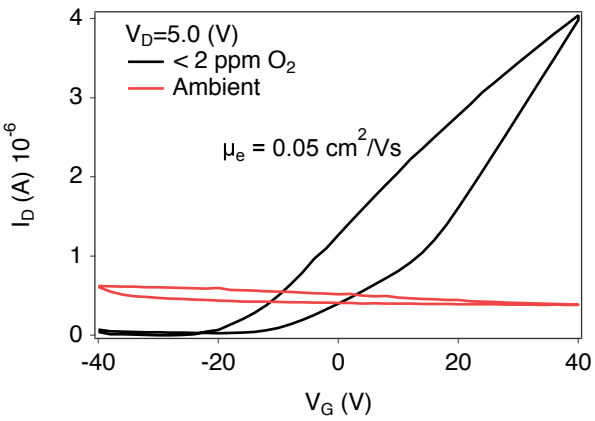

Figure 3.10: Effect of ambient air on electron mobility in epitaxially connected PbSe NCs. An initially n-type sample with electron mobility of $0.05\,\mathrm{cm}^2/\mathrm{Vs}$ becomes heavily p-doped after exposure to ambient for 5 min.

transport is completely quenched. This is likely the cause of slow p-doping in an environment of $<$ 2 ppm O_2.

Several reactants have been used to decrease oxidation of surface Pb atoms. These can be classified into three main groups: halides, oxides, and acids. The as synthesized surface species is an acid, oleic acid[42], which can be replaced by other acidic ligands such as thiols[22] or other acids[43, 28]. While the Pb-thiol bond such as with ethanedithiol is not stable to oxidation[22], Woo *et al* showed that phosphonic acid ligands do passivate PbSe in the presence of molecular oxygen[28]. Metal-oxide passivation is primarily a physical barrier against molecular oxygen, though Liu *et al* have shown that exposure of the NC surface to the metal reactant has a significant effect on transport[24, 44].

Surface bound halides have been shown to stabilize PbSe NCs against oxidation[27, 32, 36, 45]. The halide precursor can be molecular[36] or an organic salt[27, 32]. Woo *et al* showed that all halides except fluorine bind to surface Pb atoms and prevent the formation of Pb-oxide[27]. Besides providing passivation against oxidation halides also act as electron donors in agreement with the oxidation state theory[41]. Figure 3.11 shows the effect of exposure of an epitaxially connected PbSe NC superlattice to a $0.1\,\mathrm{mg\,mL}^{-1}$ solution of

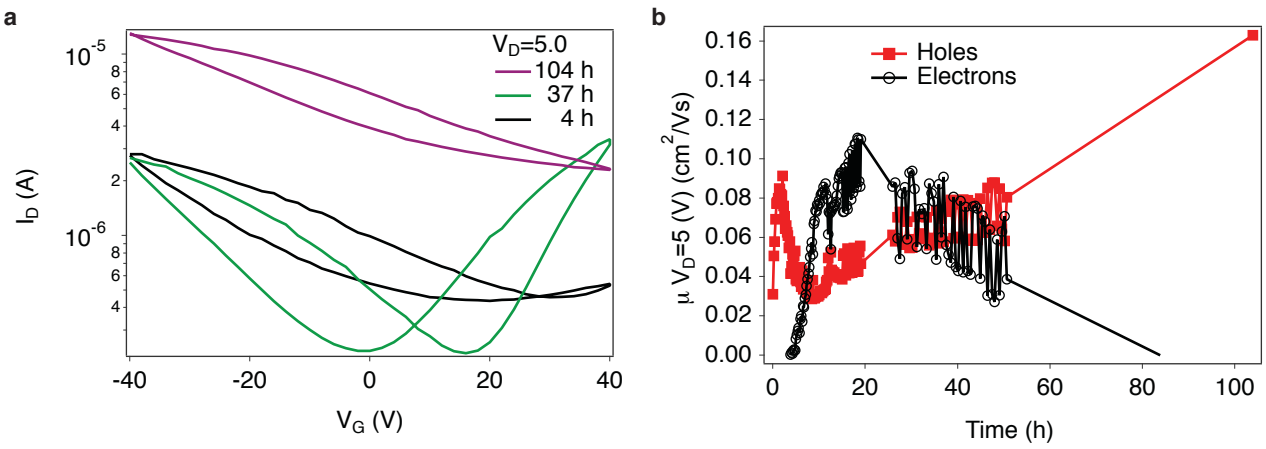

Figure 3.11: Effect of TBAI on electron mobility of an epitaxially connected PbSe NC superlattice. **a**, Transfer curves showing a transition of p-type to ambipolar and back to p-type. **b**, Mobility *vs* time showing an initial increase in electron mobility concurrent with a decrease in hole mobility, followed by the opposite trend.

tetrabutylammonium iodide (TBAI) in methanol (MeOH).

As in the unpassivated case electron mobility increases, peaks, and decreases with time while hole current follows an opposite trend after exposure to TBAI. The initial increase in electron mobility with time can be explained by diffusion and binding of iodide to surface Pb atoms. The subsequent decrease in electron mobility suggests oxidation due to incomplete passivation.

An alternative to passivation of epitaxially connected NC superlattices is passivation of NCs in solution followed by epitaxial connection. Following the method of Woo *et al*[27], we synthesized PbSe NCs with ammonium chloride (NH_4Cl). Figure 3.12 shows that the lowest energy excitonic absorption feature of PbSe NCs synthesized with NH_4Cl blueshifts at a reduced rate compared to those without NH_4Cl.

As shown in figure 3.13, passivation of PbSe NCs by NH_4Cl prior to epitaxial connection did not prevent p-type doping by oxidation.

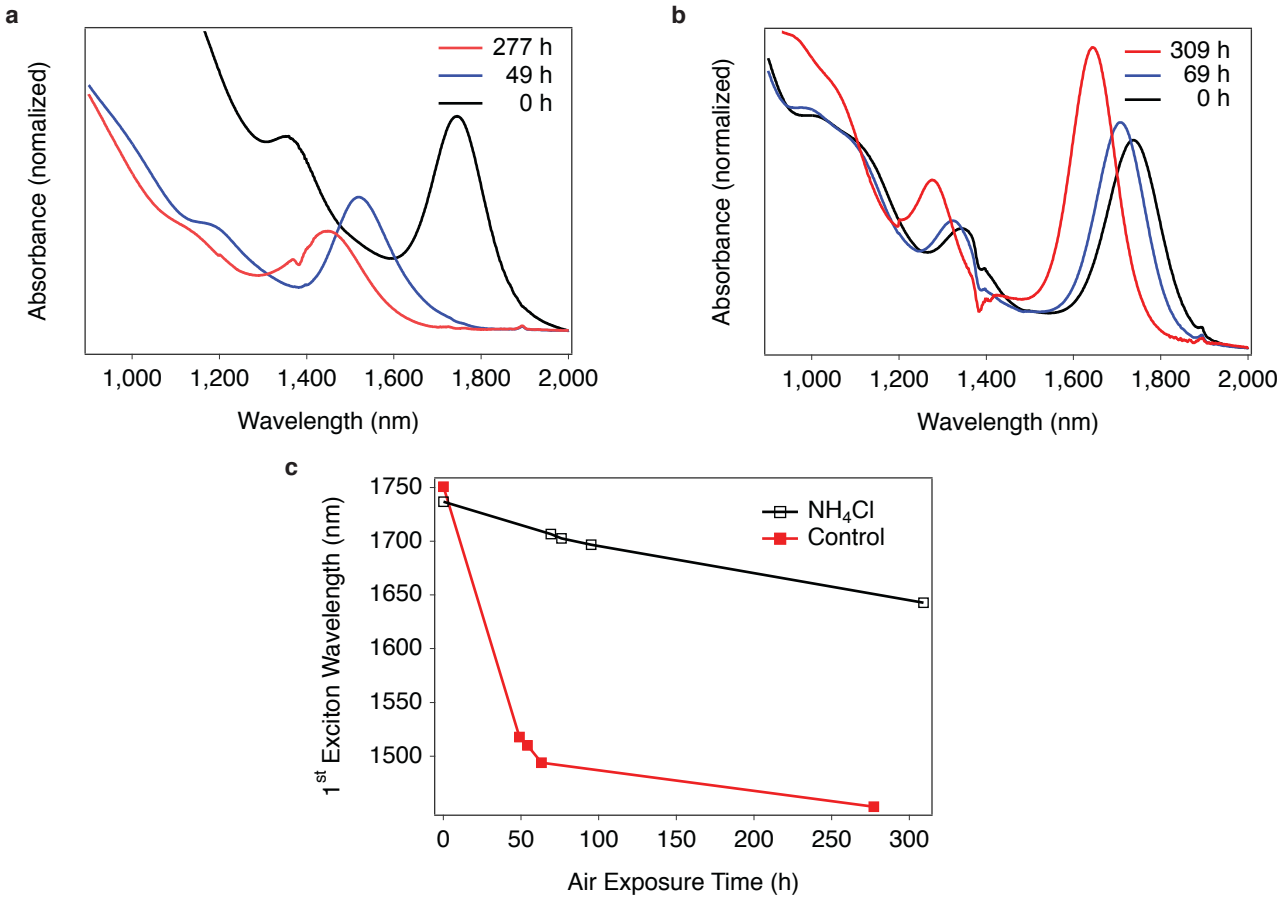

Figure 3.12: Effect of oxidation on absorbance spectra of PbSe NC suspensions in C_2Cl_4 in ambient air. Times indicate number of hours the solution was in ambient air. **a**, Control sample synthesized without NH_4Cl. **b**, Sample synthesized with NH_4Cl. **c**, Change in the lowest excitonic peak wavelength with time.

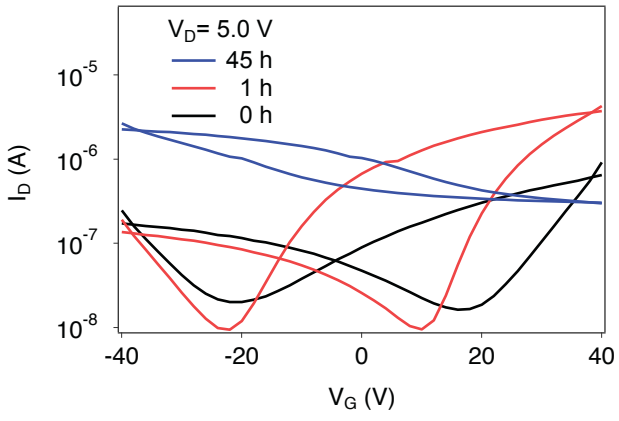

Figure 3.13: Effect of oxidation on transport of epitaxially connected NH_4Cl passivated PbSe NCs. An initial increase in electron mobility following epitaxial connection is followed by a decrease and transition to p-type behavior.

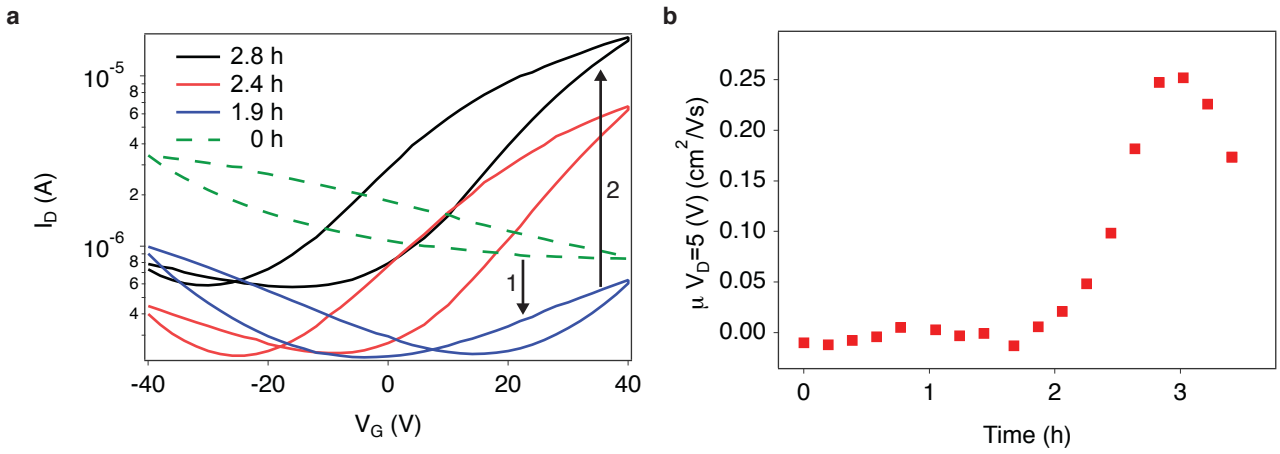

Figure 3.14: Effect of Pb-CH_3COOH^- on electron mobility in epitaxially connected PbSe NCs. **a**, Transfer curves following Pb-CH_3COOH^- exposure. **b**, Electron mobility *vs* time after exposure to Pb-CH_3COOH^-.

3.4.2 Doping by Stoichiometry

Doping of PbSe NCs can be controlled by adjusting stoichiometry, consistent with the simple theory of counting the net oxidation states of all atoms in the NC[41]. DFT calculations have shown that stoichiometric and ligand-free PbS NCs are undoped (no electron states within the band gap) regardless of NC shape[46]. Off-stoichiometric NCs exhibit mid-gap states, as do those with charge contributed from excess ligands. In agreement with Voznyy *et al*, the overall stoichiometry of the NC-ligand system determines doping by mid-gap states.

Experimental evidence for off-stoichiometric doping of PbSe NCs has been shown by thermal evaporation of Pb and Se and by solution-phase addition of the same[31, 30]. We have investigated doping in epitaxially connected PbSe NC superlattices by addition of Pb or Se species following epitaxy. Figure 3.14 shows that exposure to a 1 mM solution of lead acetate (Pb-CH_3COOH^-) in 2-methoxyethanol ($C_3H_8O_2$) for 5 min converts from p-doped to n-doped followed by oxidative p-doping. Similarly, p-type doping can be achieved by introducing excess Se, although at the concentration investigated (1 mM), the doping level is degenerate with a metallic response.

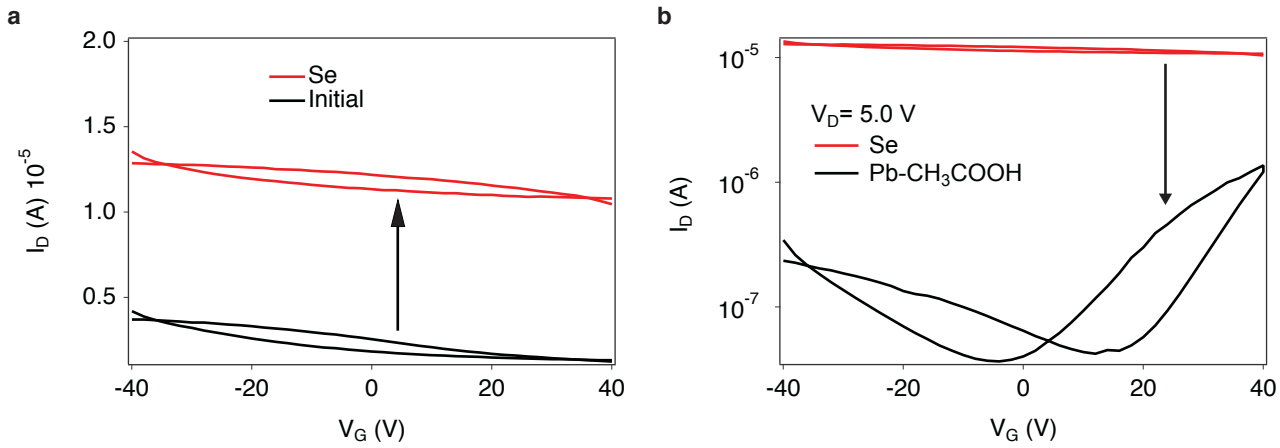

Figure 3.15: Doping epitaxially connected PbSe NCs by controlling stoichiometry. **a**, Exposure to a 1 mM solution of Se in OLA for 1 min increases hole doping. **b**, Subsequent exposure to a 1 mM solution of Pb-CH_3COOH^- in $C_3H_8O_2$ for 1 min contributes Pb as an electron donor, resulting in ambipolar behavior.

3.4.3 Passivation and Encapsulation

As shown in the previous section, the electronic properties of PbSe NC superlattices are not stable against oxidation in a controlled environment with < 2 ppm O_2 even with halide passivation by TBAI or NH_4Cl. However, stability is required for a reliable measurement of charge carrier localization length. The sample must be stable not only against oxidation, but also at the reduced pressures and temperatures required for temperature dependent conductivity measurement. Liu *et al* have shown that a physical barrier of metal oxide deposited by atomic layer deposition (ALD) is sufficient to stabilize the electronic properties of PbSe NCs[44, 24]. Similarly, we pursued a solution deposited physical barrier in the form of a photo-cured polymer.

Unlike epoxides which require curing at elevated temperatures that causes sintering of NCs, photo-cure polymers can be processed quickly at room temperature. This is important for stabilizing PbSe NC superlattices before oxidative doping occurs. A commonly used photoactive monomer is pentaerythritol-tetrakis(3-mercaptopropionate) (PETM). Absorption of a ultraviolet (UV) photon generates a thiol radical to initialize homopolymerization.

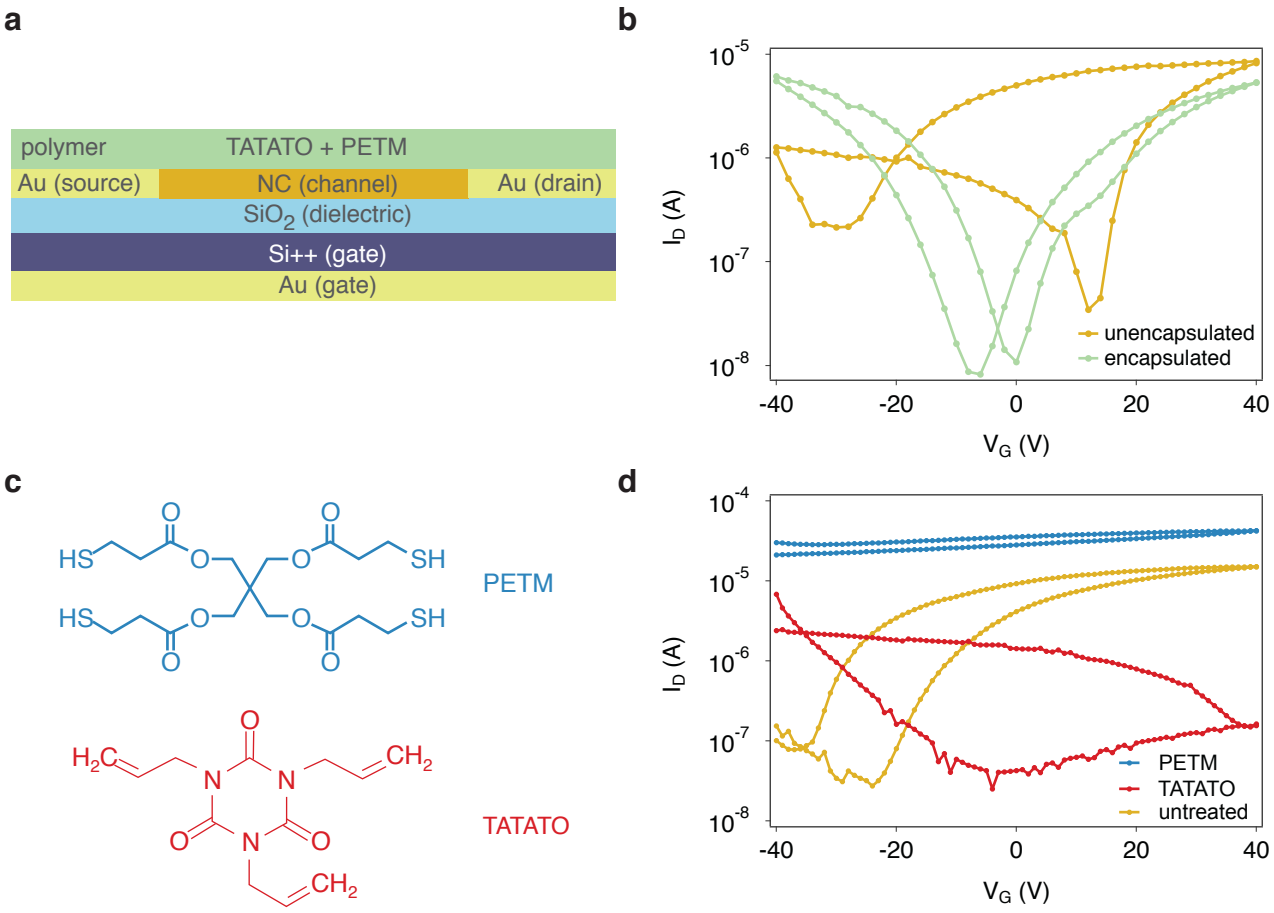

Figure 3.16: Effect of polymer encapsulation on electron and hole mobilities. **a**, Schematic cross-section of the device architecture, showing polymer encapsulation as the top layer. **b**, Transfer curves showing the change in electron and hole transconductance and hysteresis with encapsulation. **c**, Diagrams of the monomer PETM and cross-linker TATATO. **d**, Transconductance curves showing the change in electron and hole mobilities in the presence of PETM or TATATO.

However, addition of an allyl functionalized cross-linking agent such as 1,3,5-triallyl-1,3,5-triazine-2,4,6(1H,3H,5H)-trione (TATATO) improves uniformity and polymerization rate via thiol-ene click chemistry[47]. Diagrams of PETM and TATATO are shown in figure 3.16.

Figure 3.16 illustrates the primary function of the polymer as the top layer of the FET device architecture, as a physical barrier to molecular oxygen. Interestingly, we found that encapsulation not only formed a physical barrier to prevent oxidative hole doping, but also significantly reduced hysteresis and balanced electron and hole mobilities. We investigated

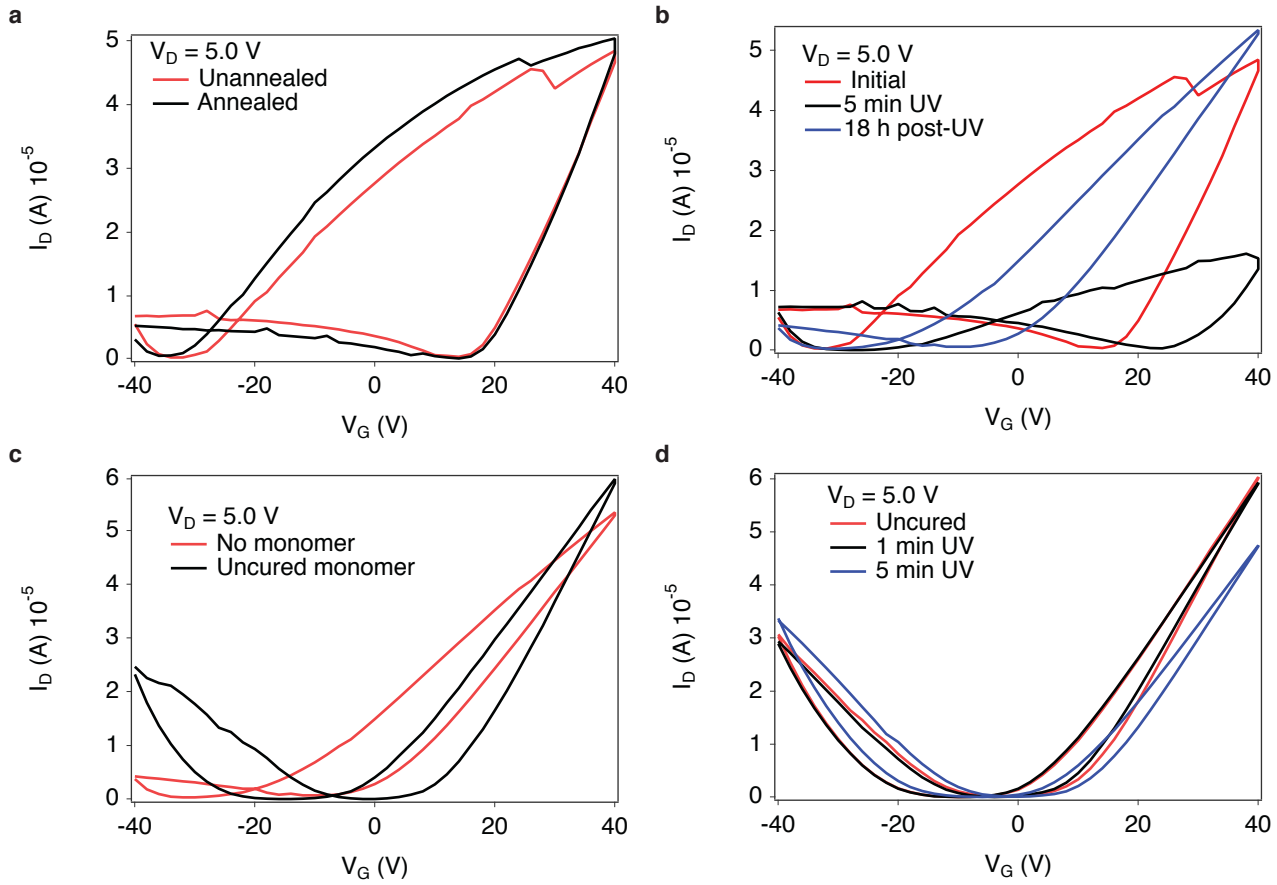

Figure 3.17: Effects of annealing, UV light, monomer, and polymer on electron and hole transport in epitaxially connected PbSe NCs. **a**, Comparison of transfer curves before and after annealing an untreated sample at 50 °C for 10 min in < 2 ppm O_2. **b**, Transfer curves after exposure to 254 nm UV light and 18 h post-exposure. **c**, Effect of deposition and annealing of PETM and TATATO. **d**, Transfer curves with unreacted monomer, after 1 min UV exposure and after 5 min UV exposure.

each step of the encapsulation process to determine the reason for reduced hysteresis and ambipolar transport. Details of the polymer deposition can be found in appendix A.4, but briefly the steps are: spin-cast a mixture of PETM and TATATO diluted with acetonitrile (ACN) as a solvent, remove the solvent by annealing at 50 °C, photo-polymerize with UV light. We performed control experiments to asses the influence of each processing step on hysteresis and change in electron and hole mobilities.

Annealing has been shown to increase the conductivity of NC solids[21, 48]. This increase can be the result of reduced interparticle distance, NC growth and sintering, or doping by

desorption of surface species. To remove excess solvent from the encapsulation layer before cross-linking, we annealed at 50 °C for 10 min. Figure 3.17 shows no change to electron or hole transconductance when annealed without the monomer solution. We therefore conclude the annealing step is not responsible for any change in hysteresis or mobility.

Exposure of NCs to UV light has been reported for photodoping of ZnO[49]. We do not observe doping of PbSe NCs on exposure to UV, but rather an initial reduction in electron mobility followed by recovery with decreased hysteresis of the transfer curves. In figure 3.17 we show that electron transconductance is reduced following 5 min of 254 nm UV light at approximately 3 cm distance from a 4 W source. After 18 h however, the electron transconductance returns to the pre-exposure value with a significant decrease in hysteresis. Further experiments are necessary to determine the mechanism by which UV exposure affects electron mobility and hysteresis. We postulate that photoexcited electron transfer to surface species could play a role.

Although UV exposure may be responsible for reduced hysteresis, it did not change the relative electron and hole mobilities. Figure 3.17 shows that both hole and electron mobility increase after PETM and TATATO were deposited and annealed at 50 °C for 10 min. This confirms that the transition to ambipolar behavior after encapsulation is due to interaction of PETM and TATATO with the sample. In figure 3.16 we show the effect of PETM or TATATO on transfer curves. PETM dopes heavily n-type while TATATO quenches electron transport. The combination of the two components appears to have an effect similar to compensation doping in crystalline semiconductors, resulting in ambipolar behavior.

The efficacy of the polymer encapsulation layer is shown in figure 3.18. Exposure to ambient atmosphere for 10 min resulted in a decrease of electron mobility and an increase in both hole mobility and hysteresis. However, after 1 h in a low oxygen environment, electron and hole mobilities returned to pre-exposure values.

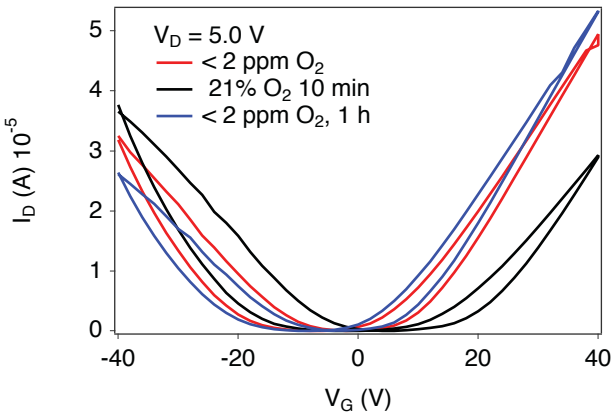

Figure 3.18: Transfer curves of epitaxially connected PbSe NCs encapsulated with UV cured polymer, showing the effect of exposure to ambient air (21% O_2) for 10 min and after return to a low oxygen environment for 1 h.

3.5 BIBLIOGRAPHY

[1] Olga L Lazarenkova and Alexander A Balandin. "Miniband formation in a quantum dot crystal". *Journal of Applied Physics* 89.10 (2001), pp. 5509–5515.

[2] E. Kalesaki et al. "Electronic structure of atomically coherent square semiconductor superlattices with dimensionality below two". *Phys. Rev. B* 88.11 (2013).

[3] E. Kalesaki et al. "Dirac Cones, Topological Edge States, and Nontrivial Flat Bands in Two-Dimensional Semiconductors with a Honeycomb Nanogeometry". *Phys. Rev. X* 4.1, 011010 (2014), p. 011010.

[4] MR Butler et al. "Electron pairing in designer materials: a novel strategy for a negative effective Hubbard U." *Nano Lett* 15.3 (2015), pp. 1597–1602.

[5] P. W. Anderson. "Absence of Diffusion in Certain Random Lattices". *Phys. Rev.* 109.5 (1958), pp. 1492–1505.

[6] D Weaire and B Kramer. "Numerical methods for electronic properties of disordered solids". *Journal of Non-Crystalline Solids* 35 (1980), pp. 9–14.

[7] Philippe Guyot-Sionnest. "Electrical Transport in Colloidal Quantum Dot Films". *The Journal of Physical Chemistry Letters* 3.9 (2012), pp. 1169–1175.

[8] BI Shklovski and AL Efros. "Percolation theory and conductivity of strongly inhomogeneous media". *Soviet Physics Uspekhi* 18.11 (1975), p. 845.

[9] NF Mott. "Conduction in glasses containing transition metal ions". *Journal of Non-Crystalline Solids* 1.1 (1968), pp. 1–17.

[10] Inuk Kang and Frank W Wise. "Electronic structure and optical properties of PbS and PbSe quantum dots". *JOSA B* 14.7 (1997), pp. 1632–1646.

[11] AS Skal and BI Shklovsky. "Mott's Formula for Low-Temperature Jump Conductivity". *Fizika Tverdogo Tela* 16.6 (1974), pp. 1820–1822.

[12] Joseph M Luther et al. "Schottky solar cells based on colloidal nanocrystal films". *Nano letters* 8.10 (2008), pp. 3488–3492.

[13] PJ Webster and KRA Ziebeck. *Landolt-Börnstein Group III Condensed Matter*. 1988.

[14] Brian L. Wehrenberg et al. "Conduction in Charged PbSe Nanocrystal Films". *The Journal of Physical Chemistry B* 109.43 (2005). PMID: 16853610, pp. 20192–20199.

[15] Brian L. Wehrenberg and Philippe Guyot-Sionnest. "Electron and Hole Injection in PbSe Quantum Dot Films". *Journal of the American Chemical Society* 125.26 (2003). PMID: 12822991, pp. 7806–7807.

[16] Dong Yu, Congjun Wang, and Philippe Guyot-Sionnest. "n-Type conducting CdSe nanocrystal solids". *Science* 300.5623 (2003), pp. 1277–1280.

[17] Jana Zaumseil and Henning Sirringhaus. "Electron and Ambipolar Transport in Organic Field-Effect Transistors". *Chemical Reviews* 107.4 (2007). PMID: 17378616, pp. 1296–1323.

[18] N Klein and H Gafni. "The maximum dielectric strength of thin silicon oxide films". *IEEE transactions on Electron Devices* 2 (1966), pp. 281–289.

[19] Iwan Moreels et al. "Composition and size-dependent extinction coefficient of colloidal PbSe quantum dots". *Chemistry of Materials* 19.25 (2007), pp. 6101–6106.

[20] G Allan and C Delerue. "Confinement effects in PbSe quantum wells and nanocrystals". *Physical Review B* 70.24 (2004), p. 245321.

[21] Matt Law et al. "Structural, optical, and electrical properties of PbSe nanocrystal solids treated thermally or with simple amines". *Journal of the American Chemical Society* 130.18 (2008), pp. 5974–5985.

[22] Joseph M Luther et al. "Structural, optical, and electrical properties of self-assembled films of PbSe nanocrystals treated with 1, 2-ethanedithiol". *ACS nano* 2.2 (2008), pp. 271–280.

[23] CS Suchand Sandeep et al. "High charge-carrier mobility enables exploitation of carrier multiplication in quantum-dot films". *Nature communications* 4 (2013), p. 2360.

[24] Y Liu et al. "PbSe quantum dot field-effect transistors with air-stable electron mobilities above 7 cm2 V(-1) s(-1)." *Nano Lett* 13.4 (2013), pp. 1578–1587.

[25] Hugo E Romero and Marija Drndic. "Coulomb blockade and hopping conduction in PbSe quantum dots". *Physical Review Letters* 95.15 (2005), p. 156801.

[26] AS Grove et al. "Investigation of thermally oxidised silicon surfaces using metal-oxide-semiconductor structures". *Solid-State Electronics* 8.2 (1965), pp. 145–163.

[27] Ju Young Woo et al. "Ultrastable PbSe nanocrystal quantum dots via in situ formation of atomically thin halide adlayers on PbSe (100)". *Journal of the American Chemical Society* 136.25 (2014), pp. 8883–8886.

[28] Ju Young Woo et al. "Air-Stable PbSe Nanocrystals Passivated by Phosphonic Acids". *Journal of the American Chemical Society* (2015).

[29] Ji-Hyuk Choi et al. "Bandlike transport in strongly coupled and doped quantum dot solids: a route to high-performance thin-film electronics". *Nano letters* 12.5 (2012), pp. 2631–2638.

[30] SJ Oh et al. "Designing high-performance PbS and PbSe nanocrystal electronic devices through stepwise, post-synthesis, colloidal atomic layer deposition." *Nano Lett* 14.3 (2014), pp. 1559–1566.

[31] SJ Oh et al. "Stoichiometric control of lead chalcogenide nanocrystal solids to enhance their electronic and optoelectronic device performance." *ACS Nano* 7.3 (2013), pp. 2413–2421.

[32] Jiang Tang et al. "Colloidal-quantum-dot photovoltaics using atomic-ligand passivation". *Nature materials* 10.10 (2011), pp. 765–771.

[33] Alexander H Ip et al. "Hybrid passivated colloidal quantum dot solids". *Nature nanotechnology* 7.9 (2012), pp. 577–582.

[34] Zhijun Ning et al. "Air-stable n-type colloidal quantum dot solids". *Nature materials* 13.8 (2014), pp. 822–828.

[35] David Zhitomirsky et al. "N-Type Colloidal-Quantum-Dot Solids for Photovoltaics". *Advanced materials* 24.46 (2012), pp. 6181–6185.

[36] Wan Ki Bae et al. "Highly effective surface passivation of PbSe quantum dots through reaction with molecular chlorine". *Journal of the American Chemical Society* 134.49 (2012), pp. 20160–20168.

[37] Dong-Kyun Ko et al. "Photovoltaic Performance of PbS Quantum Dots Treated with Metal Salts". *ACS nano* 10.3 (2016), pp. 3382–3388.

[38] Daniel M Balazs et al. "Counterion-Mediated Ligand Exchange for PbS Colloidal Quantum Dot Superlattices". *ACS nano* (2015).

[39] Milan Sykora et al. “Effect of air exposure on surface properties, electronic structure, and carrier relaxation in PbSe nanocrystals”. *ACS nano* 4.4 (2010), pp. 2021–2034.

[40] Kurtis S Leschkies et al. “Influence of atmospheric gases on the electrical properties of PbSe quantum-dot films”. *The Journal of Physical Chemistry C* 114.21 (2010), pp. 9988–9996.

[41] O Voznyy et al. “A charge-orbital balance picture of doping in colloidal quantum dot solids.” *ACS Nano* 6.9 (2012), pp. 8448–8455.

[42] William W Yu et al. “Preparation and characterization of monodisperse PbSe semiconductor nanocrystals in a noncoordinating solvent”. *Chemistry of materials* 16.17 (2004), pp. 3318–3322.

[43] Ratan Debnath et al. “Ambient-Processed Colloidal Quantum Dot Solar Cells via Individual Pre-Encapsulation of Nanoparticles”. *Journal of the American Chemical Society* 132.17 (2010). PMID: 20387887, pp. 5952–5953.

[44] Yao Liu et al. “Robust, functional nanocrystal solids by infilling with atomic layer deposition”. *Nano letters* 11.12 (2011), pp. 5349–5355.

[45] Chia-Hao M Chuang et al. “Improved performance and stability in quantum dot solar cells through band alignment engineering”. *Nature materials* 13.8 (2014), p. 796.

[46] Donghun Kim et al. “Impact of Stoichiometry on the Electronic Structure of PbS Quantum Dots”. *Phys. Rev. Lett.* 110.19 (2013), p. 196802.

[47] Charles Hoyle and Christoper Bowman. “Thiol–Ene Click Chemistry”. *Angewandte Chemie International Edition* 49.9 (2010), pp. 1540–1573.

[48] Dmitriy S Dolzhnikov et al. “Composition-matched molecular “solders” for semiconductors”. *Science* 347.6220 (2015), pp. 425–428.

[49] Joshua J Choi et al. “PbSe nanocrystal excitonic solar cells”. *Nano letters* 9.11 (2009), pp. 3749–3755.

CHAPTER 4

CONCLUSION AND FUTURE DIRECTIONS

With a myriad of tunable parameters in the synthesis, assembly, surface chemistry, and doping of semiconductor NCs, there remains many opportunities to improve performance and create new functionalities for NC solids. In this work we have shown that oriented attachment is one method to fabricate PbSe NC superlattices with interesting electronic properties. Future work will undoubtedly reveal details upon the methods presented here. A few aspects in particular present the opportunity for more investigation.

Assembly of ordered superlattices by oriented attachment is a very new and poorly understood mechanism. This process combines concepts from colloid and surface science with traditional phase transformation in atomic crystals. We have investigated one material with one ligand chemistry. However as shown in section 2.3, the ratio of NC diameter and ligand length is critical for the formation of a coherent phase boundary between connected and unconnected phases. By tuning this ratio it may be possible to fabricate highly ordered and strongly coupled NC superlattices for any size NC.

Because defects in an unconnected superlattice contribute to defects in a connected superlattice, optimized self-assembly of NC colloids is an important component towards the goal of increased order in connected superlattices. Current methods of self-assembly depend on the hard-sphere like PMF of NCs with solvated ligand shells. However we have shown that during drying this is no longer true and disorder results. New ligand-solvent systems could provide a means to control the PMF throughout the assembly process. For example, solvent-free NC systems with liquid like behavior have been synthesized using polymeric ligands[1]. Similar systems could be realized using room-temperature ionic liquids and ionic ligands to create solventless NC suspensions and avoid distortion during solvent evaporation.

Rapid progress in NC science has come from the combined efforts of synthesis, assembly,

surface chemistry, and electronic transport. In just six years the power conversion efficiency of NC PV devices has climbed from 3%[2] to 10.8%[3]. Continued efforts on synthesis and transport may lead to new behavior such as the utilization of multiple exciton generation, carrier mobility greater than other solution processed materials, or even topological states.

4.1 BIBLIOGRAPHY

[1] Jennifer L Nugent, Surya S Moganty, and Lynden A Archer. "Nanoscale organic hybrid electrolytes". *Advanced Materials* 22.33 (2010), pp. 3677–3680.

[2] Joseph M Luther et al. "Stability assessment on a 3% bilayer PbS/ZnO quantum dot heterojunction solar cell". *Advanced Materials* 22.33 (2010), pp. 3704–3707.

[3] Mengxia Liu et al. "Double-Sided Junctions Enable High-Performance Colloidal-Quantum-Dot Photovoltaics". *Advanced Materials* 28.21 (2016), pp. 4142–4148.

APPENDIX A

METHODS

A.1 PbSe Nanocrystal Synthesis

We synthesized PbSe NCs by modification of the method reported by Yu et al[1]. The entire synthesis was completed in a nitrogen glove-box with < 1 ppm oxygen and < 0.1 ppm water. Pb-oleate was prepared by heating 892 mg of PbO (Sigma, 99%) with 3.2 mL oleic acid (Sigma, tech. grade) in 15.5 mL 1-octadecene (Sigma, tech. grade) at 120 °C under 100 mTorr vacuum for 1 h. The Se precursor (TOPSe) was prepared by dissolving 2.95 g Se pellets (Sigma, 99.99%) in 22 mL tri-n-octylphosphine (Strem, 97%) (1.7 M). To 883 µL of the TOPSe solution was added 15 µL diphenylphosphine (DPP) (Sigma, 98%). The Pb-oleate solution (2.36 mL) was heated to 160 °C before injection of TOPSe:DPP under vigorous stirring, held at 160 °C for 4 min, and quenched by addition of 4 mL toluene. NCs were purified three times by centrifugation with toluene and acetonitrile as solvent and antisolvent, respectively. NC size, dispersion, and concentration were measured by absorption spectroscopy using a Cary 5000 spectrometer in tetrachloroethylene (Sigma, 99.9%)[2].

Following purification in a nitrogen glovebox, a solution was prepared for absorbance measurement. The purified NC solution was diluted by vacuum drying 100 µL from hexane and redissolving in 3 mL tetrachloroethylene. The solution was sealed in a quartz cuvette before removal from the glovebox. Background contributions from the instrument, air, and solvent were accounted for by measuring a baseline spectrum with neat tetrachloroethylene.

The mean diameter and distribution were calculated following the method of Moreels et. al.[2] Figure A.1 shows the lowest energy exciton absorption feature. We determined the peak location and peak width by fitting a Gaussian function after subtracting a linear background from the high energy tail to the low energy tail of the peak. The peak location of

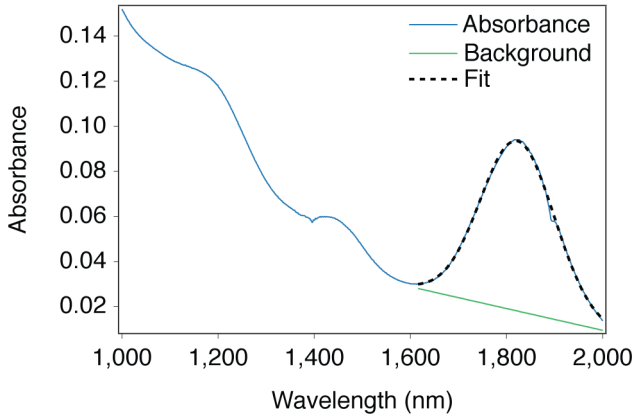

Figure A.1: Absorbance spectrum of PbSe NC colloidal solution. The mean NC diameter and standard deviation were found by fitting a Gaussian function to the first excitonic peak after subtracting a linear background.

1821 nm or 681 meV gives a mean NC diameter of 6.5 nm using the empirical equation (A.1) by Moreels et. al.

$$E_0 = 0.278 + \left(0.016d^2 + 0.209d + 0.45\right)^{-1} \tag{A.1}$$

where E_0 is the mean energy of the lowest energy exciton in eV and d is the NC diameter in nm. The standard deviation of the NC diameter was calculated by converting the standard deviation of the excitonic peak using the same equation. The standard deviation of the Gaussian peak is 76.5 nm giving energies of 710.8 meV and 653.5 meV. These energies give diameters of 6.08 nm and 6.92 nm respectively and therefore the standard deviation of the NC diameter is 0.42 nm.

A.2 Field-Effect Transistor Fabrication

Highly doped Si wafers ($< 0.02\,\Omega\,\mathrm{cm}$) with 200 nm of dry thermal oxide annealed in forming gas were purchased from Addison Engineering. Oxide was removed from the backside in 6:1 (vol.) buffered oxide etch before thermal evaporation of 10 nm Al then 70 nm Au for the gate contact. On the front side, a layer of LOR5A lift-off resist (MicroChem) was deposited

by spin-coating at 3,000 RPM for 45 s followed by a soft-bake at 180 °C for 4 min. Shipley 1805 was deposited by spin-coating at 5,000 RPM for 30 s then baked at 90 °C for 60 s before pattern transfer using an MA6 mask aligner (Karl-Suss) and developed with AZ-726 (MIF). The electrode pattern created a channel with length 100 µm and width 3 mm. Electrodes were deposited by thermal evaporation of 3 nm chrome and 30 nm gold followed by lift-off in n-methyl-2-pyrrolidone. Substrates were cleaned by sonication in acetone then 2-propanol for 5 min each followed by UV-ozone cleaning (Jelight Model 42, 42 Watts) for 10 min.

A.3 Superlattice Assembly

To form ordered NC assemblies we adapted the method reported by Dong *et al*[3, 4, 5]. The entire process was performed in a nitrogen glovebox with oxygen concentration below 2 ppm. As an example, 25 µL of a 5 µM concentration of 6.5 nm diameter PbSe NCs in n-hexane (Sigma, anhydrous 95%) was added to the surface of 1.5 mL ethylene glycol (Sigma, anhydrous 99.8%) in a 1.5 cm x 1.5 cm x 1.5 cm Teflon well and immediately covered with a glass microscope slide. After 30 min, the glass cover was removed and 25 µL of ethylene-diamine (Sigma, 99%) was injected into the ethylene glycol. After 5 min, the NC film was transferred to a substrate by the Langmuir-Schaefer method and excess ethylene glycol was rinsed using acetonitrile.

A.4 Encapsulation

The encapsulant was prepared by mixing pentaerythritol-tetrakis(3-mercaptopropionate) (Sigma, >95%), 1,3,5-triallyl-1,3,5-triazine-2,4,6(1H,3H,5H)-trione (Sigma, 98%), and acetonitrile (Sigma, anhydrous 99.8%) (1:1:2 vol.). The encapsulation layer was deposited on the NC film by spin-casting at 700 RPM for 30 s, then the speed was increased to 2000 RPM

for 30 s. Solvent was removed by annealing the device on a hotplate at 50 °C for 10 min. Finally, the encapsulant was photocured using a 4 W, 256 nm wavelength lamp at a distance of 3 cm for 10 min. Mixing, deposition, and curing were performed in a nitrogen glovebox with oxygen concentration less than 2 ppm.

A.5 Electrical Characterization

Electrical measurements were performed in a Desert Cryogenics TTP4 probe station. After loading the sample, the chamber was pumped to 1×10^{-5} Torr for at least 12 h at 295 K. Current and voltage were measured using an Agilent B1500A semiconductor parameter analyzer. For temperature dependent measurements, the temperature was scanned in both directions between 85 K and 350 K at intervals of 5 K or 10 K. The temperature was scanned repeatedly to ensure the sample had equilibrated and the measurement did not depend on temperature scan rate.

A.6 Electron Microscopy

Samples were prepared for electron microscopy by transferring NC films to a carbon-coated grid using the Langmuir-Schaefer method. Excess ethylene glycol was removed by rinsing in anhydrous methanol and acetonitrile and dried under 100 mTorr vacuum. All sample preparation was completed in a nitrogen glove box. STEM images were acquired on a NION superSTEM at an accelerating voltage of 60 keV. Lens aberrations were corrected up to and including fifth order, and an aperture size of 30 mrad was used. Bright field TEM images were acquired on an FEI T12 at an accelerating voltage of 120 keV.

A.7 X-ray Diffraction

The NC diameter distribution was also determined by fitting the form factor measured by small-angle X-ray scattering (SAXS) from the as-synthesized NCs in solution. The SAXS data was acquired using an environmentally controlled sample chamber at the D1 station of the Cornell High Energy Synchrotron Source (CHESS). The sample chamber consisted of an aluminum enclosure with two kapton windows mounted on a 4-axis goniometer. A teflon block with a 1 cm x 1 cm x 5 mm rectangular well was filled with anhydrous ethylene glycol. A separate solvent reservoir inside the enclosure was filled with 1 mL of hexane before sealing the chamber. The sample chamber was positioned such that the X-ray beam grazed the apex of the curved ethylene glycol liquid surface. The enclosure was purged with helium to decrease oxygen and then left static to allow hexane in the reservoir to reach equilibrium vapor pressure at the ambient temperature of 23 °C. A 25 µL droplet of a 5 µM concentration of PbSe NCs in hexane was then deposited on the ethylene glycol liquid surface using a 100 µL air-tight syringe though a septum in the top of the enclosure.

The scattered X-rays were recorded by a Pilatus 200k detector positioned 902 mm from the sample and calibrated using a silver behenate standard and the attenuated direct beam. Incident radiation had a flux of 1×10^{12} photon s^{-1} mm^{-2} and a wavelength of 0.1157 nm. The image was corrected for dark current. We used the software package Fit2D to azimuthually integrate the image shown in figure A.2 over the azimuthal range indicated. The integrated intensity was fit using a spherical form factor with a Gaussian distribution of the sphere radii[6]. The form factor of a sphere with radius R is given by equation (A.2).

$$F_S(q,R) = \frac{4}{3}\pi R^3 \left(3\frac{\sin(qR) - qR\cos(qR)}{(qR)^3}\right) \tag{A.2}$$

The distribution of the NC radii with an average radius R_0 and standard deviation σ was modeled by equation (A.3).

$$P(R) = \frac{1}{\sqrt{2\pi}\sigma} exp\left(\frac{-(R-R_0)^2}{2\sigma^2}\right) \tag{A.3}$$

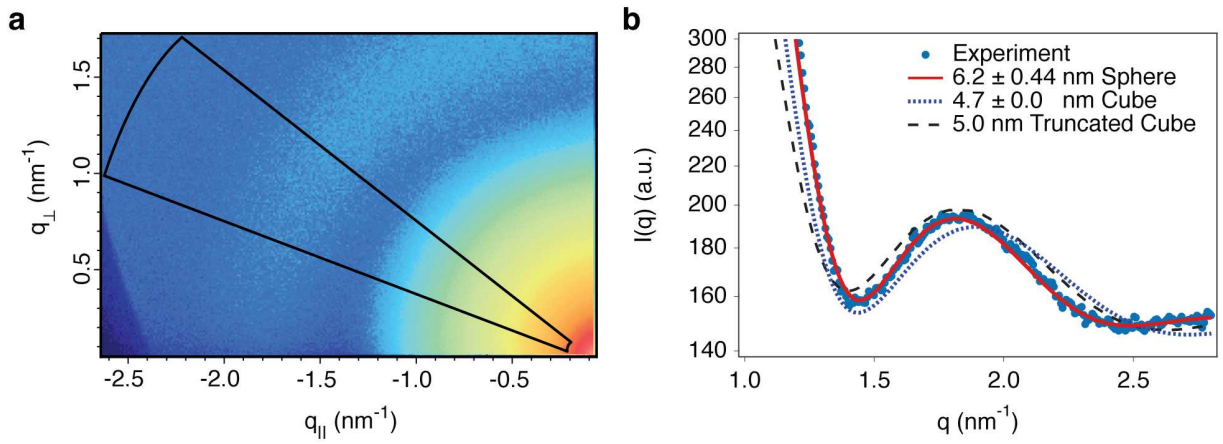

Figure A.2: NC size and distribution from X-ray scattering. a, Small angle X-ray scattering of a colloidal solution of PbSe NCs. Color values are arbitrary units and scaled logarithmically. b, Azimuthally integrated intensity from the outlined area of the scattering image. Lines show form factor functions for a sphere, a cube, and a truncated cube, see text for details.

The calculated intensity is given by equation (A.4).

$$I(q) = B_0 + S \int P(R)\,|F_S(q,R)|^2 \tag{A.4}$$

Equation (A.4) was fit to the measured intensity by including a scaling factor S to account for the photon flux and a constant background factor B_0 to account for diffuse scattering. The best fit is shown in figure A.2 with values $R_0 = 3.1\,\text{nm}$ and $\sigma = 0.22\,\text{nm}$. This gives an average NC diameter of $(6.20 \pm 0.44)\,\text{nm}$, in good agreement with values calculated from the absorbance spectrum $(6.50 \pm 0.42)\,\text{nm}$.

A.8 Computer Algorithms for Image Analysis

A.8.1 Determination of Nanocrystal Location and Diameter

Measuring NC locations and diameters from TEM images is commonly performed using image analysis software packages such as ImageJ. However, these packages are not reliable if

the boundary of the NC is not distinct such as when NCs are joined by an epitaxial bridge. The process can be done manually, but this is not feasible when analyzing dozens of images with thousands of NCs per image. Additionally, hand measurement introduces user bias. Therefore, we wrote an automated image analysis algorithm.

The main operations performed by the algorithm are threshold and watershed. Briefly, the algorithm first performs a threshold operation to convert the greyscale image to a binary image based on the distribution of pixel values. The threshold operation attempts to separate NC sample pixels from empty background pixels. The binary image is then converted to a linear scale image where the value of each pixel represents the Euclidean distance from that pixel to the nearest background pixel hereafter called a *distance image*. The sample pixels are then separated into groups, each group representing a NC. The groups are determined by a watershed operation on the distance image. The location of each NC is then found by calculating the center of mass of the group of pixels belonging to each NC pixel group. The diameter of each NC is calculated as the average of the major and minor axes of an ellipse fit to the group of pixels belonging to each NC. The threshold operation is repeated, this time after dividing the original greyscale image into sections. Each section is a rectangle with dimensions some multiple of the average NC diameter found after the first threshold operation. Running the threshold operation on each section rather than the entire image improves the quality of the binary image by compensating for non-uniform illumination in a bright-field image or non-uniform scattering due to sample thickness variation in a dark-field image. The watershed operation is repeated as before on a distance image generated from the binary image sections to refine the NC locations and diameters. The output of the algorithm on a dark-field STEM image is shown in figure A.3.

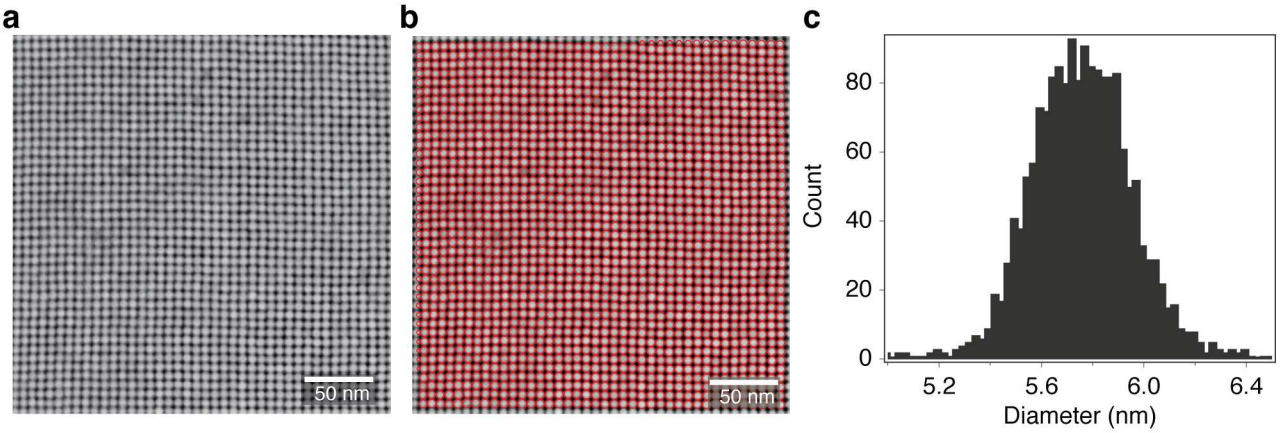

Figure A.3: Measurement of NC diameter from a transmission electron micrograph. **a**, An annular dark-field scanning transmission electron micrograph of a NC superlattice. **b**, The location and diameter of 1,693 NCs in the image was calculated using in-house code. The NC locations and diameters are represented by red circles overlaid on the image. **c**, A histogram of the diameters found in the image, with an average of 5.75 nm and a standard deviation of 0.19 nm.

A.8.2 Determination of Epitaxial Connectivity

We measured the distribution of epitaxial connection widths by analysis of microscopy images. We used both atomically resolved dark field STEM and bright field TEM images of monolayer samples. The atomically resolved STEM images were analyzed by counting the number of PbSe {200} planes in the epitaxial connection. As an example, we used the software package ImageJ to mark the width of 124 atomically resolved epitaxial connections by hand, drawing a line across each connection at the narrowest part. In figure A.4 we show each connection and a histogram of the measurements. The mean connection width is 2.9 nm with a standard deviation of 0.68 nm.

We also analyzed lower magnification bright field TEM images with more NC connections for better statistics. Figure A.5 shows such an image with 10,122 epitaxial connections.

To measure the number of connections in a bright field image, we used a software routine. Details of the software routine can be found in chapter B but we describe the routine briefly here. First a Voronoi diagram was constructed using the NC locations as cell centers. The

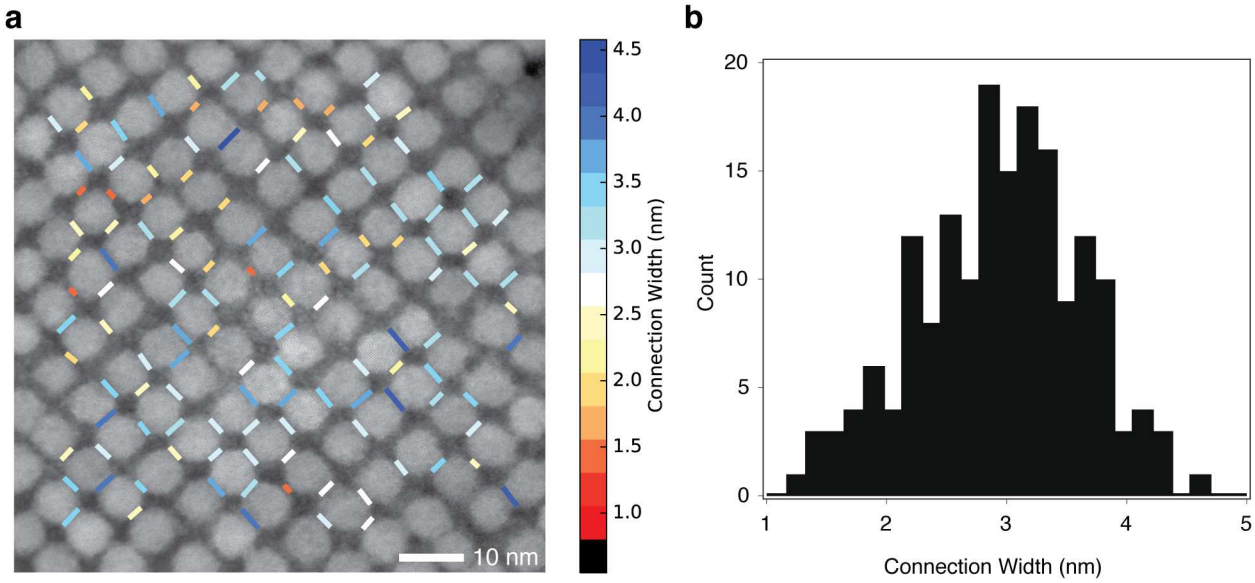

Figure A.4: Distribution of epitaxial connections from an atomically resolved image. **a**, Annular dark field STEM image of a monolayer superlattice overlaid with lines marking the widths of 124 epitaxial connections. **b**, Histogram of the epitaxial connection widths shown in the image. The mean width is 2.9 nm with a standard deviation of 0.68 nm.

line segments that make up the cells of the Voronoi diagram intersect the epitaxial connections midway between each nearest neighbor pair of NCs. To measure the width, we fit an approximately square function to the pixel intensity values along the Voronoi diagram lines, as shown in figure A.6. We use a sum of Gaussian functions instead of a true square function because the function must be differentiable for the least-squares minimization fitting routine. The fit function is given by $f(x) = \sum_{n=-5}^{5} \exp\left(\frac{-(x-x_0-0.1nw)^2}{2(0.1w/2)^2}\right)$ where x_0 is the midpoint of the connection and w is the connection width. The parameters x_0, w are found by minimizing the sum of the residuals $|f(x) - I(x)|$ where $I(x)$ is the intensity at pixel x.

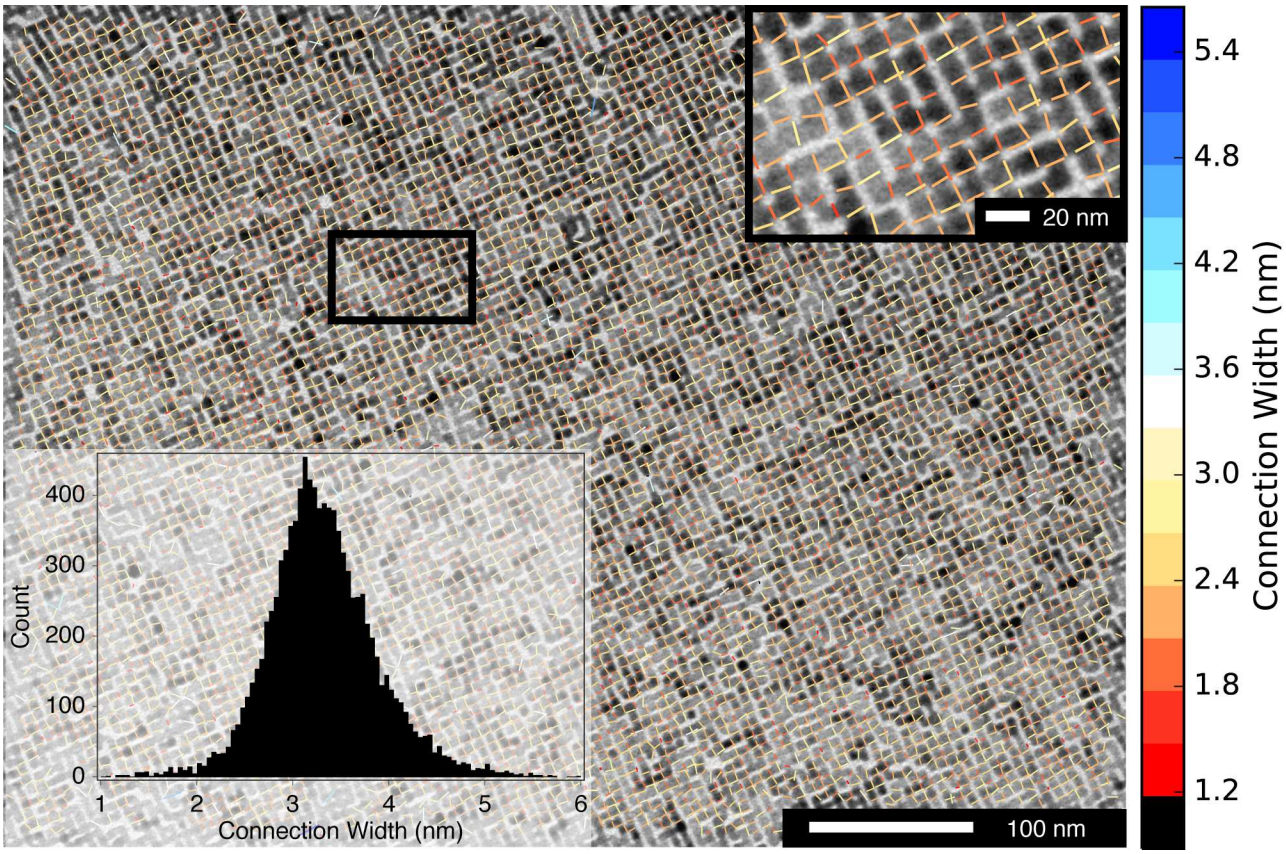

Figure A.5: Distribution of epitaxial connections from TEM. A bright field transmission electron micrograph of a monolayer superlattice. The location and width of 10,122 epitaxial connections are indicated by solid lines. Upper inset shows a magnified view of the region outlined with a black rectangle. The inset histogram shows an average width of 3.26 nm with a standard deviation of 0.48 nm.

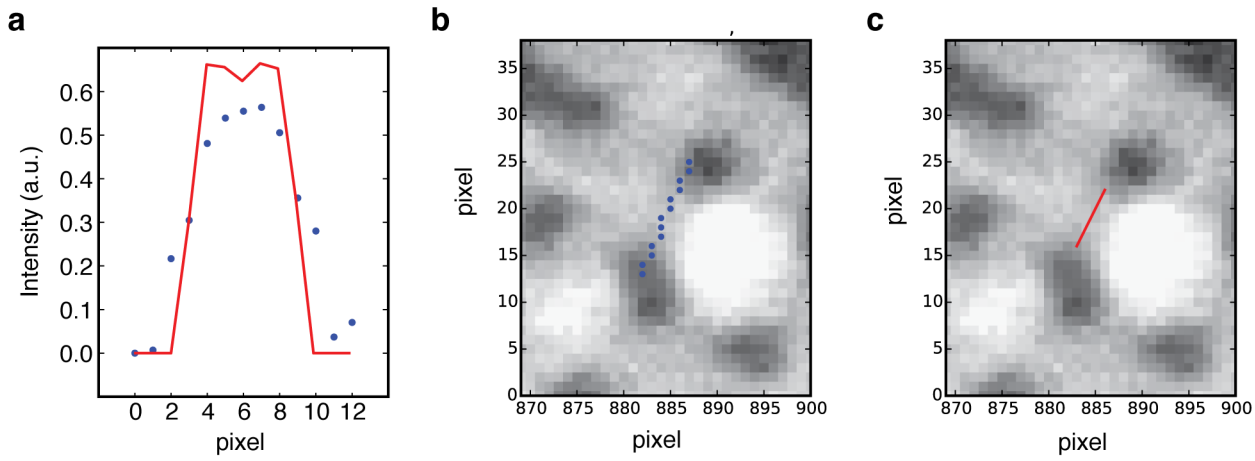

Figure A.6: Automated analysis of epitaxial connection width. **a**, Line profile of pixel intensities across an epitaxial connection. Blue markers are pixel intensity values, red line is a fit to the pixel values. **b**, The epitaxial connection under analysis. The image is a bright field TEM image that has been inverted such that the background is black and the NCs are white. The blue markers indicate the pixels analyzed. **c**, The red line indicates the location and width of the epitaxial connection from the fit.

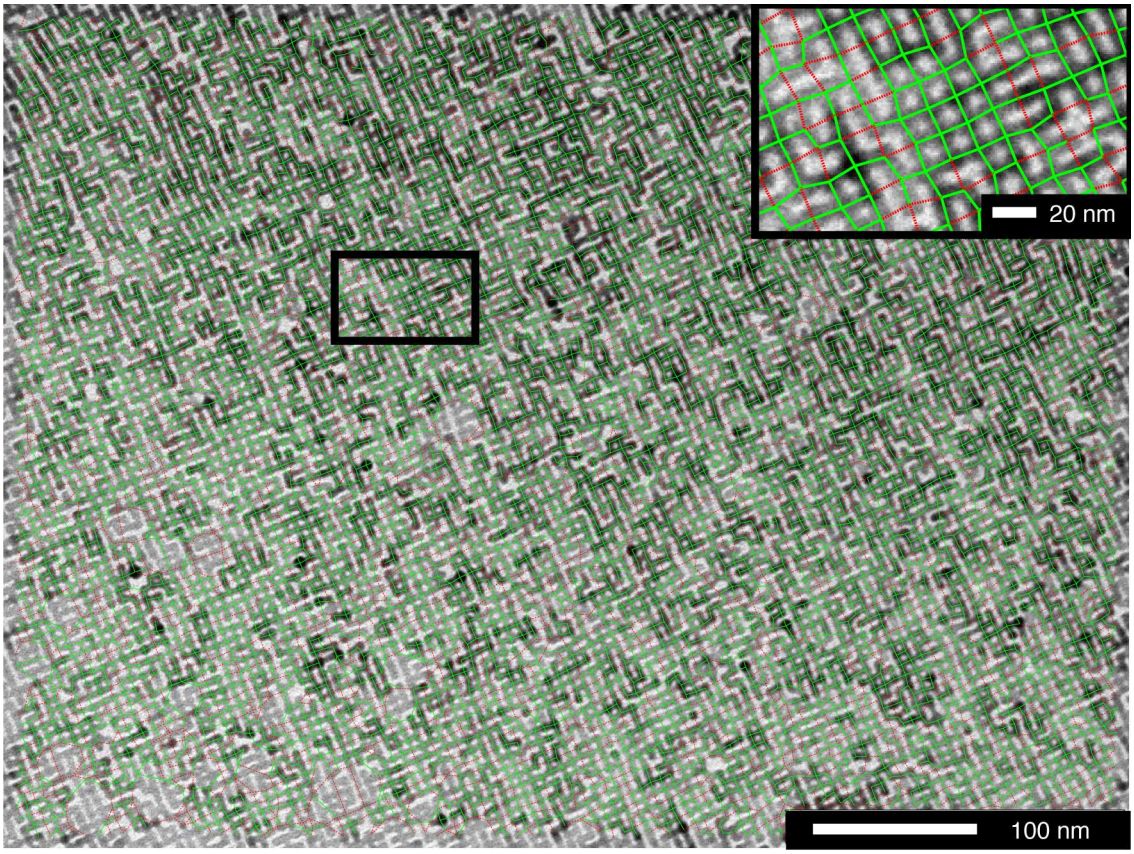

Figure A.7: Connectivity of a square superlattice. Solid green lines indicate epitaxially connected neighbors, red dashed lines indicate unconnected neighbors. Of the 13,937 nearest neighbors analyzed, 3,815 are not connected, yielding a connectivity of 73%.

A.8.3 Determination of Local Order and Defects

After locating NC sites in an image, the bond order parameter can be calculated for each site. The order parameter quantifies the deviation from ideal symmetry for a specific bond order. For example in a 2D square lattice NNs of a site are related by an angle of $\frac{\pi}{2}$ rad, in a hexagonal lattice by $\frac{\pi}{6}$ rad. The order parameter for a single site j with n_j NNs is calculated according to equation (A.5) for a bond order number n[7]. The contribution of each neighbor is weighted by the length of the facet l_k of the Voronoi cell between the sites where L is the perimeter of the Voronoi cell enclosing site j[8]. The angle of the line connecting site j to site k relative to an arbitrary axis is given by θ_{jk}. The imaginary number is given by $\mathrm{i} = \sqrt{-1}$.

$$\Psi_j^n = \sum_{k=1}^{n_j} \frac{l_k}{L} exp\left(\mathrm{i}n\theta_{jk}\right) \tag{A.5}$$

The global order parameter can be calculated by summing Ψ_j^n for all sites. However, it is useful to map the order parameter for each site in the image to visualize regions of order and disorder. In figure A.8 we show an example TEM image of a connected superlattice with square symmetry and the corresponding Ψ_4 map. It is clear in the TEM image there are two superlattice grains joined by a grain boundary. The grain boundary can be identified in the map of Ψ_4 by the continuous line of NC sites with a low Ψ_4 value. These sites are undercoordinated because of the loss of symmetry at the grain boundary. Defects common in atomic lattices can also be identified by undercoordinated sites in the Ψ_4 map such as at edge dislocations.

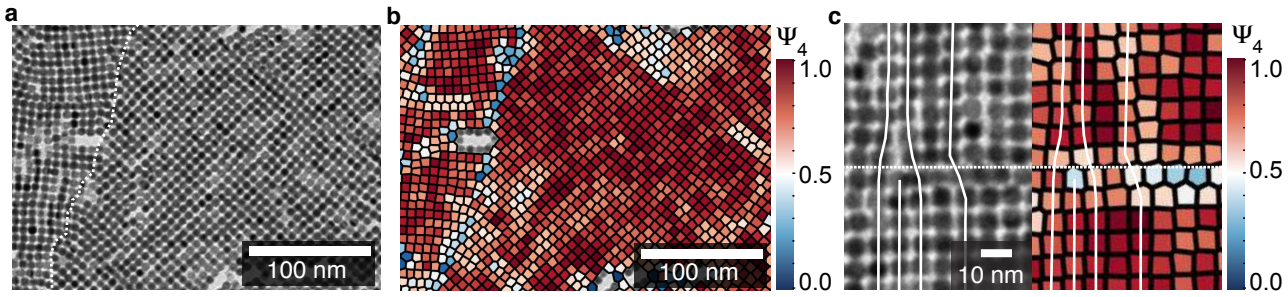

Figure A.8: Bond order map of a connected square superlattice. **a**, TEM image of 7.3 nm diameter PbSe NCs in two square superlattice grains. The boundary between the grains is marked by the dashed white line. **b**, Voronoi diagram with the cells colored according to the Ψ_4 value for each site. **c**, Detailed view of an edge dislocation.

A.9 BIBLIOGRAPHY

[1] William W Yu et al. "Preparation and characterization of monodisperse PbSe semiconductor nanocrystals in a noncoordinating solvent". *Chemistry of materials* 16.17 (2004), pp. 3318–3322.

[2] Iwan Moreels et al. "Composition and size-dependent extinction coefficient of colloidal PbSe quantum dots". *Chemistry of Materials* 19.25 (2007), pp. 6101–6106.

[3] A Dong, Y Jiao, and DJ Milliron. "Electronically coupled nanocrystal superlattice films by in situ ligand exchange at the liquid-air interface." *ACS Nano* 7.12 (2013), pp. 10978–10984.

[4] Angang Dong et al. "Binary nanocrystal superlattice membranes self-assembled at the liquid-air interface". *Nature* 466.7305 (2010), pp. 474–477.

[5] Angang Dong et al. "Two-dimensional binary and ternary nanocrystal superlattices: the case of monolayers and bilayers". *Nano letters* 11.4 (2011), pp. 1804–1809.

[6] Rémi Lazzari. "IsGISAXS: a program for grazing-incidence small-angle X-ray scattering analysis of supported islands". *Journal of Applied Crystallography* 35.4 (2002), pp. 406–421.

[7] Carlos Avendaño and Fernando A Escobedo. "Phase behavior of rounded hard-squares". *Soft Matter* 8.17 (2012), pp. 4675–4681.

[8] Walter Mickel et al. "Shortcomings of the bond orientational order parameters for the analysis of disordered particulate matter". *The Journal of chemical physics* 138.4 (2013), p. 044501.

APPENDIX B

COMPUTER CODE

The following code is available at:

https://bitbucket.org/KevinWhitham/superlattice-structure-analysis.git

B.1 Python Code for Measuring Superlattice Properties

The following code determines the location and diameter of objects in a greyscale image. It can then use this information to output the size distribution, order parameter, orientation, and connectivity of objects in an array. The data output can then be used to calculate a radial distribution function.

```
# structure_metric.py
# Calculates order parameter for a collection of points in 2D space
# Reference: Mickel, Walter, et al. "Shortcomings of the bond
↪  orientational order
#         parameters for the analysis of disordered particulate matter."
#         The Journal of chemical physics (2013)
# 20140902 Kevin Whitham, Cornell University
# License: GNU Public License (GPL) v.3.0

# general math
import numpy as np
from scipy import stats # for stats.mode
```

```
# for data structures
import collections
import scipy.sparse as sparse

# for fitting the radius distribution
from scipy.optimize import curve_fit
from scipy.stats import linregress

# for disabling annoying warnings
import warnings

# plotting
import matplotlib.pyplot as plt
from matplotlib.collections import PatchCollection
from matplotlib.collections import LineCollection
from matplotlib.patches import Polygon, Circle, Arrow
from matplotlib.colors import LinearSegmentedColormap
from mpl_toolkits.axes_grid1 import make_axes_locatable, Size

# Voronoi
from scipy.spatial import Voronoi

# for finding particle centers, diameters, etc.
from skimage.measure import regionprops
from skimage.filters import threshold_otsu,threshold_adaptive,
↪  threshold_isodata
```

```
from skimage.morphology import watershed, remove_small_objects,
↪  binary_erosion, binary_dilation, binary_closing, binary_opening
from skimage.feature import peak_local_max
from scipy import ndimage
from skimage.draw import circle
from skimage.morphology import disk

# for command line interface
import argparse
from skimage import io as skimio
from os import path
from os import walk

# regular expressions for parsing scalebar filenames
import re

# for matching the scale bar
from skimage.feature import match_template

# minkowski structure metric function written and compiled with Cython
from order_parameter import order

# globals
DEBUG_OUTPUT = False


def make_binary_image(im, white_background, min_feature_size, small):
```

```
image = im.astype('float32')

if white_background:
    image = np.abs(image-np.max(image))

# get rid of large background patches before local thresholding
# do a global threshold
thresh = threshold_isodata(image)
binary = image > thresh

# get rid of speckle
# this is not good for very small particles
if not small:
    binary_closing(binary, selem=disk(min_feature_size), out=binary)

# make a map of the distance to a particle to find
# large background patches
# this is a distance map of the inverse binary image
distance = ndimage.distance_transform_edt(1-binary)

# do a global threshold on the distance map to
# select the biggest objects
# larger than a minimum size to prevent masking of particles in images
 ↪  with
# no empty patches
dist_thresh = threshold_isodata(distance)
```

```
mask = distance < dist_thresh

# remove areas of background smaller than a certain size in the mask
# this fills in small pieces of mask between particles where the
↪  voronoi
# vertices will end up
# this gets rid of junk in the gap areas
binary_opening(mask, selem=disk(min_feature_size), out=mask)

binary = binary * mask

# get rid of speckle
binary = remove_small_objects(binary,min_size=max(min_feature_size,2))

# 3 iterations is better for large particles with low contrast
binary = ndimage.binary_closing(binary,iterations=1)

if DEBUG_OUTPUT:
    fig, ax = plt.subplots(2, 2)
    ax[0,0].imshow(binary)
    ax[0,0].set_title('Global threshold')

    ax[0,1].imshow(distance)
    ax[0,1].set_title('Distance map')

    ax[1,0].imshow(mask)
    ax[1,0].set_title('Modified Distance Based Mask')
```

```
        ax[1,1].imshow(binary)
        ax[1,1].set_title('Masked Binarized Image')
        plt.show()

    return binary, mask

def adaptive_binary_image(im, white_background, min_feature_size, std_dev,
↪   mask):

    image = im.astype('float32')

    if white_background:
        image = np.abs(image-np.max(image))

    # the block size should be large enough to include 4 to 9 particles
    local_size = 40*min_feature_size
    binary = threshold_adaptive(image,block_size=local_size)

    # close any small holes in the particles
    # 3 iterations is better for large particles with low contrast
    #binary = ndimage.binary_closing(binary,iterations=1)
    binary_closing(binary,
    ↪   selem=disk(int(max((0.414*(min_feature_size-std_dev)),2)/2.0)),
    ↪   out=binary)

    # remove speckle from background areas in the binary image
```

```
binary = binary * mask #binary_erosion(mask,
↪  selem=disk(int(min_feature_size)))
binary_opening(binary,
↪  selem=disk(int(max((min_feature_size-3.0*std_dev),2)/2.0)),
↪  out=binary)

# make a distance map of the inverted image
distance = ndimage.distance_transform_edt((1-binary))

# do a global threshold on the distance map to select the biggest
↪  objects
# larger than a minimum size to prevent masking of particles in images
↪  with no
# empty patches
dist_thresh = threshold_isodata(distance)
new_mask = distance < dist_thresh

# thresholding is selecting too much background, try AND'ing it with
↪  the binary image
# this might only be necessary for small particles
new_mask = new_mask * binary

# remove areas of background in the mask smaller than a certain size
# this fills in small pieces of mask between particles where the
↪  voronoi
# vertices will end up
```

```
    # min_feature_size here should be the radius found by global
    ↪ threshold, so triple it
    # to close any particle sized holes
    dilation_size = max(int(1),int(3 * min_feature_size))
    new_mask = binary_closing(new_mask,
    ↪ selem=np.ones((dilation_size,dilation_size)))

    if DEBUG_OUTPUT:
        fig, ax = plt.subplots(1, 2)
        ax[0].imshow(binary)
        ax[0].set_title('Adaptive thresh')

        ax[1].imshow(new_mask)
        ax[1].set_title('Adaptive mask')
        plt.show()

    return binary, new_mask

def morphological_threshold(im, white_background, mean_radius,
↪ min_feature_size, small, mask):

    if DEBUG_OUTPUT:
        print('Morphological threshold mean radius (px): '+str(mean_radius))

    im_mod = np.array(im, dtype='float64')

    if white_background:
```

```
    im_mod = np.abs(im_mod-np.max(im_mod))

# subtract the mean before running match_template
# not sure this works quite right
im_mod = im_mod - np.mean(im_mod)

# set large areas of background to zero using the mask
#im_mod = im_mod * mask

if DEBUG_OUTPUT:
    print('Maximum image value: '+str(np.max(im_mod)))
    print('Minimum image value: '+str(np.min(im_mod)))

if small:
    template_matrix =
     ↪ np.pad(disk(max(2,int(mean_radius))),pad_width=max(2,int(mean_radius)),
     ↪ mode='constant',constant_values=0)
else:
    template_matrix =
     ↪ np.pad(disk(int(mean_radius/4)),pad_width=int(mean_radius),
     ↪ mode='constant',constant_values=0)

matched_im =
 ↪ match_template(im_mod,template=template_matrix,pad_input=True)

if DEBUG_OUTPUT:
    print(np.max(matched_im))
```

```
    print(np.min(matched_im))

thresh = threshold_isodata(matched_im)
matched_im_bin = matched_im > thresh

matched_im_bin *= mask

distance = ndimage.distance_transform_edt(matched_im_bin)

# dilate the distance map to merge close peaks (merges multiple peaks
↪  in one particle)
distance = ndimage.grey_dilation(distance,size=min_feature_size)

local_maxi =
↪  peak_local_max(distance,indices=False,min_distance=min_feature_size)
markers = ndimage.label(local_maxi)[0]

with warnings.catch_warnings():
    warnings.simplefilter('ignore')
    labels_th = watershed(-distance, markers, mask=matched_im_bin)

if DEBUG_OUTPUT:
    f, ax = plt.subplots(2, 2)
    ax[0,0].imshow(matched_im, interpolation='none')
    ax[0,0].set_title('template match')

    ax[0,1].imshow(matched_im_bin)
```

```
        ax[0,1].set_title('bin')

        ax[1,0].imshow(distance, interpolation='none')
        ax[1,0].set_title('distance')

        ax[1,1].imshow(labels_th,cmap=plt.cm.flag, interpolation='none')
        ax[1,1].set_title('labels')
        plt.show()

    return labels_th, matched_im_bin


def get_particle_centers(im, white_background, pixels_per_nm, morph, small):

    if pixels_per_nm == 0:
        # default value
        pixels_per_nm = 1.0

    # minimum size object to look for
    min_feature_size = int(3) #int(3*pixels_per_nm)

    global_binary, mask = make_binary_image(im, white_background,
    ↪  min_feature_size, small)

    # create a distance map to find the particle centers
    # as the points with maximal distance to the background
    distance = ndimage.distance_transform_edt(global_binary)
```

```
# dilate the distance map to merge close peaks (merges multiple peaks
↪  in one particle)
distance = ndimage.grey_dilation(distance,size=min_feature_size)

# min_distance=5 for large particles
local_maxi =
↪  peak_local_max(distance,indices=False,min_distance=min_feature_size)
markers = ndimage.label(local_maxi)[0]

with warnings.catch_warnings():
    warnings.simplefilter('ignore')
    global_labels = watershed(-distance, markers, connectivity=None,
    ↪  offset=None, mask=global_binary)

# get the particle radii
global_regions = regionprops(global_labels)
gobal_radii = []

for props in global_regions:
    # define the radius as half the average of the major and minor
    ↪  diameters

    ↪  gobal_radii.append(((props.minor_axis_length+props.major_axis_length)/4)/pix

# minimum size object to look for
global_mean_radius = np.mean(gobal_radii)*pixels_per_nm
```

```
global_radii_sd = np.std(gobal_radii)*pixels_per_nm

print('Mean radius global threshold (px): %(rad).2f SD: %(sd).2g' %
 ↪  {'rad':global_mean_radius, 'sd':global_radii_sd})

feature_size = int(max(global_mean_radius, min_feature_size))
std_dev = int(max(global_radii_sd, min_feature_size))

adaptive_binary, adaptive_mask = adaptive_binary_image(im,
 ↪  white_background, feature_size, std_dev, mask)

# create a distance map to find the particle centers
# as the points with maximal distance to the background
distance = ndimage.distance_transform_edt(adaptive_binary)

# dilate the distance map to merge close peaks (merges multiple peaks
 ↪  in one particle)
distance = ndimage.grey_dilation(distance,size=feature_size)

local_maxi =
 ↪  peak_local_max(distance,indices=False,min_distance=feature_size)
markers = ndimage.label(local_maxi)[0]

with warnings.catch_warnings():
    warnings.simplefilter('ignore')
    adaptive_labels = watershed(-distance, markers, connectivity=None,
     ↪  offset=None, mask=adaptive_binary)
```

```
adaptive_regions = regionprops(adaptive_labels)
adaptive_radii = []

for props in adaptive_regions:
    # define the radius as half the average of the major and minor
    ↪ diameters

    ↪ adaptive_radii.append(((props.minor_axis_length+props.major_axis_length)/4)/

# minimum size object to look for
adaptive_radii_sd = np.std(adaptive_radii)
adaptive_mean_radius = np.mean(adaptive_radii)
print('Mean radius adaptive threshold (px): %(rad).2f SD: %(sd).2g' %
↪ {'rad':adaptive_mean_radius*pixels_per_nm,
↪ 'sd':adaptive_radii_sd*pixels_per_nm})


# morphological thresholding
if morph:
    print('Using morphological threshold')
    labels, morph_binary = morphological_threshold(im, white_background,
    ↪ int(adaptive_mean_radius*pixels_per_nm),
    ↪ int(adaptive_mean_radius*pixels_per_nm)/2, small, adaptive_mask)
    regions = regionprops(labels)
    binary = morph_binary
```

```
elif global_radii_sd/global_mean_radius <
↪  adaptive_radii_sd/adaptive_mean_radius:
    print('Using global threshold')
    regions = global_regions
    binary = global_binary
    labels = global_labels
else:
    print('Using adaptive threshold')
    regions = adaptive_regions
    binary = adaptive_binary
    labels = adaptive_labels

if DEBUG_OUTPUT:
    fig, ax = plt.subplots(2, 2)
    ax[0,0].imshow(im, cmap='gray')
    ax[0,0].set_title('Greyscale Image')



    ax[0,1].imshow(binary)
    ax[0,1].set_title('Binary Image')


    ax[1,0].imshow(labels,cmap=plt.cm.prism)
    ax[1,0].set_title('Labels')
    plt.show()

# get the particle centroids again, this time with better thresholding
pts = []
```

```
    radii = []

    for props in regions:
        # centroid is [row, col] we want [col, row] aka [X,Y]
        # so reverse the order
        pts.append(props.centroid[::-1])

        # define the radius as half the average of the major and minor
        ↪ diameters

        ↪ radii.append(((props.minor_axis_length+props.major_axis_length)/4)/pixels_pe

    if morph:
        print('Mean radius morphological threshold (px): %(rad).2f SD:
        ↪ %(sd).2g' % {'rad':np.mean(radii)*pixels_per_nm,
        ↪ 'sd':np.std(radii)*pixels_per_nm})


    return np.asarray(pts,dtype=np.double),
    ↪ np.asarray(radii,dtype=np.double).reshape((-1,1)), adaptive_mask,
    ↪ binary

def get_image_scale(im):

    scale = 0.0
    topleft = (0,0)
    bottomright = (0,0)
```

```
input_path = path.normpath('../resources/scalebars')

# load all files in scalebars directory and sub-directories
scalebar_filesnames = []
for (dirpath, dirnames, filesnames) in walk(input_path):
    scalebar_filesnames.extend(filesnames)

scale_bars = []
for filename in scalebar_filesnames:
    ipart = 1
    fpart = 0.0

    reg_result = re.search('Scale_(\d*)p*(\d*)',filename)

    if reg_result:

        if reg_result.group(1):
            ipart = int(reg_result.group(1))

        if reg_result.group(2):
            fpart = float('0.'+reg_result.group(2))

        px_per_nm = ipart+fpart

        # images of scale bars to match with the input image
        # second element is the scale in units of pixels/nm
```

```
        ↪ scale_bars.append([skimio.imread(input_path+'/'+filename,as_grey=True,pl

match_score = []

for scale_bar,pixels_per_nm in scale_bars:

    result = match_template(im,template=scale_bar)

    ij = np.unravel_index(np.argmax(result), result.shape)
    row, col = ij

    match_score.append(result[row][col])

    # the match score should be about 0.999999
    if result[row][col] > 0.99:

        topleft = (row,col)
        bottomright = (row+scale_bar.shape[0],col+scale_bar.shape[1])

        scale = pixels_per_nm

        print('Scale: '+str(scale)+' pixels/nm, Score:
        ↪ '+str(np.max(match_score)))
        break # we found it, stop looking

if np.max(match_score) < 0.99:
```

```
        print('!!!!!!!!!!!!!!!!!!!! No scale bar found
        ↪  !!!!!!!!!!!!!!!!!!!!')

    return [np.double(scale),(topleft,bottomright)]

def create_custom_colormap():

    # get a color map for mapping metric values to colors of some color
    ↪  scale
    value_rgb_pairs = []
    rgb_array =
    ↪  np.asarray([[0,0,0],[255,0,0],[255,50,34],[255,109,59],[255,177,102],[255,220,12
    rgb_array /= 255
    rgb_list_norm = []

    for value, color in zip(np.linspace(0,1,16),rgb_array):
        value_rgb_pairs.append((value,color))

    return LinearSegmentedColormap.from_list(name="custom",
    ↪  colors=value_rgb_pairs, N=16)

def plot_symmetry(im, msm, vor, bond_order, symmetry_colormap, mask,
↪  no_fill, map_edge_particles):

    cell_patches = []
    metric_list = []
    orientation_angle_list = []
```

```
for region_index,metric,orientation_angle in msm:
    plot_this_cell = 0
    region_index = int(region_index)
    region = vor.regions[region_index]
    verts = np.asarray([vor.vertices[index] for index in region])

    # don't plot cells inside masked off regions of the image (blank
    ↪ patches)
    int_verts = np.asarray(verts,dtype='i4')
    if map_edge_particles:
        if np.any(mask[int_verts[:,1],int_verts[:,0]] == 1):
            plot_this_cell = 1
    else:
        if np.all(mask[int_verts[:,1],int_verts[:,0]] > 0):
            plot_this_cell = 1

    if plot_this_cell:
        if no_fill:

            ↪ cell_patches.append(Polygon(verts,closed=True,facecolor='none',edgec
        else:

            ↪ cell_patches.append(Polygon(verts,closed=True,edgecolor='none'))

        metric_list.append(metric)
        orientation_angle_list.append(orientation_angle)
```

```
if no_fill:
    pc = PatchCollection(cell_patches,match_original=True,alpha=1)
else:
    pc = PatchCollection(cell_patches,match_original=False,
     ↪  cmap=symmetry_colormap, edgecolor='k', alpha=1)
    pc.set_array(np.asarray(metric_list))

plt.gca().add_collection(pc)

# set the limits for the plot
# set the x axis range
plt.gca().set_xlim(0, im.shape[1])

# set the y-axis range and flip the y-axis
plt.gca().set_ylim(im.shape[0], 0)

# save this plot to a file
plt.gca().set_axis_off()

if not no_fill:
    # add the colorbar
    divider = make_axes_locatable(plt.gca())
    cax = divider.append_axes(position='right', size='5%', pad = 0.05)
    cbar = plt.colorbar(pc, cax=cax)
    cax.set_xlabel('$\Psi_'+str(bond_order)+'$', fontsize=18)
    cax.xaxis.set_label_position('top')
    cax.xaxis.set_label_coords(0.5, 1.04)
```

```
    ↪ plt.savefig(output_data_path+'/'+filename+'_Psi'+str(bond_order)+'_map.png',bbox

    # plot a histogram of the order parameter
    plt.figure(2)
    plt.hist(metric_list,bins=len(msm)/4)
    plt.xlabel('$\Psi_'+str(bond_order)+'$')
    plt.ylabel('Count')

    ↪ plt.savefig(output_data_path+'/'+filename+'_Psi'+str(bond_order)+'_hist.png',
    ↪ bbox_inches='tight')

def plot_orientation(im, msm, vor, pts, radii, pixels_per_nm,
↪ orientation_colormap, mask, map_edge_particles):

    # set to 1 to plot a vector map instead of colormap
    vector_map = 0

    cell_patches = []
    arrow_patches = []
    orientation_angle_list = []
    mean_radius = np.mean(radii)
    for region_index,metric,orientation_angle in msm:
        plot_this_cell = 0
        region = vor.regions[region_index]
        verts = np.asarray([vor.vertices[index] for index in region])
```

```
# don't plot cells inside masked off regions of the image (blank
↪ patches)
int_verts = np.asarray(verts,dtype='i4')
if map_edge_particles:
    if np.any(mask[int_verts[:,1],int_verts[:,0]] == 1):
        plot_this_cell = 1
else:
    if np.all(mask[int_verts[:,1],int_verts[:,0]] > 0):
        plot_this_cell = 1

if plot_this_cell:
    if vector_map:
        x = np.mean(int_verts[:,0])
        y = np.mean(int_verts[:,1])
        dx = mean_radius*np.cos(np.pi/180.0 * orientation_angle)
        dy = mean_radius*np.sin(np.pi/180.0 * orientation_angle)

        ↪ arrow_patches.append(Arrow(x,y,dx,dy,facecolor='none',edgecolor='r')

        ↪ cell_patches.append(Polygon(verts,closed=True,facecolor='none',edgec
    else:

        ↪ cell_patches.append(Polygon(verts,closed=True,edgecolor='none'))

        # This works for centrosymmetric cells (hexagon, square,
        ↪ not triangles)
```

```
            # give cells with +/- 180 deg. orientation angles the same
            ↪ color
            if orientation_angle < 0:
                orientation_angle += 180.0

            if orientation_angle > 90:
                orientation_angle = np.abs(orientation_angle-180.0)

            orientation_angle_list.append(orientation_angle)

if vector_map:
    arrow_collection =
    ↪ PatchCollection(arrow_patches,match_original=True,
    ↪ cmap=orientation_colormap, alpha=1)
    cell_collection = PatchCollection(cell_patches,match_original=True,
    ↪ alpha=1)
    plt.gca().add_collection(arrow_collection)
else:
    cell_collection =
    ↪ PatchCollection(cell_patches,match_original=False,cmap=orientation_colormap,
    ↪ alpha=1)
    cell_collection.set_array(np.asarray(orientation_angle_list))

plt.gca().add_collection(cell_collection)

# set the limits for the plot
# set the x axis range
```

```
    plt.gca().set_xlim(0, im.shape[1])

    # set the y-axis range and flip the y-axis
    plt.gca().set_ylim(im.shape[0], 0)

    # save this plot to a file
    plt.gca().set_axis_off()

    # add the colorbar
    divider = make_axes_locatable(plt.gca())
    cax = divider.append_axes(position='right', size='5%', pad = 0.05)
    cbar = plt.colorbar(cell_collection, cax=cax)
    cax.set_xlabel('$\Theta$ (deg)', fontsize=18)
    cax.xaxis.set_label_position('top')
    cax.xaxis.set_label_coords(0.5, 1.04)


    ↪  plt.savefig(output_data_path+'/'+filename+'_orientation_map.png',bbox_inches='ti

def plot_particle_outlines(im, pts, radii, pixels_per_nm):

    cell_patches = []
    for center,radius in zip(pts,radii):

        ↪  cell_patches.append(Circle(center,radius*pixels_per_nm,facecolor='none',edge

    pc = PatchCollection(cell_patches,match_original=True,alpha=1)
```

```
    plt.gca().add_collection(pc)

    # set the limits for the plot
    # set the x axis range
    plt.gca().set_xlim(0, im.shape[1])

    # set the y-axis range and flip the y-axis
    plt.gca().set_ylim(im.shape[0], 0)

    # save this plot to a file
    plt.gca().set_axis_off()

    ↪  plt.savefig(output_data_path+'/'+filename+'_particle_map.png',bbox_inches='tight

def plot_bonds(im,line_segments,bond_list):

    # add the TEM image as the background
    implot = plt.imshow(im)
    implot.set_cmap('gray')

    bond_color_map = create_custom_colormap()

    lc = LineCollection(line_segments,cmap=bond_color_map)
    lc.set_array(np.asarray(bond_list))
    lc.set_linewidth(1)
```

```
    plt.gca().add_collection(lc)
    bond_color_bar = plt.colorbar(lc)

    # set the x axis range
    plt.gca().set_xlim(0, im.shape[1])

    # set the y-axis range and flip the y-axis
    plt.gca().set_ylim(im.shape[0], 0)

    bond_color_bar.ax.set_ylabel('Connection Width (nm)')
    plt.gca().set_axis_off()

def plot_nn_distance(im,line_segments,nn_distance_list):

    nn_dist_cmap = plt.get_cmap('RdBu_r')

    # show the background image
    implot = plt.imshow(im)
    implot.set_cmap('gray')

    nn_lc = LineCollection(line_segments,cmap=nn_dist_cmap)
    nn_lc.set_array(np.asarray(nn_distance_list))
    nn_lc.set_linewidth(1)
    plt.gca().add_collection(nn_lc)
    nn_color_bar = plt.colorbar(nn_lc)
    nn_color_bar.ax.set_ylabel('NN Dist. (nm)')
    plt.gca().set_axis_off()
```

```
    plt.gca().set_title('Neighbor Distance')


def gaussian(x, amp, mean, std):
    return amp/(np.sqrt(2.0*np.pi)*std)*np.exp(-(x-mean)**2/(2.0*std**2))


def square(x, amp, center, width):
    amp = 0.6
    out = np.zeros(len(x))
    peaks = np.linspace(center-width,center+width,10)
    for peak in peaks:
        out += gaussian(x,amp*np.sqrt(2.0*np.pi)*width/10,peak,width/10)

    return out




parser = argparse.ArgumentParser()
parser.add_argument('-b','--black', help='black background',
 ↪  action='store_true')
parser.add_argument('-np','--noplot', help='do not plot data',
 ↪  action='store_true')
parser.add_argument('-nd', '--nodata', help='do not output data files
 ↪  (images only)', action='store_true')
parser.add_argument('-m','--morph', help='use morphological filtering',
 ↪  action='store_true')
parser.add_argument('-s','--small', help='look for very small particles (<5
 ↪  pixel diameter)', action='store_true')
```

```
parser.add_argument('-o','--outline', help='draw particle outlines on the
 ↪  image', action='store_true')
parser.add_argument('-v','--voronoi', help='draw the voronoi diagram on the
 ↪  image', action='store_true')
parser.add_argument('-e','--edge', help='plot the symmetry metric of
 ↪  particles at superlattice edge', action='store_true')
parser.add_argument('-mc','--montecarlo', help='caclulate NN dist, Bond
 ↪  width, etc.', action='store_true')
parser.add_argument('-d','--debug', help='turn on debugging output',
 ↪  action='store_true')
parser.add_argument('-a', '--angle', help='plot orientation angle of each
 ↪  Voronoi cell', action='store_true')
parser.add_argument('bondprofile', nargs='?', help='shape of the fitting
 ↪  function for determining bond width (square or gaussian)',
 ↪  default='gaussian')
parser.add_argument('order',nargs='?', help='order of the structure metric',
 ↪  type=int, default=0)
parser.add_argument('img_file',help='image file to analyze')
parser.add_argument('pts_file',nargs='?',help='XY point data
 ↪  file',default='')
parser.add_argument('pix_per_nm',nargs='?',help='scale in pixels per
 ↪  nm',default=0.0,type=float)
args = parser.parse_args()

im = skimio.imread(args.img_file,as_grey=True,plugin='matplotlib')
im_original = np.empty_like(im)
np.copyto(im_original,im)
```

```
if args.debug:
    DEBUG_OUTPUT = True
    print('Debugging output turned on')

if DEBUG_OUTPUT:
    implot = plt.imshow(im)
    implot.set_cmap('gray')
    plt.gca().set_title('Imported Image')
    plt.show()

output_data_path = path.dirname(args.img_file)
filename = str.split(path.basename(args.img_file),'.')[0]

background = 1
if args.black:
    background = 0
    print('User specified black background')

if args.morph:
    print('Using morphological filtering')

if args.edge:
    print('Plotting metric of edge particles')

if args.small:
    print('Looking for small particles')
```

```
pixels_per_nm = args.pix_per_nm

if pixels_per_nm == 0:

    pixels_per_nm,bar_corners = get_image_scale(im)

    if not pixels_per_nm == 0:

        # remove the scalebar from the image
        if background:

            ↪ im[bar_corners[0][0]:bar_corners[1][0]+1,bar_corners[0][1]:bar_corners[1
            ↪ = np.max(im)
        else:

            ↪ im[bar_corners[0][0]:bar_corners[1][0]+1,bar_corners[0][1]:bar_corners[1
            ↪ = 0
    else:
        pixels_per_nm = 1

else:
    print("User specified scale: "+str(pixels_per_nm)+" pixels per nm")

if len(args.pts_file) == 0:
    # find the centroid of each particle in the image
```

```
pts, radii, mask, binary =
↪  get_particle_centers(im,background,pixels_per_nm,args.morph,args.small)

assert len(pts) == len(radii)

# fit a normal distribution function to the data at the mean +/- 3
↪  std.
radii_hist, radii_bins = np.histogram(radii, bins=len(radii)/4,
↪  range=(np.mean(radii)-3.0*np.std(radii),np.mean(radii)+3.0*np.std(radii)))
# trapezoidal approximation
radii_bins += (radii_bins[1]-radii_bins[0])/2
popt, pcov = curve_fit(gaussian, radii_bins[0:(len(radii_bins)-1)],
↪  radii_hist, p0=(np.max(radii_hist),np.mean(radii),np.std(radii)))
fit_amp, fit_mean, fit_std = popt
radii_mean = np.mean(radii)
radii_std = np.std(radii)

particle_data = np.hstack((pts,radii))

# save the input points to a file
if not args.nodata:
    header_string = 'Particle centroids in pixel units\n'
    header_string += 'Particle radii - the average of the major and
    ↪  minor radii of an ellipse fit to the particle area\n'
    header_string += 'total particles: '+str(len(pts))+'\n'
```

```
        header_string += 'mean radius (raw data): %(rad_nm).2g Std.Dev.
        ↪ %(sd_nm).2g (nm), %(rad_px).2g Std.Dev. %(sd_px).2g (pixels)\n'
        ↪ % {'rad_nm':radii_mean, 'sd_nm':radii_std,
        ↪ 'rad_px':radii_mean*pixels_per_nm,
        ↪ 'sd_px':radii_std*pixels_per_nm}
        header_string += 'mean radius (gaussian fit): %(rad_nm).2g Std.Dev.
        ↪ %(sd_nm).2g (nm), %(rad_px).2g Std.Dev. %(sd_px).2g (pixels)\n'
        ↪ % {'rad_nm':fit_mean, 'sd_nm':fit_std,
        ↪ 'rad_px':fit_mean*pixels_per_nm, 'sd_px':fit_std*pixels_per_nm}
        header_string += 'X (pixel)\tY (pixel)\tradius (nm)'

        ↪ np.savetxt(output_data_path+'/'+filename+'_particles.txt',particle_data,fmt=

        ↪ np.savez(output_data_path+'/'+filename+'_particles.npz',pixels_per_nm=pixels
        ↪ centroids=pts, radii=radii)

else:
    print("User specified points")
    pts = np.loadtxt(args.pts_file,skiprows=1,usecols=(1,2),ndmin=2)
    radii = []
    mask = np.ones(im.shape, dtype='i4')

# ideal square grid test case
#x = np.linspace(-0.5, 2.5, 5)
#y = np.linspace(-0.5, 2.5, 5)
#xx, yy = np.meshgrid(x, y)
#xy = np.c_[xx.ravel(), yy.ravel()]
```

```
#pts = xy

if not args.noplot:
    ax = plt.figure(0)

    ↪  plt.hist(2.0*radii,range=(2.0*(max(0,np.mean(radii)-3.0*np.std(radii))),2.0*(np.
    fit_hist = gaussian(2.0*radii_bins, 2.0*fit_amp, 2.0*fit_mean,
    ↪  2.0*fit_std)
    plt.plot(2.0*radii_bins, fit_hist, 'r-', linewidth=3)
    label_string = '%(count)i particles, Diameter: %(mean).3g (nm),
    ↪  $\sigma$: %(sd).2g (nm), %(percent).2g%%' % {'count':len(radii),
    ↪  'mean':2.0*fit_mean, 'sd':2.0*fit_std,
    ↪  'percent':100.0*fit_std/fit_mean }
    plt.gca().set_title(label_string)
    plt.xlabel('Diameter (nm)')
    plt.ylabel('Count')
    plt.savefig(output_data_path+'/'+filename+'_diameter_hist.png',
    ↪  bbox_inches='tight')

vor = Voronoi(pts)

bond_order = args.order

if bond_order < 0:
    raise ValueError('order parameter should be > 0')

if bond_order > 0:
```

```
    # order returns a list with region_index, order, orientation
    msm =
     ↪  order(vor.vertices,vor.regions,bond_order,(im.shape[1],im.shape[0]))

    if np.any(np.isnan(np.asarray(msm)[:,1])):
        raise ValueError('nan found in order parameter array')
else:
    print("No bond order given, only the particle locations will be found")

if not args.noplot:

    if args.outline:
        plt.figure(1)
        plt.subplot(111)
        implot = plt.imshow(im_original)
        implot.set_cmap('gray')
        plot_particle_outlines(im, pts, radii, pixels_per_nm)

    if bond_order > 0:
        plt.figure(1)
        plt.clf()
        plt.subplot(111)
        implot = plt.imshow(im_original)
        implot.set_cmap('gray')
        symmetry_colormap = plt.get_cmap('RdBu_r')
        plot_symmetry(im, msm, vor, bond_order, symmetry_colormap, mask,
         ↪  args.voronoi, args.edge)
```

```
        if args.angle:
            plt.figure(1)
            plt.clf()
            plt.subplot(111)
            implot = plt.imshow(im_original)
            implot.set_cmap('gray')
            orientation_colormap = plt.get_cmap('RdBu_r')
            plot_orientation(im,msm,vor, pts, radii, pixels_per_nm,
             ↪  orientation_colormap,mask,args.edge)


# save the metrics to a file
if bond_order > 0:
    if not args.nodata:
        header_string =     str(bond_order)+'-fold order parameter
         ↪  (Psi'+str(bond_order)+')\n'
        header_string +=    'References: Mickel, W., J. Chem. Phys. (2013),
         ↪  Escobedo, F., Soft Matter (2011)\n'
        header_string +=    'length: '+str(len(msm))+'\n'
        header_string +=    'region_index\tPsi'+str(bond_order)+'\n'
        header_string +=    'orientation (deg)'

         ↪  np.savetxt(output_data_path+'/'+filename+'_Psi'+str(bond_order)+'_data.txt',

# the rest of this should be moved to another file
if bond_order > 0:
```

```
if args.montecarlo:

    # make sure that im is not altered
    if background:
        # invert a light background image
        im_greyscale = np.abs(im_original-np.max(im_original))
    else:
        im_greyscale = im_original

    max_image_intensity = np.max(im_original)

    # Calculate the "bond strengths"
    # use the binary from get_particles above instead of:
    # make_binary_image(im,background,2*pixels_per_nm,adaptive=1)
    binary_im = binary
    nn_dist_list = []
    bond_list = []

    # graphs for random access, Monte Carlo?
    bond_graph          =
     ↪  sparse.lil_matrix((len(pts),len(pts)),dtype=np.double)
    distance_x_graph    =
     ↪  sparse.lil_matrix((len(pts),len(pts)),dtype=np.double)
    distance_y_graph    =
     ↪  sparse.lil_matrix((len(pts),len(pts)),dtype=np.double)
    facet_length_graph  =
     ↪  sparse.lil_matrix((len(pts),len(pts)),dtype=np.double)
```

```
# keep track of boundary particles
# naively there is a maximum of len(pts) particles that could be
↪  on the boundary
# since we don't want to allocate all of them or append a site
↪  more than once to
# a list, use a sparse matrix
boundary_sites = sparse.lil_matrix((len(pts),1),dtype=np.int8)

# for saving edge data to file
edges = []

# lists for plotting
bond_list_filtered = []
nn_dist_list_filtered= []
bond_line_segments = []
bond_line_segments_filtered = []
bond_width_segments = []
bond_width_list = []

if args.bondprofile == 'square':
    fit_func = square
elif args.bondprofile == 'gaussian':
    fit_func = gaussian
else:
    print('Bond profile '+args.bondprofile+' not recognized, using
    ↪  gaussian')
```

```
    fit_func = gaussian

# to draw the bonds between points
for ridge_vert_indices,input_pair_indices in
 ↪  zip(vor.ridge_vertices,vor.ridge_points):

    if np.all(np.not_equal(ridge_vert_indices,-1)):

        # get the region enclosing this point
        vertex1 = vor.vertices[ridge_vert_indices[0]]
        vertex2 = vor.vertices[ridge_vert_indices[1]]

        x_vals = zip(vertex1,vertex2)[0]
        y_vals = zip(vertex1,vertex2)[1]

        input_pt1 = pts[input_pair_indices[0]]
        input_pt2 = pts[input_pair_indices[1]]

        # check if the voronoi region vertices are within the
        ↪  image bounds
        if
        ↪  np.all((np.greater(x_vals,0),np.greater(y_vals,0),np.less(x_vals,im.

            # get the nearest neighbor distance
            # the directional x and y distance to move from pt1 to
            ↪  pt2
            nn_x_dist = (input_pt2[0]-input_pt1[0])/pixels_per_nm
```

```
nn_y_dist = (input_pt2[1]-input_pt1[1])/pixels_per_nm

# if the interparticle distance computes to zero
# it is because of the resolution of the image
# therefore set the distance to the uncertainty of the
↪  measurement
# this avoids dropping of zero-valued elements later
↪  in sparse matrix
# operations this should only be an issue for a few
↪  particles with the
# lowest magnification images
if nn_x_dist == 0.0:
    nn_x_dist = 1.0/pixels_per_nm

if nn_y_dist == 0.0:
    nn_y_dist = 1.0/pixels_per_nm

nn_distance = np.sqrt(nn_x_dist**2 + nn_y_dist**2)

# distance_..._graph[i,j] is the distance to move from
↪  point i to point j

↪  distance_x_graph[input_pair_indices[0],input_pair_indices[1]]
↪  = nn_x_dist

↪  distance_x_graph[input_pair_indices[1],input_pair_indices[0]]
↪  = -nn_x_dist
```

```
    ↪ distance_y_graph[input_pair_indices[0],input_pair_indices[1]]
    ↪ = nn_y_dist

    ↪ distance_y_graph[input_pair_indices[1],input_pair_indices[0]]
    ↪ = -nn_y_dist
nn_dist_list.append(nn_distance)

# fit a function to find the width of the bridge
facet_x_dist = np.abs(vertex2[0]-vertex1[0])+1
facet_y_dist = np.abs(vertex2[1]-vertex1[1])+1
range_len = np.max((facet_x_dist,facet_y_dist))
x_range =
    ↪ np.linspace(vertex1[0],vertex2[0],num=range_len)
y_range =
    ↪ np.linspace(vertex1[1],vertex2[1],num=range_len)

x_range = x_range.astype(int)
y_range = y_range.astype(int)

# the old, fast, simple way:
# need to do some trig here...
#bond_width =
    ↪ np.sum(binary_im[y_range,x_range])/pixels_per_nm

bond_width = 0
```

```
# the new, "improved" way, instead of counting the
 ↪  number
# of non-background pixels along the facet line in a
 ↪  binary
# image, fit some profile to the greyscale image

# find the length of the facet in pixel units
facet_length = np.sqrt(float(facet_y_dist)**2 +
 ↪  float(facet_x_dist)**2)

# save this facet length in a sparse matrix

 ↪  facet_length_graph[input_pair_indices[0],input_pair_indices[1]]
 ↪  = facet_length

 ↪  facet_length_graph[input_pair_indices[1],input_pair_indices[0]]
 ↪  = facet_length

# try extending the line in both directions a little
 ↪  to try to
# find some background pixels
extension_factor = 0.05 # extend this much in both
 ↪  directions
phi = np.arctan(float(facet_y_dist)/float(facet_x_dist))
```

```
x_facet_start = int(x_range[0] +
 ↪  np.sign(x_range[0]-x_range[-1]) * np.cos(phi) *
 ↪  (facet_length * extension_factor))
if x_facet_start < 0:
    x_facet_start = 0
elif x_facet_start > im_greyscale.shape[1]-1:
    x_facet_start = im_greyscale.shape[1]-1

y_facet_start = int(y_range[0] +
 ↪  np.sign(y_range[0]-y_range[-1]) * np.sin(phi) *
 ↪  (facet_length * extension_factor))
if y_facet_start < 0:
    y_facet_start = 0
elif y_facet_start > im_greyscale.shape[0]-1:
    y_facet_start = im_greyscale.shape[0]-1

x_facet_end = int(x_range[-1] +
 ↪  np.sign(x_range[-1]-x_range[0]) * np.cos(phi) *
 ↪  (facet_length * extension_factor))
if x_facet_end < 0:
    x_facet_end = 0
elif x_facet_end > im_greyscale.shape[1]-1:
    x_facet_end = im_greyscale.shape[1]-1

y_facet_end = int(y_range[-1] +
 ↪  np.sign(y_range[-1]-y_range[0]) * np.sin(phi) *
 ↪  (facet_length * extension_factor))
```

```
if y_facet_end < 0:
    y_facet_end = 0
elif y_facet_end > im_greyscale.shape[0]-1:
    y_facet_end = im_greyscale.shape[0]-1

range_len = max(np.abs(x_facet_start-x_facet_end)+1,
↪  np.abs(y_facet_start-y_facet_end)+1)

# overwrite x_range, y_range
x_range =
↪  np.asarray(np.round(np.linspace(x_facet_start,x_facet_end,num=ra
y_range =
↪  np.asarray(np.round(np.linspace(y_facet_start,y_facet_end,num=ra

# overwrite the length now that we've extended it
facet_length = facet_length + 2.0 * extension_factor *
↪  facet_length



# don't try to fit a flat line or a very small line
# need at least 3 points to fit a function with 3
↪  parameters
if int(range_len) > 9:

    # now we get the values to fit
    # average some values to the left and right of the
    ↪  facet line
```

```
averaging_steps = np.linspace(-0.1 * facet_length,
↪  0.1 * facet_length, num=2*0.1*facet_length+1)
facet_pixel_values = np.zeros(range_len)
theta = np.arctan2( (y_range[-1]-y_range[0]) ,
↪  (x_range[-1] - x_range[0]) )

bond_area = np.empty((0,2))

for step in averaging_steps:
    # move perpendicular to the facet line
    # a vertical line will only shift in x
    # a horizontal line will only shift in y
    x_shift = int(step * np.sin(theta))
    y_shift = -int(step * np.cos(theta))

    # for plotting in debug code
    bond_area =
    ↪  np.vstack((bond_area,np.vstack((x_range+x_shift,
    ↪  y_range+y_shift)).transpose()))

    # don't let the indices go out of bounds using
    ↪  np.clip
    facet_pixel_values +=
    ↪  im_greyscale[np.clip(y_range + y_shift, 0,
    ↪  im_greyscale.shape[0]-1), np.clip(x_range +
    ↪  x_shift, 0, im_greyscale.shape[1]-1)]
```

```
facet_pixel_values =
↪  facet_pixel_values/len(averaging_steps)



mean_particle_intensity = 0
for i in [0,1]:
    r = radii[input_pair_indices[i]]*pixels_per_nm
    xpts = pts[input_pair_indices[i]][0] +
    ↪  np.arange(-r,r,1)
    ypts = pts[input_pair_indices[i]][1] +
    ↪  np.arange(-r,r,1)

    xpts =
    ↪  np.clip(xpts.astype(int),0,im_greyscale.shape[1]-1)
    ypts =
    ↪  np.clip(ypts.astype(int),0,im_greyscale.shape[0]-1)

    roi = np.meshgrid(ypts, xpts)

    mean_particle_intensity +=
    ↪  np.mean(im_greyscale[roi])

mean_particle_intensity /= 2.0

# normalize the values to help the fitting
facet_pixel_values = (facet_pixel_values -
↪  np.min(facet_pixel_values))/mean_particle_intensity
```

```
# fit a square function to that vector of intensity
↪ values
# square(x, amp, center, width)
# gaussian(x, amp, mean, std)
# width is the extent of the square function on
↪ either side of center
width_guess = facet_length/4.0
center_guess = facet_length/2.0
amp_guess = facet_pixel_values.max()

# debug to find problematic values
# print(params,len(facet_pixel_values))

try:
    fit_window = np.linspace(0,facet_length,
    ↪ num=len(facet_pixel_values))
    params = (amp_guess, center_guess, width_guess)
    popt, pcov = curve_fit(fit_func, fit_window,
    ↪ facet_pixel_values, p0=params)

    if fit_func == square:
        bond_width = popt[2]*2.0/pixels_per_nm
        bond_center = popt[1]
        fit_curve = square(fit_window, popt[0],
        ↪ popt[1], popt[2])
```

```
elif fit_func == gaussian:
    # FWHM from gaussian
    # might need to modify this because the
    ↪ background isn't always near 0 if the
    ↪ image is not binary
    bond_width =
    ↪ 2.0*np.sqrt(2.0*np.log(2))*popt[2]/pixels_per_nm
    bond_center = popt[1]
    fit_curve = gaussian(fit_window, popt[0],
    ↪ popt[1], popt[2])


# do a linear regression to find the R**2 value
↪ for the fit
# bad fits will be thrown out
s, i, r, p, se =
↪ linregress(facet_pixel_values,fit_curve)


# for plotting the bond
x_bond_start = int(x_range[0] +
↪ np.sign(x_range[-1]-x_range[0]) *
↪ np.cos(phi) * (bond_center -
↪ bond_width/2*pixels_per_nm))
```

```
y_bond_start = int(y_range[0] +
↪  np.sign(y_range[-1]-y_range[0]) *
↪  np.sin(phi) * (bond_center -
↪  bond_width/2*pixels_per_nm))

x_bond_end = int(x_range[0] +
↪  np.sign(x_range[-1]-x_range[0]) *
↪  np.cos(phi) * (bond_center +
↪  bond_width/2*pixels_per_nm))
y_bond_end = int(y_range[0] +
↪  np.sign(y_range[-1]-y_range[0]) *
↪  np.sin(phi) * (bond_center +
↪  bond_width/2*pixels_per_nm))



if DEBUG_OUTPUT:
    # use this to look at the fitting of bond
    ↪  width
    # plot the pixel values and the fit curve
    plt.figure(figsize=(25,7), dpi=150 )
    plt.subplot(1, 3, 1)

    ↪  plt.scatter(np.arange(len(x_range)),facet_pixel_valu


    ↪  plt.plot(np.arange(len(x_range)),fit_curve,'r-')
```

```
# plot the image where the bond is, overlay
 ↪  the facet line and the bond width
plt.subplot(1, 3, 2)
plt.imshow(im_greyscale, cmap='gray',
 ↪  interpolation='none')
plt.scatter(x_range,y_range)
#plt.scatter(bond_area[:,0],
 ↪  bond_area[:,1], edgecolor='none',
 ↪  alpha=0.1)

# plot a line showing the fit bond_width
bond_len =
 ↪  int(max(np.abs(x_bond_start-x_bond_end),np.abs(y_bon
bond_x = np.linspace(x_bond_start,
 ↪  x_bond_end, num=bond_len)
bond_y = np.linspace(y_bond_start,
 ↪  y_bond_end, num=bond_len)


# just show the part of the image where the
 ↪  bond is

 ↪  plt.gca().set_xlim(np.min(x_range)-range_len,
 ↪  np.max(x_range)+range_len)

 ↪  plt.gca().set_ylim(np.min(y_range)-range_len,
 ↪  np.max(y_range)+range_len)
```

```
        plt.gca().set_title('Facet Length: %(facet)i
        ↪ R^2: %(rs).3f\n Bond Width: %(bond)i
        ↪ (pixels), Center: %(center)i' %
        ↪ {'facet':facet_length, 'rs':r**2,
        ↪ 'bond':bond_width*pixels_per_nm,
        ↪ 'center':bond_center})

        # plot the image where the bond is, overlay
        ↪ the bond width
        plt.subplot(1, 3, 3)
        plt.imshow(im_greyscale, cmap='gray',
        ↪ interpolation='none')
        plt.plot(bond_x, bond_y, 'r-')

        # just show the part of the image where the
        ↪ bond is

        ↪ plt.gca().set_xlim(np.min(x_range)-range_len,
        ↪ np.max(x_range)+range_len)

        ↪ plt.gca().set_ylim(np.min(y_range)-range_len,
        ↪ np.max(y_range)+range_len)

        plt.show()

    # sanity checks
    if bond_width*pixels_per_nm > facet_length:
```

```
            # we'll allow the width to be a little more
             ↪ than
            # the fitting window, because the window
             ↪ doesn't
            # always span the entire bond, but the fit
             ↪ can sometimes
            # be a good estimate based on the curvature
             ↪ of the bond
            bond_width = 0

        if bond_center < 0 or bond_center >
         ↪ facet_length:
            # if the center of the bond is outside of
             ↪ the fitting window,
            # it's probably a bad fit, throw it out
            bond_width = 0

        if r**2 < 0.6:
            # bad fit
            bond_width = 0

        # save the bond locations as line segments for
         ↪ output in an image
        if bond_width:

            ↪ bond_width_segments.append(np.asarray([(x_bond_start
            bond_width_list.append(bond_width)
```

```
    except RuntimeError:
        # this happens if the curve_fit method doesn't
        ↪ converge
        # the bond width will be zero in this case
        pass


bond_graph[input_pair_indices[0],input_pair_indices[1]]
↪ = bond_width
bond_graph[input_pair_indices[1],input_pair_indices[0]]
↪ = bond_width
bond_list.append(bond_width)

# make the line segments for plotting bonds, neighbor
↪ distances, whatever

↪ bond_line_segments.append(np.asarray([pts[input_pair_indices[0]]


↪ edges.append([input_pair_indices[0],input_pair_indices[1],nn_dis

if not bond_width == 0:
    bond_list_filtered.append(bond_width)
    nn_dist_list_filtered.append(nn_distance)

    ↪ bond_line_segments_filtered.append(np.asarray([pts[input_pai
```

```
            else:
                # at least one ridge vertex is off the image
                # these two input points are boundary
                # boundary_sites is an Nx1 matrix
                boundary_sites[input_pair_indices,0] = np.int(1)
        else:
            # why are these no good?
            # are they on the boundary?
            boundary_sites[input_pair_indices,0] = np.int(1)

if not args.noplot:

    # fit a gaussian to the bond width histogram data
    bond_hist_window = 5.0*np.std(bond_list_filtered)
    bond_hist, bond_hist_bins = np.histogram(bond_list_filtered,
    ↪  bins=len(bond_list_filtered)/4,
    ↪  range=(np.mean(bond_list_filtered)-bond_hist_window,np.mean(bond_list_fi

    # shift bin locations from edge to center
    bond_hist_bins += (bond_hist_bins[1]-bond_hist_bins[0])/2
    popt, pcov = curve_fit(gaussian,
    ↪  bond_hist_bins[0:len(bond_hist_bins)-1], bond_hist,
    ↪  p0=(np.max(bond_hist),np.mean(bond_list_filtered),np.std(bond_list_filte
    fit_amp, fit_mean, fit_std = popt
```

```
# do some filtering to set any bonds too far from the mean
 ↪  bond width to zero
# this is for calculating the connectivity, to remove "bonds"
 ↪  that aren't real

# only keep bonds within this many std. deviations of the mean
bond_distribution_window = 5.0 * fit_std

good_bond_indices =
 ↪  np.nonzero(np.abs(bond_list_filtered-fit_mean) <
 ↪  bond_distribution_window)[0]
bad_bond_indices = np.nonzero(np.abs(bond_list-fit_mean) >
 ↪  bond_distribution_window)[0]

nn_dist_list_filtered =
 ↪  np.asarray(nn_dist_list_filtered)[good_bond_indices]
bond_line_segments_filtered =
 ↪  np.asarray(bond_line_segments_filtered)[good_bond_indices]
bond_list_filtered =
 ↪  np.asarray(bond_list_filtered)[good_bond_indices]

# plot the image with lines between each particle center,
 ↪  color coded with the bond width
plt.figure(3)

 ↪  plot_bonds(im_original,bond_line_segments_filtered,bond_list_filtered)
```

```
↪ plt.savefig(output_data_path+'/'+filename+'_bond_map.png',bbox_inches='t
↪ dpi=300)

# plot the image with lines along the cross-section of each
↪ bond
plt.figure(4)
plot_bonds(im_original,bond_width_segments,bond_width_list)

↪ plt.savefig(output_data_path+'/'+filename+'_bond_width_map.pdf',bbox_inc
↪ dpi=1200)

# make a map of the nn distances
plt.figure(5)

↪ plot_nn_distance(im_original,bond_line_segments_filtered,nn_dist_list_fi

↪ plt.savefig(output_data_path+'/'+filename+'_nn_dist_map.png',bbox_inches
↪ dpi=300)

# plot a histogram of the bond widths
plt.figure(6)
plt.hist(bond_list_filtered,bins=len(bond_list_filtered)/4)
fit_curve = gaussian(bond_hist_bins,fit_amp,fit_mean,fit_std)
plt.plot(bond_hist_bins, fit_curve, 'r-', linewidth=3)
```

```
    label_string = '%(count)i bonds, Width: %(mean).3g (nm),
    ↪ $\sigma$: %(sd).2g (nm), %(percent).2g%%' %
    ↪ {'count':len(bond_list_filtered), 'mean':fit_mean,
    ↪ 'sd':fit_std, 'percent':100.0*fit_std/fit_mean }
    plt.gca().set_title(label_string)
    plt.ylabel('Count')
    plt.xlabel('Connection Width (nm)')
    plt.gca().set_title(label_string)
    plt.savefig(output_data_path+'/'+filename+'_bond_hist.png',
    ↪ bbox_inches='tight')

    # plot a histogram of the nearest neighbor distances
    plt.figure(7)
    plt.hist(nn_dist_list,bins=len(nn_dist_list)/4)
    plt.ylabel('Count')
    plt.xlabel('Neighbor Distance (nm)')
    plt.savefig(output_data_path+'/'+filename+'_nn_dist_hist.png',
    ↪ bbox_inches='tight')

# save edge data to file
header_string =     'pt1 and pt2 are the indices of the points
↪ between which the distance and bond width are given\n'
header_string +=    'total edges: '+str(len(edges))+'\n'
header_string +=    'pt1\tpt2\tdistance (nm)\tbond width (nm)'

↪ np.savetxt(output_data_path+'/'+filename+'_edges.txt',np.asarray(edges),fmt=
```

```
        # save the graphs to files to use in Monte Carlo
        bond_graph_csr = bond_graph.tocsr()
        distance_x_graph_csr = distance_x_graph.tocsr()
        distance_y_graph_csr = distance_y_graph.tocsr()
        boundary_sites_csc = boundary_sites.tocsc()
        facet_length_csc = facet_length_graph.tocsc()
        bond_graph_csc = bond_graph.tocsc()
        radii = np.asarray(radii.reshape((-1,)),dtype=np.double)

        ↪ np.savez(output_data_path+'/'+filename+'_bond_graph',bond_graph=bond_graph_c

        ↪ np.savez(output_data_path+'/'+filename+'_bond_graph_csc',data=bond_graph_csc
        ↪ indices=bond_graph_csc.indices, indptr=bond_graph_csc.indptr)

        ↪ np.savez(output_data_path+'/'+filename+'_x_distance_graph',data=distance_x_g

        ↪ np.savez(output_data_path+'/'+filename+'_y_distance_graph',data=distance_y_g

        ↪ np.savez(output_data_path+'/'+filename+'_boundary_graph',data=boundary_sites

        ↪ np.savez(output_data_path+'/'+filename+'_facet_length_graph',data=facet_leng

        ↪ np.savez(output_data_path+'/'+filename+'_site_data',radii=radii,pts=pts,pixe

# order_parameter.pyx
# Calculates the order parameter for a collection of points in 2D space
```

```
# Reference: Mickel, Walter, et al. "Shortcomings of the bond
↪ orientational order parameters for the analysis of disordered
↪ particulate matter." The Journal of chemical physics (2013)
# Reference: Carlos Avenda˜no and Fernando A. Escobedo, "Phase behavior of
↪ rounded hard spheres" Soft Matter 2011
# This file is written in the Cython language for faster execution than
↪ regular Python
# 20140902 Kevin Whitham, Ben Savitzky, Cornell University

import numpy as np
cimport numpy as np
cimport cython
from math import acos
from cpython cimport bool
from scipy.special import sph_harm

DUBTYPE = np.double
SINTYPE = np.float32

ctypedef np.double_t DUBTYPE_t
ctypedef np.float32_t SINTYPE_t

# returns the angle in radians of the interior angle made by 3 points
cdef inline double angle(double x1, double y1, double x2, double y2, double
↪ x3, double y3):

    cdef double len1,len2
```

```
    # change coordinates to put pt2 at the origin
    x1 = x1-x2
    y1 = y1-y2
    x3 = x3-x2
    y3 = y3-y2

    x2 = 0
    y2 = 0


    inner_product = x1*x3 + y1*y3
    len1 = np.sqrt(np.abs(x1)**2 + np.abs(y1)**2)
    len2 = np.sqrt(np.abs(x3)**2 + np.abs(y3)**2)
    if len1 > 0 and len2 > 0:
        return acos(inner_product/(len1*len2))
    else:
        return 0

# Calculates the order parameter of order l (Psi_4, Psi_6, etc.) for each
 ↪  Voronoi cell in vor
#         ignoring cells with vertices less than zero or greater than
 ↪  limits
# Also returns an angle that describes the orientation of each cell
 ↪  relative to image horizontal
@cython.boundscheck(False)
```

```
cpdef minkowski(np.ndarray[DUBTYPE_t, ndim=2] vor_vertices, vor_regions, int
↪  l, limits):

    cdef double x_max, y_max
    x_max = limits[0]
    y_max = limits[1]

    msm = []

    # Cython static type definitions
    cdef unsigned int region_index, region_vert_index, vert_count,
    ↪  facet_index
    cdef int m
    cdef double x1, y1, x2, y2, x3, y3, rotation, cell_perimeter, sum1_real,
    ↪  sum1_imag, sum2, zero_sum, orientation_angle
    cdef bool out_of_bounds

    # make a 1-D arrays to hold information about each facet
    # assume all regions have fewer than 20 facets
    cdef np.ndarray[DUBTYPE_t, ndim=1] facet_lengths       =
    ↪  np.zeros(20,dtype=DUBTYPE)
    cdef np.ndarray[DUBTYPE_t, ndim=1] facet_normal_angles =
    ↪  np.zeros(20,dtype=DUBTYPE)
    cdef np.ndarray[DUBTYPE_t, ndim=1] global_facet_angles =
    ↪  np.zeros(20,dtype=DUBTYPE)
    cdef np.ndarray[DUBTYPE_t, ndim=1] interior_angles     =
    ↪  np.zeros(20,dtype=DUBTYPE)
```

```
cdef np.ndarray[DUBTYPE_t, ndim=1] region              =
↪  np.zeros(20,dtype=DUBTYPE)

# get the vertices for each Voronoi cell
# region contains the indices of the vertices of the cell
for region_index in range(len(vor_regions)):

    vert_count = len(vor_regions[region_index])

    if vert_count <= 20:

        # initialize arrays for this region
        facet_lengths       = np.zeros(20,dtype=DUBTYPE)
        facet_normal_angles = np.zeros(20,dtype=DUBTYPE)
        global_facet_angles = np.zeros(20,dtype=DUBTYPE)
        interior_angles     = np.zeros(20,dtype=DUBTYPE)
        region              = np.zeros(20,dtype=DUBTYPE)

        # copy from vor_regions list to region array
        for region_vert_index in range(vert_count):
            region[region_vert_index] =
            ↪  vor_regions[region_index][region_vert_index]

        # clear the perimeter value
        cell_perimeter = 0

        # skip infinite cells and empty regions
```

```
if vert_count and np.all(np.not_equal(region[:vert_count],-1)):

    # find center of the region for global facet angle
    ↪ calculation
    x_center =
    ↪ np.mean(vor_vertices[np.asarray(region[0:vert_count-1],dtype='int')]
    y_center =
    ↪ np.mean(vor_vertices[np.asarray(region[0:vert_count-1],dtype='int')]

    for region_vert_index in range(vert_count):

        x1,y1 = vor_vertices[region[region_vert_index]]
        x2,y2 = vor_vertices[region[<unsigned
        ↪ int>((region_vert_index+1) % vert_count)]]
        x3,y3 = vor_vertices[region[<unsigned
        ↪ int>((region_vert_index+2) % vert_count)]]

        # check if any points are off the image
        if x1 >= 0 and x2 >= 0 and y1 >= 0 and y2 >= 0 and x1 <=
        ↪ x_max and x2 <= x_max and y1 <= y_max and y2 <=
        ↪ y_max:

            out_of_bounds = False


            # euclidean distance
```

```
facet_lengths[region_vert_index] =
↪  np.sqrt(np.abs(x1-x2)**2 + np.abs(y1-y2)**2)

# find the angle of the facets relative to the
↪  first facet

interior_angles[region_vert_index] =
↪  angle(x1,y1,x2,y2,x3,y3)

# the angle of the facet vertex2 to vertex3
# relative to the facet vertex1 to vertex2
# is 180-90-interior_angle + the sum of all
↪  previous interior angles
#if (region_vert_index+1) < vert_count:
#rotation =
↪  np.pi-interior_angles[region_vert_index]
#facet_normal_angles[<unsigned
↪  int>((region_vert_index+1) % vert_count)] =
↪  ((facet_normal_angles[region_vert_index]+rotation)
↪  % (2.0*np.pi))
facet_normal_angles[<unsigned
↪  int>((region_vert_index+1) % vert_count)] =
↪  ((interior_angles[region_vert_index] +
↪  facet_normal_angles[region_vert_index]) %
↪  (2.0*np.pi))

# add to the cell perimeter
```

```
                cell_perimeter += facet_lengths[region_vert_index]

                # find the angle of each facet relative to the
                ↪ horizontal axis of the matrix
                # use these to find the orientation of the cell
                ↪ relative to the matrix axes
                x_midpoint = min(x1,x2) + np.abs(x1-x2)/2.0
                y_midpoint = min(y1,y2) + np.abs(y1-y2)/2.0
                global_facet_angles[region_vert_index] =
                ↪ angle(x_center+1,y_center,x_center,y_center,x_midpoint,y_mid

                if y_midpoint < y_center:
                    global_facet_angles[region_vert_index] *= -1.0

            else:

                out_of_bounds = True
                break

        if not out_of_bounds:

            # choose the angle closest to horizontal
            orientation_angle =
            ↪ global_facet_angles[np.argmin(np.arctan(np.abs(np.tan(global_fac
            ↪ * 180.0 / np.pi

            sum = 0.0
```

```
                for facet_index in range(vert_count):
                    sum += facet_lengths[facet_index]/cell_perimeter *
                    ↪  np.exp(1.j * l *
                    ↪  facet_normal_angles[facet_index])

                msm.append([region_index,np.abs(sum),orientation_angle])
    return msm
```

B.2 Python Code for Calculating Radial Distribution Function

The following code imports information about the locations of NCs and generates a plot showing the measured radial distribution function. There is an option to fit a radial distribution function to the measured one to quantify disorder according to a static or ideal paracrystalline model.

```
# fit_rdf.py
# convenience functions for running a radial distribution function (rdf)
↪  calculation on a set of
# points and fitting the rdf to a model to quantify disorder
# 20151014 Kevin Whitham, Cornell University
# License: GNU Public License (GPL) v.3.0

import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
```

```
import plotsettings
import prettyplotlib as ppl
import seaborn as sns
from paircorrelation import PairCorrelationFunction_2D
from paircorrelation import fit_hex_paracrystal_rdf,
↪ fit_square_paracrystal_rdf, fit_hex_disordered_rdf,
↪ fit_square_disordered_rdf
from paircorrelation import generate_paracrystal_rdf,
↪ generate_disordered_rdf
import argparse
from os import path

parser = argparse.ArgumentParser()
parser.add_argument('symmetry', help='4 = square, 6 = hexagonal', type=int,
↪ default=4)
parser.add_argument('model', help='crystal disorder model for fitting:
↪ \'uniform\', \'para\'', default='uniform')
parser.add_argument('distance', help='cut off distance in number of particle
↪ diameters', type=float, default=20.0)
parser.add_argument('input_file',help='Image file')
args = parser.parse_args()

output_data_path = path.dirname(args.input_file)+'/'
filename = str.split(path.basename(args.input_file),'.')[0]

pts_file = np.load(output_data_path+filename+'_particles.npz')
pts = pts_file['centroids']
```

```
pixels_per_nm = pts_file['pixels_per_nm']
radii = pts_file['radii']
nn_dist = 2.0*np.mean(radii)*pixels_per_nm

im = plt.imread(args.input_file)
box_size = np.sqrt(float(im.shape[0])*float(im.shape[1]))

g,r,x,y = PairCorrelationFunction_2D(pts[:,0],pts[:,1], box_size,
↪  min(box_size/2.0,args.distance*nn_dist), 0.5)

# change units from pixels to nm
r /= pixels_per_nm

from scipy.optimize import curve_fit

# disordered, paracrystal, mixed
fit_functions = [[fit_square_disordered_rdf, fit_square_paracrystal_rdf], #
↪  4-fold
                 [fit_hex_disordered_rdf, fit_hex_paracrystal_rdf]]       #
                 ↪  6-fold

if args.symmetry == 4:
    fit_func_row = 0
    symmetry_label = 'square'
elif args.symmetry == 6:
    fit_func_row = 1
    symmetry_label = 'hex'
```

```
else:
    print('Symmetry not recognized')
    symmetry_label = ''

if args.model == 'uniform':
    fit_func_col = 0
    params = [0.05*r[np.argmax(g)], r[np.argmax(g)]]
elif args.model == 'para':
    fit_func_col = 1
    params = [0.05*r[np.argmax(g)], r[np.argmax(g)]]
else:
    print('Model not recognized')

fit_function = fit_functions[fit_func_row][fit_func_col]

popt, pcov = curve_fit(fit_function, r, g, p0=params)
print(popt)
print(pcov)

# fit results
# a is the length of the first basis vector
# b is the length of the second basis vector
# popt[0] is the std. dev.

# DEBUG!!!
# you can force the values here instead of using the fit value
#print('Overriding fit results!')
```

```
#popt[1] = 6.61;
#popt[0] = 0.035*popt[1]

if args.symmetry == 4:
    a=(popt[1],0.0)
    b=(0.0,popt[1])
elif args.symmetry == 6:
    a=(popt[1],0.0)
    b=(popt[1]*-0.5,popt[1]*np.sqrt(3.0)/2.0)

if args.model == 'uniform':
    full_model_string = 'Uncorrelated Disorder'
    r_fit, g_fit = generate_disordered_rdf(a, b, popt[0],
    ↪  max_distance=np.max(r), resolution=r[1]-r[0])
    func_label = r'$\sum_i\/\frac{1}{2\pi r_i \sigma}
    ↪  exp\left(\frac{-(r-r_i)^2}{2 \sigma^2}\right)$'
elif args.model == 'para':
    full_model_string = 'Paracrystal'
    r_fit, g_fit = generate_paracrystal_rdf(a, b, popt[0],
    ↪  max_distance=np.max(r), resolution=r[1]-r[0])
    func_label = r'$\sum_i\/\frac{L}{2\pi r_i^2 \sigma}
    ↪  exp\left(\frac{-(r-r_i)^2}{2 (\sigma r_i/L)^2}\right)$'
else:
    print('Model not recognized')

pub = plotsettings.Set('Nature')
pub.set_figsize(n_columns = 1, n_rows = 1)
```

```
plt.figure(1)
#plt.plot(r,g,'r-',label='Data')
#plt.plot(r_fit, g_fit, 'k-', label=func_label+' £\sigma£ =
↪  '+'%(std).2f%%, L = %(L).2f nm' % {'std':popt[0]/popt[1]*100.0,
↪  'L':popt[1]})

sns.set_style("white")
sns.despine()

# make text editable in .pdf files
mpl.rc('pdf', fonttype=42)

ppl.plot(r,g,label='Data')
#ppl.plot(r_fit, g_fit, label=func_label+' £\sigma£ = '+'%(std).2f%%, L =
↪  %(L).2f nm' % {'std':popt[0]/popt[1]*100.0, 'L':popt[1]})
ppl.plot(r_fit, g_fit, label=full_model_string)
plt.xlabel('r (nm)')
plt.ylabel('g(r)')

mpl.rc('pdf', fonttype=3)

plt.gca().legend()
plt.savefig(output_data_path+filename+'_'+args.model+'_'+symmetry_label+'_RDF.pdf',
↪  bbox_inches='tight')

header_string = 'Radial distribution function\n'
```

```
header_string += 'Model: '+args.model+' Equation: '+func_label+'\n'
header_string += 'sigma = '+'%(std).2f%%, L = %(L).2f nm\n' %
↪  {'std':popt[0]/popt[1]*100.0, 'L':popt[1]}
header_string += 'r (nm)\tg(r)'
np.savetxt(output_data_path+filename+'_'+args.model+'_'+symmetry_label+'_RDF.txt',zip(r,

# paircorrelation.py
# Contains functions for calculating the pair correlation function
# (radial distribution function (RDF)) for an array of points
# Also contains functions for fitting the RDF to models for
# paracrystalline or uniform disorder

"""
https://github.com/cfinch/Shocksolution_Examples/tree/master/PairCorrelation
This module contains routines for analyzing distributions of particles.
↪  Currently,
this consists of:
    PairCorrelationFunction_2D
    PairCorrelationFunction_3D
    computePhi
        Returns True if the given point at (x,y) lies within any of the
↪  particles with
        centers given by (x_array, y_array) with radius "radius."
↪  Otherwise, returns False.
    computePhi_PBC
    compute_AVF
"""
```

```
def PairCorrelationFunction_2D(x,y,S,rMax,dr):
    """Compute the two-dimensional pair correlation function, also known
    as the radial distribution function, for a set of circular particles
    contained in a square region of a plane.  This simple function finds
    reference particles such that a circle of radius rMax drawn around the
↪
    particle will fit entirely within the square, eliminating the need to
    compensate for edge effects.  If no such particles exist, an error is
    returned. Try a smaller rMax...or write some code to handle edge
↪   effects! ;)

    Arguments:
        x               an array of x positions of centers of particles
        y               an array of y positions of centers of particles
        S               length of each side of the square region of the
↪   plane
        rMax            outer diameter of largest annulus
        dr              increment for increasing radius of annulus

    Returns a tuple: (g, radii, interior_x, interior_y)
        g(r)            a numpy array containing the correlation function
↪   g(r)
        radii           a numpy array containing the radii of the
                        annuli used to compute g(r)
        interior_x      x coordinates of reference particles
        interior_y      y coordinates of reference particles
```

```
    """
    from numpy import zeros, sqrt, where, pi, average, arange, histogram
    # Number of particles in ring/area of ring/number of reference
    ↪ particles/number density
    # area of ring = pi*(r_outer**2 - r_inner**2)

    # Find particles which are close enough to the box center that a
    ↪ circle of radius
    # rMax will not cross any edge of the box
    bools1 = x>1.1*rMax
    bools2 = x<(S-1.1*rMax)
    bools3 = y>rMax*1.1
    bools4 = y<(S-rMax*1.1)
    interior_indices, = where(bools1*bools2*bools3*bools4)
    num_interior_particles = len(interior_indices)

    print "Analyzing "+str(num_interior_particles)+" particles out of
    ↪ "+str(len(x))

    if num_interior_particles < 1:
        raise  RuntimeError ("No particles found for which a circle of
        ↪ radius rMax\
                will lie entirely within a square of side length S.
↪   Decrease rMax\
                or increase the size of the square.")

    edges = arange(0., rMax+1.1*dr, dr)
```

```
    num_increments = len(edges)-1
    g = zeros([num_interior_particles, num_increments])
    radii = zeros(num_increments)
    numberDensity = len(x)/S**2

    # Compute pairwise correlation for each interior particle
    for p in range(num_interior_particles):
        index = interior_indices[p]
        d = sqrt((x[index]-x)**2 + (y[index]-y)**2)
        d[index] = 2*rMax

        (result,bins) = histogram(d, bins=edges, normed=False)
        g[p,:] = result/numberDensity

    # Average g(r) for all interior particles and compute radii
    g_average = zeros(num_increments)
    for i in range(num_increments):
        radii[i] = (edges[i] + edges[i+1])/2.
        rOuter = edges[i+1]
        rInner = edges[i]
        g_average[i] = average(g[:,i])/(pi*(rOuter**2 - rInner**2))

    return (g_average, radii, x[interior_indices], y[interior_indices])
####

def PairCorrelationFunction_3D(x,y,z,S,rMax,dr):
```

```
    """Compute the three-dimensional pair correlation function for a set
    ↪ of
    spherical particles contained in a cube with side length S.  This
↪   simple
    function finds reference particles such that a sphere of radius rMax
↪   drawn
    around the particle will fit entirely within the cube, eliminating the
↪   need
    to compensate for edge effects.  If no such particles exist, an error
↪   is
    returned.  Try a smaller rMax...or write some code to handle edge
↪   effects! ;)

    Arguments:
        x               an array of x positions of centers of particles
        y               an array of y positions of centers of particles
        z               an array of z positions of centers of particles
        S               length of each side of the cube in space
        rMax            outer diameter of largest spherical shell
        dr              increment for increasing radius of spherical shell

    Returns a tuple: (g, radii, interior_x, interior_y, interior_z)
        g(r)            a numpy array containing the correlation function
↪   g(r)
        radii           a numpy array containing the radii of the
                        spherical shells used to compute g(r)
        interior_x      x coordinates of reference particles
```

```
        interior_y      y coordinates of reference particles
        interior_z      z coordinates of reference particles
    """
    from numpy import zeros, sqrt, where, pi, average, arange, histogram

    # Find particles which are close enough to the cube center that a
    ↪  sphere of radius
    # rMax will not cross any face of the cube
    bools1 = x>rMax
    bools2 = x<(S-rMax)
    bools3 = y>rMax
    bools4 = y<(S-rMax)
    bools5 = z>rMax
    bools6 = z<(S-rMax)

    interior_indices, = where(bools1*bools2*bools3*bools4*bools5*bools6)
    num_interior_particles = len(interior_indices)

    if num_interior_particles < 1:
        raise  RuntimeError ("No particles found for which a sphere of
        ↪  radius rMax\
                will lie entirely within a cube of side length S.  Decrease
↪   rMax\
                or increase the size of the cube.")

    edges = arange(0., rMax+1.1*dr, dr)
    num_increments = len(edges)-1
```

```
    g = zeros([num_interior_particles, num_increments])
    radii = zeros(num_increments)
    numberDensity = len(x)/S**3

    # Compute pairwise correlation for each interior particle
    for p in range(num_interior_particles):
        index = interior_indices[p]
        d = sqrt((x[index]-x)**2 + (y[index]-y)**2 + (z[index]-z)**2)
        d[index] = 2*rMax

        (result,bins) = histogram(d, bins=edges, normed=False)
        g[p,:] = result/numberDensity

    # Average g(r) for all interior particles and compute radii
    g_average = zeros(num_increments)
    for i in range(num_increments):
        radii[i] = (edges[i] + edges[i+1])/2.
        rOuter = edges[i+1]
        rInner = edges[i]
        g_average[i] = average(g[:,i])/(4./3.*pi*(rOuter**3 - rInner**3))

    return (g_average, radii, x[interior_indices], y[interior_indices],
    ↪ z[interior_indices])
    # Number of particles in shell/total number of particles/volume of
    ↪ shell/number density
    # shell volume = 4/3*pi(r_outer**3-r_inner**3)
####
```

```
def computePhi(x_grid,y_grid,x_array,y_array,r):
    """Returns True if the given point at (x,y) lies within any of the
    ↪ particles with
    centers given by (x_array, y_array) with radius "radius." Otherwise,
↪ returns False.
    """
    from scipy import weave

    numParticles = len(x_array)
    gridsize = len(x_grid)

    code = r"""
           double d2;
           double radius_squared = 4.0*pow(r,2);
           int avail = gridsize*gridsize;
           bool collision_not_found = true;
           int p = 0;

            for (int i=0; i<gridsize; i++) {
                for (int j=0; j<gridsize; j++) {
                    p = 0;
                    collision_not_found = true;
                    while ((p<numParticles) && collision_not_found) {
                        if ((pow(x_grid[i]-x_array[p], 2) +
↪ pow(y_grid[j]-y_array[p],2)) < radius_squared) {
                            avail = avail - 1;
```

```
                        collision_not_found = false;
                        }
                    p+=1;
                }
            }
        }
        return_val = (float)avail/(float)(gridsize*gridsize);
    """
    return weave.inline(code, ['x_grid', 'y_grid', 'x_array', 'y_array',
    ↪ 'r', 'numParticles', 'gridsize'], \
        compiler='gcc')

def computePhi_PBC(x_grid,y_grid,x_array,y_array,r,width):
    """Returns True if the given point at (x,y) lies within any of the
    ↪ particles with
    centers given by (x_array, y_array) with radius "radius."  Otherwise,
↪ returns False.
    """
    from scipy import weave

    numParticles = len(x_array)
    gridsize = len(x_grid)

    code = r"""
        double d2;
        double radius_squared = 4.0*pow(r,2);
        int avail = gridsize*gridsize;
```

```
            bool collision_not_found = true;
            int p = 0;
            double x, y, x_p, y_p;

             for (int i=0; i<gridsize; i++) {
                 x = x_grid[i];
                 for (int j=0; j<gridsize; j++) {
                     y = y_grid[j];
                     p = 0;
                     collision_not_found = true;
                     while ((p<numParticles) && collision_not_found) {
                         // Handle periodic boundary conditions
                         // x sides
                         if ((x < 2*r) && (x_array[p] > width-2*r)) x_p =
↪  x_array[p]-width;
                         else if ((x > width-2*r) && (x_array[p] < 2*r)) x_p
↪  = width+x_array[p];
                         else x_p = x_array[p];
                         // y
                         if ((y < 2*r) && (y_array[p] > width-2*r)) y_p =
↪  y_array[p]-width;
                         else if ((y > width-2*r) && (y_array[p] < 2*r)) y_p
↪  = width+y_array[p];
                         else y_p = y_array[p];

                         if ((pow(x-x_p, 2) + pow(y-y_p,2)) < radius_squared)
↪  {
```

```
                        avail = avail - 1;
                        collision_not_found = false;
                    }
                    p+=1;
                }
            }
        }
        return_val = (float)avail/(float)(gridsize*gridsize);
    """
    return weave.inline(code, ['x_grid', 'y_grid', 'x_array', 'y_array',
    ↪  'r', 'width', 'numParticles', 'gridsize'], \
        compiler='gcc')

def compute_AVF(test_x, test_y, test_h, adsorbed_x, adsorbed_y, r,
↪  num_replicates, random_radius):
    """ Computes the available volume function as a function of distance
    ↪  from the surface.  This is done
    by checking whether a sphere at each of the grid points (test_x,
↪  test_y, test_h) overlaps with
    any of the adsorbed spheres centered at (adsorbed_x, adsorbed_y, r).
↪  The test particles are offset by a
    random vector for each replicate.  If image particles are required for
↪  periodic boundary conditions,
    it is assumed that these are already included in the adsorbed_x and
↪  adsorbed_y arrays.

    Assumes all spheres have radius "r".
```

```
    Arguments:
    test_x          array of x coordinates of test particles
    test_y          array of y coordinates of test particles
    test_h          array of z coordinates of test particles
    adsorbed_x      array of x coordinates of adsorbed particles
    adsorbed_y      array of y coordinates of adsorbed particles
    r               radius of all particles
    num_replicates  number of replicates to perform with different random
↪   offsets at each z level
    random_radius   each test particle will be randomly placed within this
↪   distance of its nominal position

    Returns the available volume fraction for each h value in test_h.
    """
    from compiled import compiled_functions

    return compiled_functions.compute_avf(len(test_x), test_x, test_y,
    ↪  len(test_h), test_h, len(adsorbed_x), \
            adsorbed_x, adsorbed_y, r, num_replicates, random_radius)

# Vector correlation function
def cov(u,v):
    from numpy import mean, sum
    return 1.0 / (len(u) - 1.0) * sum((u - mean(u)) * (v - mean(v)))

def vector_corr(w1, w2):
```

```
    """
    Returns the vector correlation coefficient.
    Arguments:
    w1  vector of x coordinates
    w2  vector of y coordinates

    References:
    A Proposed Definition for Vector Correlation in Geophysics:
        Theory and Application
    Crosby, D. S.; Breaker, L. C.; Gemmill, W. H.
    Journal of Atmospheric and Oceanic Technology, vol. 10, issue 3, p.
↪   355
    1993
    DOI: 10.1175/1520-0426(1993)010<0355:APDFVC>2.0.CO;2

        The Application of a Technique for Vector Correlation to Problems
↪   in
        Meteorology and Oceanography
    Breaker, L. C.; Gemmill, W. H.; Crosby, D. S.
        Journal of Applied Meteorology, vol. 33, Issue 11, pp.1354-1365
        11/1994
    DOI: 10.1175/1520-0450(1994)033<1354:TAOATF>2.0.CO;2
    """
    u1 = w1[:,0]; v1 = w1[:,1]
    u2 = w2[:,0]; v2 = w2[:,1]

    f = cov(u1,u1)*(cov(u2,u2)*cov(v1,v2)**2 + cov(v2,v2)*cov(v1,u2)**2) \
```

```
            + cov(v1,v1)*(cov(u2,u2)*cov(u1,v2)**2 \
            + cov(v2,v2)*cov(u1,u2)**2) \
            + 2*(cov(u1,v1)*cov(u1,v2)*cov(v1,u2)*cov(u2,v2)) \
            + 2*(cov(u1,v1)*cov(u1,u2)*cov(v1,v2)*cov(u2,v2)) \
            - 2*(cov(u1,u1)*cov(v1,u2)*cov(v1,v2)*cov(u2,v2)) \
            - 2*(cov(v1,v1)*cov(u1,u2)*cov(u1,v2)*cov(u2,v2)) \
            - 2*(cov(u2,u2)*cov(u1,v1)*cov(u1,v2)*cov(v1,v2)) \
            - 2*(cov(v2,v2)*cov(u1,v1)*cov(u1,u2)*cov(v1,u2))

    g = (cov(u1,u1)*cov(v1,v1) \
            - cov(u1,v1)**2)*(cov(u2,u2)*cov(v2,v2)-cov(u2,v2)**2)

    return f/g


"""
Code after this point written by
Kevin Whitham, Cornell University
20150715
License: GNU Public License (GPL) v.3.0
functions for generating pair distribution functions
for disordered 2D lattices
"""

import numpy as np
import matplotlib.pyplot as plt
from matplotlib.collections import PatchCollection
```

```
from matplotlib.patches import Circle

def generate_test_grid(shape, period, circle_radius, radius_std,
↪  lattice_std, intensity_std ):

    x = np.linspace(circle_radius, shape[1]-circle_radius,
    ↪  (shape[1]-2*circle_radius)/period)
    y = np.linspace(circle_radius, shape[0]-circle_radius,
    ↪  (shape[0]-2*circle_radius)/period)

    grid_x, grid_y = np.meshgrid(x,y)

    grid_x = grid_x.reshape((-1,1))
    grid_y = grid_y.reshape((-1,1))

    if lattice_std > 0:
        np.add(grid_x,period*np.random.normal(loc=0, scale=lattice_std,
        ↪  size=grid_x.shape),out=grid_x)
        np.add(grid_y,period*np.random.normal(loc=0, scale=lattice_std,
        ↪  size=grid_y.shape),out=grid_y)

    cell_patches = []

    for point_index in range(len(grid_x)):

        ↪  cell_patches.append(Circle((grid_x[point_index],grid_y[point_index]),radius=
        ↪  scale=intensity_std)))
```

```
    pc = PatchCollection(cell_patches,match_original=True)

    plt.figure(1)
    plt.gca().add_collection(pc)

    # set the limits for the plot
    # set the x axis range
    plt.gca().set_xlim(0, shape[1])

    # set the y-axis range and flip the y-axis
    plt.gca().set_ylim(0, shape[0])

    # save this plot to a file
    plt.gca().set_axis_off()

    ↪ plt.savefig('../test_data/input/test_image_10p_contrast.png',bbox_inches='tight'

def radius_offset(radius_std):
    if radius_std > 0:
        return np.random.normal(loc=1, scale=radius_std)
    else:
        return 1

def generate_disordered_rdf(a, b, std, max_distance, resolution):
```

```
g = np.zeros(max_distance/resolution)
r = np.linspace(0,max_distance,max_distance/resolution)

# assumes a, b are primitive lattice vectors
density = 1.0/(a[0]*b[1] - b[0]*a[1])

a_mag = np.sqrt(a[0]**2 + a[1]**2)
b_mag = np.sqrt(b[0]**2 + b[1]**2)
max_peak_num = int(np.ceil(max_distance/a_mag)+1)

dist = []
width = std

for u in range(-max_peak_num, max_peak_num):
    for v in range(-max_peak_num, max_peak_num):
        peak_r = np.sqrt((u*a[0] + v*b[0])**2 + (u*a[1] + v*b[1])**2)
        if peak_r > 0:

            ↪  np.add(g[1::],(np.sqrt(2.0*np.pi)*width)**(-1)*np.exp(-(r[1::]-peak_

# normalize by density and circumference
circumference = 2.0*np.pi*r
g[1::] = g[1::]/(circumference[1::]*density)

return r,g
```

```
def generate_paracrystal_rdf(a, b, std, max_distance, resolution):

    g = np.zeros(max_distance/resolution)
    r = np.linspace(0,max_distance,max_distance/resolution)

    # assumes a, b are primitive lattice vectors
    # find the density of lattice points from the unit cell area
    # by taking the cross-product a x b
    density = 1.0/(a[0]*b[1] - b[0]*a[1])

    a_mag = np.sqrt(a[0]**2 + a[1]**2)
    b_mag = np.sqrt(b[0]**2 + b[1]**2)
    max_peak_num = int(np.ceil(max_distance/a_mag)+1)

    width = 0

    for u in range(-max_peak_num, max_peak_num):
        for v in range(-max_peak_num, max_peak_num):
            peak_r = np.sqrt((u*a[0] + v*b[0])**2 + (u*a[1] + v*b[1])**2)
            if peak_r > 0:
                width = std * np.sqrt(np.sqrt((peak_r)**2 / ( a_mag * b_mag
                ↪  ) ))

                ↪  np.add(g[1::],(np.sqrt(2.0*np.pi)*width)**(-1)*np.exp(-(r[1::]-peak_

    # Use this code to plot the individual pair distribution functions for
    ↪  each lattice spacing in dist
```

```
    # plt.figure(1)
    # unique_dist,unique_count =
    ↪  np.unique(dist,return_index=False,return_inverse=False,return_counts=True)
    #
    # for peak,count in zip(unique_dist[1::],unique_count[1::]):
    #     peak_r = peak*period
    #     width = np.sqrt(peak)*std
    #
    ↪  plt.plot(r,count*(2.0*np.pi*width)**(-1)*np.exp(-(r-peak_r)**2/(2*(width)**2))
    # plt.xlabel('r')
    # plt.ylabel('£g(r_k)£')
    # plt.show()

    # normalize by density and circumference
    circumference = 2.0*np.pi*r
    g[1::] = g[1::]/(circumference[1::]*density)

    return r,g

# assume a,b lattice vectors are equal length
def fit_hex_paracrystal_rdf(r_data, std, r0):
    r,g = generate_paracrystal_rdf((r0,0.0), (-0.5*r0, r0 *
    ↪  np.sqrt(3.0)/2.0), std, r_data[-1]+(r_data[1]-r_data[0]),
    ↪  r_data[1]-r_data[0])
    return g

def fit_square_paracrystal_rdf(r_data, std, r0):
```

```
    r,g = generate_paracrystal_rdf((r0,0.0), (0.0, r0), std,
    ↪  r_data[-1]+(r_data[1]-r_data[0]), r_data[1]-r_data[0])
    return g

def fit_hex_disordered_rdf(r_data, std, r0):
    r,g = generate_disordered_rdf((r0,0.0), (-0.5*r0, r0 *
    ↪  np.sqrt(3.0)/2.0), std, r_data[-1]+(r_data[1]-r_data[0]),
    ↪  r_data[1]-r_data[0])
    return g

def fit_square_disordered_rdf(r_data, std, r0):
    r,g = generate_disordered_rdf((r0,0.0), (0.0, r0), std,
    ↪  r_data[-1]+(r_data[1]-r_data[0]), r_data[1]-r_data[0])
    return g

#generate_test_grid((1032,1376), period=20.0, circle_radius=10.0,
↪  radius_std=0.0, lattice_std=0.0, intensity_std=0.1)

# from scipy.optimize import curve_fit
#rh,gh = generate_paracrystal_rdf(a=(1.0,0.0),
↪  b=(-0.5*1.0,1.0*np.sqrt(3.0)/2.0), std=0.05, max_distance=50.0,
↪  resolution=0.01)
# r1,g1 = generate_paracrystal_rdf(a=(1.0,0.0), b=(0.0,1.0), std=0.05,
↪  max_distance=20.0, resolution=0.01)
# r2,g2 = generate_paracrystal_rdf(a=(1.0,0.0), b=(0.0,1.0), std=0.05,
↪  max_distance=20.0, resolution=0.01)
```

```
#r3,g3 = generate_paracrystal_rdf(a=(1.0,0.0), b=(0.0,1.0), std=0.05,
↪  max_distance=20.0, resolution=0.01)
# plt.plot(r1, g1, 'k-')
# plt.plot(r2, g2, 'b-')
#plt.plot(r3, g3, 'r-')
# plt.gca().legend()
#plt.show()

# g = g + 0.2*np.random.normal(size=len(r))
# popt, pcov = curve_fit(fit_paracrystal, r, g, p0=0.5)
# print(popt)
# print(pcov)
```